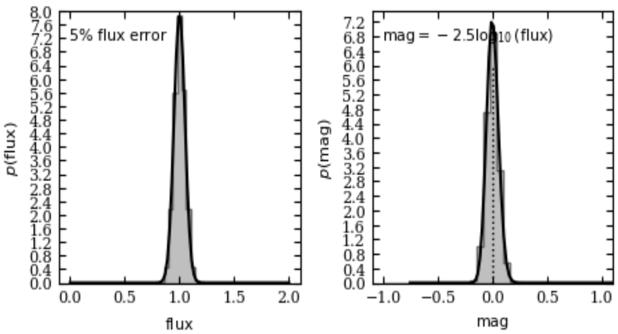
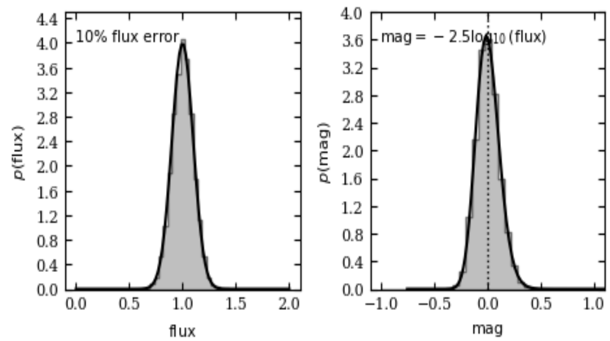
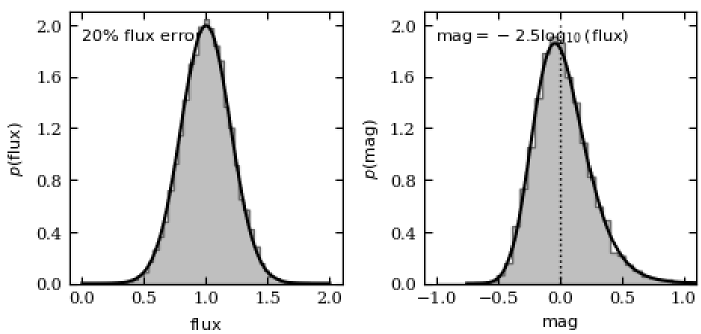
DSA ASSIGNMENT -1

Name: - Ashwin Kumar Roll no- CE21BTECH11008

Problem 1.







Code will be same for almost all three only changes happen in normal distribution function and some labelling of data and limit of the y axis. Rest of thing will be same.

```
import numpy as np
from matplotlib import pyplot as plt
from scipy.stats import norm

#----------------------------------------------------------------------
# This function adjusts matplotlib settings for a uniform feel in the textbook.
# Note that with usetex=True, fonts are rendered with LaTeX.  This may
# result in an error if LaTeX is not installed on your system.  In that case,
# you can set usetex to False.
# if "setup_text_plots" not in globals():

pip install astroML
from astroML.plotting import setup_text_plots
setup_text_plots(fontsize=8, usetex=False)
#------------------------------------------------------------
# Create our data

# generate 10000 normally distributed points
np.random.seed(1)
```

`

```python
dist = norm(1, 0.05)  #change the value here depending upon the error flux  error flux=
#std_dev*100/mean
flux = dist.rvs(10000)  #extracting 10000 random variate(sample)
flux_fit = np.linspace(0.001,2, 100)
pdf_flux_fit = dist.pdf(flux_fit)

# transform this distribution into magnitude space
mag = -2.5 * np.log10(flux) #magnitude from the intensity
mag_fit = -2.5 * np.log10(flux_fit)
pdf_mag_fit = pdf_flux_fit.copy() #taking copy
pdf_mag_fit[1:] /= abs(mag_fit[1:] - mag_fit[:-1])
pdf_mag_fit /= np.dot(pdf_mag_fit[1:], abs(mag_fit[1:] - mag_fit[:-1]))

#------------------------------------------------------------
# Plot the result
fig = plt.figure(figsize=(5, 2.5))
fig.subplots_adjust(bottom=0.17, top=0.9,
                    left=0.12, right=0.95, wspace=0.3)

# first plot the flux distribution
ax = fig.add_subplot(121)
ax.hist(flux, bins=np.linspace(0, 2, 50),
        histtype='stepfilled', fc='gray', alpha=0.5, density=True)
ax.plot(flux_fit, pdf_flux_fit, '-k')
# ax.plot([1, 1], [0, 2], ':k', lw=1)
ax.set_xlim(-0.1, 2.1)
ax.set_ylim(0, 8)#change this set limit according to highest probability value

ax.set_xlabel(r'${\rm flux}$')
ax.set_ylabel(r'$p({\rm flux})$')
ax.yaxis.set_major_locator(plt.MultipleLocator(0.4))
ax.text(0.04, 0.95, r'${\rm 5\%\ flux\ error}$',
        ha='left', va='top', transform=ax.transAxes) #change the labeling of graph depending upon
#error flux

# next plot the magnitude distribution
ax = fig.add_subplot(122)
ax.hist(mag, bins=np.linspace(-1, 2, 50),
        histtype='stepfilled', fc='gray', alpha=0.5, density=True)
ax.plot(mag_fit, pdf_mag_fit, '-k')
ax.plot([0, 0], [0, 6], ':k', lw=1)#change the coordinate of the line drawn at middle in second
ax.set_xlim(-1.1, 1.1)
ax.set_ylim(0, 7.5)#change the y limit according to probability of the magnitude
ax.yaxis.set_major_locator(plt.MultipleLocator(0.4))
ax.text(0.04, 0.95, r'${\rm mag} = -2.5\log_{10}({\rm flux})$',
        ha='left', va='top', transform=ax.transAxes)

ax.set_xlabel(r'${\rm mag}$')
ax.set_ylabel(r'$p({\rm mag})$')

plt.show()
```

As the error flux is decreasing it is clearly seen from the graph that distribution is becoming symmetric and at 5 % it is nearly symmetric. While at 20 % there is clear asymmetry can be seen. We can conclude that as the error flux will tend to 0 magnitude will behave like normal distribution.

`

PROBLEM 2:-

Importing the library

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats
import statistics as st
```

CODE

```
# np.random.seed(1)  #setting random seed of numpy as 1
dist=np.random.normal(1.5,0.5,1000)  #1.5 is mean ,0.5 is standard deviation, 1000 is the number of draws
plt.figure(facecolor='lightblue')
fig, ax = plt.subplots()
ax.set_facecolor('lightgray')
plt.hist(dist, bins=50, density=True, alpha=1,
color='yellow',edgecolor='black',label='Drawn Data')

#plotting of pdf
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 200)  #create a sequence of evenly spaced numbers x to plot against.
p =  scipy.stats.norm.pdf(x,1.5,0.5)
plt.plot(x, p, 'k', linewidth=2)  #here k dnotes black color
plt.title('Normal Distribution with Mean=1.5, Std Dev=0.5',fontsize=16)
plt.xlabel('Value',fontsize=12)
plt.ylabel('Probability Density',fontsize=12)
plt.legend()
plt.show()
# Calculate sample statistics
mean = np.mean(dist)
median = np.median(dist)
#first method
variance = np.var(dist) #more efficient and performance base than other
#second method
variance1= st.variance(dist)
#third method
variance2=dist.var()
#checked variance=variance1=variance2
skewness = scipy.stats.skew(dist)
kurtosis =  scipy.stats.kurtosis(dist)
MAD= np.median(np.abs(dist-median))
#first method
std_dev=np.std(dist)
#second method
std_dev1=1.492*MAD
#Third method
std_dev2=np.sqrt(variance)
# checked std_dev=std_dev1=std_dev2
print(f"\n mean = {mean},median= {median},variance=
{variance},skewness={skewness},kurtosis={kurtosis},MAD={MAD},standard
deviation={std_dev}")
```

`

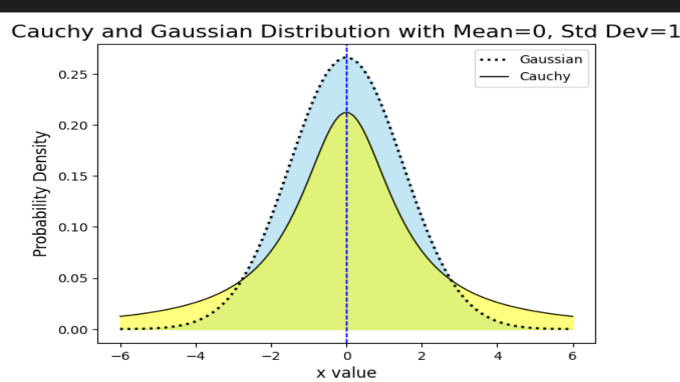
Problem 3

Importing the library

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats
```

CODE

```
#*****creating gaussian distribution *********
mean=0   #mean and standard deviation
std_dev=1.5

X= np.linspace(-4*std_dev,4*std_dev,100)  #genereating 100 number on x mod|4*std_dev| will cover almost all data

#first method
y1 = (1 / (std_dev * np.sqrt(2 * np.pi))) * np.exp((-((X - mean) / std_dev)**2)/2)
#equation of the gaussian distribution
plt.plot(X, y1, ':',color='k', linewidth=2,label='Gaussian')

#second method (as normal and gaussian distribution are same )
# y2 =scipy.stats.norm.pdf(X,0,1.5) #first is X value, 2nd is mean , 3rd parameter is standard deviation
# plt.plot(X, y2, 'k', linewidth=1)

plt.fill_between(X, y1, color='skyblue', alpha=0.5)
plt.axvline(mean, color='blue', linestyle='dashed', linewidth=1)

#*****creating Cauchy distribution *********
mean=0   #mean and standard deviation
std_dev=1
y_cauchy =  scipy.stats.cauchy.pdf(X,0,1.5)  #first is X value , 2nd is mean , 3rd arug is standard deviation
plt.plot(X, y_cauchy, 'k', linewidth=1,label='Cauchy')
plt.fill_between(X, y_cauchy, color='yellow', alpha=0.5)
plt.axvline(mean, color='blue', linestyle='dashed', linewidth=1)
plt.title('Cauchy and Gaussian Distribution with Mean=0, Std Dev=1',fontsize=16)
plt.xlabel('x value',fontsize=12)
plt.ylabel('Probability Density',fontsize=12)
plt.legend(loc='upper right')
plt.legend()

plt.show()
```



`

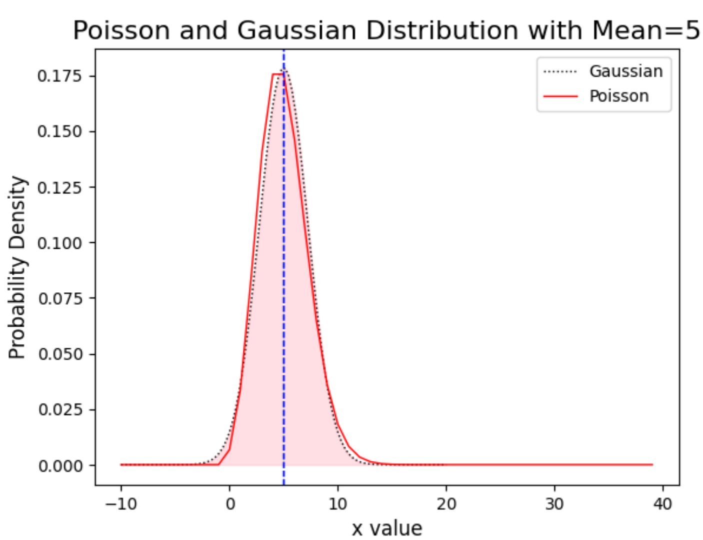
Problem-4

Importing the library

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats
```

Code

```
#crating gaussian distribution
mean=5   #mean and standard deviation
std_dev=5**0.5
X= np.linspace(-10,20,100)  #genereating 100 number on x mod|4*std_dev| will cover
almost all data

#first method
# y1 = (1 / (std_dev * np.sqrt(2 * np.pi))) * np.exp((-((X - mean) / std_dev)**2)/2)
#equation of the gaussian distribution
# plt.plot(X, y1, ':',color='k', linewidth=2,label='Gaussian')

# second method (as normal and gaussian distributon are same )
y2 =  scipy.stats.norm.pdf(X,mean,std_dev)
plt.plot(X, y2,':',color='k', linewidth=1,label='Gaussian')

plt.fill_between(X, y2, color='pink', alpha=0.5)
plt.axvline(mean, linestyle='dashed', linewidth=1)

#*****creating Poisson distribution *********
mean=5  #mean
X_poisson=np.arange(-10,40)
poisson_dist=  scipy.stats.poisson(5)
X_p= poisson_dist.pmf(X_poisson)
plt.plot(X_poisson, X_p, 'r', linewidth=1,label='Poisson')
y_cauchy =poisson_dist.pmf(X)

plt.axvline(mean, color='blue', linestyle='dashed', linewidth=1)
plt.title('Poisson and Gaussian Distribution with Mean=5',fontsize=16)
plt.xlabel('x value',fontsize=12)
plt.ylabel('Probability Density',fontsize=12)
plt.legend(loc='upper right')
plt.legend()
plt.show()
```



`

Problem-5

Importing the library

```
import numpy as np
```

CODE

```
#arranging the certain value and uncertainty in a array
certain_value= np.array([0.8920,0.881,.8913,0.9837,0.8958])
uncertainty=np.array([0.00044,0.009,0.00032,0.00048,0.00045])

weights= 1/uncertainty**2  #higher the uncertainty lesser the accuracy thus less contribution in
the weight
weighted_mean = np.average(certain_value,weights=uncertainty)
#total weight for the given measurement of the lifetime of K meson
sum_of_weights= np.sum(weights)
mean_uncertainty = np.sqrt(1/sum_of_weights)

print(f"Weighted mean lifetime = {weighted_mean*10**-10}s \nuncertainty of the mean =
{mean_uncertainty*10**-10}s")
```

Result

Weighted mean lifetime = 8.869955098222639e-11s

Uncertainty of the mean = 2.0318737026848628e-14s

Problem-6

Importing library

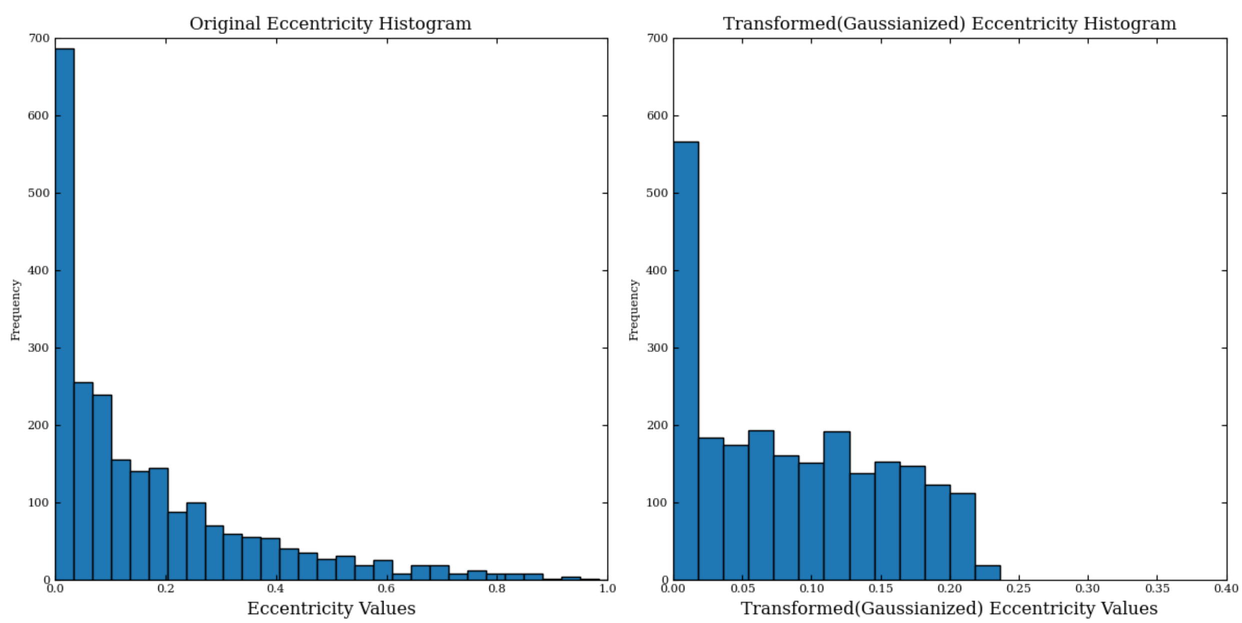
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import boxcox
import pandas as pd
```

CODE

```
df = pd.DataFrame(data)
data= pd.read_csv('data.csv')    #data for implementation
df = pd.DataFrame(data)

# Extract the column 'e' for which you want to draw the histogram
original_eccentricity = df['eccentricity'] #chosing the collumn with name
'eccentricity, and storing to varible
original_eccentricity = original_eccentricity.dropna() #removing NAN value from the
data and wquating with same

# Draw the histogram of the original eccentricity distribution
```

`

```python
plt.figure(figsize=(12, 6))
plt.title('Box-transformation',fontsize='12')
plt.subplot(1, 2, 1)
plt.hist(original_eccentricity, bins='auto', edgecolor='black',linewidth=1)
plt.title('Original Eccentricity Histogram',fontsize='12')
plt.xlabel('Eccentricity Values',fontsize='12')
plt.ylabel('Frequency')
plt.ylim(0,700)
plt.xlim(0,1)
# Apply Box-Cox transformation to Gaussianize the eccentricity distribution
trans_eccentricity, lambda_value = boxcox(original_eccentricity+1) # Adding 1 to
handle zero or negative values
# as some vaues are 0 but this model does not take negative and 0 as input

# Draw the histogram of the transformed eccentricity distribution
plt.subplot(1, 2, 2)
plt.hist(trans_eccentricity, bins='auto', edgecolor='black',lw=1)
plt.title('Transformed(Gaussianized) Eccentricity Histogram',fontsize='12')
plt.xlabel('Transformed(Gaussianized) Eccentricity Values',fontsize='12')
plt.ylabel('Frequency')
plt.ylim(0,700)
plt.xlim(0,0.4)
plt.tight_layout()
plt.show()
```



After Gaussianizing data is becoming more even and uniform .

`