J. Plasma Fusion Res. Vol.84, No.2 (2008) 108-116

講座 オープンソースソフトウェアを使った実践データ解析

3. Pythonを使った実践データ解析

鈴木康浩
核融合科学研究所

（原稿受付：2008年1月24日）

3.1 はじめに

この章では，スクリプト言語Pythonを使ったデータ解析を紹介します．Pythonの実行に必要な環境はインターネットから無償で入手できます[1]．この章で紹介するスクリプトはすべてLinux（openSUSE10.3，Python2.5），Windows Vista（Python2.5）で動作確認をしています．

Pythonとは

Pythonは軽量言語（Lightweight Language）の一種で，スクリプト言語と呼ばれるプログラミング言語です．Pythonは，スクリプトと呼ばれるプログラムリストをインタプリタとして実行します[2]．スクリプト自体はプログラムが記述されたテキストファイルなので，Pythonの実行環境があれば，マシン環境に依存することなく実行可能です．Pythonはオブジェクト指向言語とも呼ばれ，多くのツール，ライブラリを利用することが可能です．したがって，適用範囲は広くWebベースのアプリケーションから本格的なGUIソフトまで様々です．そのため，他の言語（JavaやC++，PerlやRuby）とよく比較されます．そのような中，筆者がPythonを利用する理由として，以下のようなものがあります．

1．海外を中心にユーザーが多く，科学技術計算用のライブラリやモジュールが多く開発されています．
2．必要最小限の機能のみを提供しています．PerlのTIMTOWTDI（there's more than one way to do it）とは対照的なアプローチをとり，誰がスクリプトを書いても同じようなものになるよう言語設計がなされています[3]．逆をいえば，参考書に書いてある定石をしっかり勉強するだけでよいことになります．
3．プログラミング言語としてだけでなく，いくつかのプログラムをまとめるシェルとして利用することができます．このときPythonの特徴である，ライブラリやモジュールを使うことにより，GUI環境を持つ洗練されたラッパープログラムを開発することが可能です[4]．

Pythonは，巷で言われるようにオブジェクト化したPerlのようなものです．文法はC言語に大変よく似ています．もし，PerlやRubyなど他の言語を習得しているのであれば，簡単にマスターできるでしょう．初心者でも，参考書を1冊真剣に読み込めば，簡単なテキスト処理はすぐにできるようになるでしょう[1-3]．

本講座では，Pythonの詳しい文法解説は割愛します．Pythonプログラムの書き方は，参考文献に示した参考書を参照してください．

3.2 Pythonでデータ処理をしてみよう

では，どのようにPythonでデータ処理をするか，シンプルなデータビューワーを作ることを目標にして，見ていきましょう．リスト1は"Vf-i-dia@73116.dat"というファイルを1行ずつ読み込んで表示するだけの単純なプログラムです．

リスト1

```
1  # sample script 1
2
3  import re
4
5  data_id = "Vf-i-dia"
6  shot_id = "73116"
7  extension = ".dat"
8  data_path_r = "./"
9
10 rfile=data_path_r+data_id+"@"+shot\
   _id+extension
11 fr = open(rfile, "r")
12
13 for line in fr:
14         a = re.split(",", line)
15         print a[0], a[1]
16
```

1 http://www.python.org/
2 処理系としてはバイトコードに変換して処理するのでJavaに似ています．
3 他の人が書いたモジュールを見比べても，教科書どおりの書き方であることが多いです．
4 PythonをラッパーとしてFORTRANプログラムを呼び出す例は第7章で紹介します．

Practical Data Analysis Using Open Source Software 3. Practical Data Analysis Using Python
SUZUKI Yasuhiro
author's e-mail: ysuzuki@LHD.nifs.ac.jp

ファイル"Vf-i-dia@73116.dat"の中身はリスト2のとおりです.

リスト2

```
1  Time(ms),data(V)
2   0.110000E+02,-0.374625E-02
3   0.110020E+02,-0.374625E-02
4   0.110040E+02,-0.374625E-02
5   0.110060E+02,-0.374625E-02
6   0.110080E+02,-0.374625E-02
7   0.110100E+02,-0.374625E-02
8   0.110120E+02,-0.374625E-02
9   0.110140E+02,-0.374625E-02
10  0.110160E+02,-0.374625E-02
11  0.110180E+02,-0.374625E-02
12        ・
13        ・
14        ・
15      中略
```

紙面の都合で12行目以降は割愛していますが，実際には15001行あります．これはプローブ信号の時系列データで，1列目は時間（単位はms），2列目はプローブ電圧(単位はV)になります．このデータをグラフにしたものが**図1**です．ではリスト1を詳しく見ていきましょう．

1行目はコメント行です．#以降はコメントとして処理されます．

3行目は標準ライブラリ[5]の一つである正規表現モジュールreをインポートしています．Pythonにはサードパーティ製を含め様々な機能がライブラリ，モジュールとして提供されています．外部モジュールを利用する場合は(標準で用意されているものを含め)，プログラム先頭で必ず読み込み(インポート)ます．インポートしたいモジュールが複数ある場合は，以下の例のようにカンマで区切って記述します．

```
import re, math, pylab
```

この例では，標準ライブラリに含まれる正規表現モジュールre，数学関数モジュールmath，外部ライブラリであるpylabをインポートしています．

5行目から11行目まではファイル名を指定して，ファイルを開くまでを記述しています．5行目から8行目まではファイル名を文字列を使って定義します．**Pythonは変数の型(タイプ)と名前を事前に定義する必要がありません**[6]．変数に文字列を代入すれば自動的に文字列型に，数値を代入すれば数値型になります．ここでは，ファイル名が

```
"信号名"+@+"ショット番号"+"拡張子"
```

によって定義される場合を考えます．文字列は10行目のような文字列演算によって，複数の文字列変数を合わせて新しい文字列変数を作成できます．10行目は，ファイルが存在するディレクトリのパス（例題ではカレントディレクトリ）とファイル名を組み合わせて，開くべきファイルを指定します．ディレクトリのパスは絶対パス，相対パスの両方が使用できます．この演算により変数rfileには"./Vf-i-dia@73116.dat"が代入されています．

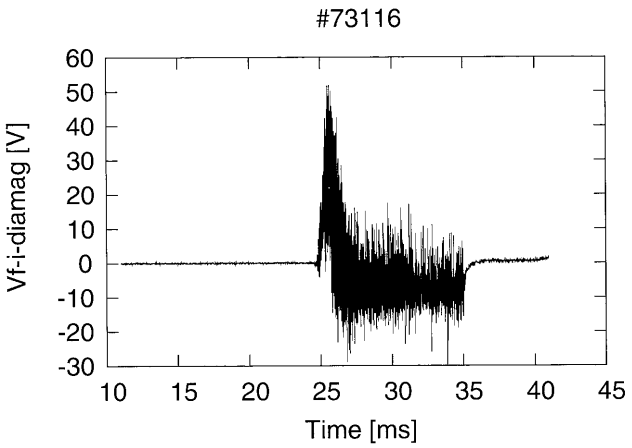

図1 Vf-i-dia@73116.datのグラフ．

11行目は関数openを用いてファイルオブジェクトfrを作成します．Pythonでは，読み込んだファイルも1つのオブジェクトとして扱います．読み込んだファイルに対する操作は，読み書きを含めすべてファイルオブジェクトに対する操作になります．

13行目以降は，読み込んだファイルを1行ずつ表示します．この処理は，繰り返しになるのでfor文を使うことになります．for文の文法は，ほぼC言語と同じです．決定的に違うのは，for文で繰り返し処理する部分を**インデントする**ことです．C言語の場合は文という概念があります．FORTRAN言語の場合DOとENDODOで挟まれたブロックが繰り返し処理の対象になります．Pythonの場合は同じインデントの部分が，ブロックとして認識されます．したがって，以下のような場合，

```
for line in fr:
    a = re.split(",",line)print a[0], a[1]
print[0], a[1]
```

print文以下は繰り返し処理の対象として認識されません．一般的に，Pythonではfor文やif文のように，まとまった処理を行うブロックは，インデントにより明示します．for文は以下のように，

```
for  繰り返し変数  in  シーケンス：
```

繰り返し変数とシーケンスで構成され，行末にコロンを付けfor文が終わったことを示します．シーケンスには，通常，リストを用います．リストはC言語やFORTRAN言語で配列と呼ばれます[7]．変数と同じく，あらかじめ型，サイ

5 Pythonの配布セットにあらかじめ含まれているライブラリです．
6 PerlやRubyを含め軽量言語に共通する仕様です．
7 Pythonでは，同じく配列のように働くタプルと辞書があるので注意が必要です．詳しくは参考書を見てください．

ズを定義する必要はありません．リストを作る場合は次のようにします．

```
list = (1,2,3,4,5,6)
```

自動的にサイズが6のリストに整数が格納されます．ところでC言語やFORTRAN言語の繰り返し処理の場合，繰り返し変数は数値でインクリメントなどを行うわけですが，Pythonの場合は繰り返し変数が数値とは限らず文字列なども扱えます．次のような文字列，

```
a = "abcde"
```

を考えます．文字列aをシーケンスにしたforループを考えてみましょう．

```
for i in a:
  print i
```

このループでは繰り返し変数iはどうなるでしょうか？このループの出力結果は次のようになります．

```
a
b
c
d
e
```

文字列そのものが繰り返し変数となります．このように，文字列やリストの要素をシーケンスとして扱える機能はデータ解析を行う場合に強力です．この例題で，読み込むファイル名を"Vf-i-dia@ショット番号.dat"としたのは理由があります．ショット番号が格納されたリストを

```
shot_id  =  ("73116","73117","73118","\
73119","73120")
```

のように考えます．するとリストshot_idをシーケンスとしたforループ

```
for id in shot_id:
      rfile = data_path_r + data_id + "@" +\
      id + extension
      fr = open(rfile, "r")
```

を作れば，リストshot_idの要素を次々と呼び出しながらファイルを開き，複数の実験データを処理することができます．一方，リスト1ではファイルオブジェクトfrを繰り返し処理のシーケンスとして利用しています．例題の処理では，ファイルオブジェクトfrが1行ずつ繰り返し変数lineに代入されます．したがって，

```
for line in fr:
      print line
```

と書けば，1行ずつ読み込んだファイルをそのまま表示するプログラムになります．では，C言語やFORTRAN言語のようにループ回数を指定したい場合はどうすればいいのでしょうか？ そのような場合は，以下のように記述します．

```
for i in range(10):
    print i
```

この場合，0から9までの整数が出力されます．関数rangeは，例のように引数10を与えると0から9まで1ずつインクリメントするリストを返します．

説明を続けましょう．14行目はモジュールre中の関数splitにより，区切り文字をカンマ","として行を要素ごとに分解し，リストaに代入します．モジュール中の関数を使用する場合は，

モジュール名.関数名(ある場合は引数)

の形で使用します．例題の14行目ではリストaが自動的に定義され，カンマで区切られていた文字列がそれぞれ1要素としてリストに格納されます．16行目では，リスト内のインデックス0と1の要素[8]をprint文で表示しています．

ファイルを開いて読み込むところまでは理解できましたか？では，読み込んだデータを実際に加工してみましょう．**図1**をみると，波形に随分リップルが乗っています．そこで，ある間隔で平均化されたデータを計算してみましょう．リスト3は，リスト1のスクリプトを改良して，ある間隔で平均化されたデータを計算するプログラムです．

リスト3

```
1  #!/usr/bin/env python
2
3  import re,   numpy
4
5  data_id = "Vf-i-dia"
6  shot_id = "73116"
7  extension = ".dat"
8  data_path_r = "./"
9
10 tstart = 20.0 # start time
```

8 リストのインデックスは0から始まり，括弧[ ]で指定します．

リスト3（続き）

```
tend  = 40.0 # end time
dtint = 1.0  # interval

rfile = data_path_r + data_id + "@" + shot_id + extension

n_data = len(open(rfile, "rU").readlines())

xx = numpy.zeros(n_data-1)
yy = numpy.zeros(n_data-1)
xx2 = numpy.zeros(n_data-1)
yy2 = numpy.zeros(n_data-1)

fr = open(rfile, "r")

a = fr.readline()

for line in range(n_data-1):
                a = fr.readline()
                b = re.split(",", a)
                xx[line] = float(b[0])
                yy[line] = float(b[1])

fr.close()

cnt = 0
mcnt = 0
ncnt = 0
Vf_ave = 0.0
tave_s = tstart - 0.5 * dtint
tave_e = tstart + 0.5 * dtint
while cnt <= n_data-2:
                if (xx[cnt] >= tstart and xx[cnt] <= tend) :
                    if (xx[cnt] >= tave_s and xx[cnt] < tave_e) :
                                Vf_ave = Vf_ave + yy[cnt]
                                ncnt += 1
                    elif (xx[cnt] >= tave_e and ncnt > 1):
                                Vf_ave = Vf_ave / ncnt
                                xx2[mcnt] = 0.5 * (tave_s + tave_e)
                                yy2[mcnt] = Vf_ave
                                mcnt = mcnt + 1
                                ncnt = 0
                                Vf_ave = 0.0
                                tave_s = tave_e
                                tave_e = tave_s + dtint
                cnt += 1

for line in range(mcnt):
                print line, xx2[line], yy2[line]
```

ここでは，20 msから40 msまで1 ms刻みで平均化したデータを計算することを考えます．ではリスト3を見ていきましょう．1行目に

```
#!/usr/bin/env python
```

と書かれています．スクリプト先頭の#!で始まる行は特別で，スクリプトファイルに実行属性を付ければ直接起動できます[9]．このあたりは通常のシェルスクリプトと同様です．Windows環境の場合，拡張子を".py"としたファイルを作成し，ダブルクリックすれば直接起動できます．ただし，Windows環境の場合はスクリプトが終了するとウィンドウがすぐに閉じてしまい，結果の確認には不便です．そのような場合，スクリプトの最後に`raw_input()`と命令を加えておけば，リターンキーが押されるまでウィンドウが開いたままになります．

3行目では標準ライブラリに含まれる正規表現モジュール`re`のほかにサードパーティー製の外部モジュールであるnumpy[10]をインポートしています．numpyは，FORTRAN言語のような多次元配列オブジェクトや，基本的な線形代数，高速フーリエ変換，特殊関数などの機能を提供するモジュールです[11]．Pythonのリストは C言語やFORTRAN言語の配列とは違います．次元やサイズをあらかじめ定義する必要はありません．ところが，実際のデータ解析の場面ではあらかじめサイズを指定した配列を用意してデータを格納し，その配列を用いて計算することが便利な場合が多くあります．numpyはFORTRAN言語のように配列を扱えるようにするものです．

numpyで配列を定義するには次のようにします．リスト3では，16行目でファイルの行数を数えています[12]．18行目から21行目でサイズが`n_data-1`[13]の配列`xx`, `yy`, `xx2`, `yy2`を定義します．`zeros`が引数のサイズで配列を定義し，0を代入して初期化する関数です．インポートしたモジュールnumpyに含まれる関数`zeros`を使うので，

```
xx = numpy.zeros(n_data-1)
```

のように記述します．

ところで，インポートしたモジュールに含まれる関数を"モジュール名．関数名"のようにいちいち記述するのは面倒な場合があります．たとえば標準ライブラリに含まれる数学関数モジュール`math`には円周率があらかじめ定義されています．次の例を見てください．

```
In [1]: import math

In [2]: math.pi
Out[2]: 3.1415926535897931
```

円周率を用いた計算をしたい場合は，`math.pi`とすればよいわけです．ただしこれを`math.cos(math.pi)`などと書いていると面倒になってくる場合があります．`pi`，あるいは`cos(pi)`と書きたい場合は，次のようにモジュールをインポートします．

```
from math import *
```

*はワイルドカードです．このようにモジュールをインポートすると，関数を利用するときにモジュール名を省略できます．ただし，`from`文を使ったモジュールのインポートは注意が必要です．たとえばモジュールnumpyと後で説明するモジュールscipyには同じ名前の関数`fft`が含まれています．関数名は多くの場合で抽象的な名前が付けられるので，異なるモジュールに同じ名前の関数が含まれることは想像に難くありません[14]．したがって，上の例のようにワイルドカード*を使ったモジュールのインポートは，名前空間が重ならないよう注意が必要です．本講座では対象としませんが，オブジェクト指向を陽にプログラミングする場合はモジュール内に階層構造をもった関数が現れる場合があるので，筆者の流儀として，面倒でも`from`文は使わないでインポートしています．ただし，リスト3程度のスクリプトでは問題ないでしょう．あるいは，以下の例

```
from math import pi, cos
```

のように必要な関数のみインポートすることお勧めします．繰り返しになりますが，関数名が重なり合わないよう，名前空間には注意してプログラミングしてください．

説明を続けます．27行目から31行目でファイル中の数値データを配列に格納します．データは配列に格納されているので，33行目でファイルを閉じます．今後の計算はすべてメモリ上で行われます．

35行目から55行目は，20 msから40 msまでの間で1 msずつ条件判定でデータを取り出しながら，平均化処理をしています．このブロックは`while`文と`if`文で入れ子状の階層構造になっています．同じブロックで処理したい命令は必ず，**同じインデントをとります**．`while`文と`if`文は特に説明はいらないないですね．それぞれ条件判定の命令文です．

最後に，ある間隔で平均化され，配列`xx2`，`yy2`に格納されていたデータを出力します．結果を**図1**と同様にプロットしたものが**図2**です．時間間隔で平均化した結果，信号の傾向が見やすくなりました．

9 Linuxなどの UNIX環境上用です．
10 http://numpy.scipy.org/
11 Numeric，numarrayと2つ存在した拡張をマージしたものです．
12 Pythonクックブックに載っている典型的手法です．
13 1行目がヘッダ行なので．
14 いくつもあります．orz...

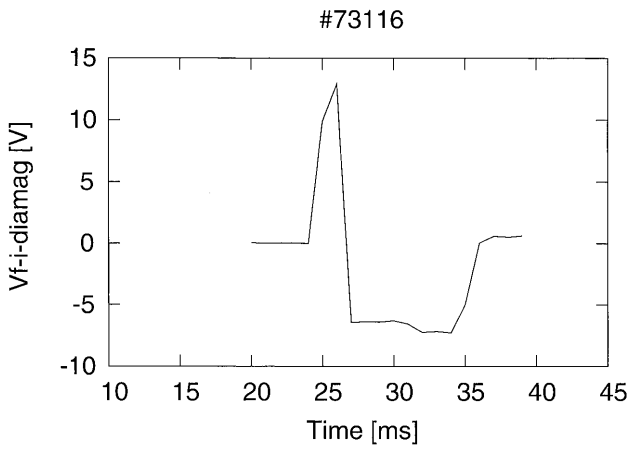

図2 Vf-i-dia@73116.dat を例題2のスクリプトに通して平均化した結果.

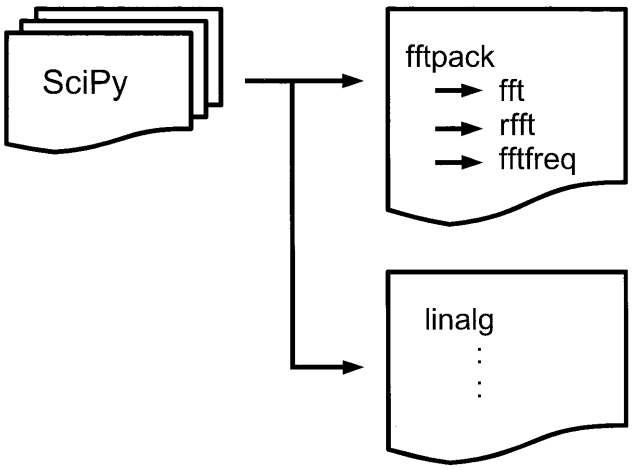

図3 モジュール SciPy の階層構造.

3.3 科学技術計算用ツールSciPyと周辺ツールの紹介

簡単なスクリプトを例に，Python を使ったデータ解析例を紹介しました．しかし，この程度の処理であれば，必ず Python で処理する理由にはなりません．筆者が Python をおすすめする理由の一つが，SciPy の存在です．SciPy[15] は，統計，最適化，積分，線形代数，フーリエ変換，信号・イメージ処理，遺伝的アルゴリズム，ODE（常微分方程式）solver，特別な関数，その他のモジュールを提供します[16]．線形代数やフーリエ変換に関しては，有名な FORTRAN 言語用数値計算ライブラリである BLAS[17]，LAPACK[18]や FFTW[19]のラッパープログラムとして働くので高速に動作します．SciPy は前節で紹介した NumPy を使用しますので，あらかじめサイズを指定した数値配列を利用することもできます．SciPy で提供される機能は，FORTRAN言語やC言語ではよく知られた枯れたライブラリで提供されているものと同等です．したがって，既存の言語環境でもこれらのライブラリを利用すれば，本質的には同等です．それでも Python をおすすめする理由は，スクリプト言語としての簡易性，充実した文字列処理，優れた外部ライブラリとの接続性になります．FORTRAN 言語や C 言語でもできますが，ファイル内のデータを加工して，計算して，さらに表示するということ考えると，簡便性と開発の容易さで Python が勝ると筆者は考えています[20]．

では，実際に SciPy を使ってみましょう．前節で用いたリスト3を改良して，データをフーリエ変換してみましょう．SciPy を使うにはモジュール `scipy` をインポートしするのですが，注意点が一つあります．前に述べたよう，SciPy は広範な機能を持つモジュールライブラリです．そこで，広範な機能を効率よく管理・利用できるようにモジュールは階層構造を持っています．SciPy の階層構造を**図3**に示します．1番上の階層として `scipy` があり，2番目の階層として各機能の高速フーリエ変換 `fftpack`，線形代数 `linalg`，補間 `interpolate`，特殊関数 `special` などがあります．ここでは高速フーリエ変換を用いたいので，`fftpack` をインポートします．`fftpack` のインポートは，次のように行います．

```
import scipy.fftpack
```

階層構造を持つモジュールで下階層のモジュールを利用したい場合には，

モジュール名．機能名

のようにドット"." でモジュール名と機能名を繋ぎます．モジュール `fftpack` の中で複素フーリエ変換を行う関数は `fft` です．この関数を用いたいときは，前に述べたルールに従って，

```
scipy.fftpack.fft(x)
```

と呼び出します．リスト3に追加する部分をリスト4に示します．

リスト4

```
1  xx3 = numpy.zeros (mcnt)
2  yy3 = numpy.zeros (mcnt)
3
4  yy3 = scipy.fftpack.fft(yy2[0:mcnt-1]) / xx2[0:mcnt-1].size
5
```

15 http://www.scipy.org
16 http://en.wikipedia.org/wiki/SciPy
17 http://www.netlib.org/blas/
18 http://www.netlib.org/lapack/
19 http://www.fftw.org/
20 Ruby を同様の理由で選択することも出来ます．どちらがよいかは趣味の問題でしょう．

```
6  dt = 1.0 / (tend - tstart)
7  for i in range(mcnt/2):
8          xx3[i] = dt * i
9          print xx3[i], abs(yy3[i])**2
```

リスト3の最後にリスト4を追加するだけで信号のフーリエ解析ができます．fftは複素フーリエ変換の関数で，フーリエ変換された信号の配列を返します．6行目で周波数の単位がkHzになるようにしています．関数fftが返すフーリエ振幅はNで規格化されているので，例題では$1/N$をかけています[21]．10行目では関数absを用いて複素共役の自乗を取ってパワースペクトルを求めています．

SciPyの中に含まれているわけではありませんが，SciPyと組み合わせて使えるすぐれたグラフィックライブラリがあります．その中で，本講座ではmatplotlib[22]を紹介します．matplotlibは，原著論文に載せるに足る品質の図を作ることができる2次元グラフィックライブラリです．また，Pythonのインタラクティブシェル IPython[23]と組み合わせることでMATLABと同等のことができるようになります[24]．ここでは，MATLAB互換としての環境はさておき，結果を手っ取り早くプロットするものとして説明します．

リスト4の結果をmatplotlibを用いてグラフにしてみましょう．matplotlibを使うには，モジュール pylab をインポートします．次にリスト5を追加してください．

リスト5

```
1  pylab.plot(xx3[0:mcnt/2],abs(yy3[0:mcnt/2])**2)
2  pylab.xlabel('Frequency [kHz]')
3  pylab.ylabel('Vf-i-dia [V$^2$/Hz]')
4  pylab.title('#73116')
5  pylab.show()
```

リスト5を追加したスクリプトを実行して得られた結果が**図4**です．たったこれだけで，結果を図にすることができます．matplotlibを使ってグラフを描く基本は関数plotで，以下のように使います．

```
plot(x,y)
```

xとyは，それぞれX軸とY軸が収められている1次元の配列です．xlabel,ylabel,titleは，それぞれX軸のラベル，Y軸のラベル，グラフのタイトルです．最後に関数showで画面を表示します．リスト5の3行目で

```
pylab.ylabel('Vf-i-dia [V$^2$/Hz]')
```

と書いていますが，**図4**を見ると^{2}の部分が上付き文字になっています．matplotlibでは，TeXと同じような文法で数式やギリシャ文字をグラフ中に書き込めます．この機能がまさに，論文品質の図が書ける所以なのです．matplotlibのプロジェクトページには様々形式のグラフのスクリーンショットとサンプルスクリプトがあるので参考にしてください[25]．**図4**の画面を見ると，図下部に様々なボタン類が配置されています．これらを使うことで，ズームや画像を保存することができます．関数savefigを使えば

```
savefig{'test.png'}
```

のようにスクリプト中で画像フォーマットにより保存することを命令できます．デフォルトの画像フォーマットはPNG (Portable Network Graphics)です．ほかに，PS,PDF,SVGなどを選べます．SVG (Scalable Vector Graphics) 形式で保存すれば，あとでInkscape[26,27]のようなベクタードローソフトで，さらに画像（グラフ）を編集できるので非

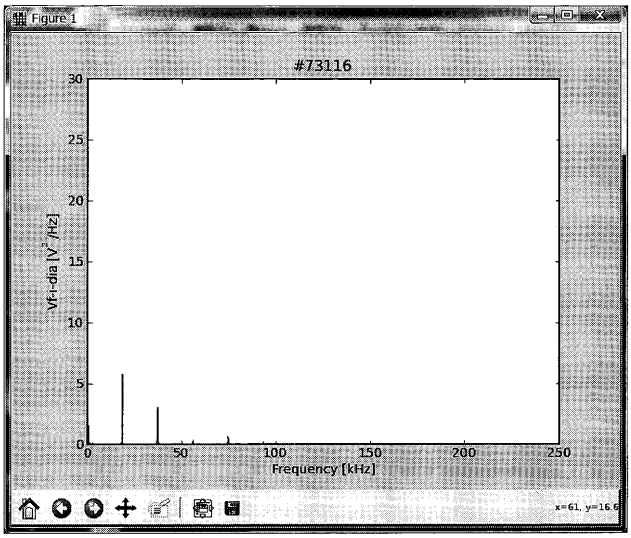

図4　リスト3＋リスト4＋リスト5を実行した結果．

21　ライブラリFFTWの仕様です．
22　http://matplotlib.sourceforge.net/
23　http://ipython.scipy.org/
24　実際，ユーザーインタフェースと命令文が似ています．
25　http://matplotlib.sourceforge.net/screenshots.html
26　http://www.inkscape.org/
27　Adobe IllustratorでもSVGを扱えます．

常に便利です．

matplotlib の関数を使えば次のようなこともできます．リスト 6 を例題 2 に追加してください．

リスト 6

```
1  pylab.subplot(211)
2  pylab.plot(xx,yy)
3  pylab.ylabel('Vf-i-dia [V]')
4  pylab.grid(True)
5  pylab.title('#73116')
6  xmin, xmax = pylab.xlim()
7  ymin, ymax = pylab.ylim()
8
9  pylab.subplot(212)
10 pylab.plot(xx2[0:mcnt-1],yy2[0:mcnt-1])
11 pylab.xlabel('Time [ms]')
12 pylab.ylabel('Vf-i-dia [V]')
13 pylab.grid(True)
14 pylab.title('averaged')
15 pylab.xlim(xmin,xmax)
16 pylab.ylim(ymin,ymax)
17
18 pylab.show()
```

リスト 6 を追加したリスト 3 を実行した結果が**図 5** です．元の時系列データとリスト 3 の結果により 20 ms から 45 ms まで 1 ms 刻みで平均化されたデータを重ねてプロットしてあります．リスト 6 で新しく出てきた関数のうち，関数 subplot が図を重ねて表示するためのものです．関数 subplot は，

```
subplot (numrows, numcols, axesnum)
```

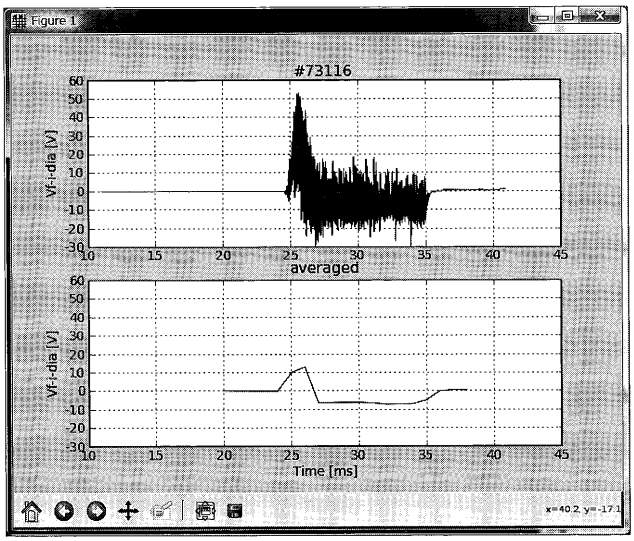

図 5　リスト 3＋リスト 6 を実行した結果．

のように使います．リスト 6 の例では，関数 subplot の引数に 1 行目では211，9 行目では212を与えています．それぞれ，2 行 1 列のグラフの 1 つ目と 2 つ目を指定します．関数 xlim,ylim はそれぞれ，X 軸と Y 軸の範囲を

```
xlim( xmin, xmax )
```

のように指定します．

最後に，スクリプトにパラメータをコマンドライン引数から与えられるようにして終わりにしましょう．コマンドラインからスクリプトに引数を渡すには，モジュール sys の関数 argv を使います．リスト 7 に，関数 argv の使い方の 1 例を示します．

リスト 7

```
1  #!/usr/bin/env python
2
3  import re, numpy, scipy.fftpack, pylab, sys
4
5  data_id = "Vf-i-dia"
6  shot_id = sys.argv[1]
7  extension = ".dat"
8  data_path_r = "./"
9
10
```

これまでのスクリプトはすべて，ショット番号やファイル名などスクリプト中に埋め込んでいました．Python のようなスクリプト言語は，実行するためにいちいちコンパイル作業はいらないために，短い作業であればスクリプトそのものを編集してしまっても手間にはなりません．ただし，大量のデータをまとめて作業するのであれば，コマンドラインからショット番号やファイル名を渡せるようにしたほうが便利です．6 行目で

```
shot_id = sys.argv[1]
```

として，shot_id に sys.argv[1] を代入しています．関数 argv はコマンドラインからの引数のリストを返します[28]．

```
./example.py 73116
```

のように実行すれば shot_id にショット番号73116が代入

28　インデックス 0 はスクリプト自体です．

されます．これを応用すれば，リスト3のtstart,tendの値も引数から与えられます．簡単に改造できるので，試してみてください．

3.4 おわりに

この章では，スクリプト言語Pythonを使って，簡単なデータビューワーを作成しながら使い方の基本を説明しました．

はじめに述べたように，Pythonはオブジェクト指向言語です．オブジェクト指向を陽に書くのであれば，汎用的な機能はクラスを定義すべきです．しかし，本講座はとにかく，データ解析にまつわる作業を簡単にすることを目的としたので，オブジェクト指向どころか，自前の関数を定義することはしていません．本章では浮動電位のデータを例にスクリプトを作成しましたが，1種類の物理データに対して作業するだけならば十分だからです．ただし，1つのスクリプトで複数の物理データを処理する汎用的なプログラムをめざすならば，必要な機能を関数にまとめ，クラスを定義したほうが作業効率は上がるでしょう．オブジェクト指向の考え方も，参考文献で紹介した参考書を1冊読みこめばわかるでしょう．本講座終了後に開設予定のサポートページでは，本章で紹介したスクリプト全文と，オブジェクト指向を用いた汎用プログラムも紹介する予定です．

謝　辞

本章では，東北大学ヘリアック装置の実験データ解析を行うためのスクリプトプログラムを流用する形で執筆しました．快くデータを提供していただいた，東北大学工学研究科の北島純男先生と東北大学ヘリアックグループの皆様に感謝いたします．また，九州大学応用力学研究所稲垣滋氏との議論が，大変参考になりました．ここに感謝いたします．

参考文献

[1] 柴田　淳：みんなのPython（ソフトバンククリエイティブ，2006）．
[2] M. Lutz and D. Ascher：はじめてのPython第2版（オライリー・ジャパン，2004）．
[3] A. Martelli, A.M. Ravenscroft and D. Ascher:Pythonクックブック　第2版（オライリー・ジャパン，2007）．