notebook

November 27, 2021

1 Trabalho Final de Mecânica dos Sólidos:

1.1 PROJETO DE DUTOS PARA SISTEMA DE CONDICIONAMENTO DE AR

Alunos:

João Gabriel Schunk

João Henrique Lima de Vasconcelos

João Vitor Bordin

1.2 Introdução

A resolução do problema será seccionada em uma série de submódulos como mostra o diagrama abaixo:

1.2.1 Perdas de Carga

Começa-se pela resolução das perdas de carga em cada uma das linhas e em A. A perda de carga na linha 2-3A é igual a perda de carga na linha 2-3B assim, a perda de carga em B não é calculada.

As perdas de

1.2.2 Trabalho da Bomba

Com as perdas de carga, calcula-se o trabalho da bomba

1.2.3

[ ]:
```
%load_ext autoreload
%autoreload 2
```

[ ]:
```
import numpy as np
import inputs as i
import aux as aux
from scipy.optimize import minimize
from itertools import *
import pandas as pd
```

[ ]:
```
def cost_function(D_array, return_cost_dict=False):
    h_12, v_12 = aux.head_loss_12(i.Q, D_array[0], i.L12, i.Leqcot, i.Leqvalv,␣
↪i. , i.μ, i.ecom, return_velocity=True)
    h_23A, v_23A = aux.head_loss_23A(i.Q, D_array[1], i.L23A, i.Leqcot, i.
↪Leqvalv, i. , i.μ, i.ecom, return_velocity=True)
    h_A, v_A = aux.head_loss_A(i.Q, i.dA, i.LA, i.Leqcurv, i. , i.μ, i.e_tref,␣
↪return_velocity=True)
    h_31, v_31 = aux.head_loss_31(i.Q, D_array[3], i.L31, i.Leqcot, i.Leqvalv,␣
↪i. , i.μ, i.ecom, return_velocity=True)
    W_bomba = aux.calc_W_bomba(h_12, h_23A, h_A, h_31, i. , i.Q)

    cost = aux.calc_C_total(D_array, i.L12, i.L23A, i.L23B, i.L31, i.b, i.F, i.
↪C2, i.t, W_bomba, i.n, i.a)

    cost_dict={"cost": cost,
               "h_12": h_12,
               "v_12": v_12,
               "h_23A": h_23A,
               "v_23A": v_23A,
               "h_A": h_A,
               "v_A": v_A,
               "h_31": h_31,
               "v_23A": v_31,
               "W_bomba": W_bomba}

    if return_cost_dict:
        return cost_dict

    else:
        return cost
```

1.3 Valores iniciais dos diâmetros

[ ]:
```
D_guess = np.zeros(4)
D_guess[0] = 0.0525 # [m]
D_guess[1] = 0.05252 # [m]
D_guess[2] = 0.01604 # [m]
D_guess[3] = 0.06271 # [m]
D_guess
```

[ ]:
```
array([0.0525 , 0.05252, 0.01604, 0.06271])
```

[ ]:
```
sol = minimize(cost_function, D_guess, bounds=[(0.001, 0.5),(0.001, 0.5),(0.
↪001, 0.5),(0.001, 0.5)])

print("Custo total = R$",sol["fun"])
```

```
Custo total = R$ 1400497.1616601162
```

[ ]:
```
ideal_diameters = sol["x"]/2.54e-2
ideal_diameters
```

[ ]:
```
array([1.68546489, 1.44791787, 0.03937008, 1.87702223])
```

2 Diametros comerciais

A norma NBR 5590 (ASTM A-53) / A106 A / API 5L B dita quais os possíveis diâmetros comerciais a tabela a seguir referencia todos os diâmetros: https://acotubo.com.br/tabelas-site/tubos-de-aco/tubos-de-conducao-com-e-sem-costura-nbr-5590-astm-53-a106-api-5l-b.html

[ ]:
```
comercial_diameters = pd.read_excel("diametros_comerciais.xlsx", skiprows =␣
↪[1,2], usecols=range(1))
comercial_diameters.dropna(subset=["Diâmetro"], inplace=True)

comercial_diameters = comercial_diameters.to_numpy()
#comercial_diameters.flatten()
possible_diameters = []
for diameter in comercial_diameters.flatten():
    diameter = eval(diameter.replace("'", "").replace(".", "+"))
    possible_diameters.append(diameter)

df0 = pd.DataFrame()
df0["Diametros Comerciais"] = possible_diameters

display(df0)
```

	Diametros Comerciais
0	0.250
1	0.375
2	0.500
3	0.750
4	1.000
5	1.250
6	1.500
7	2.000
8	2.500
9	3.000
10	3.500
11	4.000
12	5.000
13	6.000
14	8.000
15	10.000
16	12.000

```
17                14.000
18                16.000
19                18.000
20                20.000
21                22.000
22                24.000
```

[ ]:
```
all_configurations = [item for item in product(possible_diameters,␣
 ↪repeat=len(D_guess))]

all_configurations = np.array([list(elem) for elem in all_configurations])
```

[ ]:
```
display(pd.DataFrame(all_configurations,
                columns=["Diametro da Linha 1-2 [in]",
                         "Diametro da Linha 2-3A [in]",
                         "Diametro da Linha 2-3B [in]",
                         "Diametro da Linha 3-1 [in]"]))
```

```
        Diametro da Linha 1-2 [in]  Diametro da Linha 2-3A [in]  \
0                             0.25                          0.25
1                             0.25                          0.25
2                             0.25                          0.25
3                             0.25                          0.25
4                             0.25                          0.25
...                            ...                           ...
279836                       24.00                         24.00
279837                       24.00                         24.00
279838                       24.00                         24.00
279839                       24.00                         24.00
279840                       24.00                         24.00

        Diametro da Linha 2-3B [in]  Diametro da Linha 3-1 [in]
0                             0.25                         0.250
1                             0.25                         0.375
2                             0.25                         0.500
3                             0.25                         0.750
4                             0.25                         1.000
...                            ...                           ...
279836                       24.00                        16.000
279837                       24.00                        18.000
279838                       24.00                        20.000
279839                       24.00                        22.000
279840                       24.00                        24.000

[279841 rows x 4 columns]
```

[ ]:
```
costs=[]
for D_array in all_configurations:
    cost = cost_function(2.54e-2*D_array)
    costs.append(cost)
costs = np.array(costs)
```

[ ]:
```
df = pd.DataFrame(all_configurations,
                  columns=["Diametro da Linha 1-2 [in]",
                           "Diametro da Linha 2-3A [in]",
                           "Diametro da Linha 2-3B [in]",
                           "Diametro da Linha 3-1 [in]"])
df["Custo total da linha [R$]"] = costs
df.index.name = "Iteração"

result = df.sort_values("Custo total da linha [R$]", ascending=True)

display(result)
```

Iteração	Diametro da Linha 1-2 [in]	Diametro da Linha 2-3A [in]
76183	1.50	1.50
88350	2.00	1.50
76206	1.50	1.50
88373	2.00	1.50
76229	1.50	1.50
...	...	...
414	0.25	0.25
437	0.25	0.25
460	0.25	0.25
483	0.25	0.25
506	0.25	0.25

Iteração	Diametro da Linha 2-3B [in]	Diametro da Linha 3-1 [in]
76183	0.250	2.00
88350	0.250	2.00
76206	0.375	2.00
88373	0.375	2.00
76229	0.500	2.00
...	...	...
414	16.000	0.25
437	18.000	0.25
460	20.000	0.25
483	22.000	0.25
506	24.000	0.25

Iteração	Custo total da linha [R$]

```
Iteração
76183                  1.410909e+06
88350                  1.412848e+06
76206                  1.414002e+06
88373                  1.415941e+06
76229                  1.417094e+06
...                             ...
414                    7.530679e+08
437                    7.531174e+08
460                    7.531669e+08
483                    7.532164e+08
506                    7.532659e+08

[279841 rows x 5 columns]
```

[ ]:
```
comercial_diameters = result.iloc[0][0:-1].to_numpy()*2.54e-2

comercial_diameters/2.54e-2
```

[ ]:
```
array([1.5 , 1.5 , 0.25, 2.  ])
```

[ ]:
```
ideal_diameters
```

[ ]:
```
array([1.68546489, 1.44791787, 0.03937008, 1.87702223])
```

[ ]:
```
cost_function(ideal_diameters, return_cost_dict=True)
```

[ ]:
```
{'cost': 11443964.327029698,
 'h_12': 1.1947332165799362e-05,
 'v_12': 0.0012449966978261189,
 'h_23A': 1.6342986379892594e-05,
 'v_23A': 0.0010038499145793775,
 'h_A': 4098.118273613894,
 'v_A': 13.156834942599902,
 'h_31': 2.6634495604864042e-05,
 'W_bomba': 11383.662023718636}
```

[ ]:
```
cost_function(comercial_diameters, return_cost_dict=True)
```

[ ]:
```
{'cost': 1410909.0619030888,
 'h_12': 111.1541886991869,
 'v_12': 2.4364509152962786,
 'h_23A': 55.88946801177149,
 'v_23A': 1.3705036398541564,
 'h_A': 4098.118273613894,
 'v_A': 13.156834942599902,
 'h_31': 58.75812743620264,
```

```
 'W_bomba': 12010.889049336265}
```

[ ]:
```
h_12, v12 = aux.head_loss_12(i.Q, comercial_diameters[0], i.L12, i.Leqcot, i.
↪Leqvalv, i. , i.μ, i.ecom, return_velocity=True)
h_23A, v23A = aux.head_loss_23A(i.Q, comercial_diameters[1], i.L23A, i.Leqcot,␣
↪i.Leqvalv, i. , i.μ, i.ecom, return_velocity=True)
h_23B, v23B = aux.head_loss_23B(i.Q, comercial_diameters[2], i.L23B, i.Leqcot,␣
↪i.Leqvalv, i. , i.μ, i.ecom, return_velocity=True)

h_31, v31 = aux.head_loss_31(i.Q, comercial_diameters[3], i.L31, i.Leqcot, i.
↪Leqvalv, i. , i.μ, i.ecom, return_velocity=True)
```

[ ]:
```
print("VA = ", v23A)
print("V31 = ", v31)
```

```
VA =  1.461870549177767
V31 =  1.3705036398541564
```

[ ]:
```
Q_A = v23A*(np.pi*(comercial_diameters[1]**2))/4
Q_B = v23B*(np.pi*(comercial_diameters[2]**2))/4

Q_12 = v12*(np.pi*(comercial_diameters[0]**2))/4
```

[ ]:
```
print(Q_A, Q_12, Q_B)
```

```
0.0016666666666666668 0.0027777777777777783 0.0011111111111111111
```