project2

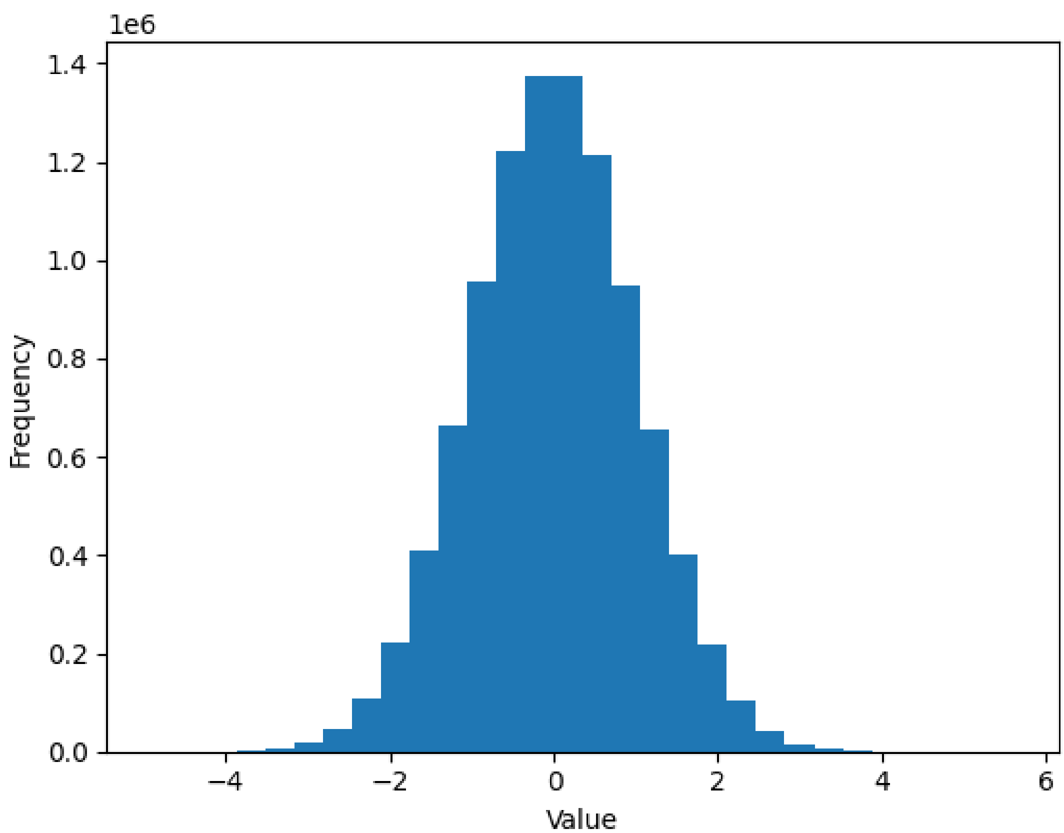
October 5, 2023

```
[ ]: import numpy as np
     import matplotlib.pyplot as plt

     import random
     from IPython.display import display, Math

     '''
     Let X+iY be a complex signal and its magnitude is given by Z=LaTeX:␣
      ↪\sqrt{X^2+Y^2}, and phase LaTeX: \theta=\tan^{-1}\left(\frac{Y}{X}\right) if␣
      ↪XLaTeX: \ge0 and phase LaTeX: \theta=\tan^{-1}\left(\frac{Y}{X}\right)+\pi␣
      ↪if X<0.

     X~N(0,1) and Y~N(0,1).

     Use the MATLAB or Python functions to create a Gaussian distributed random␣
      ↪value of X. Repeat this procedure and form a new random value of Y. Finally,␣
      ↪form a random value of Z and LaTeX: \theta, respectively. Repeat this␣
      ↪procedure many times to create a large number of realizations of Z and LaTeX:
      ↪ \theta. Using these samples, estimate and plot the probability density␣
      ↪functions of Z and LaTeX: \theta, respectively. Find analytical␣
      ↪distributions among what we learned in the lectures that seem to fit your␣
      ↪estimated PDFs.

     To clarify, you need to submit your code, plots of sample distributions and␣
      ↪analytical distributions (as well as names and parameters of the analytical␣
      ↪distributions).



     Note: X~N(0,1) denotes random variable X follows a Gaussian distribution with␣
      ↪mean 0 and variance 1.

               "Histogram" would be needed.

     '''

     print(Math(r"\left \sqrt{X^{2}+Y^{2}} \right"))
```

```python
print(Math(r"\theta=\tan^{-1}\left(\frac{Y}{X}\right)"))
```

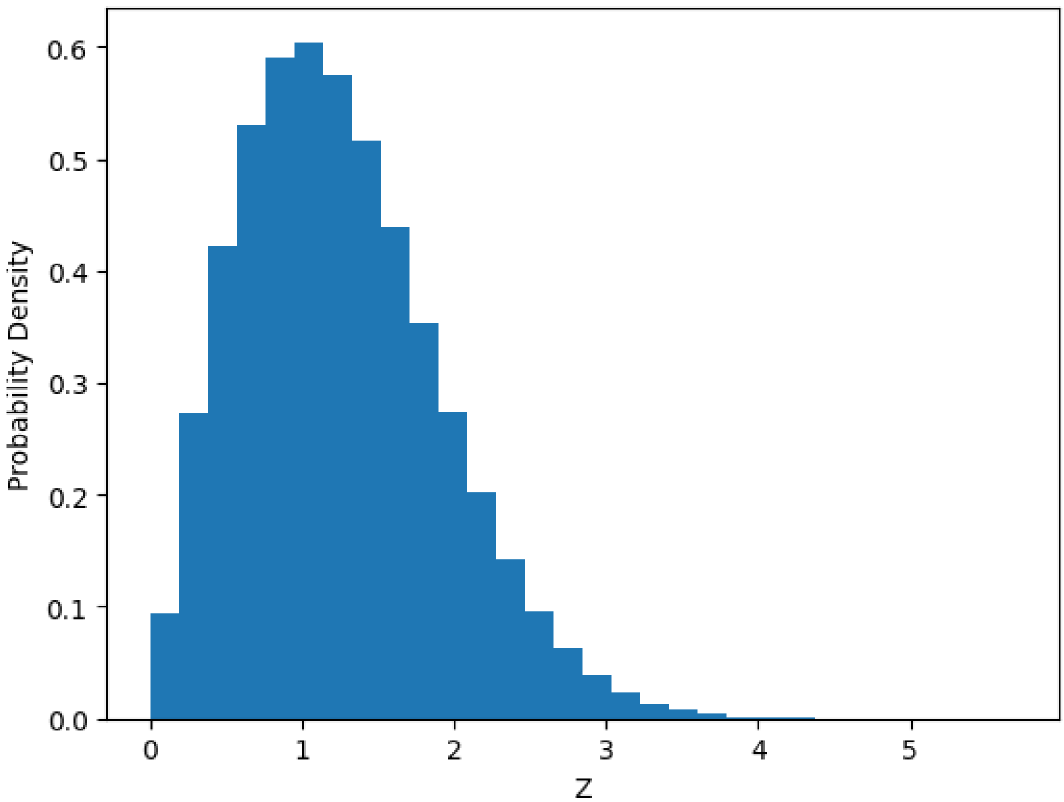
```
<IPython.core.display.Math object>
<IPython.core.display.Math object>
```

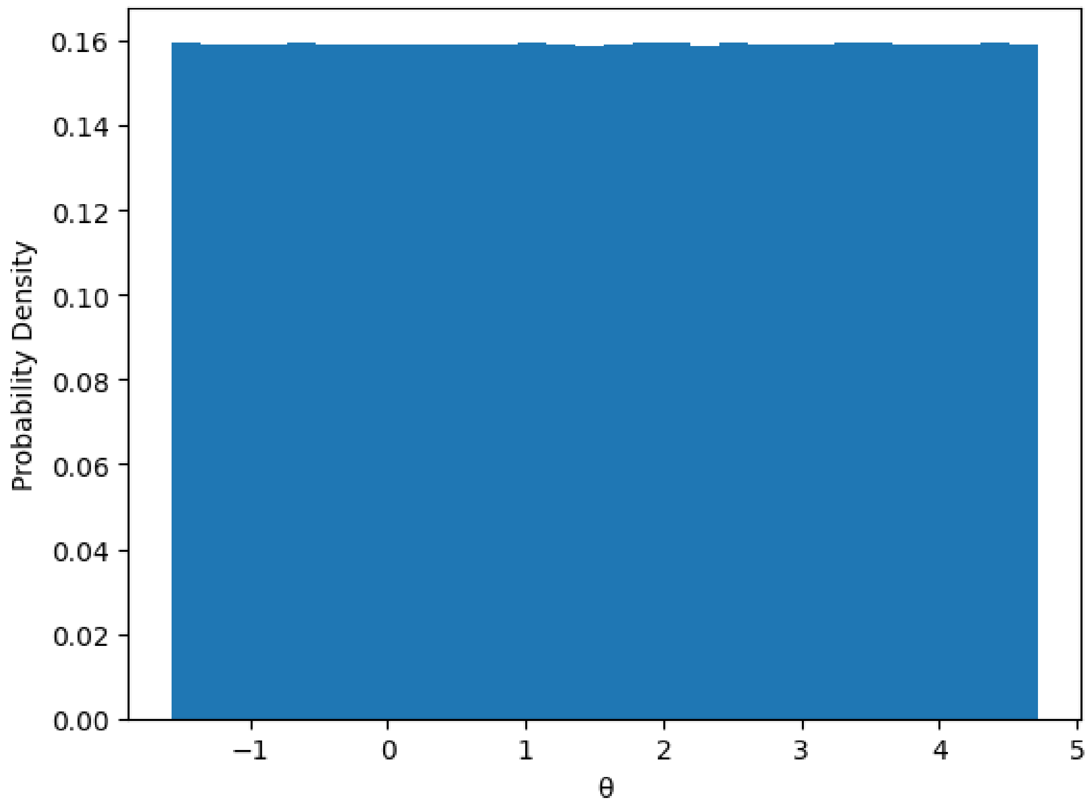
[ ]:
```python
import matplotlib.pyplot as plt
import numpy as np

# Generate 1000 random values from a Gaussian distribution with mean 0 and␣
↪standard deviation 1
random_values = np.random.normal(0, 1, 10000000)

# Plot a histogram of the random values
plt.hist(random_values, bins=30)
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()
```



[ ]:
```python
# Define the number of realizations
num_realizations = 10000000
```

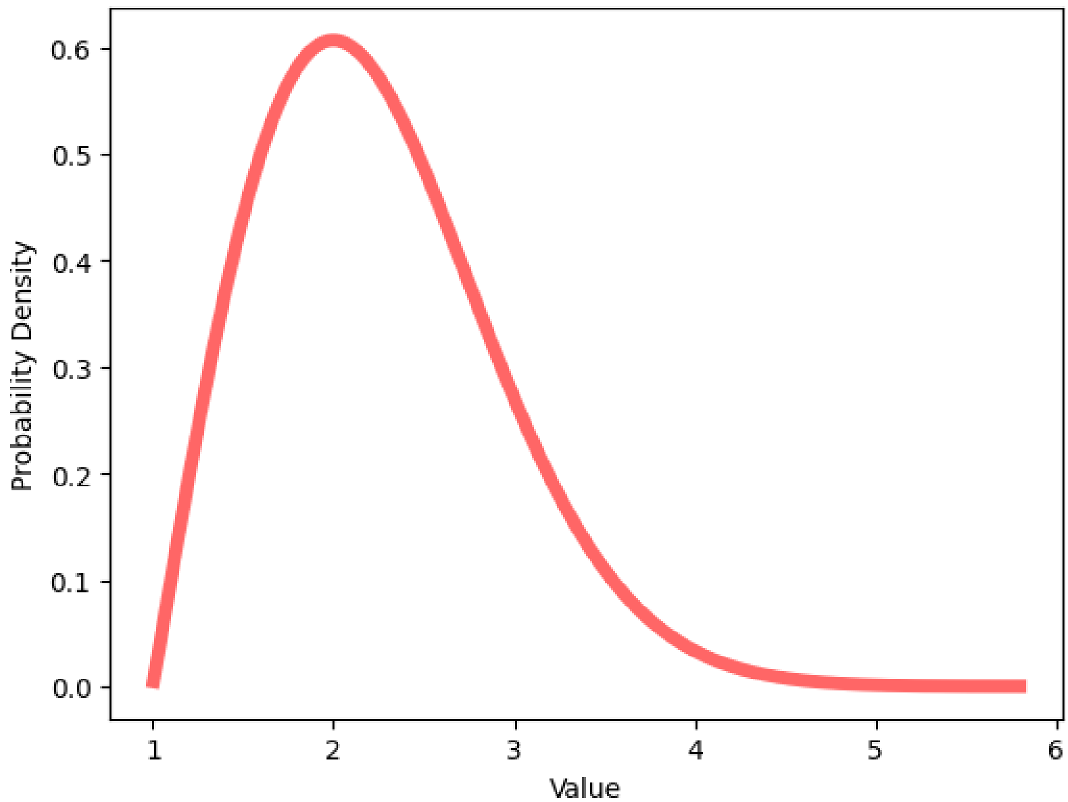
```python
# Generate Gaussian distributed random values for X and Y
X = np.random.normal(0, 1, num_realizations)
Y = np.random.normal(0, 1, num_realizations)

# Calculate Z and
Z = np.sqrt(X**2 + Y**2)
theta = np.zeros(num_realizations)
for i in range(num_realizations):
    if X[i] >= 0:
        theta[i] = np.arctan(Y[i]/X[i])
    else:
        theta[i] = np.arctan(Y[i]/X[i]) + np.pi

# Plot the probability density functions of Z and
plt.hist(Z, bins=30,density=True)
plt.xlabel('Z')
plt.ylabel('Probability Density')
plt.show()

plt.hist(theta,bins=30, density=True)
plt.xlabel(' ')
plt.ylabel('Probability Density')
plt.show()

# Find analytical distributions that fit the estimated PDFs
```


0.6
0.5
0.4
0.3
0.2
0.1
0.0
Probability Density
0
1
2
3
4
5
Z



[ ]:

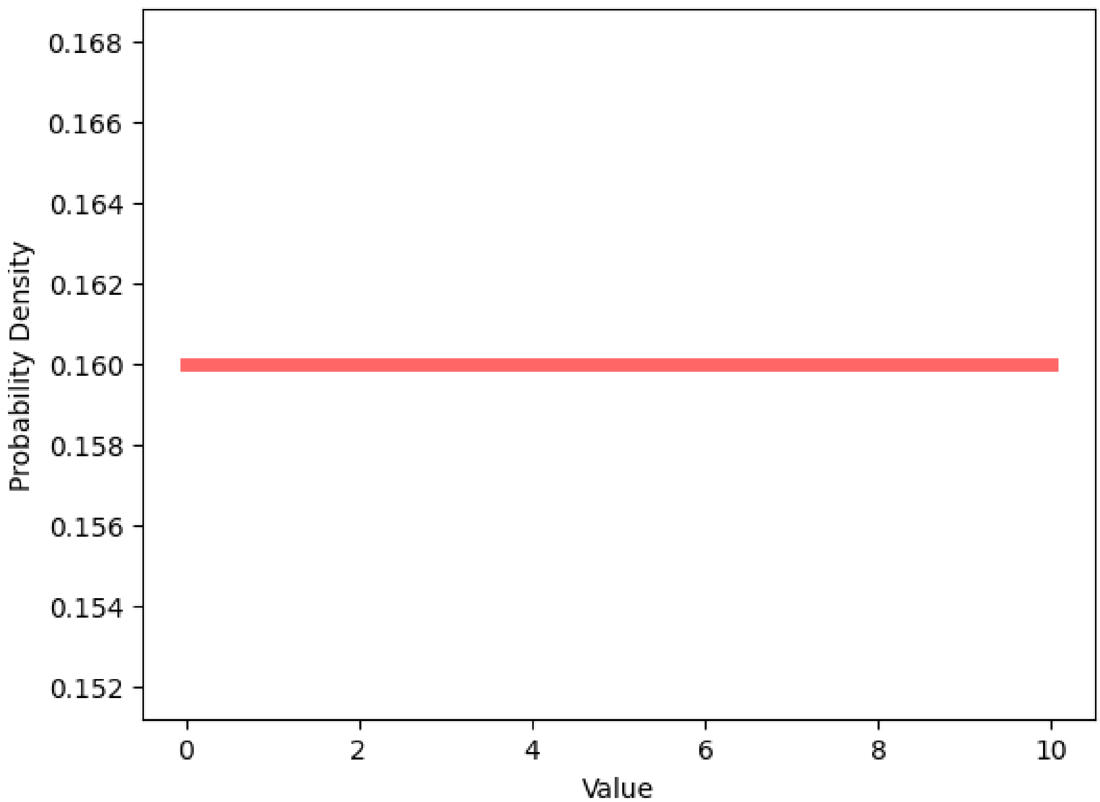
```
#Analytical distributions of Z

from scipy.stats import rayleigh, uniform

# Define the scale parameter
scale = 1

# Generate 1000 random values from a Rayleigh distribution with the given scale␣
↪parameter
x = np.linspace(rayleigh.ppf(0.00001, scale), rayleigh.ppf(0.99999, scale),␣
↪num_realizations)

# Plot the PDF of the Rayleigh distribution
plt.plot(x, rayleigh.pdf(x, scale), 'r-', lw=5, alpha=0.6, label='rayleigh pdf')
plt.xlabel('Value')
plt.ylabel('Probability Density')
plt.show()
```



[ ]:
```
#meanvalue = 1

#modevalue = np.sqrt(2 / np.pi) * meanvalue

# Generate 10000000 random values from a Rayleigh distribution with mean 0 and
 ↪standard deviation 1
#random_values = np.random.rayleigh(modevalue, 10000000)

# Plot a histogram of the random values
#plt.hist(random_values, bins=30)
#plt.xlabel('Value')
#plt.ylabel('Frequency')
#plt.show()
```

[ ]:
```
#analytical distributions of theta
# Define the lower and upper bounds of the uniform distribution
a = 0
b = 10

# Define the PDF function
def uniform_pdf(x):
```

```python
    return 0.16 if a <= x <= b else 0

# Generate 1000 evenly spaced values between the lower and upper bounds
x = np.linspace(a, b, 1000)

# Evaluate the PDF function for each value of x
pdf_values = [uniform_pdf(xi) for xi in x]

# Plot the PDF of the uniform distribution
plt.plot(x, pdf_values, 'r-', lw=5, alpha=0.6, label='uniform pdf')
plt.xlabel('Value')
plt.ylabel('Probability Density')
plt.show()
```



[ ]:
```python
# Generate 1000 random values from a uniform distribution between 0 and 1
#random_values = np.random.uniform(0, 1, 100000000)

# Plot a histogram of the random values
#plt.hist(random_values, bins=30)
#plt.xlabel('Value')
#plt.ylabel('Frequency')
```

```
#plt.show()
```