第 1 章

Python でデータ処理

物理科学科 4 回生
門野広大

はじめに

研究室に入ったら実験データをグラフにプロットしたりいろんなことをしないといけないと思います。研究室によってデータ処理をするソフトは異なると思いますが、大体は有料で個人で買うにはちょっと…っていう感じになると思います。研究室で作業すれば問題はないと思いますが、家でゆっくり一人で作業したいけど研究室のパソコンをもってかえるのはだるいといった感じになります。

そういうときに Python で簡単なデータ処理ができるようになっているといいと思います。それなら Gnuplot でいいじゃんかという声がしそうなのでちょっと補足しておきます。Python はデータ処理だけでなくいろんなことができるのでデータ処理から入って数値解析などに進展したりすることも数値計算の知識さえあれば簡単にできます。それに今絶賛流行中のラズパイも基本的に Python でプログラミングします。そのほかにもこれから出てくるマイコン (マイクロコンピュータ) の開発環境は Python が増えると思います。データ分析から入って Python の世界に入れば研究の可能性が広がっていきます。

今回の会誌は、実験データをプロットして簡単なフィッティングできるまで解説したいと思います。(自分も初心者なので信用しきらないで下さい)

1.1　環境構築

1.1.1　諸注意

Python を書く環境は調べるとたくさんあり今回紹介するのはあくまで一例なのでまあ気に入らなかったら別の環境にすぐさま移動してください。

1.1.2　vscode

Visual Studio Code (vscode) での Python 構築をします。とっても簡単です。

まず vscode をダウンロードします。`https://code.visualstudio.com/` からインストーラーをダウンロードして、起動させてください。追加タスクの選択をすべて選択しておくと今後便利かも。

次に、vscode の拡張機能を追加します。ファイル→基本設定→拡張機能で Python で検索して一番上にあるやつをインストールしてください。インストールが完了したら再読み込みして、新規ファイルを作成し、

```
print ('Hello world')
```

と書き込み、拡張子を.py のファイル名で保存すると右下に Python がダウンロードされてません的な警告が出るので、ダウンロードボタンを押してください。そうするとブラウザが開くのでそのサイトからインストーラーをダウンロードして起動してください。インストーラーの最初の画面に path の追加云々のチェックがあるのでチェックしてください。そこから先は特に特別なことをしなくても大丈夫です。ダウンロードが終わったら、再度 vscode を開き先ほど保存したファイルを開いてください。

F5 をクリックするだけでデバックができるのでクリックしてください。そうすると下のほうにターミナルが開くので、そのになんか黄色い文字が表示され、なんかコマンドを実行してくださいできなことが書いてあったらそのコマンドをターミナルに書き込んで実行してください。おそらく

```
python -m pip install -U pylint --user
python -m pip install --upgrade pip
```

の二つだと思います。

以上でおそらく環境が構築できると思いますが何か行き詰まったら Google 先生に聞いてください。いろんな人が詳しくブログで紹介してくれているのでこんな適当な説明より

も百倍いいと思います。

1.1.3 ライブラリの追加方法

この後、初期状態では入っていないライブラリを使用します。コマンド一つで追加することができます。デバックを開始したときに入ってないぞと言われたら import や from とかの後ろに書いてあるライブラリの名前を以下のコマンドに入れてターミナルで実行してください。

```
import numpy as np
```

↑これだったらライブラリの名前＝ numpy

```
from matplotlib import pyplot
```

↑これだったらライブラリの名前＝ matplotlib

コマンド

```
python -m pip install ライブラリの名前
```

1.2 Python の基礎

今回使用する主な文法についてだけです。詳しくは本やブログを参照してください。

1.2.1 主な形

Python のプログラム（Python には限らず）は基本的に上から順番に実行していきます。初めにライブラリを追加するプログラムを入れます。

```
from matplotlib import pyplot
import numpy as np
import math
```

import + ライブラリの名前

で基本的にできます。ライブラリでそのプログラムで用意されている関数をしようするとき

ライブラリの名前. 関数名 ()

のような形で使用しないといけないので、ライブラリの名前が長いと面倒なので

```
import + ライブラリの名前 + as + 今後使用する名前
```

といった形に書いてやると関数の名前を短くできます。

ライブラリを追加したらそのあとは普通にプログラムを書くだけです。

自作の関数を作成するには

```
def func(x ,intensity ,X0,HWHM):
    return intensity * HWHM**2 /((x-X0)**2 + HWHM**2)
```

といった形で書けます。上はローレンツ関数の値を返す関数です。

1.2.2 条件分岐

```
if 条件1:
    処理1
elif 条件2:
    処理2
else:
    処理3
```

という形です。

1.2.3 繰り返し

この処理が Python では遅いらしいです。あんま使わないで用意されている関数をうまく使用してください。

C 言語とはちょっと違います。C 言語とかはカウンタ変数 (大体 i, j とか使うやつ) をどこまでやるかみたいな感じでループをしますが、Python ではそういうものは使わないっぽいです。下に具体例を示します。

```
l = ['Alice', 'Bob', 'Charlie']
for name in l:
    print(name)
```

for の後に任意の適当な変数を入れループは in の後のリストなどのインテラブルオブジェクト？の要素が順番に変数が代入され処理が行われるっぽいです。

通常の C 言語みたいなループをしたい場合は range() 関数を使用します。

range(**最小値, 最大値, 数**) のように引数をとると最小値から最大値まで指定した数だけのリストを作ってくれます。普通に何回か同じ処理を繰り返したいだけだったら、

range(**回数**) という形で引数をとればその回数だけループを繰り返してくれます。回数は任意の変数に代入されます。

下に range() を使った例を示します。

```
for name in range(100):
    sum = sum + 1
```

ほかにもたくさんありますが、詳しくは書店にある Python の本を読んでください。とても丁寧に書いてあります。

1.3 データの処理

とりあえず今回は自分が実験に使用したプログラムについての解説的な感じにしたいと思います。すべて理解している人が書いているわけではないのでその辺察してくれるとありがたいです。

1.3.1 データ処理のプログラム全体図

とりあえず自分が実験データのフィッティングに使用したものを下にコピペします。ほとんど誰かのブログのパクリです。

```
import numpy as np
from matplotlib import pyplot
import matplotlib.ticker as ptick
import scipy.optimize
import glob, re
from functools import cmp_to_key

filename = 'file.dat'

with open(filename,'r') as file:
    X,Y =[],[]
    for line in file.readlines()[0:]:
        items = line[:-1].split(' ')
        X.append(float(items[0]))
        Y.append(float(items[1]))
pyplot.plot(X,Y,label = filename)

max_Y = max(Y)
min_Y = min(Y)

if np.abs(max_Y) >= np.abs(min_Y):
```

```
    intensity = max_Y
    X0 = X[Y.index(max_Y)]
else:
    intensity = min_Y
    X0 = X[Y.index(min_Y)]

pini = np.array([intensity,X0,1])

def func(x ,intensity ,X0,HWHM):
    return intensity * HWHM**2 /((x-X0)**2 + HWHM**2)

popt, pcov = scipy.optimize.curve_fit(func, X, Y, p0=pini)
perr = np.sqrt(np.diag(pcov))

print("initial parameter\toptimized parameter")
for i, v in enumerate(pini):
       print(str(v)+ '\t' + str(popt[i]) + ' ± ' + str(perr[i]))

fitline = func(X, popt[0], popt[1], popt[2])
pyplot.plot(X, fitline,label = 'FITTING')
pyplot.ylabel('intensity',fontsize = 15)
pyplot.xlabel('Shift[GHz]',fontsize = 15)
pyplot.tick_params(labelsize = 12)
pyplot.legend(loc="best")
pyplot.show()
```

このプログラムはデータファイルを読み込んで、ローレンチアンでフィッティングしようというものです。ローレンチアンって何かというと、式で書くと

$$f(x) = I_0 \frac{(\Gamma/2)^2}{(x - x_0)^2 + (\Gamma/2)^2}$$

この式をグラフ化すると

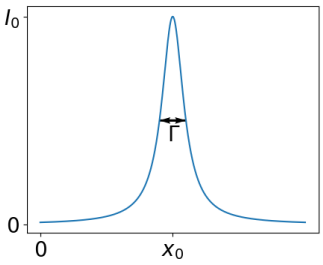

図 1.1　ローレンチアン

この形の関数は割とよくある形です。このフィッティングができたら大体良いです。上のプログラムの各部分をちょっと詳しく…

1.3.2 データプロット

はじめにデータ処理をするうえでデータを読み込ませないといけないです。上のプログラムでは初めに行っています。

```
filename = 'file.dat'

with open(filename,'r') as file:
    X,Y =[],[]
    for line in file.readlines()[0:]:
        items = line[:-1].split(' ')
        X.append(float(items[0]))
        Y.append(float(items[1]))
pyplot.plot(X,Y,label = filename)
```

この部分です。filename には今から処理したいデータのファイル名を入れてください。拡張子は *.txt,*.dat,*.plt はできました。他は知りませんが多分大丈夫です。

```
with open(filename,'r') as file:
```

この後にファイルを読み込む動作を書きます。通常だとファイルを開いて処理してファイルを閉じるといった操作をしなくてはならないですが、これはしなくてよさそうです。

```
X,Y =[],[]
```

では、これからファイルから読み込むデータの配列を宣言してます。

```
for line in file.readlines()[0:]:
```

で、のループで最後まで読み込んでます。file.readlines()[0:] で何行目から読んでどこまで読まないかを宣言できます。ここでは一行目からファイルの最後までって感じです。

```
item = line[:-1].split(' ')
```

ここでデータファイルのデータが何で区切られてるのかを宣言します。split(") の中に入れます。上ではスペースですと書いてあります。タブの場合 ('\t')、セミコロンの場合 (',') です。ほかの場合は調べてください。大体デバック開始したら vscode が教えてくれますが…

そのあとは配列に順序良く入れるという関数です。ここで、どんな変数なのかをちゃんと言わないと今後そのデータを処理できません。

そのままグラフにプロットさせたかったら、matplotlib のライブラリにある pyplot を使って

```
pyplot.plot(x 軸のデータ,y 軸のデータ)
```

でプロットできます。表示したかったら

```
pyplot.show()
```

と打ち込めばその時点で plot したデータのグラフを表示してくれます。

1.3.3 データのフィッティング

フィッティングする関数は下の部分です。

```
pini = np.array([intensity,X0,1])

def func(x ,intensity ,X0,HWHM):
    return intensity * HWHM**2 /((x-X0)**2 + HWHM**2)

popt, pcov = scipy.optimize.curve_fit(func, X, Y, p0=pini)
perr = np.sqrt(np.diag(pcov))
```

いざフィッティングする部分は

```
popt, pcov = scipy.optimize.curve_fit(func, X, Y, p0=pini)
```

の関数です。よくわかってなくそのまま使っているのでわからないのですが。引数に、フィッティングしたいグラフの関数、x 軸データ、y 軸データ、初期値を入れたらフィッティングしてくれます。データの読み込みとこのフィッティングまでの間の部分は適当な初期値を決めるところです。何となく読んで察してください。

popt に収束したパラメータ、perr に標準誤差が格納されます。

そのあとはグラフにプロットするためのものです。調べてください。

1.4 最後に

今回は自分がブログを見ながら生データにフィッティングするためのプログラムを作ったのでそれについての解説をしたいがために書いたものです。本当に参考程度で温かい目で見てほしいです。

参考文献

[1] https://qiita.com/yokot2/items/f8920f65b1037ec7009d