```
# -*- coding: utf-8 -*-
"""
Created on Sun Aug 31 04:16:05 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
# ==========================
file_path = r"C:/Users/ugims/inegi.up.pt/Teses & EstГЎgios - Teses_EstГЎgios - Miguel AntГіnio Costa - Teses_EstГЎgios - Miguel AntГіnio Costa/3. RepositГіrio do Migue

# Target dimension of mould
MOLD_X = 900.0  # mm
MOLD_Z = 420.0  # mm

# Adjust by zone (parameters are adjustable)
K = 1.5         # thermic correction factor (gain)
P_MIN = 10      # W (minimun allowed)
P_MAX = 150     # W (maximun allowed)

# real center (mm) of 15 resistances вЂ” 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)
assert len(CENTROS) == 15
assert 0 <= min(CENTROS) and max(CENTROS) <= 900

# ==========================
# READ + CLEAN
# ==========================
df = pd.read_csv(
    file_path,
    skiprows=8,
    sep=',',
    skip_blank_lines=True,
    engine='python',
    encoding='windows-1252'
)
df.columns = [c.strip() for c in df.columns]
df = df.applymap(lambda x: x.strip() if isinstance(x, str) else x)
for col in df.columns:
    df[col] = pd.to_numeric(df[col], errors='coerce')
df.dropna(inplace=True)

# ==========================
# RESCALING TO 900 x 420
# ==========================
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()
if x_max_raw == x_min_raw:
    raise ValueError("X interval in file is Null; Rescaling is impossible.")
if z_max_raw == z_min_raw:
    raise ValueError("Z interval in file is Null; Rescaling is impossible.")

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

x = df['X (mm)']
z = df['Z (mm)']
T = df['Value (Celsius)']

# ==========================
# BASE GRAPHICS (3D and 2D interpolated)
# ==========================
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
sc = ax.scatter(x, z, T, c=T, 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 вЂ” 3D')
plt.colorbar(sc, ax=ax, pad=0.1, label='Temperatura (В°C)')
plt.tight_layout()
plt.show()

xi = np.linspace(x.min(), x.max(), 220)
zi = np.linspace(z.min(), z.max(), 220)
XI, ZI = np.meshgrid(xi, zi)
TI = griddata((x, z), T, (XI, ZI), method='cubic')

plt.figure(figsize=(10, 8))
cf = plt.contourf(XI, ZI, TI, levels=100, cmap='plasma')
plt.xlabel('X (mm) - Comprimento (0вЂ“900)')
plt.ylabel('Z (mm) - Largura (0вЂ“420)')
plt.title('Mapa 2D de Temperatura вЂ” Interpolado')
plt.colorbar(cf, label='Temperatura (В°C)')
plt.tight_layout()
plt.show()

# ==========================
# 15 REAL ZONES (limited by MEDIUM DOTS between CENTERS)
# ==========================
limites = [0.0]
for i in range(1, len(CENTROS)):
    limites.append(0.5 * (CENTROS[i-1] + CENTROS[i]))
limites.append(MOLD_X)
x_edges = np.array(limites, dtype=float)  # 16 borders в†’ 15 zones

def zona15_por_x(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

df['Zona_15'] = df['X (mm)'].apply(lambda v: zona15_por_x(v, x_edges))


# ==========================
```

```python
# SUBDIVISION 15 Г— 4 (60 subzones) вЂ" aligned to 15 longitudinal real zones
# ===========================
# Borders in X are already in x_edges (16 borders в†' 15 columns)
z_edges = np.linspace(0.0, MOLD_Z, 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):
    # colunm (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  # incluir extremo superior

    # 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

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

# atributing 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)

# statistics by subzone
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 columns
    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}B°C\nO”T={delta_temp:.1f}B°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)
        )

ax.set_xlim(0.0, MOLD_X)
ax.set_ylim(0.0, MOLD_Z)
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 O”T por Subzona (60 zonas)')

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

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

# ===========================
# 3 MACRO ZONES (5+5+5)
# ===========================
def macro_zona(z15):
    if 1 <= z15 <= 5:
        return 1
    if 6 <= z15 <= 10:
        return 2
    if 11 <= z15 <= 15:
        return 3
    return np.nan

df['Zona_Macro'] = df['Zona_15'].apply(macro_zona)

# Average temperature of macro zones (based on data)
medias_macro = df.groupby('Zona_Macro')['Value (Celsius)'].mean().reindex([1, 2, 3])
print("\nAverage temperature per macro zone (confirmed data):")
for zmac in [1, 2, 3]:
    val = float(medias_macro.loc[zmac]) if zmac in medias_macro.index and pd.notna(medias_macro.loc[zmac]) else np.nan
    print(f"  Zona {zmac}: {val:.2f} B°C")

# ===========================
# ADJUST PER ZONE в†' POTENCY EMITTED IS EQUIVALENT PER ZONE
# Z1=1вЂ“5 (LEFT), Z2=6вЂ“10 (CENTER), Z3=11вЂ“15 (RIGHT)
# ===========================
def pedir_float(msg):
    while True:
        try:
            return float(input(msg))
        except ValueError:
            print("Invalid Value. Insert a number.")

# (Re)Calculate macro-zone in case it doesn't exist
if 'Zona_Macro' not in df.columns:
```

```python
    def macro_zona(z15):
        if 1 <= z15 <= 5:   return 1
        if 6 <= z15 <= 10:  return 2
        if 11 <= z15 <= 15: return 3
        return np.nan
    df['Zona_Macro'] = df['Zona_15'].apply(macro_zona)

# Average temperature per zone (SECURITY: write in missing data with average global)
medias_macro = df.groupby('Zona_Macro')['Value (Celsius)'].mean().reindex([1, 2, 3])
if medias_macro.isna().any():
    medias_macro = medias_macro.fillna(df['Value (Celsius)'].mean())

print("\nDefine a TARGET TEMPERATURE (B°C) for each zone:")
T_SET_1 = pedir_float("  Zona 1 (resistГЄncias  1вЂ“5):  ")
T_SET_2 = pedir_float("  Zona 2 (resistГЄncias  6вЂ“10): ")
T_SET_3 = pedir_float("  Zona 3 (resistГЄncias 11вЂ“15): ")

print("\nInsert an INICIAL POTENCY (W) for each zone:")
PZ1 = pedir_float("  Zona 1 (1вЂ“5):   ")
PZ2 = pedir_float("  Zona 2 (6вЂ“10):  ")
PZ3 = pedir_float("  Zona 3 (11вЂ“15): ")

# Simple equation per zone: P_new = P_old * (1 - K * (T_med - T_set)/T_set), with limits
K = K  # use K already defined on top of script
P_MIN = P_MIN
P_MAX = P_MAX

T1 = float(medias_macro.loc[1])
T2 = float(medias_macro.loc[2])
T3 = float(medias_macro.loc[3])

def ajusta(P_old, T_med, T_set):
    denom = max(T_set, 1e-6)
    fator = 1 - K * ((T_med - T_set) / denom)
    return float(np.clip(P_old * fator, P_MIN, P_MAX)), fator

PZ1_new, f1 = ajusta(PZ1, T1, T_SET_1)
PZ2_new, f2 = ajusta(PZ2, T2, T_SET_2)
PZ3_new, f3 = ajusta(PZ3, T3, T_SET_3)

print("\n=== EQUIVALENT Potency per zone (unique for each zone resistance) ===")
print(f" Z1 (1вЂ“5):   Tmed={T1:.2f}B°C в†’ Tset={T_SET_1:.2f}B°C | fator={f1:.3f} | P_old={PZ1:.1f} W в†’ P_new={PZ1_new:.1f} W")
print(f" Z2 (6вЂ“10):  Tmed={T2:.2f}B°C в†’ Tset={T_SET_2:.2f}B°C | fator={f2:.3f} | P_old={PZ2:.1f} W в†’ P_new={PZ2_new:.1f} W")
print(f" Z3 (11вЂ“15): Tmed={T3:.2f}B°C в†’ Tset={T_SET_3:.2f}B°C | fator={f3:.3f} | P_old={PZ3:.1f} W в†’ P_new={PZ3_new:.1f} W")

# Final vector to apply per resistance (uniformity within each zone)
pot_ajustada_15 = [PZ1_new]*5 + [PZ2_new]*5 + [PZ3_new]*5
pot_inicial_15  = [PZ1]*5    + [PZ2]*5    + [PZ3]*5
targets_vec     = [T_SET_1]*5 + [T_SET_2]*5 + [T_SET_3]*5

# ==========================
# GRAPHICS: PER ZONE (BEFORE vs AFTER) + errors vs target
# ==========================
fig, ax1 = plt.subplots(figsize=(9.6, 5.2))
zonas = np.array([1, 2, 3])
pot_ini = np.array([PZ1, PZ2, PZ3])
pot_new = np.array([PZ1_new, PZ2_new, PZ3_new])
erros_z = np.array([T1 - T_SET_1, T2 - T_SET_2, T3 - T_SET_3])

larg = 0.35
ax1.bar(zonas - larg/2, pot_ini, width=larg, label='PotГЄncia Inicial (W)', alpha=0.9)
ax1.bar(zonas + larg/2, pot_new, width=larg, label='PotГЄncia Nova (W)', alpha=0.9)
ax1.set_xticks(zonas)
ax1.set_xticklabels(['Z1 (1вЂ“5)', 'Z2 (6вЂ“10)', 'Z3 (11вЂ“15)'])
ax1.set_xlabel('Zona')
ax1.set_ylabel('PotГЄncia (W)')
ax1.set_title(f'Ajuste de PotГЄncia por Zona (K={K})')
ax1.legend()
ax1.grid(True, axis='y', linestyle='--', alpha=0.5)

ax2 = ax1.twinx()
ax2.plot(zonas, erros_z, linestyle='-', marker='s', label='Erro vs alvo (B°C)')
ax2.set_ylabel('Erro vs alvo (B°C)')
ax2.tick_params(axis='y')

plt.tight_layout()
plt.show()

# ==========================
# OVERLAY 2D: show 3 bands with new potency per zone
# ==========================
plt.figure(figsize=(10, 8))
cont = plt.contourf(XI, ZI, TI, levels=100, cmap='plasma')

# Vertical lines of limits from 15 zones
for e in x_edges:
    plt.axvline(e, linewidth=0.8, alpha=0.5, linestyle='--')

# Shade 3 bands (edges [0..5], [5..10], [10..15])
plt.axvspan(x_edges[0],  x_edges[5],  alpha=0.10, label=f"Z1: {PZ1_new:.0f} W (alvo {T_SET_1:.0f}B°C)")
plt.axvspan(x_edges[5],  x_edges[10], alpha=0.10, label=f"Z2: {PZ2_new:.0f} W (alvo {T_SET_2:.0f}B°C)")
plt.axvspan(x_edges[10], x_edges[15], alpha=0.10, label=f"Z3: {PZ3_new:.0f} W (alvo {T_SET_3:.0f}B°C)")

# Tags on top
plt.text(0.5*(x_edges[0]+x_edges[5]),   0.96*MOLD_Z, f"Z1: {PZ1_new:.0f} W",  ha='center', va='top', fontsize=10, weight='bold')
plt.text(0.5*(x_edges[5]+x_edges[10]),  0.96*MOLD_Z, f"Z2: {PZ2_new:.0f} W", ha='center', va='top', fontsize=10, weight='bold')
plt.text(0.5*(x_edges[10]+x_edges[15]), 0.96*MOLD_Z, f"Z3: {PZ3_new:.0f} W", ha='center', va='top', fontsize=10, weight='bold')

plt.xlabel('X (mm) - Comprimento (0вЂ“900)')
plt.ylabel('Z (mm) - Largura (0вЂ“420)')
plt.title('Mapa 2D | PotГЄncia equivalente por zona (apГіs ajuste)')
plt.colorbar(cont, label='Temperatura (B°C)')
plt.legend(loc='lower center', ncol=3, frameon=False)
plt.tight_layout()
plt.show()

# ==========================
# RESISTENCE GRAPHICS: 15 BANDS (uniform potency per zone)
# ==========================
fig, ax = plt.subplots(figsize=(12, 5.8))
idx = np.arange(1, 16, dtype=int)
macro_labels = np.array(['Z1']*5 + ['Z2']*5 + ['Z3']*5)

bars = ax.bar(idx, pot_ajustada_15, alpha=0.95, label='PotГЄncia Nova (W)')
ax.set_xlabel('ResistГЄncia | Zona | PosiГ§ГJo X (mm)')
ax.set_ylabel('PotГЄncia (W)')
ax.set_title('PotГЄncia aplicada por resistГЄncia (uniforme dentro da zona)')

ax.set_xticks(idx)
ax.set_xticklabels([f"R{i} | {macro_labels[i-1]}\n{CENTROS[i-1]:.1f} mm" for i in idx], rotation=0)

for k in range(0, 16):
    ax.axvline(k + 0.5, linestyle='--', alpha=0.35, linewidth=0.7)
```

```python
for b in bars:
    b.set_edgecolor('black'); b.set_linewidth(0.6)

y_max = max(pot_ajustada_15) if len(pot_ajustada_15) else 1
ax.set_ylim(0, y_max * 1.15)
ax.grid(True, axis='y', linestyle='--', alpha=0.5)
ax.legend()
plt.tight_layout()
plt.show()
```