Communication systems Lab 5

Harshal Abhyankar 2019A3PS0282P

**Task 1**

from scipy import stats

import scipy

from numpy import sqrt

from scipy.integrate import quad

import numpy as np

# Uniform distribution

a,b = 0,2

def uniform(x):

    return 1 / (b - a)

uniform\_verify = quad(uniform, a, b)

# Gaussian distribution

mu,sigma = 0,1

def gauss(x):

    return scipy.stats.norm.pdf(x, mu, sigma)

gaussian\_verify = quad(gauss, -np.inf, np.inf)

# Rayleigh Distribution

alpha = 2

def ray(x):

    return scipy.stats.rayleigh.pdf(x, alpha)

ray\_verify = quad(ray, 0, np.inf)

# Exponential Distribution

beta = 2

def expdist(x):

    return scipy.stats.expon.pdf(x, beta)

expdist\_verify = quad(expdis, 0, np.inf)

print("Uniform Distribution verification:", uniform\_verify)

print("Gaussian Distribution verification:", gaussian\_verify)

print("Rayleigh Distribution verification:", ray\_verify)

print("Exponential Distribution verification:", expdist\_verify)

A screenshot of a computer

Description automatically generated

**Task 2**

**Code:**

from scipy import special

import numpy as np

import matplotlib.pyplot as plt

import numpy as np

import matplotlib.pyplot as plt

sigma = 12

mu = 0

x = sigma\*np.random.randn(10000) + mu

plt.hist(x)

plt.show()

sigma = 12

mu = 0

iter\_max=100000

count=0

for iter in range(iter\_max):

    nt = sigma\*np.random.randn(1) + mu

    if nt>sigma:

      count+=1

Prob= count/iter\_max

average\_prob=print(Prob)

qfunc\_prob=0.5 - 0.5\*special.erf(1/np.sqrt(2))

theoretical\_prob=print(qfunc\_prob)

import numpy as np

import matplotlib.pyplot as plt

sigma = 12

mu = 0

nt = sigma\*np.random.randn(1000) + mu

ac=np.correlate(nt, nt, mode = 'full')

#print(ac)

plt.figure(2)

plt. plot(ac)

plt.show()

Histogram:

Chart, histogram

Description automatically generated

A screenshot of a computer

Description automatically generated

Comment: As we increase the number of iterations, the actual value will get closer to the theoretical value in the above screenshot.

**ACF correlation:**

Graphical user interface, chart, histogram

Description automatically generated

**Task 3**

Code:

import random

import time

from  numpy.fft import fft

import matplotlib.pyplot as plt

import numpy as np

for T in range(30):

    start\_time\_sinc=-10

    stop\_time\_sinc=10

    N=10

    fm=N

    fs=10\*fm

    ts=1/fs

    time=np.arange(start\_time\_sinc,stop\_time\_sinc, ts)

    m\_t = 2\*N\*np.sinc(2\*N\*time)

    a=1

    sigma=0.01 # change this

    mu = 0

    n\_t=sigma\*np.random.randn(len(m\_t)) + mu

    y\_t=a\*m\_t + n\_t

    m\_t=y\_t

    mf=fft(m\_t)/fs

    N=len(mf)

    mf\_abs\_sorted=np.fft.fftshift(abs(mf))

    freq\_axis= np.linspace(-fs/2,fs/2, N)

    plt.figure(1)

    plt.plot(time+T,m\_t)

    plt.title("Time Domain")

plt.show()

For sigma=0.01:

Chart, bar chart, histogram

Description automatically generated

For sigma=0.1:

Chart, bar chart, histogram

Description automatically generated

**Conclusion: As sigma increases the distortion in the received signal increases which is due to the noise in the channel.**