**Problem 1**

Introduction

Using python function, a uniform, exponential, and normal distribution is made starting at a and ending at b. The list of values is tallied in a bar and graphed. This is compared to the PDF and function of the corresponding distribution. The mean and average of each distribution is calculated to compare with the theoretical value

Methodology

Each function takes different arguments and creates a list of values according to the distribution function used. These values are then grouped into a bin and plotted on a graph. Each PDF Function has it’s on mathematical function, thus a switch is made to switch between the right function needed. This is plotted to compared with the bin results.

Results and Conclusion

1.1

|  |  |  |  |
| --- | --- | --- | --- |
| **Table 1: Statistics for a Uniform Distribution** | | | |
| **Expectation** | | **Standard Deviation** | |
| Theoretical Calculation | Experimental Measurement | Theoretical Calculation | Experimental Measurement |
| 3.5 | 3.5 | .866 | .866 |

A screenshot of a cell phone

Description automatically generated

1.2

|  |  |  |  |
| --- | --- | --- | --- |
| **Table 2: Statistics for Exponential Distribution** | | | |
| **Expectation** | | **Standard Deviation** | |
| Theoretical Calculation | Experimental Measurement | Theoretical Calculation | Experimental Measurement |
| .33 | .330 | .33 | .330 |

A screenshot of a cell phone

Description automatically generated

Note: I couldn’t get the line to display correctly.

**1.3**

|  |  |  |  |
| --- | --- | --- | --- |
| **Table 3: Statistics for Normal Distribution** | | | |
| **Expectation** | | **Standard Deviation** | |
| Theoretical Calculation | Experimental Measurement | Theoretical Calculation | Experimental Measurement |
| 2.5 | 2.500 | .75 | .750 |

A screenshot of a cell phone

Description automatically generated

Appendix

import numpy as np

import matplotlib.pyplot as plt

a=2.0

b=5.0

beta = .33

mu = 2.5

sigma = .75

n = 1000000

x1 = np.random.uniform(a,b,n)

x2 = np.random.exponential(beta,n)

x3 = np.random.normal(mu,sigma,n)

#Calculate The Mean and STD

Theo\_Ux = 3.5

Theo\_Ox = np.sqrt(.75)

mu\_x= np.mean(x1) #1.1: 3.5

sig\_x=np.std(x1) #1.1: .86

mu\_x2= np.mean(x2) #1.2:.33

sig\_x2=np.std(x2) #1.2: .33

mu\_x3= np.mean(x3) #1.3: 2.5

sig\_x3=np.std(x3) #1.3:.75

def Graph(a,b,x,switch):

#Create bins and histogram

nbins=30 #NUmber of bins

edgecolor='w' #Color separating bars in the bargraph

bins = [float(x) for x in np.linspace(a,b,nbins)]

h1, bin\_edges = np.histogram(x,bins, density = True)

#Define points on the horizontal axis

be1=bin\_edges[0:np.size(bin\_edges)-1]

be2=bin\_edges[1:np.size(bin\_edges)]

b1=(be1+be2)/2

barwidth=b1[1]-b1[0] #Width of bars in the bargraph

plt.close('all')

#Plot the Bar Graph

fig1=plt.figure(1)

plt.bar(b1,h1, width=barwidth, edgecolor= edgecolor)

#Plot the Uniform PDF

def UnifPDF(a,b,x):

f=(1/abs(b-a))\*np.ones(np.size(x))

return f

def ExpoPDF(x):

f= 1/beta\*np.exp(-(1/beta\*x))

return f

def NormalPDF(x):

f = np.exp(-(x-mu)\*\*2/(2\*sigma\*\*2))/(sigma\*np.sqrt(2\*np.pi))

return f

if (switch == 1):

f = UnifPDF(a,b,b1)

plt.title('Uniform Distribution', fontsize = 14, fontweight = 'bold')

elif(switch == 2):

f = ExpoPDF(b1)

plt.title('Exponential Distribution', fontsize = 14, fontweight = 'bold')

else:

f = NormalPDF(b1)

plt.title('Normal Distribution', fontsize = 14, fontweight = 'bold')

plt.xlabel('X', fontsize = 14)

plt.ylabel('PDF', fontsize = 14)

plt.plot(b1,f,'r')

#Use only graph funcition. Last one used is displayed

Graph(a,b,x1, 1)

Graph(a,b,x2, 2)

Graph(a,b,x3, 3)

**Problem 2**

Introduction

Books thickness are uniformly distributed between a and b cm. The books are stacked and the sum of the thickness set is made. The expectation and std of the books and the stacks are calculated. The stack of books is plotted on a graph.

Methodology

The number of books stacked increases the range of b by nbooks. Thus sigma and mu can be also found by replacing b with the new maximum, multiplying b by nbooks stacked. The distribution function gives a list of stacked books and compared with Gaussian . The nbooks can changed to get a new graph.

Results and Conclusion

|  |  |  |
| --- | --- | --- |
| Mean Thickness Single Book(cm) | STD of thickness (cm) |  |
| µ*w* =2.0 | σ*w*= .86 |  |

|  |  |  |
| --- | --- | --- |
| Number of books *n* | Mean thickness of a stack of *n* books (cm) | Standard deviation of the thickness for *n* books |
| *n*=1 | µ*w* = 2 | σ*w*= .86 |
| *n*=5 | µ*w* = 13.5 | σ*w*= 6.64 |
| *n*=15 | µ*w* = 38.5 | σ*w*= 21.07 |

A screenshot of a cell phone

Description automatically generatedA close up of a logo

Description automatically generatedA close up of a logo

Description automatically generated

Appendix

import numpy as np

import matplotlib

import matplotlib.pyplot as plt

# Generate the values of the RV X

N=100000

#nbooks=1

#nbooks=5

nbooks=15

a=2

b=5

mu\_x=(a+b)/2

sig\_x=np.sqrt((b-a)\*\*2/12)

X=np.zeros((N,1))

for k in range(0,N):

x=np.random.uniform(a,b,nbooks)

w=np.sum(x)

X[k]=w

# Create bins and histogram

nbins=30; # Number of bins

edgecolor='w'; # Color separating bars in the bargraph

#

bins=[float(x) for x in np.linspace(nbooks\*a, nbooks\*b,nbins+1)]

h1, bin\_edges = np.histogram(X,bins,density=True)

# Define points on the horizontal axis

be1=bin\_edges[0:np.size(bin\_edges)-1]

be2=bin\_edges[1:np.size(bin\_edges)]

b1=(be1+be2)/2

barwidth=b1[1]-b1[0] # Width of bars in the bargraph

plt.close('all')

# PLOT THE BAR GRAPH

fig1=plt.figure(1)

plt.bar(b1,h1, width=barwidth, edgecolor=edgecolor)

#PLOT THE GAUSSIAN FUNCTION

def gaussian(mu,sig,z):

f=np.exp(-(z-mu)\*\*2/(2\*sig\*\*2))/(sig\*np.sqrt(2\*np.pi))

return f

f=gaussian(mu\_x\*nbooks,sig\_x\*np.sqrt(nbooks),b1)

plt.plot(b1,f,'r')