```python
# This script is used to analyse and plot the data in different ways.

# Authors:
# Christopher Mahn
# Silas Teske
# Joshua Wolf
# Lukas Schulz
# Maria Riegel

#
#############################################################################
#

# Import of libraries
import main
import os
import matplotlib.pyplot as plt
import numpy as np
import shutil
from scipy.fft import fft, fftfreq
# from scipy import signal

#
-----------------------------------------------------------------------------
# Functions

def import_profile(filename):
    """
    This function imports a profile from a .csv-file.
    """
    try:
        with open(os.path.join("data_split", filename), "r") as f:
            data = f.readlines()
        points = {"number": [], "x": [], "y": [], "z": [], "intensity": [],
"time": []}
        for line in data:
            line = line.split(";")
            points["number"].append(int(line[1]))
            points["x"].append(float(line[2]))
            points["y"].append(float(line[3]))
            points["z"].append(float(line[4]))
            points["intensity"].append(float(line[5]))
            points["time"].append(float(line[6]))
        return(points)
    except FileNotFoundError:
        print(f'[ERROR] File "{filename}" not found in the folder
"data_split". Execute the script "profiles_split.py" first.')
        main.terminate()


def fast_fourier_transform(datenreihe, sample_rate):
    """
    Diese Funktion macht eine Fast-Fourier-Transformation und gibt die
Frequenz-
    und Amplitudengraphen zurück.
    """
    N = len(datenreihe)
    T = 1.0 / sample_rate
    yf = fft(datenreihe)
    xf = fftfreq(N, T)[:N//2]
```

```
55      return(xf, 2.0/N * np.abs(yf[0:N//2]))
56
57
58  def plot_werte(datenreihen, name=["Messwerte"], title=None, diagram="show"):
59      """
60      Diese Funktion nimmt Datenreihen und plottet diese in ein Diagramm.
61      """
62      plt.cla()
63      for i, datenreihe in enumerate(datenreihen):
64          zeit = range(len(datenreihe))
65          plt.plot(zeit, datenreihe)
66      plt.legend(name)
67      plt.grid()
68      plt.xlabel("")
69      plt.ylabel("")
70      if(title != None):
71          plt.title(title)
72      else:
73          plt.title(name[0])
74      if(diagram == "show"):
75          plt.show()
76      elif(diagram == "save"):
77          plt.savefig(os.path.join("plots", title + ".png"))
78
79
80  def plot_xy(datenreihen, name=["Messwerte"], x="X", y="Y", title=None,
    diagram="show", size={'x_min': 0, 'x_max': 0, 'y_min': 0, 'y_max': 0},
    fixed_size=False):
81      """
82      Diese Funktion nimmt je zwei Datenreihen und plottet diese in
    Abhängigkeit
83      zueinander in ein Diagramm.
84      """
85      plt.clf()
86      for datenreihe in datenreihen:
87          plt.plot(datenreihe[0], datenreihe[1])
88      plt.legend(name)
89      plt.grid()
90      plt.xlabel(x)
91      plt.ylabel(y)
92      if(fixed_size):
93          plt.xlim(size['x_min'], size['x_max'])
94          plt.ylim(size['y_min'], size['y_max'])
95      if(title != None):
96          plt.title(title)
97      else:
98          plt.title(name[0])
99      if(diagram == "show"):
100         plt.show()
101     elif(diagram == "save"):
102         plt.savefig(os.path.join("plots", title + ".png"))
103
104
105 #
    ------------------------------------------------------------------------
106 # Classes
107
108 #
    ------------------------------------------------------------------------
109 # Beginning of main program
```

```
110
111 if(__name__=='__main__'):
112     print(f'[INFO] Deleting old files in folder "plots"')
113     shutil.rmtree("plots", ignore_errors=True)
114     print(f'[INFO] Creating new folder "plots"')
115     os.mkdir("plots")
116
117     '''
118     # Plotting the profiles in 3D with intensity as color (X, Y, Z)
119     offset = 0.002  # Offset for each profile
120     subsample = 20  # Only plot every n-th profile
121     for dataset in main.datasets:
122         plot = plt.figure().add_subplot(projection='3d')
123         for profile in range(dataset["profiles"]):
124             if(profile % subsample == 0):
125                 print(f'[INFO][{profile/dataset["profiles"]*100:5.1f}%]
    Plotting profiles', end='\r')
126                 # Plot in 3D with an offset for each profile
127                 profile_points =
    import_profile(f'{dataset["filename"].split(".")[0]}_{profile+1:05d}.csv')
128                 plot.scatter(np.array(profile_points["x"])+(profile*offset),
129                              profile_points["y"],
130                              profile_points["z"],
131                              cmap='viridis',
132                              linewidth=0.5)
133         print(f'[INFO][100.0%] Plotting profiles')
134         plot.set_xlabel('X')
135         plot.set_ylabel('Y')
136         plot.set_zlabel('Z')
137         plot.set_title('3D-Plot')
138         plot.set_ylim3d(-1.5, 1.5)
139         plot.set_zlim3d(-2.0, 1.0)
140         plt.show()
141         plt.clf()
142     '''
143
144     # Plotting all profiles in 2D (Y, Z)
145     for dataset in main.datasets:
146         for profile in range(dataset["profiles"]):
147             if(profile % 25 == 0):
148                 print(f'[INFO][{profile/dataset["profiles"]*100:5.1f}%]
    Plotting profiles', end='\r')
149             profile_points = import_profile(f'{dataset["filename"].split(".")
    [0]}_{profile+1:05d}.csv')
150             plot_xy([[profile_points["y"], profile_points["z"]]], x="Y",
    y="Z",
151                     title=f'slice_{dataset["filename"].split(".")
    [0]}_profile{profile+1:05d}', diagram="save",
152                     size={'x_min': -1.5, 'x_max': 1.5, 'y_min': -1.5,
    'y_max': 0.8}, fixed_size=True)
153         print(f'[INFO][100.0%] Plotting profiles')
154         plt.clf()
155
156     # Plotting all profiles in 2D as Spectrum (value, index)
157     for dataset in main.datasets:
158         for profile in range(dataset["profiles"]):
159             if(profile % 25 == 0):
160                 print(f'[INFO][{profile/dataset["profiles"]*100:5.1f}%]
    Plotting spectrums', end='\r')
161             profile_points = import_profile(f'{dataset["filename"].split(".")
```

```
    [0]}_{profile+1:05d}.csv')
162             indices = range(len(profile_points["intensity"]))
163             plot_xy([[indices, profile_points["intensity"]],
164                      [indices, profile_points["x"]],
165                      [indices, profile_points["y"]],
166                      [indices, profile_points["z"]]],
167                     name=["Intensity", "X", "Y", "Z"], x="index [1]",
    y="Koord. [m] / Intensity [1]",
168                     size={'x_min': 0, 'x_max': len(indices), 'y_min': -1.5,
    'y_max': 1.5}, fixed_size=True,
169                     title=f'spectrum_{dataset["filename"].split(".")
    [0]}_profile{profile+1:05d}', diagram="save")
170         print(f'[INFO][100.0%] Plotting spectrums')
171         plt.clf()
172 
173     # Getting timeseries of horizontal and vertical movement
174     for dataset in main.datasets:
175         timeseries = {"left": [], "right": [], "top": []}
176         for profile in range(dataset["profiles"]):
177             if(profile % 25 == 0):
178                 print(f'[INFO][{profile/dataset["profiles"]*100:5.1f}%]
    Getting timeseries', end='\r')
179             profile_points = import_profile(f'{dataset["filename"].split(".")
    [0]}_{profile+1:05d}.csv')
180             left_0 = int(dataset["left"][0])
181             left_1 = int(dataset["left"][1])
182             right_0 = int(dataset["right"][0])
183             right_1 = int(dataset["right"][1])
184             top_0 = int(dataset["top"][0])
185             top_1 = int(dataset["top"][1])
186             left = np.array(profile_points["y"])[left_0:left_1].mean()
187             right = np.array(profile_points["y"])[right_0:right_1].mean()
188             top = np.array(profile_points["z"])[top_0:top_1].mean()
189             timeseries["left"].append(left)
190             timeseries["right"].append(right)
191             timeseries["top"].append(top)
192         plot_xy([[range(len(timeseries["left"])), timeseries["left"]],
193                  [range(len(timeseries["right"])), timeseries["right"]],
194                  [range(len(timeseries["top"])), timeseries["top"]]],
195                 name=["left", "right", "top"], x="index [1]", y="Koord. [m]",
196                 title=f'timeseries_{dataset["filename"].split(".")[0]}',
    diagram="save")
197         print(f'[INFO][100.0%] Getting timeseries')
198         print(f'[INFO] Generating FFT-plots')
199 
200         # Plotting frequency spectrum of horizontal and vertical movement
201         # left
202         fft_x, fft_y = fast_fourier_transform(timeseries["left"],
    dataset["sample_rate"])
203         plot_xy([[fft_x, fft_y]], x="Frequency [Hz]", y="Amplitude [1]",
    title=f'fft_{dataset["filename"].split(".")[0]}_left', diagram="save",
204                 size={'x_min': 0, 'x_max': 10, 'y_min': 0, 'y_max': 0.05},
    fixed_size=True)
205 
206         # right
207         fft_x, fft_y = fast_fourier_transform(timeseries["right"],
    dataset["sample_rate"])
208         plot_xy([[fft_x, fft_y]], x="Frequency [Hz]", y="Amplitude [1]",
    title=f'fft_{dataset["filename"].split(".")[0]}_right', diagram="save",
209                 size={'x_min': 0, 'x_max': 10, 'y_min': 0, 'y_max': 0.05},
```

```
    fixed_size=True)
210
211         # top
212         fft_x, fft_y = fast_fourier_transform(timeseries["top"],
    dataset["sample_rate"])
213         plot_xy([[fft_x, fft_y]], x="Frequency [Hz]", y="Amplitude [1]",
    title=f'fft_{dataset["filename"].split(".")[0]}_top', diagram="save",
214                 size={'x_min': 0, 'x_max': 10, 'y_min': 0, 'y_max': 0.05},
    fixed_size=True)
215
```