Universidad de Zaragoza

Escuela de Ingeniería y Arquitectura

Anexos - Caracterización de la cristalización del xilitol para su uso como material de cambio de fase mediante técnicas reológicas

Trabajo Fin de Grado

Grado de Ingeniería de Tecnologías Industriales

Autora: **Marta Martí Sánchez**
Director: **Miguel Navarro García**
Codirectora: **Mónica Delgado Gracia**

Julio 2024

Anexo 1

0.1. Extracción de datos

```
import pandas as pd;
import numpy as np;
from matplotlib import pyplot as plt;
import seaborn as sns;
from lmfit.models import StepModel, LinearModel;
import os;

Induction_Time = [];
Initial_Viscosity = [];
Final_Viscosity = [];
Temperature = [];
Gap = [];
Shear_Rate = [];
Stress = [];
Normal_Stress = [];
File = [];
GAP_ref = [];
Temperature_ref = [];
Shear_Rate_ref = [];

ubic = 'C:\\Users\\Marta Martí\\OneDrive\\Escritorio\\TFG\\ENSAYOS';
for root, dirs, files in os.walk(ubic):
    for file in files:
        df = pd.read_excel(ubic + '\\' + file, sheet_name='Peak hold - 1', header=[0, 1, 2]);
        print(file);

        df = df.droplevel(level=0, axis=1);
        df = df.droplevel(level=1, axis=1);

        log_visc = np.log(df['Viscosity']);
        df['Log_visc'] = log_visc;

        plt.plot(df['Viscosity'], label='Viscosity');
        plt.yscale('log');
        plt.title(file);
        plt.legend();
        plt.show();

        index = int(input('cortar: '));
        if len(df) > index:
            df = df.iloc[:index];

        step_mod = StepModel(form='logistic', prefix='step_');
        line_mod = LinearModel(prefix='line_');
        step2_mod = StepModel(form='arctan', prefix='step2_');
        pars = step2_mod.guess(df['Log_visc'], x=df.index) + line_mod.guess(df['Log_visc'], x=df.index) + step_mod.guess(df['Log_visc'], x=df.index);
        mod = step2_mod + line_mod + step_mod;
```

```
        out = mod.fit(df['Log_visc'], pars, x=df.index);

        Ajuste = out.best_fit;
        derivada_Ajuste = np.gradient(Ajuste, df.index);

        x = [2 * i for i in range(len(Ajuste))]
        f, ax1 = plt.subplots()
        ax2 = ax1.twinx()
        ax1.plot(x, Ajuste, label='Ajuste')
        ax1.plot(x, df['Log_visc'], label='Viscosity')
        ax2.plot(x, derivada_Ajuste, label='Deriv', c='r')
        ax1.set_xlabel('Step time [s]')
        ax1.set_ylabel('Logarithmic Viscosity')
        ax2.set_ylabel('Derivative')
        ax1.legend(loc='upper left')
        ax2.legend(loc='upper right')
        plt.show()


        y_der_inflex = np.max(derivada_Ajuste);
        x_inflex = np.argmax(derivada_Ajuste);

        log_visc_inflex = df['Log_visc'][x_inflex];
        log_visc_ini = df['Log_visc'][3];

        C_inflex = log_visc_inflex - x_inflex * y_der_inflex;

        x_ind = (log_visc_ini - C_inflex) / y_der_inflex;
        x_ind = round(x_ind);

        t_ind = df['Step time'][x_ind];

        plt.plot(df['Step time'], df['Log_visc']);
        plt.axvline(t_ind, color='grey', ls='--', alpha=0.8);
        plt.title(file);
        plt.show();

        File.append(file);
        Induction_Time.append(t_ind);
        Initial_Viscosity.append(df['Viscosity'][3]);
        Final_Viscosity.append(df['Viscosity'][x_ind]);
        Temperature.append(df['Temperature'][x_ind]);
        Gap.append(df['Gap'][x_ind]);
        Shear_Rate.append(df['Shear rate'][x_ind]);
        Stress.append(df['Stress'][x_ind]);
        Normal_Stress.append(df['Normal stress'][x_ind]);

        if df['Gap'][x_ind] > 750:
            GAP_ref.append(2);
        else:
            GAP_ref.append(1);

        if 68 < df['Temperature'][x_ind] < 72:
            Temperature_ref.append(70);
        elif 73 < df['Temperature'][x_ind] < 77:
            Temperature_ref.append(75);
        elif 78 < df['Temperature'][x_ind] < 82:
            Temperature_ref.append(80);
```

```
        elif 83 < df['Temperature'][x_ind] < 87:
            Temperature_ref.append(85);
        elif 88 < df['Temperature'][x_ind] < 92:
            Temperature_ref.append(90);

        if df['Shear rate'][x_ind] > 98:
            Shear_Rate_ref.append(100);
        elif df['Shear rate'][x_ind] < 5:
            Shear_Rate_ref.append(1);
        elif 5 < df['Shear rate'][x_ind] < 50:
            Shear_Rate_ref.append(10);

print(Temperature_ref);

df_salida = pd.DataFrame();
df_salida['File'] = File;
df_salida['Induction_Time'] = Induction_Time;
df_salida['Initial_Viscosity'] = Initial_Viscosity;
df_salida['Final_Viscosity'] = Final_Viscosity;
df_salida['Temperature'] = Temperature;
df_salida['Gap'] = Gap;
df_salida['Shear_Rate'] = Shear_Rate;
df_salida['Stress'] = Stress;
df_salida['Normal_Stress'] = Normal_Stress;
df_salida['GAP_ref'] = GAP_ref;
df_salida['Temperature_ref'] = Temperature_ref;
df_salida['Shear_Rate_ref'] = Shear_Rate_ref;

print(df_salida);

with pd.ExcelWriter('C:\\Users\\Marta
    ↪ Martí\\OneDrive\\Escritorio\\TFG\\df_salida_rep.xlsx', mode='a',
    ↪ if_sheet_exists='overlay') as writer:
    df_salida.to_excel(writer, sheet_name='datos',
    ↪ startrow=writer.sheets['datos'].max_row, index=False,
    ↪ header=False);
```

0.2. Tasa de nucleación con tiempo de inducción

```
import pandas as pd;
import numpy as np;
from matplotlib import pyplot as plt;
import seaborn as sns;
from lmfit.models import StepModel, LinearModel;
import os;
import math;

x = 1000;

if x == 300:
    df3 = pd.read_excel('C:\\Users\\Marta
    ↪ Martí\\OneDrive\\Escritorio\\TFG\\df_salida.xlsx',
    ↪ sheet_name='datos', header=[0]);
elif x == 600:
    df3 = pd.read_excel('C:\\Users\\Marta
    ↪ Martí\\OneDrive\\Escritorio\\TFG\\df_salida_600.xlsx',
```

```
    ↪ sheet_name='datos', header=[0]);
elif x == 1000:
    df3 = pd.read_excel('C:\\Users\\Marta
    ↪ Martí\\OneDrive\\Escritorio\\TFG\\df_salida_rep.xlsx',
    ↪ sheet_name='datos', header=[0]);

df_aux = pd.DataFrame();
df2 = pd.DataFrame();
df2_600 = [];
df2_300 = [];
df2_rep = [];
w = 1;
gap = 1;
print('Número de ensayos tratados:', len(df3));

G = 3.5 * 10**(-6);
pi = math.pi;
i = 0;
mu = 2.67;
alpha_max = 0.57;

for j in range(12):
    alpha_crist = [];
    Nucleation_rate = [];
    temp = [];
    nucl = [];

    if j <= 2:
        df2 = df3.drop(df3[df3['Shear_Rate_ref'] != w].index);
        df2 = df2.reset_index(drop=True);
        print('Shear Rate:', w);
        w *= 10;
    elif 2 < j <= 4:
        df2 = df3.drop(df3[df3['GAP_ref'] != gap].index);
        df2 = df2.reset_index(drop=True);
        print('GAP:', gap);
        gap += 1;
    elif 4 < j <= 10:
        if w == 1000:
            w = 1;
        if gap == 3:
            gap = 1;
        df2 = df3.drop(df3[df3['GAP_ref'] != gap].index);
        df2 = df2.drop(df2[df2['Shear_Rate_ref'] != w].index);
        df2 = df2.reset_index(drop=True);
        print('GAP:', gap, 'Shear rate:', w);
        if w == 100:
            gap += 1;
        w *= 10;
    elif j == 11:
        df2 = df3;

    for i in range(0, len(df2)):
        alp = alpha_max * (1 - ((df2['Final_Viscosity'][i]) /
    ↪ df2['Initial_Viscosity'][i]) ** (-1 / (mu * alpha_max)));
        alpha_crist.append(alp);
        nucl = (3 * alp) / (G ** 3 * (df2['Induction_Time'][i]) ** 4 *
    ↪ pi);
```

```
68          Nucleation_rate.append(nucl);
69
70      df2['alpha_crist'] = alpha_crist;
71      df2['Nucleation_rate'] = Nucleation_rate;
72
73      df2 = df2.sort_values(by=['Temperature_ref', 'GAP_ref',
    ↪ 'Shear_Rate_ref']);
74
75      df_aux = df2.loc[df2['alpha_crist'] <= 0];
76      df2 = df2.drop(df2[df2['alpha_crist'] <= 0].index);
77
78      df2 = df2.reset_index(drop=True);
79
80      temp = df2['Temperature_ref'];
81      nucl = df2['Nucleation_rate'];
82
83      if not temp.empty and not nucl.empty:
84          if x == 300:
85              df2_300.append(df2);
86          elif x == 600:
87              df2_600.append(df2);
88          elif x == 1000:
89              df2_rep.append(df2);
90
91      if x == 300:
92          with pd.ExcelWriter("C:\\Users\\Marta
    ↪ Martí\\OneDrive\\Escritorio\\TFG\\datos
    ↪ experimentales\\df2_300_all.xlsx") as writer:
93              for i, df in enumerate(df2_300, 1):
94                  df.to_excel(writer, sheet_name="Sheet{}".format(i),
    ↪ index=False);
95      elif x == 600:
96          with pd.ExcelWriter("C:\\Users\\Marta
    ↪ Martí\\OneDrive\\Escritorio\\TFG\\datos
    ↪ experimentales\\df2_600_all.xlsx") as writer:
97              for i, df in enumerate(df2_600, 1):
98                  df.to_excel(writer, sheet_name="Sheet{}".format(i),
    ↪ index=False);
99      elif x == 1000:
100         if df2_rep:
101             with pd.ExcelWriter("C:\\Users\\Marta
    ↪ Martí\\OneDrive\\Escritorio\\TFG\\datos
    ↪ experimentales\\df2_rep_all.xlsx") as writer:
102                 for i, df in enumerate(df2_rep, 1):
103                     df.to_excel(writer, sheet_name="Sheet{}".format(i),
    ↪ index=False);
104
105     if not temp.empty and not nucl.empty:
106         fig, ax = plt.subplots();
107         ax.scatter(x=temp, y=nucl);
108         plt.yscale('log');
109         plt.xlabel('Temperatura');
110         plt.ylabel('Nucleation rate');
111         if j <= 2:
112             plt.title('Shear rate {}'.format(w / 10));
113         elif 2 < j <= 4:
114             plt.title('GAP {}'.format(gap - 1));
115         elif 4 < j <= 10:
```

```
            plt.title('GAP and Shear Rate');
        elif j == 11:
            plt.title('ALL');
        plt.show();
```

0.3. Obtención de parámetros tasa de nucleación

```
import pandas as pd;
import numpy as np;
from matplotlib import pyplot as plt;
import seaborn as sns;
from lmfit.models import StepModel, LinearModel;
import os;
import math;
from sklearn.linear_model import LinearRegression;
from lmfit.models import LinearModel;

x = 600;
if x == 300:
    df3 = pd.read_excel('C:\\Users\\Marta Martí\\OneDrive\\Escritorio\\TFG\\datos experimentales\\df2_300_all.xlsx', sheet_name='Sheet12', index_col=0);
elif x == 600:
    df3 = pd.read_excel('C:\\Users\\Marta Martí\\OneDrive\\Escritorio\\TFG\\datos experimentales\\df2_600_all.xlsx', sheet_name='Sheet8', index_col=0);
elif x == 1000:
    df3 = pd.read_excel('C:\\Users\\Marta Martí\\OneDrive\\Escritorio\\TFG\\datos experimentales\\df2_rep_all.xlsx', sheet_name='Sheet4', index_col=0);

A = [];
B = [];
df_aux_XY = pd.DataFrame();
Tm = 92.4;
ln = 0;
w = 1;
gap = 2;
df4 = pd.DataFrame(index=None);

for j in range(12):
    X = [];
    Y = [];

    if j <= 2:
        df4 = df3.drop(df3[df3['Shear_Rate_ref'] != w].index);
        df4 = df4.reset_index(drop=True);
        print(w);
        w *= 10;
    elif 2 < j <= 4:
        df4 = df3.drop(df3[df3['GAP_ref'] != gap].index);
        df4 = df4.reset_index(drop=True);
        print(gap);
```

```
        gap -= 1;
    elif 4 < j <= 10:
        if w == 1000:
            w = 1;
        if gap == 0:
            gap = 2;
        df4 = df3.drop(df3[df3['GAP_ref'] != gap].index);
        df4 = df4.drop(df4[df4['Shear_Rate_ref'] != w].index);
        df4 = df4.reset_index(drop=True);
        print(gap, w);
        if w == 100:
            gap -= 1;
        w *= 10;
    elif j == 11:
        df4 = df3;

    print(df4);
    print(len(df4));

    for i in range(len(df4)):
        T3 = df4['Temperature_ref'][i] * (Tm -
    ↪ (df4['Temperature_ref'][i]) ** 2);
        X.append(T3);
        ln = math.log((df4['Nucleation_rate'][i]) *
    ↪ (df4['Initial_Viscosity'][i]));
        Y.append(T3 * ln);

    line_mod = LinearModel(prefix='line_');
    if X and Y:
        pars = line_mod.guess(Y, x=X);
        mod = line_mod;
        out = mod.fit(Y, pars, x=X);
        Ajuste = out.best_fit;
        Slope = out.params['line_slope'].value;
        term_ind = out.params['line_intercept'].value;

        plt.scatter(X, Y, label='Datos Originales');
        plt.plot(X, Ajuste, color='red', label='Ajuste Lineal');
        plt.xlabel('X');
        plt.ylabel('Y');
        plt.legend();
        if j <= 2:
            plt.title('Shear rate');
        elif 2 < j <= 4:
            plt.title('GAP');
        elif 4 < j <= 10:
            plt.title('GAP and Shear Rate');
        elif j == 11:
            plt.title('ALL');
        plt.show();

        print(Slope);

        A.append(math.exp(Slope));
        B.append(-term_ind);

df_aux_XY['X'] = X;
df_aux_XY['Y'] = Y;
```

```
97
98 AB = pd.DataFrame({'A': A, 'B': B});
99 print(AB);
100
101 if x == 300:
102     AB.to_excel("C:\\Users\\Marta
    ↪ Martí\\OneDrive\\Escritorio\\TFG\\AB.xlsx", sheet_name="Sheet1");
103     df_aux_XY.to_excel("C:\\Users\\Marta
    ↪ Martí\\OneDrive\\Escritorio\\TFG\\datos
    ↪ experimentales\\XY_300.xlsx", sheet_name="Sheet1");
104 elif x == 600:
105     AB.to_excel("C:\\Users\\Marta
    ↪ Martí\\OneDrive\\Escritorio\\TFG\\AB_600.xlsx",
    ↪ sheet_name="Sheet1");
106     df_aux_XY.to_excel("C:\\Users\\Marta
    ↪ Martí\\OneDrive\\Escritorio\\TFG\\datos
    ↪ experimentales\\XY_600.xlsx", sheet_name="Sheet1");
107 elif x == 1000:
108     AB.to_excel("C:\\Users\\Marta
    ↪ Martí\\OneDrive\\Escritorio\\TFG\\AB_rep.xlsx",
    ↪ sheet_name="Sheet1");
109     df_aux_XY.to_excel("C:\\Users\\Marta
    ↪ Martí\\OneDrive\\Escritorio\\TFG\\datos
    ↪ experimentales\\XY_rep.xlsx", sheet_name="Sheet1");
```

0.4. Tasa de nucleación en función de la temperatura

```
1 import pandas as pd;
2 import numpy as np;
3 from matplotlib import pyplot as plt;
4 import seaborn as sns;
5 from lmfit.models import StepModel, LinearModel;
6 import os;
7 import math;
8
9 x = 1000;
10 if x == 300:
11     params = pd.read_excel('C:\\Users\\Marta
    ↪ Martí\\OneDrive\\Escritorio\\TFG\\AB.xlsx', sheet_name='Sheet1',
    ↪ index_col=0);
12 elif x == 600:
13     params = pd.read_excel('C:\\Users\\Marta
    ↪ Martí\\OneDrive\\Escritorio\\TFG\\AB_600.xlsx',
    ↪ sheet_name='Sheet1', index_col=0);
14 elif x == 1000:
15     params = pd.read_excel('C:\\Users\\Marta
    ↪ Martí\\OneDrive\\Escritorio\\TFG\\AB_rep.xlsx',
    ↪ sheet_name='Sheet1', index_col=0);
16
17 T = np.linspace(60, 90, 31);
18 Tm = 92.4;
19
20 A = params['A'].tolist();
21 B = params['B'].tolist();
22 Jt = pd.DataFrame();
23 print("Longitud de parámetros:", len(params));
```

```
24
25 for j in range(len(params)):
26     A1 = A[j];
27     B1 = B[j];
28     J = [];
29
30     for i in range(len(T)):
31         mu = 0.0079 * 10**(-10) * math.exp(9955 / (T[i] + 273.15));
32         J.append((A1 / mu) * math.exp(-B1 / (T[i] * (Tm - T[i]) ** 2)));
33
34     Jt['Nucleation_Rate'] = J;
35     Jt['Temperature_ref'] = T;
36
37     plt.plot(T, J, label=f'Lista {j + 1}');
38
39     if x == 300:
40         Jt.to_excel("C:\\Users\\Marta
   ↪ Martí\\OneDrive\\Escritorio\\TFG\\datos
   ↪ teoricos\\df4_{}.xlsx".format(j + 1), index=False,
   ↪ sheet_name="Sheet1");
41     elif x == 600:
42         Jt.to_excel("C:\\Users\\Marta
   ↪ Martí\\OneDrive\\Escritorio\\TFG\\datos
   ↪ teoricos\\df4_600_{}.xlsx".format(j + 1), index=False,
   ↪ sheet_name="Sheet1");
43     elif x == 1000:
44         Jt.to_excel("C:\\Users\\Marta
   ↪ Martí\\OneDrive\\Escritorio\\TFG\\datos
   ↪ teoricos\\df4_rep_{}.xlsx".format(j + 1), index=False,
   ↪ sheet_name="Sheet1");
45
46 plt.xlabel('Índice');
47 plt.ylabel('Valor');
48 plt.yscale('log');
49 plt.title('Gráfico de 12');
50 plt.legend();
51 plt.show();
```

0.5. Comparación de las tasas de nucleación

```
1 import pandas as pd;
2 import numpy as np;
3 from matplotlib import pyplot as plt;
4 import seaborn as sns;
5 from lmfit.models import StepModel, LinearModel;
6 import os;
7 import math;
8
9 caso = 600;
10
11 def plot_data(file_type, rep):
12     if file_type == 300:
13         file_pattern = 'C:\\Users\\Marta
   ↪ Martí\\OneDrive\\Escritorio\\TFG\\datos
   ↪ experimentales\\df2_300_all.xlsx';
14     elif file_type == 600:
```

```
        file_pattern = 'C:\\Users\\Marta Martí\\OneDrive\\Escritorio\\TFG\\datos experimentales\\df2_600_all.xlsx';

    print(file_pattern);

    for j in range(rep):
        exp = pd.read_excel(file_pattern, sheet_name='Sheet{}'.format(j + 1));

        if caso == 300:
            teo = pd.read_excel('C:\\Users\\Marta Martí\\OneDrive\\Escritorio\\TFG\\datos teoricos\\df4_{}.xlsx'.format(j + 1), sheet_name='Sheet1');
        elif caso == 600:
            teo = pd.read_excel('C:\\Users\\Marta Martí\\OneDrive\\Escritorio\\TFG\\datos teoricos\\df4_600_{}.xlsx'.format(j + 1), sheet_name='Sheet1');

        fig, ax = plt.subplots();
        ax.scatter(x=exp['Temperature_ref'], y=exp['Nucleation_rate'], label='experimental');
        ax.plot(teo['Temperature_ref'], teo['Nucleation_Rate'], color='red', label='teórico');
        plt.yscale('log');
        plt.xlabel('Temperatura');
        plt.ylabel('Nucleation rate');

        if caso == 300:
            if j <= 2:
                plt.title('Shear rate');
            elif 2 < j <= 4:
                plt.title('GAP');
            elif 4 < j <= 10:
                plt.title('GAP and Shear Rate');
            elif j == 11:
                plt.title('ALL');
        elif caso == 600:
            if j <= 2:
                plt.title('Shear rate');
            elif j == 3:
                plt.title('GAP');
            elif 4 <= j <= 6:
                plt.title('GAP and Shear Rate');
            elif j == 7:
                plt.title('ALL');

        plt.legend();
        plt.show();

if caso == 300:
    rep = 12;
elif caso == 600:
    rep = 8;

plot_data(caso, rep);
```

0.6. Método estadístico

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.neighbors import KernelDensity
from matplotlib.patches import Patch
from matplotlib.lines import Line2D
from sklearn.linear_model import LinearRegression


ubic = 'C:\\Users\\GITSE\\Downloads\\TODOS_ANOVA.csv'
df = pd.read_csv(ubic, delimiter = ';')
pd.set_option('display.max_columns', None)

df = df.dropna(axis=0, how='all')




def indicator_function(x, y):
    if x >= y:
        return 1
    else:
        return 0
def curva_prob(tiempos_de_induccion):
    min_tiempo = min(tiempos_de_induccion)-10
    max_tiempo = max(tiempos_de_induccion)+200
    probabilidades = []
    current_tiempo_values = []
    paso = (max_tiempo-min_tiempo)/1000
    current_tiempo = min_tiempo
    while current_tiempo <= max_tiempo:
        suma_indicadores = sum(indicator_function(current_tiempo,
   ↪ tiempo) for tiempo in tiempos_de_induccion)
        probabilidad = suma_indicadores / len(tiempos_de_induccion)
        probabilidades.append(round(probabilidad,4))
        current_tiempo_values.append(current_tiempo)
        current_tiempo += paso

    probs = list(set(probabilidades))
    probs.sort()
    print(type(probs))
    index_l=[]
    times = []
    for prob in probs:
        index_l.append(probabilidades.index(prob))
    for i in index_l:
        times.append(current_tiempo_values[i])

    return current_tiempo_values, probabilidades
    #return times, probs


temperaturas = pd.unique(df['Temperature_ref'])
print(df.columns)
print(pd.unique(df['GAP_ref']))
```

```
print(pd.unique(df['Shear_Rate_ref']))
print(temperaturas)
color = ['r','b','g','k','purple']
i=0
J=[]
tg=[]
for temp in temperaturas:

    if temp == 75:
        i+=1
        continue
    if temp == 85:
        i+=1
        continue


    df_filt = df[df['Temperature_ref']==temp]
     df_filt = df_filt[df_filt['GAP_ref']==1]
    df_filt = df_filt[df_filt['Shear_Rate_ref']==10]

    tiempos_ind = np.array(df_filt['Induction_Time'])
    tiempo,prob = curva_prob(tiempos_ind)

    ###### Modelo ######
    # Step 2: Calculate the ECDF
    sorted_induction_times = sorted(tiempos_ind)
    ecdf = np.arange(len(sorted_induction_times)) / len(tiempos_ind)

    # Step 4: Transform the ECDF
    y = -np.log(1 - ecdf)

    # Create the design matrix X for linear regression
    t_array = np.array(sorted_induction_times).reshape(-1, 1)
    X = t_array

    # Step 5: Fit the linear regression model
    model = LinearRegression()
    model.fit(X, y)

    # The parameters A and B are now available in the model's
    ↪ coefficients
    A = model.coef_[0]
    B = -model.intercept_ / A

    # Calculate the model
    t_plot = tiempo
    P_t_plot = 1 - np.exp(-A * (t_plot - B))
    P_t_plot2 = [x if x>0 else 0 for x in P_t_plot]
    J.append(-A)
    tg.append(B)

    #plt.plot(tiempo,prob, label = str(int(temp))+'$^\circ$C', color =
    ↪ color[i])   #Descomentar para la comparación
    #plt.scatter(tiempo,prob,color = color[i])
    plt.scatter(sorted_induction_times, ecdf, color = color[i])
    plt.plot(t_plot, P_t_plot2, '-', color = color[i])
    i+=1
    sorted_data = np.sort(tiempos_ind)
```

```
    cumulative_density = np.linspace(0, 1, len(sorted_data))
    #plt.scatter(sorted_data, cumulative_density, label='CDF '+
    ↪ str(int(temp))+'$^\circ$C')
    #plt.plot(sorted_data, cumulative_density)
    file = open("list"+str(temp)+".csv", "w")
    for index in range(len(prob)):
        file.write(str(tiempo[index]) + ";" + str(prob[index]) + "\n")
file.close()
print(J)
plt.grid()
custom_lines = [Line2D([0], [0], color='gray', lw=2,label='ECDF and
    ↪ data'),
                Line2D([0], [0], color=color[0],
    ↪ lw=2,label='70'+'$^\circ$C'),
                Line2D([0], [0], color=color[1],
    ↪ lw=2,label='75'+'$^\circ$C'),
                Line2D([0], [0], color=color[2],
    ↪ lw=2,label='80'+'$^\circ$C'),
                Line2D([0], [0], color=color[3],
    ↪ lw=2,label='85'+'$^\circ$C'),
                Line2D([0], [0], color=color[4],
    ↪ lw=2,label='90'+'$^\circ$C')]

custom_lines = [Line2D([0], [0], color=color[0],
    ↪ lw=2,label='70'+'$^\circ$C'),
                Line2D([0], [0], color=color[2],
    ↪ lw=2,label='80'+'$^\circ$C'),
                Line2D([0], [0], color=color[4],
    ↪ lw=2,label='90'+'$^\circ$C')]


plt.ylabel('Probabilidad')
plt.xlabel('TIempo de inducción [s]')
plt.title('Probabilidad acumulada')
plt.legend(handles = custom_lines)
plt.show()
```