МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

Факультет радіоелектроніки, комп'ютерних систем та інфокомунікацій

**Лабораторна робота № 3 `**

з «Обробка даних засобами Python»

(назва дисципліни)

тема: «Дослідження можливостей бібліотеки Pandas»

Виконав: студент 5 курсу групи № 555вМн

напряму підготовки (спеціальності)

123 Системне програмування

(шифр і назва напряму підготовки (спеціальності))

Логачов М.Г.

(прізвище й ініціали студента)

Прийняв: д.т.н., доцент

Дергачов К.Ю.

(посада, науковий ступінь, прізвище й ініціали)

Національна шкала: \_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_

Харків – 2023

Лабораторна робота №3

**Тема**: Дослідження можливостей бібліотеки Pandas.

**Мета роботи :** Сформувати у здобувачів компетенції використання засобів бібліотеки Pandas для вирішення прикладних завдань.

**Хід роботи**

1. За даними отриманого файлу формату Novatel DualGPS необхідно побудувати графічні залежності кутів курсу( head(t) (стовпчик даних № 13)) та куту крену (pitch(t) стовпчик № 14) від часу вимірювань, час знаходиться у 7 стовпчику та вимірюється у секундах, а кути у градусах.
2. Необхідно додати графічні залежності для усереднених даних вказаних кутів із шагом 10 с, та 20 с від часу вимірювань
3. Отримати статистичні характеристики (середнє значення, середньо квадратичне відхилення та максимальне відхилення від середнього значення)

Фрагмент набору даних вимірювань 2-антеного GPS-приймача згідно варінту (№3).

Повний набір данних можна знайти за посиланням на [GitHub](https://raw.githubusercontent.com/lohachovKhai/geodata_processing/main/Lab_3/gpsData.csv)

#DUALANTENNAHEADINGA,COM1\_1,0,50.0,FINESTEERING,2121,299352.000,02000000,d426,15823;SOL\_COMPUTED,NARROW\_INT,3.978764772,97.283775330,-0.042816162,0.0,0.057393141,0.101939708,"T28M",33,25,25,22,04,01,30,33\*19660dd2

#DUALANTENNAHEADINGA,COM1\_1,0,50.0,FINESTEERING,2121,299353.000,02000000,d426,15823;SOL\_COMPUTED,NARROW\_INT,3.978662252,97.276016235,-0.022979736,0.0,0.057645224,0.102100216,"T28M",33,25,25,22,04,01,30,33\*d5907093

#DUALANTENNAHEADINGA,COM1\_1,0,50.0,FINESTEERING,2121,299354.000,02000000,d426,15823;SOL\_COMPUTED,NARROW\_INT,3.977462292,97.265411377,-0.043304443,0.0,0.058915213,0.104407594,"T28M",33,26,26,22,04,01,30,33\*48c86360

...

#DUALANTENNAHEADINGA,COM1\_1,0,54.0,FINESTEERING,2121,303170.000,02000000,d426,15823;SOL\_COMPUTED,NARROW\_INT,3.974961519,97.320175171,-0.099822998,0.0,0.058004253,0.083366916,"T28M",31,24,24,22,04,01,30,33\*7661de58

#DUALANTENNAHEADINGA,COM1\_1,0,53.5,FINESTEERING,2121,303171.000,02000000,d426,15823;SOL\_COMPUTED,NARROW\_INT,3.975607872,97.315719604,-0.065124512,0.0,0.058050513,0.083430059,"T28M",31,24,24,22,04,01,30,33\*e5ac5c21

#DUALANTENNAHEADINGA,COM1\_1,0,53.5,FINESTEERING,2121,303173.000,02000000,d426,15823;SOL\_COMPUTED,NARROW\_INT,3.975305319,97.325744629,-0.131896973,0.0,0.058161497,0.083857000,"T28M",31,24,24,22,04,01,30,33\*4508ef00

Приклад парсингу отриманих даних

*3.978764772* – набір даних вимірювань кута крену  
*97.283775330* – набір даних вимірювань кута курсу

*299352.000* – час вимірювання

В ході виконання лабораторної роботи було створено програмний код за допомогою мови програмування Python. Використовиуючи бібліотеки *NumPy  
matplotlib, mplcursors, pandas* були побудовані графіки залежностей кута курсу та кута крену від часу.

Необхідні залежності представлені у файлі *requirements.txt* додаток 1.

Для роботи з цим файлом необхідно в робочій деректорії виконати команди за добомогою терміналу GitBash.

* ця команда дозволить встановити віртуальне оточення в робочій дерикторії

source venv/Scripts/activate

* ця команда вказує на файл з необхідними залежностями

pip freeze > requirements.txt

* безпосереднє встановлення потрібних бібліотек

pip install -r requirements.txt

За допомогою програмного обчислення для отриманих даних кутів курсу та кутів крену в залежності від часу було розраховано середнє значення, середньо квадратичне відхилення та максимальне відхилення від середнього значення.

Використовуючи бібліотеки *Pandas* та *Python-Docx* отримані результати представлені в таблиці 1.

A picture containing diagram

Description automatically generatedСереднє значення - сума всіх фіксованих значень набору, поділена на кількість елементів набору (1).

(1)

Функція **numpy.mean**(*a*, *axis=None*, *dtype=None*, *out=None)* бібліотеки *NumPy* дозволяє швидко та без зайвих кроків знайти середнє значення набору даних «*a».*

Text, schematic

Description automatically generatedСереднє квадратичне відхилення так само, як і середнє лінійне відхилення, показує, на скільки в середньому відхиляються конкретні значення ознаки від середнього їх значення (2).

(2)

Для масиву значень «*a*» Середнє квадратичне відхилення можна обчислити, використовуючи функцію «з коробки» :

**numpy.std**(a, axis=None, dtype=None, out=None, ddof=0, keepdims=<no value>)

Таблиця 1. Результати обчисленнь

|  |  |  |
| --- | --- | --- |
| Index | Parameter | Value |
| 1 | Середнє значення (head) | 97.30772704 |
| 2 | Середнє значення (pitch) | 0.01615009 |
| 3 | Максимальне відхилення від середнього значення (head) | 0.13384706 |
| 4 | Максимальне відхилення від середнього значення (pitch) | 0.23900598 |
| 5 | Середньо квадратичне відхилення (head) | 0.04177437 |
| 6 | Середньо квадратичне відхилення (pitch) | 0.06520468 |
| 7 | Середнє значення з кроком 10с (head) | 97.3076231 |
| 8 | Середнє значення з кроком 10с (pitch) | 0.0161583 |
| 9 | Максимальне відхилення від середнього значення з кроком 10с (head) | 0.10883427 |
| 10 | Максимальне відхилення від середнього значення з кроком 10с (pitch) | 0.19327074 |
| 11 | Середньо квадратичне відхилення з кроком 10с (head) | 0.04000564 |
| 12 | Середньо квадратичне відхилення з кроком 10с (pitch) | 0.06025725 |
| 13 | Середнє значення з кроком 20с (head) | 97.30750947 |
| 14 | Середнє значення з кроком 20с (pitch) | 0.01618329 |
| 15 | Максимальне відхилення від середнього значення з кроком 20с (head) | 0.10547995 |
| 16 | Максимальне відхилення від середнього значення з кроком 20с (pitch) | 0.18761266 |
| 17 | Середньо квадратичне відхилення з кроком 20с (head) | 0.03952901 |
| 18 | Середньо квадратичне відхилення з кроком 20с (pitch) | 0.05874905 |

Графічні залежності кута крену від часу, кута курсу від часу та їх середніх значень в інтервалах 10 та 20 секунд представлені на рисунках 1-6.

Chart, histogram

Description automatically generatedРисунок 1. Графік залежності кутів курсу від часу

Chart

Description automatically generatedРисунок 2. Графік залежності кутів крену від часу

Chart

Description automatically generatedРисунок 3. Графік залежності середніх значень кутів курсу від часу з інтервалом 10 секунд

Chart

Description automatically generatedРисунок 4. Графік залежності середніх значень кутів крену від часу з інтервалом 10 секунд

Рисунок 5. Графік залежності середніх значень кутів курсу від часу з інтервалом 20 секунд

Chart

Description automatically generated

Рисунок 6. Графік залежності середніх значень кутів крену від часу з інтервалом 20 секунд

Chart

Description automatically generated

Висновки

Використання компєютерного обчислюівання при обробці статистичних даних значно пришвидшує процес та робить більш точним. Також великою переваю являє собою можливість автоматичного представлення результатів у вигляді графіків і таблиць.

Додаток 1

[Посилання на Git репозиторій з вихідним кодом](https://github.com/lohachovKhai/geodata_processing/tree/main/Lab_3)

Модуль main.py

import matplotlib.pyplot as plt

import numpy as np

import mplcursors

import pandas as pd

import docx as docx

def statCalulator(array, type):

match type:

# середнє значення

case "average":

return round(np.mean(array), 8)

# середнє квадратичне відхилення

case "std":

return round(np.std(array), 8)

# максимальне відхилення від середнього значення

case "maxAverageDeviation":

return round(max(list(map(lambda x: x != 0 and abs(np.mean(array) - x), array))), 8)

def ticksSeparator(timeArray, dataArray, ticks):

result = []

result.append([timeArray[0], dataArray[0]])

for i in range(ticks,len(timeArray)-1,ticks):

tmpRes = []

for j in range(i-ticks, i):

if timeArray[i] > (timeArray[i - ticks] + ticks):

dataArray[i] = (dataArray[i-1]+dataArray[i+1])/2

elif timeArray[i] < (timeArray[i - ticks] + ticks):

dataArray[i] = (dataArray[i] + dataArray[i + 1]) / 2

tmpRes.append(dataArray[j])

tmp= np.mean(tmpRes)

result.append([timeArray[i], tmp])

return np.array(result)

def chartBuilder(xData, yData, title):

average = statCalulator(yData, 'average')

maxAverDiv = statCalulator(yData, 'maxAverageDeviation')

def textRun(annotation):

syt = annotation.replace('x', "seconds")

syt = syt.replace('y', "degree")

dd = list(map(str, syt.split('=')))

valueCon = None

try:

valueCon = float(dd[len(dd) - 1])

except ValueError:

valueCon = float(dd[len(dd) - 1][1:]) \* (-1)

diff = abs(average - valueCon)

syt = syt + "\n deviation " + str(diff)

return syt

averageArray = np.full(len(xData),average)

plt.style.use('seaborn-v0\_8')

fig, ax = plt.subplots()

fig.set(facecolor='lightblue')

ax.plot(xData, yData, color="blue")

ax.plot(xData, averageArray, color='red')

ax.set\_xlabel("seconds", fontsize=20)

ax.set\_ylabel("degree", fontsize=20)

plt.title(title + " average value: " + str(round(average, 8)), fontsize=20)

dots = ax.scatter(xData, yData, color='darkred')

crs = mplcursors.cursor(dots, hover=True)

crs.connect("add", lambda sel: sel.annotation.set\_text(textRun(sel.annotation.get\_text())))

for x, y in zip(xData, yData):

curPoint = (float(abs(float(y) - average)))

if curPoint == maxAverDiv:

ax.scatter(x, y, color='red')

plt.legend(['dependency', 'Average', 'checkpoint', 'maxDif'], loc='center left', bbox\_to\_anchor=(1.0, 0.5),

fontsize=12, frameon=True)

plt.show()

def genearateTable(headArr, pitchArr, timeArr, fileName):

doc = docx.Document()

parameters = ['Середнє значення (head)',

'Середнє значення (pitch)',

'Максимальне відхилення від середнього значення (head)',

'Максимальне відхилення від середнього значення (pitch)',

'Середньо квадратичне відхилення (head)',

'Середньо квадратичне відхилення (pitch)',

'Середнє значення з кроком 10с (head)',

'Середнє значення з кроком 10с (pitch)',

'Максимальне відхилення від середнього значення з кроком 10с (head)',

'Максимальне відхилення від середнього значення з кроком 10с (pitch)',

'Середньо квадратичне відхилення з кроком 10с (head)',

'Середньо квадратичне відхилення з кроком 10с (pitch)',

'Середнє значення з кроком 20с (head)',

'Середнє значення з кроком 20с (pitch)',

'Максимальне відхилення від середнього значення з кроком 20с (head)',

'Максимальне відхилення від середнього значення з кроком 20с (pitch)',

'Середньо квадратичне відхилення з кроком 20с (head)',

'Середньо квадратичне відхилення з кроком 20с (pitch)'

]

values = [statCalulator(headArr, 'average'),

statCalulator(pitchArr, 'average'),

statCalulator(headArr, 'maxAverageDeviation'),

statCalulator(pitchArr, 'maxAverageDeviation'),

statCalulator(headArr, 'std'),

statCalulator(pitchArr, 'std'),

statCalulator((ticksSeparator(timeArr, headArr, 10)[:, 1]), 'average'),

statCalulator((ticksSeparator(timeArr, pitchArr, 10)[:, 1]), 'average'),

statCalulator((ticksSeparator(timeArr, headArr, 10)[:, 1]), 'maxAverageDeviation'),

statCalulator((ticksSeparator(timeArr, pitchArr, 10)[:, 1]), 'maxAverageDeviation'),

statCalulator((ticksSeparator(timeArr, headArr, 10)[:, 1]), 'std'),

statCalulator((ticksSeparator(timeArr, pitchArr, 10)[:, 1]), 'std'),

statCalulator((ticksSeparator(timeArr, headArr, 20)[:, 1]), 'average'),

statCalulator((ticksSeparator(timeArr, pitchArr, 20)[:, 1]), 'average'),

statCalulator((ticksSeparator(timeArr, headArr, 20)[:, 1]), 'maxAverageDeviation'),

statCalulator((ticksSeparator(timeArr, pitchArr, 20)[:, 1]), 'maxAverageDeviation'),

statCalulator((ticksSeparator(timeArr, headArr, 20)[:, 1]), 'std'),

statCalulator((ticksSeparator(timeArr, pitchArr, 20)[:, 1]), 'std')

]

index = np.arange(1, np.array(values).size + 1)

table\_data = {'Index': index, 'Parameter': parameters, 'Value': values}

df = pd.DataFrame(data=table\_data)

# Initialise the table

t = doc.add\_table(rows=(df.shape[0] + 1), cols=df.shape[1])

t.style = 'Light Shading Accent 1'

# Add the column headings

for j in range(df.shape[1]):

t.cell(0, j).text = df.columns[j]

# Add the body of the data frame

for i in range(df.shape[0]):

for j in range(df.shape[1]):

cell = df.iat[i, j]

t.cell(i + 1, j).text = str(cell)

doc.save(fileName + '.docx')

dataGPS = []

d = {}

gpsData = open('gpsData.csv', 'r')

for line in gpsData.readlines():

s = list(map(str, line.split(',')))

dataGPS.append(s)

def valueParser(columnNumber):

tmp = []

for i in range(len(dataGPS)):

tmp.append(dataGPS[i][columnNumber - 1])

return np.array(tmp, dtype=np.float64)

# Get values according to the variant

d['head'] = valueParser(13)

d['pitch'] = valueParser(14)

d['time'] = valueParser(7)

chartBuilder(d['time'], d['head'], 'Dependence of time on (head) degrees')

chartBuilder(d['time'], d['pitch'], 'Dependence of time on (pitch) degrees')

chartBuilder((ticksSeparator(d['time'], d['head'], 10)[:, 0]), (ticksSeparator(d['time'], d['head'], 10)[:, 1]), 'Dependence of time on (head) degrees 10s intervel')

chartBuilder((ticksSeparator(d['time'], d['pitch'], 10)[:, 0]), (ticksSeparator(d['time'], d['pitch'], 10)[:, 1]), 'Dependence of time on (pitch) degrees 10s intervel')

chartBuilder((ticksSeparator(d['time'], d['head'], 20)[:, 0]), (ticksSeparator(d['time'], d['head'], 20)[:, 1]), 'Dependence of time on (head) degrees 20s intervel')

chartBuilder((ticksSeparator(d['time'], d['pitch'], 20)[:, 0]), (ticksSeparator(d['time'], d['pitch'], 20)[:, 1]), 'Dependence of time on (pitch) degrees 20s intervel')

genearateTable(d['head'], d['pitch'], d['time'], 'Results')

Файл requirements.txt

matplotlib

numpy

mplcursors

pandas

python-docx