Universidade Federal de Juiz de Fora

Instituto de Engenharia Mecânica

Máquinas de Fluxo

Eduardo Santos de Oliveira Marques
Gabriela Fonseza Franck de Souza
Henrique Vargas Mendonça

SIMULAÇÃO NUMÉRICA

Juiz de Fora

2022

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

A	Área da superfície
α	Constante dependente da bomba
β	Constante dependente da bomba
D	Diâmetro
ε	Rugosidade do tubo
f	Fator de atrito de Moody
L	Comprimento
p	Pressão
Q	Vazão volumétrica
Re	Número de Reynolds
ρ	Densidade
μ	Viscosidade dinâmica
$\overline{V}$	Velocidade
z	Altura de carga

SUMÁRIO

1 INTRODUÇÃO

1.1 OBJETIVOS

Este projeto consiste em um relatório referente a uma simulação numérica de um sistema de bombeamento, referente a disciplina de Máquinas de Fluxo [1]. O objetivo é resolver as equações para as seguintes incógnitas: α_A, β_A, α_B, β_B, α_F, β_F, f_C, f_D, f_G,f_E, Re_C, Re_D, Re_G, Re_E, Q_C, Q_D, Q_G, Q_E, p_2, p_3, p_4 e p_6. A simulação é feita através de um programa de computador, na linguagem ou software escolhidos pelos autores, sendo possível inserir os valores: ρ, μ, ε_C, ε_D, ε_E, ε_G, $(z_5 - z_4)$, D_C, L_C, D_D, L_D, D_G, L_G, D_E e L_E. O projeto consiste em três etapas, sendo elas:

- Fluxograma;
- Pesquisa bibliográfica;
- Simulação feita de forma computacional.

1.2 ENUNCIADO

O sistema de bombeamento para a simulação é encontrado na figura abaixo [2]:

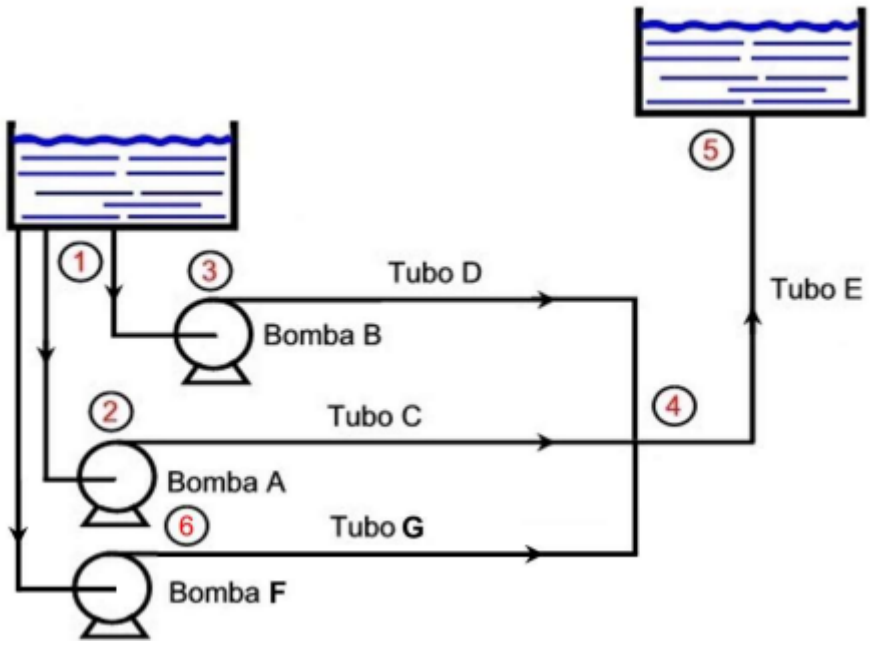

Figura 1 – Sistema de bombeamento

Considera-se a água escoando com uma vazão Q (gpm), fluindo do ponto 1 ao ponto 2 em um tubo vertical de comprimento L (ft) e diâmetro D (in). A queda de pressão entre os diferentes níveis de referência pode ser calculada através da equação de Fanning (1.1):

$$(p_i - p_j) = \frac{f\rho\overline{V}^2 L}{2D} + \rho g(z_j - z_i) \tag{1.1}$$

Assumindo o fator de atrito de Moody f como uma função da rugosidade do tubo ϵ (in), onde μ é a viscosidade dinâmica do fluido, o número de Reynolds é dado por:

$$Re = \frac{D\rho\overline{V}}{12\mu} \tag{1.2}$$

Para valores de $Re \leq 2000$, têm-se:

$$f = \frac{64}{Re} \tag{1.3}$$

Para $Re > 2000$, o fator de Moody é dado pela equação de Colebrook:

$$\frac{1}{\sqrt{f}} = -2\log_{10}\left(\frac{\varepsilon}{3,7D} + \frac{2,51}{Re\sqrt{f}}\right) \tag{1.4}$$

Para escoamentos turbulentos em tubos de rugosidade média, utiliza-se a seguinte equação:

$$(z_2 - z_1) + 2,31(p_2 - p_1) + 8,69 \times 10^{-4}\left(\frac{LQ^2}{D^5}\right) = 0 \tag{1.5}$$

A curva típica de altura de carga para uma bomba centrífuga pode ser representada pela equação abaixo, onde Δp é o aumento de pressão em $psig$ através da bomba, Q é a vazão volumétrica em gpm, α e β são constantes dependentes da bomba.

$$\Delta p = \alpha - \beta Q^2 \tag{1.6}$$

Considerando o sistema da figura 1, as pressões p_1 e p_5 são atmosféricas (0 $psig$). Existem um aumento da elevação entre os pontos 4 e 5, mas os tubos C, D e G são horizontais. Com base nisso, têm-se as equações governantes:

$$Q_E = Q_C + Q_D + Q_G \tag{1.7}$$

$$p_2 = \alpha_A - \beta_A Q_C^2 \tag{1.8}$$

$$p_3 = \alpha_B - \beta_B Q_D^2 \tag{1.9}$$

$$p_6 = \alpha_F - \beta_F Q_G^2 \tag{1.10}$$

$$2,31(p_4 - p_2) + 8,69 \cdot 10^{-4} \left(\frac{L_C Q_C^2}{D_C^5} \right) = 0 \tag{1.11}$$

$$2,31(p_4 - p_3) + 8,69 \cdot 10^{-4} \left(\frac{L_D Q_D^2}{D_D^5} \right) = 0 \tag{1.12}$$

$$2,31(p_4 - p_6) + 8,69 \cdot 10^{-4} \left(\frac{L_G Q_G^2}{D_G^5} \right) = 0 \tag{1.13}$$

$$(z_5 - z_4) + 2,31(0 - p_4) + 8,69 \cdot 10^{-4} \left(\frac{L_e Q_E^2}{D_E^5} \right) = 0 \tag{1.14}$$

$$Q = A\overline{V} \tag{1.15}$$

1.2.1 **Considerações**

Além das equações de Fanning, Moody, Colebrook, curva típica e governantes; é preciso definir alguns parâmetros e realizar considerações sobre o sistema. Tais considerações são a diferença de cotas, curva de desempenho das bombas, diâmetro, comprimento e rugosidade da tubulações, e o fluido de trabalho.

A diferença de cotas entre os pontos z_5 e z_4 é dada por: $(z_5 - z_4) = 130\ ft$

Com relação as bombas, têm-se os seguintes valores adimensionais:

Tabela 1 – Bombas

Bomba	α $[psi]$	β $[psi/(gpm)^2]$
A	Curva H vs Q para o rotor D2	Curva H vs Q para o rotor D2
B	Curva H vs Q para o rotor D3	Curva H vs Q para o rotor D3
F	Curva H vs Q para o rotor D4	Curva H vs Q para o rotor D4

As curvas de desempenho são mostradas abaixo.

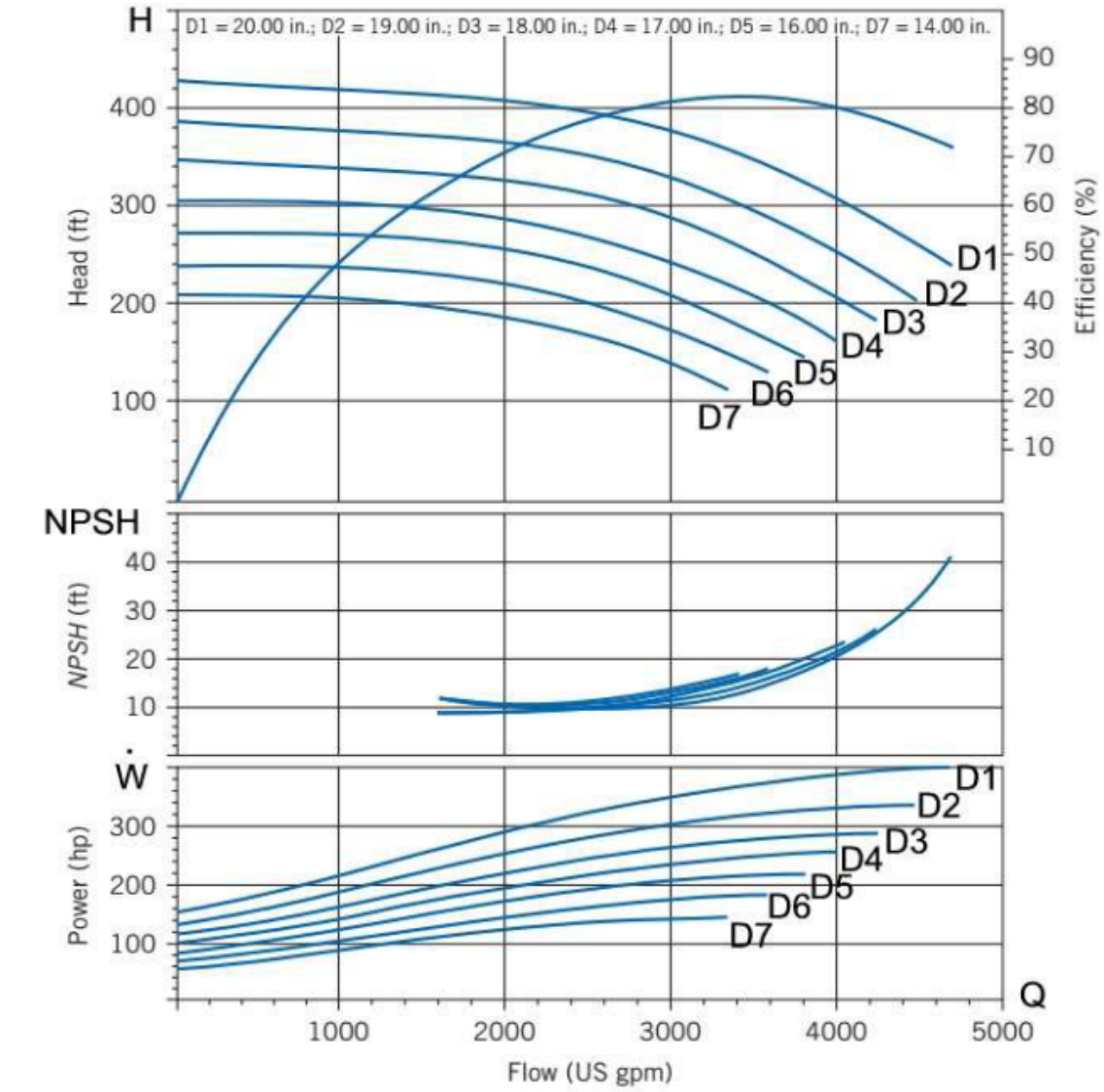

Figura 2 – Curvas de desempenho

Sobre as tubulações, têm-se as seguintes características:

Tabela 2 – Tubulações

Tubo	$D\ [in]$	$L\ [ft]$	$\varepsilon\ [ft]$
C	2,50	140	0,0025
D	2,75	140	0,0020
E	3,50	190	0,0035
G	2,25	150	0,0030

Por fim, o fluido de trabalho em bombas e tubulações é dado por:

Tabela 3 – Fluidos de trabalho

Fluido	$\rho\ [lb_m/ft^3]$	$\mu\ [lb_m/(ft \cdot s)]$
A	62,15	0,0005796
B	62,15	0,0005796
F	62,15	0,0005796

Também considera-se que os comprimentos dos tubos acima já incluíram os comprimentos equivalentes de todas as conexões e válvulas.

2 METODOLOGIA

Para realizar as simulações numéricas, utiliza-se o Google Colab [3], um serviço de nuvem gratuito que permite utilizar a linguagem de programação Python em um ambiente colaborativo, podendo também ser possível adicionar texto no código, permitindo assim um melhor entendimento dos processos [4]. Abaixo é mostrado um fluxograma do processo da modelagem.

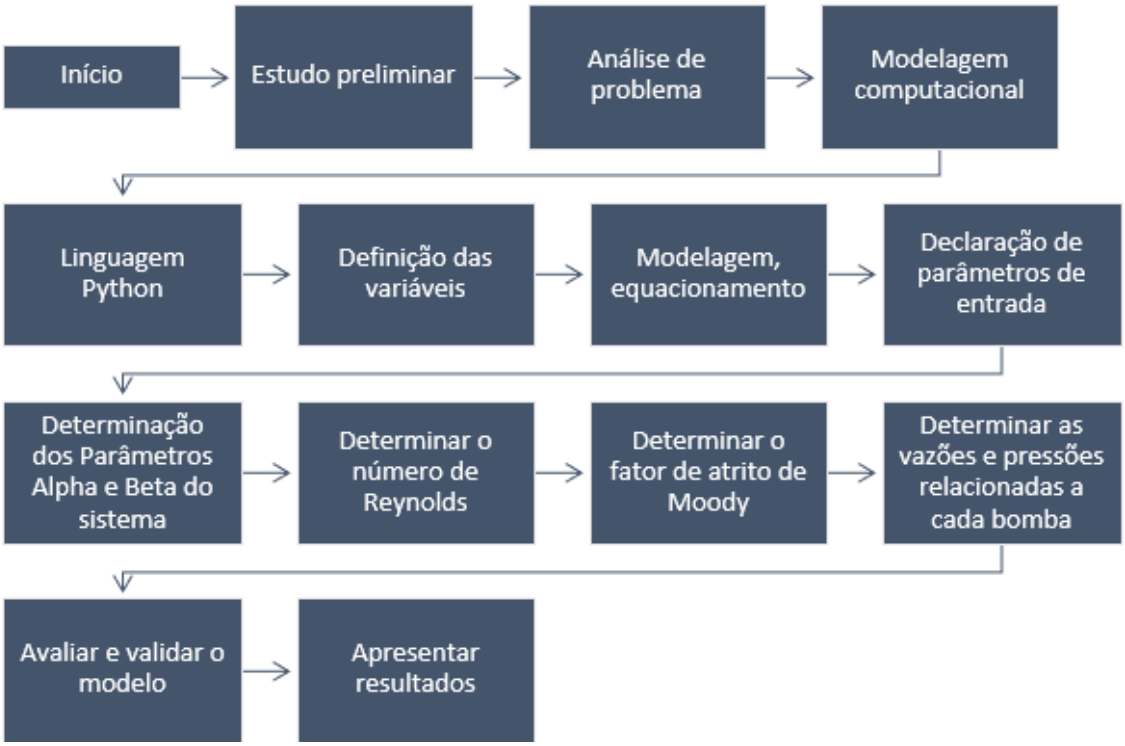

Figura 3 – Fluxograma da modelagem

3 DESENVOLVIMENTO

Para iniciar o programa é necessário importar as bibliotecas que serão utilizadas ao longo do desenvolvimento da simulação, bibliotecas permitem inputs simples para utilizar alguns recursos que facilitam a programação do código:

```
[ ]  import numpy as np
     import math
     import sympy as sp
     import matplotlib.pyplot as plt
     from scipy.optimize import fsolve
```

Figura 4 – Bibliotecas importadas para utilização no código

Em sequência, definiu-se as variáveis de entrada que foram estabelecidas.

```
[ ]  # Densidade do fluido (a mesma para todos) [lb_m/ft^3]
     rho_A, rho_B, rho_F = 62.15, 62.15, 62.15
     rho = 62.15
     # Viscosidade do fluido (a mesma para todos) [lb_m/ft.s]
     u_A, u_B, u_F = 0.0005796, 0.0005796, 0.0005796
     u = 0.0005796
     # Comprimento dos tubos [ft]
     L_C = 140
     L_D = 140
     L_E = 190
     L_G = 150
     # Rugosidade dos tubos [ft]
     e_C = 0.0025
     e_D = 0.0020
     e_E = 0.0035
     e_G = 0.0030
```

Figura 5 – Dados de entrada

Além disso, é preciso definir manualmente os pontos da curva de desempenho para poder calcular os valores de α e β, foram definidos 5 pontos.

```
[ ]  # Pontos das curvas (5 pontos)
     pontos_D2 = np.array([[0,385], [1000,380], [2000,375], [3000,340], [4000,260]], dtype=float)
     pontos_D3 = np.array([[0,345], [1000,340], [2000,330], [3000,290], [4000,210]], dtype=float)
     pontos_D4 = np.array([[0,305], [1000,300], [2000,285], [3000,245], [4000,160]], dtype=float)
```

Figura 6 – Dados da curva de desempenho

Por fim, têm-se os dados específicos para cada grupo, definidos previamente no escopo do projeto da disciplina.

```
[ ]  # Diferença de cotas (z_5 - z_4) [ft]
     z_54 = 130
     # Diâmetros nos tubos [in]
     D_C = 2.50
     D_D = 2.75
     D_E = 3.50
     D_G = 2.25
```

Figura 7 – Dados específico para o grupo

Como o escopo do projeto define a possibilidade de inserir os dados manualmente, foi-se criada uma lógica para poder utilizar tal recurso.

```
[ ]  # Explanação dos valores sugeridos
     print("DADOS DE ENTRADA:\n")
     print("Valores Sugeridos:")
     print("Densidade: rho = 62.15 [lb_m/ft^3]")
     print("Viscosidade: u = 0.0005796 [lb_m/(ft.s)]")
     print("Diferenças entre as cotas z_5 e z_4: z_54 = 130 [ft]")
     print("Rugosidade, Diâmetro e Comprimento do tubo C: e_C = 0.0025 [ft], D_C = 2.50 [in], L_C = 140 [ft]")
     print("Rugosidade, Diâmetro e Comprimento do tubo D: e_D = 0.0020 [ft], D_D = 2.75 [in], L_D = 140 [ft]")
     print("Rugosidade, Diâmetro e Comprimento do tubo E: e_E = 0.0035 [ft], D_E = 3.50 [in], L_E = 190 [ft]")
     print("Rugosidade, Diâmetro e Comprimento do tubo G: e_G = 0.0030 [ft], D_G = 2.25 [in], L_G = 150 [ft]")

     # Perguntar se o usuário deseja usar tais valores ou definir manualmente
     continuar =  input("\nDeseja continuar com tais valores? Informar [S] para continuar ou [N] para inserir outros: ").upper()
     print()

     if continuar == 'S':
       print("Valores mantidos.")

     elif continuar == 'N':
       # Densidade
       rho = float(input("Digite a densidade (rho) [lb_m/ft^3]: "))
       print()
       # Viscosidade
       u = float(input("Digite a viscosidade (u) [lb_m/(ft.s)]: "))
       print()
       # Diferença entre as cotas
       z_5_4 = float(input("Digite a diferença entre as cotas z_5 e z_4 (z_54) [ft]: "))
       print()
       # Tubo C
       print("Tubo C:\n")
       D_C = float(input("Digite o Diâmetro (D_C) [in]: "))
       e_C = float(input("Digite a Rugosidade (e_C) [ft]: "))
       L_C = float(input("Digite o Comprimento (L_C) [ft]: "))
       print()
       # Tubo D
       print("Tubo D:\n")
       e_D = float(input("Digite a Rugosidade (e_D) [ft]: "))
       D_D = float(input("Digite o Diâmetro (D_D) [in]: "))
       L_D = float(input("Digite o Comprimento (L_D) [ft]: "))
       print()
       # Tubo E
       print("Tubo E:\n")
       e_E = float(input("Digite a Rugosidade (e_E) [ft]: "))
       D_E = float(input("Digite o Diâmetro (D_E) [in]: "))
       L_E = float(input("Digite o Comprimento (L_E) [ft]: "))
       print()
       # Tubo G
       print("Tubo G:\n")
       e_G = float(input("Digite a Rugosidade (e_G) [ft]: "))
       D_G = float(input("Digite o Diâmetro (D_G) [in]: "))
       L_G = float(input("Digite o Comprimento (L_G) [ft]: "))
       print()
       print("Novos valores inseridos.")

     else:
       print("Resposta não reconhecida.")
```

Figura 8 – Dados específico para o grupo

São mostrados valores sugeridos, depois pergunta-se ao usuário se ele manter tais valores ou definir manualmente, caso seja selecionada a segunda opção, será feita a imputação dos dados. Caso não for selecionada nenhuma resposta, o código mostrará que tal valor não foi reconhecido.

```
DADOS DE ENTRADA:

Valores Sugeridos:
Densidade: rho = 62.15 [lb_m/ft^3]
Viscosidade: u = 0.0005796 [lb_m/(ft.s)]
Diferenças entre as cotas z_5 e z_4: z_54 = 130 [ft]
Rugosidade, Diâmetro e Comprimento do tubo C: e_C = 0.0025 [ft], D_C = 2.50 [in], L_C = 140 [ft]
Rugosidade, Diâmetro e Comprimento do tubo D: e_D = 0.0020 [ft], D_D = 2.75 [in], L_D = 140 [ft]
Rugosidade, Diâmetro e Comprimento do tubo E: e_E = 0.0035 [ft], D_E = 3.50 [in], L_E = 190 [ft]
Rugosidade, Diâmetro e Comprimento do tubo G: e_G = 0.0030 [ft], D_G = 2.25 [in], L_G = 150 [ft]

Deseja continuar com tais valores? Informar [S] para continuar ou [N] para inserir outros: s

Valores mantidos.
```

Figura 9 – Output da Figura 8

Após definidos os dados de entrada, criou-se funções para cada equação que será utilizada para resolução do problema proposto. A definição dessas funções auxiliam o desenvolvimento do código, tornando-o mais limpo e fluido. A primeira função diz respeito para o cálculo de α e β.

```
[ ]  def AlphaBeta(pontos_D, nome_curva):
       # Eixos do gráfico
       x_D = []
       y_D = []
       for i in range(len(pontos_D)):
         x_D.append(pontos_D[i][0])
         y_D.append(pontos_D[i][1])

       # Plotagem dos pontos
       plt.scatter(x_D, y_D)
       plt.plot(x_D, y_D, label='Pontos')
       plt.xlabel('Flow (US gpm)')
       plt.ylabel('Head (psi)')
       plt.title(nome_curva)
       plt.tight_layout()
       # Ajuste da equação
       p_1 = pontos_D[1,1]
       Q_1 = pontos_D[1,0]
       p_2 = pontos_D[4,1]
       Q_2 = pontos_D[4,0]
       # Ajuste do gráfico
       A = np.array([[1,-Q_1**2], [1,-Q_2**2]])
       B = np.array([[p_1], [p_2]])
       X = np.linalg.solve(A, B)
       alpha = X[0,0]
       beta = X[1,0]
       # Eixos do gráfico ajustado
       x = np.linspace(0, x_D[len(x_D) - 1])
       y = alpha - beta*x**2
       plt.plot(x, y, label='Ajuste')
       plt.legend()
       plt.show()
       # Fator de conversão de [ft] para [psi]
       psi = 0.43
       alpha = alpha * psi
       beta = beta * psi
       print("alpha = ", alpha, "[psi]")
       print("beta = ", beta, "[psi/gmp^2]")
       return alpha, beta
```

Figura 10 – Determinação da função para o cálculo de α e β

A segunda se refere à vazão e a pressão.

```
[ ]  # Vazão
     def Vazao(solucao, valor, nome):
       Q = solucao[0][valor]
       print(nome + ' = ' + str(Q))
       return Q

     # Pressão
     def Pressao(alpha, beta, Q, nome):
       p = alpha - beta * Q**2
       print(nome + ' = ' + str(p))
       return p
```

Figura 11 – Determinação da função para o cálculo de Q e p

Por fim, têm-se as funções para o número de Reynolds e o fator de atrito.

```
[ ]  # Número de Reynalds
     def Reynolds(D, Q, rho, u, nome):
       A = math.pi * (D / 2)**2
       V = Q / A
       Re = (D * rho * V) / (12 * u)
       print(nome + ' = ' + str(Re))
       return Re

     # Fator de atrito
     def Moody(e, D, Re, nome):
       # Equação de Colebrook
       def Colebrook(f):
         colebrook = - 1 / math.sqrt(f) - 2 * math.log10(e / (3.7 * D) + 2.51 / (Re * math.sqrt(f)))
         return colebrook

       # Moody
       if (Re > 2000):
         f = fsolve(Colebrook, 0.02, xtol = 1.5e-08)
         f = f[0]
       else:
         f = 64 / Re
       # Resultado
       print(nome + ' = ' + str(f))
       return f
```

Figura 12 – Determinação da função para o cálculo de Re e f

Calculou-se então os parâmetros α e β, chamando a função apresentada na Figura 10 e inserindo os pontos determinados no passo anterior. Nessa função, o programa realiza a plotagem dos pontos, faz os ajustes necessários para a equação da curva e do gráfico, e determina os parâmetros α e β com fator de conversão aplicados.

```
[ ]  alpha_A, beta_A = AlphaBeta(pontos_D2, 'D2 (alpha_A, beta_A)')
     alpha_B, beta_B = AlphaBeta(pontos_D3, 'D3 (alpha_B, beta_B)')
     alpha_F, beta_F = AlphaBeta(pontos_D4, 'D4 (alpha_F, beta_F)')
```

Figura 13 – Chamando as funções para determinação dos parâmetros α e β

O resultado dessa função apresenta as curvas plotadas com os pontos oferecidos, ajuste da equação das curvas e o cálculo dos parâmetros para as bombas A, B e F.

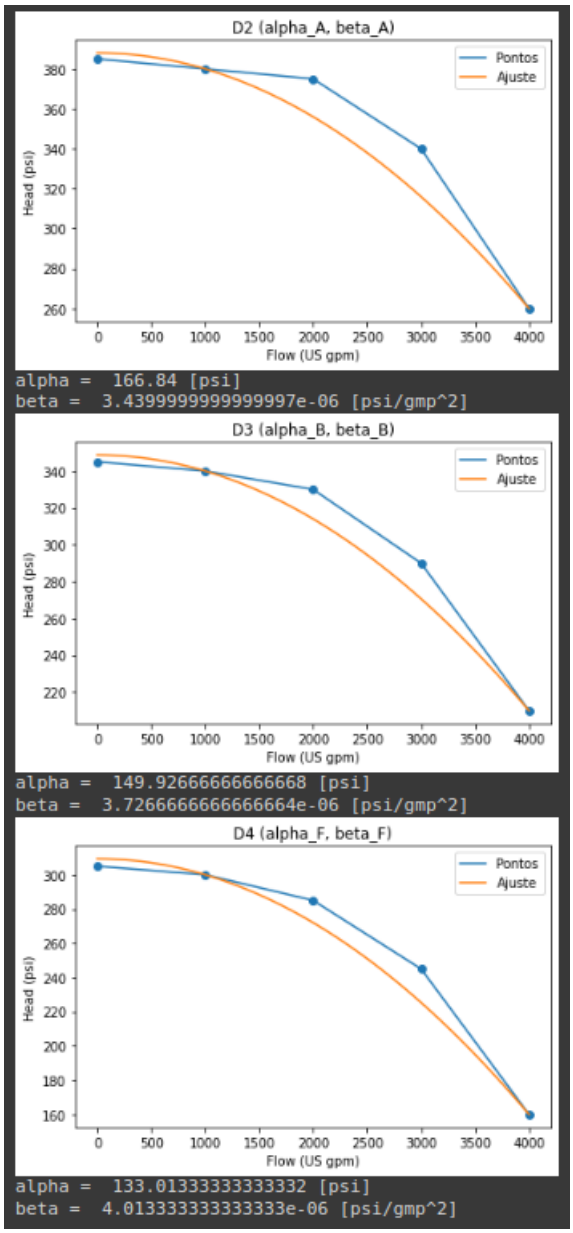

Figura 14 – Resultado dos parâmetros α e β para as bombas A, B e F.

A partir da definição dos parâmetros α e β das bombas A, B e F , é possível seguir para a determinação das vazões e pressões associadas a cada bomba.

Como de costume, primeiro são definidas as funções que irão auxiliar o desenvolvimento do código, tornando-o menos poluído. As funções auxiliares que usaremos para o cálculo da vazão e pressão são apresentadas na Figura 11. São apresentados as linhas de código e o resultado das funções utilizando os resultados encontrados para α e β e as equações 1.11, 1.12 e 1.13 para determinação das vazões nos tubos e as pressões na saída das bombas.

```
[ ] # Constante
    c = 8.69 * 10**(-4)

    # OBS: Colocar todas as equações juntas faz o 'SymPy' rodar indefinidamente, por isso utilizou-se apenas 4 equações

    # Termos
    p_4, Q_C, Q_D, Q_E = sp.symbols('p_4, Q_C, Q_D, Q_E', positive=True)
    # Equações
    eq_1 = sp.Eq(Q_C + Q_D, Q_E)
    eq_2 = sp.Eq(2.31 * (p_4 - (alpha_A - beta_A * Q_C**2)+ c * (L_C * Q_C**2 / D_C**5)), 0)
    eq_3 = sp.Eq(2.31 * (p_4 - (alpha_B - beta_B * Q_D**2)+ c * (L_D * Q_D**2 / D_D**5)), 0)
    eq_5 = sp.Eq(z_54 + 2.31 * (0 - p_4) + c * (L_E * Q_E**2 / D_E**5), 0)
    # Solução
    solucao = sp.solve([eq_1, eq_2, eq_3, eq_5], [p_4, Q_C, Q_D, Q_E], dict=True)
    # Resultados
    Q_C = Vazao(solucao, Q_C, 'Q_C')
    Q_D = Vazao(solucao, Q_D, 'Q_D')
    Q_E = Vazao(solucao, Q_E, 'Q_E')
    p_4 = Vazao(solucao, p_4, 'p_4')
    p_2 = Pressao(alpha_A, beta_A, Q_C, 'p_2')
    p_3 = Pressao(alpha_B, beta_B, Q_C, 'p_3')

    # Equacioanmento 2 (para o 'Q_G' e 'p_6')
    Q_G = sp.symbols('Q_G', positive=True)
    eq_4 = sp.Eq(2.31 * (p_4 - (alpha_F - beta_F * Q_G**2)+ c * (L_G * Q_G**2 / D_G**5)), 0)
    # Solução
    solucao2 = sp.solve([eq_4], [Q_G], dict=True)
    # Resultados
    Q_G = Vazao(solucao2, Q_G, 'Q_G')
    p_6 = Pressao(alpha_F, beta_F, Q_G, 'p_6')

    Q_C = 243.967037436951
    Q_D = 271.847952323528
    Q_E = 515.814989760479
    p_4 = 92.4854361728314
    p_2 = 166.635251491176
    p_3 = 149.704855782108
    Q_G = 133.780440610894
    p_6 = 132.941505878756
```

Figura 15 – Determinação das vazões e pressões

Com estes parâmetros parâmetros, é possível seguir para o cálculo de número de Reynolds e o fator de atrito. O cálculo é baseado no valor de Re, caso ele for superior a 2000, utiliza-se a equação de Colebrook. Para fazer isso, é preciso utilizar a biblioteca *SciPy*, onde é possível encontrar as raízes de uma função, definindo uma estimativa inicial, neste caso sendo 0.02 com uma tolerância de $1.5 \times e^{-8}$.

```
[ ]  # Número de Reynalds
     Re_C = Reynolds(D_C, Q_C, rho, u, 'Re_C')
     Re_D = Reynolds(D_D, Q_D, rho, u, 'Re_D')
     Re_E = Reynolds(D_E, Q_E, rho, u, 'Re_E')
     Re_G = Reynolds(D_G, Q_G, rho, u, 'Re_G')

     # Fator de atrito
     f_C = Moody(e_C, D_C, Re_C, 'f_C')
     f_D = Moody(e_D, D_D, Re_D, 'f_D')
     f_E = Moody(e_E, D_E, Re_E, 'f_E')
     f_G = Moody(e_G, D_G, Re_G, 'f_G')
```

```
Re_C = 1110280.65400859
Re_D = 1124695.61263677
Re_E = 1676747.01993502
Re_G = 676474.970897140
f_C = 0.01991362543935783
f_D = 0.018543841482332304
f_E = 0.01982120828023836
f_G = 0.0214523343083544
```

Figura 16 – Determinação do número de Reynolds e fator de atrito

Finalmente, são apresentados os inputs e outputs do modelo.

```
[ ]  print('DADOS DE ENTRADA:\n')
     print('rho =', rho, '[lb_m/ft^3]')
     print('u =', u, '[lb_m/(ft.s)]')
     print('e_C =', e_C, '[ft]')
     print('e_D =', e_D, '[ft]')
     print('e_E =', e_E, '[ft]')
     print('e_G =', e_G, '[ft]')
     print('z_54 =', z_54, '[ft]')
     print('D_C =', D_C, '[in]')
     print('D_D =', D_D, '[in]')
     print('D_E =', D_E, '[in]')
     print('D_G =', D_G, '[in]')
     print('L_C =', L_C, '[ft]')
     print('L_D =', L_D, '[ft]')
     print('L_E =', L_E, '[ft]')
     print('L_G =', L_G, '[ft]')
     print()
```

```
DADOS DE ENTRADA:

rho = 62.15 [lb_m/ft^3]
u = 0.0005796 [lb_m/(ft.s)]
e_C = 0.0025 [ft]
e_D = 0.002 [ft]
e_E = 0.0035 [ft]
e_G = 0.003 [ft]
z_54 = 130 [ft]
D_C = 2.5 [in]
D_D = 2.75 [in]
D_E = 3.5 [in]
D_G = 2.25 [in]
L_C = 140 [ft]
L_D = 140 [ft]
L_E = 190 [ft]
L_G = 150 [ft]
```

Figura 17 – Inputs do modelo

Para se ter uma melhor visualização dos resultados, os dados menores que 1 foram arredondados utilizando a função a seguir.

```
[ ]  # Arredondamebnto para números positivos menores que 1
     def special_round(x, n) :
         lx = math.log10(abs(x))
         if lx >= 0 : return round(x, n)
         return round(x, n - math.ceil(lx))
```

Figura 18 – Função para arrendondamento de números menores que 1

```
[ ]  print('RESULTADOS:\n')
     print('alpha_A =', round(alpha_A, 2), '[psi]')
     print('alpha_B =', round(alpha_B, 2), '[psi]')
     print('alpha_F =', round(alpha_F, 2), '[psi]')
     print('beta_A =', special_round(beta_A, 3), '[psi/(gpm)^2]')
     print('beta_B =', special_round(beta_B, 3), '[psi/(gpm)^2]')
     print('beta_F =', special_round(beta_F, 3), '[psi/(gpm)^2]')
     print("p_2 =", round(p_2, 2), "[psig]")
     print("p_3 =", round(p_3, 2), "[psig]")
     print("p_4 =", round(p_4, 2), "[psig]")
     print("p_6 =", round(p_6, 2), "[psig]")
     print("Q_C =", round(Q_C, 2), "[gpm]")
     print("Q_D =", round(Q_D, 2), "[gpm]")
     print("Q_E =", round(Q_E, 2), "[gpm]")
     print("Q_G =", round(Q_G, 2), "[gpm]")
     print("Re_C =", format(Re_C, '.2f'), "[adimensional]")
     print("Re_D =", round(Re_D, 2), "[adimensional]")
     print("Re_E =", round(Re_E, 2), "[adimensional]")
     print("Re_G =", round(Re_G, 2), "[adimensional]")
     print("f_C =", round(f_C, 5), "[adimensional]")
     print("f_D =", round(f_D, 5), "[adimensional]")
     print("f_E =", round(f_E, 5), "[adimensional]")
     print("f_G =", round(f_G, 5), "[adimensional]")
```

```
RESULTADOS:

alpha_A = 166.84 [psi]
alpha_B = 149.93 [psi]
alpha_F = 133.01 [psi]
beta_A = 3.44e-06 [psi/(gpm)^2]
beta_B = 3.73e-06 [psi/(gpm)^2]
beta_F = 4.01e-06 [psi/(gpm)^2]
p_2 = 166.64 [psig]
p_3 = 149.70 [psig]
p_4 = 92.49 [psig]
p_6 = 132.94 [psig]
Q_C = 243.97 [gpm]
Q_D = 271.85 [gpm]
Q_E = 515.81 [gpm]
Q_G = 133.78 [gpm]
Re_C = 1110294.00 [adimensional]
Re_D = 1124704.49 [adimensional]
Re_E = 1676729.40 [adimensional]
Re_G = 676422.28 [adimensional]
f_C = 0.01991 [adimensional]
f_D = 0.01854 [adimensional]
f_E = 0.01982 [adimensional]
f_G = 0.02145 [adimensional]
```

Figura 19 – Outputs do modelo

4 RESULTADOS E DISCUSSÕES

Através das simulação numéricas feita em código, obtiveram-se os valores de α e β, além do gráfico de desempenho das curvas. Também foi possível calcular os valores de Q e p através de matemática simbólica com o *SymPy*. E por fim, o cálculo de Re e f através do *SciPy*. Para os gráficos ajustados, os pontos tiveram de ser definidos manualmente, sendo eles escolhidos através da grades dos eixos presentes na Figura 2.

Em resumo, com o avanço do código obtivemos os seguintes resultados:

Tabela 4 – Resultados encontrados

Bomba	α $[psi]$	β $[psi/gmp^2]$ $\times 10^{-6}$
A	166,840	3,439
B	149,926	3,726
F	133,013	3,726

Tabela 5 – Resultados encontrados - Vazão (Q)

Tubo	Q $[gpm]$
C	243.97
D	271.85
G	133.77
E	515.81

Tabela 6 – Resultados encontrados - Pressão (p)

Ponto	p $[psi]$
4	92.49
2	166.64
3	149.70
6	132.94

Tabela 7 – Resultados encontrados - Número de Reynolds (Re) e fator de atrito (f)

Tubo	Re [adimensional]	f [adimensional]
C	1110294.00	0.01991
D	1124704.49	0.01854
E	1676729.40	0.01982
G	676422.28	0.02145

5 CONCLUSÕES

A realização do relatório em conjunto com o diagrama de fluxo, foi de grande auxilio para o entendimento do problema proposto. O código atende aos requisitos do projeto, com comandos, tais como permitir que o usuário digite os valores pelo teclado. Para aprimorar o código, foi utilizadas as bibliotecas *SymPy* e *SciPy* do Python, efetuando cálculos de equações simultâneas e raízes de funções. Com o *framework* Jupyter disponibilizado pelo Google Colab, é possível adicionar textos e equações no script dos códigos, deixando o processo muito mais intuitivo e didático, além de ser mais fácil de realizar o processo de *debug*.

Considerando a proposta da entrega, obtive-se com êxito o cálculo dos parâmetros α e β das bombas A, B e F; as vazões, número de Reynolds e fator de atrito dos tubos C, D, E e G; e as pressões nos pontos 2, 3, 4 e 6.

REFERÊNCIAS

[1] BOHORQUEZ, W. I. *MEC070 – Fluid Machinery*. 2022. Último acesso: 07/10/2022. Disponível em: <https://www.ufjf.br/washington_irrazabal/teaching/mec008-sistemas-fluido-mecanicos/>.

[2] BOHORQUEZ, W. I. *Prova: Simulação Numérica*. 2022. Último acesso: 07/10/2022. Disponível em: <https://www.ufjf.br/washington_irrazabal/files/2014/03/Questão-Única_Simulação-Numérica_MEC070_2022-3.pdf>.

[3] GOOGLE. *Conheça o Colab*. 2022. Último acesso: 07/10/2022. Disponível em: <https://colab.research.google.com/>.

[4] ALURA. *Google Colab: o que é, tutorial de como usar e criar códigos*. 2022. Último acesso: 07/10/2022. Disponível em: <https://www.alura.com.br/artigos/google-colab-o-que-e-e-como-usar>.