Levenberg-Marquardt Method

2017135002/최성윤

기본원리

$$f(x_i, \boldsymbol{\beta} + \boldsymbol{\delta}) \approx f(x_i, \boldsymbol{\beta}) + J_i \boldsymbol{\delta}$$

$$J_i = \frac{\partial f(x_i, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}}$$ Linear로 근사

$$\chi^2 \approx \sum_{i=1}^{m} (y_i - f(x_i, \boldsymbol{\beta}) - J_i \boldsymbol{\delta})^2$$

$$\chi^2 \approx \|\mathbf{y} - \mathbf{f}(\boldsymbol{\beta}) - \mathbf{J}\boldsymbol{\delta}\|^2$$

$$= [\mathbf{y} - \mathbf{f}(\boldsymbol{\beta}) - \mathbf{J}\boldsymbol{\delta}]^T [\mathbf{y} - \mathbf{f}(\boldsymbol{\beta}) - \mathbf{J}\boldsymbol{\delta}]$$

$$= [\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})]^T [\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})] - [\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})]^T \mathbf{J}\boldsymbol{\delta} - (\mathbf{J}\boldsymbol{\delta})^T [\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})] + \boldsymbol{\delta}^T \mathbf{J}^T \mathbf{J}\boldsymbol{\delta}$$

$$= [\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})]^T [\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})] - 2[\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})]^T \mathbf{J}\boldsymbol{\delta} + \boldsymbol{\delta}^T \mathbf{J}^T \mathbf{J}\boldsymbol{\delta}$$

By taking the derivative of the above and letting the result become zero, we get

$$(\mathbf{J}^T \mathbf{J})\boldsymbol{\delta} = \mathbf{J}^T [\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})] \quad \Rightarrow \quad (\mathbf{J}^T \mathbf{J} + \lambda \mathbf{I})\boldsymbol{\delta} = \mathbf{J}^T [\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})]$$

Noise 제거

실행

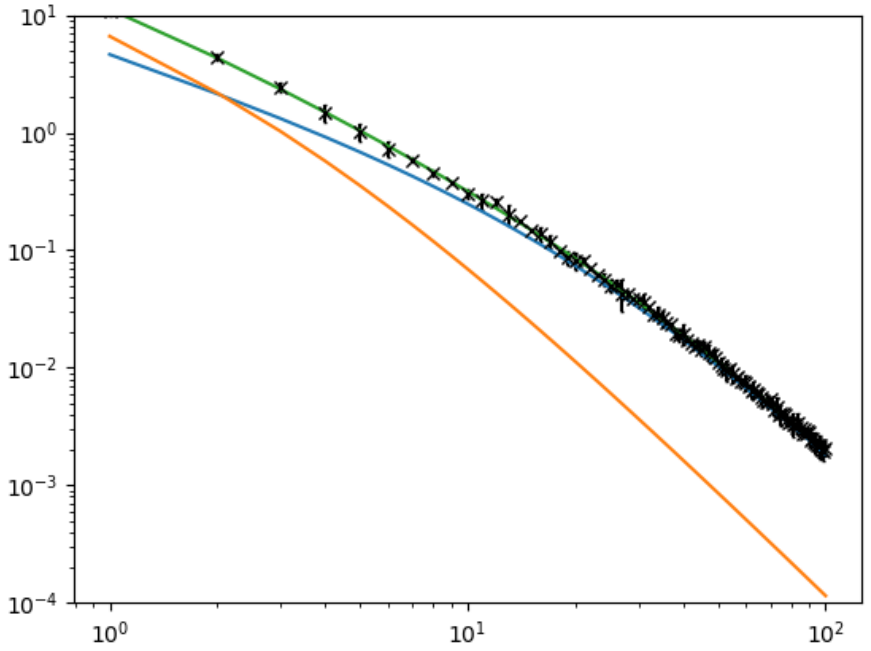
```
def snfw(R,rr1,rr2,dd1,dd2):
    return dd1/ ((R/rr1)*(1+R/rr1)**2) + dd2/ ((R/rr2)*(1+R/rr2)**2)
popt,pcov=curve_fit(snfw,r,d,sigma=derr)
```

2개의 NFW halo 합으로 간주하여 Fitting 진행

결과

```
[23.57858345  3.300218    0.21317921  3.40105426]
```

rs(1) rs(2) rh0(1) rh0(2) 값을 나타낸다

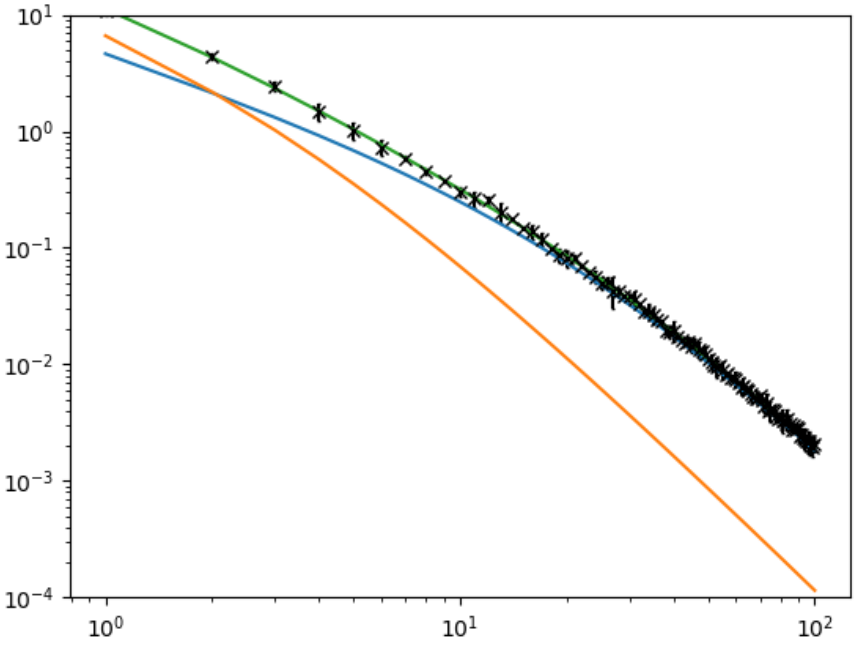


주황색은 rs=3.300218 rh0=3.40105426
파랑색은 rs=23.57858345 rh0=0.21317921
값의 NFW을 나타내며
초록색은 이 2개의 그래프의 합

토의사항

1개의 NFW 일때 Fitting과 비교

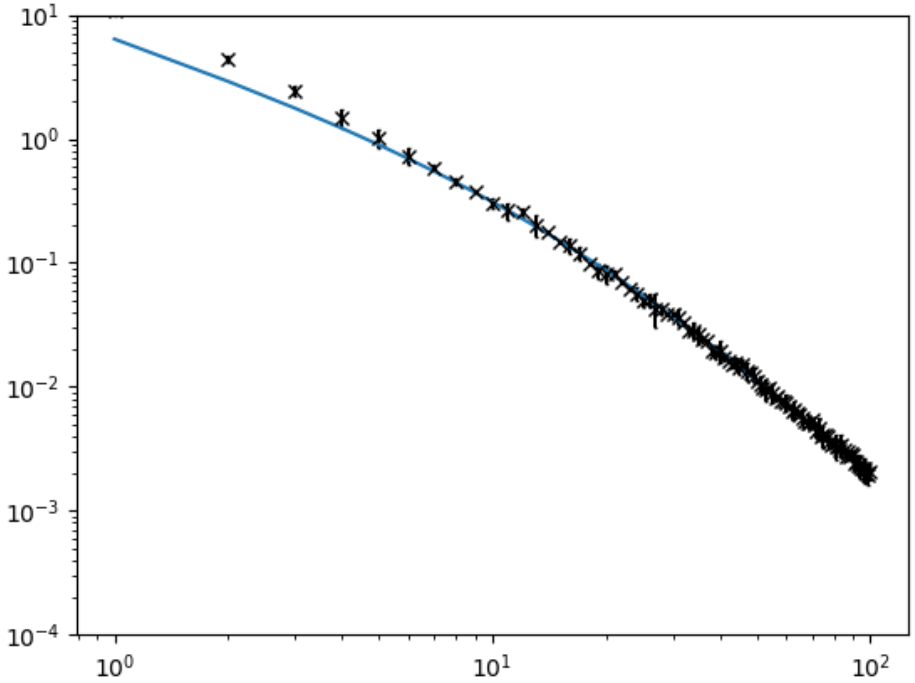
2개의 NFW의 합으로 가정



Reduced chi square

```
1.013861792762907
```

1개의 NFW

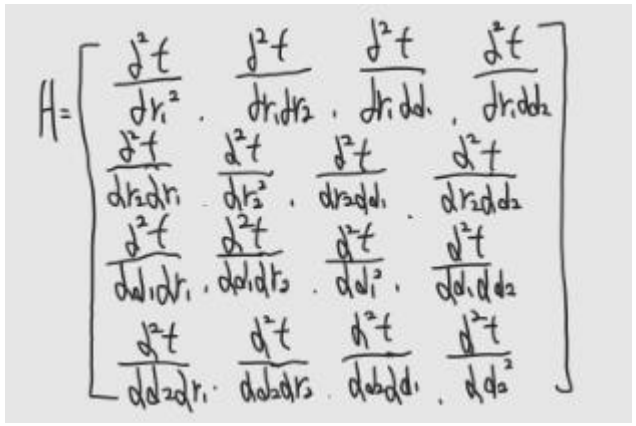


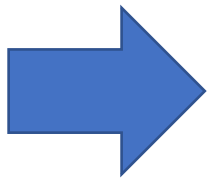
Reduced chi square

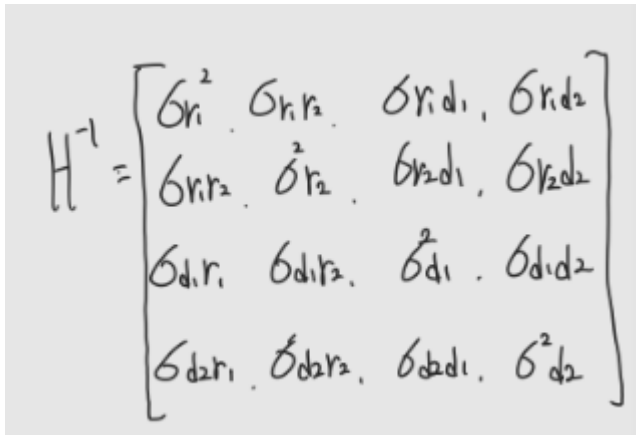
```
1.7890774108763885
```

2개의 NFW 일때 더 잘 Fitting 되었다

Uncertainty 구하기







```
[[ 1.76113384e-03 -1.34640077e-27 -3.58942411e-29 -3.34123470e-27]
 [-1.34640077e-27  1.51077213e-03 -2.99293143e-29 -3.98213453e-27]
 [-3.58942411e-29 -2.99293143e-29  7.45520715e-07 -7.51392017e-29]
 [-3.34123470e-27 -3.98213453e-27 -7.51392017e-29  1.07122460e-02]]
```

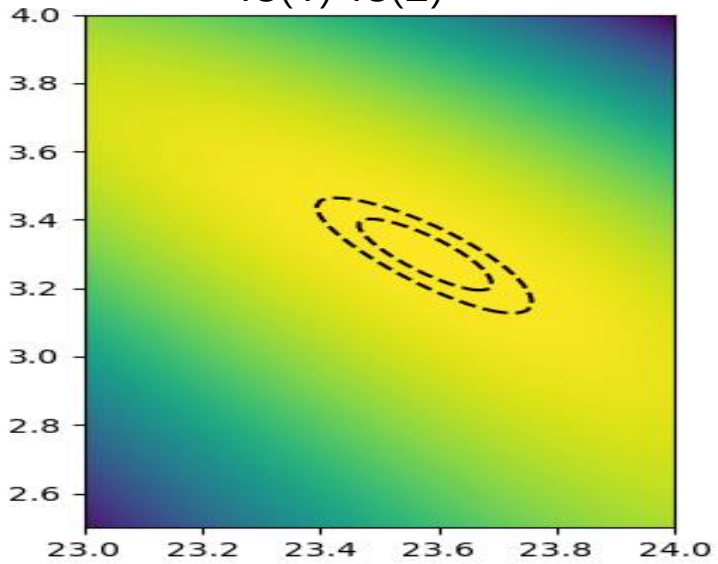
```
0.041965865146448136 0.03886865225336646 0.0008634354146099655 0.10349998082610357
```

rs(1) rs(2) rh0(1) rh0(2) 의 uncertainty를 나타낸다

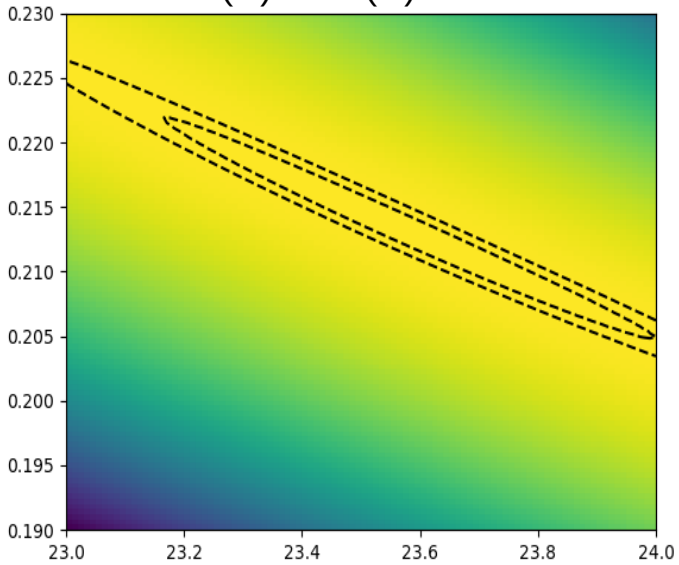
각 파라미터의 관계를 나타내는 contour 출력

Underflow를 피하기 위하여 각 파라미터의 범위를 좁게 설정하였다

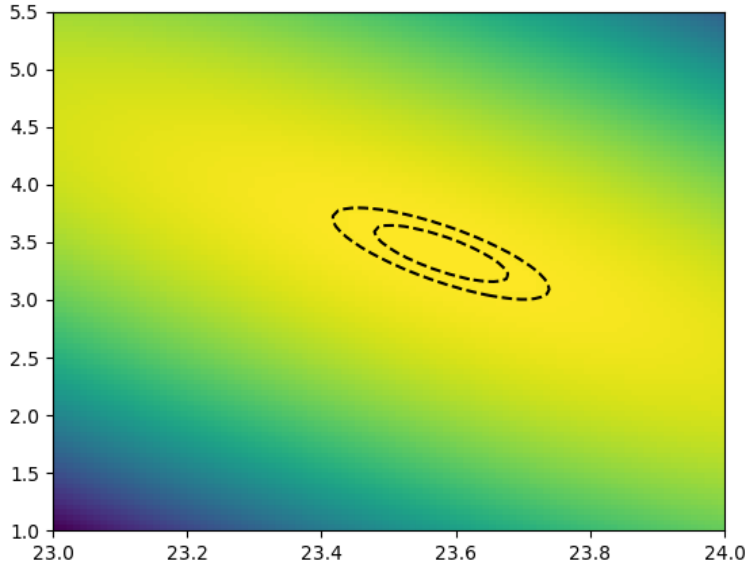
rs(1) rs(2)



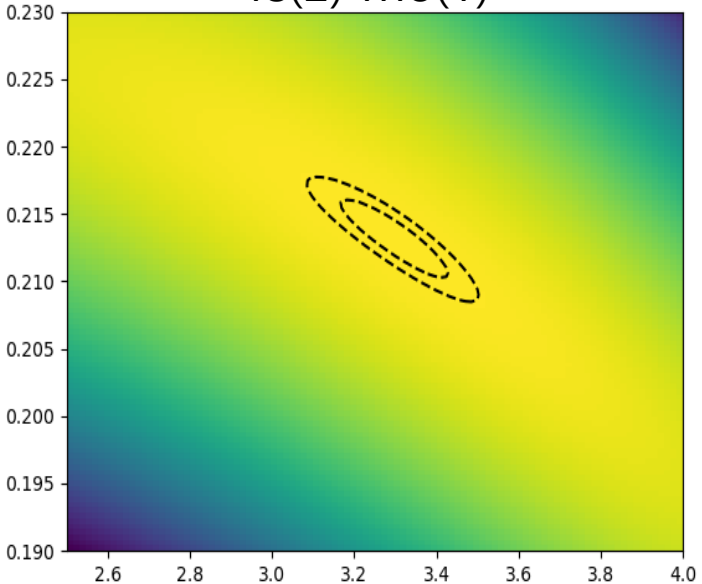
rs(1) rh0(1)



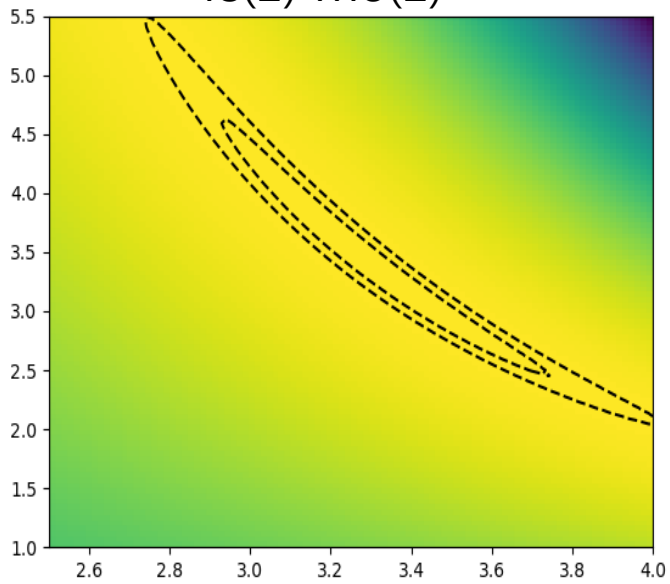
rs(1) rh0(2)



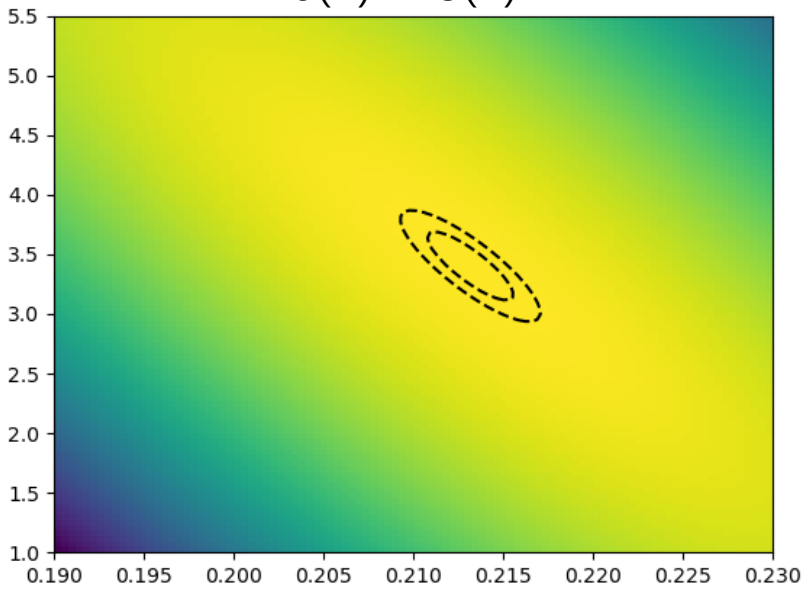
rs(2) rho(1)



rs(2) rho(2)



rh0(1) rho(2)



파라미터 간의 contour 와 uncertainty를 통하여 알수 있는 것은?

```
[[ 1.76113384e-03 -1.34640077e-27 -3.58942411e-29 -3.34123470e-27]
 [-1.34640077e-27  1.51077213e-03 -2.99293143e-29 -3.98213453e-27]
 [-3.58942411e-29 -2.99293143e-29  7.45520715e-07 -7.51392017e-29]
 [-3.34123470e-27 -3.98213453e-27 -7.51392017e-29  1.07122460e-02]]
```

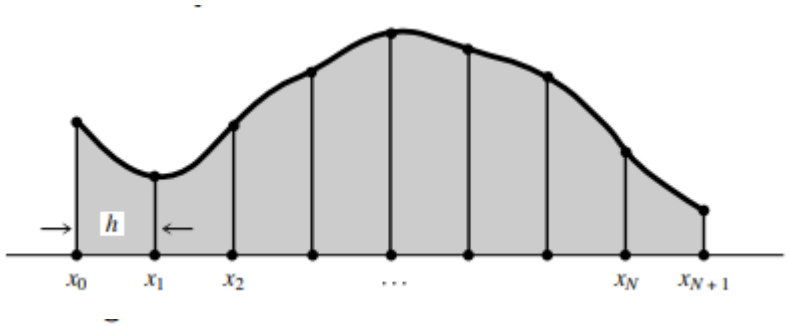
각 파라미터간의 uncertainty$[ex)\sigma_{r_1 r_2}$]는
모두 음수를 가짐을 알수있다. 이는 예를 들어 r1이 증가하면 r2는 감소하는 형태를 가진다는 것을 의미하는데

이는 contour 형태에서도 타원의 형태가 오른쪽 밑으로 향하는 방향을 가진다는 것을 의미한다.

또한 uncertainty가 비록 음수이지만 거의 0에 근접한 값을 가지는 것을 볼수 있다.
이는 각 파라미터의 관계를 contour 하면 원에 가까운 형태를 가질 것으로 추측할수 있지만
범위 지정으로 인하여 왜곡된 타원 형태가 나온다.

기본원리

Extended Trapezoidal Rule

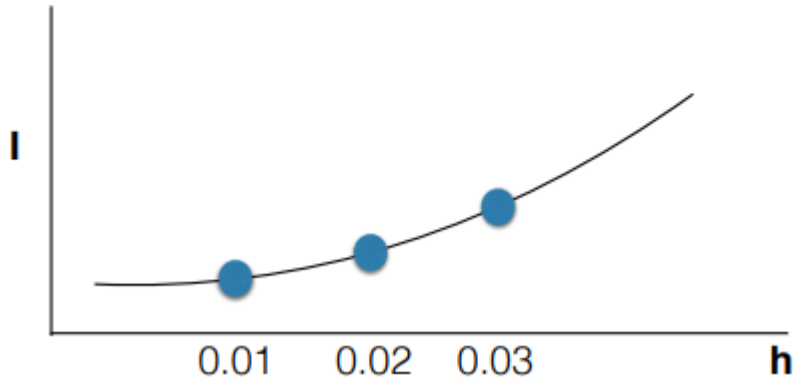


$$\int_{x_1}^{x_N} f(x)dx = h\left[\frac{1}{2}f_1 + f_2 + f_3 + \cdots + f_{N-1} + \frac{1}{2}f_N\right] + O\left(\frac{(b-a)^3 f''}{N^2}\right)$$

Romberg Algorithm

The basic idea is to use the results from k successive refinements of the extended trapezoidal rule.

Perform some numerical algorithm for various values of a parameter h and then extrapolate the result to the limit $h = 0$.



실행

```
def snfw(R):
    return 4*np.pi*(R**2)*( 1.5/ ((R/10)*(1+R/10)**2) )
```

NFW 식을 통하여
R=100까지의 halo의 질량 계산

$$\rho(r) = \frac{\rho_0}{r/r_s(1+r/r_s)^2} \qquad M = \int_0^{100} 4\pi r^2 \rho dr$$

```
h=0.01 #h값 변경하면서 변화 확인
r=np.arange(0.0001,100,h)
t=list(r)
t.append(100)
rr=np.array(t)

ET=0
for i in range(len(rr)):
    if i==0:
        ET+= (h*snfw(rr[i]))/2

    elif i==len(rr)-1:
        ET+=(h*snfw(rr[i]))/2

    else:
        ET += (h * snfw(rr[i]))
```

Extended Trapezoidal Rule

의 값을 계산

$$\int_{x_1}^{x_N} f(x)dx = h\left[\frac{1}{2}f_1 + f_2 + f_3 + \cdots + f_{N-1} + \frac{1}{2}f_N\right] + O\left(\frac{(b-a)^3 f''}{N^2}\right)$$

Romberg의 경우 라이브러리 사용

```
romberg(snfw,0.0001,100,show='True')
```

결과

Extended Trapezoidal Rule

(h=10) = 26776.28973588286
(h=4) = 27821.101211840185
(h=2) = 28000.79392655319
(h=1) = 28047.549954900536
(h=0.5) = 28059.36627094748
(h=0.1) = 28063.158562721754
(h=0.01) = 28063.31510641415
(h=0.001) = 28063.31667138152

Romberg Algorithm

```
  1 99.999900 7790.007364
  2 49.999950 16984.951255 20049.932552
  4 24.999975 23001.321142 25006.777771 25337.234119
  8 12.499987 26204.923457 27272.790896 27423.858437 27456.979458
 16  6.249994 27502.457021 27934.968209 27979.113363 27987.926934 27990.009081
 32  3.124997 27913.059087 28049.926442 28057.590324 28058.835990 28059.114065 28059.181616
 64  1.562498 28024.969499 28062.272969 28063.096071 28063.183464 28063.200513 28063.204508 28063.205490
128  0.781249 28053.666774 28063.232532 28063.296503 28063.299684 28063.300140 28063.300237 28063.300261 28063.300266
256  0.390625 28060.889253 28063.296747 28063.301028 28063.301100 28063.301105 28063.301106 28063.301106 28063.301106 28063.301106
512  0.195312 28062.697940 28063.300835 28063.301108 28063.301109 28063.301109 28063.301109 28063.301109 28063.301109 28063.301109 28063.301109
```

해당식의 정적분 값은 28063.3011089923 으로
2개의 방법 둘 다 step을 거듭할수록 (h 간격이 좁을수록) 정적분 값에 가까워 짐을 확인 할수 있다

Romberg의 경우가 error를 고려하였기 때문에 조금 더 정확하다.