Estimation of complete dyke dimensions by Ellipse DE method

Biswas et al.

```
import pandas as pd
import numpy as np
from pathlib import Path


def smooth_elements(data):
    vec = np.zeros(len(data['x']))

    for i in range(len(data['x'])):
        if i < 2 or i > len(data['x']) - 3:
            vec[i] = data['y'][i]
        else:
            if (data['y'][i] >= max(data['y'][i - 2:i + 3])) or
(data['y'][i] <= min(data['y'][i - 2:i + 3])):
                weights = [data['x'][i + 1] - data['x'][i], data['x'][i] -
data['x'][i - 1]]
                values = [data['y'][i - 1], data['y'][i + 1]]
                vec[i] = np.average(values, weights=weights)
            else:
                vec[i] = data['y'][i]

    return pd.DataFrame({'x': data['x'], 'y': vec})


def estimate_parameters(data):
    dy = np.zeros(len(data['x']) - 2)
    d2y = np.zeros(len(data['x']) - 4)
    newval = np.zeros(len(data['x']) - 4)
    x_est = np.zeros(len(data['x']) - 2)

    for i in range(len(data['x']) - 2):
        dy[i] = (data['y'][i + 2] - data['y'][i]) / (data['x'][i + 2] -
data['x'][i])

    for i in range(len(dy) - 2):
        d2y[i] = (dy[i + 2] - dy[i]) / (data['x'][i + 3] - data['x'][i +
1])
        newval[i] = data['y'][i + 2] * d2y[i] + dy[i + 1] ** 2

    for i in range(len(data['x']) - 2):
        x_est[i] = data['y'][i + 1] * dy[i] / np.mean(newval)

    b_a_estimate = round(np.sqrt(-np.mean(newval)), 6)
    Gd = round(np.sum(d2y < 0) / (len(data['x']) - 4), 6)
    c = round(np.median(data['x'][1:-1]) - np.median(x_est), 6)
    x_hat = data['x'] - c
    b = round(np.sqrt(np.median(-np.mean(newval) * (x_hat ** 2) +
(data['y']) ** 2)), 6)
    a = round(b / np.sqrt(-np.mean(newval)), 6)

    return {
        'B/A': b_a_estimate,
        'Gd': Gd,
```

```
        'A': 2*a,
        'B': 2*b
    }


def p_params(data, cycle, file_name):
    params = {'Dyke No': file_name}
    params.update(estimate_parameters(data))
    params['Cycle'] = cycle

    print(params)


def s_params(data, cycle, file_name):

    params = {'Dyke No.': file_name}
    params.update(estimate_parameters(data))
    params['Cycle'] = cycle
    params_df = pd.DataFrame(params, index=[0])

    return params_df


input_dir = Path.home() / 'Desktop' / 'D1.xlsx'
output_dir = Path.home() / 'Desktop' / 'D1 OUT.xlsx'

df = pd.read_excel(input_dir)
df.rename(columns={df.columns[0]: 'x', df.columns[1]: 'y'}, inplace=True)


prams = estimate_parameters
df1 = smooth_elements(df)
df2 = smooth_elements(smooth_elements(df))
df3 = smooth_elements(smooth_elements(smooth_elements(df)))

if not estimate_parameters(df)["Gd"] < 0.5 and not
np.iscomplex(estimate_parameters(df)["B/A"]):
    p_params(df, cycle=0, file_name=input_dir.stem)
else:
    if not estimate_parameters(df1)["Gd"] < 0.5 and not
np.iscomplex(estimate_parameters(df1)["B/A"]):
        p_params(df1, cycle=1, file_name=input_dir.stem)
    else:
        if not np.iscomplex(estimate_parameters(df2)["B/A"]):
            p_params(df2, cycle=2, file_name=input_dir.stem)
        else:
            p_params(df3, cycle=3, file_name=input_dir.stem)

param_dataframes = [s_params(df, cycle=0, file_name=input_dir.stem),
                    s_params(df1, cycle=1, file_name=input_dir.stem),
                    s_params(df2, cycle=2, file_name=input_dir.stem),
                    s_params(df3, cycle=3, file_name=input_dir.stem)]

result_df = pd.concat(param_dataframes, ignore_index=True)

result_df.to_excel(output_dir, index=False, header=True)
```