UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA QUÍMICA
CURSO DE GRADUAÇÃO EM ENGENHARIA QUÍMICA

PÂMELA OLIVEIRA DE CARVALHO

A UTILIZAÇÃO DE PYTHON PARA A RESOLUÇÃO DE PROBLEMAS APLICADOS À ENGENHARIA QUÍMICA: IMPLEMENTAÇÃO NO ENSINO DE CINÉTICA DE PROCESSOS FERMENTATIVOS

FORTALEZA
2025

PÂMELA OLIVEIRA DE CARVALHO

A UTILIZAÇÃO DE PYTHON PARA A RESOLUÇÃO DE PROBLEMAS APLICADOS À ENGENHARIA QUÍMICA: IMPLEMENTAÇÃO NO ENSINO DE CINÉTICA DE PROCESSOS FERMENTATIVOS

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

Orientadora: Profª. Drª. Luciana Rocha Barros Gonçalves.

FORTALEZA

2025

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

C327u Carvalho, Pâmela Oliveira de.
A utilização de Python para a resolução de problemas aplicados à Engenharia Química : implementação no ensino de cinética de processos fermentativos / Pâmela Oliveira de Carvalho. – 2025.
51 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Tecnologia, Curso de Engenharia Química, Fortaleza, 2025.
Orientação: Profa. Dra. Luciana Rocha Barros Gonçalves.

1. Engenharia Química. 2. Python. 3. Ensino Tecnológico. I. Título.

CDD 660

PÂMELA OLIVEIRA DE CARVALHO

A UTILIZAÇÃO DE PYTHON PARA A RESOLUÇÃO DE PROBLEMAS APLICADOS À ENGENHARIA QUÍMICA: IMPLEMENTAÇÃO NO ENSINO DE CINÉTICA DE PROCESSOS FERMENTATIVOS

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

Aprovada em: ____/____/_____

BANCA EXAMINADORA

_____

Profª. Drª. Luciana Rocha Barros Gonçalves (Orientador)
Universidade Federal do Ceará (UFC)

_____

Prof. Dr. Sebastião Mardônio Pereira de Lucena
Universidade Federal do Ceará (UFC)

_____

Prof. Dr. Ivanildo José da Silva Junior
Universidade Federal do Ceará (UFC)

A Deus.

Aos meus pais, família e amigos.

RESUMO

A utilização de ferramentas computacionais no ensino da Engenharia Química tem se mostrado cada vez mais relevante para a compreensão de fenômenos complexos, além de serem essenciais para a otimização de processos e inovação na área. Desse modo, é fundamental que os cursos de Engenharia Química ofereçam uma grade curricular alinhada às transformações e às inovações tecnológicas, de forma a preparar os alunos para esse novo cenário. O objetivo deste trabalho é analisar como o Python pode ser aplicado no curso de Engenharia Química da Universidade Federal do Ceará (UFC), com foco na implementação de uma aula prática para o ensino de cinética de processos fermentativos da disciplina de Processos e Reatores Bioquímicos. A análise iniciou com uma breve discussão sobre aplicações do Python em conceitos fundamentais do curso, como métodos matemáticos, cinética enzimática e termodinâmica química. Em seguida, o estudo concentrou-se na cinética de processos fermentativos, com o desenvolvimento e a aplicação de uma aula prática que permitiu aos alunos alterarem variáveis e observar seus efeitos no sistema. A metodologia envolveu a realização de uma revisão bibliográfica sobre o uso da linguagem em Engenharia Química, seguida da aplicação da aula e da coleta de percepções dos estudantes por meio de um questionário. Os resultados indicaram que 70,6% dos alunos relataram uma melhora significativa na compreensão dos conceitos de cinética de Monod, e 82,4% concordaram totalmente que a abordagem com Python pode ser útil em outras disciplinas. A experiência demonstrou o potencial da linguagem como ferramenta de ensino, destacando a necessidade de uma integração mais estruturada da programação ao longo da graduação.

Palavras-chave: Engenharia Química; Python; Ensino Tecnológico.

ABSTRACT

The use of computational tools in teaching Chemical Engineering has proven to be increasingly relevant for understanding complex phenomena, in addition to being essential for process optimization and innovation in the field. Therefore, it is essential that Chemical Engineering courses offer a curriculum aligned with technological transformations and innovations, in order to prepare students for this new scenario. The objective of this work is to analyze how Python can be applied in the Chemical Engineering course at the Federal University of Ceará (UFC), focusing on the implementation of a practical class for teaching fermentation process kinetics in the Biochemical Processes and Reactors discipline. The analysis began with a brief discussion on Python applications in fundamental course concepts, such as mathematical methods, enzymatic kinetics, and chemical thermodynamics. The study then focused on the kinetics of fermentation processes, with the development and implementation of a practical class that allowed students to change variables and observe their effects on the system. The methodology involved conducting a bibliographical research on the use of the language in Chemical Engineering, followed by the application of the class and the collection of students' perceptions through a questionnaire. The results indicated that 70.6% of the students reported a significant improvement in their understanding of Monod kinetics concepts, and 82.4% strongly agreed that the Python approach could be useful in other disciplines. The experience demonstrated the potential of the language as a teaching tool, highlighting the need for a more structured integration of programming throughout the undergraduate course.

Keywords: Chemical Engineering; Python; Technological Education.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

A Indústria 4.0 se caracteriza pela implementação de sistemas ciber-físicos, de forma a monitorar e controlar processos de produção, armazenando dados virtuais da realidade e permitindo que decisões sejam tomadas de forma descentralizada. Além disso, esses sistemas estão interconectados através da Internet das Coisas (IoT), o que possibilita a comunicação e cooperação em tempo real entre máquinas e humanos, promovendo uma gestão integrada e dinâmica das operações industriais (HERMANN et al., 2015).

De acordo com Hermann et al. (2015) existem alguns princípios básicos para aplicação de iniciativas da Indústria 4.0, que são: interoperabilidade, virtualização, descentralização, orientação ao serviço, modularidade e capacidade de resposta em tempo real. Esse último é essencial, pois se relaciona com a necessidade de coleta e análise de dados de forma instantânea pelos sistemas, para garantir o entendimento do processo e possibilitar uma tomada de decisão mais rápida e assertiva.

Na Engenharia Química, essa capacidade de resposta em tempo real é de extrema importância para a otimização e controle dos processos industriais. Equipamentos como destiladores e trocadores de calor possuem variáveis complexas e interdependentes e, caso não monitoradas de forma adequada, podem comprometer a qualidade do produto e a segurança do processo. Com a integração da tecnologia na indústria, pode-se coletar dados de sensores em tempo real, analisar as informações de forma instantânea e ajustar parâmetros da operação de forma automática, o que resulta em uma redução de custos e aumento da produtividade do processo. Além disso, a capacidade de detectar falhas precocemente permite ações preventivas e corretivas de forma imediata, minimizando os riscos de acidentes e garantindo uma maior segurança operacional.

Nesse contexto do crescente uso de tecnologias avançadas, é essencial que os futuros engenheiros químicos estejam preparados para lidar com as ferramentas digitais envolvidas na análise e no controle de processos industriais e de gestão. Dentre as diversas linguagens de programação disponíveis, o Python destaca-se como uma das mais relevantes para a Engenharia Química, tanto no âmbito acadêmico quanto industrial. Sua popularidade se deve, em grande parte, à sua sintaxe simples e intuitiva, que facilita o aprendizado e a aplicação por parte de estudantes e profissionais. Além disso, o Python possui uma gama de bibliotecas especializadas, como *NumPy*, *SciPy* e *Matplotlib*, que são amplamente utilizadas para modelagem matemática, análise de dados, simulação de processos e visualização de resultados. Essas características tornam o Python uma ferramenta poderosa para a resolução de problemas

complexos, como a simulação de reatores químicos, a otimização de processos de separação e a análise de dados em tempo real provenientes de sensores industriais.

Outro ponto que diferencia o Python de outras linguagens é a sua capacidade de integração com outras tecnologias, como a Internet das Coisas (IoT) e a Inteligência Artificial (IA), que são pilares da Indústria 4.0. Enquanto linguagens como C++ e Java exigem um conhecimento mais aprofundado de programação e têm uma curva de aprendizado mais acentuada, o Python permite que os usuários desenvolvam soluções robustas com menos linhas de código e em menor tempo. Essa eficiência é valiosa em um contexto industrial, onde a agilidade na tomada de decisões e a capacidade de adaptação a mudanças são fundamentais.

Na Engenharia Química, o Python tem sido amplamente adotado para aplicações como a modelagem de cinética de reações, a simulação de processos de destilação e a análise de dados termodinâmicos. Sua flexibilidade permite que seja utilizado para tarefas simples, como a criação de scripts para automatizar cálculos repetitivos, e para projetos complexos, como a implementação de algoritmos de *machine learning* para previsão de falhas em equipamentos industriais. Em comparação com outras linguagens, como MATLAB ou R, o Python oferece a vantagem de ser *open-source*, o que reduz custos e facilita a colaboração entre os usuários.

A programação em linguagem Python, portanto, destaca-se como uma ferramenta poderosa, sendo amplamente utilizada na Engenharia Química para uma variedade de aplicações. Além disso, sua capacidade de análise de dados permite interpretar grandes volumes de informações provenientes de processos industriais, enquanto suas bibliotecas gráficas facilitam a criação de visualizações detalhadas para suporte à tomada de decisões. Contudo, apesar da relevância do Python no ensino da Engenharia Química e da necessidade de inovação tecnológica no curso de graduação, o ensino de programação no currículo ainda é defasado. A disciplina de Programação Computacional para Engenharia é ministrada no 1° semestre e aborda conceitos básicos de Python, como lógica de programação, criação de variáveis e funções básicas. Após essa disciplina, o Python é abordado de forma pontual em algumas disciplinas do curso, como Fenômenos de Transporte II, Operações Unitárias II e Dinâmica e Controle de Processos. Porém, como os conceitos ensinados na disciplina de programação são básicos, o acompanhamento do Python em disciplinas da Engenharia Química torna-se complicado para os alunos, pois são necessários conceitos mais avançados e, com isso, é necessário que o ensino do Python durante a graduação aconteça de uma forma gradual, intencional e aplicada, para uma melhor consolidação do aprendizado.

2 OBJETIVOS

2.1 Gerais

Esse trabalho tem como objetivo analisar como o Python pode ser aplicado no ensino das disciplinas de Engenharia Química da Universidade Federal do Ceará (UFC), com foco na implementação de uma aula prática para o ensino de cinética de processos fermentativos da disciplina de Processos e Reatores Bioquímicos. O estudo pretende avaliar como o curso de Engenharia Química da UFC pode ser aprimorado, integrando o uso do Python para garantir uma formação mais alinhada às competências tecnológicas exigidas atualmente no mercado de trabalho e aos avanços no ensino de Engenharia Química, avaliando aplicações da linguagem para alguns conteúdos do curso.

2.2 Específicos

- Identificar aplicações de Python para as disciplinas de Métodos Matemáticos e Computacionais, Termodinâmica Química e Reatores e Processos Bioquímicos;
- Realizar uma aula prática aplicando o Python para o ensino de conceitos de cinética de processos fermentativos na disciplina de Reatores e Processos Bioquímicos;
- Analisar a percepção dos alunos sobre a aula prática de Python aplicada à cinética de processos fermentativos.

3 REFERENCIAL TEÓRICO

3.1 A linguagem Python

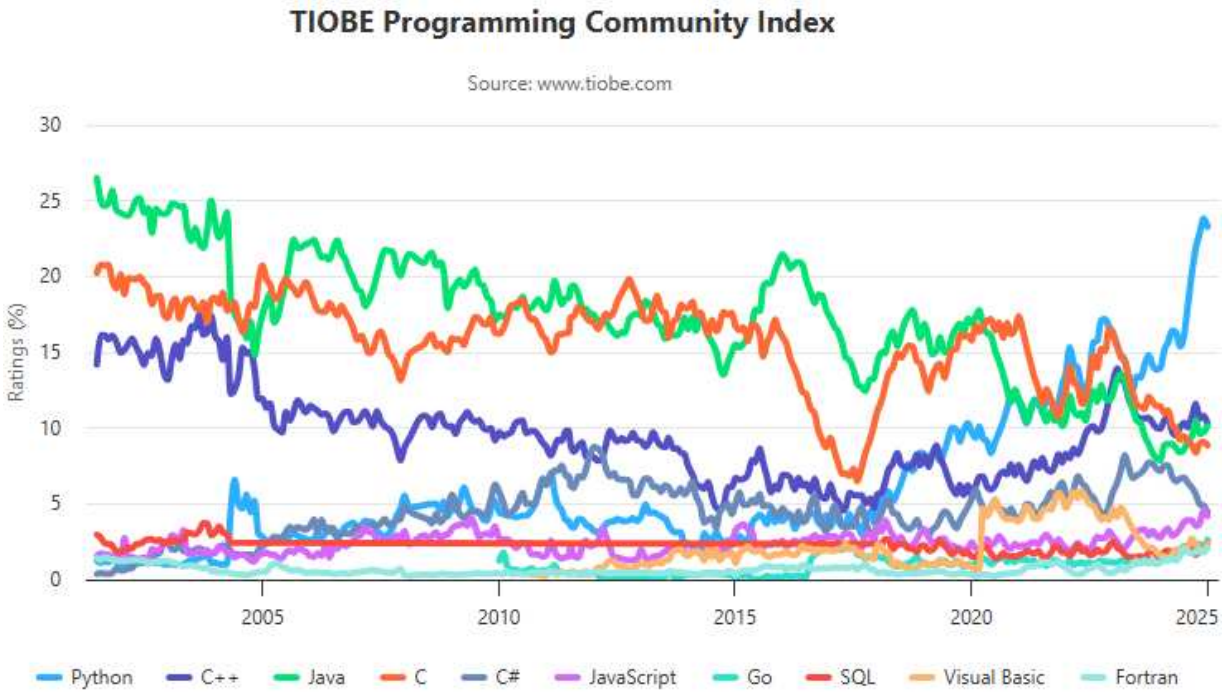
O Python é uma linguagem de programação computacional amplamente utilizada, interpretada, orientada a objeto e de alto nível. Criado por Guido Van Rossum e lançada pela primeira vez no dia 20 de fevereiro de 1991, a linguagem foi desenvolvida com o objetivo de ser simples e fácil de ler, com uma sintaxe que foca na clareza e legibilidade do código. Além disso, o Python é uma linguagem que pode ser utilizada em diferentes sistemas operacionais: Windows, MacOS e Linux (ELHALID et al., 2023)

A linguagem é extremamente versátil, podendo ser utilizada em diversos campos, como o desenvolvimento de softwares, ciência de dados e automação. Outro ponto que representa a versatilidade do Python é o fato de ser uma linguagem interpretada, executando o código linha a linha, facilitando a depuração, que é o processo de localização e ajuste de erros, e o teste interativo.

Além da sua versatilidade, o Python também se destaca como uma linguagem popular e muito utilizada. O índice TIOBE é uma ferramenta amplamente conhecida pela comunidade de desenvolvimento de software, que possui como objetivo a medição da popularidade das linguagens de programação. A TIOBE Software é uma empresa holandesa fundada por Paul Jansen em 2001 e é a responsável pelo índice TIOBE, que coleta dados de diversas fontes, incluindo mecanismos de busca como Google, Bing, Yahoo!, além de informação sobre ofertas de emprego e cursos relacionados a linguagem de programação. Esses dados são processados por um algoritmo para definir a classificação mensal de popularidade das linguagens de programação (KRILL, 2025).

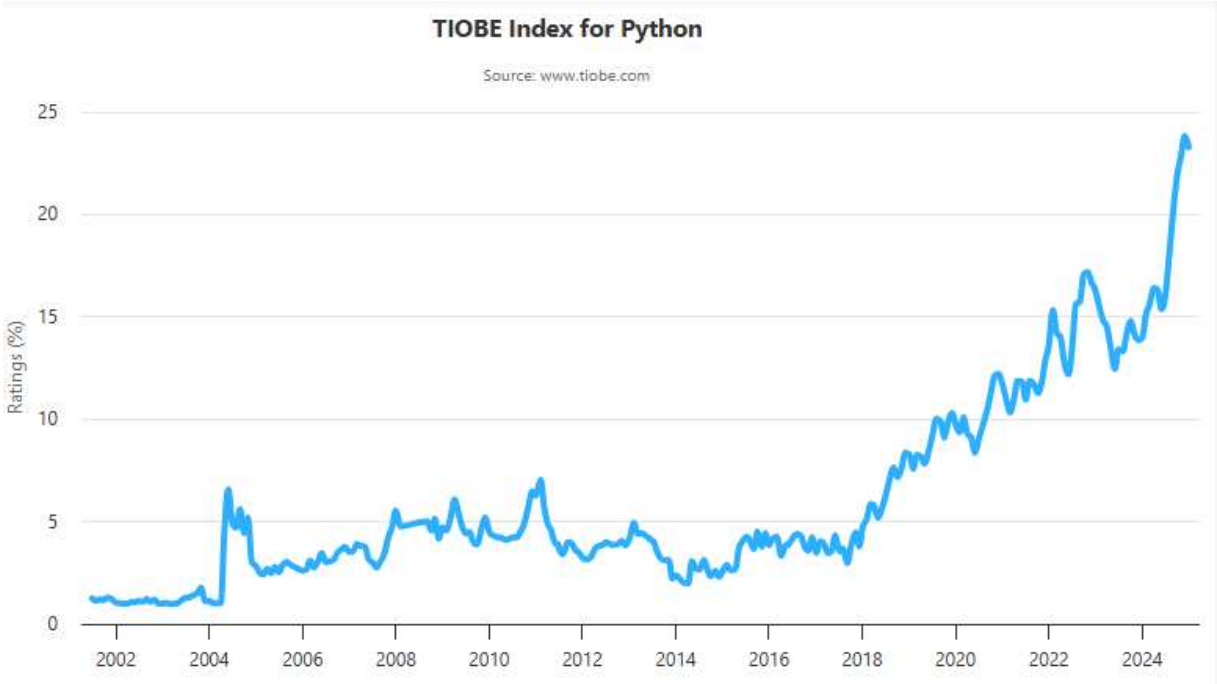
O Python se consolidou como a linguagem de programação mais popular em 2024, de acordo com o TIOBE Index (2025). Na figura 1 é possível observar a popularidade do Python em relação a outras linguagens de programação ao longo dos anos. Além disso, a figura 2 traduz de forma mais clara apenas a evolução do Python, demonstrando o quanto a linguagem vem se popularizando de forma expressiva ao longo dos anos, tendo um aumento de 9,3% em sua popularidade quando comparado os anos de 2023 e 2024. Outras linguagens de programação que tiveram crescimento, como Java e JavaScript, não tiveram um aumento tão expressivo quanto o Python, tendo crescimentos de 2,3% e 1,4%, respectivamente.

Figura 1 - Índice TIOBE ao longo dos anos



Fonte: TIOBE Index (2025)

Figura 2 - Índice TIOBE para o Python ao longo dos anos



Fonte: TIOBE Index (2025)

3.2 *NumPy*

A biblioteca *NumPy*, abreviação para *Numerical Python*, é um pacote fundamental para computação científica em Python. Criada em 2005 por Travis Oliphant, a biblioteca de código aberto possibilita operações em *arrays* multidimensionais, chamadas de *ndarray* na biblioteca, sendo uma abreviação para *N-dimensional array*. Nesse contexto, um *array* é uma estrutura multidimensional que possibilita o armazenamento de dados na memória do computador, no qual cada item dessa estrutura pode ser encontrado por um esquema de indexação, podendo ter de uma a três dimensões (MULINARI, 2024).

O *NumPy* possui grande importância pela sua capacidade de realizar cálculos de alto desempenho, oferecendo uma gama de funções para realização de operações matemáticas, lógicas e estatísticas. Além disso, a biblioteca é bastante otimizada por permitir a realização de operações em *arrays* de forma muito mais rápida e eficiente do que a maneira convencional de estruturas tradicionais do Python, como as listas (BIGELOW, 2024).

3.3 *SciPy*

O *SciPy* é uma biblioteca de código aberto, construída sobre a *NumPy*, amplamente utilizada por matemáticos e engenheiros, por ser bastante útil para resolução de problemas que necessitam de precisão e eficiência (VIRTANEN et al., 2020).

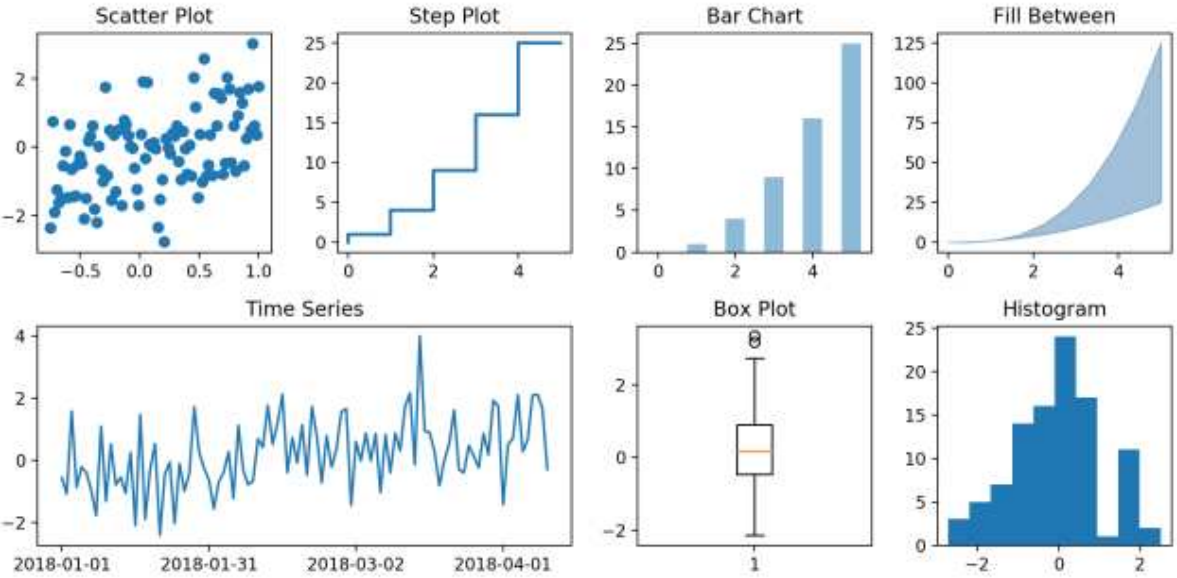
A principal diferença entre o *NumPy* e o *SciPy* é que, enquanto o primeiro foca em criação e manipulação de *arrays* multidimensionais e operações matemáticas mais simples, como adição, soma e álgebra linear simples, o segundo possibilita o uso de funções mais avançadas. Com o *SciPy*, é possível a construção algoritmos para otimização, integração numérica, interpolação de dados e resolução de equações algébricas, diferenciais e estatísticas, demonstrando a sua versatilidade e aplicabilidade em problemas de engenharia (SCIPY.ORG, 2025).

3.4 *Matplotlib*

Na ciência de dados, a visualização das informações é um ponto crucial para análise e interpretação dos dados. Nesse contexto, o *Matplotlib* é uma biblioteca Python de código aberto criada em 2003 por John D. Hunter, que oferece recursos de plotagem para a criação de gráficos 2D e 3D, além de permitir visualizações estáticas, animadas e interativas (HUNTER, 2008).

Além disso, a biblioteca pode criar diferentes tipos de relatórios de visualização, como histogramas, gráficos de linhas, gráficos de dispersão, gráficos de barras, gráficos de pizza e diagrama de caixa, sendo útil em diversas aplicações e facilitando a análise visual de grandes volumes de dados (HUNTER, 2008).

Figura 3- Exemplo de gráficos plotados com *Matplotlib*



Fonte: Machine Learning Plus

3.5 *Py-pde*

O *Py-pde*, abreviação para *Python partial differential equations*, é uma biblioteca em Python criada para resolver equações diferenciais parciais (EDPs) de uma forma eficiente, utilizando, para isso, o método das diferenças finitas (ZWICKER, 2020). A biblioteca permite a simulação de análise de EDPs da forma:

$$\partial t u(x,t) = D[u(x,t)] + \eta(u,x,t) \quad (4)$$

Da equação acima, o termo "D" é um operador diferencial não linear, que descreve como um ou mais campos físicos "u" evoluem ao longo do tempo. Tais campos podem ser escalares ou tensores e dependem das coordenadas espaciais "x" e do tempo "t". Além disso, o *Py-pde* também oferece suporte para equações diferenciais estocásticas na representação de Itô, indicada pelo termo de ruído "η" na equação acima (ZWICKER, 2020).

3.6 *Coolprop*

A biblioteca *Coolprop* é uma biblioteca de código aberto, desenvolvida por Ian Bell, sendo bastante utilizada para consulta de propriedades termodinâmicas e no fornecimento de equações de estado para uma ampla variedade de fluidos. A biblioteca também é utilizada para a resolução de equações cúbicas de estado, a citar as equações de Peng-Robinson e Soave-Redlich-Kwong, sendo bastante útil a sua aplicação para cálculos de flash com entradas do tipo TP (temperatura e pressão), PQ (pressão e qualidade), DT (diferença de temperatura) e QT (qualidade e temperatura), além do cálculo dos pontos críticos de misturas (BELL, 2014).

3.7 *Thermo*

A *Thermo* é uma biblioteca bastante relevante para Termodinâmica Química, pois permite o cálculo de propriedades de substâncias puras e misturas, além de possibilitar a utilização de equações de estado cúbicas, como Peng-Robinson, Soave-Redlich-Kwong e Van der Waals, e modelos de coeficientes de atividade, como UNIFAC e UNIQUAC. A biblioteca pode ser também aplicada em simulações de equilíbrio de fases e cálculos de flash, permitindo análises em diferentes valores de temperatura e pressão (BELL, 2016-2024).

3.8 A importância do Python para a Engenharia Química

A Engenharia Química é uma área que demanda a aplicação de ferramentas computacionais para a resolução de problemas complexos, como a modelagem de processos, a simulação de reatores químicos, a otimização de operações unitárias e a análise de grandes volumes de dados provenientes de processos industriais. Nesse contexto, o Python se destaca como uma linguagem de programação essencial para a formação e atuação do engenheiro químico, oferecendo uma série de vantagens que a tornam ideal para aplicações na área.

Uma das principais aplicações do Python na Engenharia Química é a modelagem de sistemas químicos e bioquímicos. A sua capacidade de criar modelos computacionais que representam fenômenos físicos e químicos complexos, como a cinética de reações, o equilíbrio de fases e a transferência de calor e massa, é fundamental para a simulação de processos industriais. Esses modelos permitem que os engenheiros químicos simulem cenários, prevejam resultados e tomem decisões mais assertivas, reduzindo custos e aumentando a eficiência dos processos. Bibliotecas como *NumPy* e *SciPy* são amplamente utilizadas para a implementação de modelos matemáticos e a resolução de equações diferenciais que descrevem esses fenômenos

(SCIPY.ORG, 2025).

Além da modelagem, o Python é amplamente utilizado para a análise de dados em Engenharia Química. Com o contexto de análise de *big data*, a capacidade de processar grandes volumes de dados tornou-se essencial para a otimização de processos industriais. Bibliotecas como a *Matplotlib* permitem a manipulação e visualização de dados de forma eficiente, facilitando a identificação de padrões e tendências que podem ser utilizados para melhorar a eficiência de processos químicos. Um exemplo a ser citado é a análise de dados provenientes de sensores instalados em equipamentos como reatores e colunas de destilação que podem fornecer observações valiosas sobre o desempenho do processo e ajudar na detecção precoce de falhas.

Outra aplicação importante do Python na Engenharia Química é a simulação de processos, que permite que os engenheiros químicos testem diferentes cenários e avaliem o impacto de mudanças nas condições de operação sem a necessidade de realizar experimentos físicos, o que pode ser um processo caro e demorado. O *Py-pde* se demonstra como uma ferramenta útil para a simulação de equações diferenciais parciais, presente em problemas de transferência de calor e massa (ZWICKER, 2020). Além disso, a simulação de processos bioquímicos, como a fermentação, pode ser realizada com o uso de modelos matemáticos implementados em Python, permitindo a otimização de parâmetros como temperatura, pH e concentração de substrato.

A otimização de processos é outra área em que o Python se destaca. A capacidade de resolver problemas de otimização complexos, como a minimização de custos ou a maximização da eficiência de processos, é essencial para a Engenharia Química. Nesse contexto, o *SciPy* oferece ferramentas para a implementação de algoritmos de otimização, como o método dos mínimos quadrados e a programação linear, que podem ser aplicados para melhorar o desempenho de processos industriais. Pode-se utilizar, por exemplo, a biblioteca para a otimização de colunas de destilação com o uso de modelos matemáticos permitindo a identificação das condições operacionais ideais para maximizar a pureza do produto e minimizar o consumo de energia, diminuindo os custos operacionais.

A flexibilidade e a facilidade de aprendizado do Python são fatores que também contribuem para sua utilização no ensino de Engenharia Química. A linguagem permite que os estudantes comecem a desenvolver soluções computacionais desde o início da graduação, aplicando conceitos teóricos em problemas práticos. Essa integração entre teoria e prática é fundamental para a formação de profissionais capacitados a lidar com os desafios da indústria moderna. Além disso, a grande quantidade de bibliotecas disponíveis, como *Coolprop* e *Thermo*,

facilita a realização de cálculos termodinâmicos e a análise de propriedades de substâncias puras e misturas, o que é essencial para a prática da Engenharia Química (BELL, 2016-2024).

Além disso, o ensino de lógica de programação é fundamental para o desenvolvimento de uma mentalidade analítica e estruturada, essencial para a resolução de problemas complexos em áreas como a Engenharia Química. Aprender a programar envolve a capacidade de decompor problemas em partes menores, identificar padrões e criar soluções sistemáticas e eficientes. Essa habilidade é especialmente relevante para engenheiros químicos, que lidam com sistemas complexos, como a modelagem de reatores, a otimização de processos e a análise de dados. A lógica de programação permite abordar esses desafios com clareza e eficiência, identificando causas raízes do problema e desenvolvendo soluções eficazes.

Por fim, a lógica de programação estimula o pensamento crítico e a criatividade, incentivando a exploração de diferentes abordagens para resolver problemas. Essa capacidade é crucial para a inovação e a otimização de processos na Engenharia Química, onde a busca por soluções eficazes é constante. Ao integrar a programação na formação do engenheiro químico, prepara-se o profissional para enfrentar os desafios da indústria com uma visão estruturada e inovadora.

4 METODOLOGIA

A metodologia do trabalho foi desenvolvida em três etapas principais: revisão bibliográfica, elaboração e preparação de uma aula prática, e análise dos resultados obtidos relacionados à percepção dos alunos com a aula ministrada.

Inicialmente, realizou-se uma revisão bibliográfica utilizando o Google Acadêmico para a busca de artigos científicos, dissertações e periódicos de Engenharia Química que apresentassem códigos em Python aplicados ao curso. A pesquisa focou nas disciplinas Métodos Matemáticos e Computacionais aplicados à Engenharia Química, Termodinâmica Química e Processos e Reatores Bioquímicos. Para a realização da revisão bibliográfica, utilizaram-se palavras-chave combinando o nome das disciplinas escolhidas e alguns conteúdos específicos com a linguagem Python, tanto em português quanto em inglês. Alguns exemplos dos termos empregados foram "Métodos Matemáticos Python", "Termodinâmica Python", "*Biochemical Reactors Python*" e "*Differential equations in Python*". Para esse estudo, priorizou-se a seleção de pesquisas recentes, de forma a garantir a atualidade do material e compatibilidade com as versões mais recentes do Python.

A segunda etapa consistiu na elaboração e aplicação de uma aula prática sobre cinética de processos fermentativos, destinada a alunos da disciplina de Processos e Reatores Bioquímicos. A aula foi realizada no laboratório de informática, com duração total de 1 hora. Para facilitar o acesso e a execução do código por todos os alunos, optou-se pelo uso do *Google Colab,* uma plataforma online que permite a execução de códigos em Python sem a necessidade de instalação local. A aula foi dividida em três partes: nos primeiros 20 minutos, foram revisados os conceitos fundamentais de cinética de processos fermentativos, com ênfase nos modelos matemáticos aplicados a reatores bioquímicos e na cinética de Monod. Em seguida, foram dedicados 10 minutos para a explicação do código em Python adaptado de Santos (2024), que foi desenvolvido utilizando as bibliotecas *NumPy*, *SciPy* e *Matplotlib*. A estrutura do código foi explicada de forma simplificada, com foco nos parâmetros que poderiam ser alterados pelos alunos.

Para a elaboração do código Python, foram utilizadas as bibliotecas *NumPy*, para cálculos numéricos e manipulação de *arrays*, *SciPy*, para resolução de equações diferenciais, essenciais para modelar a cinética de processos fermentativos, e *Matplotlib*, para visualização gráfica dos resultados, permitindo a plotagem de gráficos de concentração de substrato, células e produto ao longo do tempo.

Em seguida, para a parte prática da aula, que durou 30 minutos, os alunos foram orientados a abrir o código no Google Colab e alterar parâmetros específicos, como Ks (constante de saturação), μmax (taxa máxima de crescimento microbiano) e as constantes α e β de Ludwig Piret (relacionadas à cinética de fermentação). Após a execução do código, os alunos observaram as mudanças nos gráficos gerados e os resultados foram discutidos em grupo, relacionando as alterações nos parâmetros com os conceitos teóricos abordados, garantindo uma maior consolidação do conhecimento. Participaram da aula 17 alunos, com conhecimentos prévios equilibrados entre básico e intermediário em Python. Em relação à cinética de processos fermentativos, os alunos já possuíam uma base teórica, o que permitiu um maior enfoque na parte prática com a linguagem de programação.

Ao final da aula, foi aplicada uma pesquisa de satisfação e aprendizado por meio do Google Forms. A pesquisa foi respondida por 100% dos alunos presentes e continha perguntas fechadas obrigatórias e uma pergunta aberta opcional. As perguntas fechadas focaram em entender o nível de compreensão dos conceitos após a aula e a utilidade do código Python no aprendizado. A pergunta aberta permitiu que os alunos pudessem compartilhar sugestões e dificuldades encontradas durante a atividade. As respostas foram analisadas quantitativamente, por meio de análise gráfica, e qualitativamente, a partir da pergunta aberta de percepções sobre a aula.

Em relação às limitações do estudo, destaca-se o número reduzido de participantes na aula prática (17 alunos), o que pode limitar a análise dos resultados. Além disso, o tempo disponível para a explicação do código foi curto (10 minutos), o que, considerando a presença de alunos com nível básico de Python, pode ter dificultado o entendimento completo da ferramenta e sua aplicação nos conceitos de cinética de processos fermentativos. Nesse contexto, sugere-se que o estudo seja realizado novamente ao longo do semestre na disciplina de Processos e Reatores Bioquímicos, aplicando o Python em diferentes conteúdos, como modelagem de reatores ideais, cinética enzimática, balanços de massa em sistemas fermentativos e análise de dados experimentais. Essa abordagem permitiria uma construção gradual do conhecimento em Python pelos alunos, além de proporcionar uma análise mais abrangente e consistente do impacto da ferramenta no aprendizado. Ao integrar o Python de forma contínua ao longo da disciplina, seria possível coletar percepções mais detalhadas e representativas dos alunos, avaliando como a linguagem contribui para o entendimento dos conceitos e para a formação de habilidades computacionais essenciais na Engenharia Química.

5 RESULTADOS E DISCUSSÃO

5.1 Métodos Matemáticos e Computacionais aplicados à Engenharia Química

A disciplina de métodos matemáticos e computacionais aplicados à Engenharia Química é ministrada no curso de Engenharia Química da Universidade Federal do Ceará durante o 3° período. Na ementa da disciplina, estão conteúdos como equações diferenciais ordinárias, séries de potências, soluções de equações diferenciais ordinárias por série de potências, sistemas de equações diferenciais e equações diferenciais parciais.

Em relação aos conteúdos abordados, existem diversos exemplos que podem ser calculados de forma analítica, utilizando o método de separação de variáveis e o método do fator integrante. Um exemplo são as equações diferenciais ordinárias (EDOs) de primeira ordem lineares homogêneas, que podem ser resolvidas utilizando o método do fator integrante (FONTANA, 2019, p. 45). Apesar de serem possíveis de solucionar de forma analítica, a utilização de Python no entendimento de EDOs é bastante válida. Além disso, o Python também se destaca como uma ferramenta eficiente para resolver equações diferenciais parciais (EDPs) de forma numérica, facilitando a resolução de equações complexas que não podem ser resolvidas por métodos analíticos simples.

5.1.1 Resolução de equações diferenciais ordinárias

Com a utilização das bibliotecas *NumPy, SciPy e matplotlib*, é possível implementar um código para a resolução de EDOs. A biblioteca *NumPy* é útil para realizar operações matemáticas e manipulação de *arrays*, estruturas que permitem o armazenamento e organização dos dados, de forma eficiente, enquanto a *SciPy* oferece funções específicas para a resolução de equações diferenciais, como métodos numéricos para integração. Já a *matplotlib* permite a visualização dos resultados por meio de gráficos, facilitando a análise e interpretação das soluções (ALVES; OLIVEIRA, 2021).

Para exemplificação, será analisado um código em Python para resolução da equação 1, utilizando as equações 2 e 3 como auxiliares para solucionar o problema. Essa equação pode ser aplicada para reatores batelada com volume constante, considerando processos com cinética de primeira ordem. Essa modelagem pode representar, por exemplo, a decomposição de um reagente químico, no qual k representa a constante de velocidade e $y(t)$ a concentração do reagente ao longo do tempo.

$$\frac{dy(t)}{dt} = -k * y(t) \quad (1)$$

$$k = 0{,}3 \quad (2)$$

$$y(0) = 5 \quad (3)$$

O código apresentado na figura 4 inicialmente importa as bibliotecas necessárias para o código e, em seguida, define a equação diferencial e delimita sua condição inicial. Em seguida, a solução é obtida utilizando a função *odeint* da biblioteca *scipy.integrate* para integrar numericamente a equação diferencial, proporcionando uma solução aproximada ao longo do tempo.

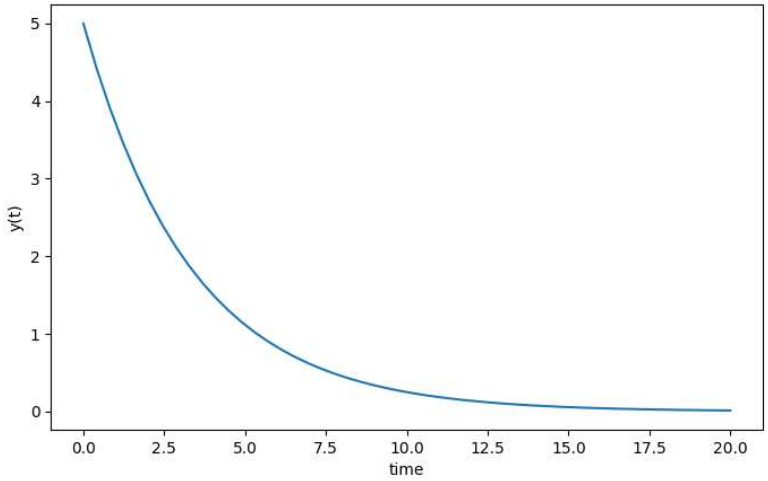
Figura 4 - Código para resolução de EDO utilizando Scipy

```
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt

# function that returns dy/dt
def model(y,t):
    k = 0.3
    dydt = -k * y
    return dydt

# initial condition
y0 = 5

# time points
t = np.linspace(0,20)

# solve ODE
y = odeint(model,y0,t)

# plot results
plt.plot(t,y)
plt.xlabel('time')
plt.ylabel('y(t)')
plt.show()
```

Fonte: Solve Differential Equations with ODEINT

A Figura 5 apresenta a resolução da EDO, ilustrando o comportamento dos resultados ao longo do tempo. Com o código implementado, é possível alterar o valor da constante e o valor da condição inicial para o entendimento de como esses parâmetros afetam a equação. Isso demonstra como o Python se torna uma ferramenta poderosa para a análise e compreensão do comportamento de EDOs, facilitando o estudo de diferentes cenários e parâmetros.

Figura 5 - Concentração de reagente ao longo do tempo



Fonte: Solve Differential Equations with ODEINT

5.1.2 Resolução de equações diferenciais parciais

Existem algumas equações mais complexas que as EDOs que não são possíveis de serem resolvidas de forma analítica, exigindo métodos numéricos para obtenção de soluções aproximadas. Um exemplo são as equações diferenciais parciais (EDPs), bastante utilizadas na modelagem de fenômenos como transferência de calor e difusão. Nesse caso, o Python se destaca como uma excelente ferramenta para o aprendizado desses conteúdos, permitindo implementar métodos numéricos eficientes e visualizações claras e intuitivas dos resultados.

Pode-se utilizar a biblioteca *Py-pde* no ensino de Métodos Matemáticos na Engenharia Química para a resolução de uma EDP cujas condições de contorno são dependentes do tempo. Um exemplo é a equação representada abaixo, junto com as suas condições de contorno:

$$\frac{\partial C}{\partial t} = \nabla^2 C \quad (5)$$

$$C(0,t) = C(10,t) = sin(t) \quad (6)$$

A equação apresentada pode ser utilizada para modelar fenômenos da Engenharia Química, como a difusão de massa. No entanto, para que esse fenômeno seja modelado corretamente, é necessário incluir o coeficiente de difusividade do soluto no meio na equação 5, ajustando-a para representar a Segunda Lei de Fick, como pode ser observado na equação 7 (GAMA, 2022).

$$\frac{\partial C}{\partial t} = \mathrm{D} * \nabla^2 C \ (7)$$

Para resolução do problema, utiliza-se a biblioteca *Py-pde* de acordo com o código da figura 6, que resolve uma equação de Laplace, definindo a condição de contorno. Por fim, um gráfico de espaço-tempo é plotado por meio da função *plot_kymograph*.

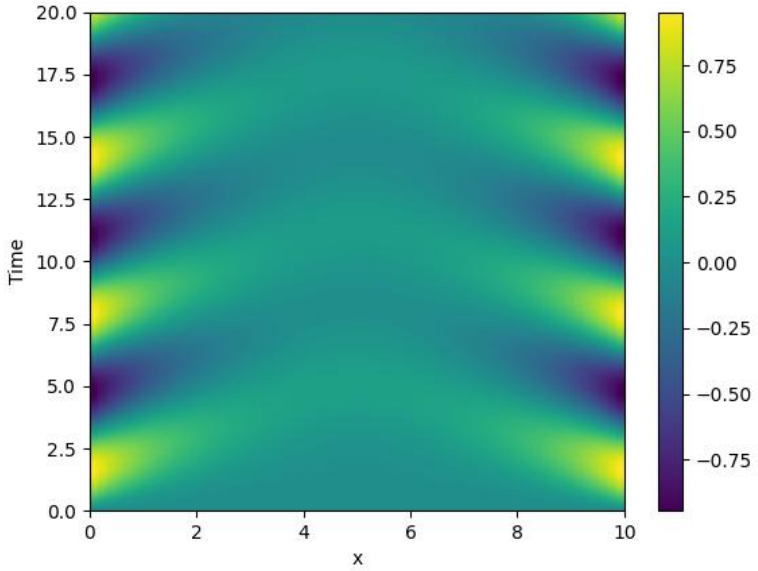
Figura 6 - Resolução de EDP utilizando *Py-pde*

```
from pde import PDE, CartesianGrid, MemoryStorage, ScalarField, plot_kymograph

grid = CartesianGrid([[0, 10]], [64])  # generate grid
state = ScalarField(grid)  # generate initial condition

eq = PDE({"c": "laplace(c)"}, bc={"value_expression": "sin(t)"})

storage = MemoryStorage()
eq.solve(state, t_range=20, dt=1e-4, tracker=storage.tracker(0.1))

# plot the trajectory as a space-time plot
plot_kymograph(storage)
```

Fonte*: Py-Pde*

Os resultados encontrados pela resolução da EDP foram plotados em um gráfico espaço-tempo, como pode ser observado na figura 7.

Figura 7 - Concentração em função do espaço e do tempo



Fonte: *Py-Pde*

Uma grande vantagem de utilizar o Python no ensino de Métodos Matemáticos e Computacionais é a possibilidade de os alunos conseguiremos visualizar os resultados obtidos das simulações em tempo real, o que permite uma compreensão mais clara do comportamento das equações diferenciais, já que é possível alterar parâmetros como coeficientes, condições de contorno, intervalo da discretização e observar imediatamente como essas mudanças afetam a resolução do problema. Essa abordagem interativa facilita tanto o entendimento teórico da disciplina, quanto estimula a curiosidade e o pensamento crítico, incentivando os alunos a explorarem equações mais complexas e a relacionarem a teoria observada no curso com aplicações práticas da Engenharia Química.

5. 2 Termodinâmica Química

A disciplina de termodinâmica química no curso de Engenharia Química da UFC é ministrada no 5° e 6° períodos. Dentre os conteúdos abordados no componente curricular estão produção de potência a partir de calor; refrigeração e liquefação; relações termodinâmicas; equações de estado; equilíbrio de fases em um fluido puro; introdução aos sistemas multicomponentes; equilíbrio de fases em misturas por uma equação de estado; modelos de atividade; tópicos em equilíbrio de fases e sistemas reagentes.

5.2.1 Ciclo termodinâmico

O *CoolProp* é uma biblioteca disponível para Python muito utilizada para o cálculo de propriedades físicas de substâncias puras e misturas, como pressão, temperatura, entalpia, entropia e densidade, sendo uma ótima ferramenta para o ensino de Termodinâmica Química.

A biblioteca pode ser bastante útil para plotar diagramas pressão-entalpia, que são essenciais no estudo e aplicação de sistemas de refrigeração e ciclos termodinâmicos. Com o uso da biblioteca *CoolProp*, é possível calcular as propriedades termodinâmicas de todo o ciclo: compressão, condensação, expansão e evaporação. Além disso, é possível plotar gráficos desses processos, facilitando a visualização e o entendimento dos conceitos, permitindo uma análise clara das transformações de estado e do comportamento dos sistemas, sendo útil para o ensino de Termodinâmica Química (BELL, 2014).

Um exemplo prático de como o Python pode ser utilizado para a visualização de ciclos termodinâmicos pode ser encontrado na documentação da biblioteca *CoolProp*, desenvolvida por Ian H. Bell, apresentado na figura 8, que exemplifica uma simulação de um ciclo de

compressão simples para um fluido refrigerante, no caso foi utilizado para o R134a, e exibe um gráfico pressão-entalpia.

Detalhando um pouco mais sobre o código, ele inicialmente importa a biblioteca principal *CoolProp* e, em seguida, importa as bibliotecas necessárias para a simulação do ciclo que são: *PropertyPlot* e *SimpleCompressionCycle*. A primeira é utilizada para a criação de gráficos das propriedades dos fluidos, demonstrando as isolinhas do processo, permitindo a análise visual das mudanças de estado do ciclo. Já a *SimpleCompressionCycle* é importada para realizar a simulação do ciclo, levando em consideração as pressões e temperaturas nos pontos de entrada e de saída do compressor.

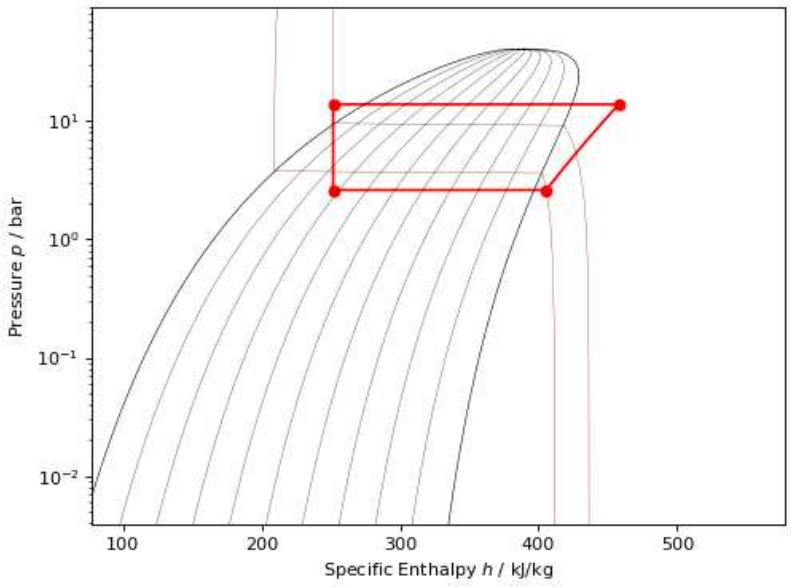
Figura 8 - Código para o cálculo de ciclos termodinâmicos com CoolProp

```
import CoolProp
from CoolProp.Plots import PropertyPlot
from CoolProp.Plots import SimpleCompressionCycle
pp = PropertyPlot('HEOS::R134a', 'PH', unit_system='EUR')
pp.calc_isolines(CoolProp.iQ, num=11)
cycle = SimpleCompressionCycle('HEOS::R134a', 'PH', unit_system='EUR')
T0 = 280
pp.state.update(CoolProp.QT_INPUTS,0.0,T0-10)
p0 = pp.state.keyed_output(CoolProp.iP)
T2 = 310
pp.state.update(CoolProp.QT_INPUTS,1.0,T2+15)
p2 = pp.state.keyed_output(CoolProp.iP)
pp.calc_isolines(CoolProp.iT, [T0-273.15,T2-273.15], num=2)
cycle.simple_solve(T0, p0, T2, p2, 0.7, SI=True)
cycle.steps = 50
sc = cycle.get_state_changes()
pp.draw_process(sc)
import matplotlib.pyplot as plt
plt.close(cycle.figure)
pp.show()
```

Fonte: *Bell (2014)*

Por fim, após a execução das linhas de código relacionadas aos cálculos termodinâmicos, o gráfico de pressão-entalpia é gerado e exibido utilizando a biblioteca *matplotlib*. O resultado da simulação é apresentado na figura 9, ilustrando de forma clara as mudanças de estado ao longo do ciclo termodinâmico.

Figura 9 - Gráfico do ciclo termodinâmico



Fonte: *Bell (2014)*

5.2.2 Ponto de bolha e de orvalho

A biblioteca *Thermo* é bastante útil para a realização de cálculos de constantes termodinâmicas, diagrama de fases e análise do comportamento de sistemas multicomponentes, por meio da utilização de equações de estado. Além disso, com a *Thermo* é possível calcular equilíbrio de fases, composição de líquidos e vapores e gerar gráficos de diagramas ternários e diagrama de fases, sendo uma ferramenta bem interessante para a compreensão de processos termodinâmicos (BELL, 2016-2014).

Alguns conceitos relevantes abordados na disciplina são os pontos de bolha e de orvalho. Esses conceitos são fundamentais no ensino da Termodinâmica, especialmente em sistemas de equilíbrio entre líquido e vapor. O ponto de bolha representa a temperatura ou pressão na qual a primeira bolha de vapor em uma mistura líquida é formada e, nesse ponto, se dá início à vaporização da mistura líquida, delimitando o limite inferior da composição do líquido em equilíbrio com o vapor. Em relação ao ponto de orvalho, esse se refere à temperatura ou pressão na qual a primeira gota de líquido se condensa a partir de uma mistura de vapor ao ser resfriada ou comprimida, estabelecendo o limite superior da composição do vapor em equilíbrio com o líquido.

Sendo assim, os pontos de bolha e de orvalho são essenciais para processos como a destilação, pois facilitam o entendimento de como as composições de líquido e de vapor variam com a temperatura ou pressão, sendo úteis na análise de sistemas multicomponentes e para a

determinação das condições de operação de equipamentos industriais.

Nesse contexto, pode-se utilizar o Python para melhor entendimento dos conceitos de ponto de bolha e de orvalho. A figura 10 detalha um código desenvolvido com a biblioteca *Thermo*, que simula o equilíbrio de fases de uma mistura binária de etanol e água. Em um primeiro momento, a função "*ChemicalConstantsPackage.from_IDs"* importa as propriedades termodinâmicas, que fornecerá informações importantes para os cálculos, como temperatura crítica, pressão crítica e fator acêntrico.

Em relação ao método de cálculo, o código utiliza a equação de estado de Peng-Robinson para a fase vapor, que expressa as propriedades do fluido em termo das propriedades críticas e do fator acêntrico de cada espécie envolvida na mistura (ROBINSON; PENG; CHUNG, 1985) e o modelo UNIFAC para a fase líquida, que calcula os coeficientes de atividade com base em interações dos grupos funcionais das moléculas da mistura (FREDENSLUND et al., 1975).

Além disso, a função denominada "*flasher"* é utilizada para calcular o equilíbrio de fases, permitindo a determinação da composição das fases líquida e vapor em equilíbrio para as condições de temperatura e pressão definidas inicialmente.

Figura 10 - Código para análise dos pontos de bolha e de orvalho

```
from thermo import *
from thermo.unifac import DOUFSG, DOUFIP2016
# Load constants and properties
constants, properties = ChemicalConstantsPackage.from_IDs(['ethanol', 'water'])
# Objects are initialized at a particular condition
T = 300
P = 1e5
zs = [.5, .5]

# Use Peng-Robinson for the vapor phase
eos_kwargs = {'Pcs': constants.Pcs, 'Tcs': constants.Tcs, 'omegas': constants.omegas}
gas = CEOSGas(PRMIX, HeatCapacityGases=properties.HeatCapacityGases, eos_kwargs=eos_kwargs)

# Configure the activity model
GE = UNIFAC.from_subgroups(chemgroups=constants.UNIFAC_Dortmund_groups, version=1, T=T, xs=zs,
                           interaction_data=DOUFIP2016, subgroups=DOUFSG)
# Configure the liquid model with activity coefficients
liquid = GibbsExcessLiquid(
    VaporPressures=properties.VaporPressures,
    HeatCapacityGases=properties.HeatCapacityGases,
    VolumeLiquids=properties.VolumeLiquids,
    GibbsExcessModel=GE,
    equilibrium_basis='Psat', caloric_basis='Psat',
    T=T, P=P, zs=zs)

# Create a flasher instance, assuming only vapor-liquid behavior
flasher = FlashVL(constants, properties, liquid=liquid, gas=gas)
```

Fonte: BELL (2016-2024)

Em adição ao código mostrado na figura 10, com a inclusão das linhas de código apresentadas na figura 11, é gerado um gráfico T-xy, utilizado para representação de relações de equilíbrio entre temperatura (T) e composição dos componentes presentes na mistura (x e y) a uma pressão constante.

A biblioteca *Thermo* por meio da classe *flasher*, permite a simulação com base nos modelos termodinâmicos de Peng-Robinson e UNIFAC com a utilização do método *plot_Txy*. Além disso, no código é definido a pressão constante do sistema em 1 bar, construindo um gráfico com 100 pontos para garantir uma boa resolução das curvas de equilíbrio.

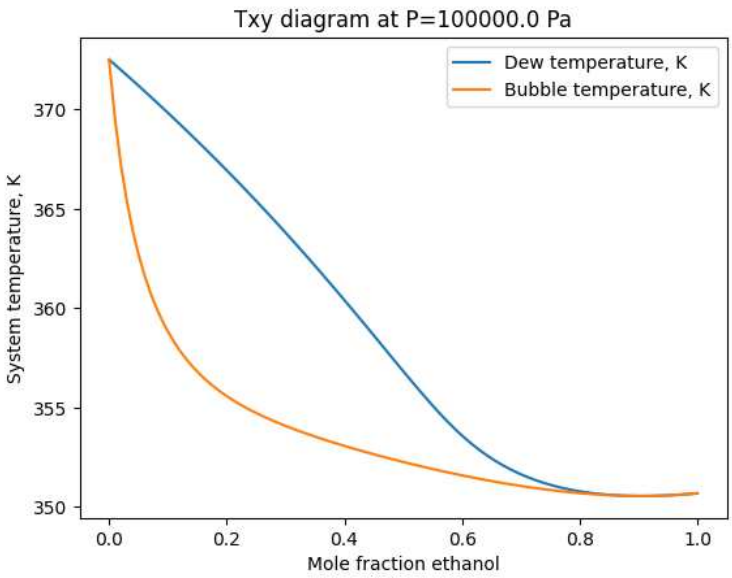
Figura 11 - Código para geração do gráfico de T-xy

```
# Create a T-xy plot at 1 bar
_ = flasher.plot_Txy(P=1e5, pts=100)
```

Fonte: BELL (2016-2024)

O gráfico resultante dos códigos das figuras 10 e 11 é apresentado na figura 12, que ilustra as curvas de equilíbrio líquido-vapor para a mistura binária de etanol e água nas condições especificadas, além de demonstrar as temperaturas de ponto de bolha e de orvalho para diferentes composições. No gráfico, a curva azul representa o ponto de orvalho e a curva laranja o ponto de bolha.

Figura 12 - Gráfico T-xy para o sistema de etanol e água



Fonte: BELL (2016-2024)

A biblioteca *Thermo* também permite a geração de gráficos P-xy, por meio da utilização da função *plot_Pxy* a uma temperatura fixa de 373K, conforme ilustrado na figura 13.

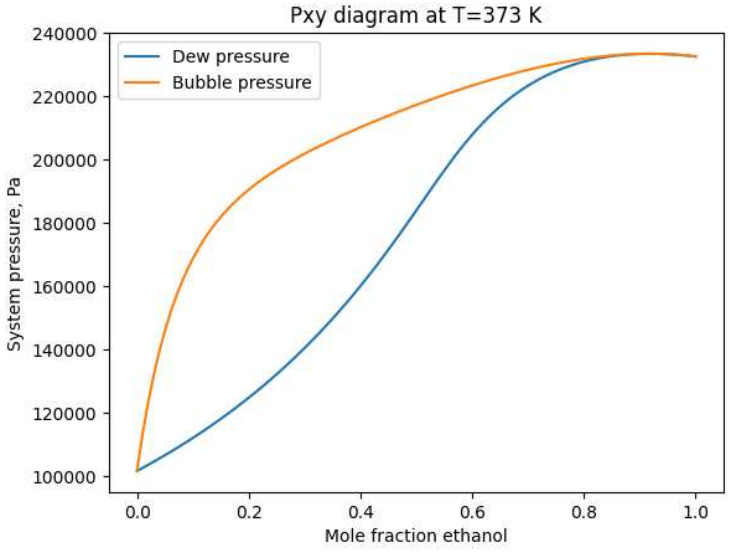
Figura 13 - Código para geração do gráfico de P-xy

```
# Create a P-xy plot at 373 Kelvin
_ = flasher.plot_Pxy(T=373, pts=100)
```

Fonte: BELL (2016-2024)

O gráfico resultante, mostrado na figura 14, representa os pontos de bolha e de orvalho em função da pressão, permitindo o estudo do comportamento do sistema em uma temperatura fixa. Nesse gráfico, a curva azul também representa o ponto de orvalho e a curva laranja o ponto de bolha.

Figura 14 - Gráfico P-xy para o sistema de etanol e água



Fonte: BELL (2016-2024)

A utilização do Python é uma ferramenta promissora para o ensino de termodinâmica para Engenharia Química. Por meio de bibliotecas como a *Thermo*, é possível uma visualização mais fácil e rápida de conceitos teóricos como os pontos de bolha e de orvalho. Com o uso da biblioteca, cálculos complexos são realizados de forma mais rápida e precisa. Além disso, bibliotecas termodinâmicas permitem a simulação de diferentes sistemas termodinâmicos e a sua análise em diferentes condições de temperatura e de pressão, facilitando o entendimento dos conteúdos abordados na disciplina.

5. 3 Processos e Reatores Bioquímicos

O Projeto Pedagógico do curso de Engenharia Química da Universidade Federal do Ceará propõe que a disciplina de Processos e Reatores Bioquímicos seja ministrada no 8° período. A ementa da disciplina aborda diversos conteúdos essenciais, como natureza, classificação e principais enzimas industriais; produção de enzimas; introdução à cinética de reações catalisadas por enzimas; processos enzimáticos e fermentativos (descontínuos e contínuos); projeto de reatores bioquímicos; transferência de massa em sistemas biológicos (fermentadores); ampliação de escala (*scale-up*) em processos envolvendo microrganismos; recuperação dos produtos da fermentação (*downstream*) e noções de bioinformática.

5.3.1 Cinética de processos fermentativos

O estudo da cinética de processos fermentativos consiste na análise e no acompanhamento dos valores de concentração de um ou mais componentes do sistema de cultivo em função do tempo de fermentação. Os componentes dessa reação são o microrganismo, representado pela letra “X”, os produtos do metabolismo (metabólitos), representados pela letra “P” e, por fim, os nutrientes que compõem o meio de cultura e são denominados de substrato, representados pela letra “S” (FERRARI, 2013).

Na disciplina de Reatores Bioquímicos na graduação de Engenharia Química também é abordado o conteúdo de cinética de processos fermentativos. Um dos modelos mais empregados para o acompanhamento e simulação de processos fermentativos é o modelo proposto por Monod. A equação de Monod, bastante análoga à equação de Michaelis-Menten, fornece a taxa de crescimento específica de uma cultura celular, representada abaixo, juntamente com a definição dos termos da equação (KOVÁROVÁ-KOVAR, 1998):

$$\mu = \frac{1}{X} * \frac{dX}{dt} = \frac{\mu máx * [S]}{Ks + [S]} \quad (8)$$

A definição dos termos da equação está descrita abaixo:

1. “μ” é definida como a taxa de crescimento específica;
2. “μmáx” é a taxa máxima de crescimento observada no cultivo;
3. O tempo de cultivo é representado por “t”, dado em horas;

4. K_S é a constante de saturação, equivalente a concentração de substrato no ponto em que μ = μmáx/2, dada em g/L;
5. [S] é a concentração de substrato, dada em g/L.

Desse modo, faz-se necessário a realização de três balanços de massa, um relacionado à célula, outro em relação ao substrato e, por fim, um balanço de massa do produto no meio, as quais são dependentes umas das outras, além de possuírem dependência com o tempo de cultivo. As relações para esses três balanços de massa em um processo de batelada alimentada são descritas pelas equações 9 a 11, sendo a equação 11 a representação do modelo de Luedeking-Piret (SURENDHIRAN et al., 2015).

A equação 9 descreve o balanço de massa de células, demonstrando que a taxa de crescimento é proporcional à concentração celular, multiplicada pela taxa de crescimento específica. Já o balanço de massa para o substrato em um processo de batelada alimentada está representado na equação 10, o qual leva em consideração o coeficiente de rendimento estequiométrico (Y_{XS}), que representa a quantidade de célula formada pela quantidade de substrato consumido. Por fim, a equação 11 representa o balanço de massa do produto em um processo de batelada alimentada e possui em seu equacionamento os coeficientes α e β, que representam a constante de formação de produto associada ao crescimento e não associada ao crescimento, respectivamente.

$$\frac{dX}{dt} = X * \mu \quad (9)$$

$$\frac{dS}{dt} = \frac{Q}{V} * (Si - S) - \frac{X * \mu}{Y_{XS}} \quad (10)$$

$$\frac{dP}{dt} = \frac{Q}{V} * (Pi - P) + X * (\alpha * \mu + \beta) \quad (11)$$

Dessas relações, são definidas as constantes para as equações segundo Santos (2024):

1. “X” é a concentração celular;
2. “μ” é definida como a taxa de crescimento específica;
3. “V” é o volume atual de meio no reator, representado pela expressão $V_0 + Q*t$;
4. “V_0” é o volume inicial do reator;

5. “Q” é a vazão de alimentação;
6. “S_i” é a concentração inicial de substrato alimentada no reator;
7. “Y_{XS}” é o coeficiente de rendimento estequiométrico, constante que representa o fator de conversão de substrato em biomassa, dado pela razão da massa de células produzidas pela massa de substrato consumido;
8. “P_i” é a concentração inicial de produto alimentada no reator;
9. “α” e “β” são as constantes associada ao crescimento do microrganismo e não associada ao crescimento, respectivamente.

A partir das equações 8 a 11, pode-se analisar por meio do Python os parâmetros descritos em um processo fermentativo. Para isso, Santos (2004) estabeleceu valores de $\mu = 2,0$; $K_S = 10,0$; tempo de cultivo até 10h; $X_0 = 2,0$; $S_0 = 5,0$ (concentração do substrato no início do cultivo); $S_i = 5,0$ (concentração de substrato alimentado); $P_0 = 0,0$ (concentração de produto no início do cultivo); $P_i = 0,0$ (concentração de produto alimentado); $V_0 = 5,0$ (volume inicial de cultivo); $Q = 10,0$; $Y_{XS} = 0,5$; $\alpha = 2,0$ e $\beta = 0,0$.

As figuras 15 a 17 apresentam as três partes do código desenvolvido em Python para a modelagem do problema, sendo a primeira parte a definição das constantes a serem utilizadas na simulação.

Figura 15 - Código para o processo fermentativo (Parte 1)

```
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt

#-----------------------------------------------------------#
#|               Definicao das constantes                  |#
#-----------------------------------------------------------#
mimax = 2
Ks = 10
t0 = 0
tf = 10
X0 = 2
S0 = 5
Si = 5
P0 = 0
Pi = 0
V0 = 5
Q = 10
Yxs = 0.5
alfa = 2
beta = 0
```

Fonte: Santos (2004)

A parte 2 do código em Python resolve numericamente o sistema de equações diferenciais que descreve o processo de crescimento microbiano. Inicialmente, é definida uma função chamada “crescimento”, que calcula as derivadas das variáveis biomassa (X), substrato (S) e produto (P) em função do tempo, utilizando parâmetros como vazão (Q), volume do reator (V), e rendimento celular (Y_{xs}).

Em seguida, os valores iniciais das variáveis são armazenados em uma lista, e o tempo é discretizado pela função *np.linspace* para gerar pontos ao longo do intervalo de simulação. Além disso, a função *odeint* presente na biblioteca *SciPy*, é utilizada para integrar as equações e obter as soluções numéricas para X, S e P em cada instante de tempo. Por fim, os resultados são armazenados na variável “Solucao”.

Figura 16 - Código para o processo fermentativo (Parte 2)

```
#-----------------------------------------------------------#
#|                 Declaracao da Funcao                    |#
#-----------------------------------------------------------#

def crescimento (Concentracoes, t):
  X, S, P = Concentracoes
  mi = mimax * S/ (Ks + S)
  V = V0 + Q * t
  dXdt = X * mi
  dSdt = (Q/V) * (Si - S) - X * mi/ Yxs
  dPdt = (Q/V) * (Pi - P) + X * (alfa * mi + beta)
  return dXdt, dSdt, dPdt

#-----------------------------------------------------------#
#|                  Integracao Numerica                    |#
#-----------------------------------------------------------#

iniciais = [X0, S0, P0]
t = np.linspace(t0, tf, 100)
Solucao = odeint(crescimento, iniciais, t)

#Separacao dos vetores solucao
X = Solucao[:, 0]
S = Solucao[:, 1]
P = Solucao[:, 2]
```

Fonte: Santos (2004)

A figura 17 apresenta a última parte do código, na qual utiliza-se a biblioteca *matplotlib* para geração do gráfico que descreve a concentração de células, substrato e produto, em função do tempo de cultivo.

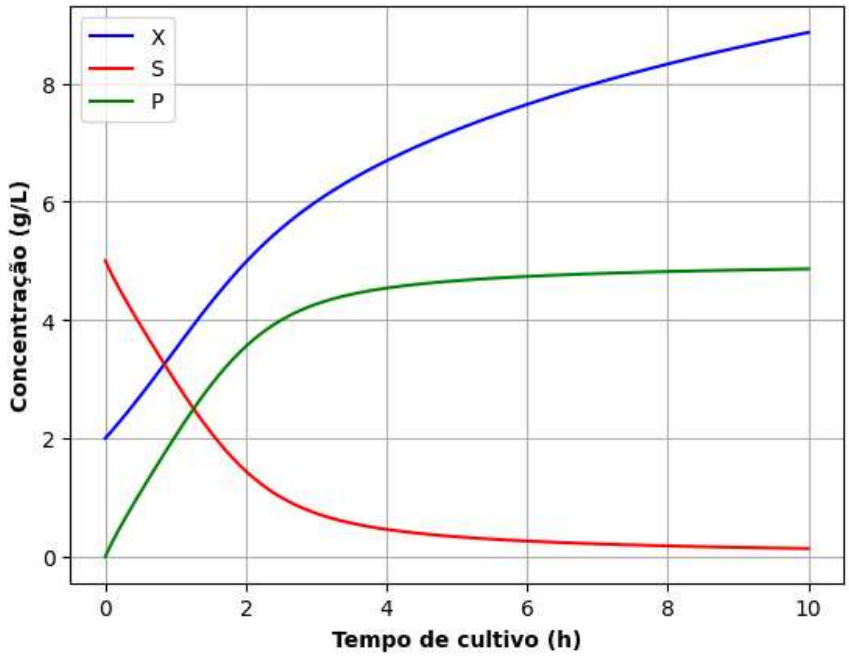
Figura 17 - Código para o processo fermentativo (Parte 3)

```
#-----------------------------------------------------------#
#|                  Plotagem do Grafico                    |#
#-----------------------------------------------------------#

plt.plot(t, X, "b-", label="X")
plt.plot(t, S, "r-", label="S")
plt.plot(t, P, "g-", label="P")
plt.xlabel("Tempo de cultivo (h)", weight="bold")
plt.ylabel("Concentração (g/L)", weight="bold")
plt.legend(loc="best")
plt.grid(True)
plt.show()
```

Fonte: Santos (2004)

Por fim, a figura 18 apresenta o gráfico obtido, demonstrando o comportamento esperado segundo a cinética de Monod para um processo com formação de produto associada ao crescimento: a concentração de células (biomassa) aumenta progressivamente, acompanhada pela produção do metabólito (produto), enquanto a concentração de substrato diminui ao longo do tempo, devido ao seu consumo pelas células para crescimento e geração de energia.

Figura 18 - Concentração de células, substrato e produto em função do tempo de fermentação



Fonte: Santos (2004)

A utilização do Python para o ensino da cinética dos processos fermentativos é uma ferramenta eficaz para a compreensão do comportamento desses sistemas, é possível consolidar melhor os conceitos teóricos, permitindo uma análise detalhada das variáveis envolvidas, como concentração de biomassa, substrato e de produto ao longo do tempo. Essa abordagem, além de facilitar a visualização dos conceitos, estimula também o aprendizado ativo, visto que o estudante pode alterar as variáveis do processo e visualizar de forma rápida o impacto dessas mudanças.

5.3.2 Aplicação em sala de aula

Para demonstrar os benefícios da aplicação prática do Python no processo educacional no curso de Engenharia Química, foi desenvolvida uma aula sobre o tema de cinética de processos fermentativos, com o foco na análise de sensibilidade dos parâmetros que influenciam no comportamento do processo. O objetivo da aula foi consolidar os conceitos teóricos com o uso do Python para proporcionar aos alunos uma compreensão mais profunda, visual e didática do processo fermentativo, além de fortalecer a utilização da programação para modelagem de sistemas.

Figura 19 - Registro da aula prática com Python

Fonte: Elaborado pelo autor

A aula foi planejada para contemplar tanto uma revisão dos conceitos teóricos estudados previamente na disciplina quanto a aplicação prática em Python. Inicialmente, foram revisados os principais conceitos da cinética fermentativa e seus respectivos equacionamentos, sendo

discutidos temas como crescimento microbiano, consumo de substrato e formação de produto - associada, não associada e parcialmente associada ao crescimento. Além disso, foram discutidos parâmetros abordados ao processo fermentativo como a taxa de crescimento microbiano (μmax), a constante de saturação (Ks), o coeficiente de rendimento estequiométrico (Yxs), além dos parâmetros α e β, relacionados à formação de produto associada e não associada ao crescimento, respectivamente.

Na sequência da aula, detalhou-se quais bibliotecas foram utilizadas para o desenvolvimento do código, tendo o uso de *Numpy, SciPy* e *Matplotlib*. O código aplicado na aula foi uma adaptação do código de Santos (2004), ajustado para operação em batelada e modificado para o objetivo da aula de análise de sensibilidade dos parâmetros. A turma foi apresentada ao código ajustado e cada bloco do código foi explicado de forma didática, para garantir o entendimento de cada seção.

A primeira parte do código, representado pela figura 20, inicialmente importa as bibliotecas necessárias para o desenvolvimento do cálculo e para plotagem do gráfico. Em seguida, são definidas as variáveis para o sistema, como tempo da fermentação, que ocorre de 0 a 10 horas. Além disso, há dois blocos de definição dos parâmetros: o primeiro define as constantes cinéticas que serão utilizados para o gráfico de controle e o segundo bloco define os parâmetros para o gráfico que será ajustado. Ou seja, o gráfico controle será estático, para efeitos de comparação, e o gráfico ajustado irá refletir as mudanças realizadas na análise de sensibilidade.

Figura 20 - Código para aula de processo fermentativo (Parte 1)

```
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt

# Tempo da fermentação
t0 = 0
tf = 10

# Definição das constantes (parâmetros para o gráfico de controle)
mimax = 0.25
Ks = 5
X0 = 20
S0 = 60
P0 = 0
Yxs = 0.5
alfa = 1
beta = 0

# Parâmetros para o gráfico ajustado
mimax_2 = 0.5
Ks_2 = 5
X0_2 = 20
S0_2 = 60
P0_2 = 0
Yxs_2 = 0.5
alfa_2 = 1
beta_2 = 0
```

Fonte: Elaborado pelo autor

A figura 21, representa a segunda parte do código, no qual inicialmente é definida a função “crescimento”, que engloba os balanços de massa para o crescimento de células, consumo de substrato e formação de produto, além de definir a cinética de Monod. Em seguida, é definida uma função para calcular o tempo de estabilização do crescimento de células, de modo a determinar o tempo de finalização da fase Log. Isso é feito por meio da função *np.diff(X)*, que calcula a diferença entre valores consecutivos de X. Utiliza-se um laço de repetição para realizar esse cálculo e, quando a variação entre esses valores for menor que 0.01, considera-se que o crescimento estabilizou e é retornado o vetor de tempo para essa condição. Por fim, nessa parte do código é realizada a integração da função “crescimento”, definida anteriormente, utilizando a função *odeint*, que recebe a função de crescimento, os valores iniciais definidos, e os argumentos. O resultado da integração é armazenado em vetores para cada um dos balanços de massa realizados (células, substrato e produto). O tempo de estabilização é retomado e o ponto no qual esse momento ocorre é calculado para futura plotagem no gráfico.

Figura 21 - Código para aula de processo fermentativo (Parte 2)

```
# Função de crescimento (cinética de Monod)
def crescimento(Concentracoes, t, mimax, Ks, Yxs, alfa, beta):
    X, S, P = Concentracoes
    mi = mimax * S / (Ks + S)
    dXdt = X * mi
    dSdt = -X * mi / Yxs
    dPdt = X * (alfa * mi + beta)
    return dXdt, dSdt, dPdt

# Função para encontrar o tempo de estabilização de X
def tempo_estabilizacao(t, X):
    diferenca = np.diff(X)
    for i in range(1, len(diferenca)):
        if abs(diferenca[i]) < 0.01:
            return t[i]

# Função para plotagem do gráfico
def plotar_graficos(mimax, Ks, Yxs, alfa, beta, X0, S0, titulo, ax):
    # Integração numérica
    iniciais = [X0, S0, P0]
    t = np.linspace(t0, tf, 100)
    Solucao = odeint(crescimento, iniciais, t, args=(mimax, Ks, Yxs, alfa, beta))
    # Separando as soluções
    X = Solucao[:, 0]
    S = Solucao[:, 1]
    P = Solucao[:, 2]
    # Calcular o tempo de estabilização de X - Final da fase LOG
    t_estabilizacao = tempo_estabilizacao(t, X)
    X_estabilizado = np.interp(t_estabilizacao, t, X)
```

Fonte: Elaborado pelo autor

O trecho seguinte do código irá realizar a plotagem do gráfico propriamente dita. Primeiro, as curvas para as três variáveis - X, S e P - são geradas com a utilização da função *ax.plot* e definidas as cores para diferenciação: a curva X em azul, a curva S em vermelho e a curva P em verde. Além disso, são gerados os títulos dos eixos do gráfico, além da plotagem do ponto de estabilização da fase Log, cuja cor definida é azul. Os parâmetros utilizados na função são plotados abaixo do gráfico, de modo a facilitar a consulta e visualização dessas variáveis, e a legenda das curvas é plotada e localizada no canto superior esquerdo do gráfico. Por fim, o código cria uma figura com dois gráficos lado a lado e realiza a função para a plotagem final dos gráficos de controle e ajustado, permitindo uma comparação visual entre os dois cenários da simulação.

Figura 22 - Código para aula de processo fermentativo (Parte 3)

```
    # Plotando o gráfico
    ax.plot(t, X, "b-", label="X")
    ax.plot(t, S, "r-", label="S")
    ax.plot(t, P, "g-", label="P")
    ax.set_xlabel("Tempo de cultivo (h)", weight="bold")
    ax.set_ylabel("Concentração (g/L)", weight="bold")
    ax.plot(t_estabilizacao, X_estabilizado, 'bo', label=f"Final - Fase Log: {t_estabilizacao:.2f} h")

    # Exibindo os parâmetros abaixo do gráfico
    parametros = f"μmax = {mimax} / Ks = {Ks}\n\nα = {alfa} / β = {beta}\n\nX0 = {X0} / S0 = {S0}\n\nYxs = {Yxs}"
    ax.text(0.5, -0.3, parametros, transform=ax.transAxes, ha="center", va="center", fontsize=12)

    # Adicionando a legenda
    ax.legend(loc="upper left")
    ax.grid(True)
    ax.set_title(titulo)

    return X, S, P

# Criando uma figura com 2 subgráficos lado a lado
fig, axes = plt.subplots(1, 2, figsize=(16, 8))

# Plotando o gráfico de controle e o gráfico ajustado lado a lado
X1, S1, P1 = plotar_graficos(mimax, Ks, Yxs, alfa, beta, X0, S0,
                             "Controle: Concentrações de X, S e P ao longo do tempo", axes[0])
X2, S2, P2 = plotar_graficos(mimax_2, Ks_2, Yxs_2, alfa_2, beta_2, X0_2, S0_2,
                             "Ajustado: Concentrações de X, S e P ao longo do tempo", axes[1])
```

Fonte: Elaborado pelo autor

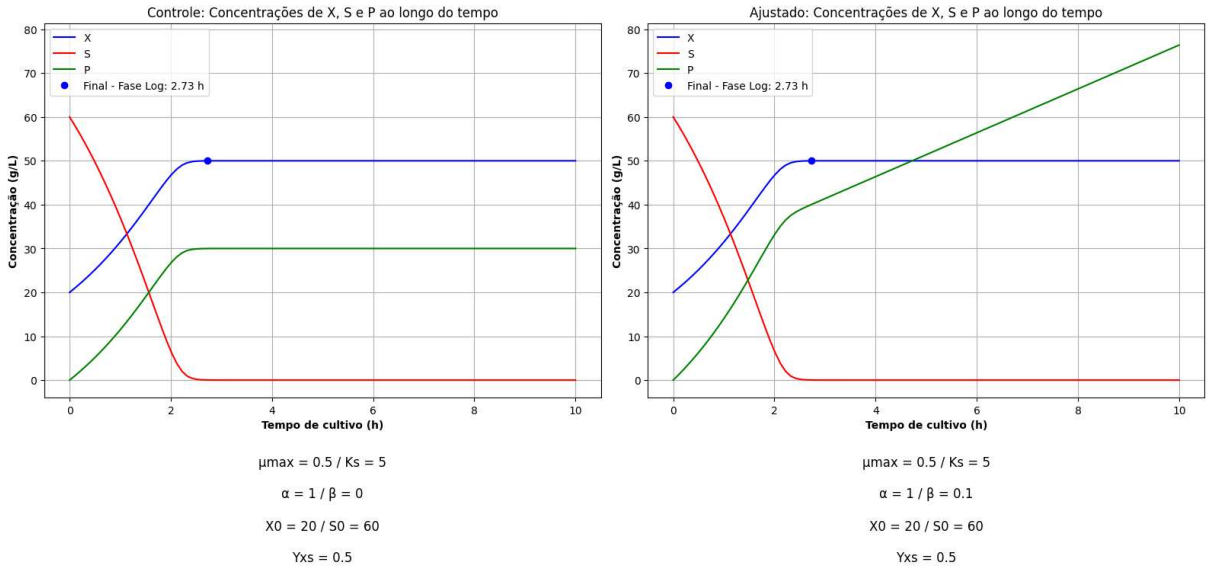
O último trecho do código possui o objetivo de garantir que o eixo y seja mantido sempre na mesma escala, de modo a permitir uma melhor comparação entre o gráfico controle e ajustado. Isso é feito calculando os valores máximos e mínimos do eixo y para X, S e P e define-se esses limites para os dois gráficos. Em seguida, um incremento de cinco unidades é aplicado no valor máximo do eixo y para melhor visualização. Por fim, o resultado é exibido, como pode ser observado na figura 24, exemplo que representa a comparação de uma formação de produto associada ao crescimento com um sistema com formação de produto parcialmente associada ao crescimento.

Figura 23 - Código para aula de processo fermentativo (Parte 4)

```
# Ajustando os eixos y
y_max = max(np.max(X1), np.max(S1), np.max(P1), np.max(X2), np.max(S2), np.max(P2))
y_min = min(np.min(X1), np.min(S1), np.min(P1), np.min(X2), np.min(S2), np.min(P2))

# Definindo limites do eixo y para ambos os gráficos
for ax in axes:
    ax.set_ylim(y_min, y_max + 5)

plt.tight_layout()
plt.show()
```

Fonte: Elaborado pelo autor

Figura 24 - Gráficos controle e ajustado para aula prática com Python



Fonte: Elaborado pelo autor

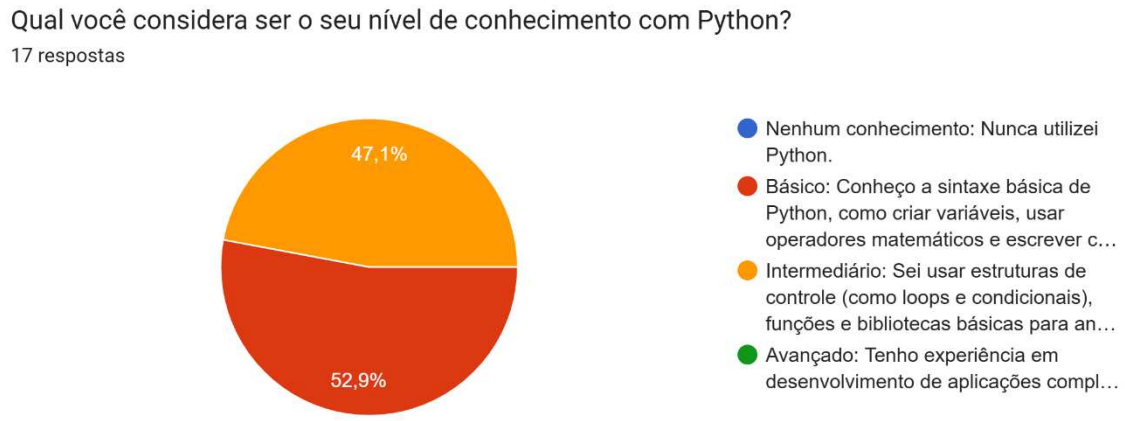
Após a explicação do funcionamento do *script*, a turma acessou os computadores individualmente no laboratório de informática e receberam o link do código, hospedado no *Google Colab*, para criarem uma cópia e avaliarem o comportamento dos gráficos com a mudança dos parâmetros citados, observando os efeitos de aumento e de diminuição e os tipos de formação de produto. Foi pontuado para observarem caso houvesse mudanças no perfil da curva, no tempo para finalizar a fase log e, com base no significado físico de cada parâmetro, refletir o porquê de isso ter acontecido. Para cada parâmetro, a turma avaliava esses pontos durante cerca de 5 minutos e depois era discutido os resultados obtidos para cada mudança e explicado o significado físico desses comportamentos.

Por fim, foi aplicado um formulário para captar a percepção dos alunos em relação à relevância da aula e do uso de Python nesse contexto. Na aula estavam presentes 17 alunos e todos os participantes responderam ao questionário, resultando em 100% de adesão. Dessa forma, os dados obtidos refletem de forma completa o sentimento dos alunos em relação à aula. O formulário aplicado está no apêndice ao final do trabalho.

Observando a figura 25, é possível analisar o nível de conhecimento de Python da turma. O conhecimento dos alunos em relação à linguagem de programação está distribuído de uma forma equilibrada entre básico e intermediário. O nível básico representa 52,9% dos respondentes, o nível intermediário 47,1% e na aula não houve presença de alunos com nível avançado da linguagem. Os alunos que estiveram presentes na aula estão entre o 8° e 10° período do curso, o que implica que ao longo da graduação tiveram algum contato com o Python

e, portanto, possuem nível a partir do básico. Desse modo, é necessário planejar uma aula nivelada para atender a esses dois níveis de conhecimento. Para isso, deve-se retomar a explicação das bibliotecas e funções utilizadas, para que os estudantes com nível básico consigam acompanhar o conteúdo e o conhecimento seja construído de forma gradual para todos.

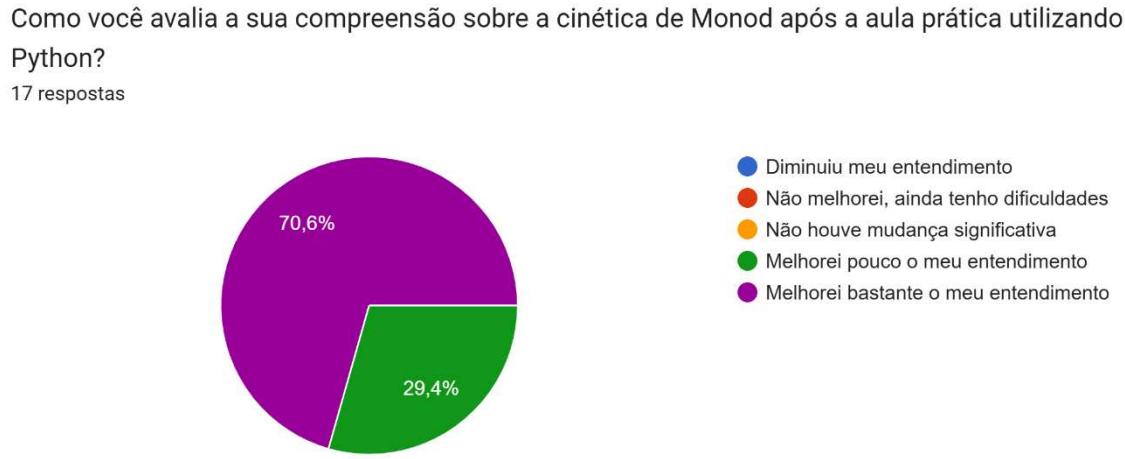
Figura 25 - Nível de conhecimento de Python



Fonte: Elaborado pelo autor

Em relação à compreensão dos conceitos sobre a cinética de Monod, pela figura 26, pode-se observar que 70,6% dos alunos afirmam ter melhorado de forma significativa seu entendimento dos conceitos, enquanto 29,4% consideram que melhoraram de forma mais discreta o entendimento. Além disso, ao analisar a figura 27, nota-se que 52,9% concordam totalmente que o uso de Python ajudou a entender melhor os parâmetros da cinética de Monod, enquanto os outros 47,1% concordam com a afirmação. Esses resultados indicam que a visualização gráfica dos conceitos, aliada ao uso da linguagem Python, foi eficaz em diferentes níveis, demonstrando que ajudou toda a totalidade da turma na assimilação dos conceitos abordados.

Figura 26 - Compreensão do conteúdo com utilização do Python



Fonte: Elaborado pelo autor

Figura 27 - Compreensão dos parâmetros do sistema com utilização do Python

Fonte: Elaborado pelo autor

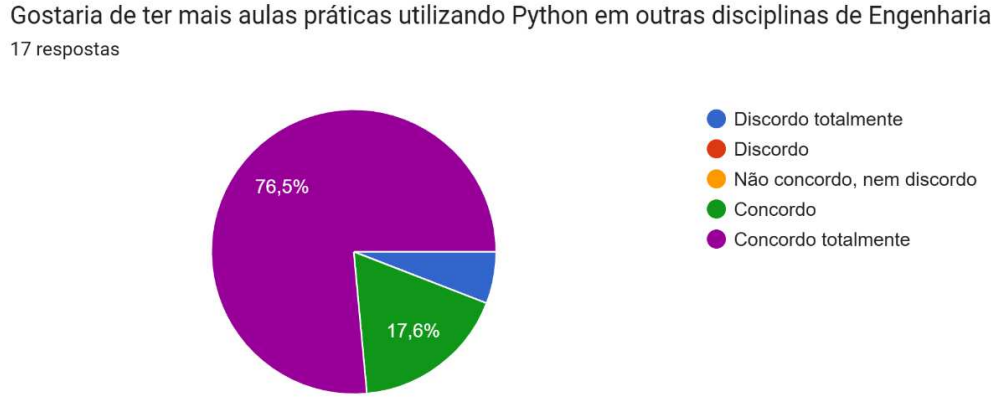
Outra observação interessante sobre as respostas do formulário é que, como pode-se observar na figura 28, 82,4% dos respondentes concordam totalmente com a afirmação de que o uso do Python pode ser uma ferramenta útil em outras disciplinas de Engenharia Química, enquanto os 17,6% dos alunos concordam com a afirmação. Esses resultados demonstram uma aceitação e abertura da turma quando à aplicação do Python, reconhecendo o potencial da linguagem para resolver problemas diversos, sendo uma ferramenta bastante valiosa para o aprendizado.

Figura 28 - Percepção sobre relevância do Python em outras disciplinas

Fonte: Elaborado pelo autor

Quanto à avaliação da afirmação "Gostaria de ter mais aulas práticas utilizando Python em outras disciplinas da Engenharia Química", 76,5% dos respondentes concordaram totalmente, 17,6% concordaram parcialmente com a afirmação e apenas 1 aluno, representando 5,9% do total, discordou totalmente, como pode ser visualizado na figura 29. Ao ser analisada as respostas desse estudante que discordou totalmente com essa afirmativa, foi observado que ele considera possuir o nível básico da linguagem. Isso pode indicar que houve uma dificuldade no acompanhamento e no entendimento do código em Python, o que intensifica a necessidade de nivelar a aula para os diferentes níveis de linguagem de programação, garantindo que todos consigam acompanhar o conteúdo de maneira eficaz, mesmo ao aplicar essa metodologia de ensino em outras disciplinas do curso.

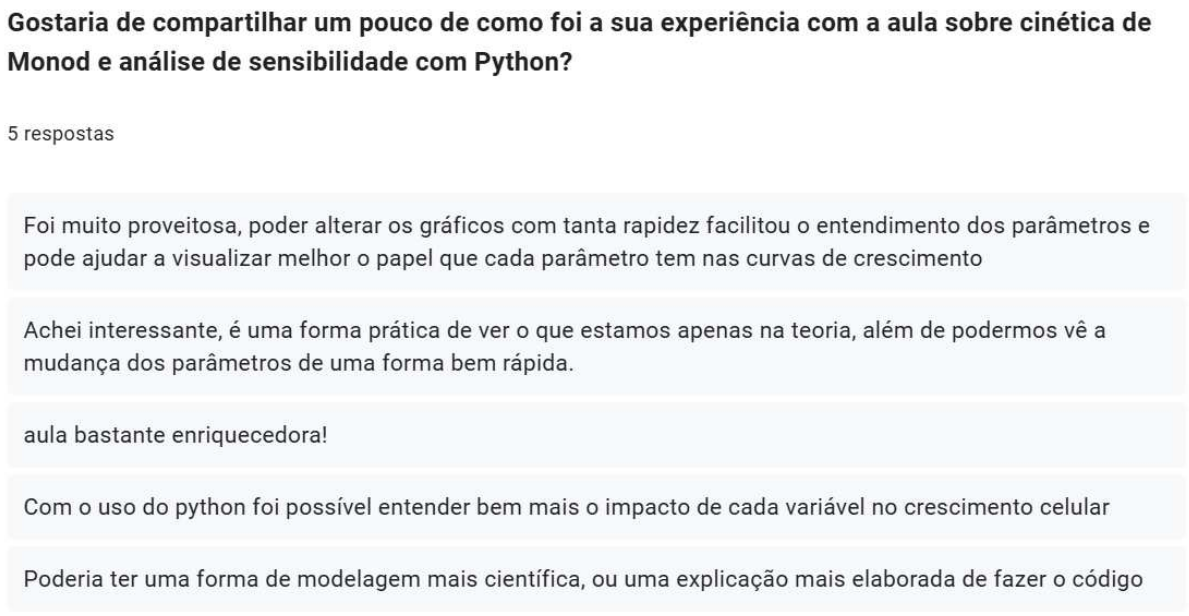
Figura 29 - Percepção sobre aplicação de Python em outras disciplinas



Fonte: Elaborado pelo autor

Por fim, no questionário aplicado após a aula prática, havia um espaço aberto para os alunos que desejassem pudessem compartilhar sobre a sua experiência com a aula sobre cinética de Monod e análise de sensibilidade com Python. As respostas obtidas nesse campo estão apresentadas na figura 30.

Figura 30 - Detalhamento da experiência dos alunos com a aula prática



Fonte: Elaborado pelo autor

De forma geral, os alunos que responderam essa seção do formulário (29,4%) relataram uma experiência positiva com a aula, destacando pontos como a rapidez na alteração dos gráficos, facilitando a visualização e o entendimento dos parâmetros. Além disso, também foi pontuado que a utilização do Python permitiu uma melhor compreensão dos impactos das mudanças nos parâmetros e do papel de cada um deles nas curvas de crescimento. No entanto, um dos alunos sugeriu que poderia ter uma abordagem mais científica ou uma explicação mais detalhada sobre a criação do código, o que indica oportunidades de melhoria no planejamento da aula para, além de garantir uma explicação para que alunos com nível mais básico consigam acompanhar, também promover uma abordagem mais detalhada para que estudantes com um maior entendimento consigam aprimorar ainda mais o conhecimento da linguagem e garantir uma compreensão mais profunda do conteúdo abordado.

6 CONCLUSÃO

O estudo realizado demonstra que na literatura existem diversas aplicações do uso de Python em conteúdos e problemas relacionados à Engenharia Química, como na resolução de equações diferenciais ordinárias e parciais, no ensino de termodinâmica química e no estudo da cinética de processos fermentativos. Essa ferramenta pode ser utilizada em sala de aula para proporcionar um ensino mais tecnológico, dinâmico e prático, e ajudar os alunos a desenvolverem habilidades de programação, preparando profissionais mais capacitados para o mercado de trabalho.

Além disso, a aula prática realizada na disciplina de Reatores e Processos Bioquímicos demonstrou uma grande aceitação e interesse por parte dos alunos. 82,4% dos respondentes concordam totalmente que o uso de Python pode ser útil em outras disciplinas, enquanto os 17,6% restantes concordam parcialmente e reconhecem a utilidade, refletindo um ótimo engajamento com a metodologia.

Porém, os resultados obtidos da aula realizada também indicam a necessidade de ajustes e melhorias, como garantir uma explicação mais detalhada do código, para atender a diferentes níveis de conhecimento. Além disso, para que o aprendizado seja construído de forma gradual e intencional, é necessário um estudo mais aprofundado da grade curricular do curso de Engenharia Química da UFC, com o intuito de entender em quais conteúdos é possível a aplicação de metodologia com Python, de forma que o conhecimento em linguagem de programação seja construído de forma sólida com os estudantes ao longo de toda a duração do curso de graduação.

REFERÊNCIAS

ALVES, Danielson Braga; OLIVEIRA, Fabricio de Figueredo. **Solução de equações diferenciais em Python aplicada a problemas de engenharia.** 2021. Trabalho de Conclusão de Curso (Bacharelado em Ciência e Tecnologia) - Universidade Federal Rural do Semiárido, UFERSA, 2021.

BELL, Caleb. **Thermo: chemical properties component of Chemical Engineering Design Library (ChEDL).** 2016-2024. Disponível em: https://github.com/CalebBell/thermo. Acesso em: 26 jan. 2025.

BELL, Ian H.; WRONSKI, Jorrit; QUOILIN, Sylvain; LEMORT, Vincent. **Pure and Pseudo-pure Fluid Thermophysical Property Evaluation and the Open-Source Thermophysical Property Library CoolProp.** Industrial & Engineering Chemistry Research, v. 53, n. 6, p. 2498-2508, 2014. DOI: 10.1021/ie4033999. Disponível em: http://pubs.acs.org/doi/abs/10.1021/ie4033999. Acesso em: 22 jan. 2025.

BIGELOW, S. **What is NumPy? Explaining how it works in Python.** Disponível em: https://www.techtarget.com/whatis/definition/What-is-NumPy-Explaining-how-it-works-in-Python. Acesso em: 02 fev. 2025.

DYNAMICS AND CONTROL**. Solve Differential Equations with ODEINT**. Disponível em: <https://apmonitor.com/pdc/index.php/Main/SolveDifferentialEquations>. Acesso em: 11 jan. 2025.

ELHALID, O. B.; ALM ALHELAL, Z.; HASSAN, S. **Exploring the Fundamentals of Python Programming: A Comprehensive Guide for Beginners**. Disponível em: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4612765. Acesso em: 12 jan. 2025.

FERRARI, Fernanda Cristina dos Santos. **Fatores operacionais e cinética do processo fermentativo para otimização da produção de etanol em escala industrial.** 2013. xvii, 74 p. Dissertação (mestrado) - Universidade Estadual Paulista, Faculdade de Ciências Agrárias e Veterinárias de Jaboticabal, 2013.

FONTANA, Fábio. **Apostila de Métodos Numéricos I.** Florianópolis: Universidade Federal de Santa Catarina, 2019. Disponível em:

https://fontana.paginas.ufsc.br/files/2017/02/apostila_metI_20191.pdf. Acesso em: 12 jan. 2025.

FREDENSLUND, A.; JONES, R. L.; PRAUSNITZ, J. M. **Group-contribution estimation of activity coefficients in nonideal liquid mixtures.** AIChE Journal, v. 21, n. 6, p. 1086-1099, nov. 1975.

GAMA, R. **A segunda lei de Fick com coeficiente de difusão dependente da concentração**. 2022. Dissertação (Mestrado em Engenharia Mecânica) - Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2022. Disponível em: http://www.bdtd.uerj.br/handle/1/17744. Acesso em: 18 fev. 2025.

HERMANN, M.; PENTEK, T.; OTTO, B. **Design Principles for Industrie 4.0 Scenarios: A Literature Review.** 2015.

HUNTER, J. **History — Matplotlib 3.10.0 documentation.** Disponível em: https://matplotlib.org/stable/project/history.html. Acesso em: 03 fev. 2025.

KOVÁROVÁ-KOVAR, K.; EGLI, T. **Growth Kinetics of Suspended Microbial Cells: From Single-Substrate-Controlled Growth to Mixed-Substrate Kinetics.** Microbiology and Molecular Biology Reviews, v. 62, n. 3, p. 646-666, 1 set. 1998.

KRILL, P. **Python a shoo-in for Tiobe language of the year.** InfoWorld, 2025. Disponível em: https://www.infoworld.com/article/3619998/python-a-shoo-in-for-tiobe-language-of-the-year.html. Acesso em: 02 fev. 2025.

MACHINE LEARNING PLUS. **Matplotlib - Introduction to Python Plots with Examples | ML+.** Disponível em: https://www.machinelearningplus.com/plots/matplotlib-tutorial-complete-guide-python-plot-examples/. Acesso em: 03 fev. 2025.

MULINARI, B. **Numpy Python: O que é, vantagens e tutorial inicial.** Disponível em: https://harve.com.br/blog/programacao-python-blog/numpy-python-o-que-e-vantagens-e-tutorial-inicial/. Acesso em: 02 fev. 2025.

PY-PDE. **2.2.9 Time-dependent boundary conditions.** Disponível em: https://py-pde.readthedocs.io/en/latest/examples_gallery/simple_pdes/time_dependent_bcs.html#sphx-glr-examples-gallery-simple-pdes-time-dependent-bcs-py. Acesso em: 11 jan. 2025.

ROBINSON, D. B.; PENG, D.-Y.; CHUNG, S. Y-K. **The development of the Peng - Robinson equation and its application to phase equilibrium in a system containing methanol.** Fluid Phase Equilibria, v. 24, n. 1-2, p. 25-41, jan. 1985.

SANTOS, Mariana. **Resolvendo problemas de Engenharia de Bioprocessos e Biotecnologia com Python.** Disponível em: https://repositorio.unesp.br/items/789e4386-f27c-46d2-9578-8daf0d946e92. Acesso em: 19 jan. 2025.

SCI-PY. **SciPy user guide, 2025.** Disponível em: https://docs.scipy.org/doc/scipy/tutorial/index.html. Acesso em: 02 fev. 2025.

TIOBE. **TIOBE Index | TIOBE - The Software Quality Company.** Disponível em: https://www.tiobe.com/tiobe-index/. Acesso em: 02 fev. 2025.

VIRTANEN, P. et al. **SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature Methods**, v. 17, n. 3, p. 261-272, 3 fev. 2020.

ZWICKER, D. **py-pde: A Python package for solving partial differential equations.** Journal of Open Source Software, v. 5, n. 48, p. 2158, 3 abr. 2020.

APÊNDICE A - FORMULÁRIO UTILIZADO PÓS-AULA

Avaliação da metodologia de ensino com Python

Objetivo do Formulário:

Este formulário tem o objetivo de coletar a opinião dos alunos sobre o uso do Python na análise da cinética de Monod, aplicada na disciplina de Reatores e Processos Bioquímicos.

As respostas ajudarão a avaliar como essa abordagem contribui para o aprendizado e o interesse em utilizá-la em outras disciplinas da Engenharia Química, como parte do meu Trabalho de Conclusão de Curso (TCC).

As respostas são **anônimas** e serão usadas apenas para fins acadêmicos.

* Indica uma pergunta obrigatória

1. **Qual você considera ser o seu nível de conhecimento com Python?** *

Marcar apenas uma oval.

- ◯ Nenhum conhecimento: Nunca utilizei Python.
- ◯ Básico: Conheço a sintaxe básica de Python, como criar variáveis, usar operadores matemáticos e escrever código simples.
- ◯ Intermediário: Sei usar estruturas de controle (como loops e condicionais), funções e bibliotecas básicas para análise de dados (ex: numpy, scipy, matplotlib).
- ◯ Avançado: Tenho experiência em desenvolvimento de aplicações complexas, automação de processos, integração de sistemas e uso de bibliotecas como TensorFlow, além de programação orientada a objetos e design de software

2. **Como você avalia a sua compreensão sobre a cinética de Monod após a aula prática utilizando Python?** *

Marcar apenas uma oval.

- ◯ Diminuiu meu entendimento
- ◯ Não melhorei, ainda tenho dificuldades
- ◯ Não houve mudança significativa
- ◯ Melhorei pouco o meu entendimento
- ◯ Melhorei bastante o meu entendimento

3. **O uso de Python ajudou você a entender melhor os efeitos dos parâmetros na cinética de Monod.** *

Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo
- ◯ Não concordo, nem discordo
- ◯ Concordo
- ◯ Concordo totalmente

4. **Acredito que o uso de Python pode ser uma ferramenta útil em outras disciplinas de Engenharia Química.** *

Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo
- ◯ Não concordo, nem discordo
- ◯ Concordo
- ◯ Concordo totalmente

5. **Gostaria de ter mais aulas práticas utilizando Python em outras disciplinas de Engenharia Química.** *

Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo
- ◯ Não concordo, nem discordo
- ◯ Concordo
- ◯ Concordo totalmente

6. **Gostaria de compartilhar um pouco de como foi a sua experiência com a aula sobre cinética de Monod e análise de sensibilidade com Python?**