Sur le dimentionement d'une ventilation en millieux homogène

Gaggini lorenzo et Ferreti Thibault

Dans ce rapport, en s'imspirant d'un précédent travaille sur l'isolation thermique, on cherche à minimisé le coup electrique d'un system de ventilation, en fonction du rapport entre la température de l'objet chauffant et celle de l'air ambient.

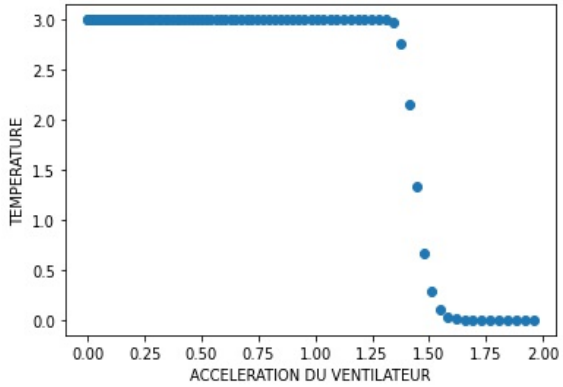
Pour ce faire, on introduit la notion d'optimum de pareto, qui, étant donnée un ensemble de point, nous permet de trouvé un sous ensemble de point limite noté front de pareto, qui nous serviront à déterminé la condition minimale rechercher.

Dans l'application qui suit, la température de l'objet chauffant est donné par la fonction

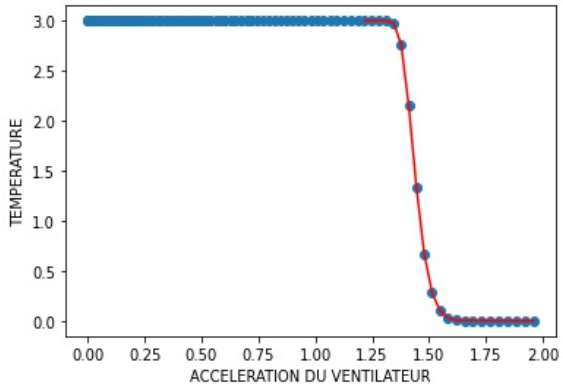
$f(x) = 3 \times e^{-1 \times 10e(x-0,5)^8}$

Le flux d'air est suposé constant et agissant instantanemment sur le rapport de température,

Ainsi, en adaptant un algorithme de cacule de front de pareto aux donné offertent par le modéle utilisé dans notre précédent rapport sur l'isolation thermique, on produit un algorithme (page 2) nous permettant de visualiser les graphique suivant :



Le nuages de points formé par le flux de chaleur pour chaque niveau d'accélération du ventilateur



Le même graphique, avec en rouge le front de pareto

NB: l'algorithm nous donne également l'ensemble pris dans le front, et leur posision dans la liste des points

Algorithme pour le calcule d'un front de pareto pour un ensemble de point données

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

N=100
Fmax = 3   # max heat production source level
U=[]
V=[]
J=[]
F=[]
T=[]
scores = np.array([ [ None for y in range( 2 ) ] for x in range( N ) ])
for i in range(0,N):
    U.append(i/190)
    V.append(i/50)
for i in range(0,N):
    J.append(1/2*V[i]**2)

for j in range (0,N):
    F.append(np.exp(-1.e10*(U[j]-0.5)**8)*Fmax)  #RHS modelling heat source term (processor)
for k in range (0,N):
    T.append(min(3,-F[k]+3))
for a in range (0,N):
     scores[a]=([T[a],J[a]])
x = scores[:, 1]
y = scores[:, 0]

print('Tableau des points à évaluer',scores)
plt.scatter(x,y)
plt.ylabel('TEMPERATURE')
plt.xlabel('ACCELERATION DU VENTILATEUR')
plt.show()

def identify_pareto(scores):
    # Count number of items
    population_size = (scores.shape[0])
    # Create a NumPy index for scores on the pareto front (zero indexed)
    population_ids = np.arange(population_size)
    # Create a starting list of items on the Pareto front
    # All items start off as being labelled as on the Parteo front
    pareto_front = np.ones(population_size, dtype=bool)
    # Loop through each item. This will then be compared with all other items
    for i in range(population_size):
        # Loop through all other items
        for j in range(population_size):
            # Check if our 'i' pint is dominated by out 'j' point
            if all(scores[j] >= scores[i]) and any(scores[j] > scores[i]):
                # j dominates i. Label 'i' point as not on Pareto front
                pareto_front[i] = 0
                # Stop further comparisons with 'i' (no more comparisons needed)
                break
    # Return ids of scenarios on pareto front
    return population_ids[pareto_front]

pareto = identify_pareto(scores)
print ('Points on Pareto front: \n',pareto)

pareto_front = scores[pareto]
print ('\nPareto front scores')
print (pareto_front)

pareto_front_df = pd.DataFrame(pareto_front)
pareto_front_df.sort_values(0, inplace=True)
pareto_front = pareto_front_df.values

x_all = scores[:, 1]
y_all = scores[:, 0]
x_pareto = pareto_front[:, 1]
y_pareto = pareto_front[:, 0]


plt.scatter(x_all, y_all)
plt.plot(x_pareto, y_pareto, color='r')
plt.xlabel('ACCELERATION DU VENTILATEUR')
plt.ylabel('TEMPERATURE')
plt.show()
```