```python
# -*- coding: utf-8 -*-
"""
Created on Sat Aug 30 19:28:30 2025

@author: ugims
"""

# -*- coding: utf-8 -*-
"""
Mould 900 x 420 mm вЂў Control in 3 zones (5+5+5 resistances)
- Reescaling coordenates of CSV to 900x420
- Map dots to all 15 longitudinal real zones (via centre)
- Request potency to zone 1, zone 2, zone 3 (to each appl 5 resistances)
- Automatic adjusting per zone (K * relative error) with limits [P_MIN, P_MAX]
- Export CSVs: inicial plan and ajusted plan
- Generate graphics: 3D, 2D, 3 zone bars and 2D overlay of bands
"""

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 (fixed values) вЂ“ no input()
# ==========================
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

MOLD_X = 900.0  # mm (height)
MOLD_Z = 420.0  # mm (width)

# proportional controler (gain) and physical limit of potencies
K = 1.5
P_MIN = 10
P_MAX = 150

# --- fixed border (mm) вЂ” Edit here ---
BX_esq = 100
BX_dir = 100
BZ_inf = 80
BZ_sup = 80

# --- fixed target temperature (В°C) by macrozone вЂ” Edit here ---
# Z1: resistГЄncias 1вЂ“5 | Z2: 6вЂ“10 | Z3: 11вЂ“15
T_SET_1 = 140
T_SET_2 = 140
T_SET_3 = 140

# real centers (mm) dofas 15 resistences
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) <= MOLD_X

# ==========================
# READ + CLEAN + RESCALE
# ==========================
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)

# Rescale to 900x420
xr, zr = df['X (mm)'], df['Z (mm)']
x_min_raw, x_max_raw = xr.min(), xr.max()
z_min_raw, z_max_raw = zr.min(), zr.max()
if x_max_raw == x_min_raw:
    raise ValueError("X interval is Null; Rescaling impossible.")
if z_max_raw == z_min_raw:
    raise ValueError("Z interval is Null; Rescaling impossible.")

df['X (mm)'] = (xr - x_min_raw) / (x_max_raw - x_min_raw) * MOLD_X
df['Z (mm)'] = (zr - z_min_raw) / (z_max_raw - z_min_raw) * MOLD_Z

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

# ==========================
# 15 REAL ZONES (Limited by MEDIUM DOTS between CENTER)
# ==========================
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))

# 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)

# ==========================
# FIXED BORDER (ignore edge without any piece)
# ==========================
# Validate simple borders
if BX_esq + BX_dir >= MOLD_X - 1e-6:
    raise ValueError("Sum of X borders exceeds mould's height.")
if BZ_inf + BZ_sup >= MOLD_Z - 1e-6:
```

```
    raise ValueError("Sum of Z borders exceeds mould's width.")

x0, x1 = BX_esq, MOLD_X - BX_dir
z0, z1 = BZ_inf, MOLD_Z - BZ_sup

mask_util = (df['X (mm)'] >= x0) & (df['X (mm)'] <= x1) & \
            (df['Z (mm)'] >= z0) & (df['Z (mm)'] <= z1)
df_util = df.loc[mask_util].copy()

# Average for macro zones WITH border applied
medias_macro = df_util.groupby('Zona_Macro')['Value (Celsius)'] \
                      .mean().reindex([1, 2, 3])
print("\Medium temperature for macro zone (with border applied):")
for zmac in [1, 2, 3]:
    val = float(medias_macro.loc[zmac]) if pd.notna(medias_macro.loc[zmac]) else np.nan
    print(f"  Zona {zmac}: {val:.2f} B°C")

# Function to draw useful area
def desenhar_area_util(ax):
    rect = patches.Rectangle(
        (x0, z0), x1 - x0, z1 - z0,
        fill=False, linestyle='--', linewidth=2, edgecolor='white',
        label='ГЃrea Гєtil (sem bordo)'
    )
    ax.add_patch(rect)

# ===========================
# MAPS INTERPOLATION
# ===========================
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')

# ===========================
# GRAPHICS BASE (3D e 2D), with usefull area
# ===========================
fig = plt.figure(figsize=(10, 8))
ax3d = fig.add_subplot(111, projection='3d')
sc = ax3d.scatter(x, z, T, c=T, cmap='plasma')
ax3d.set_xlabel('X (mm)'); ax3d.set_ylabel('Z (mm)'); ax3d.set_zlabel('T (B°C)')
ax3d.set_title('DistribuiГ§ГЈo de Temperatura вЂ” 3D')
plt.colorbar(sc, ax=ax3d, pad=0.1, label='Temperatura (B°C)')
plt.tight_layout(); plt.show()

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')
ax2d = plt.gca()
desenhar_area_util(ax2d)
plt.colorbar(cf, label='Temperatura (B°C)')
plt.legend(loc='lower right', frameon=False)
plt.tight_layout(); plt.show()

# ===========================
# SUBDIVISION 15 Г— 4 (diagnosis) вЂ” shows complete mould + usefull area
# ===========================
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):
    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
    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

df['Zona_15x4'] = df.apply(lambda r: identificar_zona_15x4(r['X (mm)'], r['Z (mm)'], x_edges, z_edges), axis=1)
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')

fig, ax = plt.subplots(figsize=(16, 9))
for i in range(ncols):
    for j in range(nrows):
        zona_id = j * ncols + i + 1
        xs, zs = x_edges[i], 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((xs, zs), largura, altura, facecolor=cor, edgecolor='black', linewidth=0.5)
        ax.add_patch(rect)
        ax.text(xs + largura/2, zs + altura/2,
                f"{media_temp:.1f}B°C\nО”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, MOLD_X); ax.set_ylim(0, MOLD_Z)
ax.set_xlabel('X (mm)'); ax.set_ylabel('Z (mm)')
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()))
plt.colorbar(sm, ax=ax).set_label('Temperatura MГ©dia (B°C)')
# usefull area
ax.add_patch(patches.Rectangle((x0, z0), x1-x0, z1-z0, fill=False, linestyle='--', linewidth=2, edgecolor='white',
                               label='ГЃrea Гєtil (sem bordo)'))
plt.legend(loc='lower right', frameon=False)
plt.grid(False); plt.tight_layout(); plt.show()

# ===========================
# INPUT: INICIAL POTENCY for macrozone (kept)
# ===========================
def pedir_float(txt):
    while True:
```

```python
        try:
            return float(input(txt))
        except ValueError:
            print("Invalid value. Insert new number.")

print("\nInsert potency (W) for each macro zone (applied to 5 resistences from said zone):")
PZ1 = pedir_float("  Zone 1: ")
PZ2 = pedir_float("  Zone 2: ")
PZ3 = pedir_float("  Zone 3: ")

# ==========================
# ADJUST (proportions are multoplicative) WITH usefull area averages
# ==========================
T_targets = {1: T_SET_1, 2: T_SET_2, 3: T_SET_3}
ajuste = []
for zmac, P_old in zip([1, 2, 3], [PZ1, PZ2, PZ3]):
    Tz   = float(medias_macro.loc[zmac]) if pd.notna(medias_macro.loc[zmac]) else np.nan
    Tset = T_targets[zmac]
    denom = max(Tset, 1e-6)
    erro  = (Tz - Tset) if not np.isnan(Tz) else 0.0
    fator = 1 - K * (erro / denom) if not np.isnan(Tz) else 1.0
    P_new = float(np.clip(P_old * fator, P_MIN, P_MAX))
    ajuste.append({
        'Zona': zmac,
        'T_mГ©dia (В°C)': Tz,
        'T_alvo (В°C)':  Tset,
        'Erro (В°C)':    erro,
        'Fator ajuste': fator,
        'PotГЄncia Inicial (W)': P_old,
        'Nova PotГЄncia (W)':    P_new
    })

ajuste_df = pd.DataFrame(ajuste)
print("\n=== Adjust by macro zone (with applied border) ===")
print(ajuste_df.to_string(index=False, formatters={
    'T_mГ©dia (В°C)': lambda v: f"{v:.2f}",
    'T_alvo (В°C)':  lambda v: f"{v:.2f}",
    'Erro (В°C)':    lambda v: f"{v:+.2f}",
    'Fator ajuste': lambda v: f"{v:.3f}",
    'PotГЄncia Inicial (W)': lambda v: f"{v:.1f}",
    'Nova PotГЄncia (W)':    lambda v: f"{v:.1f}"
}))

# adjusted vectors by resistence (repeat the new potency of macro zone)
PZ1_new, PZ2_new, PZ3_new = ajuste_df.set_index('Zona').loc[[1, 2, 3], 'Nova PotГЄncia (W)'].tolist()
pot_ajustada_15 = [PZ1_new]*5 + [PZ2_new]*5 + [PZ3_new]*5

# ==========================
# GRAPHIC: POTENCY BY ZONE (BEFORE and AFTER)
# ==========================
fig, ax1 = plt.subplots(figsize=(9.5, 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])

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='Nova PotГЄncia (W)', alpha=0.9)
ax1.set_xticks(zonas)
ax1.set_xlabel('Zona Macro')
ax1.set_ylabel('PotГЄncia (W)')
ax1.set_title('Ajuste de PotГЄncia por Zona (3 zonas)')
ax1.legend()
ax1.grid(True, axis='y', linestyle='--', alpha=0.5)

ax2 = ax1.twinx()
ax2.plot(zonas, ajuste_df['Erro (В°C)'], linestyle='-', marker='s', label='Erro tГ©rmico (В°C)')
ax2.set_ylabel('Erro tГ©rmico (В°C)')
ax2.tick_params(axis='y')
plt.tight_layout(); plt.show()

# ==========================
# OVERLAY 2D: limits of 15 zones + bands of 3 macrozones + usefull area
# ==========================
plt.figure(figsize=(10, 8))
cf = plt.contourf(XI, ZI, TI, levels=100, cmap='plasma')
for e in x_edges:
    plt.axvline(e, linewidth=0.9, alpha=0.55, linestyle='--')

# Macro-zones: [0..5], [5..10], [10..15] in x_edges
plt.axvspan(x_edges[0],  x_edges[5],  alpha=0.10, label=f"Z1: {PZ1_new:.0f} W")
plt.axvspan(x_edges[5],  x_edges[10], alpha=0.10, label=f"Z2: {PZ2_new:.0f} W")
plt.axvspan(x_edges[10], x_edges[15], alpha=0.10, label=f"Z3: {PZ3_new:.0f} W")

# Useful Area
ax = plt.gca()
desenhar_area_util(ax)

# Tags no topo
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)'); plt.ylabel('Z (mm)')
plt.title('Mapa 2D com Zonas Macro e PotГЄncia Ajustada')
plt.colorbar(cf, label='Temperatura (В°C)')
plt.legend(loc='lower center', ncol=3, frameon=False)
plt.tight_layout(); plt.show()

# ==========================
# GRAPHICS BY RESISTANCE: 15 BANDS (with real position on mm)
# ==========================
fig, ax = plt.subplots(figsize=(12, 5.8))
idx = np.arange(1, 16, dtype=int)

# background by macrozone
ax.axvspan(0.5, 5.5,  alpha=0.08)
ax.axvspan(5.5, 10.5, alpha=0.08)
ax.axvspan(10.5, 15.5, alpha=0.08)

bars = ax.bar(idx, pot_ajustada_15, alpha=0.95, label='PotГЄncia Ajustada (W)')
ax.set_xlabel('ResistГЄncia (R)  |  PosiГ§ГЈo ao longo de X (mm)')
ax.set_ylabel('PotГЄncia (W)')
ax.set_title('PotГЄncia por ResistГЄncia (apГіs ajuste) вЂ” com posiГ§ГЈo real')

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

# thin lines between bands (equivalent as real mid-distances)
for k in range(0, 16):  # 0.5..15.5
    ax.axvline(k + 0.5, linestyle='--', alpha=0.35, linewidth=0.7)

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()
```