Shuffling Method

2017135002 최성윤

목표

이전의 Schrage’s Algorithm을 사용하여 만든 Random Number Generator에서 단점들을 보안한 Random Number Generator를 만들 수 있다.

이후 새로운 Random Number Generator로 발생한 균일한 난수들을 확률변환을 이용하여 Salpeter IMF 분포를 가지는 별의 질량으로 바꿀 수 있다.

또한 Rejection Method를 이용하여 Gaussian Function의 분포를 가지는 난수들을 발생시킬 수 있다.

기본원리(Shuffling Method)

Schrage's Algorithm으로 발생한 난수들을 바탕으로 Shuffling Method을 통하여 난수발생기를 작성한다.

N개의 random number 발생시켜 배열을 채운다 (Shuffling 방 생성)

두개의 난수를 발생 (X,Y)

Y를 0부터 N-1까지 범위의 정수(i)로 변환한다

Shuffling방의 i번째 난수를 꺼내 출력한다

Shuffling 방의 i번째에 X를 집어넣는다.

myran 함수를 작성한다

- myran(): 0과 1사이의 난수 한 개 발생
- myran(n): 0과 1사이의 난수 n개 발생
- myran(-1): Shuffling 방 초기화
- myran(seed=1038291): 시드값 초기화

실행

```
A = 16807
C = 2147483647
Q = 127773
R = 2836
X = 1
M = C+1
T=[]
N=[]
for i in range(32):
    X = A * (X % Q) - R * (X // Q)
    if X < 0:
        X = X + C
    N.append(X)
for i in N:
    T.append(i / (M + 1))
```

myran 함수 지정 전에 shuffling 방을 만들어 둔다.
(T를 글로벌 변수로 지정하여 사용)

```
def myran(n=1,seed=0):
    global T,A,C,Q,R,M,N,X
```

myran함수에서 n은 발생할 난수의 개수, seed는 시드값을 의미.
(n,seed값이 입력되지 않을경우 n=1, seed=0이 되도록 설정한다.)
앞에서 사용한 변수들을 글로벌로 지정하여 그대로 myran 함수에서 사용한다

```
if seed:
    X=seed
    for i in range(32):
        X = A * (X % Q) - R * (X // Q)
        if X < 0:
            X = X + C
        N.append(X)
    for i in N:
        T.append(i / (M + 1))
    print("New Seed")
```

Seed 변수에 0이외의 값이 주어졌을때 if문 실행
'X=seed'로 시드값을 재설정한후
다시 shuffling방을 만든다

```
elif n==-1:
    T=[]
    for i in range(32):
        X = A * (X % Q) - R * (X // Q)
        if X < 0:
            X = X + C
        N.append(X)
    for i in N:
        T.append(i / (M + 1))
    print("the room is reset")
```

n=-1일 경우 shuffling 방을 초기화후 재생성
(이때 시드값을 초기화 시키지 않아
처음에 만든 shuffling 방과 다른 배열 발생)
->초기화후 다시 난수 발생시 다른값 생성

```
print(myran(2))
print(myran(2))
myran(-1)
print(myran(2))
```

```
[0.13153778802066213, 0.3835020771326953]
[0.8461668897205187, 0.5194163715842104]
the room is reset
[0.09073289479560549, 0.23777443345739765]
```

실행

```python
else:
    for i in range(n):
        N=[]
        T1=[]
        for i in range(2):
            X=A*(X%Q)-R*(X//Q)
            if X < 0:
                X = X + C
            N.append(X)
        for i in N:
            T1.append(i/(M+1))
        x=T1[0]
        y=math.trunc(T1[1]*32)
        T2.append(T[y])
        T[y]=x
    return T2
```

난수를 n개 발생시켜 T2 배열에 담는 for문
시드값을 초기화 시키지 않고 이전 값을 이어서 사용하였다.

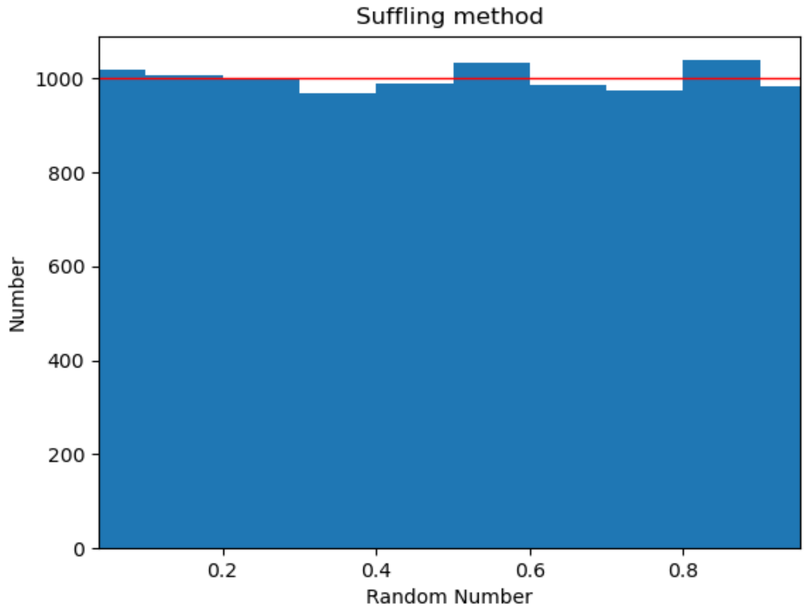
발생한 난수 2개(x,y) 중 y는 32를 곱해줘서 0~31까지의 숫자로 변환해 주었다.

T2에 발생한 난수 저장

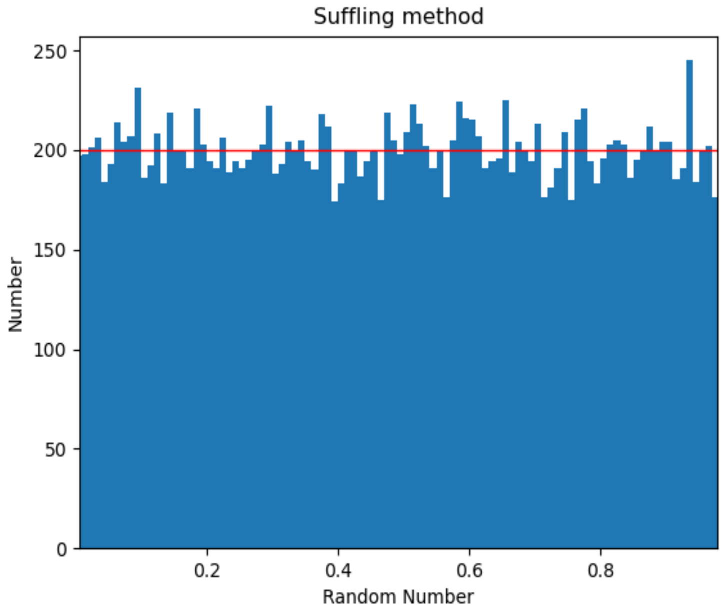
T2를 결과값으로 돌려준다

결과

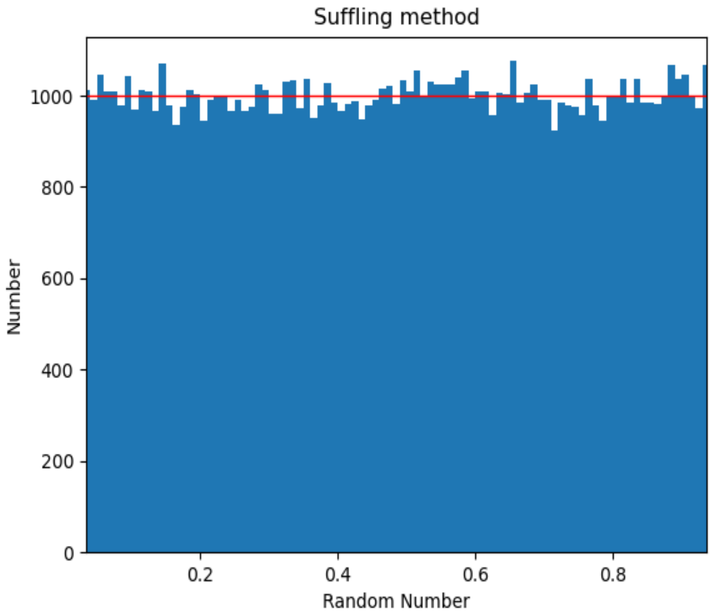
n=10000, bins=10



n=20000, bins=100



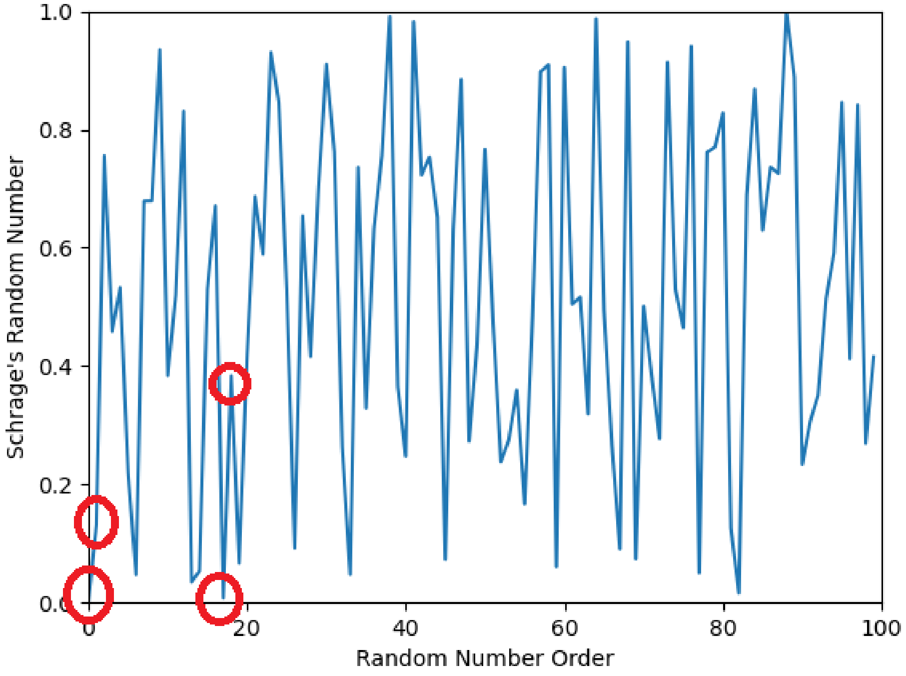
n=20000, bins=100



토의사항

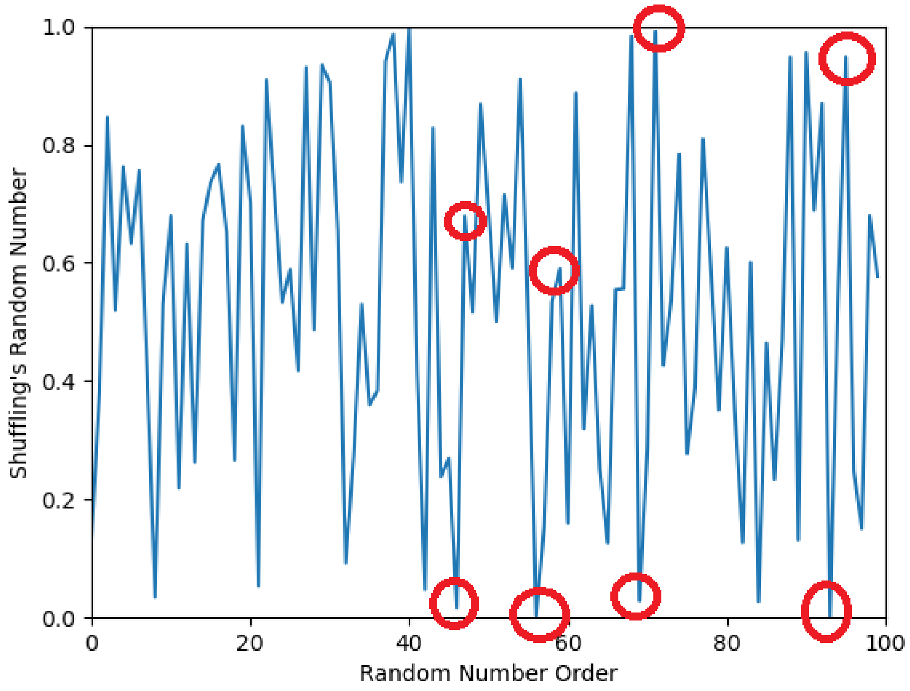
Schrage's Algorithm으로 발생시킨 난수들의 문제점이었던 연속하는 난수들의 correlation이 해결되었을까?

Schrage's Algorithm



Schrage's 에서는 직전의 난수가
작을경우그다음 난수값의 범위
가 제한 되었다.
(Correlation exists)

Shuffling Method



Shuffling Method에서는 직전의
난수값으로 그 다음 난수값의 범위
가 제한 되지 않는걸 볼수있다.
(Correalation not exists)

기본원리(확률변환)

확률변환

$$y = f(x)$$

$$|p(x)dx| = |p(y)dy|$$

$$p(x)|dx| = p(y)|dy|$$

$$p(y) = p(x)\left|\frac{dx}{dy}\right|$$

$$p(y)\,dy = dx$$

$$P(y) = x$$

균일한 난수 분포들을 불균일한 분포를 가지는 별의 질량으로 변환

대입

Salpeter IMF

$$N(m)dm = m^{-\alpha}\,\mathrm{dm}$$

$N(m)$은 별의 개수, dm은 질량간격을 나타낸다.

$p(x)$는 처음 발생한 난수들의 분포
(균일하기 때문에 $p(x) = 1$)

$p(y)$는 만들고자 하는 수들의 분포
(Salpeter's IMF를 사용하기 때문에
$p(y) = CM^{-2.35}$이다)
C는 normalization constant

실행

```
m=sp.symbols('m')
SI=m**-2.35
si=sp.integrate(SI,m)
C=1/(si.subs(m,100)-si.subs(m,1))
a=sp.integrate(C*SI,(m,1,m))
x=sp.symbols('x')
M=((1.00199925134584-x)/1.00199925134584)**(-1/1.35)
ML=[]
for i in rn.myran(100000):
    ML.append( ((1.00199925134584-i)/1.00199925134584)**(-1/1.35) )
```

m 변수로 지정

$$1 = \int_{m_1}^{m} C \cdot N(m')\, dm'$$

을 이용하여 정규화상수(C)를 구한다.-확률분포로 변환

확률변환을 통하여 x(난수), m(별의 질량)사이의 관계식을 구한다.

$$x = \int_1^m Cm^{-2.35}\, dn$$

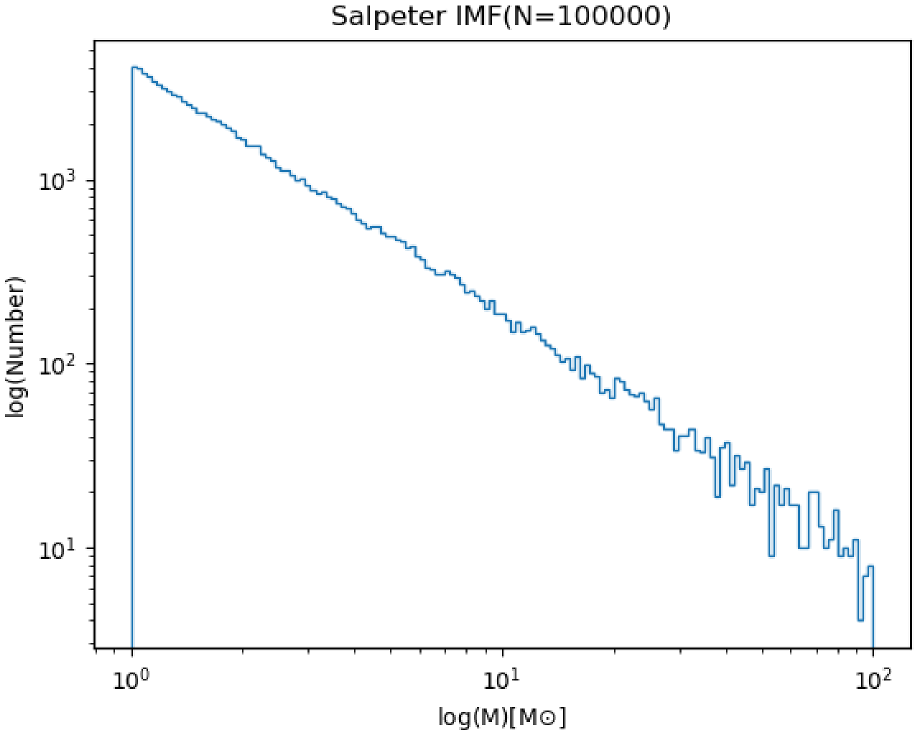
균일한 분포를 가지는 p(x)=1에서 0~x까지의 적분값 과 $p(y) = Cm^{-2.35}$에서 0~m까지의 적분값이 같다는것을 의미한다.(확률보전법칙)

```
rn.plt.hist(ML,bins=np.logspace(np.log10(1),np.log10(100),150),histtype='step')
rn.plt.title('Salpeter IMF(N=10000)')
rn.plt.xscale('log') #x축 logscale로 변경
rn.plt.yscale('log') #y축 logscale로 변경
rn.plt.xlabel("log(M)[M$\odot$]")
rn.plt.ylabel("log(Number)")
rn.plt.show()
```

각 축과 bin을 logscale로 변경

결과

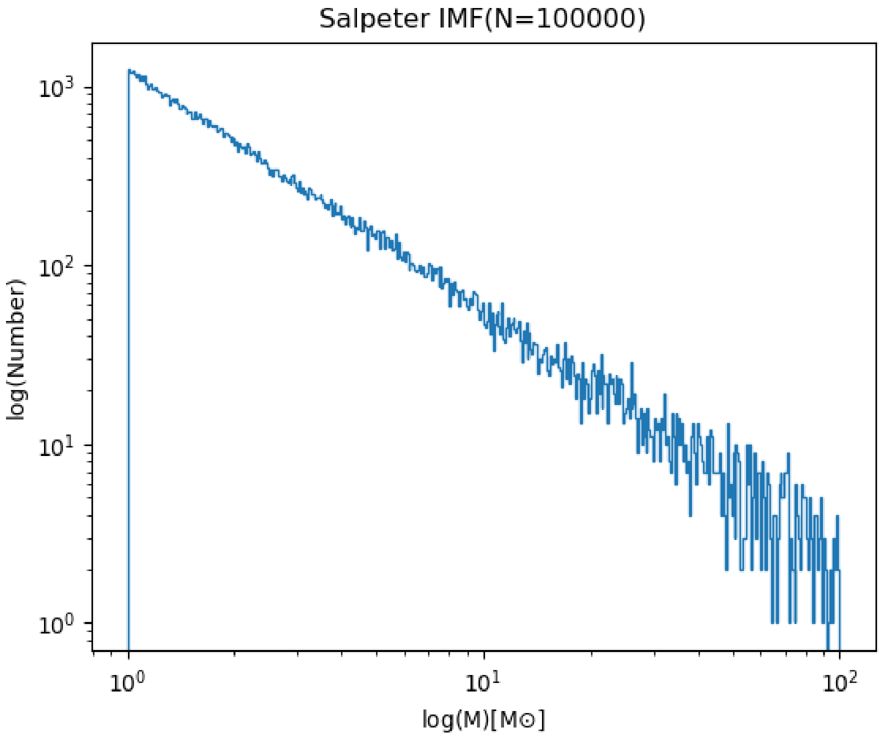
Bins=150



개수감소

질량 증가

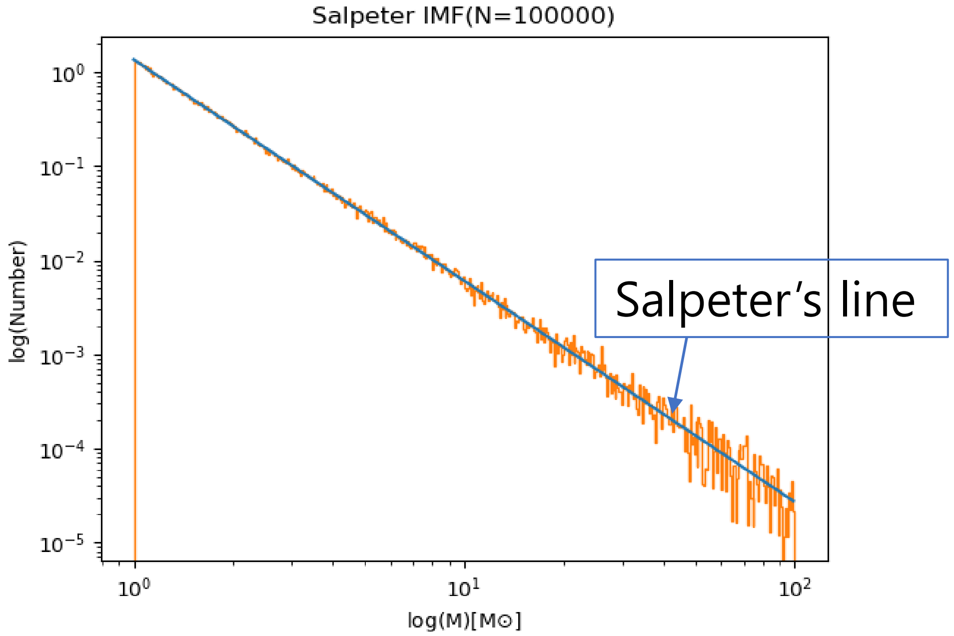
Bins=500



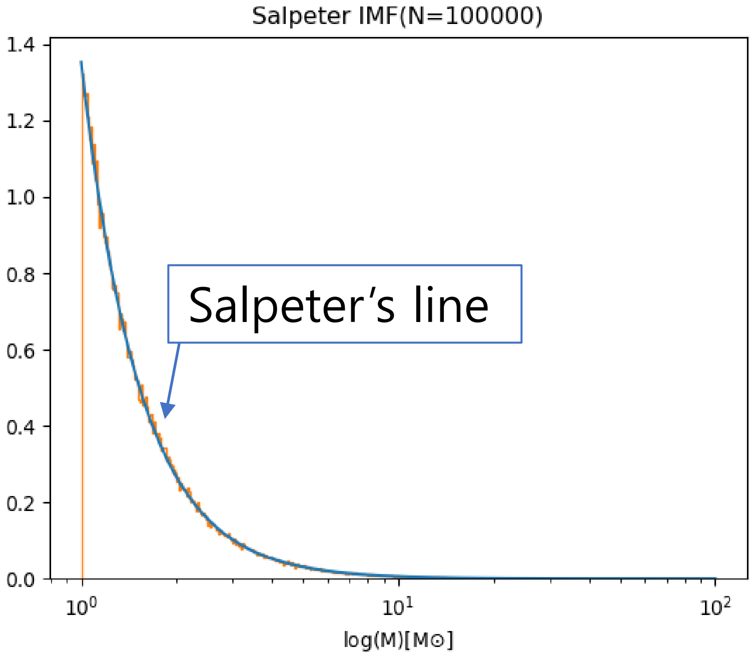
별의질량이 커질수록 별의 개수가 감소한다.

Salpeter's line과 비교

Y축 log



확률분포




```
x1=np.linspace(1,100,10000)
y1=C*x1**(-2.35)
rn.plt.plot(x1,y1)
```

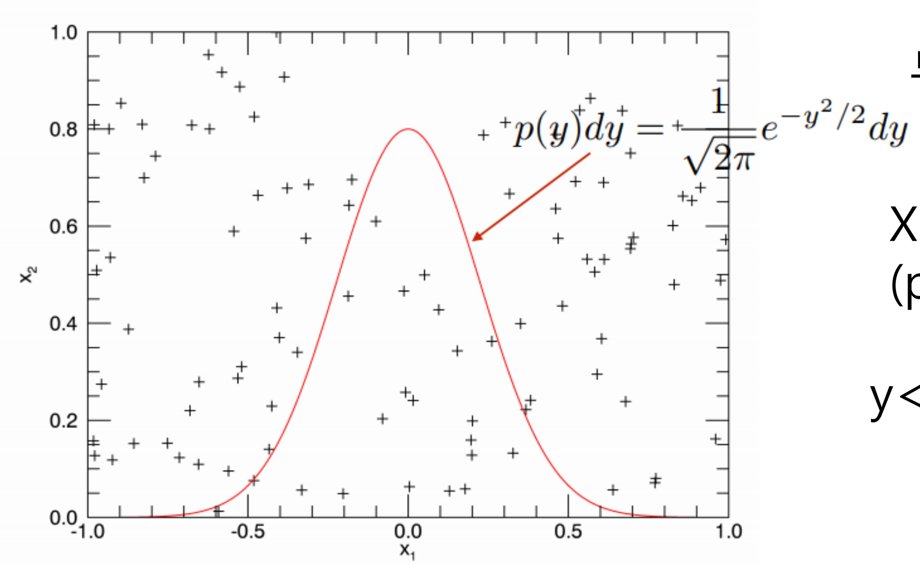
Salpeter's line의 확률분포를 plot한다

```
rn.plt.hist(ML,bins=np.logspace(np.log10(1),np.log10(100),500), histtype='step',density=True)
rn.plt.title('Salpeter IMF(N=100000)')
rn.plt.xscale('log') #x축 logscale로 변경
rn.plt.yscale('log') #y축 logscale로 변경
rn.plt.xlabel("log(M)[M$\odot$]")
rn.plt.ylabel("Number")
rn.plt.show()
```

Salepter's line 분포를 가지는 별의 질량의 확률분포를 히스토그램으로 표현

기본원리(Rejection Method)

확률변환 도중 적분 혹은 역함수를 구하기 어려운 함수의 분포를 가지는 경우 Rejection Method를 사용한다.



두개의 난수(x,y)를 발생

X값에 대한 Gaussian 함수 계산
(p라고 가정)

y<p인 경우 x저장

그래프 내부에 있는 값들만 저장한다.

Gaussian Function

$$f(x) = \boxed{\frac{1}{\sigma\sqrt{2\pi}}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

정규화 상수

의 분포를 가진다
(μ: 평균 ,σ: 분산)

실행(mean=0, min=-5,max=5, std=1인 가우스 분포 사용)

```python
while 1: #break문이 나올때까지 무한 반복
    T=rn.myran(2) #난수 2개 발생
    x1=10*T[0]-5 #-5~5의 범위를 가지는 난수로 변경
    x2=T[1]
    if x2< (1/mp.sqrt(2*mp.pi)) * mp.exp( (-x1**2)/2 ):
        G.append(x1)
        X1.append(x1)
        X2.append(x2)
    if len(G)>10000:
        break #발생한 난수 값이 10000개 넘을경우 while 반복문 중지
```

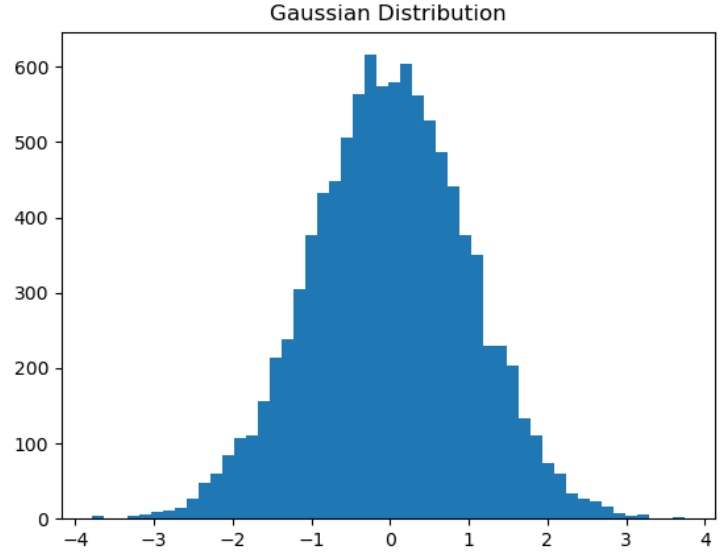
난수 2개 발생(x1,x2)후 x1은 -5~5 범위로 변환

$f(x1) = \frac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{-\frac{(\mathrm{x1}-\mu)^2}{2\sigma^2}}$가 x2보다 작을 경우 x2를 저장한다.

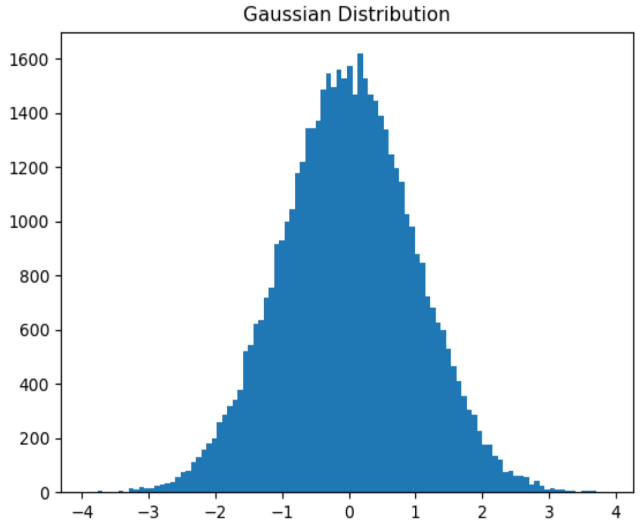
발생한 난수가 10000개(Gaussian분포내) 일 경우 while문 중단

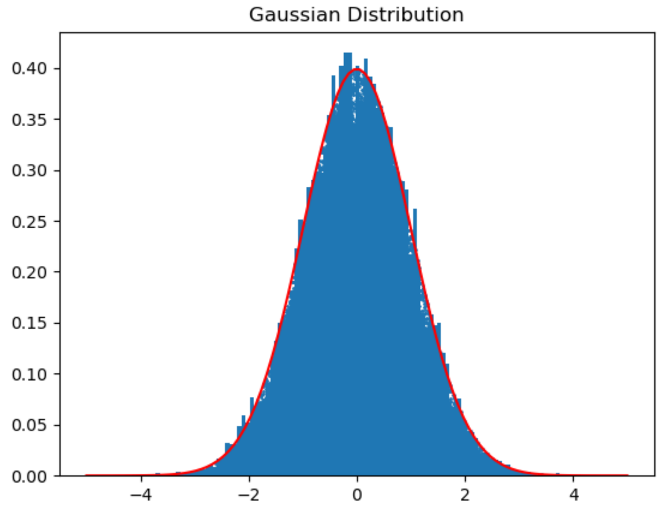
결과

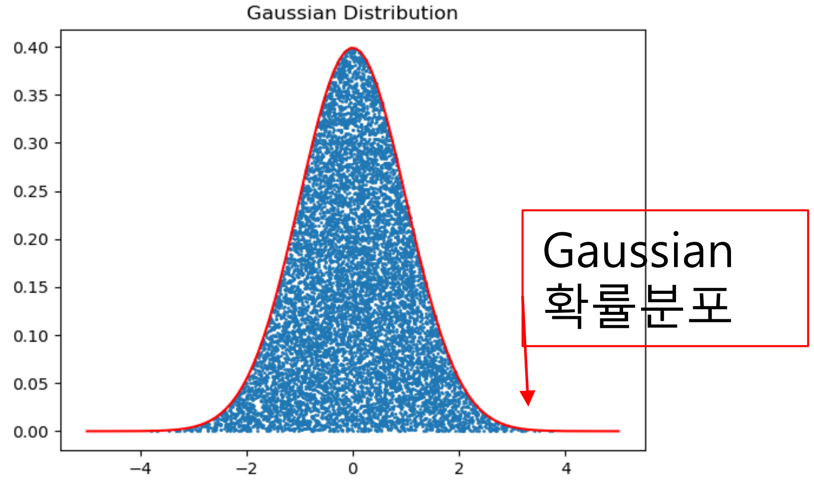
N=10000



N=50000







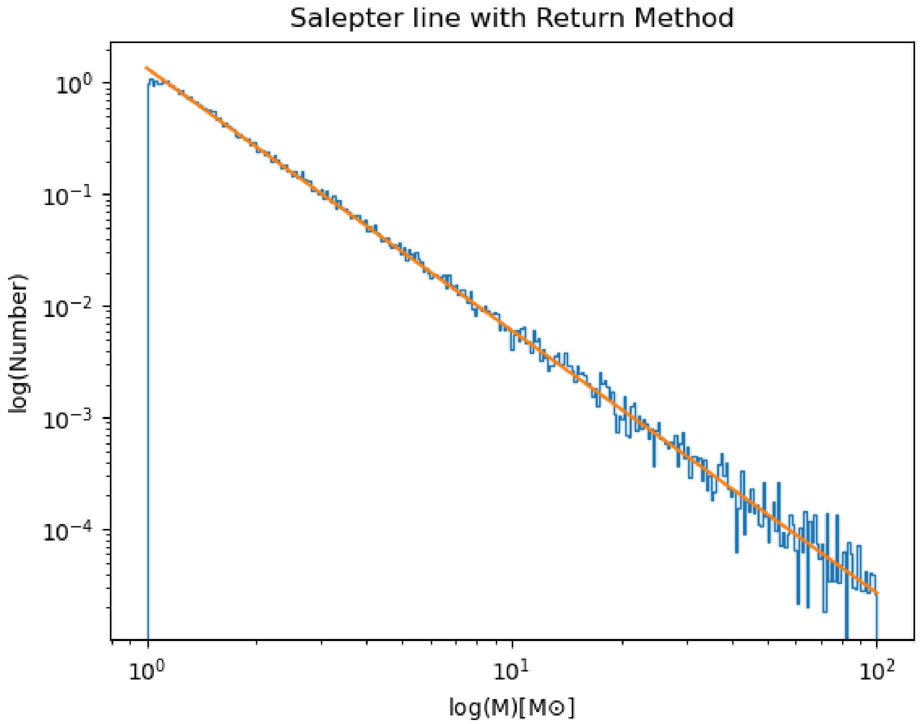
변수가 많을수록 정규분포의 형태를 가진다.

토의사항

과제2에서 한 Salpeter's line의 분포를 Rejection Method를 사용하여 난수 발생

($C * m^{-2.35}$의 분포를 가지는 난수 발생) C=1.35269898931688 :정규화상수

```
while 1: #break문이 나올때까지 무한 반복
    T=rn.myran(2) #난수 2개 발생
    x1=T[0]
    x2=99*T[1]+1 #1~100까지의 난수로 변경(별의 질량)
    if x1<1.35269898931688*x2**(-2.35): #Salpeter라인의 확률분포
        G.append(x1)
        X2.append(x2)
    if len(X2)>50000:
        break #발생한 난수 값이 50000개 넘을경우 while 반복문 중지
```



Salpeter's line의 분포와 유사하게
난수분포가 형성이 된다

Reference

·How to have logarithmic bins in a Python histogram, stackoverflow, 2013/7/13

https://stackoverflow.com/questions/6855710/how-to-have-logarithmic-bins-in-a-python-histogram

·Integrate with sympy, stackoverflow,2020/10/27

https://stackoverflow.com/questions/64556783/integrate-with-sympy