1. For the points generated in C, by rotating the plot (1) I could find angle such that the points lay in parallel planes. I didn't see the same effect with numpy's random.

```
x,y,z = np.genfromtxt('rand_points.txt', unpack=True)
ax = plt.subplot(121,projection='3d')
ax.plot(x,y,z,'.')
ax.set_title('Random points C')

x,y,z = np.random.rand(3,10000)
ax = plt.subplot(122,projection='3d')
ax.plot(x,y,z,'.')
ax.set_title('Random points Python')
```

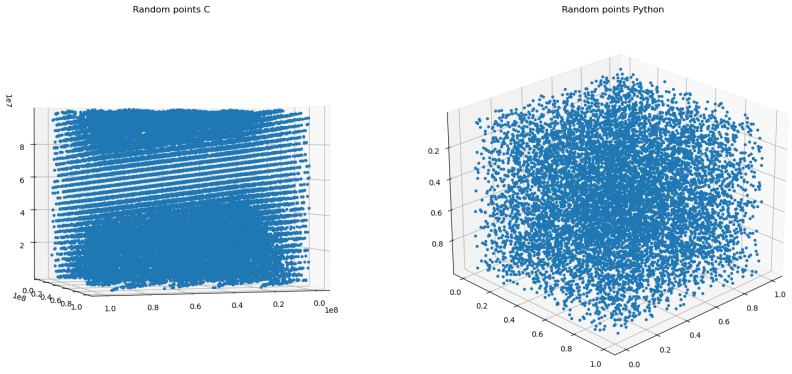


Figure 1: Correlated random numbers generated by C lie in planes.

I wasn't able to use the C Library on my windows machine.

2. Power laws and Lorentzians can be used to bound the exponential distribution, but a Gaussian decays faster than an exponential, so it's not larger for long enough, unless you shift it up, but that's less efficient. See Figure 2.

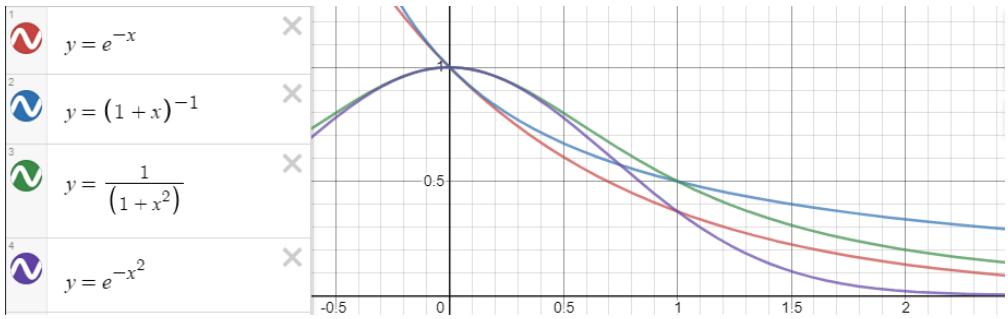


Figure 2: Bounding an exponential (red) by a shifted power law (blue), a Lorentzian (green) and a Gaussian (purple). The Lorenzian fits closest to the exponential without going under.

In 'rejection.py', the following functions are used to generate Lorentzian-distributed samples, and accept or reject with probability based on the value. The histogrammed samples are shown in Figure 3.

```
def lorentz_dev(n):
    '''Random variable with lorentzian distribution'''
    x=np.random.rand(n)
    return np.tan(np.pi*(x/2))

def exp_from_lorentz(x):
    '''Reject values by probability of target distribution'''
    accept_prob=np.exp(-x)/(1/(1+1*x**2))
    assert(np.max(accept_prob)<=1)
    accept= np.random.rand(len(accept_prob))<accept_prob
    return x[accept]
```

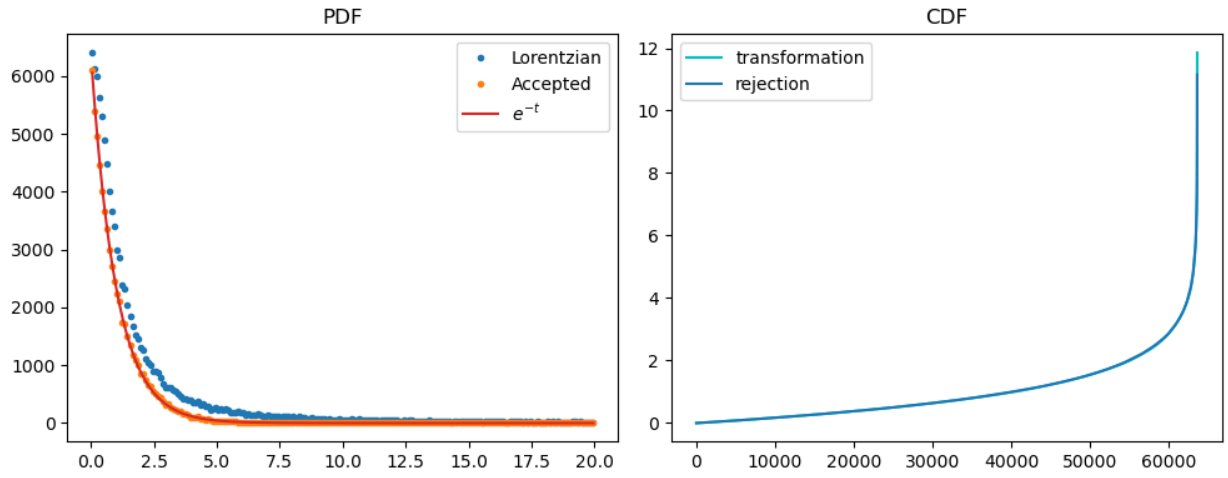


Figure 3: Left: Histogram of 10000 samples of Lorentzian distribution, and exponentially distributed expected values. Exponential curve scaled to the maximum of the accepted value distribution. Right: Cumulative distribution of generated samples compared to analytical result found by transformation.

The probability of acceptance in my rejection method is 64%. (This can go up to 77% by choosing a slightly more tightly fitting Lorenztian, though to represent that properly I would have to scale the original distribution accordingly.)

3. See Figure 4. We want $0 < u < \sqrt{P(v/u)} = e^{-\frac{v}{2u}}$, so $v < -2u \ln u$, which is maximal at $v = \frac{2}{e}$. We only want positive values of $\frac{v}{u}$, so $v > 0$. This increases the efficiency from 25% with $0 < v < 1$, to 34%.

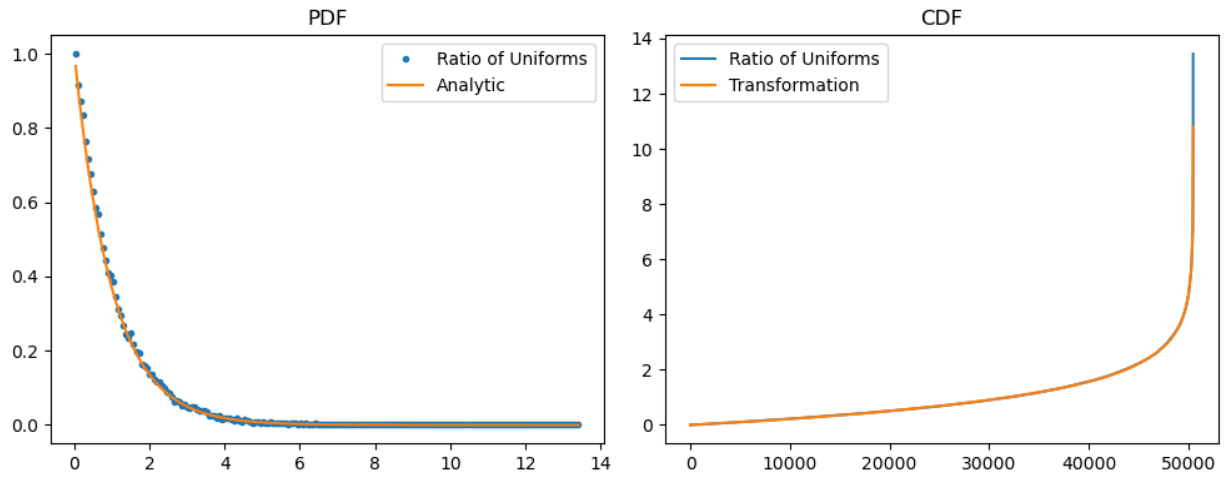


Figure 4: Left: Normalized histogram of values generated by ratio of uniforms, compared to analytical exponential. Right: Cumulative density function of generated samples (orange) and analytical result found by transformation (blue).