```python
# -*- coding: utf-8 -*-
"""
Created on Sat Aug 30 18:34:23 2025

@author: ugims
"""


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D  # noqa: F401
from scipy.interpolate import griddata
import matplotlib.patches as patches

# ==========================
# CONFIGURATION AND READING
# ==========================
file_path = r"C:/Users/ugims/inegi.up.pt/Teses & EstГЎgios - Teses_EstГЎgios - Miguel AntГinio Costa - Teses_EstГЎgios - Miguel AntГinio Costa/3. RepositГrio do Migue

# Mould's dimentions (### ALTARED)
MOLD_X = 900.0  # mm
MOLD_Z = 420.0  # mm

# Read file, ignore first 8 lines and for a ","
df = pd.read_csv(
    file_path,
    skiprows=8,
    sep=',',
    skip_blank_lines=True,
    engine='python',
    encoding='windows-1252'
)

# Clean names of all columns and spaces around data
df.columns = [col.strip() for col in df.columns]
df = df.applymap(lambda x: x.strip() if isinstance(x, str) else x)

# Convert columns into a numerical type
for col in df.columns:
    df[col] = pd.to_numeric(df[col], errors='coerce')

# Remove invalid lines
df.dropna(inplace=True)

# ==========================
# RESCALE TO 900 x 420 mm (### ALTARED)
# ==========================
# Consider original columns 'X (mm)' and 'Z (mm)' as coordenated data and
# rescale linearly to fit exactly withing [0, MOLD_X] and [0, MOLD_Z]
x_raw = df['X (mm)']
z_raw = df['Z (mm)']

x_min_raw, x_max_raw = x_raw.min(), x_raw.max()
z_min_raw, z_max_raw = z_raw.min(), z_raw.max()

# Avoid dividing by 0
if x_max_raw == x_min_raw:
    raise ValueError("X interval in file is Null; Rescaling impossible.")
if z_max_raw == z_min_raw:
    raise ValueError("Z interval is file is Null; Rescaling impossible.")

# Rescaling linear
df['X (mm)'] = (x_raw - x_min_raw) / (x_max_raw - x_min_raw) * MOLD_X
df['Z (mm)'] = (z_raw - z_min_raw) / (z_max_raw - z_min_raw) * MOLD_Z

# Shortcuts
x = df['X (mm)']
y = df['Z (mm)']  # Width (Mould's Z axis)
temperatura = df['Value (Celsius)']

# Show sample
print(df.head())

# (Opcional) Save clean file
df.to_csv("dados_superficie_organizados.csv", sep=';', index=False)

# ==========================
# 3D DISPERSION GRAPHIC
# ==========================
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
scatter = ax.scatter(x, y, temperatura, c=temperatura, cmap='plasma')

ax.set_xlabel('X (mm) - Comprimento (0вЂ“900)')
ax.set_ylabel('Z (mm) - Largura (0вЂ“420)')
ax.set_zlabel('Temperatura (В°C)')
ax.set_title('DistribuiГ§ГЈo de Temperatura na SuperfГcie Superior do Molde (900 Г— 420 mm)')

cbar = plt.colorbar(scatter, ax=ax, pad=0.1)
cbar.set_label('Temperatura (В°C)')

plt.tight_layout()
plt.show()

# ==========================
# 2D INTERPOLATED MAPS
# ==========================
xi = np.linspace(x.min(), x.max(), 200)
yi = np.linspace(y.min(), y.max(), 200)
xi, yi = np.meshgrid(xi, yi)
zi = griddata((x, y), temperatura, (xi, yi), method='cubic')

fig2 = plt.figure(figsize=(10, 8))
contour = plt.contourf(xi, yi, zi, levels=100, cmap='plasma')
plt.xlabel('X (mm) - Comprimento (0вЂ“900)')
plt.ylabel('Z (mm) - Largura (0вЂ“420)')
plt.title('Mapa 2D de Temperatura na SuperfГcie Superior (900 Г— 420 mm)')
cbar2 = plt.colorbar(contour)
cbar2.set_label('Temperatura (В°C)')
plt.tight_layout()
plt.show()

# ==========================
# ZONE DIVISION 3 Г— 2 (average per zone) вЂ” Kept (operate at scale 900 Г— 420)
# ==========================
x_min, x_max = 0.0, MOLD_X
z_min, z_max = 0.0, MOLD_Z

x_div = np.linspace(x_min, x_max, 4)  # 3 zones in X в†’ 4 limits
z_div = np.linspace(z_min, z_max, 3)  # 2 zones in Z в†’ 3 limits

def identificar_zona(xp, zp):
```

```python
    for i in range(3):  # X
        for j in range(2):  # Z
            if x_div[i] <= xp < x_div[i+1] and z_div[j] <= zp < z_div[j+1]:
                return j * 3 + i + 1
    # include limit dots for maximun superior limit
    if np.isclose(xp, x_max) and np.isclose(zp, z_max):
        return 6
    return np.nan

df['Zona'] = df.apply(lambda row: identificar_zona(row['X (mm)'], row['Z (mm)']), axis=1)

medias_por_zona = df.groupby('Zona')['Value (Celsius)'].mean().reset_index()
print("\nTemperatura mГ©dia por zona (B°C):")
for _, row in medias_por_zona.iterrows():
    print(f"Zona {int(row['Zona'])}: {row['Value (Celsius)']:.2f} B°C")

zona_temp_dict = dict(zip(medias_por_zona['Zona'], medias_por_zona['Value (Celsius)']))

# Map 3Г—2 drawn by averages
fig, ax = plt.subplots(figsize=(10, 8))
for i in range(3):  # X
    for j in range(2):  # Z
        zona_id = j * 3 + i + 1
        x_start = x_div[i]
        z_start = z_div[j]
        largura = x_div[i+1] - x_div[i]
        altura = z_div[j+1] - z_div[j]
        media_temp = zona_temp_dict.get(zona_id, np.nan)

        cor = plt.cm.plasma(
            (media_temp - medias_por_zona['Value (Celsius)'].min()) /
            (medias_por_zona['Value (Celsius)'].max() - medias_por_zona['Value (Celsius)'].min())
        )
        rect = patches.Rectangle((x_start, z_start), largura, altura, facecolor=cor, edgecolor='white', linewidth=1)
        ax.add_patch(rect)

        ax.text(
            x_start + largura / 2, z_start + altura / 2,
            f"{media_temp:.1f}B°C\nZona {zona_id}",
            color='white', ha='center', va='center', fontsize=7, weight='bold',
            bbox=dict(boxstyle='round,pad=0.3', facecolor='black', edgecolor='white', linewidth=0.8, alpha=0.6)
        )

ax.set_xlim(x_min, x_max)
ax.set_ylim(z_min, z_max)
ax.set_xlabel('X (mm) - Comprimento (0вЂ“900)')
ax.set_ylabel('Z (mm) - Largura (0вЂ“420)')
ax.set_title('Zonas de Aquecimento 3Г—2 com Temperatura MГ©dia (Cores Uniformes)')
sm = plt.cm.ScalarMappable(
    cmap='plasma',
    norm=plt.Normalize(vmin=medias_por_zona['Value (Celsius)'].min(),
                       vmax=medias_por_zona['Value (Celsius)'].max())
)
cbar = plt.colorbar(sm, ax=ax)
cbar.set_label('Temperatura MГ©dia (B°C)')
plt.grid(False)
plt.tight_layout()
plt.show()

# ==========================
# SUBDIVISION 15 Г— 4 (60 subzonas) вЂ” aligned to 15 longitudinal real zones
# ==========================
import matplotlib.patches as patches

# Real controls (mm) of 15 resistences (of drawing)
centros = [
    29.60, 88.80, 148.00, 207.20, 266.40,
    331.60, 390.80, 450.00, 509.20, 568.40,
    633.60, 692.80, 752.00, 811.20, 870.40
]
centros = sorted(centros)

# Irregular limits of 15 zones per average dot between centers
limites = [0.0]
for i in range(1, len(centros)):
    limites.append(0.5 * (centros[i-1] + centros[i]))
limites.append(900.0)
x_edges = np.array(limites, dtype=float)   # 16 borders в†’ 15 columns
z_edges = np.linspace(z_min, z_max, 5)     # 5 borders в†’ 4 lines

ncols = len(x_edges) - 1  # 15
nrows = len(z_edges) - 1  # 4

def identificar_zona_15x4(xp, zp, xed, zed):
    # coluna (X)
    i = None
    for ii in range(len(xed) - 1):
        if xed[ii] <= xp < xed[ii + 1]:
            i = ii
            break
    if i is None and np.isclose(xp, xed[-1]):
        i = len(xed) - 2  # include maximun

    # line (Z)
    j = None
    for jj in range(len(zed) - 1):
        if zed[jj] <= zp < zed[jj + 1]:
            j = jj
            break
    if j is None and np.isclose(zp, zed[-1]):
        j = len(zed) - 2  # include maximun

    if i is None or j is None:
        return np.nan
    return j * (len(xed) - 1) + i + 1  # line by line numeration

# Atribute subzone 15Г—4 to each dot
df['Zona_15x4'] = df.apply(lambda r: identificar_zona_15x4(r['X (mm)'], r['Z (mm)'], x_edges, z_edges), axis=1)

# Statistic per sobzone
estat_15x4 = df.groupby('Zona_15x4')['Value (Celsius)'].agg(['min', 'max', 'mean']).reset_index()
estat_15x4['Delta (max - min)'] = estat_15x4['max'] - estat_15x4['min']
estat_dict_15x4 = estat_15x4.set_index('Zona_15x4').to_dict(orient='index')

# GRAPHIC of 60 subzones
fig, ax = plt.subplots(figsize=(16, 9))

for i in range(ncols):         # 15 columms
    for j in range(nrows):     # 4 lines
        zona_id = j * ncols + i + 1
        x_start = x_edges[i]
        z_start = z_edges[j]
        largura = x_edges[i + 1] - x_edges[i]
```

```
        altura  = z_edges[j + 1] - z_edges[j]

        stats = estat_dict_15x4.get(zona_id)
        if stats:
            media_temp = stats['mean']
            delta_temp = stats['Delta (max - min)']
            cor = plt.cm.plasma(
                (media_temp - estat_15x4['mean'].min()) /
                (estat_15x4['mean'].max() - estat_15x4['mean'].min() + 1e-12)
            )
        else:
            media_temp = np.nan
            delta_temp = np.nan
            cor = 'gray'

        rect = patches.Rectangle((x_start, z_start), largura, altura,
                                 facecolor=cor, edgecolor='black', linewidth=0.5)
        ax.add_patch(rect)

        ax.text(
            x_start + largura / 2, z_start + altura / 2,
            f"{media_temp:.1f}В°C\nО”T={delta_temp:.1f}В°C\nZona {zona_id}",
            color='white', ha='center', va='center', fontsize=6.2, weight='bold',
            bbox=dict(boxstyle='round,pad=0.28', facecolor='black', edgecolor='white', linewidth=0.8, alpha=0.6)
        )

# final configuration
ax.set_xlim(x_min, x_max)
ax.set_ylim(z_min, z_max)
ax.set_xlabel('X (mm) - Comprimento (0вЂ“900)')
ax.set_ylabel('Z (mm) - Largura (0вЂ“420)')
ax.set_title('SubdivisГЈo 15Г—4: Temperatura MГ©dia e О”T por Subzona (60 zonas)')

sm = plt.cm.ScalarMappable(
    cmap='plasma',
    norm=plt.Normalize(vmin=estat_15x4['mean'].min(), vmax=estat_15x4['mean'].max())
)
cbar = plt.colorbar(sm, ax=ax)
cbar.set_label('Temperatura MГ©dia (В°C)')

plt.grid(False)
plt.tight_layout()

# (Opcional) rectangle outline between two zones (exemple)
zona_inf_esq = 12
zona_sup_dir = 25

def get_indices(zona_id, nx):
    i = (zona_id - 1) % nx
    j = (zona_id - 1) // nx
    return i, j

i_min, j_min = get_indices(zona_inf_esq, ncols)
i_max, j_max = get_indices(zona_sup_dir, ncols)

x_start = x_edges[i_min]
z_start = z_edges[j_min]
largura = x_edges[i_max + 1] - x_start
altura  = z_edges[j_max + 1] - z_start

rect_contorno = patches.Rectangle(
    (x_start, z_start), largura, altura,
    linewidth=2.5, edgecolor='blue', facecolor='none', linestyle='solid'
)
ax.add_patch(rect_contorno)

plt.grid(False)
plt.tight_layout()
plt.show()

# ==========================
# ADJUST POTENCY вЂ” 15 ZONES FOR TARGET TEMPERATURE
# ==========================
K = 1.5         # Thermic correction gain (adjust how aggressive it is)
P_MIN = 10      # Minimin allowed potency (W)
P_MAX = 150     # Maximun allowed potency (W)

# --- Resistences centers in mm (of drawing) ---
centros = [
    29.60, 88.80, 148.00, 207.20, 266.40,
    331.60, 390.80, 450.00, 509.20, 568.40,
    633.60, 692.80, 752.00, 811.20, 870.40
]
centros = sorted(centros)

# --- Irregular limits of 15 zones per average dot between centers ---
limites = [0.0]
for i in range(1, len(centros)):
    limites.append(0.5 * (centros[i-1] + centros[i]))
limites.append(900.0)
limites = np.array(limites)  # 16 borders в†’ 15 zones

# Function to map X -> zone [1..15] using irregular limits
def identificar_zona_x_custom(xp, edges):
    for i in range(len(edges) - 1):
        if edges[i] <= xp < edges[i + 1]:
            return i + 1
    if np.isclose(xp, edges[-1]):
        return len(edges) - 1
    return np.nan

# Atribute zone to each dot and calculate T_mГ©dia per zone
df['Zona_X'] = df['X (mm)'].apply(lambda v: identificar_zona_x_custom(v, limites))
medias_por_zona_x = df.groupby('Zona_X')['Value (Celsius)'].mean().reset_index()
zona_temp = medias_por_zona_x.copy().rename(columns={'Value (Celsius)': 'T_mГ©dia'})
zona_temp = zona_temp.sort_values('Zona_X').reset_index(drop=True)

# --- INPUT: Target  Temperature ---
while True:
    try:
        T_SET = float(input("\nInsert TARGET for SURFACE TEMPERATURE (В°C): "))
        break
    except ValueError:
        print("Invalid Value. Insert a number (В°C).")

# --- INPUT: Only potency of zones 1..8 (everything else is simetrical) ---
print("\nInsert applicable potency for each zone (1 a 8) вЂ” side around zona 8 is simetrical:")
potencias_parciais = {}
for i in range(1, 9):  # 1..8
    while True:
        try:
            p = float(input(f"  Zona {i}: "))
            potencias_parciais[i] = p
```

```python
            break
        except ValueError:
            print("Invalid Value. Insert a valid number.")

# Vector reconstruction complete 1..15 (center = zone 8)
potencias_aplicadas = [
    potencias_parciais[1],  # 1
    potencias_parciais[2],  # 2
    potencias_parciais[3],  # 3
    potencias_parciais[4],  # 4
    potencias_parciais[5],  # 5
    potencias_parciais[6],  # 6
    potencias_parciais[7],  # 7
    potencias_parciais[8],  # 8 (center)
    potencias_parciais[7],  # 9
    potencias_parciais[6],  # 10
    potencias_parciais[5],  # 11
    potencias_parciais[4],  # 12
    potencias_parciais[3],  # 13
    potencias_parciais[2],  # 14
    potencias_parciais[1],  # 15
]
zona_temp['PotГЄncia Aplicada (W)'] = potencias_aplicadas

# --- TARGET ADJUSTMENT ---
# Error per zone VS. target (positive = It is above target)
denom = max(T_SET, 1e-6)  # evitar divisГЈo por zero
zona_temp['Erro vs alvo (В°C)'] = zona_temp['T_mГ©dia'] - T_SET

# Adjusting proporcional facter (if T_mГ©dia > target в†’ reduce potency; if T_mГ©dia < target в†’ increase potency)
zona_temp['Fator ajuste'] = 1 - K * (zona_temp['Erro vs alvo (В°C)'] / denom)

# New potency (with limits)
zona_temp['Nova PotГЄncia (W)'] = (zona_temp['PotГЄncia Aplicada (W)'] * zona_temp['Fator ajuste']).clip(P_MIN, P_MAX)

# --- RESULTS ---
print("\n=== Ajuste por Zona para Alvo = {:.2f} В°C ===".format(T_SET))
for _, row in zona_temp.iterrows():
    print(f"Zona {int(row['Zona_X']):2d}: TmГ©dia = {row['T_mГ©dia']:.2f} В°C | "
          f"Erro = {row['Erro vs alvo (В°C)']:+.2f} В°C | "
          f"Fator = {row['Fator ajuste']:.3f} | "
          f"P_old = {row['PotГЄncia Aplicada (W)']:.1f} W в†’ P_new = {row['Nova PotГЄncia (W)']:.1f} W")

# --- GRAPHICS: Conmparison of Applied potency vs New potency + Error vs Target ---
fig, ax1 = plt.subplots(figsize=(12, 5.8))
ax1.bar(zona_temp['Zona_X'], zona_temp['Nova PotГЄncia (W)'], label='Nova PotГЄncia', alpha=0.85)
ax1.plot(zona_temp['Zona_X'], zona_temp['PotГЄncia Aplicada (W)'],
         linestyle='--', marker='o', label='PotГЄncia Aplicada')

ax1.set_xlabel('Zona Longitudinal (X) вЂ” 15 zonas alinhadas aos centros reais')
ax1.set_ylabel('PotГЄncia (W)')
ax1.set_title(f'Ajuste de PotГЄncia por Zona вЂ” Alvo = {T_SET:.1f} В°C (K = {K})')
ax1.legend()
ax1.grid(True, axis='y', linestyle='--', alpha=0.5)
ax1.set_xticks(range(1, 16))

# Secundary axis: error vs target
ax2 = ax1.twinx()
ax2.plot(zona_temp['Zona_X'], zona_temp['Erro vs alvo (В°C)'],
         marker='s', linestyle='-', label='Erro vs alvo (В°C)')
ax2.set_ylabel('Erro vs alvo (В°C)')
ax2.tick_params(axis='y')

plt.tight_layout()
plt.show()

# --- OVERLAY: draw 15 limits on 2D map (for visual conference) ---
fig = plt.figure(figsize=(10, 8))
contour = plt.contourf(xi, yi, zi, levels=100, cmap='plasma')
for e in limites:
    plt.axvline(e, linewidth=0.9, alpha=0.75)  # linhas dos limites de zona
# opcional: center markings
for c in centros:
    plt.axvline(c, linewidth=0.8, linestyle=':', alpha=0.6)
plt.xlabel('X (mm) - Comprimento (0вЂ“900)')
plt.ylabel('Z (mm) - Largura (0вЂ“420)')
plt.title('Mapa 2D com limites das 15 zonas (linhas) e centros (tracejado)')
cbar2 = plt.colorbar(contour)
cbar2.set_label('Temperatura (В°C)')
plt.tight_layout()
plt.show()
```