Практическая работа № 8

Работа с полетными логами

На прошлых занятиях мы начали знакомство с массивами NumPy и инструментами их визуализации. Сегодня мы используем их для анализа лога, записанного во время полета экспериментального БПЛА. Лог содержит следующую информацию:

- высота по показаниям барометрического датчика (BaroAlt)
- положения левого и правого элевонов (LE, RE)
- данные с GPS (широта, долгота, высота над уровнем моря, модуль скорости над землей, курс) (Lang, Lat, MSL_A, SoG, CoG)
- угол тангажа (Pitch)
- угол крена (Roll)
- положение ручки газа (Throttle)
- атмосферное давление (Press)

Даные снимались с частотой 50 Гц.

Логирующее устройство создавалось в парадигме модельно-ориентированного проектирования, поэтому лог был сохранен как mat-файл. Мы начнем с того, что прочитаем этот файл средствами Python.

1. Чтение .mat-файлов

In [ ]:

```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
```

.mat - довольно специфичный формат хранения данных, чтобы его корректно прочитать, нам понадобится специальный модуль из библиотеки SciPy. SciPy - специальная библиотека функций для научных вычислений, основана на массивах NumPy. Если вы использует Анаконду, эта библиотека у вас уже установлена. В противном случае установите ее при помощи команды

pip install scipy

In [ ]:

```
import scipy
from scipy.io import loadmat
```

In [ ]:

```
data = loadmat('Log.mat')
```

In [ ]:

```
data
```

Итак, данные прочитаны и сразу сохранены в словарь. При этом имена переменных были использованы как ключи, а значения - как значения по этим ключам.

In [ ]:

```
type(data)
```

В начале идут некоторые метаданные, сейчас они нам мало интересны. Собственно лог начинается с ключа BaroAlt. Мы можем извлечь любой интересующий нас параметр просто по ключу

In [ ]:

```
data.keys()
```

Оценим количество измерений по любому из параметров

In [ ]:

```
len(data['Lang'])
```

Данные в исходном файле являются "сырыми", т.е. могут содержать помехи, артефакты, могли быть вообще не записаны, если у какого-то из датчиков пропал контакт, отпало питание и пр., в полете всякое случается. Давайте отстроим их.

2. Визуализация сырых данных

Начнем с данных о высоте полета. Она фиксировалась двумя датчиками: барометрическим и GPS. Высота записывалась в метрах. Для барометра относительно некоторого колибровочного значения, по GPS - относительно уровня моря.

In [ ]:

```
plt.figure(figsize = [15,8], dpi=200)
plt.subplot(2,1,1)
plt.plot(data['BaroAlt'])
plt.grid(True)
plt.title('Высота по барометрическому датчику')

plt.subplot(2,1,2)
plt.plot(data['MSL_A'])
plt.grid(True)
plt.title('Высота по GPS')
```

Что можно сказать по этим графикам? Во-первых, в них наблюдается значительное количество артефактных значений. Во-вторых, есть вопросы к калибровке барометрического датчика. Однако, в общем и целом, сырые данные выглядят вполне правдоподобно.

Посмотрим теперь, как во время полета менялось положение "ручки газа"

In [ ]:

```
plt.figure(figsize = [15,8], dpi=200)
plt.plot(data['Throttle'])
plt.grid(True)
```

График выглядит не слишком обнадеживающим. В таком виде по нему сложно судить, записалось ли что-либо вообще, из-за чудовищной величины артефакта (34-я степень(!)). Поскольку на данном этапе стоит задача визуализации данных (ну еще возможно первичного анализа "методом пристального взгляда")), а не обработки, давай попробуем просто настроить область отображения осей. Как выбрать эту область? Нужно понимать порядок величины, которую мы собираемся наблюдать. Есть разные способы его узнать (кстати подумайте сами - какие?). Для примера давайте просто определим среднее значение исследуемого параметра в области, где он сейчас визуально выглядит нулевым.

In [ ]:

```
np.mean(data['Throttle'][0:1000])
```

Попробуйте изменять размер этой области и понаблюдайте за результатом

Видно, что порядок данных это - в лучшем случае десятки, но никак не сотни, и уж точно не 1e34. Давайте настроим пределы отображения осей с учетом этой информации.

In [ ]:

```
plt.figure(figsize = [15,8], dpi=200)
plt.plot(data['Throttle'])
plt.grid(True)
plt.ylim(0,20)
plt.title('Throttle')
```

Так уже можно наблюдать что-то осмысленное. Как и предыдущие, этот параметр был записан с некотором количеством артефактных значений. Также можно наблюдать смещение 0. (Но в целом, график выглядит правдоподобно, и если смотреть внимательно, то видно, что он согласуется с графиками изменения высоты полета. Т.е. после некоторй обработки эти данные будут пригодны для дальнейшего анализа.

Теперь посмотрим, как изменялись углы установки управляющих элементов

Кстати, поскольку тут речь идет об углах (в градусах), мы можем сразу предположить, что корректные значения точно не могут покинуть пределов +- 90 градусов, а реально должны находиться в гораздо более узких пределах

In [ ]:

```
# Правый элевон
plt.figure(figsize = [15,8], dpi=200)
plt.subplot(2,1,1)
plt.plot(data['RE'])
plt.grid(True)
plt.ylim(5,10)
plt.title('RE')

# Левый элевон
plt.subplot(2,1,2)
plt.plot(data['LE'])
plt.grid(True)
plt.ylim(5,10)
plt.title('LE')
```

Теперь рассмотрим координаты

In [ ]:

```
# Широта
plt.figure(figsize = [15,8], dpi=200)
plt.subplot(2,1,1)
plt.plot(data['Lat'])
plt.grid(True)
plt.title('Lat')

# Долгота
plt.subplot(2,1,2)
plt.plot(data['Lang'])
plt.grid(True)
plt.title('Lang')
```

Как и до этого, наличие артефактов мешает сразу увидеть изменение интересующих нас параметров. Давайте напечатаем небольшую часть этих данных, чтобы понимать, в каком формате координаты были записаны.

In [ ]:

```
for ind, item in enumerate(data['Lat'][0:50]):
    print(ind,'Lat: ',item,'Lang: ', data['Lang'][ind])
```

Можно видеть следующее: во-первых, датчик начал передавать данные только с 45 тика, т.е. через 45x0.2 = 9 секунд, во-вторых, с учетом того, что лог был записан где-то в окрестностях Таганога, координаты не вполне типичны: похоже, что точкой отделены секунды, а не минуты. Попробуйте взять первую значимою пару и переместить точку на 2 знака вперед: (47.14021 38.544978). Эти координаты уже болше похожи на координаты окрестностей Таганрога. Скопируйте их в поисковую строку на google-картах, и получите примерное положение точки взлета на поле, на котором был записан исследуемый лог.

Пока не стоит задача какой-либо обработки данных, но на графике мы для собственного удобства можем переместить оси к стартовой точке. Что касается пределов отображения, очевидно, что поле, на котором проводились испытания, не настолько огромно, чтобы в координатах менялись градусы или даже минуты, значит мы ожидаем изменения только в секундах. В соответствии с этим и настроим пределы.

In [ ]:

```
start_spot = [4714.021, 3854.4978]
# Широта
plt.figure(figsize = [15,8], dpi=200)
plt.subplot(2,1,1)
plt.plot(data['Lat'] - start_spot[0])
plt.grid(True)
plt.ylim(-0.05,0.05)
plt.title('Lat')

# Долгота
plt.subplot(2,1,2)
plt.plot(data['Lang'] - start_spot[1])
plt.grid(True)
plt.ylim(-0.05,0.05)
plt.title('Lang')
```

Настроить пределы отображения для углов крена и тангажа значительно проще, т.к. у них есть очевидные экстремальные значения.

In [ ]:

```
# Крен
plt.figure(figsize = [15,8], dpi=200)
plt.subplot(2,1,1)
plt.plot(data['Roll'])
plt.grid(True)
plt.ylim(-90,90)
plt.title('Roll')

# Тангаж
plt.subplot(2,1,2)
plt.plot(data['Pitch'])
plt.grid(True)
plt.ylim(-90,90)
plt.title('Pitch')
```

Пределы для графиков курса, линейной скорости относительно земли и атмосферного давления также следует выбирать, изходя из природы этих величин и соответствующих адекватных значений.

In [ ]:

```
# Курс
plt.figure(figsize = [15,8], dpi=200)
plt.subplot(2,1,1)
plt.plot(data['CoG'])
plt.grid(True)
plt.ylim(0,360)
plt.title('CoG')

# Скорость над землей
plt.subplot(2,1,2)
plt.plot(data['SoG'])
plt.grid(True)
plt.ylim(0,25)
plt.title('SoG')
```

In [ ]:

```
plt.figure(figsize = [15,8], dpi =150)
Press = np.array(data['Press'])
plt.plot(Press)
plt.grid(True)
plt.ylim(75000, 78000)
```

Итак, мы отстроили графики собранных в полете данных. Очевидно, что оборудование, при помощи которого этот лог был записан, на тот момент не было отлажено, лог содержит много артефактов. Тем не менее, мы все еще можем сделать данные пригодными для дальнейшей работы, очистив от ошибок.

3. Очистка данных

Наблюдаемые в данный момент артефакты выглядят как огромной величины скачки. Давайте создадим функцию, которая будет обнаруживать их. Внимание, вопрос: как обнаружить эти мега-скачки?

Давайте начнем данных о координатах аппарата и высоте полета (Lang и Lat, MSL_A).

Самостоятельно Создайте функцию, которая принимает на вход словарь с данными и строит все 3 графика (Lang и Lat, MSL_A) на одной фигуре друг под другом.

In [ ]:

```
# Широта
def plot_GPS(data):

    plt.figure(figsize = [15,8], dpi=200)
    plt.subplot(3,1,1)
    plt.plot(data['Lat'] - start_spot[0])
    plt.grid(True)
    plt.ylim(-0.05,0.05)
    plt.title('Lat')

    # Долгота
    plt.subplot(3,1,2)
    plt.plot(data['Lang'] - start_spot[1])
    plt.grid(True)
    plt.ylim(-0.05,0.05)
    plt.title('Lang')

    plt.subplot(3,1,3)
    plt.plot(data['MSL_A'])
    plt.grid(True)
    plt.title('MSL_A')
```

In [ ]:

```
plot_GPS(data)
```

Можно предложить немало способов первичной очистки данных от артефактных скачков. Мы пойдем по самому простому пути, и создадим функцию, которая будет сравнивать значение параметров с пороговыми значениями, и если параметр выходит за этот толеранс, подменять его предыдущим значением.

Для начала зададимся порогами

In [ ]:

```
threshholds = {}
threshholds['Lat'] = [-0.05 + start_spot[0] ,0.05 + start_spot[0]]
threshholds['Lang'] = [-0.05 + start_spot[1] ,0.05 + start_spot[1]]
threshholds['MSL_A'] = [6, 100]
threshholds
```

Создадим корректирующую функцию

In [ ]:

```
def correct_errors(raw_data,thrsh):
    dt = raw_data[45:].copy()
    for i in range(1, len(dt)):
        if (dt[i] < thrsh[0]) or (dt[i] > thrsh[1]):
            dt[i] = dt[i-1]
    return dt
```

и применим ее к нашим данным

In [ ]:

```
data_clean = {}
data_clean['Lat'] = correct_errors(data['Lat'],threshholds['Lat'])
data_clean['Lang'] = correct_errors(data['Lang'],threshholds['Lang'])
data_clean['MSL_A'] = correct_errors(data['MSL_A'],threshholds['MSL_A'])
```

In [ ]:

```
plot_GPS(data_clean)
```

Итак, мы смогли очистить данные от артефактов записи, и они теперь пригодны для дальнейшей работы: анализа, построения мат.модели БПЛА, и т.п.. Отметим, что мы все это время фактически работали со словарем, содержащим списки. Это не самый эффективный способ использования ресурсов. Что бы вы сделали для оптимизации процесса?

Выводы

Где почитать про SciPy подробнее: https://docs.scipy.org/doc/scipy/getting_started.html (https://docs.scipy.org/doc/scipy/getting_started.html)