INSTITUTO LATINO-AMERICANO DE TECNOLOGIA, INFRAESTRUTURA E TERRITÓRIO (ILATIT)

ENGENHARIA QUÍMICA

DESENVOLVIMENTO DE UM MÓDULO COMPUTACIONAL PARA PROJETO E ANÁLISE HIDRÁULICA DE SISTEMAS DE TUBULAÇÃO

IGOR WILIS MAUERBERG BARBOSA

Foz do Iguaçu
2025

INSTITUTO LATINO-AMERICANO DE
TECNOLOGIA, INFRAESTRUTURA E TERRITÓRIO
(ILATIT)

ENGENHARIA QUÍMICA

DESENVOLVIMENTO DE UM MÓDULO COMPUTACIONAL PARA PROJETO E ANÁLISE HIDRÁULICA DE SISTEMAS DE TUBULAÇÃO

IGOR WILIS MAUERBERG BARBOSA

Trabalho de Conclusão de Curso apresentado ao Instituto Latino-Americano de Tecnologia, Infraestrutura e Território da Universidade Federal da Integração Latino-Americana, como requisito parcial à obtenção do título de Bacharel em Engenharia Química.

Orientador: Prof. Dr. César Adolfo Rodríguez Sotomonte

Foz do Iguaçu
2025

IGOR WILIS MAUERBERG BARBOSA

DESENVOLVIMENTO DE UM MÓDULO COMPUTACIONAL PARA PROJETO E ANÁLISE HIDRÁULICA DE SISTEMAS DE TUBULAÇÃO

Trabalho de Conclusão de Curso apresentado ao Instituto Latino-Americano de Tecnologia, Infraestrutura e Território da Universidade Federal da Integração Latino-Americana, como requisito parcial à obtenção do título de Bacharel em Engenharia Química.

BANCA EXAMINADORA

_____
Orientador: Prof. Dr. César Adolfo Rodríguez Sotomonte
UNILA

_____
Profa. Drª Marlei Roling Scariot
UNILA

_____
Prof. Dr. Fabyo Luiz Pereira
UNILA

Foz do Iguaçu, _____ de _____ de _____.

AGRADECIMENTOS

Em primeiro lugar, agradeço a Jesus, que é meu Deus e Salvador sempre presente, minha força, sustento e a minha inspiração a não desistir e me dedicar por inteiro em todas as áreas da minha vida.

Aos meus pais, Michel Wilis Barbosa e Helloise Montemor Mauerberg Barbosa, que desde sempre foram exemplo de amor, cuidado e caráter; que durante estes anos de graduação foram fundamentais nessa caminhada, com paciência, força e compreensão. Desde o início eu sempre soube, e agora no final tenho ainda mais certeza, que este diploma também é de vocês.

Ao meu irmão, Brayan Wilis Mauerberg Barbosa, que é meu amigo de todas as horas desde a infância. Obrigado pela parceria, paciência e pela sua ajuda em suportar as dificuldades que se apresentaram ao longo do caminho. Pela ajuda em entender, configurar e arrumar os workflows do VS Code e Git que, de fato, não foram tarefas fáceis e tampouco rápidas.

Ao meu orientador, César Adolfo Rodríguez Sotomonte, por acreditar em mim, e embarcar nessa jornada cheia de imprevistos e surpresas. Obrigado pelos insights, que foram fundamentais para a realização deste trabalho. Pela paciência e pelos apontamentos de melhoria que tornaram possível a realização desse trabalho.

Ao professor Fabyo Luiz Pereira, pela atenção e tempo dedicados em tirar várias dúvidas importantes em relação aos sistemas de tubulação, o seu comportamento e modelagem matemática.

À Marcia Medeiros, pelo constante apoio ao longo da graduação, com paciência e ânimo em ajudar com dúvidas da graduação, ainda que, em alguns casos durante a noite.

RESUMO

É comum à prática de engenharia o contato com sistemas, equipamentos e fenômenos de transporte de fluidos. Na prática, as formas mais comuns que estes sistemas assumem é em uma rede de tubulações e bombeamento. A forma mais comum de simular estes sistemas é com o uso de softwares comerciais. Infelizmente, estes programas não são de fácil acesso aos alunos de engenharia, uma vez que a maioria deles só possui licenças pagas. Portanto, com a finalidade de criar uma ferramenta didática, foi desenvolvido o Conduit Forge, um pacote computacional escrito na linguagem Python, capaz de simular diversos formatos e tipos de redes de tubulações e bombeamento. Visando a organização e legibilidade do código, o pacote foi desenvolvido usando a Programação Orientada a Objetos (POO). Esta metodologia tornou possível a implementação de métodos dunder, o que facilita e simplifica a interação do usuário com o modelo, contribuindo para a redução da curva de aprendizado. Todas as equações foram escritas como equações do SymPy, que é o pacote que desempenhou um papel fundamental ao longo do desenvolvimento deste trabalho. Dessa forma, o solver escolhido foi do tipo Newton-Raphson, já que esta é a opção mais eficiente oferecida como função do SymPy, além de também ser o solver usado em programas comerciais, como o PipeFlow Expert. A livraria Thermo foi escolhida como provedor de propriedades do fluido, por contar com uma vasta base de dados e diversas opções de equações de estado. Os sistemas de tubulação são representados por gráficos direcionados, implementados usando a livraria NetworkX, que é um poderoso pacote de criação, analise e exploração de gráficos direcionados. O gerenciamento da posição dos componentes no espaço foi automatizado com métodos internos a cada um dos diferentes componentes, o que contribui para a organização do código, e previne erros por parte dos usuários. Para validar o módulo desenvolvido, foram selecionados alguns exercícios de livros de mecânica dos fluidos e hidráulica que se encaixavam no escopo deste trabalho. Estes exercícios foram resolvidos tanto com o Conduit Forge quanto com o PipeFlow Expert, a fim de comparar os resultados calculados por ambos os softwares. Os resultados se demonstram satisfatórios para os sistemas avaliados, com uma margem de erro em relação aos valores de referência por volta de 4%.

Palavras-chave: fluidos; softwares; tubulação; bombeamento; simulação.

RESUMEN

En la práctica de la ingeniería, el contacto con sistemas, equipos y fenómenos de transporte de fluidos es común. En la práctica, las formas más comunes que adoptan estos sistemas son redes de tuberías y sistemas de bombeo. La forma más habitual de simular estos sistemas es mediante software comercial. Desafortunadamente, estos programas no son fácilmente accesibles para los estudiantes de ingeniería, ya que la mayoría solo ofrecen licencias de pago. Por lo tanto, con el fin de crear una herramienta didáctica, se desarrolló Conduit Forge, un paquete computacional escrito en Python, capaz de simular diversos formatos y tipos de redes de tuberías y bombeo. Con el objetivo de lograr una organización y legibilidad del código, el paquete se desarrolló utilizando Programación Orientada a Objetos (POO). Esta metodología permitió implementar métodos de dunder, lo que facilita y simplifica la interacción de los usuarios con el modelo, contribuyendo para una reducción de la curva de aprendizaje. Todas las ecuaciones se escribieron como ecuaciones de SymPy, un paquete que desempeñó un papel fundamental durante el desarrollo de este trabajo. Por lo tanto, el solucionador elegido fue del tipo Newton-Raphson, ya que es la opción más eficiente que ofrece SymPy y también se utiliza en programas comerciales como PipeFlow Expert. Se eligió la biblioteca Thermo como proveedor de propiedades de fluidos debido a su amplia base de datos y diversas opciones para ecuaciones de estado. Los sistemas de tuberías se representan mediante grafos dirigidos, implementados con la biblioteca NetworkX, un potente paquete para crear, analizar y explorar grafos dirigidos. La gestión del posicionamiento de los componentes en el espacio se automatizó mediante métodos internos para cada uno de los diferentes componentes, lo que contribuye a la organización del código y previene errores del usuario. Para validar el módulo desarrollado, se seleccionaron algunos ejercicios de libros de texto de mecánica de fluidos y hidráulica que se ajustaban al alcance de este trabajo. Estos ejercicios se resolvieron utilizando tanto Conduit Forge como PipeFlow Expert para comparar los resultados calculados por ambos programas. Los resultados son satisfactorios, para los sistemas evaluados, con un margen de error con los valores de referencia cerca de 4%.

Palabras clave: fluidos; software; tuberías; bombeo; simulación.

ABSTRACT

In engineering practice, the contact with fluid systems, equipment, and transport phenomena is frequent. The most common forms these systems take are in a network of pipes and pumping systems. The most common way to simulate these systems is using commercial software. Unfortunately, these programs are not easily accessible to engineering students, since most of them only have paid licenses. Therefore, in order to create a didactic tool, Conduit Forge was developed, a computational package written in the Python language, capable of simulating various formats and types of pipe and pumping networks. Aiming at code organization and readability, the package was developed using Object-Oriented Programming (OOP). This methodology made it possible to implement dunder methods, which facilitates and simplifies the commands that users must write, smoothing the learning curve. All equations were written as SymPy equations, a package that played a fundamental role throughout the development of this work. Therefore, the chosen solver was of the Newton-Raphson type, as this is the most efficient option offered by SymPy and is also used in commercial programs such as PipeFlow Expert. The Thermo library was chosen as the fluid property provider due to its vast database and diverse options for equations of state. Piping systems are represented by directed graphs, implemented using the NetworkX library, a powerful package for creating, analyzing, and exploring directed graphs. The management of component positioning in space was automated using methods internal to each of the different components, which contributes to code organization and prevents user errors. To validate the developed module, some exercises from fluid mechanics and hydraulics textbooks that fit the scope of this work were selected. These exercises were solved using both Conduit Forge and PipeFlow Expert in order to compare the results calculated by both software programs. The results, for the evaluated systems, are satisfactory, with an error margin from the reference values around 4%.

Key words: fluids; software; pipes; pumping; simulation.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

EPA	United States Environmental Protection Agency
PSF	Python Software Foundation
POO	Programação Orientada a Objetos
ITU	*International Telecomunication Union*
UUID	*Universally Unique Identifier*
ULID	*Universally Unique Lexicographically Sortable Identifier*
SymPy	*Symbolic Python*
Thermo	*Chemical Properties Component*

LISTA DE SÍMBOLOS

A_i	Área de Seção Transversal no ponto i
C_H	Carga Hidrodinâmica
Re	Número de Reynolds
Ma	Número de Mach
f_i	Fator de Atrito de Darcy-Weisbach no ponto i
ϵ	Rugosidade Superficial
$\sum_e i$	Somatório da propriedade i na entrada de um volume de controle
$\sum_s i$	Somatório da propriedade i na saída de um volume de controle
D	Diâmetro
L	Comprimento
C_1	Constante Numérica do número de Reynolds
C_2	Constante Numérica da Carga de trabalho de eixo
C_3	Constante Numérica da Carga Hidrodinâmica
C_4	Constante Numérica da perda de carga Localizada
C_5	Constante Numérica da perda de carga Extensa
μ	Viscosidade Absoluta
η	Eficiência
$\eta_{b,e}$	Eficiência mecânica do trabalho de eixo de uma bomba
ρ_i	Densidade do fluido no ponto i
g	Aceleração da Gravidade
$\dot{m}_i$	Vazão Mássica
m_{VC}	Massa no Volume de Controle
V_i	Velocidade linear no ponto i
$\dot{V}_i$	Vazão Volumétrica no ponto i
Q	Calor
$\dot{Q}$	Taxa de transferencial de Calor

$\dot{W}$	Taxa de Trabalho Mecânico de Eixo
h_L	Perda de Carga
h_b	Carga de Trabalho Mecânico de Eixo
K_L	Coeficiente de Perda de Carga Localizada
h	Entalpia específica
z_i	Elevação no ponto i
P_i	Pressão no ponto i
P	Conjunto de pontos de um gráfico
ΔP	Diferença de Pressão entre dois pontos do escoamento
T_i	Temperatura no ponto i
G_i	Gráfico Direcionado i
E_g	Conjunto de arestas de um gráfico direcionado
α	Número de aberturas em um componente de tubulação
β	Número de equações de conservação de energia criadas a partir de um componente

SUMÁRIO

1 INTRODUÇÃO

A vida cotidiana é permeada por sistemas, fenômenos e equipamentos destinados à manipulação de fluidos. Alguns exemplos são: o preparo de alimentos, aeronaves, automóveis, tubulações e encanamentos residenciais, equipamentos de refrigeração, sistemas de drenagem urbana e esgoto (ÇENGEL, CIMBALA, 2015; WHITE, 2018).

Para viabilizar a utilização desses sistemas em larga escala, é necessário o esforço conjunto de cientistas e engenheiros de diversas áreas, voltado ao desenvolvimento de dispositivos capazes de controlar estes fenômenos (BISTAFA, 2017). Nesse contexto, tais sistemas são uma parte essencial na prática da engenharia, estando presentes em diversas áreas, como Fenômenos de Transporte, Hidrologia, Hidráulica, Compressores, Turbinas, Aerodinâmica, Operações Unitárias (BISTAFA, 2017).

Devido à complexidade destas áreas, torna-se fundamental o uso de ferramentas computacionais no projeto e análise destes sistemas, uma vez que os cálculos envolvidos são extensos e repetitivos. De modo geral, tais ferramentas podem ser condensadas em três categorias principais: softwares comerciais, planilhas eletrônicas e linguagens de programação.

As planilhas eletrônicas são ferramentas amplamente utilizadas no ensino, pesquisa e na prática profissional de engenharia (SIRUANA; ESCAMILLA, 2022). Entre as opções mais usadas, estão Microsoft Excel, LibreOffice Calc e Google Planilhas. Estes softwares apresentam como principal vantagem a facilidade de uso, associada a sua baixa curva de aprendizado, o que as torna acessíveis a usuários com pouca experiencial em programação ou com programas especializados (SIRUANA; ESCAMILLA, 2022; STAMMITTI, 2013). Adicionalmente, estas ferramentas oferecem diversas funcionalidades como análise estatística, plotagem, filtragem condicional, e cálculos automáticos.

Por outro lado, uma das suas desvantagens é a susceptibilidade a erros em fórmulas inseridas pelos usuários, assim como na organização em sistemas complexos ou com grande volume de dados (SIRUANA; ESCAMILLA, 2022). Nestes casos, pode ocorrer queda no desempenho computacional, bem como instabilidades no funcionamento destes softwares (SIRUANA; ESCAMILLA, 2022).

No que se refere a softwares comerciais, destacam-se ferramentas como PipeFlow Expert, EpaNET e Datacor Fathom. Apesar da variedade de funcionalidades

oferecidas, estas soluções requerem licenças pagas e treinamento especializado, o que frequentemente limita a sua acessibilidade em alguns contextos.

No entanto, linguagens de programação como Python, Scilab e MATLAB são alternativas versáteis para a modelagem e simulação de sistemas de engenharia. Estas ferramentas permitem a criação de códigos personalizados e reutilizáveis, o que torna possível a modelagem de fenômenos complexos. Esta abordagem é particularmente útil na simulação de fenômenos altamente específicos e de pouco suporte por parte de softwares comerciais; ou no caso de fenômenos que possuem uma grande quantidade de parâmetros.

Contudo, a aplicação de tais ferramentas para a modelagem de fenômenos complexos pode demandar um tempo significativo, principalmente na implementação de funções e modelos. Por outro lado, com a criação de código modular, o crescimento e manutenção de ferramentas independentes é favorecido, o que contribui para a criação de uma ampla gama de diferentes módulos ao longo do tempo.

Cabe salientar que as ferramentas destas três categorias também podem ser utilizadas em conjunto. Programas comerciais, frequentemente permitem o uso de dados tabulares (armazenados em planilhas) como entradas de dados usadas nos cálculos e simulações. Adicionalmente, tais programas também possuem suporte à integração com rotinas automatizadas através códigos de linguagens de programação. De forma semelhante, as linguagens de programação frequentemente tem suporte a leitura de dados tabulares, o que possibilita a utilização destes dados em modelos computacionais mais complexos.

Neste contexto, o presente trabalho tem como objetivo desenvolver um pacote computacional em Python que simule sistemas de tubulação comuns ao estudo de engenharia.

1.1 OBJETIVOS

1.1.1 Objetivo geral

Desenvolver um módulo computacional para calcular a pressão e a vazão volumétrica em um sistema de tubulações.

1.1.2 Objetivos específicos

i) Realizar a modelagem matemática da conservação de energia e massa para toda e qualquer seção de um sistema de tubulações;

ii) Implementar as diversas entidades e submódulos computacionais para o funcionamento correto do módulo principal;

iii) Comparar os resultados obtidos pelo módulo computacional com outros programas de funcionalidade semelhante, para os mesmos sistemas de tubulações;

2 REVISÃO BIBLIOGRÁFICA

Nesta seção apresentam-se as ferramentas e conceitos de engenharia e programação utilizados neste trabalho, bem como as ferramentas disponíveis que se encaixam no escopo do mesmo.

2.1 FERRAMENTAS SEMELHANTES

Na categoria de pacotes de linguagem de programação, foram encontradas três ferramentas semelhantes ao módulo desenvolvido neste trabalho:

- *Fluids* (BELL *et al, 2016b*);
- Python HVAC (CHRISTIAENS, 2023);
- Pandapipes (LOHMEIER; CRONBACH; DRAUZ *et al, 2020*).

A livraria *Fluids* é voltada a projetos de tubulação com diversos tipos de escoamento (como aberto, fechado, multifásico, leito fluidizado e etc) (BELL *et al, 2016b*). Apesar da variedade de funcionalidade, esta ferramenta não é capaz de simular redes de tubulação complexas, já que o seu escopo abrange somente componentes de tubulação ligados em série. As primeiras versões foram publicadas em 2016 (BELL *et al, 2016b*).

A livraria P*ython HVAC*, de maneira semelhante à *Fluids*, é capaz de lidar com diversos tipos de fluido, incluindo gases. Um diferencial é que esta ferramenta é capaz de lidar com escoamentos com transferência de calor, assim como com redes de tubulação (CHRISTIAENS, 2023). O solucionamento do sistema de equações é feito usando o método de Hardy-Cross, o que reduz a sua eficiência e robustez na resolução de sistemas complexos ou com um grande número de componentes. A primeira versão publicada foi a 0.1.0., disponível desde 22 de março de 2023 (CHRISTIAENS, 2023).

A livraria P*andapipes*, dentre as demais ferramentas pesquisadas, é a ferramenta com escopo e funcionalidades que mais se assemelha ao *Conduit Forge*. Esta ferramenta conta com suporte a redes de tubulação com diversos tipos diferentes de componentes e arranjos complexos (como loops e ramificações). As funcionalidades também abrangem sistemas em regime transiente, e com transferencial de calor. Assim como o Conduit Forge, esta ferramenta usa um solver do tipo Newton-Raphson. Uma das diferenças entre o Pandapipes e o Conduit Forge é que o gerenciamento das juntas, criação de sistemas e posição espacial são feitos de maneira manual, sem nenhuma

automatização. A primeira versão publicada foi a 0.8.4., disponível desde 2 de fevereiro de 2023.

Na categoria de softwares, foi encontrada apenas uma ferramenta, chamada EPANET. Esta ferramenta foi criada com o objetivo de analisar e simular sistemas de distribuição de água (EPA, 2025). As suas funcionalidades incluem uma interface interativa para a criação de sistemas de tubulação, além de diversas funcionalidades para a análise da qualidade da água (EPA, 2025). Apesar de ter suporte ao uso de fluidos com propriedades diferentes da água, uma vez que o foco é em sistemas de distribuição de água, estas funcionalidades são básicas. As primeiras versões desta ferramenta datam da década de 90, até versões mais recentes, como a versão distribuída em 2020 (EPA, 2025).

2.2 A LINGUAGEM PYTHON

A linguagem Python foi criada por Guido van Rossum, em 1989 (MOLS, 2015). A inspiração para criar esta nova ferramenta foi uma outra linguagem na qual van Rossum trabalhou alguns anos antes, a ABC (MOLS, 2015). A partir do conhecimento obtido com a criação da ABC, Guido desenvolveu o Python de forma a priorizar a legibilidade e expansividade.

Segundo a *Python Software Foundation (tradução nossa, [s.d.]a) "*Python é uma linguagem interpretada, orientada a objetos, de alto nível e com semântica dinâmica*"*. Isto significa que, diferentemente das linguagens compiladas (nas quais é preciso compilar o código antes de executá-lo), o processo de execução de um determinado código só possui um passo.

Na versão atual (v3.13), o Python conta com suporte para diversas ferramentas e metodologias, como programação assíncrona, Programação Orientada a Objetos (POO), programação funcional entre outros paradigmas.

Conforme descrito pela *Python Software Foundation* (tradução nossa, [s.d.]b), "[…] as aplicações mais frequentes incluem: Machine Learning, Inteligência Artificial, Análise de dados, Computação Científica e Educação".

A seguir são apresentadas algumas das bibliotecas mais proeminentes da linguagem Python (no escopo de engenharia), atualmente:

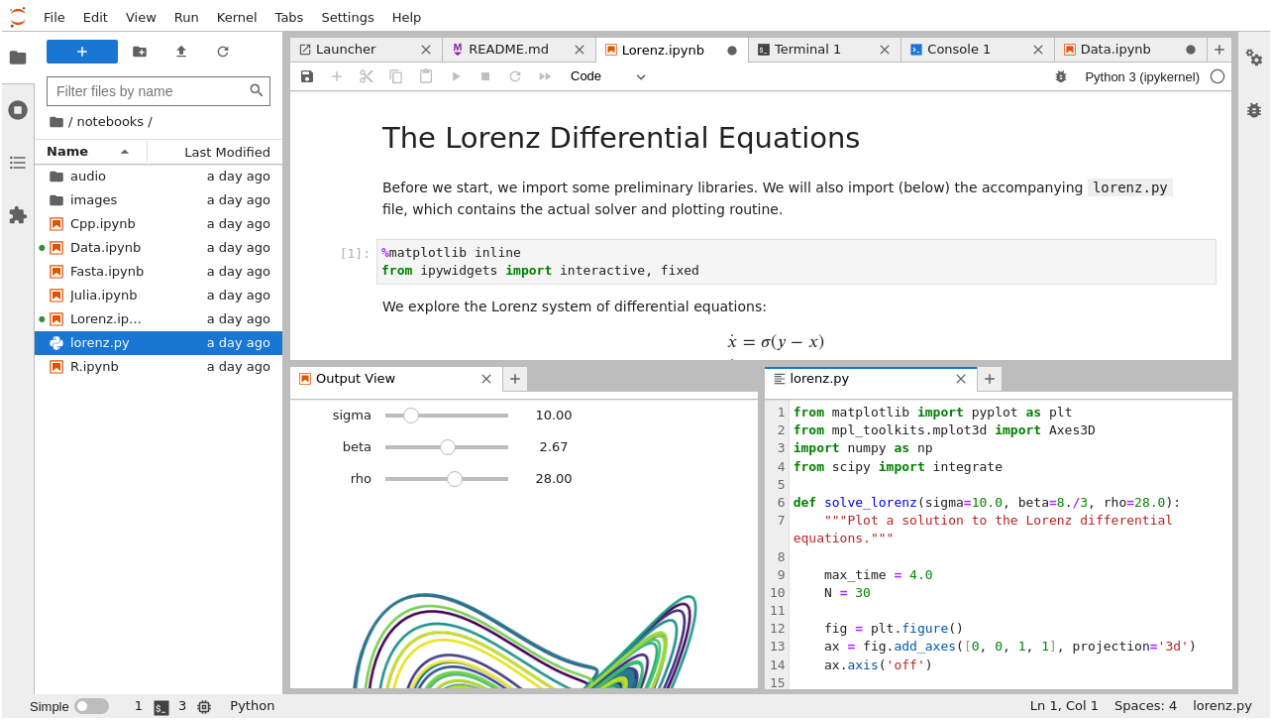
- Pandas (leitura, análise e manipulação de dados, compatível com arquivos csv e

xlsx)

- NumPy (computação numérica científica);
- SymPy (computação matemática simbólica);
- SciPy (ferramentas de algoritmos matemáticos, visualização e funções matemáticas convenientes);
- Matplotlib (pacote gráfico de visualização e plotagem).

Outra ferramenta de grande relevância aos usuários de Python é o “notebook Jupyter”. Esta é uma plataforma de pesquisa científica desenvolvida para padronizar e facilitar o uso de Python para fins acadêmicos, científicos e de engenharia (KLUYVER, 2016). Com ela, o usuário tem uma forma organizada e interativa de escrever, executar e compartilhar códigos em Python (KLUYVER, 2016). Apresenta-se na Figura 1 um exemplo de notebook que contém seções de código, texto, imagens, tabelas e gráficos que podem ser criados e manipulados facilmente (JUPYTER TEAM, [s.d.]).

Figura 1 – Notebook *Jupyte*r aberto na plataforma *JupyterLab*.



Fonte: *Jupyter Notebook Documentation*, *Jupyter Team*.

A partir das funcionalidades nativas e complementares à linguagem Python, observa-se que se trata de um vasto ecossistema, com um grande apoio da comunidade, focado em tecnologias de código aberto e de livre acesso.

Diante dos aspectos previamente discutidos, a linguagem Python foi

selecionada como plataforma para o desenvolvimento do pacote computacional proposto neste trabalho. Essa escolha se fundamenta em um conjunto de características que a tornam particularmente adequada para aplicações científicas e de engenharia, tais como sua sintaxe clara e de fácil leitura, ampla adoção na comunidade acadêmica e disponibilidade de um ecossistema consolidado de bibliotecas especializadas.

2.2.1 Módulos e Pacotes

A *Python Software Foundation* mantém uma plataforma chamada *Python Package Index* (*PyPI*), que é “um repositório de software para a linguagem de programação Python” (PSF, [s.d.]c). Nesta plataforma estão disponíveis módulos, livrarias, pacotes e ferramentas desenvolvidas por vários usuários (por volta de 1 milhão) ao redor de todo o mundo (PSF, [s.d.]c).

Conforme definido pela PSF, um módulo é “a unidade básica de código reutilizável”(tradução nossa, PSF, [s.d.]e). Para o contexto deste trabalho (que trata somente de módulos puros), um módulo puro é “um modulo escrito em Python, e contido em um único arquivo .py” (tradução nossa, PSF, [s.d.]e).

Quando diversos módulos são organizados em conjunto, de forma que estes componentes estejam dentro de um outro módulo, este conjunto é chamado de pacote (PSF, [s.d.]e). Nesta linguagem, também é possível criar pacotes que tenham subpacotes internos, assim como também é possível criar módulos com submódulos internos (PSF, [s.d.]d). Logo, no contexto deste trabalho, a expressão módulo pode ser usada de forma intercambiável com a expressão pacote.

A partir desta organização, é possível que diferentes usuários compartilhem, criem ou alterem código de forma organizada. A criação de módulos permite usar ferramentas já implementadas por outros usuários para esta finalidade.

Além da otimização do tempo gasto pelos usuários, o aspecto modular da linguagem Python também torna a programação mais segura e completa. Desta forma, é comum que sejam implementadas medidas de prevenção de erros e melhorias de usabilidade ao código já existente. Este efeito é amplificado quando existe a contribuição de diferentes usuários a um dado módulo de interesse mútuo.

Nesse processo de constante melhoria e crescimento, os módulos podem se manter atualizados, e até expandir o seu escopo e funcionalidades.

2.3 PROGRAMAÇÃO ORIENTADA A OBJETOS (POO)

A maior parte do código desenvolvido neste trabalho foi estruturada de acordo com a metodologia de Programação Orientada a Objetos (POO). Este é, segundo Santos, Saraiva e Fátima (2018), “um dos paradigmas mais utilizados atualmente no desenvolvimento de sistemas”.

Na POO, são criadas classes que representam computacionalmente entidades materiais (como uma cadeira) ou imateriais (como o número de faltas de um aluno) do mundo real (SANTOS; SARAIVA; FÁTIMA, 2018).

Conforme descrito por Silva Filho (2010, apud *SANTOS; SARAIVA; FÁTIMA, 2018*),

> Classes são as unidades principais do paradigma de orientação a objetos e servem como modelo ou categoria para um determinado conjunto de objetos, isto é, são abstrações que representam as características (propriedades) comuns de todos os objetos que pertencem a elas, bem como definem o seu comportamento.

Logo, os objetos de uma classe são as entidades computacionais criadas a partir da classe. Estes objetos também podem ser chamados de instâncias, e ambas as nomenclaturas são frequentemente intercambiáveis.

Segundo apontado por Santos, Saraiva e Fátima, (2018) “[…] Essa organização passa pela compreensão de conceitos importantes desse paradigma, que buscam melhorar a manutenção dos programas e proporcionar ganho de produtividade por meio do reúso e da subdivisão dos programas”.

Na criação de uma classe, é possível definir atributos. Estes atributos são variáveis que armazenam as propriedades que cada um dos objetos carrega (*SANTOS; SARAIVA; FÁTIMA, 2018*). Como exemplo, considerando que exista uma classe chamada cadeira, usada para criar um ou mais objetos do tipo cadeira. Esta classe pode ter um atributo (de objeto) chamado altura, que armazenaria a altura do assento da cadeira em relação ao chão, no formato de um número.

Logo, observa-se que é possível criar múltiplas instâncias desta classe, onde cada uma delas pode ter um valor do atributo altura diferente dos valores de altura dos demais objetos cadeira. Adicionalmente, no caso em que vários objetos da classe cadeira sejam declarados, e o valor do atributo de cada um deles seja igual, eles ainda são objetos diferentes no sistema, e têm uma identidade única.

De forma semelhante aos atributos, cada classe pode ter operações,

procedimentos e/ou funções (geralmente chamados de métodos). Estes métodos podem ser executados ao longo do código, a fim de que alguma operação seja realizada, como um cálculo, ou a alteração de um atributo.

Um exemplo disso poderia ser o caso de uma classe criada para descrever carros, de forma geral, chamada carro. Considerando que o carro tenha um atributo booleano chamado “ligado”. Se o atributo ligado for igual a *True* (verdadeiro) isto significa que o objeto carro está ligado, Se este valor for *False* (falso), isto significa que o carro está desligado. A partir disso, esta classe pode ter dois métodos, chamados ligar e desligar. O estado padrão dos objetos carro é ligado = *False*, se o método ligar fosse acionado em uma dada instância da classe carro, o atributo ligado seria alterado para *True*.

Considerando uma classe chamada alunos, usada para criar instâncias que representam os alunos que fazem parte de uma turma. Um exemplo de atributo de classe poderia ser uma lista que, quando um objeto da classe é criado, recebe o atributo “nome” (que vem da instância recém-criada). Após a criação das várias instâncias, esta lista pode ser usada como um registro (que pode ser acessado posteriormente pelo usuário) de quais alunos estão na turma.

Outra funcionalidade da programação orientada a objetos é a herança. Conforme Horstmann (2005), “[…] a herança é um mecanismo para melhorar classes existentes, ou seja, se uma classe nova precisa ser implementada e há uma classe com um conceito geral já disponível, a nova classe poderá herdar da classe existente” (apud *SANTOS; SARAIVA; FÁTIMA, 2018*).

Neste trabalho, a herança é uma estrutura fundamental para evitar repetições desnecessárias no código, e facilitar a leitura e manutenção do mesmo. Deste modo, as classes oferecidas ao usuário herdam atributos e métodos de várias classes ocultas.

2.4 MÓDULOS COMPUTACIONAIS EXTERNOS USADOS NESTE TRABALHO

Ao longo deste trabalho foram usados diversos pacotes computacionais (escritos na linguagem Python). Estes pacotes são uteis para diminuir o tempo total de desenvolvimento de uma livraria, além de promoverem uma melhor organização do código, o que aprimora a documentação dos trabalhos realizados. De modo geral, cada

um deles é criado para uma série de tarefas em tópicos relacionados. Por exemplo, a livraria SymPy (*Symbolic Python*) é usada para criar, manipular e organizar equações algébricas (tanto simbólicas quanto numéricas), além de também possuir outras ferramentas, como *solvers* e funções de plotagem (de equações e gráficos).

Outra vantagem da utilização de módulos de código aberto é que, geralmente, todos eles possuem uma extensa documentação sobre a funcionalidade, funcionamento e aplicação de cada uma das suas ferramentas, além de vários métodos internos de resolução de erros.

Todos os pacotes listados nas seções abaixo são gratuitos e de código aberto (*open source*).

2.4.1 Livraria UUID

Em um documento criado por membros da União Internacional de Telecomunicações (ITU, *International Telecommunication Union*), os Identificadores Universalmente Únicos (UUID, *Universally Unique Identifier*) foram criados para identificar informações, objetos e operações (LEACH, MEALLING, SALZ, 2005).

Estes identificadores são um conjunto de números e letras, gerados de forma aleatória, e atribuídos a objetos para os diferenciar uns dos outros de forma prática e segura. Eles são chamados únicos por que cada um deles é único em relação ao espaço de todos os UUIDs criados e ao longo do tempo (LEACH, MEALLING, SALZ, 2005).

Um dado importante é que estes identificadores são números inteiros de 128 bits, que podem ser representados como um conjunto de 36 caracteres hexadecimais, no formato de texto (KAKOLAKI, 2025)(LEACH, MEALLING, SALZ, 2005). Um exemplo de UUIDv4 (da quarta versão), no formato de caracteres hexadecimais poderia ser: “d6a1532c-e4b4-49f3-b67f-3ec30d65abaa”.

Apesar da aleatoriedade, a probabilidade de que dois identificadores sejam iguais (também chamada de probabilidade de colisão) é extremamente baixa e, segundo Kakolaki (2025), para que esta probabilidade chegue a 50%, seria necessário criar $1{,}9 \times 10^{18}$ destes identificadores.

Portanto, estes objetos são considerados únicos e seguros para o uso no contexto deste trabalho. Esta biblioteca foi usada para nomear correntes, aberturas,

componentes e mosaicos, a fim de que cada um deles possa ser facilmente diferenciado dos demais.

2.4.2 Livraria ULID

De modo semelhante aos UUIDs, os ULIDs são chamados de Indicadores Universalmente Únicos e Lexicograficamente Ordenados (*Universally Unique Lexicographically Sortable*) (KAKOLAKI, 2025). Estes indicadores são números de 128 bits (assim como os UUIDs), representados como 26 caracteres. Um exemplo deste tipo de identificador é: “01K9HZE32VNHKDFM89VD6NTE42”.

Os ULIDs também são considerados únicos, por conta da baixíssima probabilidade de colisão (para que a probabilidade de colisão seja de 50%, é necessário criar 1.1×10^{12} identificadores) (KAKOLAKI, 2025). Desta forma, estes objetos nomeiam as equações criadas no pacote desenvolvido neste trabalho, onde o mosaico armazena cada uma delas em um dicionário, com cada ULID usado como chave para a sua respectiva equação em um dicionário.

2.4.3 Livraria SymPy (*Symbolic Python*)

A partir do equacionamento demonstrado na seção 3.2, observa-se que uma parte substancial das operações realizadas são manipulações algébricas. Estas manipulações podem ocorrer tanto na etapa de composição de equações individuais quanto na etapa de solucionamento do sistema de equações, por exemplo, na formação da matriz jacobiana (que é uma matriz usada no algoritmo de Newton-Raphson).

Portanto, para que este processo seja eficiente e compatível com substituições, cálculos e aproximações numéricas, a livraria SymPy foi escolhida para servir como base para todo e qualquer equacionamento no escopo deste trabalho.

Esta livraria é compatível com derivação, integração, simplificação e manipulação de matrizes (tanto simbolicamente quanto numericamente). Além destas funcionalidades, tal ferramenta conta com um grande leque de opções de *solvers* numéricos e métodos de checagem de soluções (MEURER, SMITH, PAPROKI, 2017).

Uma outra vantagem das equações do SymPy é que elas tem funções que retornam cada uma das suas variáveis como um conjunto, que é de grande utilidade

na comparação de igualdade e diferença entre expressões. Adicionalmente, esta livraria é compatível com a declaração de variáveis usando UUIDs, de modo que cada uma das variáveis de todas as equações foram declaradas como UUIDs convertidos para o formato de texto.

2.4.4 Livraria Thermo

Como demonstrado na seção 3.2, os cálculos das propriedades de um escoamento requerem vários dados específicos, como: composição química, viscosidade, temperatura, pressão e densidade. De modo geral, é possível descrever um sistema termodinâmico com duas variáveis de estado que, neste caso, são a temperatura e a pressão. No escopo deste projeto, em se tratando de escoamento incompressível no estado líquido, o efeito da pressão sobre as propriedades do fluido (como viscosidade e densidade) são desprezíveis e podem ser equacionadas como propriedades com forte dependência da temperatura.

Portanto, para determinar estas propriedades, é necessário usar um pacote de cálculo de propriedades termodinâmicas e de transporte. Neste trabalho, o pacote escolhido é chamado Thermo (abreviado de *Chemical properties component of Chemical Engineering Design Library (ChEDL)*).

Este pacote, criado por Bell *et al* (2016a), conta com várias funções e submódulos para projeto e análise de sistemas de engenharia química. Uma das funcionalidades desta ferramenta é a avaliação de propriedades termodinâmicas e de transporte (tanto para componentes químicos puros quanto para misturas de diferentes espécies químicas), que oferece ao usuário a possibilidade de escolha sobre quais equações de estado usar (o que pode ser de grande utilidade na expansão do código desenvolvido no trabalho atual).

As funcionalidades desta ferramenta tornaram possível a criação de alertas de mudança de fase, que são acionados quando uma dada corrente material tem temperatura e pressão tais que exista uma fase gasosa ou sólida, uma vez que este caso foge ao escopo do presente trabalho.

2.4.5 Livraria NetworkX

Conforme descrito na seção 3.2.5, as redes de tubulação (no que diz respeito à relação de conexão de diferentes componentes entre si) podem ser representadas por gráficos direcionados.

A livraria NetworkX foi escolhida para criar, analisar e manipular todos os gráficos direcionados necessários para o escopo deste trabalho. Segundo os criadores desta livraria “NetworkX é um pacote (escrito para a linguagem Python) para exploração e análise de redes e algoritmos de redes” (tradução nossa, HAGBERG; SCHULT; SWART, 2008).

Este é um pacote robusto, que conta com diversas ferramentas e funcionalidades para tal função. Dentre as ferramentas deste modulo usadas no presente trabalho estão: identificação de *self-loops* (que são arestas entre um ponto e ele mesmo), cálculo do grau de entrada de um nó (*in-degree*), cálculo do grau de saída de um nó (*out-degree*), declaração de propriedades personalizadas para cada nó e aresta, ferramentas de plotagem e identificação de *loops*.

3 METODOLOGIA

Neste capítulo serão apresentadas as ferramentas usadas no desenvolvimento deste trabalho, bem como o arranjo destas para alcançar os objetivos estabelecidos na seção 1.1. Isto inclui a modelagem matemática, a estrutura do pacote e o seu funcionamento.

3.1 ESCOPO DO PROGRAMA

O módulo computacional foi desenvolvido para estudantes, para simular sistemas comuns encontrados na pesquisa e prática de engenharia. Portanto, as condições frequentemente encontradas nestes contextos foram usadas para delimitar as funcionalidades do programa.

O escopo do pacote está definido pelas seguintes condições:

- Escoamento incompressível;
- Escoamento isotérmico;
- Escoamento adiabático;
- Escoamento monofásico;
- Fluido composto por somente um componente químico;
- Fluidos Newtonianos;
- Escoamento em regime permanente;
- Sistemas de tubulação com somente uma corrente de entrada;
- Sistemas de tubulação com uma ou mais aberturas de saída;
- Componentes de perdas extensas e localizadas de carga;
- Componentes de entrada de trabalho de eixo no volume de controle (bombas);
- $0 < Re < 10^{8}$;
- $10^{-6} < \epsilon/D < 10^{-2}$;
- Escoamento Subsônico, $Ma < 0{,}3$.

Além das condições citadas, o módulo também foi restringido para resolver somente sistemas nos quais é necessário o cálculo iterativo. Estes sistemas não possuem solução algébrica, e são dependentes de um valor de estimativa inicial para resolver suas equações. Isto se deve, entre vários fatores, à natureza implícita e não linear das equações que compõem o sistema.

Os sistemas de tubulações englobados no escopo deste trabalho podem ser descritos por quatro partes: aberturas de entrada, aberturas de saída, componentes de entrada de trabalho mecânico e seções de tubulação. Destaca-se que as seções de tubulação também englobam os componentes de perda de carga localizada, uma vez que estes componentes são descritos pelas mesmas equações de conservação aplicáveis a tubos.

Para que um sistema arbitrário seja do tipo que requer uma solução iterativa, é necessário que, para cada uma das aberturas de entrada e saída, somente a pressão ou somente a vazão volumétrica sejam definidas. É fundamental que, ao menos uma das aberturas de fronteira do sistema com a vizinhança tenham a pressão definida. Portanto, esta pressão é chamada de “referencia de pressão”. Adicionalmente, é necessário que a potência de trabalho mecânico de eixo das bombas seja definida na sua declaração.

Portanto, as variáveis calculadas pelo pacote computacional são a pressão e vazão volumétrica para todo e qualquer ponto dentro de um dado sistema de tubulação. Dado este escopo, o módulo foi chamado de *Conduit Forge*, que pode ser traduzido como forja de condutos.

3.2 MODELAGEM MATEMÁTICA

Nesta seção são apresentadas as equações e cálculos necessários para criar e resolver o sistema de equações não lineares para todo e qualquer sistema de tubulação criado dentro do escopo deste trabalho.

3.2.1 Conservação de Massa no Volume de Controle

A partir das condições apresentadas na seção 3.1, foram definidas matematicamente as equações que descrevem os sistemas de tubulações no escopo desta livraria.

Uma vez que os componentes de tubulação (e os sistemas de tubulações como um todo) não são sistemas fechados, é necessário considerar que todos os volumes internos dos mesmos é um volume de controle, descrito pelas suas superfícies internas.

Portanto, a partir das considerações feitas no APÊNDICE B – Modelagem Matemática (Conservação de Massa), a equação da conservação de massa pode ser rescrita na forma apresentada na Equação 1.

$$\sum_{e} \dot{V} = \sum_{s} \dot{V} \qquad \text{Equação 1}$$

Logo, utiliza-se a Equação 1 como base para criar as equações de conservação de massa em cada um dos componentes de tubulação suportados pelo pacote desenvolvido neste trabalho.

3.2.2 Conservação de energia no Volume de Controle

De maneira semelhante à modelagem da conservação de massa no volume de controle, aos cálculos realizados para a conservação de energia são feitos levando em conta os pontos delimitados na seção 3.1 Escopo do Programa.

No caso de escoamento turbulento em tubos fechados, o cálculo do fator de atrito de Darcy-Weisbach (*f*) é feito utilizando correlações empíricas, devido à complexidade deste regime de escoamento (ÇENGEL; CIMBALA, 2015).

Existem diversas correlações do fator de atrito, algumas delas são escritas como funções implícitas, que devem convergir em um valor numérico após uma série de iterações. No entanto, também existem correlações explícitas, com as quais é possível calcular diretamente o fator de atrito.

Uma destas correlações é a correlação de Swamee-Jain, que foi escolhida como padrão para todos os casos e exemplos avaliados neste trabalho. Esta correlação foi escolhida por conta da sua relativa simplicidade matemática, e por poupar recursos computacionais.

É importante notar que cada correlação do fator de atrito possui uma faixa de valores numéricos nos quais elas podem ser aplicadas (geralmente do número de Reynolds, ou da rugosidade relativa do tubo). No caso da correlação de Swamee-Jain, a restrição está definida em uma faixa do número de Reynolds, e também em uma faixa da rugosidade relativa do tubo (ϵ / D), como apresentado a seguir, no Quadro 2.

Desta forma, a partir das considerações no APÊNDICE C – Modelagem Matemática (Conservação de Energia), a equação geral da energia foi rearranjada como uma função das pressões e vazões em dois pontos distintos do escoamento, como apresentado na Equação 2.

$$\frac{P_1}{\rho g}+C_3\dot{V}_1^{\,2}+z_1+h_b=\frac{P_2}{\rho g}+C_3\dot{V}_2^{\,2}+z_2+h_L \qquad \text{Equação 2}$$

3.2.3 Criação Automatizada do Sistema de Equações

Conforme apresentado no APÊNDICE D – Configurações e Estados do Sistema de Equações, o pacote desenvolvido cria equações a fim de que o sistema sempre esteja definido, sem prover equações em excesso ou em falta ao solver numérico. O procedimento geral para a composição de equações é apresentado a seguir:

I. Equações de conservação de massa para cada componente individual do mosaico;
II. Equações de conservação de energia e massa para cada uma das juntas;
III. Equações de conservação de energia para cada componente individual do mosaico;
IV. Equações de pressão/vazão para a corrente de entrada (de acordo com os valores definidos pelo usuário);
V. Equações de pressão/vazão para cada uma das correntes de saída (de acordo com os valores definidos pelo usuário).

Cada uma das etapas acima foi estruturada usando como base o livro Engenharia hidráulica (HOUGHTALEN; HWANG; AKAN, 2012), que apresenta um exemplo de quais equações são necessárias para resolver um sistema de tubulações usando o método de Newton-Raphson. Um ponto diferente, no entanto, é que o *Conduit Forge* não escreve as equações de conservação de massa e energia em loops, já que a conservação de massa e energia é tomada sobre cada um dos componentes, independente da sua posição no gráfico direcionado.

Esta alteração evita alguns possíveis erros na identificação de ciclos e ramos do gráfico direcionado, simplificando significativamente a checagem condicional

que uma equação passa antes de ser adicionada à lista de equações de um dado sistema. Apesar desta diferença, ambas as abordagens devem ser matematicamente equivalentes, retornando os mesmos valores.

3.2.4 Geometria Espacial de Sistemas de Tubulação

Conforme apresentado na seção 3.2.2., a única variável espacial utilizada na resolução do sistema de equações é a elevação dos pontos do sistema de tubulação em relação à origem. Contudo, as dimensões dos componentes de tubulação também podem influenciar a disposição espacial de cada ponto do sistema de tubulação.

Portanto, para determinar a posição de cada abertura no espaço, o *Conduit Forge* possui um solver geométrico, que faz este cálculo no espaço cartesiano 2D (na direção z e na direção x), o que já é suficiente para compor as equações de conservação de energia. As definições específicas sobre a organização interna do solver são apresentadas no APÊNDICE A – O MÓDULO GEOMETRY HANDLER.

No caso de uma junta entre duas aberturas, o pacote define que as duas têm as mesmas coordenadas (já que estas coordenadas são consideradas em pontos sobrepostos no espaço). A partir desta definição, o método do mosaico que gerencia as juntas (chamado de *joint*) chama o solver de posição de cada componente de forma individual e automática, começando pelos componentes “mais antigos do mosaico”, aqueles que estão presentes na primeira junta registrada na lista de juntas.

A primeira junta que contem uma abertura e uma corrente geralmente é usada como base para definir a origem do sistema de coordenadas, de forma que a posição da abertura seja (0,0,0). A partir deste passo, todos os componentes do mosaico têm a sua posição resolvida a partir da origem.

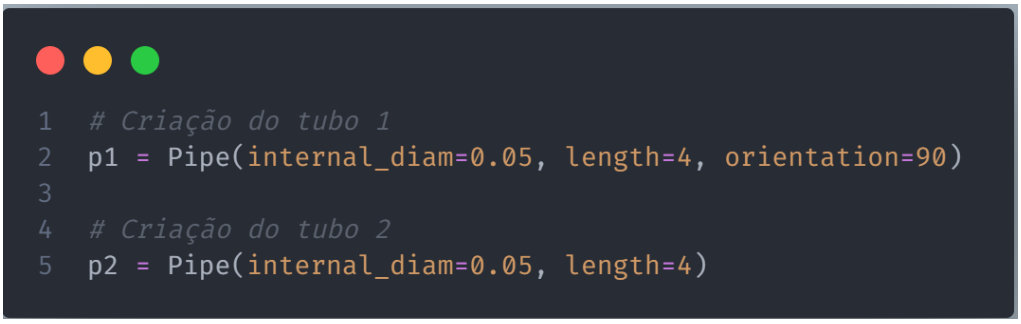
Destaca-se que, devido ao diminuto tamanho dos conectores de tubulação (e também das bombas) em relação ao comprimento das seções de tubulação, os cálculos das posições das aberturas do sistema no espaço tratam estes componentes como pontos. Desta forma, todas as aberturas dos conectores (conectores T, joelhos, cotovelos e bombas) tem as mesmas coordenadas no espaço.

Os benefícios da definição das dimensões de conectores e bombas (a possibilidade de criar visualizações 3D interativas do sistema de tubulações montado pelo usuário) seriam poucos e/ou de pouco proveito para os objetivos deste trabalho (além de

acrescentar complexidade ao uso e à programação do módulo). Portanto, a resolução de coordenadas em 3D não foi implementada. Apesar desta simplificação, o sistema de tubulações continua sendo resolvido sem que as juntas entre aberturas sejam afetadas. Isto por que o gráfico direcionado leva em conta a identidade das conexões entre aberturas, e não a sua posição no espaço.

O cálculo de posição dos tubos ocorre de maneira diferente, uma vez que os tubos têm um comprimento significativo ao sistema. Por conta disso, estes objetos têm um atributo chamado orientação, que representa o ângulo (em graus) entre o tubo e o eixo x. Este atributo tem o valor 0º por padrão, e pode ser definido com outros valores pelo usuário, no comando de criação do objeto Pipe, como demonstrado na figura abaixo (Figura 2).

Figura 2 – Criação de um objeto Pipe, com e sem a definição da propriedade orientação.



```
# Criação do tubo 1
p1 = Pipe(internal_diam=0.05, length=4, orientation=90)

# Criação do tubo 2
p2 = Pipe(internal_diam=0.05, length=4)
```

Fonte: o autor.

3.2.5 Diagramas Relacionais de Aberturas

Um outro ponto tão importante quanto a modelagem matemática dos componentes de tubulação é a informação de como cada componente está conectado aos demais. Para isto, foi escolhida a representação relacional de um sistema no formato de um gráfico direcionado.

As propriedades dos gráficos direcionados usadas ao longo deste trabalho são discutidas com mais ênfase no APÊNDICE E – Gráficos Direcionados.

Como a livraria usa instâncias da classe *Port* (chamadas em português de aberturas) para definir cada um dos pontos de interesse dentro de cada componente, as aberturas foram escolhidas como os objetos usados na representação dos nós do gráfico

direcionado. Portanto, cada um dos nós do gráfico de um mosaico recebem o “nome” dos *UUID*s dos objetos os quais eles representam (correntes ou aberturas).

No gráfico direcionado de um dado mosaico, as arestas representam que duas aberturas estão unidas. Esta união pode ter dois significados diferentes de acordo com o contexto dos vértices. Se os dois vértices de uma aresta fizerem parte de um único componente, esta aresta representa a parte interna de um componente. Se os dois vértices de uma aresta fizerem parte de dois componentes diferentes, esta aresta representa a união entre dois componentes, chamada de junta.

Com a informação obtida a partir do gráfico, o mosaico é capaz de compor as equações de conservação de energia para cada uma das arestas do gráfico, além das equações de continuidade, tanto para componentes individuais, quanto para arestas entre diferentes componentes.

No *Conduit Forge*, o gráfico direcionado também é usado para verificar quais são as correntes de entrada e saída do mosaico, através da seleção dos nós do gráfico que participam de somente uma aresta em todo o gráfico de um dado mosaico. Isso é possível pela verificação do grau de entrada/saída (na notação do *NetworkX*, *in-degree* e *out-degree*) de cada um dos nós, que são os valores que indicam quantas arestas estão “apontando” para dentro ou para fora de um dado nó.

3.3 O ALGORITMO

O *Conduit Forge* foi estruturado para que os usuários pudessem realizar os cálculos do sistema de tubulações da forma mais automatizada possível. Para alcançar este objetivo, o pacote foi estruturado para que o processo de criação e solução do sistema de tubulações ocorresse em uma sequência de poucos passos.

Estes passos são (na ordem de execução do código):

- Criação da(s) Corrente(s) de Entrada;
- Criação do(s) Componente(s) do Sistema;
- Criação do Sistema de Tubulação;
- Solução do Sistema;

Conforme apresentado na Figura 6, para que o sistema de tubulação seja “criado” usando a livraria desenvolvida ao longo deste trabalho, é necessário que o usuário declare a composição química, temperatura e pressão das correntes de entrada,

bem como as ligações entre os diferentes componentes de tubulação. Quando o usuário executa o código, as operações de cálculo da posição das aberturas no espaço, criação de juntas e criação do gráfico direcionado são executadas de forma automática.

3.3.1 Criação da(s) Corrente(s) de Entrada

Nesta etapa, o primeiro passo é a criação do objeto *Chemical* (da livraria *Thermo*). Este objeto é o responsável por armazenar a composição química de uma dada corrente, bem como a pressão e a temperatura da mesma. Estes dados são necessários para que a densidade e a viscosidade do fluido sejam calculados (pela livraria Thermo) e usados para compor as equações de continuidade e energia do sistema de equações.

Por padrão, todas as correntes devem ter a mesma composição, temperatura e pressão. Logo, ainda que seja possível que existam múltiplos objetos de composição (objetos *Chemical*, da livraria *Thermo*), para que todos eles possam ser usados em um mesmo mosaico, é necessário que eles tenham os mesmos valores de composição química (pode ser definida com o nome em inglês da espécie química, ou usando a sua fórmula molecular), temperatura (em Kelvin) e pressão (em Pa). Apesar deste arranjo ser possível, ele é altamente não recomendado, uma vez que pode levantar alertas e erros internos.

Em seguida, a corrente é criada, conforme apresentado na Figura 6, tomando como argumento o objeto *Chemical* criado.

3.3.2 Criação do(s) Componente(s) do Sistema

Nesta etapa, cada um dos componentes do sistema de tubulação são criados. O pacote computacional também cria automaticamente as aberturas dentro de cada componente, que tem os diâmetros declarados pelo usuário.

A maior parte os componentes disponíveis no momento são tubulares, o que significa que todos eles têm duas aberturas. Estes componentes são: tubos, bombas, joelhos (nas opções de 45º, 90º e 180º com flange ou rosqueados) e cotovelos. Todos os componentes disponíveis têm todas as aberturas com o mesmo diâmetro interno, o que diminui a possibilidade de erros por parte do usuário na etapa de conexão das aberturas entre diferentes componentes e/ou correntes.

Nos componentes de perdas menores, o único dado necessário para a criação destes objetos é o diâmetro interno. Nos conectores customizados, é possível declarar o coeficiente de perda de carga localizada.

Nos tubos, como apresentado na Figura 6, é necessário declarar o diâmetro interno e o comprimento (com a orientação como uma variável não obrigatória).

No caso das bombas, é necessário declarar o diâmetro interno, a potência mecânica e a eficiência. Cabe observar que a eficiência é a eficiência de eixo, que indica a eficácia da transformação de trabalho mecânico de eixo em energia mecânica no escoamento.

3.3.3 Criação do Sistema de Tubulação

Nesta etapa, os componentes e correntes são usados para criar o objeto mosaico. Quando um componente é combinado a um outro componente ou corrente usando o operador de soma “+”, o mesmo tem o seu método *__add__()* acionado. Este método que faz parte de uma família de métodos especiais, chamados *dunder*.

É neste método que, durante a primeira soma entre dois objetos de um sistema de tubulação, se a combinação for possível (sem que haja nenhum posicionamento incorreto ou impossível entre componentes), um objeto mosaico é criado, e associado à variável escolhida pelo usuário. No sistema da Figura 6, o mosaico está associado à variável m1.

Na criação do mosaico, o código cria automaticamente a primeira junta adicionada à lista de juntas do mesmo. Para que mais componentes e/ou correntes sejam adicionadas a um dado mosaico, o usuário pode escrever novas “somas” (como apresentado na Figura 3)

Figura 3 – Criação de um mosaico com múltiplos componentes.

```
water = thermo.Chemical("water", T=293.15, P=300_000)
ca = Current(composition=water)

p1 = Pipe(internal_diam=0.03, length=0.4, orientation=0)
p2 = Pipe(internal_diam=0.03, length=0.7, orientation=90)
p3 = Pipe(internal_diam=0.03, length=0.4, orientation=0)

t1 = T_connector(internal_diam=3)

m1 = p1 + ca
m1 = p1 + t1
m1 = t1 + p2
m1 = t1 + p3
```

Fonte: o autor.

Por fim, ainda após a criação de cada um dos componentes, e antes de criação do mosaico, deve ser feita a declaração das propriedades (pressões ou vazões volumétricas) nas entradas e saídas do sistema.

3.3.4 Solução do Sistema

Para que o sistema de equações seja resolvido, o usuário deve chamar o método solve do mosaico em questão. Este método é o responsável por:

- Coletar os valores de chute inicial para cada uma das variáveis (o chute inicial é gerado de forma totalmente automática);
- Montar o sistema de equações;
- Transformá-lo no sistema algébrico no formato próprio para o solver de Newton-Raphson;
- Resolver o sistema
- Retornar os resultados encontrados como valores numéricos dentro de cada um dos objetos corretos.

É possível que o usuário retorne esses valores em forma de texto com legenda, matriz com valores numéricos (sem legenda) ou no formato de um gráfico direcionado, conforme apresentado na Figura 14.

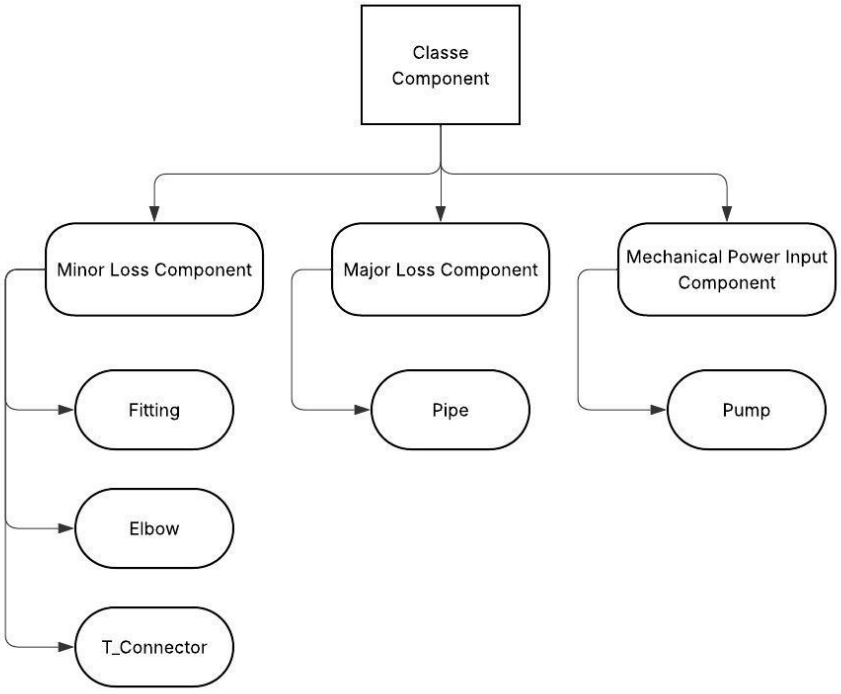
3.4 ESTRUTURA DO PACOTE COMPUTACIONAL

As classes base da livraria são: *Port* (aberturas), *Current* (Correntes materiais), *Component* (Componente) e *Mosaic* (Mosaico). Estas classes foram criadas de forma a ter os atributos e métodos fundamentais que as instâncias das classes filhas possam requerer.

A classe *Port* é descrita em mais detalhes no APÊNDICE F – A Classe Port. De maneira semelhante, as subclasses da classe *Component* tem os seus atributos e funcionalidades explicados de maneira mais específica no APÊNDICE G – As Subclasses da classe Component.

O código foi estruturado visando a modularidade, o que torna o pacote o mais escalável possível, ajudando na organização, expansão e documentação do mesmo. Por conta desta arquitetura, é possível traçar um diagrama de herança da classe *Component*, que é a classe principal da livraria (como apresentado abaixo, na Figura 4).

Figura 4 – Diagrama da Estrutura de Herança da classe *Component*.



Fonte: o autor.

Conforme apresentado na Figura 4, a classe *Component* é a classe usada para definir atributos e métodos que toda e qualquer classe filha deverá ter. No que diz respeito aos componentes de tubulações, cada um dos componentes criados precisa ter os seguintes atributos:

- Duas ou mais aberturas;
- Um material;
- Um ID.

De forma semelhante, cada um dos componentes criados precisa ter os seguintes métodos (excetuando-se, evidentemente, o método construtor __init__):

- reynolds;
- darcy_pressure_drop_coefficient;
- E_constant.

A partir destes métodos base, cada uma das classes filhas tem atributos e métodos específicos, que interagem e se combinam aos métodos padrão de diferentes formas.

3.4.1 A Classe *Current*

Esta classe armazena todas as informações das correntes materiais do escoamento em um dado ponto: composição química, temperatura, pressão, vazão volumétrica, vazão mássica, viscosidade e densidade.

A composição química é armazenada como um objeto *Chemical*, da livraria Thermo (BELL, 2016a). Este pacote é capaz de calcular a densidade e a viscosidade de compostos químicos, com base nos valores de temperatura, pressão e composição química; o que será particularmente útil nos cálculos numéricos que ocorrem antes e depois do sistema de equações ser solucionado.

3.4.2 A Classe *Component*

A classe *Component* é a classe base de todos os componentes de tubulação existentes na livraria. Logo, ela contém como atributos os aspectos mais

básicos que um componente de tubulação pode ter como: quantidade de aberturas, lista de aberturas, id, material e *location*.

Tomando como base cada um desses atributos, as classes filhas da classe *Component* limitam ou definem esses parâmetros de acordo com as suas características. Por exemplo, na classe Pipe, a quantidade de aberturas é sempre limitada a duas.

Esta classe também têm os seguintes métodos: *__add__*, *reynolds*, *position_solver*, C3*_constant* e *darcy_perssure_drop_coefficient*. Estes são os métodos básicos que todo e qualquer componente de tubulação deve ter, uma vez que eles serão usados na etapa de coleta de equações, montagem do sistema de tubulações e na solução do sistema de equações.

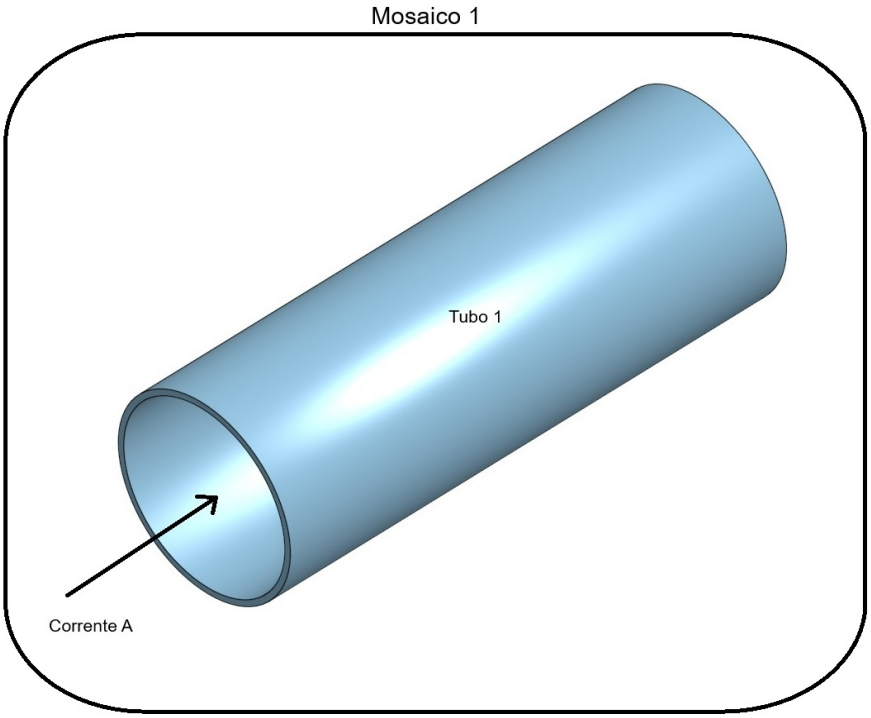
As seções abaixo são uma listagem de todas as subclasses da classe *Component*, bem como a descrição dos seus atributos e métodos característicos.

3.4.3 A classe Mosaic

A classe *Mosaic* é responsável por criar objetos (chamados mosaicos) que gerenciam e armazenam as conexões de todos os diferentes componentes de um sistema de tubulação.

As instâncias desta classe representam um dado sistema de tubulações componentes de tubulação. Portanto, quando um sistema de tubulação é criado, este é representado por um objeto mosaico. Nas Figuras 5 e 6 são presentadas as representações gráfica e computacional, respectivamente, do sistema de tubulação simples (um tubo ligado a uma corrente de entrada).

Figura 5 – Representação gráfica de um sistema simples de tubulação.



Fonte: o autor.

Figura 6 – Representação computacional de um sistema simples de tubulação.

```
1
2  # Criação da composição química da corrente a
3  pure_water = thermo.Chemical("water", T=303, P=300_000)
4
5  # Criação da corrente a
6  ca = Current(composition=pure_water)
7
8  # Criação do tubo 1
9  p1 = Pipe(internal_diam=0.05, length=4, orientation=0)
10
11 # Criação do mosaico 1
12 m1 = p1 + ca
13
```

Fonte: o autor.

O atributo de maior importância nos objetos da classe *Mosaic* (chamados de mosaicos) é a lista joints_list. É nesta lista que são armazenadas as informações relacionais entre diferentes correntes e aberturas que fazem parte do mosaico avaliado.

De modo geral, esta lista armazena juntas, que são conexões entre aberturas (e/ou correntes) de diferentes componentes. A partir da lista de juntas é possível criar o gráfico direcionado que descreve todas as relações de conexão de um dado sistema de tubulação. No caso de um sistema de tubulações no qual existe uma ligação de uma corrente com a abertura de um componente, a junta da corrente é criada da mesma forma que uma junta entre duas aberturas, o que faz com que correntes (criadas pelo usuário, e com junta com uma das aberturas de um componente) sejam representadas como vértices do gráfico direcionado.

Este gráfico direcionado, por sua vez, é usado para compor cada uma das equações de conservação de massa e energia (que são funções das pressões e das vazões volumétricas em cada uma das aberturas e correntes do sistema de tubulações) do sistema de equações que é resolvido para calcular as variáveis desejadas.

Este gráfico também é usado como *input* de uma etapa de verificação, na qual ocorre a checagem de todas as arestas e nós do gráfico, para conferir se nenhum deles viola regras de formato e quantidade de ligações.

4 RESULTADOS E DISCUSSÕES

A fim de validar o módulo criado, foram selecionados exercícios de livros de mecânica dos fluidos, cujos resultados servirão como valores de referência.

Nos exercícios escolhidos, é recorrente a manipulação algébrica das equações de conservação até que estas estejam na forma adequada para os cálculos iterativos. A parte iterativa dos exercícios geralmente é resolvida por meio do software EES ou de planilhas eletrônicas.

Portanto, dada a natureza dos problemas escolhidos, os exercícios serão resolvidos no Conduit Forge e no software PipeFlow Expert, que é um software comercial capaz de realizar os mesmos cálculos e sistemas que o módulo desenvolvido.

4.1 VALIDAÇÃO DO MÓDULO COMPUTACIONAL

Apresentam-se nesta seção os sistemas simulados com o módulo em questão, em comparação com os valores de referência (tanto de programas comerciais quanto de livros de mecânica dos fluidos). Estes exercícios foram selecionados com base na natureza iterativa dos problemas e a composição química dos fluidos nos sistemas analisados. De modo geral, as composições químicas dos escoamentos dos demais exercícios são misturas de diferentes espécies químicas, como o óleo diesel, gasolina e os óleos SAE. Assim, por não se enquadrarem no escopo atual deste trabalho, estes exercícios não foram selecionados para a etapa de validação.

Os exercícios selecionados de White (2018) buscam calcular a vazão volumétrica como resultado, e exercícios desse tipo são encontrados com maior abundância neste material. Entretanto, como a vazão e a pressão ao longo do sistema são variáveis acopladas, os algoritmos de solução sempre calculam a pressão e vazão em cada um dos pontos do sistema de tubulações. Desta forma, os exercícios que buscam calcular a pressão em algum ponto do sistema também são possíveis, ainda que estejam em menor quantidade nos materiais selecionados, ou estejam entre os sistemas compostos por múltiplas espécies químicas.

O exercício 2, selecionado de Couto (2018), é um dos casos no qual as incógnitas são os valores de pressão, e os valores fixos são as vazões volumétricas. O sistema também é composto de 12 tubos interligados, que formam uma rede residencial

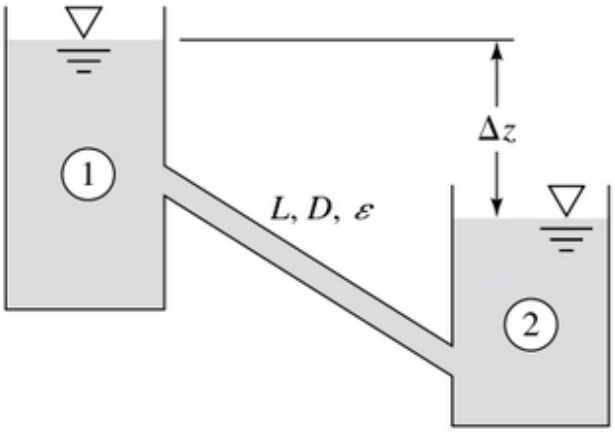
de distribuição de água.

4.1.1 Exercício P6.55 (WHITE, 2018)

Neste exercício, dois reservatórios com diferentes níveis estão ligados por um tubo.

Os reservatórios apresentados na Figura 7 contêm água a 20 ^{o}C. Se o tubo é liso com L = 4.500 m e d = 4 cm, qual será a vazão em m^3/h para Δz = 100 m?

Figura 7 – Sistema de tubulações analisado no exercício P6.55.



Fonte: *White (2018)*.

Como a pressão nos reservatórios é dada em termos de diferença de nível entre os dois, considerou-se que o reservatório 1 está com nível de 0 m, e o reservatório 2 está com nível de -100 m. Estas elevações são definidas indiretamente no *Conduit Forge*, por meio do ângulo entre o tubo e o eixo x. Com estes dados, o ângulo calculado entre o tubo e o eixo x é de -1,27 º.

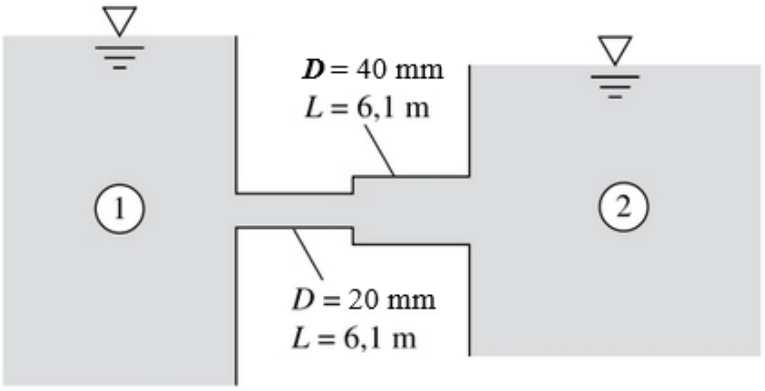
Por fim, obteve-se que a vazão volumétrica do reservatório 1 para o reservatório 2 foi 0,0010 m^3/s.

4.1.2 Exercício P6.103 (WHITE, 2018)

Os reservatórios na Figura 8 estão conectados por tubos de ferro fundido unidos abruptamente, com entrada e saída em canto vivo. Incluindo as perdas

localizadas, calcule a vazão de água a 20 ºC, se a superfície do reservatório 1 está 13,7 m mais alta que a do reservatório 2.

Figura 8 – Sistema de tubulações analisado no exercício P6.103.



Fonte: *White (2018)*.

Na primeira etapa, a corrente material é criada com a temperatura definida no exercício, e a pressão é definida como a carga estática da coluna d'água formada acima do nível z = 0. Logo, a pressão no reservatório de saída é zero, uma vez que a elevação da superfície deste reservatório é zero. A composição dessa corrente deve ser definida com a pressão atmosférica absoluta, uma vez que as propriedades da corrente material serão calculadas levando em conta a pressão absoluta, tanto na entrada quanto na saída do sistema.

Em seguida, são criados os tubos e os componentes de perda de carga localizada, que representam a entrada em canto vivo, a expansão brusca entre os dois tubos, e a saída em canto vivo.

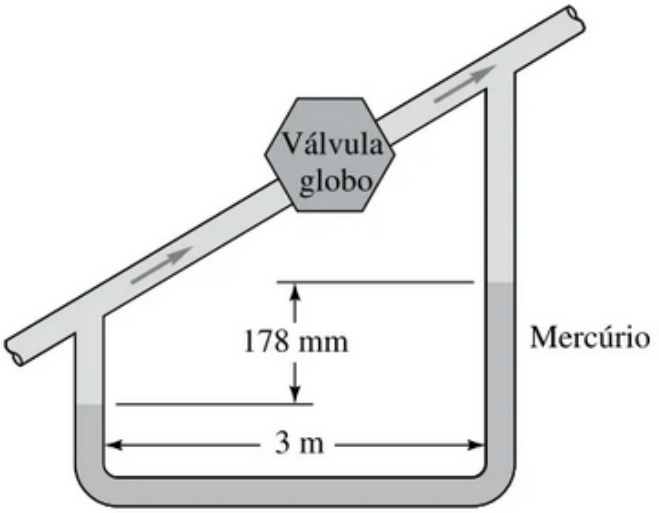
A próxima etapa é criar o mosaico que representa o sistema de tubulação. A sua criação começa com a conexão da corrente material ao componente de entrada em canto vivo. A entrada em canto vivo é conectada ao tubo 1, e assim por diante, com todos os componentes em série. O resultado obtido é que a vazão volumétrica entre os dois reservatórios foi 0,0011 m^3/s.

4.1.3 Exercício P6.106 (WHITE, 2018)

O tubo de água na Figura 9 está inclinado para cima a 30º. O tubo é liso e tem 25 mm de diâmetro. A válvula globo flangeada está completamente aberta. Se o

manômetro de mercúrio indica uma leitura de 178 mm, qual é vazão em L/s?

Figura 9 – Sistema de tubulações analisado no exercício P6.106.



Fonte: *White (2018)*.

O primeiro passo para resolver esse exercício é calcular a pressão em um dos pontos do sistema. Para isso, é possível calcular a diferença entre as pressões em 1(ponto com menor elevação) e 2(ponto com maior elevação). Esta diferença é definida conforme apresentado na Equação 3.

$$P_1 - P_2 = (\rho_{Hg} - \rho_{água}) \cdot g \cdot \Delta_{z,Hg} + \rho_{água} \cdot g \cdot \Delta_{Z,água} = 41{,}5l$$
$$\Delta_{z,Hg} = z_{2,Hg} - z_{1,Hg}, \quad z_{1,Hg} = 0$$
$$\Delta_{z,água} = z_{2,água} - z_{1,água}, \quad z_{1,água} = 0$$

Equação 3

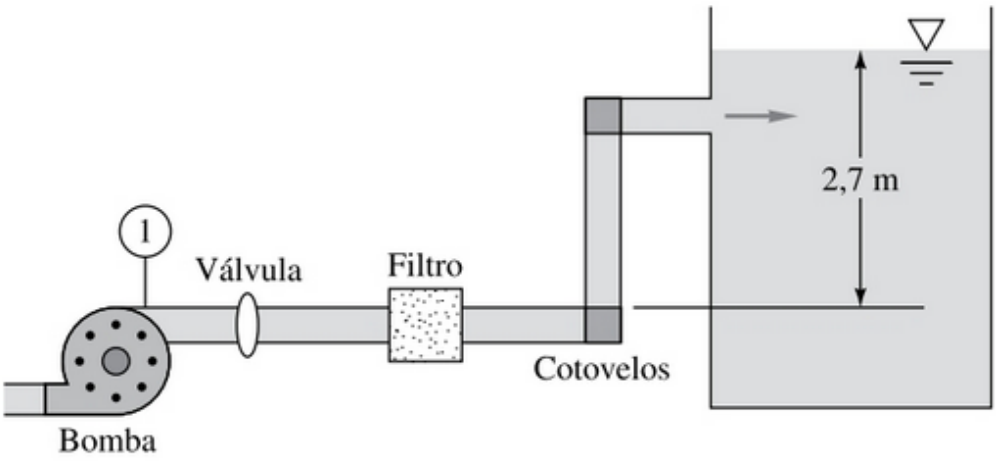
A pressão no ponto 2 foi definida como 0 Pa, a fim de facilitar o cálculo da vazão. Com este dado, a corrente material de entrada pode ser criada, com a pressão calculada. Em seguida, o tubo foi criado, e teve a pressão na abertura de saída definida como 0 kPa.

Por fim, é necessário acionar o solver, e escolher as opções de plotagem. Obteve-se que a vazão volumétrica entre os dois reservatórios foi 0,00084 m^3/s.

4.1.4 Exercício P6.108 (WHITE, 2018)

A bomba d'água na Figura 10 mantém uma pressão de 45 kN/m2 no ponto 1. Há um filtro, uma válvula de disco aberta pela metade e dois cotovelos normais rosqueados. Há 24 m de tubo de aço comercial de 100 mm de diâmetro. Se a válvula de disco é aberta totalmente e Kfiltro = 7, qual é a vazão resultante?

Figura 10 – Sistema de tubulações analisado no exercício P6.108.



Fonte: *White (2018)*.

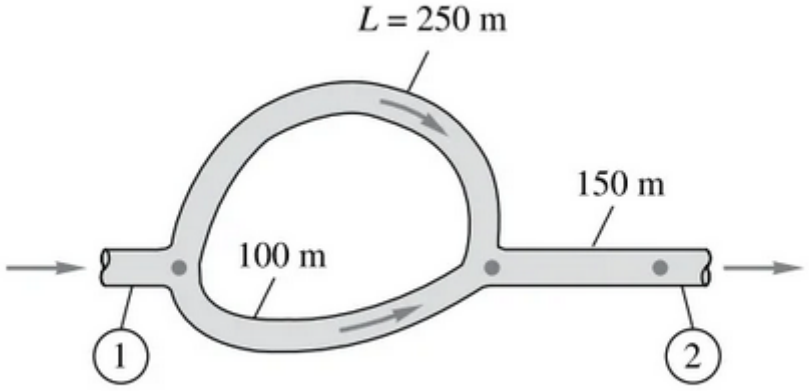
Considerou-se que a elevação no ponto 1 é zero, logo, a elevação no ponto 2 é 2,7 m. Desta forma, foi calculado o ângulo de inclinação do tubo 1 em relação ao eixo x, para que a elevação das duas portas do tubo tenham os valores calculados.

Isto posto, é possível criar o tubo e os componentes de perda de carga localizada. Por fim, o resultado obtido é que a vazão volumétrica de água que entrou no reservatório é de 0,0129 m^3/s.

4.1.5 Exercício P6.116 (WHITE, 2018)

Para o sistema série-paralelo da Figura 11, todos os tubos são de ferro fundido asfaltado de 8 cm de diâmetro. Se a queda total de pressão p1 – p2 = 750 kPa, determine a vazão resultante Q em m^3/h para água a 20 ºC. Despreze as perdas localizadas.

Figura 11 – Sistema de tubulações analisado no exercício P6.116.



Fonte: White (2018).

Foi definido que a pressão no ponto 2 é zero, logo, a pressão no ponto 1 é 750 kPa. Logo, a pressão da corrente de entrada foi definida como 750 kPa, e a pressão da abertura de saída (no ponto 2) foi definida como 0 kPa.

Neste sistema, como todas aberturas estão à mesma elevação, foi definido que todas as aberturas estão a 0 m de elevação. Portanto, a inclinação em todos os tubos do sistema é 0º.

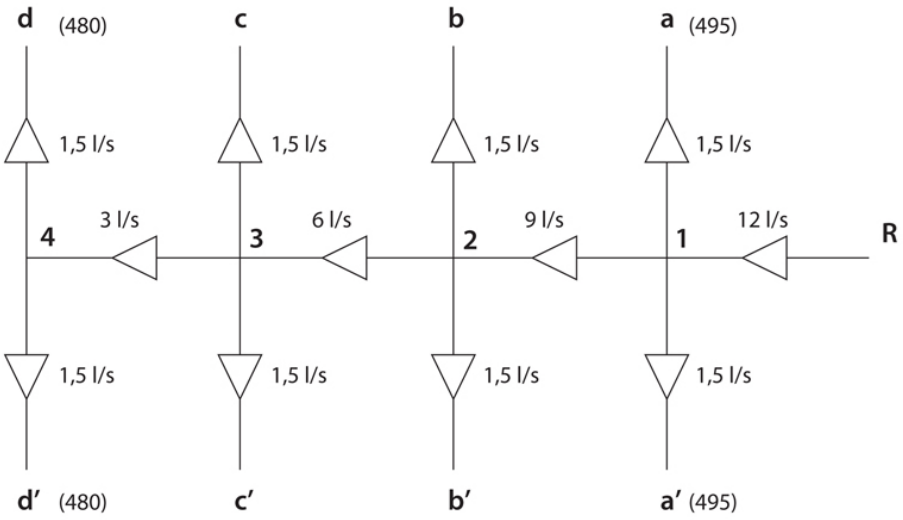
Por fim, o resultado obtido é que a vazão volumétrica de água que entra no reservatório foi de 0,027 m^3/s.

4.1.6 Exercício 2 (COUTO, 2018)

Os condutos instalados nos trechos da avenida central não distribuem água. O reservatório foi construído em D, na cota 500 m (solo) e seu nível d'água está na cota 510 m. A declividade do terreno é de 5% ao longo da avenida central. As ruas não apresentam qualquer declividade. O diâmetro dos trechos da avenida central foi fixado em 100 mm. O diâmetro de todos os trechos de distribuição das ruas residenciais foi fixado em 50 mm. O coeficiente de Hazen é C = 130 para todos os condutos. Determine a pressão no ponto, ou pontos, de menor pressão e no ponto, ou pontos, de maior pressão da rede. Nomeie estes pontos. Verifique se estas pressões estão compreendidas entre os limites mínimo (10 mca) e máximo da rede (50 mca). Justifique a escolha dos pontos de maior e menor pressão e apresente os cálculos necessários à escolha. Admita que os

lotes tenham 10 x 30 metros, sendo a testada (largura confrontante com a rua) de 10 metros.

Figura 12 – Sistema residencial de tubulações.



Fonte: Couto (2018).

Conforme os dados apresentados na Tabela 1, O ponto de menor pressão na rede é o nó do reservatório (R), uma vez que a pressão nos ramos sempre tende a ser menor que a pressão na rua principal, por conta das perdas extensas de carga. Desta forma, o ponto de maior pressão é no nó 4, uma vez que a sua pressão é a maior dentre os nós da rua principal.

4.1.7 Resultados

Os resultados dos problemas selecionados são apresentados abaixo, nos Quadros 1 e 1.

Quadro 1 – Resultados dos problemas selecionados de White (2018).

Exercício	Resultado de Referência (m^3/s)	Resultado do Conduit Forge (m^3/s)	Resultado do PipeFlow Expert (m^3/s)
P6.55	0,0011	0,001	0,001
P6.103	0,00195	0,00204	0,00194
P6.106	0,00084	0,00084	0,00084
P6.108	0,0136	0,0129	0,0129
P6.116	0,027	0,0268	0,0269

Fonte: Adaptado de (ÇENGEL; CIMBALA, 2015).

Tabela 1 – Resultados do exercício 2.

Nó	Resultado de Referência (Pa)	Resultado do *Conduit Forge* (Pa)	Erro Relativo do *Conduit Forge* (%)
R	98000	97925	0,08
1	120834	116232	3,81
2	154448	147737	4,35
3	196196	188765	3,79
4	243236	235630	3,13

Fonte: o autor, Couto(2018).

4.2 DISCUSSÕES

Abaixo, na Tabela 2, são apresentados os valores calculados e os valores de referência dos sistemas analisados.

Tabela 2 – Comparação dos resultados calculados e exatos.

Exercício	Resultado de Referência (m^3/s)	Erro Relativo do *Conduit Forge*(%)	Erro Relativo do Pipe-Flow Expert(%)
P6.55	0,0011	9,09	9,09
P6.103	0,00195	4,62	0,51
P6.106	0,00084	0	0
P6.108	0,0136	5,15	5,15
P6.116	0,027	0,74	0,74

Fonte: o autor, White (2018).

Nos exercícios selecionados de White (2018), os resultados calculados no Conduit Forge são, na maioria, coincidentes com os dados calculados no PipeFlow Expert.

O que parece ser a exceção a essa tendência é o caso do exercício P6.103, no qual o Conduit Forge teve um erro relativo (4,62 %) consideravelmente maior que o erro observado no PipeFlow Expert (0,51%). Este comportamento pode ser explicado pelas propriedades do método de Newton-Raphson, que é o algoritmo de solução numérica usado no pacote desenvolvido.

Por definição, este algoritmo permite que todas as variáveis calculadas convirjam tanto em valores positivos quanto negativos. Neste caso, apesar da proximidade da vazão calculada com o valor de referência, alguns valores de pressão ao longo do sistema avaliado tem valores negativos. Portanto, é possível que o valor de vazão volumétrica calculado teve a sua convergência afetada pelos valores de pressão. Uma explicação que corrobora esta hipótese é que o método de Newton-Raphson encontra uma dentre múltiplas soluções possíveis para o sistema de equações, de forma que, é possível que a solução encontrada não seja, de fato, a solução exata (ou condizente com o comportamento físico real do sistema).

Entretanto, é possível “forçar” a convergência de valores positivos de pressão, através de algumas mudanças de variáveis, para que os termos de pressão sejam substituídos por uma função exponencial dependente de uma variável auxiliar arbitrária (como apresentado na Equação 4). Com esta estratégia, ainda que o valor de U encontrado seja negativo, é impossível que o valor de pressão associado a ele seja negativo.

$$P_i = f(u) = e^u$$

$$P_i = \begin{cases} 0 < f < 1 & \text{se} \quad u < 0 \\ f = 1 & \text{se} \quad u = 0 \\ f > 0 & \text{se} \quad u > 0 \end{cases}$$

Equação 4

No exercício 2, os resultados calculados se mantiveram com erros próximos de 4% em relação aos valores de referência. Esta diferença pode ser explicada pelo uso de diferentes correlações de perda de carga entre os dois resultados comparados. Couto (2018) utiliza a equação de Hazen-Williams, e o Conduit Forge utiliza a equação de Darcy-Weisbach.

5 CONSIDERAÇÕES FINAIS

Ao longo do desenvolvimento deste trabalho, foram criadas classes e métodos que abstraem o comportamento de sistemas de tubulação reais de forma virtual. O SymPy foi uma ferramenta fundamental para tais tarefas, por conta do suporte à criação de funções definidas por partes, além dos *solvers* disponíveis.

Os componentes de tubulação foram todos projetados para que o seu uso seja eficiente, e de uma suave curva de aprendizado, portanto, a implementação das conexões entre componentes foi feita com a criação de métodos *dunder*. Estes métodos foram as ferramentas chave na simplificação dos comandos que o usuário deve realizar para criar juntas.

O gerenciamento das posições das aberturas no espaço foi organizado usando somente as funções matemáticas nativas da linguagem Python, dada a simplicidade dos cálculos envolvidos nesta tarefa. Uma vez que o calculo das posições dos componentes no espaço 2D é uma tarefa menos complexa que as equações do escoamento, o uso de funções matemáticas nativas de Python tornou o código mais simples, robusto e legível.

A criação de gráficos direcionados para representar as redes de tubulação tornou possível várias checagens de redundância lógica e formatos de plotagem gráfica dos resultados.

O pacote implementado se mostrou promissor como ferramenta didática, por conta da proximidade dos resultados calculados com os resultados de referência. Adicionalmente, as funções de segurança, redundância e plotagem oferecem dados suficientes para que os usuários avaliem e compreendam o comportamento dos sistemas analisados.

Por tanto, uma vez que este pacote cumpre os objetivos definidos o mesmo será disponibilizado para download através da página PyPI[1] (o que torna possível instalá-la pelo pip) e no repositório oficial do projeto[2].

Durante o desenvolvimento deste trabalho, foram observados alguns pontos de possíveis melhorias, dentre os principais estão: a criação de variáveis secundárias para representar variáveis de pressão (para forçar a convergência para valores positivos de pressão), otimização das verificações lógicas da coleta de equações,

1 Acesso em: <https://pypi.org/project/conduit-forge/>.

2 Acesso em: <https://github.com/Fugueth/Conduit_Forge>.

criação de classes para representar turbinas, criação de novas classes para representar os diferentes tipos de bombas, atualização da classe Current para comportar fluidos multicomponente, criação de uma interface gráfica interativa para projeto dos sistemas e visualização de resultados.

REFERÊNCIAS

[a]BELL, C. Et al. 2016-2024. **Thermo: Chemical properties component of Chemical Engineering Design Library (ChEDL)**. Disponível em: <https://github.com/CalebBell/thermo>. Acesso em: 08 nov. 2025.

[b]BELL, C. Et al. 2016-2024. **Fluids: Fluid dynamics component of Chemical Engineering Design Library (ChEDL)**. Disponível em: <https://github.com/CalebBell/thermo>. Acesso em: 15 abr. 2026.

BISTAFA, Sylvio R. **Mecânica dos fluidos**. São Paulo: Editora Blucher, 2017. p.20. ISBN 9788521210337.

ÇENGEL, Yunus A.; BOLES, Michael A. **Termodinâmica**. 7. ed. Porto Alegre: Bookman, 2013. p.219. ISBN 9788580552010.

ÇENGEL, Yunus A.; CIMBALA, John M. **Mecânica dos fluidos**. 3. ed. Porto Alegre: AMGH, 2015. ISBN 9788580554915.

COUTO, Luiz M. **Hidráulica na Prática**. Rio de Janeiro: GEN LTC, 2018. p.4. ISBN 9788595153202.

CHRISTIAENS, T. Python HVAC. 2023. Disponível em: <https://github.com/TomLXXVI/python-hvac>. Acesso em:15 abr. 2026.

ELGER, Donald F. Mecânica dos Fluidos para Engenharia, 11ª edição. Rio de Janeiro: LTC, 2019. ISBN 9788521636168.

United States Environmental Protection Agency (EPA). **EPANET: Application for Modeling Drinking Water Distribution Systems**. 2025. Disponível em: <https://www.epa.gov/water-research/epanet>. Acesso em: 16 abr. 2026.

FREITAS, Raphael O.; CORRÊA, Rejane I L.; VAZ, Patrícia M S. Método de Newton-

Raphson. In: **Cálculo numérico**. Porto Alegre: SAGAH, 2019. p.61-76. ISBN 9788595029453.

HAGBERG, Aric A.; SCHULT, Daniel A.; SWART, Pieter J. Exploring network structure, dynamics, and function using NetworkX. In **Proceedings of the 7th Python in Science Conference (SciPy2008)**. v. [s.v.]. n.[s.n.]. p. 11–15, 2008.

HOUGHTALEN, Robert J.; HWANG, Ned H. C.; AKAN, A. O. [tradução: Luciana Teixeira] **Engenharia hidráulica**. In: Tubulações e redes de tubos. São Paulo: Pearson Education do Brasil, 2012. p.54-90.

HUNTER, John. D. **Matplotlib: A 2D Graphics Environment**. Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.

JUPYTER TEAM. **Jupyter Notebook Documentation**. [s.d.].Disponível em: <https://jupyter-notebook.readthedocs.io/en/latest/>. Acesso em: 03 de nov. 2025.

KAKOLAKI, N. K. **A Comparative Analysis of Identifier Schemes: UUIDv4, UUIDv7, and ULID for Distributed Systems**. 10 set. 2025. Cornell University. Disponível em: <https://arxiv.org/abs/2509.08969v1>. Acesso em: 08 nov. 2025.

KLUYVER, Thomas; RAGAN-KELLEY, Benjamin; PÉREZ, Fernando. Jupyter Notebooks – a publishing format for reproducible computational workflows. In: **Positioning and Power in Academic Publishing: Players, Agents and Agendas**. [s.v.]. p. 87-90. 2016.

LEACH, P.; MEALLING, M.; SALZ, R.. **RFC 4122: A Universally Unique IDentifier (UUID) URN Namespace**. 01 jul.2005. Disponível em: <https://dl.acm.org/doi/abs/10.17487/rfc4122>. Acesso em: 08 nov. 2025.

LOHMEIER, D.; CRONBACH, D.; DRAUZ, S.R. et al. Pandapipes: An Open-Source Piping Grid Calculation Package for Multi-Energy Grid Simulations. **Sustainability** 2020, 12, 9899.

MALISKA, Clovis R. **Transferência de Calor e Mecânica dos Fluídos Computacional**, 2. ed. Rio de Janeiro: LTC, 2004. p.1. ISBN 9788521633365.

MCKINNEY, Wes. Data Structures for Statistical Computing in Python. In: **Procedings of the 9th Python in Science Conference**. 2010. v.445. p. 56-61. Disponível em: <https://pandas.pydata.org/about/citing.html>. Acesso em: 03 de nov 2025.

MEURER, A.; SMITH, C. P.; PAPROCKI, M. et al. **SymPy: symbolic computing in Python**. 2017. PeerJ Computer Science 3:e103. Disponível em: <https://doi.org/10.7717/peerj-cs.103>. Acesso em: 03 de nov 2025.

MOLS, B. **25 Years of Python at CWI**. 4 mar. 2015. Disponível em: <https://www.cwi.nl/en/news/25-years-of-python-at-cwi/>. Acesso em: 27 de set. 2025.

OLIPHANT, Travis, E. **Guide to NumPy**. 2006. Disponível em: <https://archive.org/details/NumPyBook>. Acesso em: 03 de nov 2025.

[a]PYTHON SOFTWARE FOUNDATION. **What Is Python? Executive Summary**. Disponível em: <https://www.python.org/doc/essays/blurb/>. Acesso em: 27 de out. 2025.

[b]PYTHON SOFTWARE FOUNDATION. **Applications for Python**. Disponível em: <https://www.python.org/about/apps/>.Acesso em: 27 de out. 2025.

[c]PYTHON SOFTWARE FOUNDATION. **Encontre, instale e publique pacotes Python com o Python Package Index**. Disponível em: <https://pypi.org/>. Acesso em: 6 out. 2025.

[d]PYTHON SOFTWARE FOUNDATION. **Glossary**. Disponível em: <https://pypi.org/>. Acesso em: 6 out. 2025.

[e]PYTHON SOFTWARE FOUNDATION. **Installing Packages**. Disponível em: <https://packaging.python.org/en/latest/tutorials/installing-packages/>. Acesso em: 6 out. 2025.

R. ALBERT; A.-L. BARABÁSI. **Statistical mechanics of complex networks. Reviews of Modern Physics**. n. 74. p. 47-97, 2002. Disponível em:

<https://arxiv.org/abs/cond-mat/0106096>. Acesso em: 17 de out. 2025.

SANTOS, Marcela G.; SARAIVA, Maurício O.; FÁTIMA, Priscila G. **Linguagem de programação**. Porto Alegre: SAGAH, 2018. ISBN 9788595024984.

SIRUANA, A. G.; ESCAMILLA, A. F. Novel Spreadsheets to Enhance Learning of Rigorous Equilibrium-Based Methods for Multicomponent Spearations. In: **Chemical Engineering Education, Class and Home Problems (CHP).** v. 56, n. 2, spring 2022. Disponível em: <https://journals.flvc.org/cee/article/view/128971>. Acesso em: 29 de out 2025.

STAMMITTI, A. **Spreadsheets for assisting Transport Phenomena Laboratory experiences**. Education for Chemical Engineers, v. 8 n. 2. 2013. ISSN: 1749-7728. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1749772813000067>. Acesso em: 29 out. 2025.

VIRTANEN, P., GOMMERS, R., OLIPHANT, T.E. et al. **SciPy 1.0: Fundamental algorithms for scientific computing in Python**. Nature Methods. v. 17, p. 261–272 (2020). https://doi.org/10.1038/s41592-019-0686-2.

WHITE, Frank M. **Mecânica dos fluidos**. 8. ed. Porto Alegre: ArtMed, 2018. ISBN 9788580556070

ZEGHADNIA, L.; LOUP Robert J.; ACHOUR, B. **Explicit solutions for turbulent flow friction factor: A review, assessment and approaches classification**. Ain Shams Engineering Journal. v.10, n.1. p.243-251, 2019.

APÊNDICES

APÊNDICE A – O MÓDULO *GEOMETRY HANDLER*

Este módulo gerencia todas as operações sobre a posição espacial de cada abertura dentro de um dado sistema de tubulações. As suas principais funcionalidades são a criação dos objetos *Point* e *Vector*, que serão usados para representar a posição das aberturas dentro de um Mosaico.

Estes objetos contam com métodos auxiliares úteis ao calculo vetorial como o cálculo da magnitude, o ângulo entre esses vetores e os eixos cartesianos (x, y, z) e etc.

Conforme descrito previamente, ainda que a livraria só manipule a posição das aberturas nas direções x e z, o módulo *Geometry Handler* conta com o suporte para a adição de mais uma coordenada, a fim de facilitar as possíveis expansões futuras que podem da livraria.

APÊNDICE B – MODELAGEM MATEMÁTICA (CONSERVAÇÃO DE MASSA)

A partir da modelagem de cada componente de tubulação como um volume de controle, é possível escrever a equação geral da conservação de massa para volumes de controle (Equação 5) (ÇENGEL; BOLES, 2013).

$$\sum_{e} \dot{m} - \sum_{s} \dot{m} = \frac{d(m_{VC})}{dt} \qquad \text{Equação 5}$$

Aplicando a definição de escoamento em regime permanente (Equação 6), a expressão de conservação de massa pode ser simplificada para a forma apresentada na Equação 7.

$$\frac{d(m_{vc})}{dt} = 0 \qquad \text{Equação 6}$$

$$\sum_{e} \dot{m} - \sum_{s} \dot{m} = 0 \quad \rightarrow \quad \sum_{e} \dot{m} = \sum_{s} \dot{m} \qquad \text{Equação 7}$$

Em seguida, foi aplicada a condição de escoamento incompressível (Equação 22) sobre a equação de continuidade (Equação 8) (ÇENGEL; CIMBALA, 2015).

$$\dot{m} = \rho V A \qquad \text{Equação 8}$$

Na Equação 8, ρ é a densidade do fluido no volume de controle (medida em kg/m^3), que pode ser aproximada como uma constante em cada um dos pontos internos e na superfície do volume de controle ($\rho \sim cte$, $\frac{d(\rho)}{dr} = 0$, onde r é um ponto dentro ou na superfície do volume de controle). V é a velocidade média do escoamento (medida em m/s) em uma dada seção transversal do mesmo, em qualquer posição r dentro ou na superfície do volume de controle, que será denotada como V ao longo deste trabalho. Por fim, A é a área da seção transversal do escoamento (medida em m^2, no SI) em um determinado ponto r dentro do volume de controle.

$$\dot{m}_e = \rho_e V_e A_e \qquad \text{Equação 9}$$

$$\dot{m}_s = \rho_s V_s A_s \qquad \text{Equação 10}$$

$$\rho_e = \rho_s \qquad \text{Equação 11}$$

$$\sum \rho_i V_i A_i = \sum \rho V_i A_i = \rho \sum V_i A_i \qquad \text{Equação 12}$$

$$\rho \sum_e V A = \rho \sum_s V A \qquad \text{Equação 13}$$

Na Equação 14, denota-se o produto V A como $\dot{V}$, que é a vazão volumétrica, medida em m^3/s (ÇENGEL; CIMBALA, 2015).

$$\dot{V} = V A \qquad \text{Equação 14}$$

$$A = \pi r^2, \; r = \frac{D}{2} \;\rightarrow\; A = \frac{\pi D^2}{4} \qquad \text{Equação 15}$$

APÊNDICE C – MODELAGEM MATEMÁTICA (CONSERVAÇÃO DE ENERGIA)

À Equação 17 aplica-se a limitação de escoamento adiabático (sem

transferência de calor entre o interior e o exterior do volume de controle) à Primeira Lei da Termodinâmica para um volume de controle (Eq. 16) (ÇENGEL; BOLES, 2013).

$$\dot{Q}-\dot{W}=\sum_{s}\dot{m}\left(h+\frac{v^2}{2}+gz\right)-\sum_{e}\dot{m}\left(h+\frac{v^2}{2}+gz\right) \qquad \text{Equação 16}$$

Destaca-se que, na Equação 16, o termo $\dot{W}$ representa o trabalho mecânico realizado pelo volume de controle sobre a vizinhança (ÇENGEL; BOLES, 2013). No entanto, o escopo do presente trabalho só comporta equipamentos de entrada de trabalho mecânico no escoamento (como bombas e/ou ventiladores). Portanto, o termo $\dot{W}$ sempre terá um valor negativo, ou será igual a zero (para os componentes de tubulação diferentes das bombas). Logo, o sinal deste termo pode ser mudado para positivo, uma vez que nesta forma o termo da taxa de trabalho indica o trabalho que entra no volume de controle.

Acrescenta-se que, como os sistemas de tubulação no escopo deste trabalho são adiabáticos, o termo de calor transferido do volume de controle para a vizinhança é zero ($\dot{Q}=0$).

$$\dot{W}+\sum_{e}\dot{m}\left(h+\frac{v^2}{2}+gz\right)=\sum_{s}\dot{m}\left(h+\frac{v^2}{2}+gz\right) \qquad \text{Equação 17}$$

Desta forma, é possível determinar que as transformações e o transporte de energia no volume de controle ocorrem somente na forma de energia mecânica.

O engenheiro Osborne Reynolds, na década de 1880, estudou os diferentes regimes de escoamento de fluidos, e os caracterizou em três categorias: Laminar, Transicional e Turbulento. Em sua pesquisa, Reynolds observou que é possível usar a razão das forças inerciais e viscosas de um escoamento para caracterizar em qual regime este escoamento está (ÇENGEL; CIMBALA, 2015). Esta razão (apresentada na Equação 18) foi chamada de número de Reynolds (que é um número adimensional), em homenagem do engenheiro.

$$\mathrm{Re} = \frac{\text{Forças de Inerciais}}{\text{Forças Viscosas}} = \frac{\rho V L_{car}}{\mu}$$ Equação 18

Nesta equação, ρ é a densidade do fluido, V é a velocidade média do escoamento, L_{car} é o comprimento característico da seção de escoamento analisado (em tubos cilíndricos, L_{car} é o diâmetro da tubulação, D) e μ é a viscosidade absoluta do fluido (medida em Pa*s) (ÇENGEL; CIMBALA, 2015).

Segundo definido por Çengel e Cimbala (2015), o escoamento laminar é todo aquele no qual Re <= 2300, e o escoamento turbulento Re >= 4000 (considerando que os escoamentos ocorram em tubos de seção transversal circular). Entre estes dois regimes, existe o escoamento em regime de transição, que apresenta regiões turbulentas e regiões laminares (ÇENGEL; CIMBALA, 2015).

Como todas as equações do sistema não linear devem ser variáveis de pressão ou de vazão volumétrica, é necessário que as equações sejam manipuladas do seu formato original para o formato adequado ao *solver*.

Portanto, a partir da Equação 18 é possível rearranjar a equação do número de Reynolds para uma forma na qual Re = f($\dot{V}$), em um tubo de seção transversal circular, conforme apresentado na Equação 20.

$$V = \frac{\dot{V}}{A}, \quad L = D, \quad V = \frac{4\dot{V}}{\pi D^2}$$
$$\mathrm{Re} = \frac{\rho D}{\mu} \cdot \left(\frac{4\dot{V}}{\pi D^2}\right) \quad \rightarrow \quad \mathrm{Re} = \frac{4\rho}{\mu \pi D}\dot{V}$$ Equação 19

$$C_1 = \frac{4\rho}{\mu \pi D} = cte \quad \rightarrow \quad \mathrm{Re} = C_1 \dot{V}$$ Equação 20

Apresenta-se na Equação 21, a expressão da conservação da energia mecânica usada, que é descrita como um balanço entre duas aberturas de um dado componente. Esta relação é compatível com escoamentos em regime permanente, adiabáticos e incompressíveis, em diferentes regimes, e em escoamentos nos quais a viscosidade não é desprezível (ÇENGEL; CIMBALA, 2015).

$$\frac{P_1}{\rho g}+\frac{V_1^2}{2g}+z_1+h_b=\frac{P_2}{\rho g}+\frac{V_2^2}{2g}+z_2+h_L \quad \text{Equação 21}$$

Na equação acima, os termos P_i são valores de pressão manométrica (medida em Pa) em um dado ponto do escoamento (neste caso, em dois pontos arbitrários 1 e 2); g é a aceleração da gravidade no ponto do escoamento analisado (medida em m/s^2); z é a elevação vertical em um dado ponto do sistema de tubulações (medido em m); h_b é o termo de carga de energia mecânica, por conta de uma bomba no sistema de tubulações (medido em m); h_L é o termo de perda de energia mecânica dentro do sistema de tubulações (medido em m).

É possível descrever o termo h_b como uma função da potência de trabalho mecânico de eixo realizado por um componente sobre o volume de controle, a partir da relação apresentada na Equação 22 (ÇENGEL; CIMBALA, 2015).

$$\dot{W}_{b,e}=\frac{h_{b,e}\ \rho g \dot{V}}{\eta_{b,e}} \quad \text{Equação 22}$$

De maneira semelhante à manipulação realizada sobre a forma algébrica do número de Reynolds, é possível rearranjar a Equação 23 para que ela esteja na forma $h_b=t(\dot{V})$, conforme apresentado na Equação 24

$$h_b=\frac{\dot{W}_{b,e}\ \eta_{b,e}}{\rho g \dot{V}} \rightarrow h_b=\frac{\dot{W}_{b,e}\ \eta_{b,e}}{\rho g}\cdot\frac{1}{\dot{V}} \quad \text{Equação 23}$$

$$C_2=\frac{\dot{W}_{b,e}\ \eta_{b,e}}{\rho g}=cte \rightarrow h_b=U\frac{1}{\dot{V}} \quad \text{Equação 24}$$

Em seguida, também é possível rearranjar o segundo termo da equação de conservação de energia do escoamento (Eq. 21), o termo que leva em consideração a velocidade do escoamento em um ponto (como apresentado a seguir, na Equação 25), que é a chamada Carga Hidrodinâmica (C_H).

$$C_H = \frac{V_i^2}{2g}$$ Equação 25

Levado em consideração as definições de $\dot{V}$ e A (apresentadas nas Equações 25 e 26), é possível fazer o rearranjo deste termo como apresentado a seguir, na Equação 27.

$$\frac{V_i^2}{2g} = \frac{1}{2g}\left(\frac{4\dot{V}}{\pi D^2}\right)^2 = \left(\frac{8}{g\pi^2 D^4}\right)\dot{V}^2$$ Equação 26

$$C_3 = \frac{8}{g\pi^2 D^4} = cte \quad \rightarrow \quad \frac{V_i^2}{2g} = C_3\dot{V}^2$$ Equação 27

O termo h_L considera as perdas de energia mecânica dissipada por efeitos da viscosidade (chamadas comumente de perdas maiores, já que elas ocorrem ao longo de uma seção de tubulação). Este termo também representa as perdas de energia pela mudança de direção e separação do escoamento (chamadas comumente de perdas menores, ou localizadas, já que elas ocorrem em conexões, joelhos, juntas, bifurcações e demais componentes de tubulação (ÇENGEL; CIMBALA, 2015).

Conforme apresentado por Çengel e Cimbala (p.374, 2015), as perdas menores são fenômenos de elevada complexidade, e elas geralmente são avaliadas utilizando uma definição empírica (Equação 28), que utiliza de um coeficiente de perda localizada (K_L), que é fornecido pelos fabricantes do componente em questão.

$$K_L = \frac{h_L}{V^2/2g} = \frac{2g}{V^2}h_L \quad \rightarrow \quad h_L = \frac{K_L V^2}{2g}$$ Equação 28

Em componentes de tubulação nos quais ocorre a perda de carga localizada, o termo de h_L pode ser reescrito como uma função dependente da vazão volumétrica ($\dot{V}$), pela equação de Darcy-Weisbach simplificada.

$$h_L = \frac{K_L}{2g}\left(\frac{4\dot{V}}{\pi D^2}\right)^2 \quad \rightarrow \quad h_L = \frac{8K_L}{g\pi^2 D^4}\dot{V}^2 \qquad \text{Equação 29}$$

$$C_4 = \frac{8K_L}{g\pi^2 D^4} = cte \quad \rightarrow \quad h_L = C_4 \dot{V}^2 \qquad \text{Equação 30}$$

Nos componentes de tubulação nos quais ocorrem as perdas maiores (gerada por efeitos viscosos do escoamento), o termo h_L pode ser escrito na forma apresentada a seguir (Eq. 31), que é uma simplificação da equação de Darcy-Weisbach (ÇENGEL; CIMBALA, 2015). A partir desta equação, usando as considerações apresentadas nas equações 25, 26 e 19, este termo pode ser reescrito como uma função dependente da vazão volumétrica ($\dot{V}$), em algum ponto do componente em questão (Eq. 32).

$$h_L = \frac{\Delta P_L}{\rho g} = f\frac{L}{D}\cdot\frac{V^2}{2g} \quad , \quad f_{laminar} = \frac{64}{\mathrm{Re}} \qquad \text{Equação 31}$$

$$h_L = \frac{\Delta P_L}{\rho g} = \frac{16 f L}{2 g D^5 \pi^2}\cdot\dot{V}^2 \rightarrow C_5 = \frac{8 f L}{g D^5 \pi^2} = cte \rightarrow h_L = C_5 \cdot \dot{V}^2 \qquad \text{Equação 32}$$

No caso de escoamento turbulento em tubos fechados, o cálculo do fator de atrito de Darcy-Weisbach (*f*) é feito utilizando correlações empíricas, devido à complexidade deste regime de escoamento (ÇENGEL; CIMBALA, 2015).

Existem diversas correlações do fator de atrito, algumas delas são escritas como funções implícitas, que devem convergir em um valor numérico após uma série de iterações. No entanto, também existem correlações explícitas, com as quais é possível calcular diretamente o fator de atrito.

Uma destas correlações é a correlação de Swamee-Jain, que foi escolhida como padrão para todos os casos e exemplos avaliados neste trabalho. Esta correlação foi escolhida por conta da sua relativa simplicidade matemática, e por poupar recursos computacionais.

É importante notar que cada correlação do fator de atrito possui uma faixa de valores numéricos nos quais elas podem ser aplicadas (geralmente do número de

Reynolds, ou da rugosidade relativa do tubo). No caso da correlação de Swamee-Jain, a restrição está definida em uma faixa do número de Reynolds, e também em uma faixa da rugosidade relativa do tubo (ϵ/D), como apresentado a seguir, no Quadro 2.

Quadro 2 – Correlações do fator de atrito de Darcy-Weisbach (f) e suas propriedades.

Nome	Forma matemática	Tipo	Faixa de Aplicação
[a, b] Colebrook-White	$\frac{1}{\sqrt{f}} = -2{,}0 \log\left(\frac{\epsilon/D}{3{,}7} + \frac{2{,}51}{\mathrm{Re}\sqrt{f}}\right)$	Implícita	$5{,}235 \cdot 10^3 \leq \mathrm{Re} \leq 10^8$ $0 \leq \epsilon/D \leq 5 \cdot 10^{-2}$
[c] H. Blasius	$f = \frac{0{,}316}{\mathrm{Re}^{1/4}}$	Explicita	$4000 < \mathrm{Re} < 10^5$ Tubos lisos
[a, b] S. E. Haaland	$\frac{1}{\sqrt{f}} \simeq -1{,}8 \log\left[\frac{6{,}9}{\mathrm{Re}} + \left(\frac{\epsilon/D}{3{,}7}\right)^{1{,}11}\right]$	Explicita	$4 \cdot 10^3 \leq \mathrm{Re} \leq 10^8$ $10^{-6} \leq \epsilon/D \leq 10^{-2}$
[d] Swamee-Jain	$f = 0{,}25 \cdot \left(\log\left(\frac{\epsilon/D}{3{,}7} + \frac{5{,}74}{\mathrm{Re}^{0{,}9}}\right)\right)^{-2}$	Explicita	$5000 < \mathrm{Re} < 10^8$ $10^{-6} < \epsilon/D < 10^{-2}$

Fonte: Adaptado de [a](ZEGHADNIA; LOUP; ACHOUR, 2019), [b](ÇENGEL; CIMBALA, 2015), [c](WHITE, 2018) e [d](ELGER, 2019).

A seguir, no Quadro 3, estão apresentadas quais são as equações usadas em cada um dos diferentes tipos de componentes de tubulação analisados neste trabalho.

Quadro 3 – Diferentes formas dos termos de trabalho de bomba e perda de carga nos componentes de tubulação avaliados neste trabalho.

Tipo de componente/ tipo de carga	h_b	h_L
Componente de Perdas Menores	$h_b = 0$	$h_L = \frac{8 K_L}{g \pi^2 D^4} \dot{V}^2$
Componente de Perdas Maiores	$h_b = 0$	$h_L = \frac{16 f L}{2 g D^5 \pi^2} \cdot \dot{V}^2$
Componente de Entrada de Trabalho de eixo	$h_b = \frac{\dot{W}_{bomba,eixo} \quad \eta_{bomba,eixo}}{\rho g} \cdot \frac{1}{\dot{V}}$	$h_L = 0$

Fonte: Adaptado de (ÇENGEL; CIMBALA, 2015).

APÊNDICE D – CONFIGURAÇÕES E ESTADOS DO SISTEMA DE EQUAÇÕES

Por analogia a sistemas de equações algébricas, existem três estados nos quais um sistema de equações está, são eles: sistema determinado, sobredeterminado e impossível.

Considerando um componente individual genérico, com um número α de aberturas, o *Conduit Forge* aplica a ele a equação da continuidade e a equação da conservação de energia (que são armazenadas em uma lista, e usadas como argumentos do solver). Como a equação de energia (Eq. 2) leva em conta a conservação de energia entre dois pontos diferentes de um dado componente de tubulação, em componentes com mais de duas aberturas úteis, é necessário aplicar a equação da conservação da energia mais de uma vez para que o sistema de equações possa ser resolvido.

Por exemplo, no caso dos conectores T, que tem 3 aberturas na fronteira do volume de controle com a vizinhança, é necessário aplicar a equação da conservação da energia sobre dois pares de aberturas. Isto se deve ao fato de que, apesar de ser possível aplicar a equação da conservação de energia em três pares de aberturas de um T, a terceira equação faria com que o número de equações no sistema seja maior que o número de variáveis, o que tornaria o sistema sobredeterminado.

Portanto, a regra geral para componentes com duas ou três aberturas funcionais (que podem fazer ligações com outros componentes e correntes) é que o número de equações de energia é igual a $\beta = \alpha - 1$.

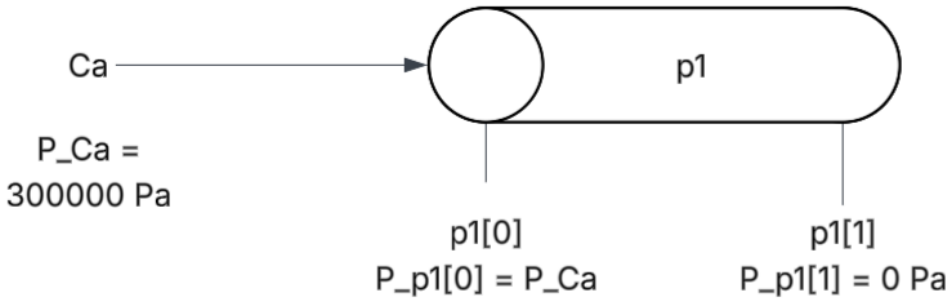
É possível generalizar ainda mais a análise acima, extrapolando essas

definições para um sistema de tubulação inteiro. No caso de um sistema de tubulações, em semelhança com um componente genérico, podem existir múltiplas aberturas, onde múltiplas delas podem ser aberturas de entrada ou saída de fluido, em relação ao sistema.

Partindo destas observações, cabe salientar que todas as aberturas de entrada e/ou saída precisam ter a pressão ou a vazão volumétrica definidas. Por isso, os usuários devem definir a pressão ou a vazão mássica nas correntes de entrada do sistema para que o sistema tenha a possibilidade de solução (além, é claro, da composição química destas correntes). Contudo, é necessário que o sistema tenha, ao menos em uma abertura, um valor de pressão definido. Este valor é chamado de “referencial de pressão”, e é fundamental para que o sistema possa ser resolvido.

Tomando como exemplo o sistema de tubulação simples: um único tubo com uma corrente material de entrada (conforme apresentado a seguir, na Figura 13); é possível analisar estes comportamentos de maneira mais detalhada.

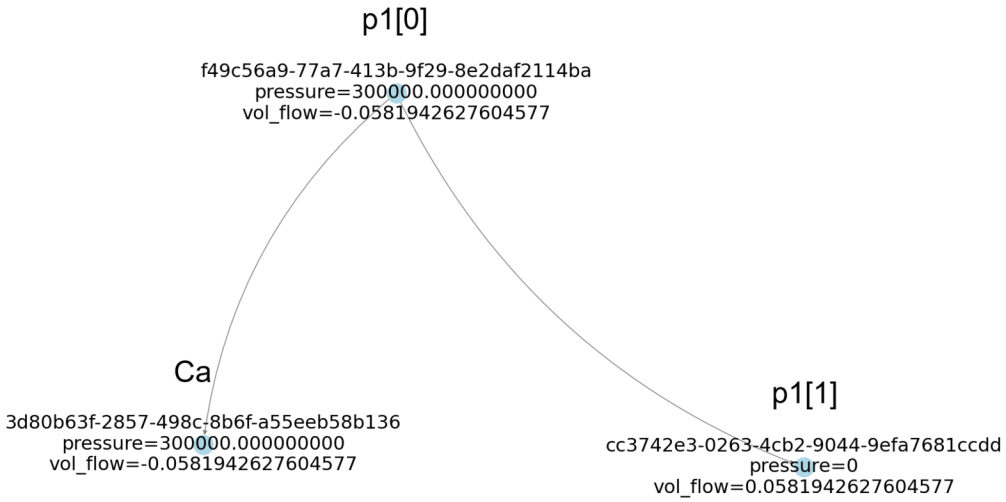
Figura 13 – Diagrama de um sistema de tubulação simples.



Fonte: o autor.

O gráfico direcionado criado para este mosaico possui 3 nós, ligados em série (conforme apresentado na Figura 14).

Figura 14 – Gráfico direcionado de um sistema simples de tubulação gerado pelo *Conduit Forge*.



Fonte: o autor.

Destaca-se que as legendas Ca, p1[0] e p1[1] não são criadas automaticamente pelo pacote, uma vez que os nomes Ca e p1 só podem ser usados como representação para o usuário, e não podem ser acessados por nenhum dos objetos *Component*, *Mosaic*, *Current* e *Port*, por conta da limitação na linguagem Python. Logo, estes nomes só podem ser adicionados à imagem do gráfico direcionado como elemento didático, para os fins deste trabalho.

As incógnitas deste sistema são: Vazão volumétrica de Ca, Vazão volumétrica na abertura de índice 0 do tubo 1 e vazão volumétrica na abertura de índice 1 do tubo 1.

Com a plicação da lei da conservação da massa global do sistema, observa-se que as três incógnitas devem ter o mesmo valor numérico ao final da simulação. Como o tubo está na horizontal, ambas as elevações são iguais ($z_{p1[0]} = z_{p1[1]} = 0$). Além destas, como as constantes E são dependentes somente da geometria da abertura na qual esta constante é avaliada, os valores de E também são iguais ($E_{p1[0]} = E_{p1[1]}$).

Apesar destas simplificações, o sistema ainda conta com 6 equações, conforme apresentado a seguir, na Equação 33.

$$\begin{cases} P_{Ca} = 300000\,Pa \\ P_{p1[0]} = P_{Ca} \\ P_{p1[1]} = 0\,Pa \\ \frac{P_{p1[0]}}{\rho \cdot g} + E_{p1[0]} * \dot{V}^2_{p1[0]} + z_{p1}[0] = \frac{P_{p1[1]}}{\rho \cdot g} + E_{p1[1]} * \dot{V}^2_{p1[1]} + z_{p1}[1] + h_L \\ \dot{V}_{p1[0]} = \dot{V}_{Ca} \\ \dot{V}_{p1[0]} = \dot{V}_{p1[1]} \end{cases}$$ Equação 33

Destas seis equações, observa-se que três delas são declarações de valores numéricos para as variáveis do problema. Portanto, estas “equações” acabam não sendo “resolvidas” dentro do solver, já que são consideradas condições iniciais. Logo, todas as suas derivadas parciais são iguais a zero. Desta forma, os seus vetores gradientes são linhas nulas na matriz jacobiana, que são eliminadas antes da inversão.

Rearranjando as equações acima no formato indicado na seção 3.2.1., o sistema ficaria no seguinte formato (Eq. 34).

Equação 34

$$\begin{cases} f_4: \ \frac{P_{p1[0]} - P_{p1[1]}}{\rho \cdot g} + E_{p1[0]} * \dot{V}^2_{p1[0]} - E_{p1[1]} * \dot{V}^2_{p1[1]} + z_{p1[0]} - z_{p1[1]} + h_L = 0 \\ f_5: \ \dot{V}_{p1[0]} - \dot{V}_{Ca} = 0 \\ f_6: \ \dot{V}_{p1[0]} - \dot{V}_{p1[1]} = 0 \end{cases}$$

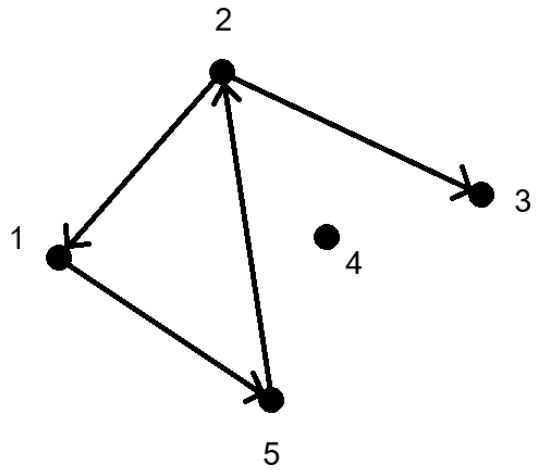
Das três funções restantes, ainda é possível simplificar o sistema para somente uma função, já que as funções f5 e f6 indicam que todas as três variáveis têm o mesmo valor numérico. Portanto, somente a equação f4 precisaria ser passada ao solver para que os cálculos do sistema sejam resolvidos. Apesar disso, as funções de 1 a 6 podem ser passadas como argumento para a função *nsolve* (da livraria SymPy), que é capaz de organizá-las de forma correta e eficiente.

APÊNDICE E – GRÁFICOS DIRECIONADOS

Os gráficos são diagramas que definem uma rede de objetos (representados por nós) e a relação entre cada um dos nós com os demais (representadas por arestas, sejam elas direcionadas ou não) (ALBERT; BARABÁSI, 2002).

Estas entidades matemáticas podem ser usadas para representar diferentes tipos de sistemas (ALBERT; BARABÁSI, 2002). Alguns exemplos práticos são: redes neurais, grupos populacionais, a internet e culturas microbianas (ALBERT; BARABÁSI, 2002). Na Figura 15 está apresentado um exemplo de um gráfico direcionado simples.

Figura 15 – Gráfico direcionado de uma rede simples.



Fonte: Adaptado de (ALBERT; BARABÁSI, 2002).

Conforme Albert e Barabási (tradução nossa, 2002), em termos matemáticos, um gráfico é um par de conjuntos G = {P, E_g}, onde P é um conjunto de pontos (ou vértices/nós) e E_g é um conjunto de arestas (também chamadas de linhas) que conectam dois pontos. Desta forma, o gráfico direcionado apresentado na Figura 15 é representado matematicamente na forma apresentada na Equação 35.

$$G_{dir}=\{P,E_g\},\ P=\{1,2,3,4,5\},\ E_g=\{\{2,1\},\{1,5\},\{5,2\},\{2,3\}\}$$
$$\{2,1\}_{E_g} \neq \{2,1\}_{E_g}$$

Equação 35

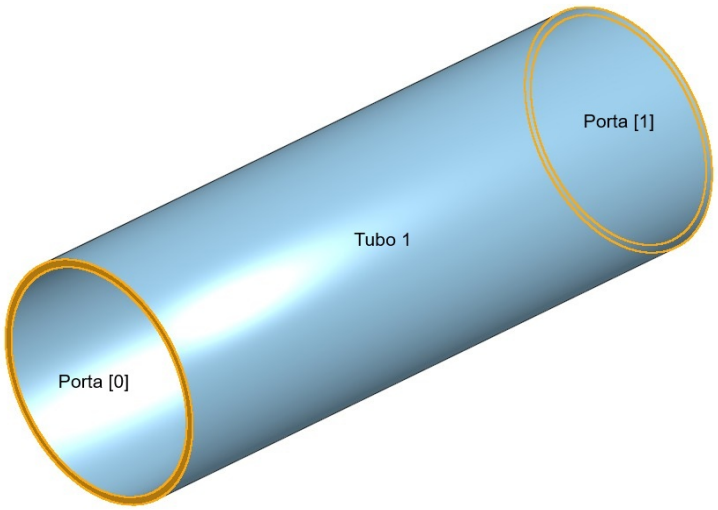
Nos gráficos direcionados, destaca-se que cada aresta tem direcionalidade, ou seja, são influenciadas pela ordem de ligação entre dois pontos, em oposição a gráficos não direcionados (nos quais a ordem dos pontos dentro de uma aresta não alteram o gráfico). Destaca-se que, apesar do ponto 4 não fazer parte de nenhuma aresta, ele ainda está no conjunto P e, portanto, faz parte do gráfico direcionado em questão.

APÊNDICE F – A CLASSE *PORT*

Esta classe armazena dados da geometria, composição química e o estado termodinâmico do escoamento (em valores de pressão e temperatura) na região da abertura.

De maneira geral, a abertura representa toda e qualquer extremidade de um dado componente de tubulação, nas quais ocorre (ou nas quais pode ocorrer) a entrada ou saída de fluido. Por exemplo, em um tubo, só existem duas extremidades abertas, que são nas suas extremidades físicas (como apresentado na Figura 16, abaixo). Portanto, o código foi escrito para que os objetos da classe Pipe tenham somente duas aberturas.

Figura 16 – Representação das aberturas em um tubo cilíndrico.

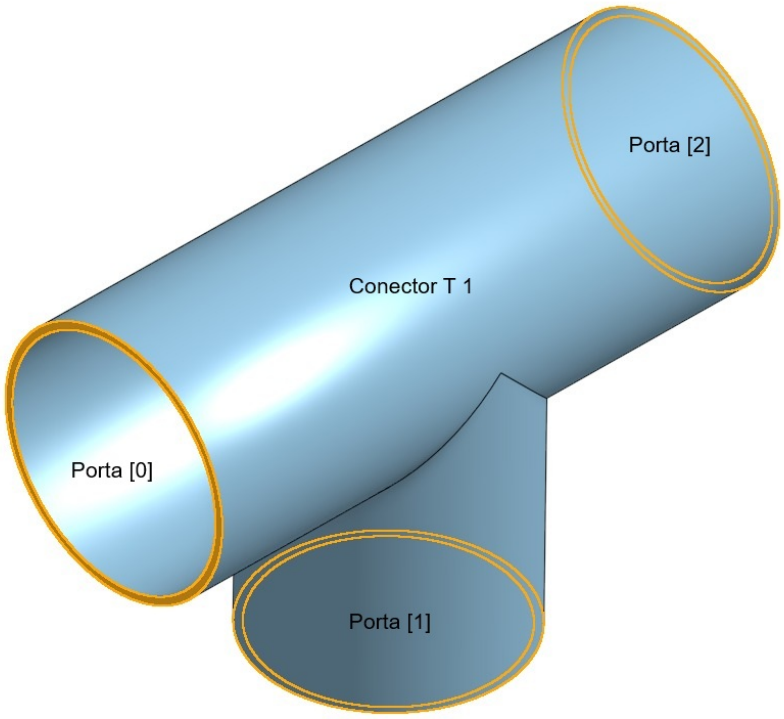


Fonte: o autor.

No caso dos conectores T, as aberturas são cada uma das três

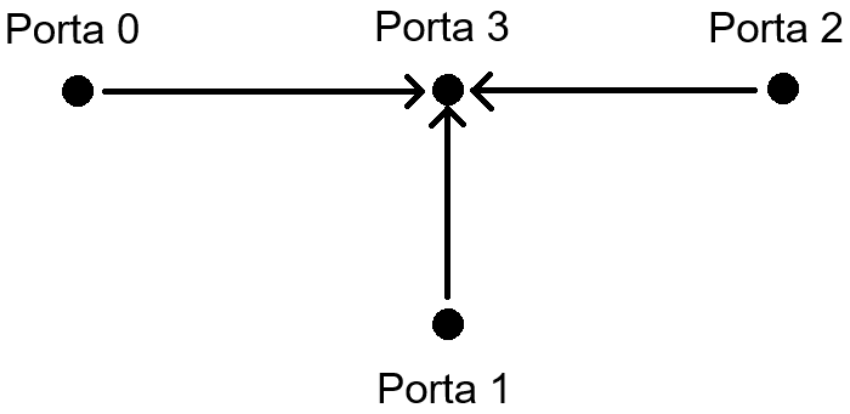
extremidades (como apresentado na Figura 17) do T que podem ser conectadas a qualquer outro componente de tubulação. Contudo, o caso destes conectores é um pouco diferente dos demais, já que este componente possui 4 aberturas, apesar de somente três estarem disponíveis para conexões com outros componentes. Esta quarta abertura faz parte deste componente para servir como “ponto de referência” na criação do gráfico direcionado. No gráfico, esta “abertura” está ligada a cada uma das demais aberturas, conforme apresentado na Figura 18. Portanto, como esta abertura só existe para manter a coesão estrutural do gráfico direcionado, a sua corrente local não possui valores de pressão e vazão volumétrica, uma vez que estas aberturas não tem nenhum dos seus atributos passados como argumento para os métodos de composição das equações de conservação de energia e massa destes componentes e dos demais.

Figura 17 – Representação das aberturas disponíveis para conexão, em um conector T.



Fonte: o autor.

Figura 18 – Representação de todas as aberturas em um conector T.

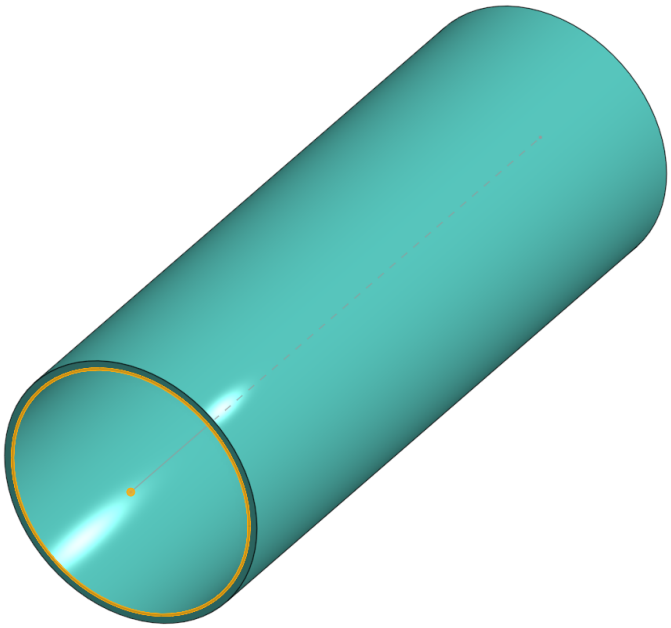


Fonte: o autor.

Em cada um dos objetos *Port*, os dados de escoamento são armazenados em um atributo da *port*, chamado *local_current*, que é um objeto da classe *Current*, e tem todos os atributos típicos de um objeto *current*.

A posição do centro da abertura no espaço (como apresentado na Figura 19) é armazenada por um objeto *Point* do modulo Geometry Handler.

Figura 19 – Seção transversal de um tubo cilíndrico, e o seu centro.



Fonte: o autor.

APÊNDICE G – AS SUBCLASSES DA CLASSE *COMPONENT*

A partir da modelagem matemática dos componentes de tubulação (apresentados na seção 3.2.1 e 3.2.2) é possível dividi-los em três grandes tipos: Componentes de perdas de cargas maiores, componentes de perdas de cargas menores e componentes de entrada de trabalho mecânico de eixo.

A principal diferença entre esses componentes é a forma e o valor dos termos de carga de bomba ($^{h}_{b}$) e perda de carga ($^{h}_{L}$). Portanto, a fim de manter a modularidade e organização do código (o que facilita a ampliação futura da livraria), foram criadas três classes filhas da classe *Component*: *MajorLossComponent*, *MinorLossComponent*, *MechanicalPowerImputComponent*. Cada um dos componentes de tubulação analisados neste trabalho tiveram as suas classes criadas como filhas de uma dessas três classes.

Como por exemplo a classe *Pipe*, que é uma classe filha da classe *MajorLossComponent*; ou a classe *T_connector*, que é uma classe filha da classe MinorLossComponent; e a classe *Pump*, que é filha da classe *MechanicalPowerImputComponent*.

A classe *MajorLossComponent* só possui uma classe filha, a classe *Pipe* (que representa tubos).

No caso da classe Pipe (que é neta da classe Component), destaca-se que a numeração das aberturas apresentadas na Figura 16 indica o índice das aberturas dentro do Tubo 1. A livraria foi organizada desta maneira por que cada uma das aberturas é armazenada em um atributo herdado da classe *Component*: a lista *component_ports*.

Portanto, o índice apresentado na Figura 16 indica qual é a posição de uma data abertura dentro da lista do objeto Componente. Na linguagem Python (assim como em várias outras linguagens) os índices de objetos dentro de listas e alguns outros iteráveis começam em zero (que é o primeiro item da lista).

As classes filhas *MinorLossComponent* são: as classes *Elbow* (que representam cotovelos) e as classes *Fitting* (que representam componentes de acoplamento, como encaixes).

A única classe filha de *MechanicalPowerInputComponent* é *Pump*, uma vez que, dentre o grupo de equipamentos de entrada de trabalho mecânico em um sistema, as bombas são as únicas no escopo deste trabalho.