e-Boletim de Física
International Centre for Physics
Instituto de Física, Universidade de Brasília
Ano 13, 2025 • http://periodicos.unb.br/index.php/e-bfis • eBFIS **XIII** 03-1(01)

Escoamento de fluidos em tubulações: uma introdução prática com o python

Cristian Bárbara Brito and Diogo Rosseti da Silva Mendes
Departamento de Física, Instituto de Ciências Naturais, Universidade Federal de Lavras, Caixa postal 3037, CEP: 37200-900, Lavras, MG, Brasil.

Saulo Luís Lima da Silva
Centro Federal de Educação Tecnológica de Minas Gerais, Avenida Monsenhor Luiz de Gonzaga, 103, CEP: 37250-000, Nepomuceno, MG, Brasil

Angélica Sousa da Mata*
Departamento de Física, Instituto de Ciências Naturais, Universidade Federal de Lavras, Caixa postal 3037, CEP: 37200-900, Lavras, MG, Brasil.

O estudo dos fluidos desempenha um papel fundamental em diversas áreas da engenharia, como a civil, mecânica, química e ambiental, sendo indispensável para compreender processos relacionados ao transporte de líquidos e gases em tubulações, canais e sistemas industriais. A análise de escoamentos está presente tanto em aplicações práticas — como no dimensionamento de redes de abastecimento de água, sistemas de ventilação ou no transporte de petróleo e gás — quanto na formação acadêmica, em disciplinas básicas de mecânica dos fluidos e hidráulica. Entretanto, a complexidade dos fenômenos envolvidos frequentemente inviabiliza a obtenção de soluções analíticas, tornando necessária a utilização de métodos numéricos e ferramentas computacionais. Este estudo destaca as vantagens da adoção do uso da programação, em particular em Python, para a realização de cálculos e obtenção de soluções em problemas relativos ao escoamento de fluidos em sistemas de tubulação. Com o intuito de alcançar uma precisão satisfatória, foram desenvolvidos códigos computacionais que englobam um conjunto abrangente de equações aplicáveis a diversas situações relacionadas ao escoamento em sistemas tubulares. Os exemplos computacionais contemplaram o cálculo do número de Reynolds, a análise das perdas de carga associadas ao atrito e a avaliação da distribuição de pressão ao longo do escoamento, entre outros aspectos de relevância técnica, demonstrando a eficácia e versatilidade da metodologia proposta. Além de sua contribuição técnica, o trabalho apresenta também um caráter didático, podendo ser utilizado como material de apoio em disciplinas de fluidos, auxiliando professores e alunos na compreensão de conceitos fundamentais e na aplicação prática de ferramentas computacionais.

Keywords: Fluidos; Escoamento; Ferramentas Computacionais; Exemplo Didático.

I. INTRODUÇÃO

Fluidos (líquidos e gases) correspondem a um estado da matéria caracterizado pela incapacidade de resistir a tensões tangenciais (tensões de cisalhamento). Assim, qualquer tensão de cisalhamento, por menor que seja, provoca o escoamento do fluido, e esse movimento persiste enquanto a tensão for aplicada. Eles se adaptam à forma do recipiente que os contém e apresentam certa resistência ao fluxo entre suas camadas, essa resistência é denominada viscosidade [1].

Definimos um fluido ideal como sendo totalmente incompressível e desprovido de viscosidade. Ou seja, ele não oferece resistência entre suas camadas ao movimento e não dissipa energia por meio de atrito interno. Essa definição simplifica significativamente os problemas de fluidos[2]. A análise de fluidos ideais é um ponto de partida para muitos problemas em engenharia e física, proporcionando equações matemáticas mais simples para calcular movimentos e forças associadas ao fluxo. No entanto, na realidade, todos os fluidos têm alguma viscosidade e, portanto, o comportamento dos fluidos reais pode ser bastante complexo, especialmente em condições extremas ou quando ocorrem turbulências [3].

O escoamento de um fluido constitui um dos problemas fundamentais na área da fluidodinâmica. Para elucidar

* angelica.mata@ufla.br

o escopo desse tópico, consideremos um exemplo prático. Ao observar a dispersão de fumaça emanada de uma xícara de café ao longo do tempo, pode-se inicialmente supor que se trata de um fenômeno de natureza simples e intuitiva. Contudo, essa percepção contrasta com a realidade, uma vez que o movimento desordenado das partículas de fumaça impede a definição de um padrão de deslocamento previsível, conferindo uma complexidade inerente ao problema. Ademais, a análise do comportamento de fluidos reveste-se de significativa relevância para a física, dado seu papel na modelagem de fenômenos presentes no cotidiano e em sistemas com diversas aplicações tecnológicas [4]. Dessa maneira, tal problemática tem sido objeto de estudo por parte de numerosos físicos ao longo da história, permanecendo, ainda hoje, um problema em aberto [5].

Para estudar fluidos reais, utilizamos métodos numéricos e simulações computacionais, pois o tratamento analítico associado ao comportamento desses fluidos é, em geral, extremamente complexo. Diferentes modelos e métodos numéricos, como a Dinâmica dos Fluidos Computacional (do inglês *Computational Fluid Dynamics* - CFD), permitem abordar as equações diferenciais que governam a dinâmica dos fluidos com precisão e detalhes. Esses métodos possibilitam a análise de problemas complexos, como fluxo turbulento, transferência de calor e fenômenos multifásicos, proporcionando insights valiosos para aplicações em engenharia, meteorologia, medicina e muitos outros campos [6].

A fluidodinâmica constitui uma disciplina fundamental para a compreensão de sistemas que envolvem o transporte de massa, energia ou quantidade de movimento por meio de um fluido. Em muitas ocasiões, o propósito primordial reside na extração de informações relevantes que possam ser empregadas para ajustar essa dinâmica, visando controlar de maneira eficaz o sistema em questão [7].

Entre as diversas aplicações da fluidodinâmica, destaca-se a análise do projeto e montagem de aeronaves. O impacto do ar em alta velocidade sobre a estrutura de um avião é significativo, tornando crucial o estudo do comportamento do fluido ao redor da aeronave. Tal análise visa não apenas preservar a integridade estrutural da aeronave, mas também otimizar sua eficiência em termos de velocidade e consumo de combustível [8].

Além disso, o conceito de regime de escoamento do fluido engloba as distintas modalidades pelas quais um fluido pode se deslocar em um sistema. Esses padrões de movimento são classificados em dois regimes fundamentais: laminar e turbulento. No regime laminar, o fluido flui a baixas velocidades, caracterizando-se por um movimento regular, uniforme, tranquilo e passível de análise simplificada. Por outro lado, o regime turbulento é identificado por um movimento caótico, desordenado e de alta velocidade, apresentando complexidades que dificultam sua análise e compreensão abrangente [9].

Compreender o regime de escoamento é de suma importância para engenheiros e cientistas envolvidos no transporte de fluidos em uma ampla gama de contextos, que vão desde aplicações industriais até processos biológicos. Essa compreensão influencia diretamente a determinação da força de arrasto, perda de energia e força de atrito entre as paredes, além de outras variáveis cruciais na concepção e otimização do design de um sistema de tubulação [10].

De fato, os desafios mais significativos estão associados ao regime turbulento do escoamento de fluidos, sendo quase inatingível determinar o comportamento do fluido sem o auxílio de software ou simuladores apropriados. As simulações realizadas em Python se destacam como uma abordagem altamente vantajosa, uma vez que facilitam a análise e oferecem praticidade, possibilitando uma visualização mais clara e detalhada dos dados e informações pertinentes ao fluido e ao ambiente no qual está inserido[11].

Além disso, Python é uma linguagem de programação versátil, de fácil aprendizado e acesso, com uma ampla gama de bibliotecas científicas que permitem a implementação eficiente de modelos numéricos e o desenvolvimento de ferramentas computacionais aplicadas à engenharia. Tal funcionalidade é crucial, uma vez que a análise de regimes turbulentos é intrinsecamente complexa, como mencionado anteriormente, e praticamente inviável sem o suporte de software especializado. Assim, a utilização de Python permite uma abordagem mais acessível e eficiente para a compreensão e manipulação de sistemas fluidodinâmicos, contribuindo significativamente para o avanço em projetos e aplicações em engenharias[12].

O presente estudo tem como objetivo demonstrar a simulação do escoamento de um fluido em tubulações utilizando a linguagem Python, com foco didático no uso de bibliotecas específicas, como Pylab e Fluids, para análise e visualização dos resultados. Diferentemente de outros trabalhos da literatura, que enfatizam teoria e métodos numéricos [11], este estudo oferece uma abordagem prática e acessível, facilitando a compreensão dos conceitos fundamentais e o desenvolvimento de habilidades computacionais em estudantes e profissionais da engenharia.

II. METODOLOGIA

Para a modelagem do escoamento em tubulações e o cálculo de perdas de carga, utilizou-se a biblioteca *fluids*[13] em Python, que fornece funções consolidadas para o cálculo de propriedades de escoamentos em tubulações, perdas por atrito e localizadas, fatores de atrito e seleção de diâmetros comerciais. No presente estudo, *fluids* foi empregada em três cenários principais:

(i) determinação da altura manométrica e potência requerida para um sistema de bombeamento de fluido incompressível, considerando perdas por atrito ao longo do tubo e coeficientes de perda em válvulas, curvas e saídas;

(ii) cálculo iterativo da vazão de um gás em regime

isoterma, incluindo a verificação de escoamento estrangulado ("choked flow") e atualização do número de Reynolds e do fator de atrito;

(iii) análise de um sistema com duas tubulações em série, incorporando mudanças de diâmetro, perdas distribuídas e localizadas, conversão de bases de perdas e cálculo das velocidades, números de Reynolds e vazão final.

O uso de *fluids.units* permitiu trabalhar com unidades físicas consistentes ao longo dos cálculos, garantindo que os resultados fossem diretamente interpretáveis em múltiplas unidades de engenharia, como lb/h, gal/min, ft/s e psi. A biblioteca possibilitou automatizar cálculos complexos e repetitivos, reduzir erros manuais e manter a coerência física das simulações hidráulicas apresentadas[14].

Inicialmente, é necessário implementar um código em Python para calcular o **Número de Reynolds (Re)**. O Número de Reynolds é um parâmetro adimensional que relaciona as forças de inércia às forças viscosas em um escoamento, sendo utilizado para caracterizar o regime do fluxo de um fluido [15] Valores de $Re < 2.000$ indicam um **escoamento laminar**, no qual o fator de atrito (f) é calculado pela expressão:

$$f = \frac{64}{Re}$$

Para **escoamentos turbulentos**, geralmente com $Re > 3.500$, o cálculo do fator de atrito requer uma abordagem iterativa devido à complexidade do escoamento. Uma equação comum utilizada é a **equação de Colebrook-White**[3, 6]:

$$\frac{1}{\sqrt{f}} = -2\log_{10}\left(\frac{\epsilon}{3,7D} + \frac{2,51}{Re\sqrt{f}}\right)$$

onde:

- f é o **fator de atrito adimensional**;
- ϵ é a **rugosidade absoluta da tubulação**;
- D é o **diâmetro interno do tubo**;
- Re é o **Número de Reynolds** do escoamento.

O fator de atrito representa a resistência que o fluido encontra ao se movimentar dentro da tubulação, incluindo efeitos viscosos e turbulentos, sendo utilizado para calcular perdas de carga por atrito ao longo do tubo.

Com estas funções, é possível estudar diferentes situações de escoamento, envolvendo fluidos variados, e realizar todos os cálculos necessários de forma automatizada em Python. A partir do Número de Reynolds calculado, determina-se uma estimativa para o **fator de rugosidade relativa** (ϵ/D) e o fator de atrito f, que podem ser usados para estimar perdas de carga e dimensionar sistemas hidráulicos. É importante ressaltar que, embora existam trabalhos que automatizam o cálculo do fator de atrito em tubulações [16], nossa abordagem se diferencia por utilizar a biblioteca `fluids` de forma didática, com exemplos aplicados a problemas de engenharia, visando apoiar o ensino e a compreensão dos conceitos de escoamento de fluidos.

III. RESULTADOS

Os códigos apresentados nos Apêndices A, B e C permitem calcular parâmetros importantes de escoamento em tubulações, como perdas de carga por atrito, vazão mássica, altura manométrica equivalente e potência de bombeamento necessária, para diferentes tipos de fluidos e geometrias de sistema.

No primeiro caso, o programa calcula a perda de carga ao longo de uma tubulação única, considerando tanto as perdas distribuídas quanto as localizadas, e estima a potência necessária da bomba para manter o fluxo desejado.

O segundo caso mostra a aplicação do mesmo método a um escoamento gasoso, considerando o regime laminar ou turbulento e a possibilidade de ocorrência de *choke flow*, permitindo determinar a vazão mássica e a velocidade média do gás. *Choke flow* é um fenômeno que ocorre em escoamentos compressíveis, geralmente de gases, quando a velocidade do fluido atinge o limite máximo permitido em uma constrição da tubulação, como um bocal convergente-divergente ou uma válvula. Nesse ponto, a vazão mássica do fluido não pode aumentar, mesmo que a pressão a jusante seja reduzida [17].

O efeito ocorre devido à combinação da conservação de energia e do efeito Venturi: à medida que o fluido passa pela menor área da constrição, sua velocidade aumenta e a pressão estática e a densidade diminuem. Para fluidos homogêneos, o estrangulamento ocorre quando a velocidade do fluido atinge a velocidade do som local na região da constrição. Assim, a vazão mássica em condições de *choke flow* depende principalmente das propriedades do fluido a montante (entrada), como pressão, temperatura e densidade, tornando-se praticamente independente da pressão a jusante (saída) [3].

Esse fenômeno é relevante em muitas aplicações de engenharia, pois permite controlar a vazão mássica de gases usando válvulas ou orifícios calibrados, garantindo previsibilidade e segurança no projeto de sistemas de tubulação.

Já o terceiro caso exemplifica a análise de sistemas com múltiplas tubulações de diferentes diâmetros, onde o código calcula as perdas combinadas, os números de Reynolds correspondentes e a vazão final do sistema. As tabelas I, II e III exemplificam a utilização dos códigos para um conjunto arbitrário de parâmetros.

Tabela I. Resumo dos principais resultados obtidos no código Python para cálculo da potência de bombeamento (Apêndice A).

Grandeza	**Descrição**	**Valor Calculado**	**Unidade**
Comprimento total da tubulação (L)	Soma dos trechos retilíneos do sistema	500	ft
Diferença de altura (dH)	Desnível entre entrada e saída	400	ft
Vazão volumétrica (Q)	Taxa de escoamento do fluido	100	gal/min
Diâmetro interno (Di)	Tubo nominal de 3" Schedule 40	0,0779	m
Número de Reynolds (Re)	Determina o regime de escoamento (turbulento)	$1,07 \times 10^5$	–
Coeficiente total de perda (K_{tot})	Soma das perdas localizadas e distribuídas	69,5	–
Perda de carga total (dP)	Inclui perdas por atrito e/ou por diferença de altura	181,85	psi
Altura manométrica total (H_T)	Energia total que a bomba deve vencer	420,35	ft
Potência requerida ($power$)	Potência no eixo da bomba	15,15	hp

Tabela II. Resumo dos principais resultados obtidos no código Python para cálculo de vazão de um gás (Apêndice B).

Grandeza	**Descrição**	**Valor Estimado**	**Unidade**
Pressão de entrada (P_1)	Pressão no início do tubo	170	psi
Pressão de saída desejada (P_2)	Pressão no final do tubo	3	atm
Comprimento da tubulação (L)	Distância entre entrada e saída	30	ft
Diâmetro interno (D)	Tubo nominal de 2" Schedule 40	0,053	m
Área da seção transversal (A)	Área do tubo para cálculo de velocidade	0,0022	m^2
Número de Reynolds inicial (Re)	Estimativa inicial do regime de escoamento	10^6	–
Fator de atrito inicial (f_d)	Estimativa inicial para iteração	0,019	–
Rugosidade (ϵ)	Rugosidade do tubo	0,0018	–
Pressão crítica de choke ($P_{2,\text{choke}}$)	Limite de escoamento obstruído	2,99	atm
Vazão mássica final ($\dot{m}$)	Massa de gás que atravessa o tubo	11626,32	lb/h
Velocidade média (v)	Velocidade do gás na tubulação	113,08	m/s

Tabela III. Resumo dos principais resultados obtidos no código Python para cálculo de vazão em sistema com duas tubulações (Apêndice C).

Grandeza	**Descrição**	**Valor Estimado**	**Unidade**
Comprimento da primeira tubulação (L_1)	Tubulação menor	10	ft
Comprimento da segunda tubulação (L_2)	Tubulação maior	20	ft
Diferença de altura (dH)	Desnível entre entrada e saída	11,5	ft
Diâmetro interno da primeira tubulação (D_{i1})	Tubo nominal de 3" Schedule 40	0,0779	m
Diâmetro interno da segunda tubulação (D_{i2})	Tubo nominal de 2" Schedule 40	0,0525	m
Área da seção transversal (A_1, A_2)	Área das tubulações para cálculo de velocidade	0,00477 / 0,00216	m^2
Vazão volumétrica total (Q)	Considerando a tubulação principal	140,5	gal/min

Os valores numéricos adotados nos códigos — viscosidade, densidade, diâmetros nominais, comprimentos das tubulações e rugosidade — correspondem a ordem de grandeza típica de sistemas hidráulicos presentes em exercícios didáticos de Engenharia. A densidade adotada ($\approx 62 lb/ft^3$) e a viscosidade dinâmica próxima de $1 cP$ são compatíveis com água a temperatura ambiente. Os diâmetros selecionados (tubos de 2"e 3", Schedule 40) e a rugosidade de $0,0018 in$ refletem tubulações metálicas padronizadas. Dessa forma, o uso desses valores permite que o aluno aplique os princípios fundamentais da Mecânica dos Fluidos e da Hidráulica em um contexto prático, mas simplificado, ideal para fins de ilustração computacional e aprendizado didático, mesmo que não repliquem uma instalação industrial específica e complexa.

Também é importante ressaltar que os resultados obtidos apresentam ordens de grandeza compatíveis com sistemas reais[15]:

- A velocidade da água no tubo de 3" ($\sim 8-10$ ft/s) está dentro da faixa recomendada para condução de líquidos em tubulações (5–12 ft/s).
- O número de Reynolds calculado ($\sim 10^5$) situa-se claramente no regime turbulento, como esperado para tubulações em escala real.
- A perda de carga total ($\sim$ 180 psi $\approx$ 12,4 atm) e a altura manométrica ($\sim$ 420 ft) são coerentes para um sistema com 500 ft de tubulação, múltiplas perdas localizadas e desnível de 400 ft.
- A potência requerida da bomba ($\sim$ 15 hp) é compatível com bombas comerciais empregadas em pequenos sistemas de recalque de água.
- Para o escoamento compressível, a vazão mássica calculada ($\sim$ 11 600 lb/h) e a velocidade média ($\sim$ 113 m/s) situam-se em faixas típicas para dutos industriais de pequeno porte transportando gás leve.
- No sistema bifurcado (Apêndice C), a vazão total ($\sim$ 140 gal/min) é consistente com o diâmetro da linha principal e com as quedas de pressão adotadas.

Assim, apesar de os casos representarem cenários simplificados, as grandezas obtidas são fisicamente plausíveis, reforçando a utilidade dos códigos como ferramentas pedagógicas.

Em todos os casos, a automação dos cálculos com Python, utilizando a biblioteca `fluids`, garante resultados consistentes, reproduzíveis e reduz significativamente a possibilidade de erros humanos, além de fornecer uma ferramenta didática útil para aplicações em ensino de engenharia.

IV. CONSIDERAÇÕES FINAIS

Observa-se que a linguagem de programação Python desempenhou um papel central neste estudo, permitindo a realização de cálculos e a resolução de problemas relacionados ao escoamento em tubulações de maneira eficiente e precisa. A utilização de ferramentas computacionais mostrou-se especialmente vantajosa em escoamentos turbulentos, nos quais os cálculos manuais seriam complexos e demorados.

Além disso, o estudo evidencia a utilidade do Python como ferramenta acessível para análise e dimensionamento de sistemas de tubulação em engenharia, oferecendo uma alternativa viável a softwares comerciais que, apesar de mais robustos, operam como sistemas proprietários pouco transparentes e requerem licenciamento pago.

AGRADECIMENTOS

Angélica S. da Mata agradece o suporte financeiro de CNPq/Fapemig APQ-06591-24.

[1] D. Halliday, R. Resnick, and J. Walker, *Fundamentals of Physics - Volume 2: Gravitation, Waves, and Thermodynamics* (LTC Editora, 2009).
[2] M. Nussenzveig, *Basic Course in Physics - Volume 2: Fluids, Oscillations, Waves, Heat* (Edgard Blucher, 1999).
[3] B. R. Munson, D. F. Young, T. H. Okiishi, and W. W. Huebsch, *Fundamentals of Fluid Mechanics*, 8th ed. (John Wiley & Sons, 2016).
[4] P. A. Tipler, *Physics for Scientists and Engineers - Volume 1: Mechanics, Oscillations, and Waves, Thermodynamics* (LTC Editora, 2010).
[5] G. K. Batchelor, *An Introduction to Fluid Dynamics* (Cambridge University Press, 2000).
[6] J. D. Anderson, *Computational Fluid Dynamics: The Basics with Applications* (McGraw-Hill Education, 2015).
[7] P. K. Kundu, I. M. Cohen, and D. R. Dowling, *Fluid Mechanics*, 5th ed. (Academic Press, 2012).
[8] J. F. Douglas, J. M. Gasiorek, and J. A. Swaffield, *Fluid Mechanics*, 5th ed. (Pearson Education, 2005).
[9] R. W. Fox, A. T. McDonald, and P. J. Pritchard, *Introduction to Fluid Mechanics*, 8th ed. (John Wiley & Sons, 2011).
[10] Y. A. Cengel and J. M. Cimbala, *Fluid Mechanics: Fundamentals and Applications*, 3rd ed. (McGraw-Hill Education, 2014).

[11] H. K. Versteeg and W. Malalasekera, *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*, 2nd ed. (Pearson Education, 2007).

[12] P. S. Foundation, "Python documentation," (2023), accessed: 2024-05-17.

[13] `https://pypi.org/project/fluids/`.

[14] **Observação:** Os códigos foram executados no Google Colab. Para executar os exemplos, é necessário instalar previamente os pacotes: `!pip install pint fluids CoolProp`.

[15] F. Brunnetti, *Mecânica dos Fluidos*, 2nd ed. (Pearson Prentice Hall, São Paulo, 2008).

[16] P. Dumka *et al.*, International Education and Research Paper **8** (2022), accessed: October 22, 2023.

[17] F. M. White, *Fluid Mechanics*, 7th ed. (McGraw-Hill Education, 2011).

APÊNDICE A: CÓDIGO PYTHON PARA CÁLCULO DE POTÊNCIA

```python
# ------------------------------
# Importacao de bibliotecas
# ------------------------------
from fluids.units import *             # trabalhar com unidades fisicas
from math import pi
# Se precisar de funcoes de 'fluids' (nearest_pipe, friction_factor, etc.)
# verifique se sao importadas de 'fluids' (ou 'fluids.piping') no seu ambiente.

# ------------------------------
# Parametros do sistema
# ------------------------------
L = (30 + 100 + 70 + 300) * u.foot    # comprimento total da tubulacao
dH = 400 * u.foot                     # diferenca de altura entre inicio e fim
efficiency = 0.7                      # eficiencia da bomba
Q = 100 * u.gallon / u.min            # vazao volumetrica do fluido
mu = 0.95 * u.cP                      # viscosidade dinamica do fluido
rho = 62.298 * u.lb / u.ft**3         # densidade do fluido

# ------------------------------
# Selecao dos tubos
# ------------------------------
NPS, Di, Do, t = nearest_pipe(Do=3*u.inch, schedule='40')  # tubo principal
Di_reducer = nearest_pipe(Do=2.5*u.inch, schedule='40')[1] # redutor

# ------------------------------
# Area da secao e velocidade media
# ------------------------------
A = 0.25 * pi * Di**2
v = Q / A

# ------------------------------
# Numero de Reynolds e fator de atrito
# ------------------------------
Re = rho * v * Di / mu
fd = friction_factor(Re=Re, eD=0.0022*u.inch/Di)

# ------------------------------
# Perdas localizadas (valvulas, cotovelos, saida)
# ------------------------------
K_exit = exit_normal()
K_gate = K_gate_valve_Crane(D1=Di, D2=Di, angle=0.0*u.degrees)
K_elbow = bend_rounded(Di=Di, angle=90*u.degrees, Re=Re, method='Crane standard')
K_lift_valve = K_lift_check_valve_Crane(D1=Di_reducer, D2=Di, angled=False)

# ------------------------------
# Soma das perdas
# ------------------------------
K_tot = K_exit + K_gate + 4 * K_elbow + K_lift_valve
K_tot += K_from_f(fd=fd, L=L, D=Di)

# ------------------------------
# Perda de pressao total
# ------------------------------
dP = dP_from_K(K=K_tot, rho=rho, V=v) + rho * dH * u.gravity
# Conversao de unidades (opcional) para exibir
dP_psi = dP.to(u.psi)
v_ft_s = v.to(u.foot / u.s)

# ------------------------------
# Altura manometrica equivalente
# ------------------------------
head = head_from_P(dP, rho).to(u.foot)
print('Altura manometrica (H_T): %s' % head)
# ------------------------------
# Potencia da bomba
# ------------------------------
power = Q * dP / efficiency
print('Potencia requerida da bomba: %s' % (power.to(u.hp)))
```

APÊNDICE B: CÓDIGO PYTHON PARA CÁLCULO DE VAZÃO DE UM GÁS

```python
# ------------------------------
# Importacao de bibliotecas
# ------------------------------
from fluids.units import *      # unidades e utilitarios do pacote fluids
from math import pi
from CoolProp.CoolProp import PropsSI    # se usar propriedades termofisicas

# ------------------------------
# Condicoes de contorno
# ------------------------------
P1 = 170 * u.psi          # pressao de entrada
P2_spec = 3 * u.atm       # pressao de saida desejada
L = 30 * u.foot           # comprimento da tubulacao

# ------------------------------
# Selecao da tubulacao
# ------------------------------
NPS, D_pipe, Do_pipe, t = nearest_pipe(NPS=2, schedule=40)
A = 0.25 * pi * D_pipe**2  # area da secao transversal

# ------------------------------
# Propriedades iniciais e estimativas
# ------------------------------
fd = 0.019                    # fator de atrito inicial (estimativa)
Re = 1e6                      # numero de Reynolds inicial (estimativa)
mu = 1.8e-5 * u.Pa * u.s      # viscosidade dinamica do gas (valor tipico, verifique)
rho = 5.9886 * u.kg / u.m**3 # densidade do gas na entrada (verifique para o fluido real)
roughness = 0.0018 * u.inch  # rugosidade do tubo

# ------------------------------
# Iteracao para calculo da vazao
# ------------------------------
for i in range(5):
    # perdas distribuidas e localizadas
    K = K_from_f(fd=fd, L=L, D=D_pipe)
    K += entrance_sharp()                    # entrada brusca
    K += exit_normal()                       # saida livre
    K += K_globe_valve_Crane(D1=D_pipe, D2=D_pipe)   # valvula globo
    K += bend_rounded(Di=D_pipe, angle=90*u.degrees,
                      fd=fd, Re=Re, roughness=roughness,
                      method='Crane')        # curva de 90 graus

    # fator de atrito equivalente obtido a partir de K total
    fd_tot = f_from_K(L=L, D=D_pipe, K=K)
    # pressao critica para regime de escoamento (choke) - ajustar se a funcao tiver assinatura diferente
    P2_choke = P_isothermal_critical_flow(P=P1, fd=fd_tot, D=D_pipe, L=L)
    # escolhe P2: a menor entre a especificada e a de choke
    if P2_choke.to_base_units().magnitude > P2_spec.to_base_units().magnitude:
        P2 = P2_spec
    else:
        P2 = P2_choke
    # calculo da vazao massica e velocidade (verifique assinatura de isothermal_gas)
    m = isothermal_gas(rho=rho, fd=fd_tot, P1=P1, P2=P2, L=L, D=D_pipe)
    Q = m / rho
    v = Q / A
    # atualizacao do numero de Reynolds e fator de atrito (verifique funcoes/Reynolds/friction_factor)
    Re = Reynolds(D=D_pipe, rho=rho, mu=mu, V=v)
    fd = friction_factor(Re=Re, eD=roughness/D_pipe)

# ------------------------------
# Saida dos resultados
# ------------------------------
print('Vazao massica final: %s' % (m.to(u.lb / u.hour)))
print('Vazao volumetrica final: %s' % (Q.to(u.gallon / u.min)))
print('Velocidade media: %s' % (v.to(u.foot / u.s)))
```

APÊNDICE C: CODIGO PYTHON PARA CÁLCULO DE VAZÃO EM SISTEMA COM DUAS TUBULAÇÕOES

```python
# ------------------------------
# Importacao de bibliotecas
# ------------------------------
from fluids.units import *
from math import pi
# ------------------------------
# Parametros do sistema
# ------------------------------
L1 = 10 * u.foot           # comprimento da primeira tubulacao
L2 = 20 * u.foot           # comprimento da segunda tubulacao
dH = 11.5 * u.foot         # diferenca de altura
mu = 1.1 * u.cP            # viscosidade dinamica
rho = 62.364 * u.lb / u.ft**3  # densidade
# ------------------------------
# Selecao de tubos e Areas de secao transversal
# ------------------------------
NPS1, Di1, Do1, t1 = nearest_pipe(NPS=3, schedule='40')
NPS2, Di2, Do2, t2 = nearest_pipe(NPS=2, schedule='40')
A1 = 0.25 * pi * Di1**2
A2 = 0.25 * pi * Di2**2
# ------------------------------
# Fatores de atrito iniciais
# ------------------------------
fd1 = ft_Crane(Di1)
fd2 = ft_Crane(Di2)
roughness = 0.0018 * u.inch
dP = rho * dH * 1 * u.gravity
fd1fd2 = 0.018  # chute inicial
# ------------------------------
# Calculo das perdas de carga
# ------------------------------
# referencia: tubo de 3"
for i in range(10):
    K_entrance = entrance_sharp(method='Crane')
    K_exit = change_K_basis(exit_normal(), 2*u.inch, 3*u.inch)
    K_gate = K_gate_valve_Crane(D1=Di1, D2=Di1, angle=0.0*u.degrees)
    K_elbow = bend_miter(Di=Di1, angle=90*u.degrees, method='Crane')
    K_contraction = change_K_basis(
        contraction_conical_Crane(3*u.inch, 2*u.inch, L=0*u.m),
        2*u.inch, 3*u.inch
    )
    # perdas localizadas + distribuida
    K_tot = K_entrance + K_elbow + K_gate + K_exit + K_contraction
    K_f1 = K_from_f(fd=fd1, L=L1, D=Di1)
    K_f2 = change_K_basis(K_from_f(fd=fd2, L=L2, D=Di2), 2*u.inch, 3*u.inch)
    K_tot += K_f1 + K_f2
# ------------------------------
# Conversao de base de perdas
# ------------------------------
K_tot_basis2 = change_K_basis(K_tot, 3*u.inch, 2*u.inch)
# ------------------------------
# Velocidades
# ------------------------------
v1 = (2*dP / (K_tot * rho))**0.5
v2 = (2*dP / (K_tot_basis2 * rho))**0.5
# ------------------------------
# Numeros de Reynolds
# ------------------------------
Re1 = rho * v1 * Di1 / mu
Re2 = rho * v2 * Di2 / mu
# Atualizacao dos fatores de atrito
fd1 = friction_factor(Re=Re1, eD=roughness/Di1)
fd2 = friction_factor(Re=Re2, eD=roughness/Di2)
# ------------------------------
# Vazao final
Q = A1 * v1
Q.to(u.gal/u.min)
```