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

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

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

#
------------------------------------------------------------------------
# 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.')
        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]
    return(xf, 2.0/N * np.abs(yf[0:N//2]))


def plot_werte(datenreihen, name=["Messwerte"], title=None, diagram="show"):
```

```
57      """
58      Diese Funktion nimmt Datenreihen und plottet diese in ein Diagramm.
59      """
60      plt.cla()
61      for i, datenreihe in enumerate(datenreihen):
62          zeit = range(len(datenreihe))
63          plt.plot(zeit, datenreihe)
64      plt.legend(name)
65      plt.grid()
66      plt.xlabel("")
67      plt.ylabel("")
68      if(title != None):
69          plt.title(title)
70      else:
71          plt.title(name[0])
72      if(diagram == "show"):
73          plt.show()
74      elif(diagram == "save"):
75          plt.savefig(os.path.join("plots", title + ".png"))
76
77
78  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):
79      """
80      Diese Funktion nimmt je zwei Datenreihen und plottet diese in
    Abhängigkeit
81      zueinander in ein Diagramm.
82      """
83      plt.clf()
84      for datenreihe in datenreihen:
85          plt.plot(datenreihe[0], datenreihe[1])
86      plt.legend(name)
87      plt.grid()
88      plt.xlabel(x)
89      plt.ylabel(y)
90      if(fixed_size):
91          plt.xlim(size['x_min'], size['x_max'])
92          plt.ylim(size['y_min'], size['y_max'])
93      if(title != None):
94          plt.title(title)
95      else:
96          plt.title(name[0])
97      if(diagram == "show"):
98          plt.show()
99      elif(diagram == "save"):
100         plt.savefig(os.path.join("plots", title + ".png"))
101
102
103 #
    ------------------------------------------------------------------------
104 # Classes
105
106 #
    ------------------------------------------------------------------------
107 # Beginning of main program
108
109 if(__name__=='__main__'):
110     print(f'[INFO] Deleting old files in folder "plots"')
111     shutil.rmtree("plots", ignore_errors=True)
```

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

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

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