PHYS 512 Assignment 7

Michelle Lam, SN: 261005326

November 20, 2021

Problem 1

I wasn't able to run Jon's code from 2020, the library that's being referenced, my laptop can't seem to download the ctypes library (so I just loaded Jon's output file). Hence, I couldn't run the libc.so library either. Refer to ps7_Problem1.py for code.

I did a 2D plot with x+y vs. z, to compare with python, i used the 'random' library to generate numbers from 0 to 1e8 for x,y,z. And comparing the 2 figures below we see that the random numbers generated from the ctypes library exhibits a stripy pattern whereas the numpy has less of an obvious pattern. Therefore, it seems like the ctypes library is biased and not as random since it has uniform spacing giving it a stripe pattern.

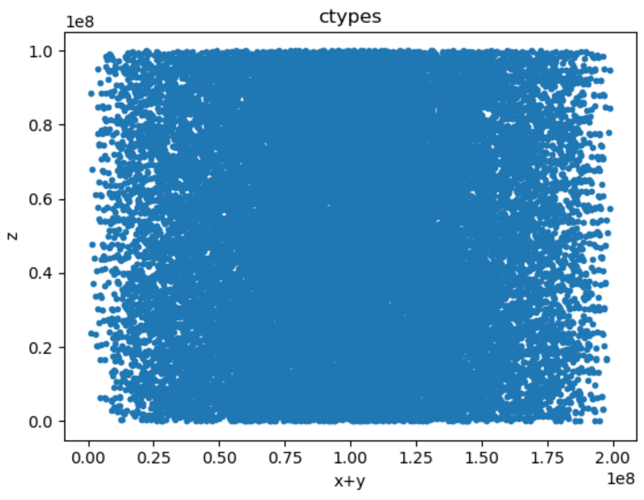


Figure 1: random numbers from ctypes library

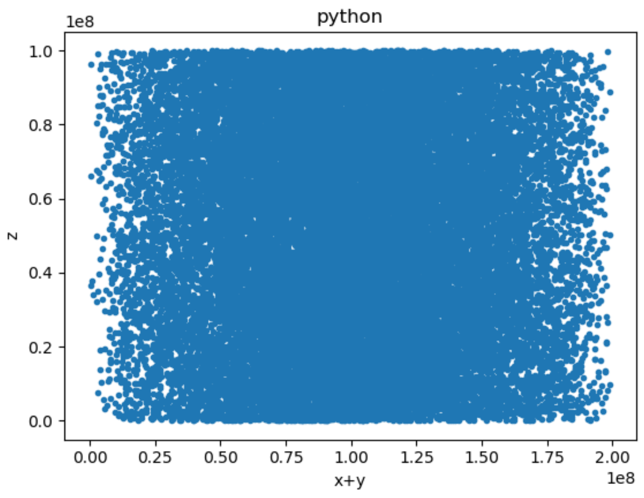


Figure 2: random numbers from python, by importing library 'random'

Problem 2

Refer to ps7_Problem2.py for code. Using a lorentzian as the bounding distribution for the rejection method and throwing away values that sit outside our exponential, arbitrarily chosen to be e^{-x}. A gaussian wouldn't work because it would eventually cross the exponential, instead of being above the exponential the entire time.

We can see that the expected and sampled points fall along the curve, and were able to achieve this with about 82% efficiency. I was able to improve the efficiency by not scaling the lorentzian but making sure the exponential wasn't ever above the lorentzian.

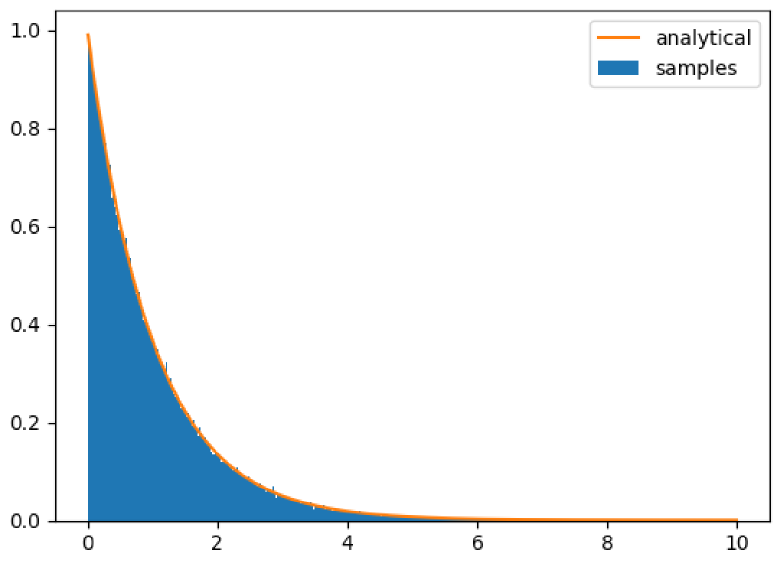
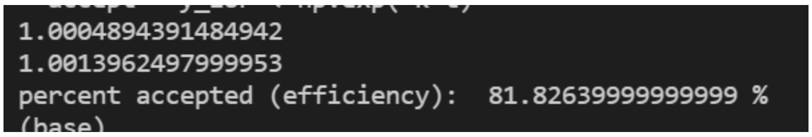


Figure 3: Rejection method using lorentzian to sample exponential



```
1.0004894391484942
1.0013962497999953
percent accepted (efficiency):  81.82639999999999 %
(base)
```

Figure 4: Rejection method efficiency

Problem 3

Refer to ps7_Problem3.py for code. Now using ratio-of-transforms, using the approximate method, where our PDF $= e^{-r}$,

$$u = \sqrt{e^{-r}}$$

,

$$v = ru$$

.

We take the min and max of u and v, and sample the random numbers within that range. Take ratio v/u and we get an efficiency of 68%

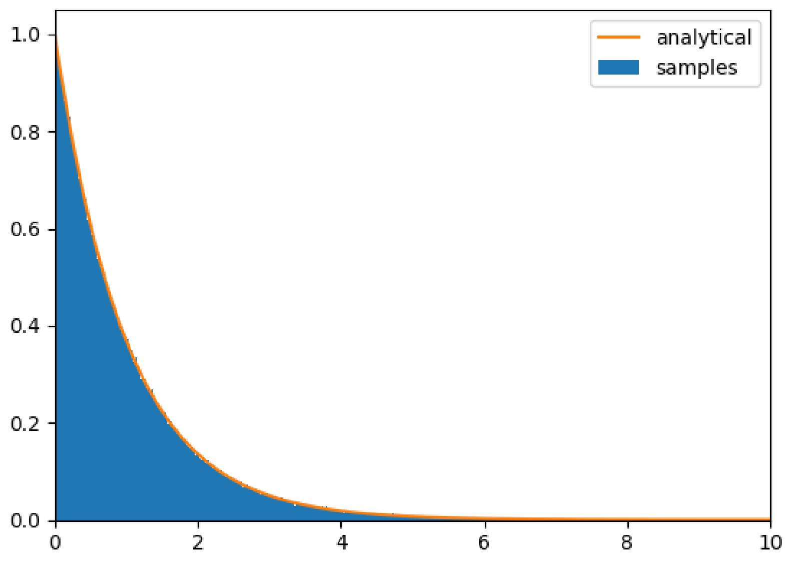
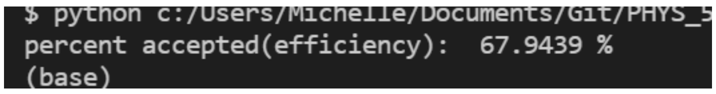


Figure 5: Ratio of uniforms method - exponential deviates



```
$ python c:/Users/Michelle/Documents/Git/PHYS_5
percent accepted(efficiency):  67.9439 %
(base)
```

Figure 6: Ratio of uniforms method efficiency