전산천문학 HW3

2013-12239 서유경
제출일 :17.04.20(하루 늦었습니다)

1번. 적분하기

```
import numpy as np
from numpy import *

n=100000
def f(x):
    return sin(x**4)   #함수 정의

h=(4.-0.)/n #trapezoidal rule과 simpsons rule에 쓰일 h 크기 지정
ST=(f(0)+f(4))*h/2 #두 방법 모두 f(a),f(b)의 합은 미리 지정해놓는다.
SS=(f(0)+f(4))*h/3
SG=0.  #가우시안 르장드르법의 합계 변수 정의

#(a) the composite trapezoidal rule
for i in range(1,n):
    x=0+i*h
    ST+=f(x)*h

#(b) the composite Simpson's Rule
for i in range(1,n):
    if(i%2==1):  #i가 짝수 번째면 2h*f(x)를 더하고, 홀수 번째면 4h*f(x)를 더하는것을 if문으로 이용
        x=i*h
        SS+=4*h*f(x)/3
    else:
        x=i*h
        SS+=2*h*f(x)/3

#(c) Gaussian Quadrature
t, w=np.polynomial.legendre.leggauss(1000)
for i in range(1000):
    SG+=(4-0)/2*w[i]*f((t[i]*(4-0)+4)/2)
print "The composite trapezoidal rule : {:.6f}".format(ST)
print "The composite Simpson's rule : {:.6f}".format(SS)
print "The Gaussian Quadrature : {:.6f}".format(SG)
```

결과 : 셋 다 0.347032

IPython console

Console 1/A

```
    ...:         x=i*h
    ...:         SS+=2*h*f(x)/3
    ...:
    ...:

In [632]: t, w=np.polynomial.legendre.leggauss(1000)

In [633]: for i in range(1000):
    ...:     SG+=(4-0)/2*w[i]*f((t[i]*(4-0)+4)/2)
    ...:
    ...:

In [634]: print "The composite trapezoidal rule : {:.6f}".format(ST)
The composite trapezoidal rule : 0.347032

In [635]: print "The composite Simpson's rule : {:.6f}".format(SS)
The composite Simpson's rule : 0.347032

In [636]: print "The Gaussian Quadrature : {:.6f}".format(SG)
The Gaussian Quadrature : 0.347032

In [637]:
```

History log | IPython console

ermissions: RW | End-of-lines: CRLF | Encoding: UTF-8 | Line: 44 | Column: 1 | Memory: 64 %

2번. 함수의 정의에 적분이 들어가 있는 경우

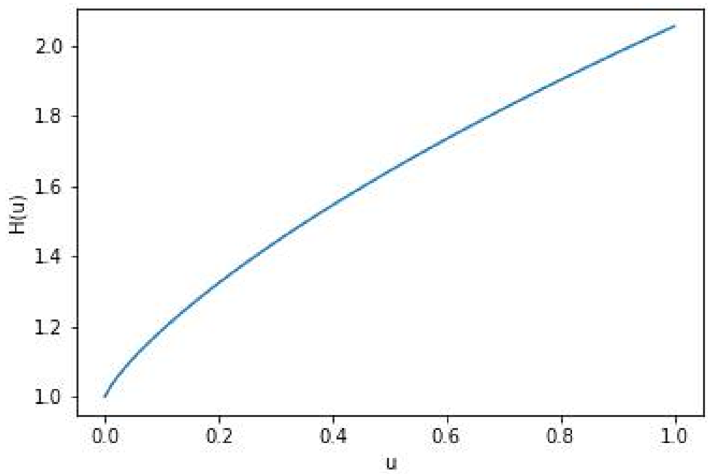
```
import numpy as np
from numpy import *
import matplotlib.pyplot as plt
n=100000
ui=np.arange(0,1.01,0.01) #함수가 H(u)이므로, u값을 array로 0.01간격씩 지정

def H(p,u):
    return (p*arctan(u*tan(p))/(1-p/tan(p)))
#함수에서 적분구간에 해당하는 부분만 따로 때내서 함수로 정의

Hi=np.zeros(len(ui)) #그래프를 만들기 위해 zeros 설정
h=(0.5*pi-0)/n

#the composite Simpson's Rule
```

simposon's rule은 f(a)와 f(b)는 2나 4를 곱하지 않고 그냥 더하는데 문제는 phi가 0과 0.5pi이면 분모가 0이되거나, 분자가 무한이 되버려 컴퓨터로는 계산이 안된다. 0에 근접한 매우 작은 값과, 0.5pi에 근접하지만 0.5pi가 아닌 값을 잡는다.

```
for j in range(len(ui)):
    SS=(H(0.0000001,ui[j])+H(0.4999999*pi,ui[j]))*h/3
    for i in range(1,n):
        if(i%2==1):
            x=i*h
            SS+=4*h*H(x,ui[j])/3
        else:
            x=i*h
            SS+=2*h*H(x,ui[j])/3
    Hi[j]=exp(SS/pi)/(1+ui[j])
```

-> j번째 u값에 따른 H(u) 함수값을 simpson's rulr로 구한다. 마지막의 Hi[j]의 오른쪽 식은 원래 H(u) 식을 넣은 것.

```
plt.plot(ui,Hi),plt.xlabel('u'),plt.ylabel('H(u)'),plt.savefig('Hw3-2.png')
```

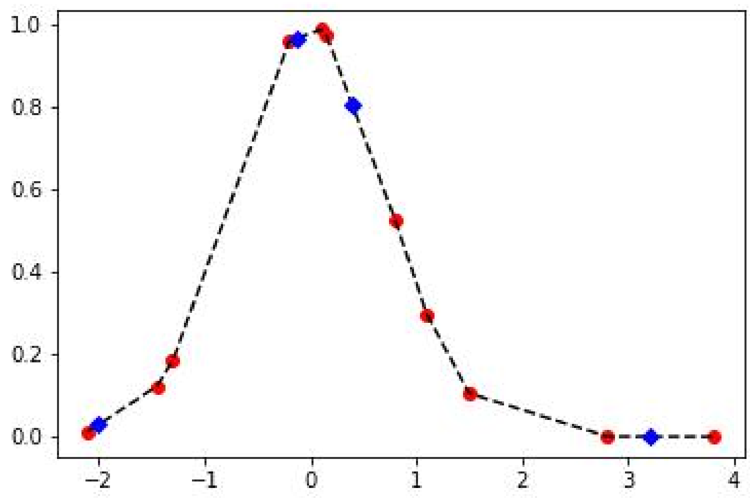


3번. Interpolation 하기

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

x=np.array([-2.1,-1.45,-1.3,-0.2,0.1,0.15,0.8,1.1,1.5,2.8,3.8])
y=np.array([0.012155,0.122151,0.184520,0.960789,0.990050,0.977751,0.527292,0.298197,0.105399,3.936690e-4,5.355348e-7])

from scipy.interpolate import inter1d
f1=interp1d(x,y)
f10=interp1d(x,y,kind=10)
#(a)의 linear polation이 된 함수를 f1, (b)의 10차 다항식으로 polation이 된 함수를 f10으로 지정한다.


x1=np.array([3.2,0.4,-0.128,-2.0])
n=np.arange(-1.9,3.8,0.1)
y1=f1(x1)
y10=f10(x1)
```

#(a) linear polation

```
plt.plot(x,y,'ro'),plt.plot(x,f1(x),'k--',x1,f1(x1),'bD'),plt.savefig('Hw3-3-(a)-1.png')
# x=(3.2) -> y=2.36415614e-04
# x=(0.4) -> y=8.04497538e-01
# x=(-0.128) -> y=9.67811640e-01
# x=(-2.0) -> y=2.90774615e-02
```



붉은 점이 원래 x,y점, 파란점이 문제에서 구하라고한 x값들과 그 결과물

#(b) 10th order polation

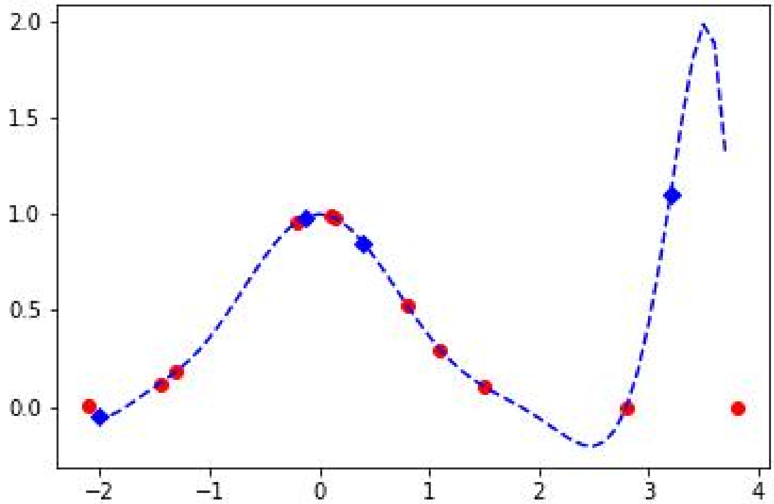
plt.plot(x,y,'ro'),plt.plot(n,f10(n),'b--',x1,y10,'bD'),plt.savefig('Hw3-3-(b)-2.png')

x=(3.2) -> y=1.09718507

x=(0.4) -> y=0.85256479

x=(-0.128) -> y=0.98389252

x=(-2.0) -> y=-0.04753688



#(c) Cubic Spline 하기

from scipy import interpolate

result=interpolate.CubicSpline(x,y,bc_type='not-a-knot')

plt.plot(n,result(n),'y'),plt.plot(x,y,'ro'),plt.plot(x1,result(x1),'bD'),plt.savefig('Hw3-3-(c).png')

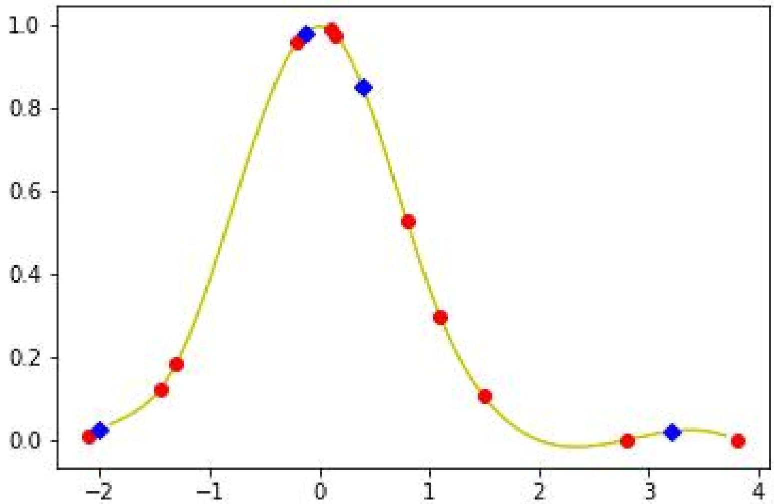
y_cubic=result(x1)

x=(3.2) -> y=0.02091338

x=(0.4) -> y=0.8508707

x=(-0.128) -> y=0.98271128

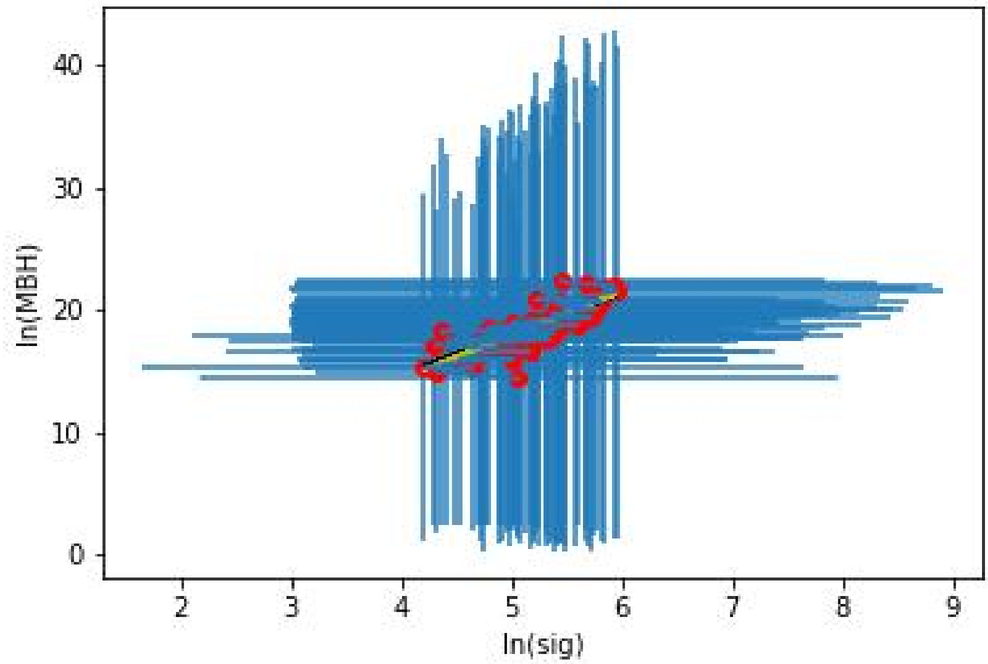
x=(-2.0) -> y=0.02681691



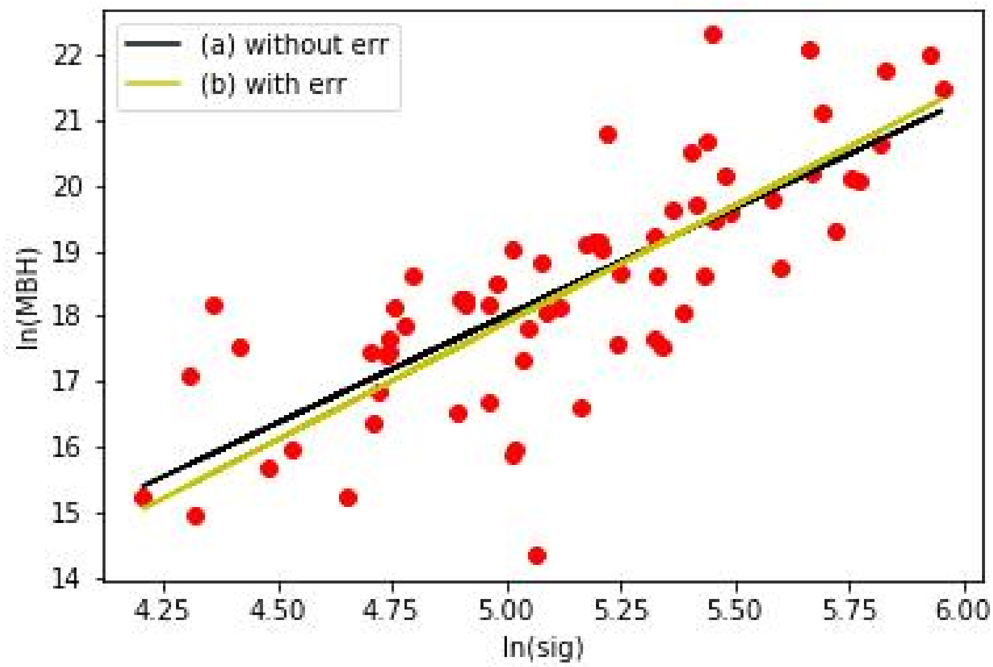
4번. Blackhole 질량과 속도분산 linear 그래프 그리기.
<코드>

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

file = 'BlackHall.txt'
MBH, dMBH, sig, dsig=np.loadtxt(file,unpack=True,usecols=[0,1,2,3])


#(a) 측정 오차 무시하고 linear 함수와 a,b값 구하기
cofa, resa, _, _, _=np.polyfit(log(sig),log(MBH),1,full=True)

print cofa
print resa

#a=1.56205097
#b=3.28965514
#err=85.82221847

#(b) Consider the measurement Error
def lin_func(p,X):
    return p[0]+p[1]*X

lin_model=Model(lin_func)
data=RealData(log(sig),log(MBH),sx=log(dsig),sy=log(dMBH))
odr=ODR(data,lin_model, beta0=[0.,1.0])

out=odr.run()
pb=out.pprint
beta=out.beta
sd_beta=out.sd_beta
s_sq=out.sum_square
beta
s_sq
#a=-0.03526799
#b=3.58736461
#err=0.2817686570823044


#(c) plotting
def fa(x):
    return cofa[1]+cofa[0]*x
plt.errorbar(log(sig),log(MBH),xerr=log(dsig),yerr=log(dMBH)),plt.plot(log(sig),log(MBH),'ro'),plt.plot(l
```

og(sig),fa(log(sig)),'k-'),
plt.plot(log(sig),lin_func(beta,log(sig)),'y-'),plt.xlabel('ln(sig)'),plt.ylabel('ln(MBH)'),plt.savefig('Hw3-4-(c)-1.png')
plt.plot(log(sig),log(MBH),'ro'),plt.plot(log(sig),fa(log(sig)),'k-',label='(a) without err'), plt.plot(log(sig),lin_func(beta,log(sig)),'y-',label='(b) with err'),plt.legend(loc=2),plt.xlabel('ln(sig)'),plt.ylabel('ln(MBH)'),plt.savefig('Hw3-4-(c)-2.png')



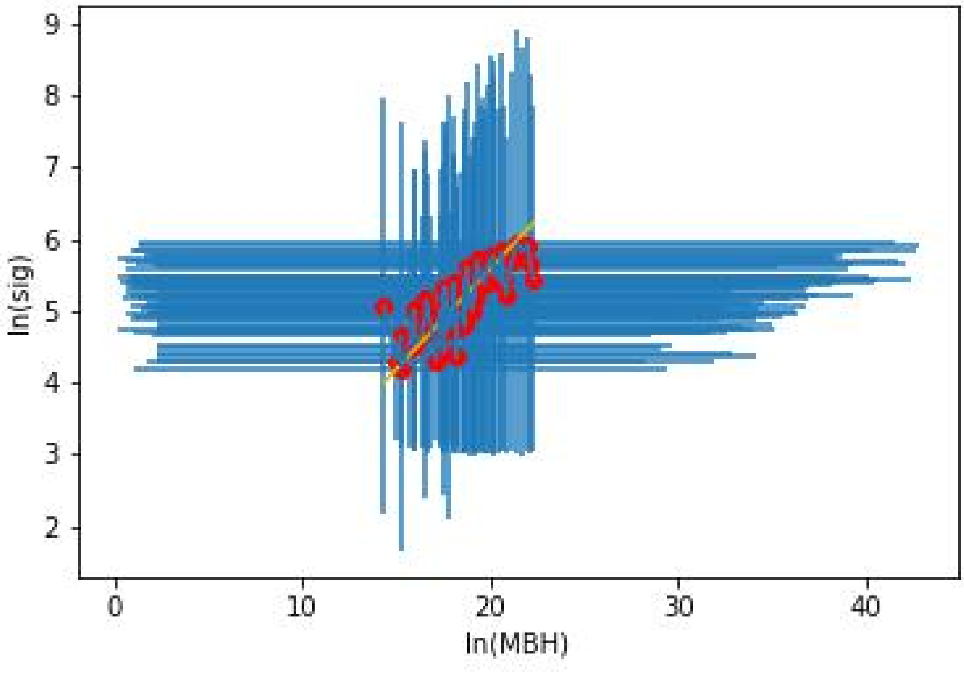
4-(c). Errorbar를 넣은 경우
(Error bar의 크기는 measurment error의 log 값이다)



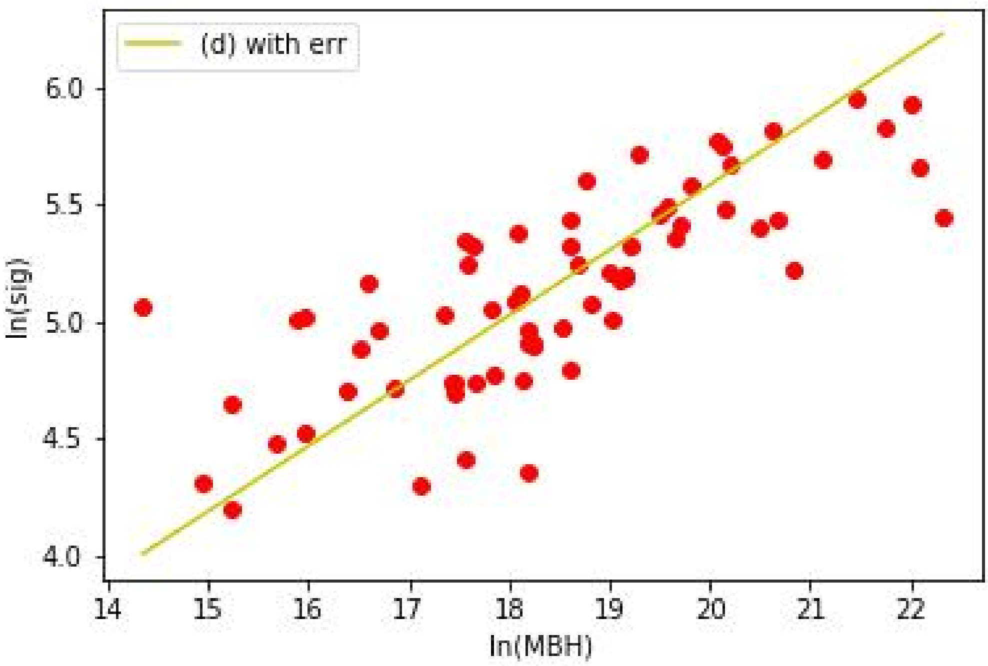
4-(c) Errorbar를 넣지 않은 경우

#(d) 측정오차 고려하여, x를 MBH로 y를 속도분산으로 잡고 선형관계식 구하기.

```
data=RealData(log(MBH),log(sig),sx=log(dMBH),sy=log(dsig))
odr=ODR(data,lin_model, beta0=[0.,1.0])

out=odr.run()
pd=out.pprint
betad=out.beta
sd_betad=out.sd_beta
s_sqd=out.sum_square


plt.errorbar(log(MBH),log(sig),xerr=log(dMBH),yerr=log(dsig)),plt.plot(log(MBH),log(sig),'ro'),
plt.plot(log(MBH),lin_func(betad,log(MBH)),'y-'),plt.xlabel('ln(MBH)'),plt.ylabel('ln(sig)'),plt.savefig('Hw3-4-(d)-1.png')
plt.plot(log(MBH),log(sig),'ro'), plt.plot(log(MBH),lin_func(betad,log(MBH)),'y-',label='(d) with err'),plt.legend(loc=2),plt.xlabel('ln(MBH)'),plt.ylabel('ln(sig)'),plt.savefig('Hw3-4-(d)-2.png')
```



(Errorbar 넣은 경우, Error bar는 measurment error의 log 값이다)



● (b)에서 구한 a(절편),b(기울기)와 (d)에서 구한 c(절편),d(기울기)의 관계
#c=0.00990783
#d=0.27875184
#err=0.2817686571030568
이렇게 나오는데

-a/b=0.009831169961154523
1/b=0.27875616444315915
즉, 완벽하게 일치하지 않지만, (d)에서 구한 식은 (b)에서 구한 식의 역함수가 된다는 얘기고 일차함수의 역함수끼리의 계수관계와 같다. 즉 c는 -a/b랑 비슷하고, d는 1/b랑 비슷하다.

5번. 가우시안과 로렌지안으로 피팅하기

```
import numpy as np
from numpy import *
import matplotlib.pyplot as plt
from scipy.odr import *

file = 'hw3p5.dat'
x, y=np.loadtxt(file,unpack=True,usecols=[0,1])
n=np.arange(min(x),max(x),0.1)
```

#(a) fit Gauss function

```
 def Gaussf(p,t):
     tc=t-p[2]
     sig2=p[3]**2
     return p[0]+p[1]*exp(-0.5*tc**2/sig2)

model=Model(Gaussf)
data=RealData(x,y)
odr=ODR(data,model,beta0=[1.,1.,10.,1.])
out=odr.run()
G_p=out.pprint
G_beta=out.beta
G_sd_beta=out.sd_beta
G_sq=out.sum_square
```

#p0=2.24597071, p1=1.21002339, p2=9.85323186, p3=3.2682955
#G_sq=7.6927121533619935

#(b) fit lorentz

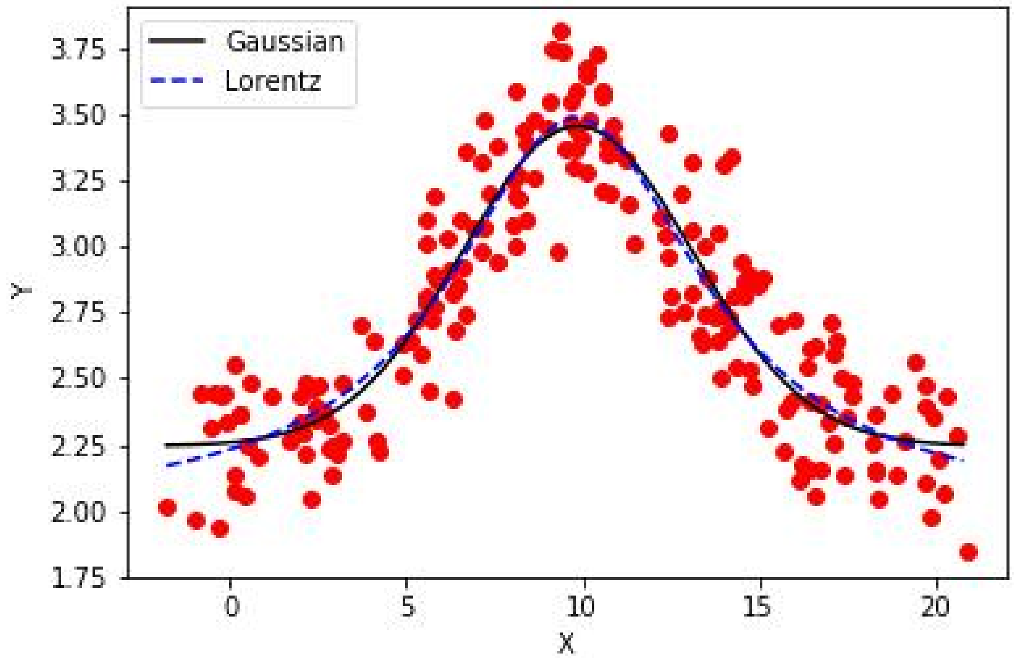
```
def Lorf(q,t):
    tc=t-q[3]
    return q[0]+q[1]/(q[2]+tc**2)

model=Model(Lorf)
data=RealData(x,y)
odr=ODR(data,model, beta0=[1.,1.,2.,10.])
out=odr.run()
L_p=out.pprint
L_beta=out.beta
L_sd_beta=out.sd_beta
```

L_sq=out.sum_square

q0=1.98483518, q1=28.60639057, q2=19.0576559, q3=9.8159773
#L_sq=7.770760814699901

#(c) plotting

plt.plot(x,y,'ro'),plt.plot(n,Gaussf(G_beta,n),'k-',label='Gaussian'),plt.plot(n,Lorf(L_beta,n),'b--',label='Lorentz'),plt.xlabel('X'),plt.ylabel('Y'),plt.legend(loc=2),plt.savefig('Hw3-5.png')



오차표준편차 값만 놓고보면 Gaussian이 7.69, Lorentzian이 7.77이어서 가우시안이 더 적합하다.