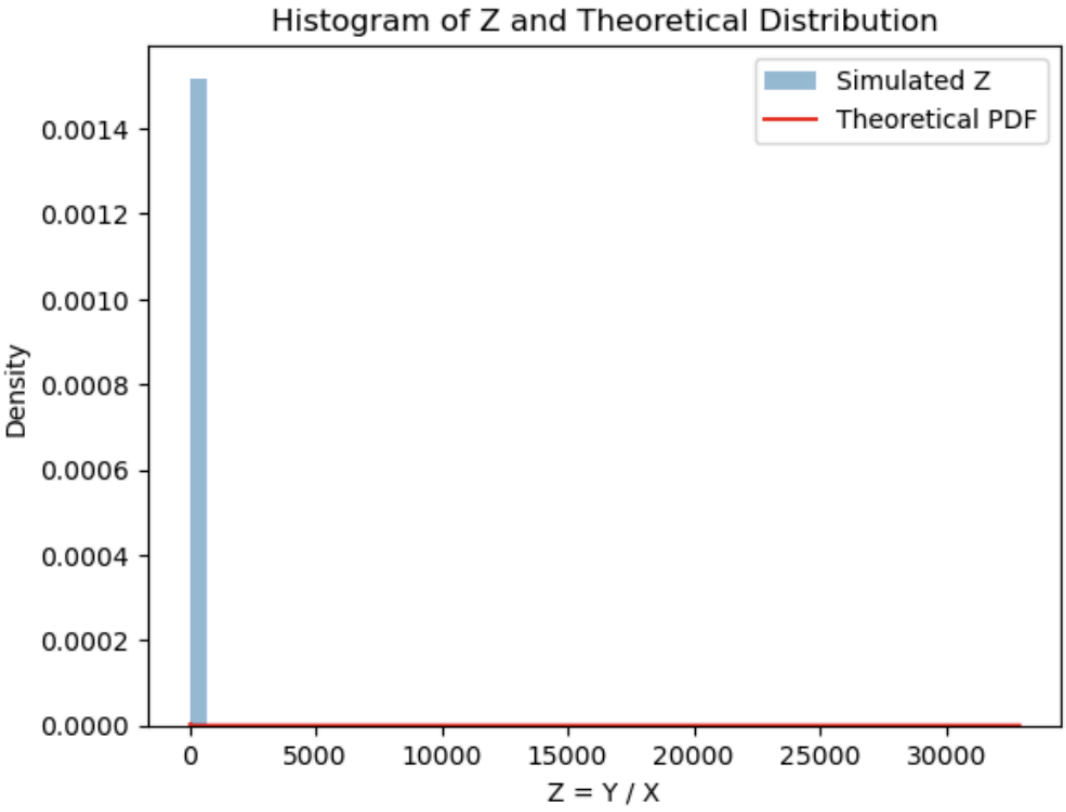
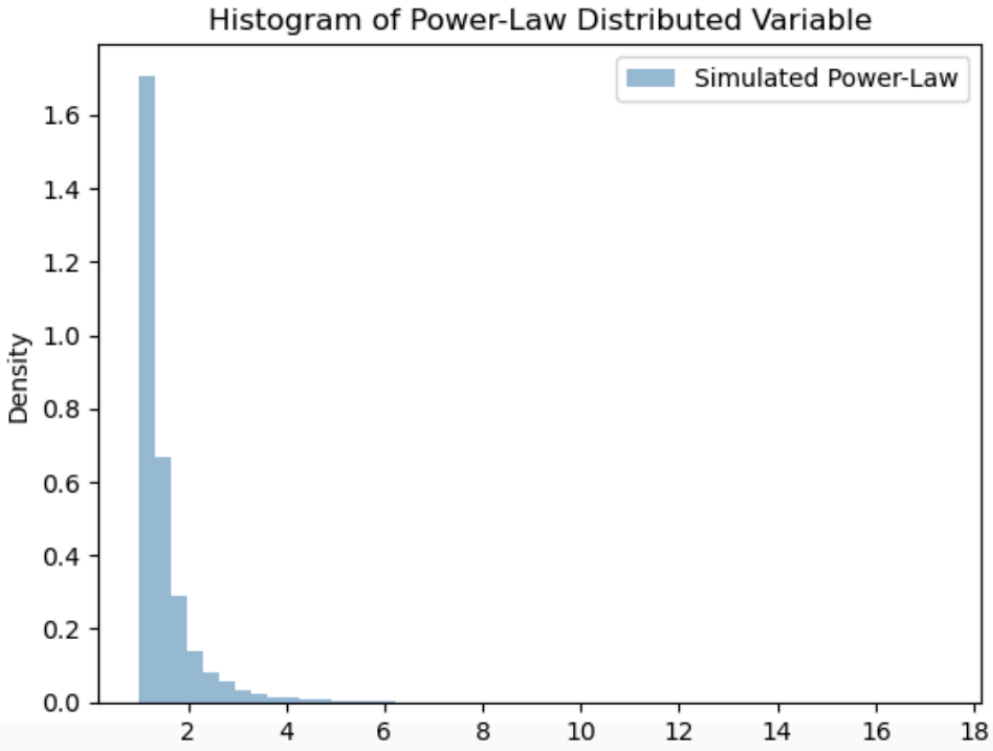
```python
import numpy as np
import matplotlib.pyplot as plt

# Parameters for the simulation
lambda_1 = 1
lambda_2 = 2
num_samples = 10000

# Simulate samples of X and Y
X_samples = np.random.exponential(scale=1/lambda_1, size=num_samples)
Y_samples = np.random.exponential(scale=1/lambda_2, size=num_samples)

# Compute Z = Y / X
Z_samples = Y_samples / X_samples

# Plot histogram of Z
plt.hist(Z_samples, bins=50, density=True, alpha=0.5, label='Simulated Z')

f_Z = lambda z: 3*z**(-4)
z_values = np.linspace(start=0, stop=np.max(Z_samples), num=1000)
theoretical_pdf = np.array([f_Z(z) for z in z_values])
plt.plot(z_values, theoretical_pdf, label='Theoretical PDF', color='red')

plt.xlabel('Z = Y / X')
plt.ylabel('Density')
plt.title('Histogram of Z and Theoretical Distribution')
plt.legend()
plt.show()
```

```
/tmp/ipykernel_146/3310432535.py:19: RuntimeWarning: divide by zero encountered in d
```


Histogram of Z and Theoretical Distribution
Simulated Z
Theoretical PDF
Density
0.0014
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
0
5000
10000
15000
20000
25000
30000
Z = Y / X

4]:
```python
# Number of samples
num_samples = 10000

# Generate uniform samples
U_samples = np.random.uniform(low=0, high=1, size=num_samples)

# Apply inverse CDF: F^{-1}(u) = u^{-1/3}
X_samples = U_samples**(-1/3)

# Plot histogram of X
plt.hist(X_samples, bins=50, density=True, alpha=0.5, label='Simulated Power-Law')

plt.xlabel('X')
plt.ylabel('Density')
plt.title('Histogram of Power-Law Distributed Variable')
plt.legend()
plt.show()
```



7. One option is to simulate n bernoulli's, as follows.
count = 0
for(i from 1 to n)
  Generate $U \sim U(0,1)$
  If $U < p$, count = count + 1
return count

~~Another option is to use the normal approximation~~
~~Calculate~~
~~Calculate sigma = sqrt(n · p · (1-p))~~
~~Generate~~

Another option is to use the Box-Muller method
Generate U_1 and $U_2 \sim U(0,1)$ independently
Calculate $Z_0 = \text{sqrt}(-2 \cdot \log(U_1)) \cdot \cos(2 \cdot \pi \cdot U_2)$
Return Z_0

~~This must be used last~~
This is used instead of the inverse transform method because the binomial coefficients become too large as n increases. However, this is an approximation and may introduce error.

~~Ex~~

9. a. Dish 2 is the most likely dish in the long run because it is the most likely to be selected again (or tied) for all initial states

b. $P_{t+1}(1) = 0.1P_t(2) + 0.2P_t(3)$

$P_{t+1}(2) = 0.3P_t(1) + 0.7P_t(2) + 0.8P_t(3)$

$P_{t+1}(3) = 0.5P_t(1) + 0.2P_t(2)$

c. $P(t+1) = \begin{bmatrix} 0 & 0.1 & 0.2 \\ 0.3 & 0.7 & 0.8 \\ 0.5 & 0.2 & 0 \end{bmatrix} P(t)$

Homework 2

Alex Ojemann
Markov Processes
2/1/24

1. $M_X(s) = (1-p+pe^s)^n$
$M_Y(s) = (1-p+pe^s)^m$
Since X and Y are independent, the MGF of their sum is the product of their MGFs
$M_{X+Y}(s) = (1-p+pe^s)^m \cdot (1-p+pe^s)^n = (1-p+pe^s)^{n+m}$
Thus, $X+Y \sim$ binomial$(m+n, p)$. This makes sense because they represent the number of bernoulli trials that succeed with the same probability of success, so adding them together effectively just increases the number of trials.

2. $M_X(s) = e^{\sigma^2 s^2/2}$
$\frac{d}{ds}\left(e^{\sigma^2 s^2/2}\right) = e^{\sigma^2 s^2/2} \cdot \frac{d}{ds}\left(\frac{\sigma^2 s^2}{2}\right) = \sigma^2 s\, e^{\sigma^2 s^2/2}$
Taking successive derivatives will introduce factors of $\sigma^2 s$ multiplied by the previous derivative
$\frac{d^2}{ds^2} = \sigma^2 e^{\sigma^2 s^2/2}(1+\sigma^2 s^2)$
All odd derivatives end up with a factor of s, so all odd moments are 0. The even moments can be expressed as:
$E[X^n] = \sigma^n (n-1)!_{odd}$
Where $!_{odd}$ is the product of all odd integers up to and including $n-1$

3. Generate $U \sim U(0,1)$
If $U < 0.9$
return 3
else if $u < 0.95$
return 2
else if $u < 0.99$
return 4
else
return 1
This algorithm is efficient because it orders the if statements by liklihood so that the most likely scenario is that it only has to execute one if statement

4. Generate $U \sim U(0,1)$
Solve $3x^2 - 2x^3$ (integral of PDF) $= U$. Use the solution where $0 \leq x \leq 1$
return x

5 and 6: Code on first page of pdf

7. $\frac{e^y}{1+e^y} = u$ $\quad \lambda e^{-\lambda x} = u'$
$y = \ln(u/(1-u))$ $\quad u' = -\frac{1}{\lambda} \ln(1 - u')$

Algorithm:
Generate $U \sim U(0,1)$
Calculate $y = \ln(u/(1-u))$
Generate $U' = U(0,1)$
Calculate $x = -1/y^2 \cdot \ln(1-u')$
return x