Assignment 7

1.

we can simply plot a 3D scatter plot of the triplets in rand_points.txt.

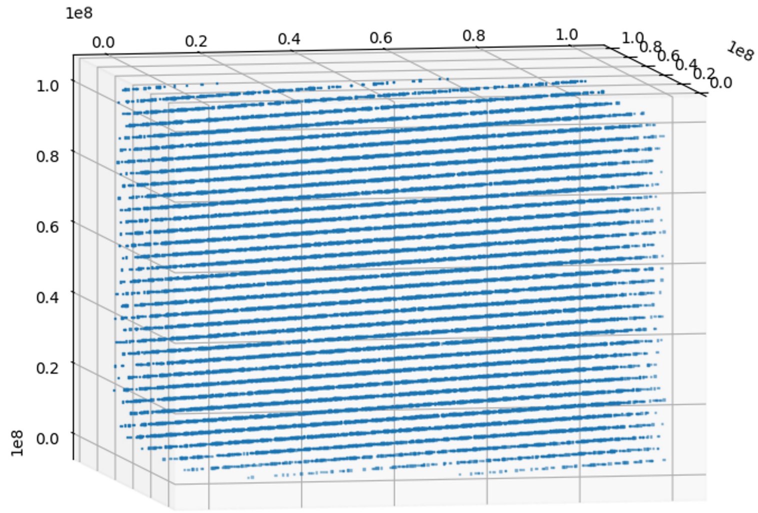
First of all, we need to set the projection as orthogonal to better see this effect. we also need to make the scatter point marker small.

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

rand_trip = np.loadtxt('rand_points.txt')
fig = plt.figure()
ax = plt.axes(projection='3d')

xdata = rand_trip[:,0]
ydata = rand_trip[:,1]
zdata = rand_trip[:,2]

ax.set_proj_type('ortho')
ax.scatter(xdata, ydata, zdata, s = 1, marker=',')
plt.show()
```

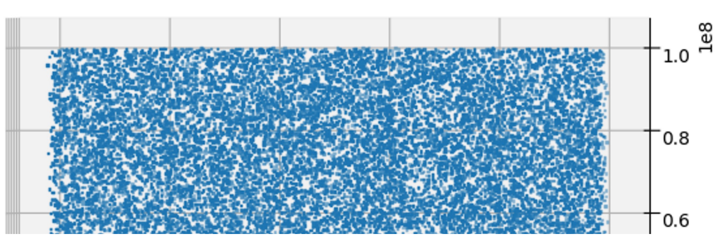


we can also try this out
with numpy.random.randint

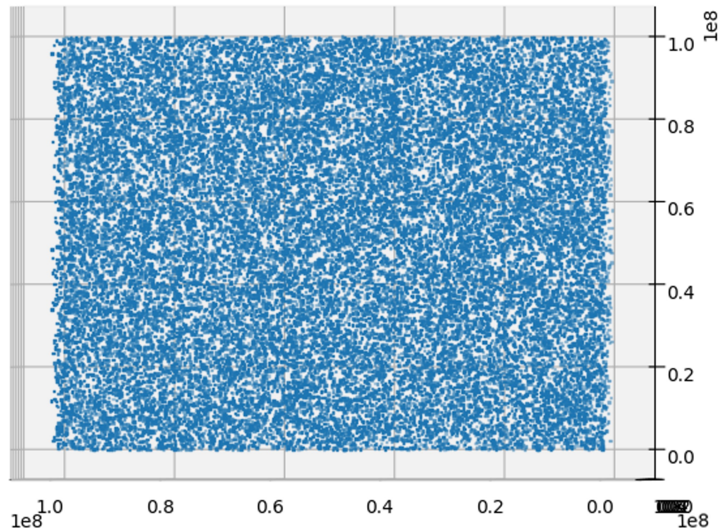
```
rand_trip_python = np.random.randint(1e8,size=(len(xdata),3))
xdata_p = rand_trip_python[:,0]
ydata_p = rand_trip_python[:,1]
zdata_p = rand_trip_python[:,2]

fig = plt.figure()
ax = plt.axes(projection='3d')
ax.set_proj_type('ortho')
ax.scatter(xdata_p, ydata_p, zdata_p, s = 1, marker=',')
plt.show()
```

By rotating the 3D figure around
I am unable to see these planes
as in C. An example perspective
looks like this.



looks like this.



2. I stole most of the code from gauss_rejection_class.py from last year's class repo.

First, we need to use a power law since exp < power-law

Then, recall that we can get power law distribution by using

$$s = x^{(1/(1-a))}$$

where x is uniform from 0 to 1.

Can see this by:

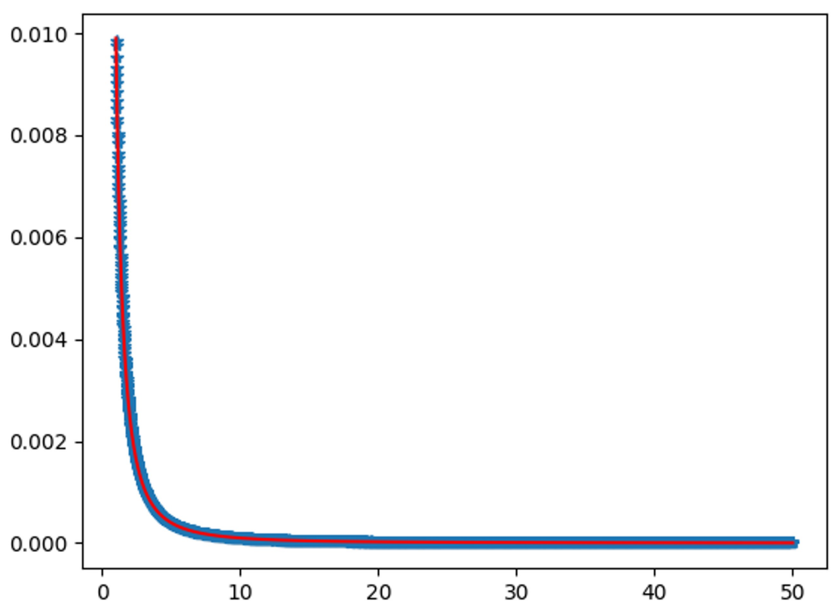
```
import numpy as np
```

```python
from matplotlib import pyplot as plt
def power_law(n):
    k=2
    q = np.random.rand(n)**(1/(1-k))
    return q

n=10000000
t=power_law(n)

bins=np.linspace(1,50,5001)
aa,bb=np.histogram(t,bins)
aa=aa/aa.sum()
cents=0.5*(bins[1:]+bins[:-1])
pred=cents**-2
pred=pred/pred.sum()

plt.clf()
plt.plot(cents,aa,'*')
plt.plot(cents,pred,'r')
plt.savefig('power_law.png')
plt.show()
```



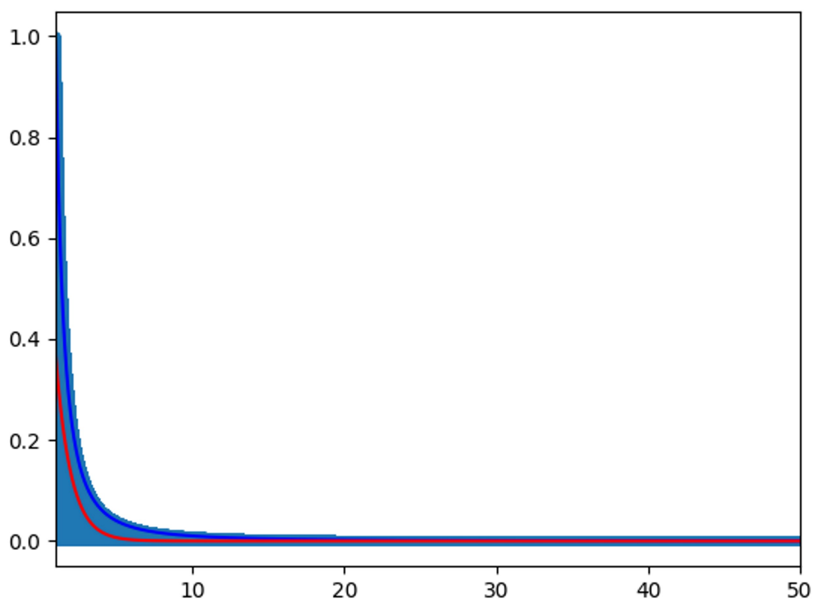
Note here I cut the distribution for greater than 1.

Then as discussed in class, we can draw a random sample from power law, and see if it falls below

law, and see if it falls below
our exponential.

```
mypower=cents**-2
myexp=np.exp(-cents)
y=t**-2*np.random.rand(n)

plt.clf()
plt.plot(t,y,'.')
plt.plot(cents,mypower,'b')
plt.plot(cents,myexp,'r')
plt.xlim(1,50)
plt.savefig('power_vs_exp.png')
plt.show()
```



we want the points that falls below
the red curve.

we can then plot these points that
we use into histogram and compare
with expectation.

we can get the acceptance rate

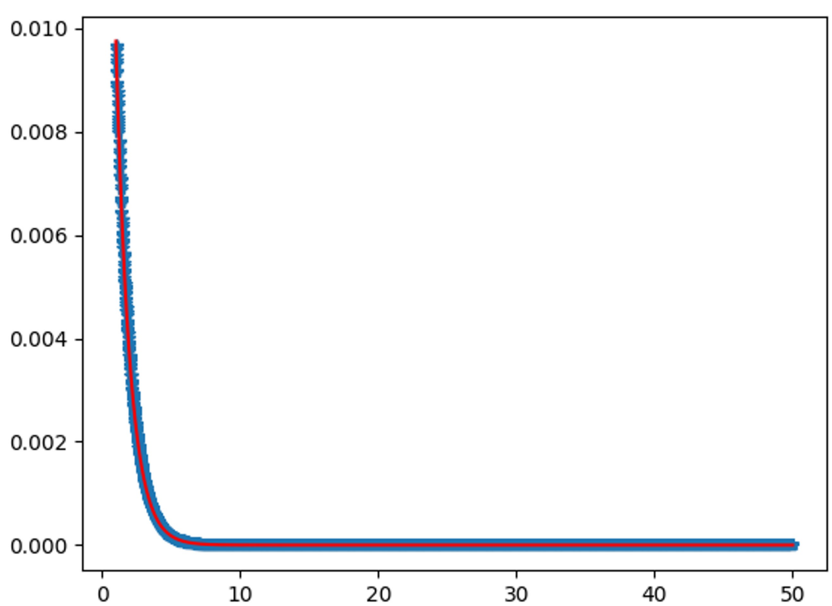
we can get the acceptance rate simply by seeing how many points we end up using.

```
accept=y<np.exp(-t)
t_use=t[accept]

accept_rate = len(t_use)/len(t)
print('accept rate is:', accept_rate)

aa,bb=np.histogram(t_use,bins)
aa=aa/aa.sum()
pred=np.exp(-cents)
pred=pred/pred.sum()

plt.clf()
plt.plot(cents,aa,'*')
plt.plot(cents,pred,'r')
plt.savefig('exp_from_power.png')
plt.show()
```



accept rate is: 0.3679607

we can see that we indeed got an exp from power law.

an exp from power law.

3. we need $0 < u < P(\frac{v}{u})$

we have

$$u < \left[\exp\left(-\frac{v}{u}\right)\right]^{\frac{1}{2}}$$

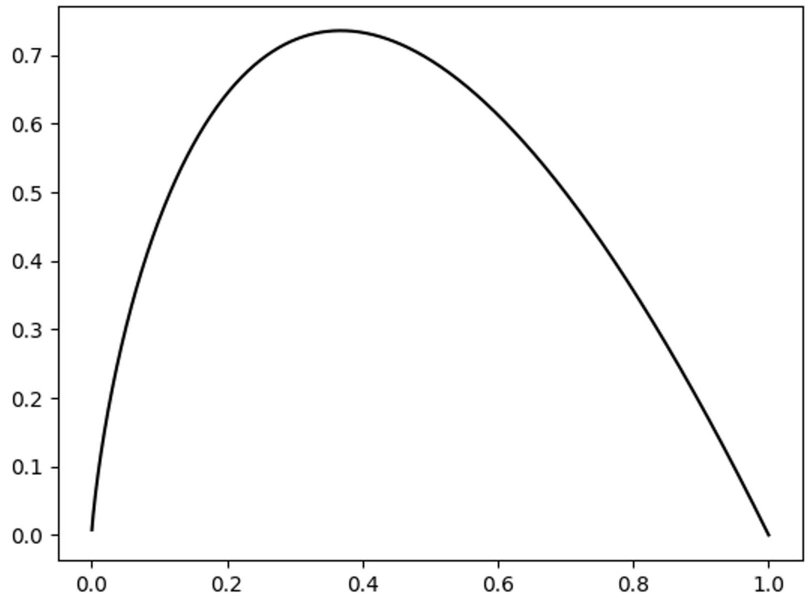
$$\Rightarrow \log(u) < -\frac{v}{2u}$$

$$v < -2u\log(u)$$

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

u=np.linspace(0,1,2001)
u=u[1:]
v=-2*u*np.log(u)
print('max v is ',v.max())

plt.clf()
plt.plot(u,v,'k')
plt.savefig('exp_region.png')
plt.show()
```



max v is 0.7357588428385197

The we see that we can some what confine v to be below 0.75

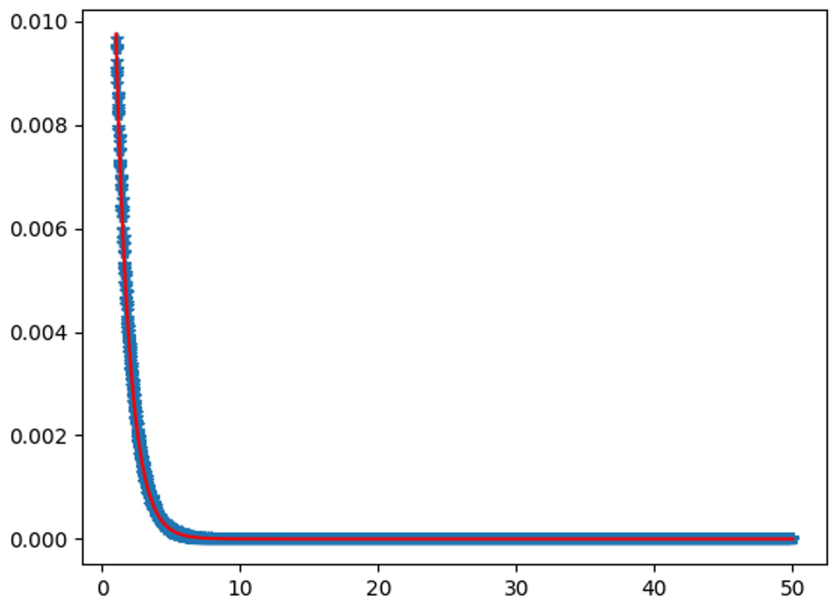
we can then plot the histogram and see the accept ratio :

```
N=1000000
u=np.random.rand(N)
#.75 seems to be max value of v
v=np.random.rand(N)*0.75
r=v/u

accept=u<np.exp(-0.5*r)
exp=r[accept]
accept_rate = len(exp)/len®
print('accept rate is:', accept_rate)

bins=np.linspace(1,50,5001)
cents=0.5*(bins[1:]+bins[:-1])
aa,bb=np.histogram(exp,bins)
aa=aa/aa.sum()
pred=np.exp(-cents)
pred=pred/pred.sum()
```

```
plt.clf()
plt.plot(cents,aa,'*')
plt.plot(cents,pred,'r')
plt.savefig('exp_ratio_uniform.png')
plt.show()
```



accept rate is: 0.665835

we can see that we indeed get a very good exp distribution, and our accept rate nearly doubled comparing to Q2.