```python
# -*- coding: utf-8 -*-
"""
Created on Sat Aug 30 21:20:35 2025

@author: ugims
"""

# -*- coding: utf-8 -*-
"""
Mould 900 Г— 420 mm вЂ“ Zonal analysis + per-heater adjustment
- Rescales CSV coordinates to 900Г—420
- Applies a fixed border to focus on the useful area (like previous script)
- Uses fixed target temperatures per macro-zone (Z1/Z2/Z3) defined in code
- Plots: 3D scatter, 2D interpolated map, 3Г—2 coloured means, 15Г—4 subzones with O”T,
         15-bar chart (before vs after) + error line, and 2D overlay with zone limits & centres
"""

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; edit here)
# ==========================
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

# Mould size
MOLD_X = 900.0  # mm (length)
MOLD_Z = 420.0  # mm (width)

# Proportional controller and power limits
K = 1.5
P_MIN = 10
P_MAX = 150

# Fixed border (mm) вЂ” applied to all analyses/statistics
BX_left   = 100.0
BX_right  = 100.0
BZ_bottom = 80.0
BZ_top    = 80.0

# Fixed target temperatures (В°C) per macro-zone
# Z1 = LEFT (heaters 1вЂ“5) | Z2 = CENTRE (6вЂ“10) | Z3 = RIGHT (11вЂ“15)
T_SET_1 = 160.0
T_SET_2 = 140.0
T_SET_3 = 120.0

# Real centres (mm) of the 15 heaters (left в†’ right)
CENTRES = [
    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
]
CENTRES = sorted(CENTRES)
assert len(CENTRES) == 15 and 0 <= min(CENTRES) and max(CENTRES) <= MOLD_X

# ==========================
# READ + CLEAN + RESCALE
# ==========================
df = pd.read_csv(
    file_path,
    skiprows=8,
    sep=',',
    skip_blank_lines=True,
    engine='python',
    encoding='windows-1252'
)

# Clean headers and whitespace in string cells (no applymap warning)
df.columns = [col.strip() for col in df.columns]
df = df.apply(lambda col: col.map(lambda x: x.strip() if isinstance(x, str) else x))

# Convert to numeric where possible and drop invalid rows
for col in df.columns:
    df[col] = pd.to_numeric(df[col], errors='coerce')
df.dropna(inplace=True)

# Rescale to [0..MOLD_X], [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()
if x_max_raw == x_min_raw:
    raise ValueError("X range in the file is null; rescaling is not possible.")
if z_max_raw == z_min_raw:
    raise ValueError("Z range in the file is null; rescaling is not possible.")
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)']
T = df['Value (Celsius)']

# Useful area (apply border)
if (BX_left + BX_right) >= MOLD_X - 1e-6:
    raise ValueError("Sum of X borders exceeds mould length.")
if (BZ_bottom + BZ_top) >= MOLD_Z - 1e-6:
    raise ValueError("Sum of Z borders exceeds mould width.")
x0, x1 = BX_left,  MOLD_X - BX_right
z0, z1 = BZ_bottom, MOLD_Z - BZ_top
mask_useful = (df['X (mm)'] >= x0) & (df['X (mm)'] <= x1) & (df['Z (mm)'] >= z0) & (df['Z (mm)'] <= z1)
df_useful = df.loc[mask_useful].copy()

def draw_useful_area(ax):
    ax.add_patch(
        patches.Rectangle((x0, z0), x1 - x0, z1 - z0,
                          fill=False, linestyle='--', linewidth=2, edgecolor='white',
                          label='Useful area (without border)')
    )

# ==========================
# 3D SCATTER (full map, with useful area overlay)
# ==========================
fig = plt.figure(figsize=(10, 8))
ax3d = fig.add_subplot(111, projection='3d')
sc = ax3d.scatter(x, y, T, c=T, cmap='plasma')
ax3d.set_xlabel('X (mm) - Length (0вЂ“900)')
ax3d.set_ylabel('Z (mm) - Width (0вЂ“420)')
ax3d.set_zlabel('Temperature (В°C)')
ax3d.set_title('Temperature Distribution on Top Surface (900 Г— 420 mm)')
```

```
cbar = plt.colorbar(sc, ax=ax3d, pad=0.1); cbar.set_label('Temperature (B°C)')
plt.tight_layout(); plt.show()

# ==========================
# 2D INTERPOLATED MAP (full map, with useful area overlay)
# ==========================
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), T, (XI, YI), method='cubic')

plt.figure(figsize=(10, 8))
cf = plt.contourf(XI, YI, ZI, levels=100, cmap='plasma')
plt.xlabel('X (mm) - Length (0вЂ“900)')
plt.ylabel('Z (mm) - Width (0вЂ“420)')
plt.title('2D Temperature Map вЂ” Interpolated')
ax2d = plt.gca()
draw_useful_area(ax2d)
plt.colorbar(cf, label='Temperature (B°C)')
plt.legend(loc='lower right', frameon=False)
plt.tight_layout(); plt.show()

# ==========================
# 3 Г— 2 ZONES (means over the useful area)
# ==========================
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
z_div = np.linspace(z_min, z_max, 3)  # 2 zones in Z

def zone_3x2(xp, zp):
    for i in range(3):
        for j in range(2):
            if x_div[i] <= xp < x_div[i+1] and z_div[j] <= zp < z_div[j+1]:
                return j * 3 + i + 1
    if np.isclose(xp, x_max) and np.isclose(zp, z_max):
        return 6
    return np.nan

df_useful['Zone_3x2'] = df_useful.apply(lambda r: zone_3x2(r['X (mm)'], r['Z (mm)']), axis=1)
means_3x2 = df_useful.groupby('Zone_3x2')['Value (Celsius)'].mean().reset_index()
print("\nMean temperature per 3Г—2 zone (B°C) вЂ” useful area:")
for _, row in means_3x2.iterrows():
    print(f"Zone {int(row['Zone_3x2'])}: {row['Value (Celsius)']:.2f} B°C")

zone_to_mean = dict(zip(means_3x2['Zone_3x2'], means_3x2['Value (Celsius)']))

fig, ax = plt.subplots(figsize=(10, 8))
vmin, vmax = means_3x2['Value (Celsius)'].min(), means_3x2['Value (Celsius)'].max()
for i in range(3):
    for j in range(2):
        zone_id = j * 3 + i + 1
        xs, zs = x_div[i], z_div[j]
        w, h = x_div[i+1] - x_div[i], z_div[j+1] - z_div[j]
        m = zone_to_mean.get(zone_id, np.nan)
        color = plt.cm.plasma((m - vmin) / (vmax - vmin + 1e-12)) if not np.isnan(m) else 'gray'
        ax.add_patch(patches.Rectangle((xs, zs), w, h, facecolor=color, edgecolor='white', linewidth=1))
        ax.text(xs + w/2, zs + h/2, f"{m:.1f}B°C\nZone {zone_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) - Length (0вЂ“900)'); ax.set_ylabel('Z (mm) - Width (0вЂ“420)')
ax.set_title('3Г—2 Heating Zones with Mean Temperature (Useful Area)')
sm = plt.cm.ScalarMappable(cmap='plasma', norm=plt.Normalize(vmin=vmin, vmax=vmax))
plt.colorbar(sm, ax=ax).set_label('Mean Temperature (B°C)')
draw_useful_area(ax)
plt.grid(False); plt.tight_layout(); plt.show()

# ==========================
# 15 Г— 4 SUBZONES (values in moul'd's TODO) вЂ” overlay mostra a ГЎrea Гєtil
# ==========================
# X edges per average dots between real centers
x_edges = [0.0]
for i in range(1, len(CENTRES)):
    x_edges.append(0.5 * (CENTRES[i-1] + CENTRES[i]))
x_edges.append(MOLD_X)
x_edges = np.array(x_edges, dtype=float)  # 16 borders вЂ“ 15 columns
z_edges = np.linspace(0.0, MOLD_Z, 5)     # 5 borders вЂ“ 4 lines

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

def zone_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

# >>> Statistics 15Г—4 with ALL the dots (df), for all values outside the boundaries
df['Zone_15x4_ALL'] = df.apply(lambda r: zone_15x4(r['X (mm)'], r['Z (mm)'], x_edges, z_edges), axis=1)
stats_15x4_all = df.groupby('Zone_15x4_ALL')['Value (Celsius)'].agg(['min', 'max', 'mean']).reset_index()
stats_15x4_all['Delta (max - min)'] = stats_15x4_all['max'] - stats_15x4_all['min']
stats_map_all = stats_15x4_all.set_index('Zone_15x4_ALL').to_dict(orient='index')

# Graphic 15Г—4 with values for entire mould; usefull area is drawn in dashes
fig, ax = plt.subplots(figsize=(16, 9))
vmin2 = stats_15x4_all['mean'].min()
vmax2 = stats_15x4_all['mean'].max()
den = max(vmax2 - vmin2, 1e-12)

for i in range(ncols):
    for j in range(nrows):
        zone_id = j * ncols + i + 1
        xa, xb = x_edges[i], x_edges[i + 1]
        za, zb = z_edges[j], z_edges[j + 1]
        w, h = xb - xa, zb - za

        st = stats_map_all.get(zone_id)
        if st and np.isfinite(st['mean']):
            m, d = st['mean'], st['Delta (max - min)']
            color = plt.cm.plasma((m - vmin2) / den)
            ax.add_patch(patches.Rectangle((xa, za), w, h, facecolor=color, edgecolor='black', linewidth=0.5))
            ax.text(xa + w/2, za + h/2, f"{m:.1f}B°C\nO”T={d:.1f}B°C\nZone {zone_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))
```

```
        else:
            # In case a subzone does not exist, Show grey with no label
            ax.add_patch(patches.Rectangle((xa, za), w, h, facecolor='0.7', edgecolor='black', linewidth=0.5))

ax.set_xlim(0.0, MOLD_X); ax.set_ylim(0.0, MOLD_Z)
ax.set_xlabel('X (mm)'); ax.set_ylabel('Z (mm)')
ax.set_title('Subdivision 15Γ—4: Mean Temperature and O”T per Subzone (Full Map)')

sm = plt.cm.ScalarMappable(cmap='plasma', norm=plt.Normalize(vmin=vmin2, vmax=vmax2))
cb = plt.colorbar(sm, ax=ax); cb.set_label('Mean Temperature (B°C)')

# Overlay of usefull area (pallied boundary) вЂ” only for visual reference
draw_useful_area(ax)

plt.legend(loc='lower right', frameon=False)
plt.grid(False); plt.tight_layout(); plt.show()

# ============================
# PER-HEATER ADJUSTMENT (targets per macro-zone; means over useful area)
# ============================
# Map each useful-area point to one of the 15 heaters by X zone
def zone_15_by_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_useful['Zone_15'] = df_useful['X (mm)'].apply(lambda v: zone_15_by_x(v, x_edges))

# Average temperature per heater (useful area). Fill any missing with overall mean as a fallback.
T_mean_by_heater = df_useful.groupby('Zone_15')['Value (Celsius)'].mean().reindex(range(1, 16))
if T_mean_by_heater.isna().any():
    T_mean_by_heater = T_mean_by_heater.fillna(df_useful['Value (Celsius)'].mean())

# ===== FIXED TARGETS per heater via macro-zone labels
targets_vec = np.empty(15, dtype=float)
macro_labels = []
for i in range(1, 16):
    if 1 <= i <= 5:
        targets_vec[i-1] = T_SET_1; macro_labels.append('Z1')  # Left
    elif 6 <= i <= 10:
        targets_vec[i-1] = T_SET_2; macro_labels.append('Z2')  # Centre
    else:
        targets_vec[i-1] = T_SET_3; macro_labels.append('Z3')  # Right

# ===== INPUTS: INITIAL POWER per heater (kept as user input; can be fixed if you prefer)
def ask_float(msg):
    while True:
        try:
            return float(input(msg))
        except ValueError:
            print("Invalid value. Please enter a number.")

print("\nEnter the INITIAL POWER (W) for each heater (1 to 15):")
P_initial = np.zeros(15, dtype=float)
for i in range(1, 16):
    while True:
        try:
            P_initial[i-1] = float(input(f"  Heater {i:2d} (X = {CENTRES[i-1]:.1f} mm): "))
            break
        except ValueError:
            print("  Invalid value. Try again.")

# ===== ADJUSTMENT
T_mean_vec = T_mean_by_heater.values.astype(float)
denom = np.maximum(targets_vec, 1e-6)
errors = T_mean_vec - targets_vec
factors = 1 - K * (errors / denom)
P_new = np.clip(P_initial * factors, P_MIN, P_MAX)

# ===== CONSOLE REPORT
print("\n=== Individual Adjustment per Heater (targets by macro-zone; using useful area) ===")
for i in range(1, 16):
    print(f"R{i:02d} [{macro_labels[i-1]} @ {CENTRES[i-1]:.1f} mm]: "
          f"Tmean={T_mean_vec[i-1]:.2f}B°C, Tset={targets_vec[i-1]:.2f}B°C, "
          f"error={errors[i-1]:+.2f}B°C, factor={factors[i-1]:.3f} в†’ "
          f"P_old={P_initial[i-1]:.1f} W в†’ P_new={P_new[i-1]:.1f} W")

# ===== BAR CHART: 15 heaters (before vs after) + error line
fig, ax1 = plt.subplots(figsize=(13, 6.0))
idx = np.arange(1, 16, dtype=int)
bw = 0.4
ax1.bar(idx - bw/2, P_initial, width=bw, label='Initial Power (W)', alpha=0.9)
ax1.bar(idx + bw/2, P_new,     width=bw, label='New Power (W)',     alpha=0.9)
ax1.set_xlabel('Heater | Macro-Zone | X Position (mm)')
ax1.set_ylabel('Power (W)')
ax1.set_title('Individual Heater Adjustment вЂ” Targets by Macro-Zone (Useful Area)')
ax1.set_xticks(idx)
ax1.set_xticklabels([f"R{i} | {macro_labels[i-1]}\n{CENTRES[i-1]:.1f} mm" for i in idx], rotation=0)
ax1.legend()
ax1.grid(True, axis='y', linestyle='--', alpha=0.5)

ax2 = ax1.twinx()
ax2.plot(idx, errors, linestyle='-', marker='s', label='Error vs target (B°C)')
ax2.set_ylabel('Error vs target (B°C)')
ax2.tick_params(axis='y')

# Macro-zone shading
for k in range(0, 16):
    ax1.axvline(k + 0.5, linestyle='--', alpha=0.25, linewidth=0.7)
ax1.axvspan(0.5,  5.5,  alpha=0.08, label='Z1 (Left)')
ax1.axvspan(5.5, 10.5,  alpha=0.08, label='Z2 (Centre)')
ax1.axvspan(10.5,15.5,  alpha=0.08, label='Z3 (Right)')

plt.tight_layout(); plt.show()

# ===== 2D OVERLAY: 15 zone limits + centres + useful area
plt.figure(figsize=(10, 8))
contour = plt.contourf(XI, YI, ZI, levels=100, cmap='plasma')
for e in x_edges:
    plt.axvline(e, linewidth=0.9, alpha=0.75)  # zone boundaries
for c in CENTRES:
    plt.axvline(c, linewidth=0.8, linestyle=':', alpha=0.6)  # centres
plt.xlabel('X (mm) - Length (0вЂ“900)')
plt.ylabel('Z (mm) - Width (0вЂ“420)')
plt.title('2D Map with 15 Zone Limits (solid) and Centres (dotted)')
cbar2 = plt.colorbar(contour); cbar2.set_label('Temperature (B°C)')
draw_useful_area(plt.gca())
plt.tight_layout(); plt.show()
```