「データ・誤差解析の基礎」の Python コード再訪

Revisiting the Python Codes for “A Student’s Guide to Data and Error Analysis”

林 茂雄

2023 年 3 月

1. はじめに

H.J.C. Berendsen 著，林・馬場共訳「データ・誤差解析の基礎」(東京化学同人, 2013) が刊行されて 10 年になります．この本の第 III 部には Python プログラムが示されていますが，Python 自体のバージョンが上がっているので今でもプログラムが走るか気がかりです．そこで代表的なプログラムをいくつか選んで動作試験を行ってみました．その結果，多かれ少なかれ修正を必要とすることが分かりました．以下，個々のプログラムについての試験結果と代替プログラム案を記載しますので参考にしてください．ただし，私はこれをきっかけに勉強し始めた初心者ですので，よいプログラムではない可能性が高いことを予めお断りしておきます．

2023 年 3 月　　林　茂雄

2. システム構成

(1) Dr. Berendsen が用いたシステム

python 2.6

使用モジュール＝ numpy + scipy + plotsvg + physcon

(2) 今回用いたシステム

python 3.9

使用モジュール＝ numpy + scipy + matplotlib

動作環境は Visual Studio Code /Windows 11

Python のバージョンが 2 から 3 に変わっているので，プログラムがすんなりと走らないことは十分予想されます.

3. 検証結果の概略

(1) 著者のウェブサイト(テキスト p.175) http://hjcb.nl/python/ は存在しない

ブラウザーが URL を https://hjcb.nl/python/ に変更してくれるのでサーバー自体にはアクセスできますが Error 404 が表示されます．Home (https://hjcb.nl/) には Python の案内が出ているのでかつて Python サイトが存在していたことは確かなようです.

(2) plotsvg と physcon は認識されない

これらのモジュールは原著者のウェブサイトからダウンロードできるとのことですが(p.175)今となってはできません.

また，　python -m pip install plotsvg -user

によるインストールも失敗しました．よって代替プログラムが必要となります.

(3) モジュール構成は変わりうる

scipy から numpy に配置換えになった変数や関数があります．また autoplotp (p.194)は認識されません.

（注）以後ページ番号は特に断らない限り「データ・誤差解析の基礎」のページ番号を指します.

4. プログラムの検証結果

4.1 Python コード 0.1（plotsvg のデモ：累積分布図の作成）

【検証結果】

走らない．plotsvg が確認できないため．

【対応策】

標準的ライブラリー(numpy, scipy, matplotlib) を用いて作り直す．

§4.3 でも累積分布図を扱うのでここでまとめて説明しよう．まずフローチャート的な発想で考えてみると次のような 3 ステップに分割できることが分かる．

(i) ステップ 1（データを準備する）

n 個のデータ r_i (i=0,..,n-1)を用意する．大別すると次の二つになる．

(a) 実行時に正規乱数を発生させる（コード 0.1）．その都度異なる乱数の組が生成される．

(b) 実際の測定データの組を入力する．コンソール画面で一つずつ入力するのは（特に数が多くなれば）面倒くさい．それよりは一旦ファイルに書き込んでおき，実行時にその都度読み出すほうが現実的（§4.3）．

(ii) ステップ 2（度数分布のデータを作る）

(a) データを小さい順に並べて x_i (i=0, … , n-1)の数列を得る．ただし，(i)(a)の時点で既に順序化されていることもある．

(b) 横軸を x_i, 縦軸を順番 k_i =i+1 とする累積分布データの組 (x_i, k_i)を作る．

$k_i = 1, \dots, n$ であるが，テキスト図 2.1 では度数の相対値 $k_i = \frac{1}{n}, \frac{2}{n}, \dots, 1$ としている．

(c) 折れ線グラフの節点 (u_j, v_j) $(j = 0, \dots, m-1)$ を作る(テキスト 図 2.1)．ここ

まで済めば狭義の累積分布図は容易に作図できる．

(iii) ステップ 3（確率紙上で確率分布のグラフを描く）

確率紙とは累積分布を表現するのに適したグラフ用紙であるが（§6 参照），目盛りが等間隔でないのでコンピュータ上で作図しようというわけである．

(a) i 番目のデータ x_i に確率 y_i を割り振る．この確率は上の(c)で述べた度数の相対値と同じではない．Wikipedia の”Normal probability plot”によれば

$$y_i = \frac{i+0.5}{n} \quad (i = 0,1,2,\dots,n-1) \qquad (1)$$

である．あるいは $y_i = \frac{i+1}{n+1}$ がよいという説もある．いずれにせよ確率 y は $0 \le y \le 1$ の範囲にある．

(b) この確率 y を次式で z に変換する．

$$z = f(y) = \frac{1}{2}\left[1 - \frac{1-2\varepsilon}{\mathrm{erf}^{-1}(1-2\varepsilon)}\mathrm{erf}^{-1}(1-2y)\right] \qquad (2)$$

ここで $\mathrm{erf}^{-1}(\cdot)$ は誤差関数 $\mathrm{erf}(\cdot)$ の逆関数であり，$\varepsilon = 0.01$ (1%)である．

(c) 同様にして目盛り $\boldsymbol{y}$=(0.02, 0.05, 0.1, ⋯) に対応する $\boldsymbol{z}$ 値を求める．

(d) 折れ線グラフの節点 (u_j, v_j) $(j = 0,\dots,m-1)$ を作る．

(e) (x_i, z_i) の散布図あるいは (u_j, v_j) の折れ線グラフを描く．テキストは後者だがネット上では前者が多いように見受けられる．

(f) 縦軸に目盛りを入れる(テキスト 図 2.2).

以上でひとまず作業は終わるが，プログラムのバージョンアップを考えると，まとまった作業をモジュールとしてまとめる便利である．今回該当する作業としては

class zigzag(x, y, ndata, eps)	折れ線グラフの節点を作る: (ii)(c),(iii)(d)
class tic_label(eps)	確率紙上で目盛り位置を決める: (iii)(c)

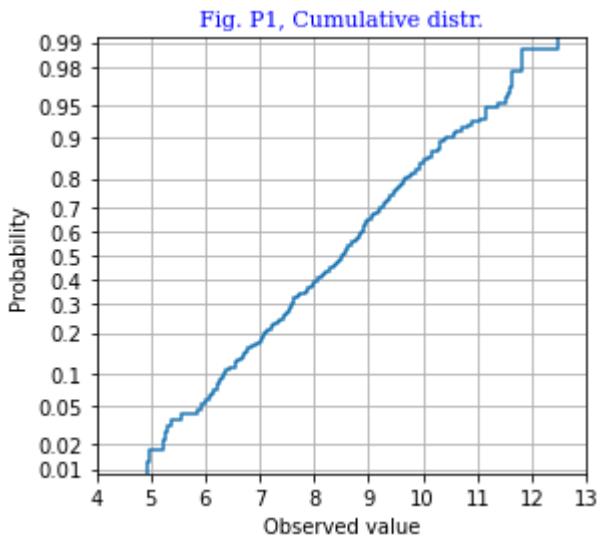
がある．それらを hwblines2.py という名前のモジュールに移す（§5 参照）．

【プログラム例と画像出力】

入力は 200 個のランダムデータである．# File name は，実際に走らせたファイル名(内部データ)である．下線部は上記オブジェクトの使用箇所を示す.

```
# Python Code 0.1
# File name = code2.2ccc.py
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import numpy.random as rnd
import hwblines2 as hl
import statpos as sp

ndata = 200
eps = 0.01
yax = hl.tic label(eps)

x = rnd.normal(8.5, 1.5, ndata)
y = np.arange(ndata, dtype="f")
z = np.arange(ndata, dtype="f")
print()
print('ndata=',ndata)
for i in range(ndata):
    y[i] = (i + 0.5)/ndata
    z[i] = yax.finvcumulat(y[i])
    print(i+1,")  ",f"{y[i]:,.3f}","  ",f"{z[i]:,.3f}")

med = np.median(x); print('median = ',f"{med:,.3f}")
aver = np.average(x); print('mean = ',f"{aver:,.3f}")
sigma = np.std(x); print('sigma = ',f"{sigma:,.3f}")
```

```
ax = plt.axes()
ax.set_xlim(4.0, 13.0)
ax.set_ylim(0.0, 1.0)
ax.set_aspect(8./1.)

xtic = np.arange(4, 14.0, 1.0)
ax.set_xticks(xtic)
ax.set_xlabel('Observed value')

ax.set_yticks(yax.tics)
ax.set_yticklabels(yax.labels)
ax.set_ylabel('Probability')

f3 = hl.zigzag(x, z, ndata, eps)
u = f3.ua
v = f3.va
plt.plot(u, v)
# plt.plot(x,z,lw=0,ls='',marker='o',ms=3)      #dots
ax.grid(True)
font1 = {'family':'serif','color':'blue','size':11}
font2 = {'family':'serif','color':'darkred','size':12}
plt.title("Fig. P1, Cumulative distr.", fontdict = font1)
#plt.xlabel("Observed values", fontdict = font2)
plt.show()
```



4.2　Python コード 0.2（physcon のデモ：物理定数）

【検証結果】

走らない．plyscon が確認できないため．

【対応策】

標準的ライブラリーである numpy と scipy を用いて書き換える．

【プログラム例と出力】

```
# Python Code 0.2
# File name = code0.2.py
import numpy
import numpy as np
import scipy.constants as cst

print('--- Mathematical and physical constants ---')
print()

print('--- Mathematical constants ---')
print('e =',np.e)
print('pi = ',cst.pi)
print()
print('--- Physical constants ---')
print('c = ', cst.c)
print('e = ', cst.e)
print('h = ', cst.h)
print('R = ', cst.N_A)
print('\n--- 2018 CODATA Database ---')
s = 'Boltzmann constant'
print(s + ' = ', cst.physical_constants[s][0],' ',\
    cst.physical_constants[s][1],' ',\
    cst.physical_constants[s][2])
s = 'vacuum electric permittivity'
```

```
print(s + ' = ', cst.physical_constants[s][0],' ',\
    cst.physical_constants[s][1],' ',\
    cst.physical_constants[s][2])
```

```
--- Mathematical constants ---
e = 2.718281828459045
pi =  3.141592653589793

--- Physical constants ---
c =  299792458.0
e =  1.602176634e-19
h =  6.62607015e-34
R =  6.02214076e+23

--- 2018 CODATA Database ---
Boltzmann constant =  1.380649e-23   J K^-1   0.0
vacuum electric permittivity =  8.8541878128e-12
                                F m^-1   1.3e-21
```

4.3　Python コード 2.1（累積分布図）

【検証結果】

走らない．plotsvg が確認できないため．

【対応策】

標準的ライブラリー(numpy, scipy, matplotlib)を用いて作り直す．

§4.1 のコード 0.1 との違いは，30 個のデータをファイルから読み込むことにある．ここでは自作の fileRead を使って tab2.1x.csv というテキストファイルを読み込む．このオブジェクトは hwbfiles というモジュールに作ってある（§5 参照）．またモジュール statpos にある posn というオブジェクトで以て平均，メジアン，モード，$\pm\sigma$，$\pm 2\sigma$ に対応する(x, y)の値を決める．

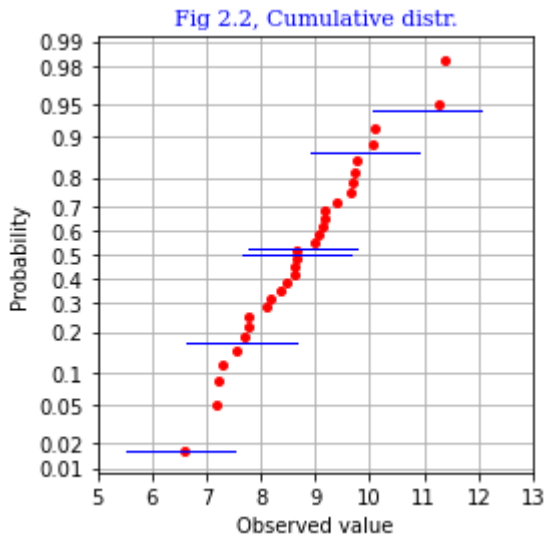
【プログラム例と画像出力】

散布図で表す．下線部は上記オブジェクトの使用箇所である．

```
# Python Code 2.1
# File name = code2.2bbb.py
import matplotlib
import matplotlib.pyplot as plt
import scipy.stats as sp
import numpy as np
import hwbfiles as hf
import hwblines2 as hl
import statpos as ps

nline = 6
eps = 0.01
lines = np.arange(nlinedtype='f')
posx  = np.arange(nline,dtype='f')
posy  = np.arange(nline,dtype='f')
```

```
tocheck = True

f = hf.fileRead('tab2.1x.csv', False)
ndata = f.ndata
x = f.xa
y = f.ya
z = f.za

print(ndata, ' points in ' + f.fname)
print(y)
med = np.median(x)
aver = np.average(x)
sigma = np.std(x)
print("median=", f"{med:,.3f}")
print("mean=", f"{aver:,.3f}")
print("sigma=", f"{sigma:,.3f}")

yax = hl.tic_label(eps)
yax.info()

z = np.arange(ndata, dtype="f")
for i in range(ndata):
    y[i] = (y[i] - 0.5)/ndata  # See Wikipedia, "Normal prob. plot"
    z[i] = yax.finvcumulat(y[i])
    print(i+1,") ",f"{x[i]:,.3f}"," ",f"{y[i]:,.3f}"," ,
                                        f"{z[i]:,.3f}")

ax = plt.axes()
ax.set_xlim(5.0, 13.0)
ax.set_ylim(0.0, 1.0)
ax.set_aspect(8./1.)

xtic = np.arange(5, 14.0, 1.0)
ax.set_xticks(xtic)
ax.set_xlabel('Observed value')
```

```
ax.set_yticks(yax.tics)
ax.set_yticklabels(yax.labels)
ax.set_ylabel('Probability')
f3 = hl.zigzag(x, z, ndata, eps)

u = f3.ua
v = f3.va
plt.plot(x,z,lw=0,ls='',marker='o',ms=3,mec='r',mew=2,mfc='r')

fp = ps.posn(lines,nline,x,y,ndata,aver,sigma,med)
posx = fp.posx
posy = fp.posy
for i in range(nline):
    print(i,'-th point =',f'{posx[i]:,.2f}, {posy[i]:,.2f}')
    xs = np.array([posx[i]-1, posx[i]+1])
    tmp = yax.finvcumulat(posy[i])
    ys = np.array([tmp, tmp])
    plt.plot(xs, ys, color='blue',lw=1)

ax.grid(True)
font1 = {'family':'serif','color':'blue','size':11}
font2 = {'family':'serif','color':'darkred','size':12}
plt.title("Fig 2.2, Cumulative distr.", fontdict = font1)plt.show()
```



4.4 Python コード 2.2（ヒストグラム）

【検証結果】

走らない．pllotsvg が認識できないため．

【対応策】

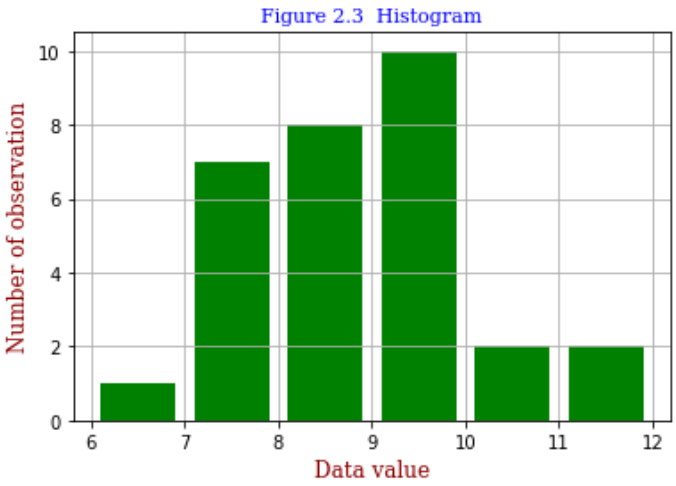
標準的ライブラリーである numpy と matplotlib を用いて書き換える．

【プログラム例と画像出力】

fileRead を用いて 'tab2.1.csv' からデータを読み込む．

```
# histogram, Code 2.2
# File name = code2.2h.py
import matplotlib
import matplotlib.pyplot as plt
import numpy
import numpy as np
import hwbfiles as hf

fname = 'tab2.1.csv'
f = hf.fileRead(fname, False)
ndata = f.ndata
nitem = f.nitem
x = f.xa
print(ndata, ' lines and ', nitem, ' items in ' + fname)

bins = np.array([6.,7.,8.,9.,10.,11.,12.])
plt.grid()
plt.hist(x, bins, histtype='bar', rwidth=0.8, color='green')
font1 = {'family':'serif','color':'blue','size':11}
plt.title("Figure 2.2  Histogram", fontdict = font1)
font2 = {'family':'serif','color':'darkred','size':12}
```

```
plt.title("Figure 2.3  Histogram", fontdict = font1)
plt.xlabel("Data value", fontdict = font2)
plt.ylabel("Number of observation", fontdict = font2)
plt.show()
```



4.5　Python コード 2.5（対数グラフとエラーバー）

【検証結果】

走らない．plotsvg が確認できないため．

【対応策】

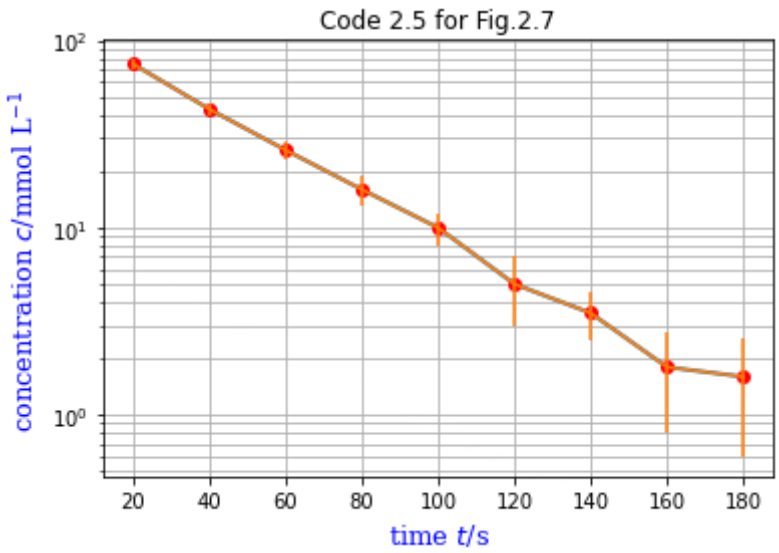
標準的ライブラリーである numpy と matplotlib を用いて書き換える．

【プログラム例と画像出力】

'tab2.5.csv' からデータを読み込む．

```
# Code 2.5 for Fig.2.7
# File name = code2.5file.py
import matplotlib.pyplot as plt
import numpy as np
import hwbfiles as hf

f    = hf.fileRead('tab2.5.csv', False)
time = f.xa
conc = f.ya
err  = f.za

fig, ax = plt.subplots()
ax.plot(time,conc,lw=2,ls='-,marker='o',ms=5,mec='r',mew=2,mfc='r')
ax.set_yscale('log')

pos = [1, 10, 100]
plt.errorbar(time, conc, err)
plt.grid(axis='y', which='both')
plt.grid(axis='x', which='major')

font1 = {'family':'serif','color':'blue','size':13}
```

```
plt.xlabel('time $t$/s',fontdict = font1 )
plt.ylabel('concentration $c$/mmol L$^{-1}$',fontdict = font1)
plt.title('Code 2.5 for Fig.2.7')
plt.show()
```



4.6 Python コード 6.1（最小自乗法によるフィッティング）

【検証結果】

走らない．いくつかの関数と定数が numpy に配置換えされたため．

【対応策】

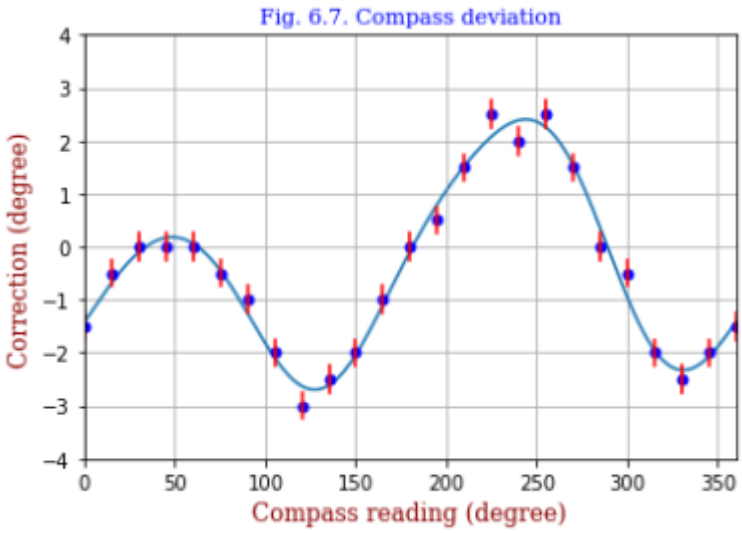
arrange, array, pi が numpy に属することを明示する．

【プログラム例と画像出力】

(1)原著とおり，optimize.leastsq で最適化を行う．小修正のみで走る．

```
# Python code 6.1
# File name = code6.1.py
from scipy import optimize
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(0., 361., 15.)
y = np.array([-1.5, -0.5, 0., 0., 0., -0.5, -1., -2., \
    -3., -2.5, -2., -1., 0., 0.5, 1.5, 2.5, 2.0, 2.5, \
    1.5, 0., -0.5, -2., -2.5, -2., -1.5])
t = np.arange(0.,361.,1.)

def fitfun(x,p):
    phi = x*np.pi/180.
    s = p[0]
    for i in range(1,5,1):
        s += p[2*i-1]*np.sin(i*phi) + p[2*i]*np.cos(i*phi)
    return s

def residual(p): return y - fitfun(x, p)
pin = [0.]*9
```

```
output = optimize.leastsq(residual, pin)
print(output)
pout = output[0]
def fun(x): return fitfun(x, pout)

# Plotting the results
plt.xlim(0,360)
plt.ylim(-4,4)
plt.plot(t, fun(t))

for i in range(25):
    plt.plot(x[i],y[i],marker="o",markersize=5,color="blue")
    xs = np.array([x[i], x[i]])
    ys = np.array([y[i]-0.25, y[i]+0.25])
    plt.plot(xs,ys,color="red")

font1 = {'family':'serif','color':'blue','size':11}
font2 = {'family':'serif','color':'darkred','size':12}
plt.title("Fig. 6.7. Compass deviation", fontdict = font1)
plt.xlabel("Compass reading (degree)", fontdict = font2)
plt.ylabel("Correction (degree)", fontdict = font2)
plt.grid()
plt.show()
```



(2)(別法) optimize.curve_fit を用いる.

```
# Python code 6.1
# File name = code6.1bb.py
from scipy import optimize
import matplotlib.pyplot as plt
import numpy as np
nfree = 9
x = np.arange(0., 361., 15.)
y = np.array([-1.5, -0.5, 0., 0., 0., -0.5, -1., -2., \
    -3., -2.5, -2., -1., 0., 0.5, 1.5, 2.5, 2.0, 2.5, \
    1.5, 0., -0.5, -2., -2.5, -2., -1.5])  # 25 elems
t = np.arange(0.,361.,1.)

def fitfun(x, *q):
    phi = x*np.pi/180.
    i = 0
    j = 0
    s = q[0]
    while True:
        if i>=nfree-1:
            break
        i += 1
        w = q[i]
        j = (i+1) // 2
        if i%2 != 0:
            s += w*np.sin(j*phi)
        else:
            s += w*np.cos(j*phi)
    return s

pin = [0.]*nfree
output = optimize.curve_fit(fitfun, x, y, pin)
#print('----------------------------')
pout = output[0]
```

```
print(pout)
def fun(x):
    return fitfun(x, *pout)

# Plotting the results
plt.xlim(0,360)
plt.ylim(-4,4)
plt.plot(t, fun(t))

for i in range(25):
    plt.plot(x[i],y[i],marker="o",markersize=5,color="blue")
    xs = np.array([x[i], x[i]])
    ys = np.array([y[i]-0.25, y[i]+0.25])
    plt.plot(xs,ys,color="red")

font1 = {'family':'serif','color':'blue','size':11}
font2 = {'family':'serif','color':'darkred','size':12}
plt.title("Fig. 6.7. Compass deviation", fontdict = font1)
plt.xlabel("Compass reading (degree)", fontdict = font2)
plt.ylabel("Correction (degree)", fontdict = font2)
plt.grid()
plt.show()
```

図は p.18 のものと同じ.

4.7　Python コード 7.3（Δ χ^2 の等高線図）

【検証結果】

走らない．autoplotp が確認できないため．

【対応策】

matplotlib.pyplot で作画する．

【プログラム例と画像出力】

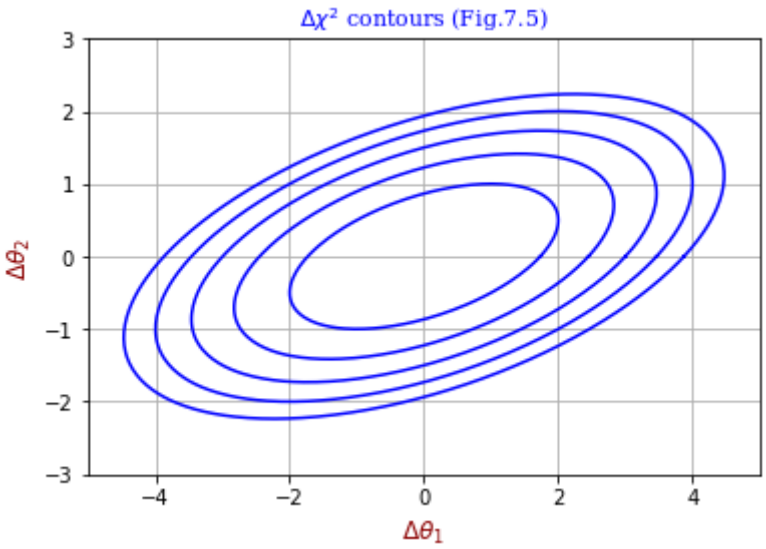
著者 Dr. Berendsen オリジナルの contour 関数を活かして作図する（ただし $f_{xy}(x,y)=\mathbf{v}\,\mathbf{B}\,\mathbf{v}^T$ は展開しない）．また contour 関数の 返しを xlist，ylist の二つにする．

(1)塗りつぶしなし

```
# Python code 7.3 (p.184) for Fig.7.5 (p.103)
# File name = code7.3.py
from scipy import *
from scipy import optimize
import numpy as np
import matplotlib.pyplot as plt

matrx = np.array([[1.,-1.],[-1.,4.]])/3.0

def contour(fxy,z,xycenter,xyscale=[1.,1.],radius=0.05,nmax=500):
    x0 = xycenter[0]; y0 = xycenter[1]
    xscale = xyscale[0]; yscale = xyscale[1]

    def funx(x):
        return fxy(x,y0) - z

    def funphi(phi):
```

```
        sinphi = np.sin(phi); cosphi = np.cos(phi)
        x = xa + (dxs*cosphi + dys*sinphi)*xscale
        y = ya + (-dxs*sinphi + dys*cosphi)*yscale
        return fxy(x,y) - z

    xx = optimize.brentq(funx, x0, x0 + 5.0*xscale)
    xlist = [xx]; ylist = [y0]; #pt = np.array([xx, y0])
    dxs = radius; dys = 0
    xa = xx; ya = y0
    phi = optimize.brentq(funphi, -np.pi, 0.)
    sinphi = np.sin(phi); cosphi = np.cos(phi)
    xb = xa + (dxs*cosphi + dys*sinphi)*xscale
    yb = ya + (-dxs*sinphi + dys*cosphi)*yscale
    xlist += [xb]; ylist += [yb];
    radsq= radius*radius
    dsq = 4.*radsq
    n = 0
    while  (dsq>radsq) and (n<nmax):
        n += 1
        dxs = (xb - xa)/xscale
        dys = (yb - ya)/yscale
        xa = xb; ya = yb
        phi = optimize.brentq(funphi, -0.5*np.pi, 0.5*np.pi)
        sinphi = np.sin(phi); cosphi = np.cos(phi)
        xb = xa + (dxs*cosphi + dys*sinphi)*xscale
        yb = ya + (-dxs*sinphi + dys*cosphi)*yscale
        xlist += [xb]; ylist += [yb];
        dsq = ((xb-xx)/xscale)**2 + ((yb-y0)/yscale)**2
    xlist += [xx]; ylist += [y0]; # data = array([xlist,ylist])
    return xlist,ylist #,data

def fxy(x,y):
    v = np.array([[x,y]])
    w = np.dot(v, matrx @ v.T)
    return w
```

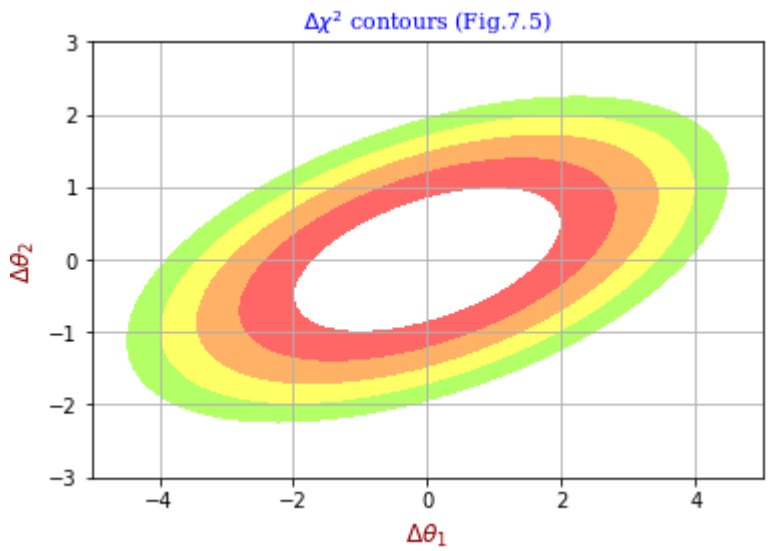
```
# figure,axis =plt.subplots(1,1);axis.set_aspect('equal')
plt.xlim(-5., 5.)
plt.ylim(-3., 3.)
xycenter= np.array([0.0, 0.0])


for i in range(1,6):
    z = i
    xlist,ylist = contour(fxy,z,xycenter)
    plt.plot(xlist,ylist,color='blue')

font1 = {'family':'serif','color':'blue','size':11}
font2 = {'family':'serif','color':'darkred','size':12}
plt.title(r"$\Delta \chi^2$ contours (Fig.7.5)", fontdict = font1)
plt.xlabel(r"$\Delta \theta_1$", fontdict = font2)
plt.ylabel(r"$\Delta \theta_2$", fontdict = font2)
# See DelftStack:https://www.delftstack.com/howto/matplotlib/greek-
letters-matplotlib-in-python/
plt.grid()
plt.show()
```



(2)塗りつぶしあり

```
# Code 7.3 (p.184)
# File name = code7.3a.py
from scipy import *
from scipy import optimize
import numpy as np
import matplotlib.pyplot as plt

matrx = np.array([[1.0, -1.0],[-1.0, 4.0]])/3.0
nxpt = 201 # 3
nypt = 121 # 2
tosee = False

x = np.linspace(-5., 5.,nxpt)
y = np.linspace(-3., 3.,nypt)
z = np.zeros((nypt,nxpt))
xx, yy = np.meshgrid(x,y)
if tosee:
    print('linspace',x.shape)
    print('meshgrid xx.shape and yy.shape',xx.shape,yy.shape)
    print(xx)
    print(yy)

def findz(x,y):
    for i in range(nypt):
        yi = y[i,0]
        for j in range(nxpt):
            xj = x[0,j]
            v = np.array([xj,yi])
            w = matrx @ v.T
            if tosee and (i==1)and(j==2):
                 print('v(iy=1,jx=2)=',end=''); print(v)
                 print('v.T(iy=1,jx=2)=',end=''); print(v.T)
                 print('A v.T=',end=''); print(w)
```

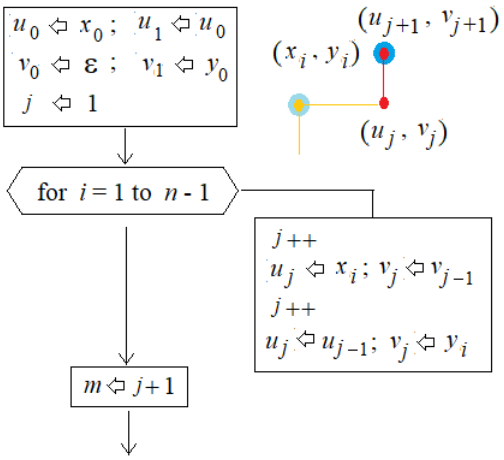
```
                print('v A v.T=',end=''); print(np.dot(v,w))
            z[i,j] = np.dot(v, w)
    return z

zz = findz(xx, yy)
xycenter= np.array([0.0, 0.0])
figure = plt.contourf(xx,yy,zz,levels=[1.,2.,3.,4.,5.],\
    colors=['#FF6666','#FFB266','#FFFF66','#B2FF66', \
        '#66FF66','#66FFFF'])

font1 = {'family':'serif','color':'blue','size':11}
font2 = {'family':'serif','color':'darkred','size':12}
plt.title(r"$\Delta \chi^2$ contours (Fig.7.5)", fontdict = font1)
plt.xlabel(r"$\Delta \theta_1$", fontdict = font2)
plt.ylabel(r"$\Delta \theta_2$", fontdict = font2)
plt.grid()
plt.show()
```



5. モジュール

5.1. “hwblines2.py”

zigzag は，右のフローチャートに従って入力データ(x_i, k_i)からジグザグ曲線を作ります．(x_i, k_i)は階段の踏み板の端に位置しますが真ん中に来る図もあります(p.61)．Array 向きのフローチャートですが，モジュールでは List で作ってあります．



tick_label は，指定した目盛り（tick という短い横線と数字）を確率紙に適合するように作画します．

```
import numpy as np
import hwbfiles as hf
from scipy.special import erfinv

class zigzag:
    def __init__(self, x, y, ndata, eps):
        inc = True
        for i in range(ndata-1):
            inc = inc and (x[i]<=x[i+1])
        if not inc:
            x.sort()
            print(' hwblines: x is sorted now')
        else:
            print(' hwblines: x is already sorted')
        u = []; v = []
        u.append(x[0]); u.append(u[0]);
        v.append(eps); v.append(y[0])
        j = 1
        for i in range(1,ndata):
            j += 1; u.append(x[i]); v.append(v[j-1]);
            j += 1; u.append(u[j-1]); v.append(y[i])
```

```
        self.ndata = ndata
        self.npoint = j + i
        self.xa = x
        self.ya = y
        self.ua = u
        self.va = v

class tic_label:

    def __init__(self, eps):
        feps = erfinv(1.0 - 2.0*eps)

        def finvcumulat(y):
            z = (1.0 - 2.0*eps)*erfinv(1.0 - 2.0*y)/feps
            return 0.5*(1.0 - z)

        p = np.array([eps, 0.02, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5,\
            0.6, 0.7, 0.8, 0.9, 0.95, 0.98, 1.0-eps])
        s = str(eps)
        ss = str(1.0 - eps)
        q = np.array([s, '0.02', '0.05', '0.1', '0.2', '0.3', '0.4', \
            '0.5', '0.6', '0.7', '0.8', '0.9', '0.95', '0.98', ss])
        nticks = 15
        r = np.arange(0.0, float(nticks), 1.0)

        for i in range(nticks):
            r[i] = finvcumulat(p[i])

        self.tics = r
        self.labels = q
        self.nticks = nticks
        self.finvcumulat = finvcumulat

    def info(self):
        print("Information on class 'tic_label'")
        print('nticks:',self.nticks)
        print('labels:',self.labels)
        print('tick position:')
        for i in range(self.nticks):
            print(f"{self.tics[i]:,.3f}",end=" ")
        print()
```

5.2. “hwbfiles.py”

例えば tab2.1x.csv を Excel で開くと 右図の通りです．B 列はなくても構わないのですがプログラミングに習熟することを目的として入れてあります．1 行目はヘッダー（見出し）です． I または i で始まれば以下は整数，さもなければ実数であると決めました．このファイルを開くには自作モジュール hwbfiles.py のオブジェクト

	A	B	C
1	data	i-bin	
2	6.61	1	
3	7.19	2	
30	11.28	29	
31	11.39	30	
32			

class fileRead(fname, check)

を使います．check=True であれば読み込みの様子が表示されます．

（注）csv という公式モジュールを使えばエレガントに読み込めるようです．

```
import numpy as np
class fileRead:
# file format:  header = 'x', 'i-y', 'z'
# if first letter is i or I, the item is an integer
#              data 1  = '2.5', 10, '0.1'
#              data 2  = '3.5', 12, '0.2'
#               ...
# number of data equals number of data lines
# number of items is limited by maxitem, i.e. 3

    def __init__(self,fname,check):
        xa = []
        ya = []
        za = []
        maxitem = 3
        isf = np.arange(maxitem)
        file = open(fname, 'r', encoding='utf-8')
        a_line = file.readline()
        self.header = a_line

        l = len(a_line)
        a_line = a_line[:l-1]
        result = a_line.split(',')
        if check:
            print('Header = ',self.header)

        for x in reversed(result):  # remove ""
```

```
    if len(x)>0:
        break
    else:
        result.remove(x)

n_item = len(result)
for i in range(n_item):
    s = result[i].upper()
    isf[i] = (s[0] != 'I')
    if isf[i]==True:
        print('float, ', end="")
    else:
        print('integer, ', end="")
print()

i = 0
while (True):
    a_line = file.readline()
    l = len(a_line)
    a_line = a_line[:l-1]
    if len(a_line)==0:            # end of file
        break
    result = a_line.split(',')
    if check:
        print(i+1,') ',result)

    if isf[0]:
        xa.append(float(result[0]))
    else:
        xa.append(int(result[0]))
    if n_item > 1:
        if isf[1]:
            ya.append(float(result[1]))
        else:
            ya.append(int(result[1]))
    if n_item > 2:
        if isf[2]:
            za.append(float(result[2]))
        else:
            za.append(int(result[2]))
    if check:
        print(i+1,') ',result[0],' ', end='')
```

```
            if n_item>1:
                print(result[1],' ', end='')
                if n_item>2:
                    print(result[2])
            print()

        i += 1
    file.close()

    self.fname = fname
    self.ndata = i
    self.nitem = n_item
    self.xa = xa
    self.ya = ya
    self.za = za
```

5.3. “statpos.py”

posn で以て平均，メジアン，モード，$\pm\sigma$，$\pm 2\sigma$に対応する累積分布上の点(x, y)を決めます.

```
import numpy as np
class posn:

    def __init__(self,lines,nline,x,y,ndata,aver,sigma,med):
        posx = np.arange(nline, dtype = 'f')
        posy = np.arange(nline, dtype = 'f')

        def findXpos(i):
            k = 0
            while True:
                if k>=ndata:
                    break
                if x[k]>lines[i]:
                    break
                k += 1
            return k

        lines[0] = aver - 2.0*sigma
        lines[1] = aver - sigma
        lines[2] = aver
        lines[3] = med
```

```
lines[4] = aver + sigma
lines[5] = aver + 2.0*sigma

for i in range(nline):
    k = findXpos(i)
    if k>0:
        dx2 = x[k]-lines[i]
        dx1 = lines[i]-x[k-1]
        posx[i] = lines[i]
        posy[i] = (y[k]*dx1 + y[k-1]*dx2)/(dx1+dx2)
        print("pos="+f"{lines[i]:,.2f}"+":dx1="+f"{dx1:,.2f}"+",
       dx2="+f"{dx2:,.2f}")
    else:
        posx[i] = lines[i]
        posy[i] = y[k]

self.posx = posx
self.posy = posy
```

6. 確率紙：累積分布で刻みの間隔を変える

6.1. バーチャルな確率紙

正規分布に従う離散的なデータ点があるとします．その累積分布を確率紙の上で作れば直線上に並びます(テキスト p.43)．なぜなら正規分布のピークに相当する確率（50%）付近で確率紙の罫線間隔が狭く，逆に裾野（0%と 100%の近く）で間隔が広くなっているからです．その場合，元の累積分布曲線(x,y)と確率紙上の曲線(x,z)を結びつける関係式が p.2, 式(2) の $f(y)$です．この $f(y)$関数が分かっていればパソコン上でグラフが作画できますのでもはや確率紙は不要です．そこで $f(y)$ を連続的な正規分布 $N(x)$ について求めることにしましょう．

6.2. 連続分布の累積分布

平均値が a，標準偏差がσの正規分布であれば累積分布(x, y)は

$$y = \int_{-\infty}^{x} N(x)dx = \frac{1}{2}\mathrm{erf}\left(\frac{x-a}{\sigma\sqrt{2}}\right) \tag{3}$$

となります．ここで $N(x)$ は正規分布関数，`erf` は誤差関数です．直線にもっていくには逆関数を用いて

$$x' = \sqrt{2}\sigma\,\mathrm{erf}^{-1}(2y-1) + a \tag{4}$$

をまず作ります．次に

$$z = Bx' + C \tag{5}$$

という形を仮定して係数B, Cを決めます．erf^{-1} の発散を防ぐために

$$f(\varepsilon) = \varepsilon\,,\ f(1-\varepsilon) = 1-\varepsilon \tag{6}$$

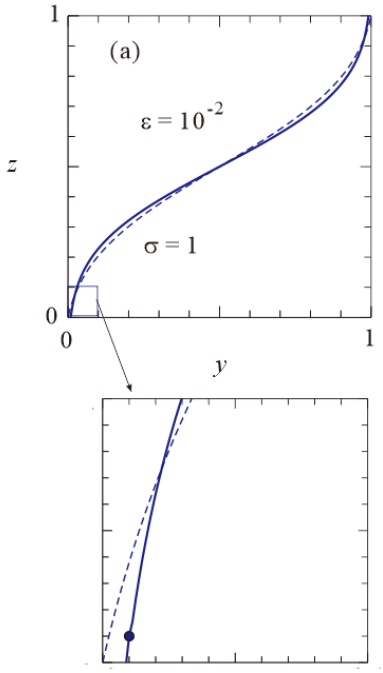
という条件を課すことにすれば次の式が導かれます．

$$z = f(y) = \frac{1}{2}\left[1 - \frac{1-2\varepsilon}{\mathrm{erf}^{-1}(1-2\varepsilon)}\mathrm{erf}^{-1}(1-2y)\right] \tag{7}$$

この式は先の式(1)と同じです． $\mathrm{erf}^{-1}(x)$ は誤差関数 erf(x) の逆関数で，Python では erfinv という名前がついています．この$f(x)$関数は §4.1 で紹介した `hwbline2` というモジュールに入れてあって

```
tick_label(eps).finvcumulat(x)
```

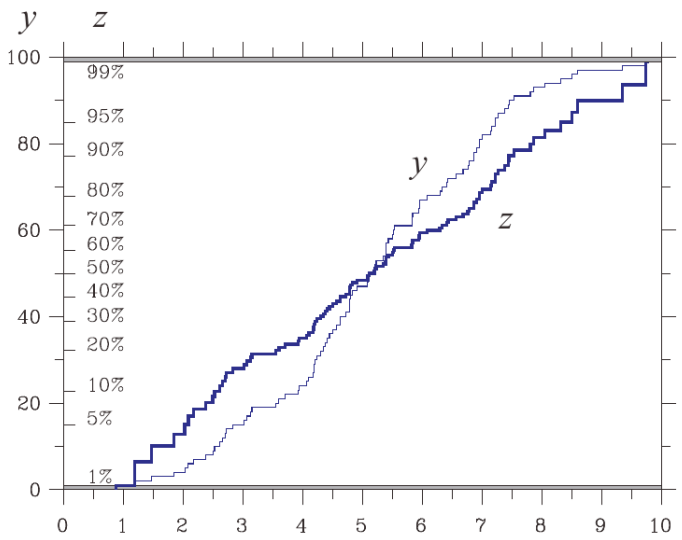
のようにして使います．



$\varepsilon = 0.01$ として得られた y-z 関係が右のグラフの実線です．その下の原点付近の拡大図からも$f(y)$関数が発散することが分かります．●はεの位置(0.01)です．εよりも y が小さくなると z が負になるという異常事態が起きてしまいます．よって y, z の有効範囲は ε と $1-\varepsilon$ の間とすべきです．当然 0%と100%は除外されます．なお，図の破線は式(1)の-∞を実際の下限にとった場合の結果です．

6.3. 離散分布への適用例

有限個のデータに対しても$f(y)$が適用できると考えます．まず Box-Müller 法で生成した 100 個の正規乱数 x について累積度数を求めて，その相対値 y の累積分布(x, y)を作りました(p.4, (ii)(c)に相当)．次に式(7)の変換によって確率紙上の累積分布(x, z)を作りました．下図がその結果です．確かに x-z 曲線は直線部分が長くなっています．



7．おわりに

「データ・誤差解析の基礎」に掲載されているプログラムのうち代表的なものを 7 つ選んで今でも走るかを試験しました．その結果，すべてが大なり小なり修正を必要とすることが分かりました．著者の Dr. Berendsen が中心となって開発した plotsvg が使えなくなったのは残念です．しかし，現在の Python システムでも同等の作業が十分可能であることを皆さんに示すことができたと思います．