INTERNATIONAL
UNIVERSITY OF
APPLIED SCIENCES

Advance Statistics Course Task

University of Applied Science - Online

Masters in Computer Science (MSCS60ECTS)

Workbook Advanced Statistics(DLMDSAS01)

Vijaylaxmi Lendale

Matriculation : 92121824

Customer ID: 10525778

(vijaylaxmi.lendale@iu-study.org)

Professor: Paul Libbrecht

Acknowledged with the input of the Parameter Generator having signature:
7912e6285945781dd5b529b89f5533c59f4fb973

Submission Date: February 28, 2023

Contents

I List of Figures

1 Introduction

In this assignment workbook, I will solve each task by applying the knowledge gained from the advanced statistics course. A parameter generator helped me to obtain individual parameters, where every task parameter is replaced with the dynamically generated parameter from this signature - 7912e6285945781dd5b529b89f5533c59f4fb973 and I will use them in the respective task. To complete the six tasks present in the workbook, I used jupyter notebook to write python code. To create visualizations, I used matplotlib, and the seaborn libraries. The source codes are present in the respective assignment task sections.

2 Workbook

2.1 ASSIGNMENT 1: BASIC PROBABILITIES AND VISUALIZATIONS (1)

Parameter Generated: $\zeta_1 = 1$ **,** $\zeta_2 = 31$
Problem Statement: Please provide the requested visualization as well as the numeric results. In both cases, please provide how you realized these (calculations, code, steps…) and why it is the appropriate tools. Do not forget to include the scale of each graphic so a reader can read the numbers represented.
The number of meteorites falling on an ocean in a given year can be modeled by the following distribution. Give a graphic showing the probability of one, two, or three … meteorites falling (until the probability remains provably less than 0.5% for any bigger number of meteorites).

Calculate the expectation and median and show them graphically on this graphic:
As $\zeta_1 = 1$, Poisson distribution with an expectation of $\lambda = 31$

Solution (260 words):
The Poisson probability distribution((IU 21a),(IU 21b)) is a discrete probability distribution that expresses the probability of a given number of events occurring in a fixed interval of time or space if these events occur with a known average rate and independently of the time since the last event.

Suppose λ represents the expected number of meteorites that will fall into an ocean within a year. The probability mass function for a Poisson distribution with an expected value of $\lambda = 31$ can be expressed using the following equation

$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!} \tag{2.1}$$

$$P(k) = \frac{31^k e^{-31}}{k!} \tag{2.2}$$

where k is the number of events, λ is the average number of events per interval, and e is the base of the natural logarithm.

The goal is to create a bar chart that visualizes the probability of up to three meteorites falling, until the probability becomes less than 0.5% for any higher number of meteorites.
The code below utilizes the PMF to compute the probability for each value of k and generate the bar chart.

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

# Set the average number of meteorites per year
```

```
lambda_value = 31

# Calculate the probability for each value of k
k = np.arange(0, 50)
prob = poisson.pmf(k, lambda_value)

# Plot the probabilities on a bar chart
plt.bar(k, prob)
plt.xlabel('Number of meteorites')
plt.ylabel('Probability of Meteorites Falling in an Ocean')
plt.title('Poisson Distribution with lambda = 31')
plt.grid()
plt.show()
```

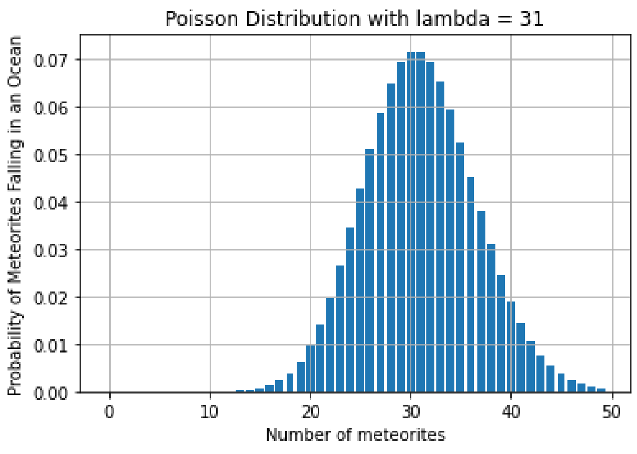


Figure 1: Distribution of Meteorites falling until p = 0.5%

The expectation of a Poisson distribution is given by the mean, which is equal to λ. In this case, the expectation is equal to 31. To show this on the graph, we can add a vertical line at x = 31 and label it "E(X) = λ = 31".The median of a Poisson distribution is given by the value k such that

$$P(X \leq k) \geq 0.5 \tag{2.3}$$

$$P(X \leq k - 1) < 0.5 \tag{2.4}$$

$$median = floor(\lambda) + (1/2) \tag{2.5}$$

The variance is given by

$$Var[X] = E[X^2] - (E[X])^2 \tag{2.6}$$

$$Var[X] = \lambda = 31 \tag{2.7}$$

Note: In poisson distribution mean, median and variance are same. Below graph shows the distribution of meteorites with median = λ and standard deviation σ, which is given as below

$$\sigma = \sqrt{Var[X]} = \sqrt{31} = 5.56 \tag{2.8}$$

We can graphically represent the expectation and median on the same plot as the probability mass function by using following code.

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

# Set the average number of meteorites per year
lambda_value = 31

# Calculate the probability for each value of k
k = np.arange(0, 50)
prob = poisson.pmf(k, lambda_value)

# Plot the probabilities on a bar chart
plt.bar(k, prob)

# Plot the expectation, median, variance and standard deviation
plt.axvline(lambda_value, color='r', label='Expectation')
plt.axvline(np.floor(lambda_value) + 1/2, color='g', label='Median')
plt.axvline(lambda_value, color='y', linestyle='--',label='Variance')
plt.axvline(lambda_value - np.sqrt(lambda_value), color='g',\
linestyle='dashdot', label='lambda - std')
plt.axvline(lambda_value + np.sqrt(lambda_value), color='c', \
linestyle='dashdot', label='lambda + std')
plt.xlabel('Number of meteorites')
plt.ylabel('Probability of Meteorites Falling in an Ocean')
plt.title('Expectation, Median, Variance and Standard Deviation \
of Poisson Distribution with lambda = 31')
plt.grid()
plt.legend()
plt.show()
```

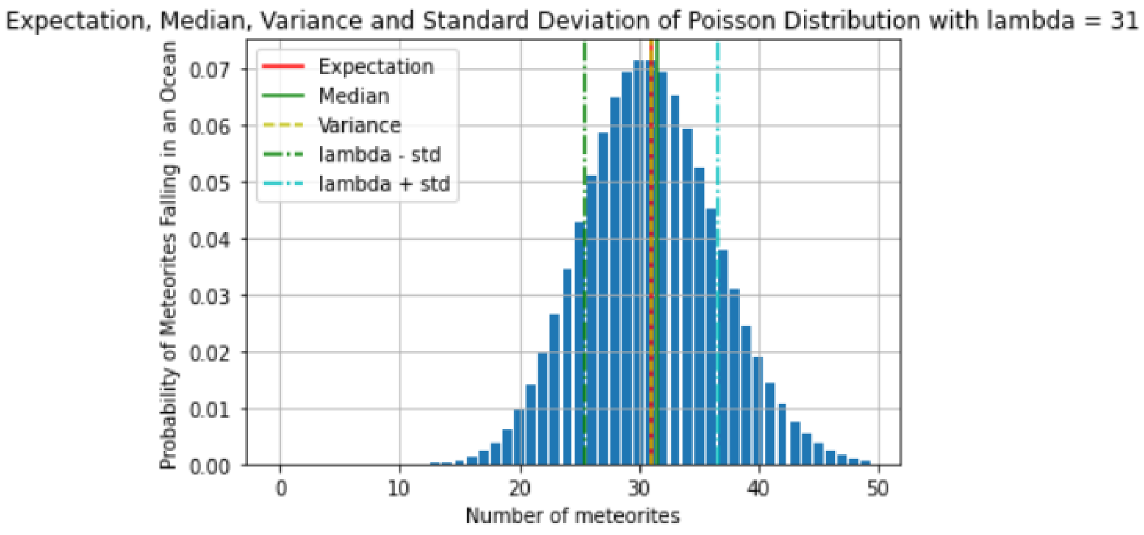


Figure 2: Distribution of Meteorites with expectation, standard deviation, median and variance

2.2 ASSIGNMENT 2: BASIC PROBABILITIES AND VISUALIZATIONS (2)

Parameter Generated: $\zeta_4 = 1$ **,** $\zeta_5 = 0.72$**,** $\zeta_6 = 8$ **,** $\zeta_7 = 0.27$ **,** $\zeta_8 = 7$
Let Y be the random variable with the time to hear an owl from your room's open window (in hours). Assume that the probability that you still need to wait to hear the owl after y hours is the probability given by

$$0.72e^{-8y^2}+0.27e^{-7y^8} \tag{2.9}$$

Find the probability that you need to wait between 2 and 4 hours to hear the owl, and compute and display the probability density function graph as well as a histogram by the minute. Compute and display in the graphics the mean, variance, and quarterlies of the waiting times. Please pay attention to the various units of time!

Solution (200 words):
To find the probability that you need to wait between 2 and 4 hours to hear the owl, we need to calculate the integral of the probability density function (PDF) of the random variable Y over the interval [2, 4].
The PDF of Y is given by the equation:

$$f(y) = 0.72e^{-8y^2}+0.27e^{-7y^8} \tag{2.10}$$

CDF is given by the following equation

$$P(Y \leq y) = 1 - P(Y > y) \tag{2.11}$$

The CDF to hear the owl in y hours can be calculated in the code and shown in the graph below.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import quad

def pdf(x):
    return 0.72 * np.exp(-8 * x ** 2) + 0.27 * np.exp(-7 * x ** 8)
```

```
def cdf(y):
    return quad(pdf, -np.inf, y)[0]

x = np.linspace(0, 10, 1000)
y = [cdf(i) for i in x]

plt.plot(x, y)
plt.xlabel('Waiting Time (hours)')
plt.ylabel('Cumulative Probability')
plt.title('CDF of Waiting Time to Hear the Owl')
plt.grid()
plt.show()
```

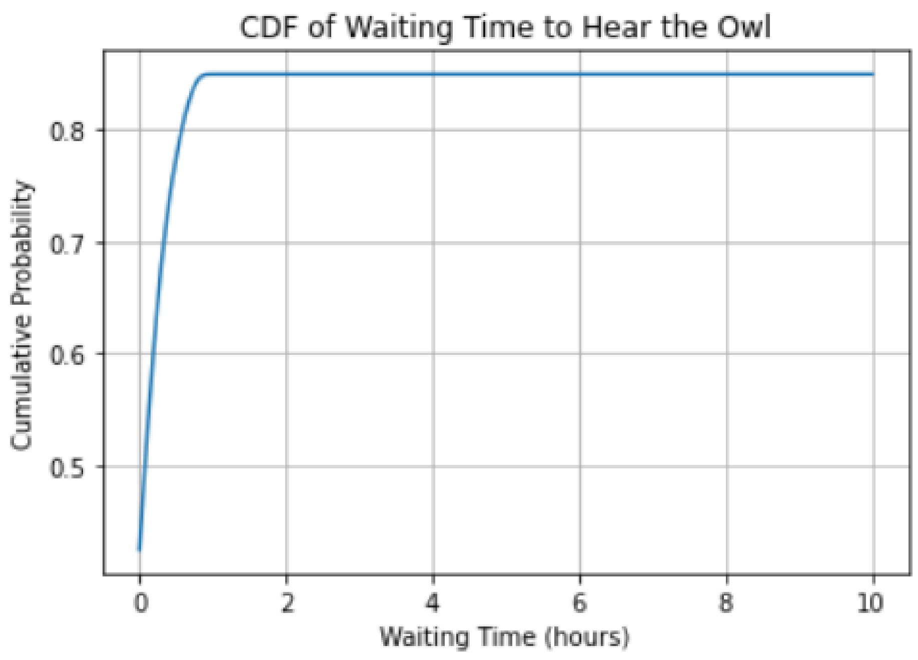


Figure 3: Cumulative Probability Distribution

So the probability of waiting for a duration between 2 and 4 hours can be determined by the following expression

$$P(2 \leq Y \leq 4) = \int_2^4 f(y)\, dy \tag{2.12}$$

$$P(2 \leq Y \leq 4) = \int_2^4 0.72e^{-8y^2} + 0.27e^{-7y^8}\, dy \tag{2.13}$$

Below is the code, which helps us to view the probability density function to hear the owl in y hours.

```
import numpy as np
from scipy.integrate import quad
import matplotlib.pyplot as plt
```

```python
def p(y):
    return 0.72 * np.exp(-8 * y**2) + 0.27 * np.exp(-7 * y**8)

def mean():
    return quad(lambda y: y * p(y), 0, np.inf)[0]

def std():
    mean_val = mean()
    return np.sqrt(quad(lambda y: (y - mean_val)**2 * p(y), 0, np.inf)[0])

mean_val = mean()
std_val = std()

y = np.linspace(0, 8, 1000)
plt.plot(y, p(y))
plt.axvline(mean_val, color='red', label='mean')
plt.axvline(mean_val + std_val, color='green', linestyle='--', label='+ 1 std')
plt.axvline(mean_val - std_val, color='green', linestyle='--', label='- 1 std')
plt.legend()
plt.xlabel('y (hours)')
plt.ylabel('Probability density')
plt.title('Probability density function to hear the owl in y hours')
plt.show()
```

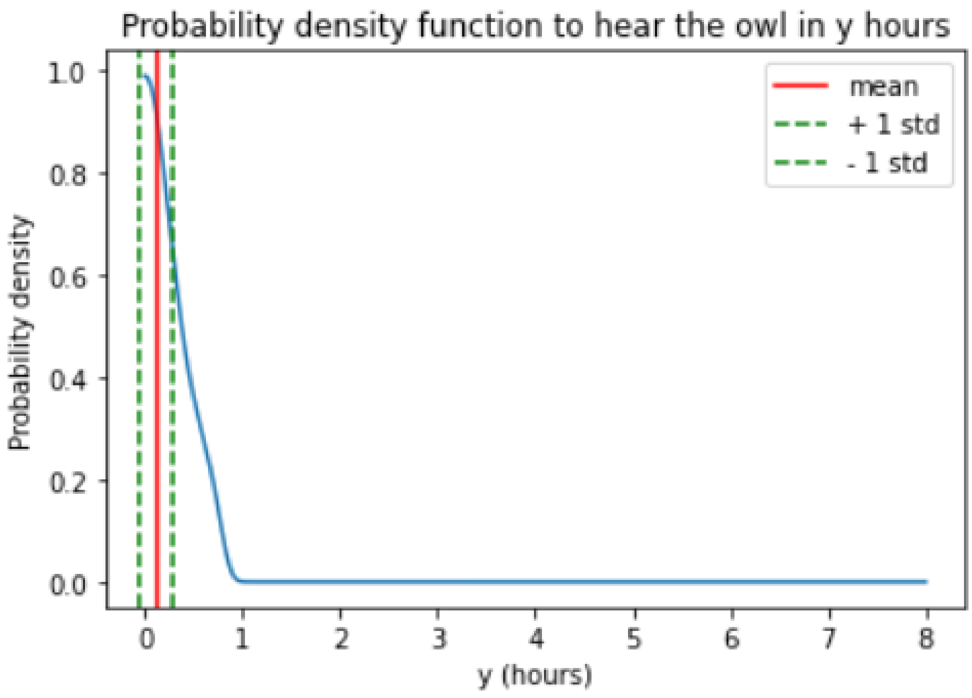


Figure 4: Density Function to hear the owl in y hours

To compute this integral, we need to use the definite integral of the function

$$F(y) = \int_{-\infty}^{y} f(x)\,dx \tag{2.14}$$

$$P(2 \le Y \le 4) = \int_{2}^{4} 0.72e^{-8y^2} + 0.27e^{-7y^8}\,dy \tag{2.15}$$

$$P(2 \le Y \le 4) = 2.8068 \tag{2.16}$$

Below is the python code to calculate the above integral.

```python
from scipy.integrate import quad
import numpy as np
import matplotlib.pyplot as plt

# Define the function for the PDF
def pdf(y):
    return 0.72*np.exp(-8*y**2) + 0.27*np.exp(-7*y**8)

# Define the limits of integration
a = 2
b = 4

# Calculate the definite integral
prob, _ = quad(pdf, a, b)

print("The probability of waiting between 2 and 4 hours to hear the owl is:", prob)
```

The expectation E[Y] can be calculated by equation and code is followed

$$E[Y] = \int_{0}^{\infty} y f_Y(y) dy \tag{2.17}$$

$$E[Y] = \int_{0}^{\infty} 0.72e^{-8y^2} dy + \int_{0}^{\infty} 0.27e^{-7y^8} dy$$

$$E[Y] = 0.1202 \tag{2.18}$$

```python
from scipy.integrate import quad

def pdf(y):
    return 0.72*np.exp(-8*y**2) + 0.27*np.exp(-7*y**8)
```

```
# Define the limits of integration
a = 0
b = np.inf

# Calculate the definite integral
expectation, _ = quad(lambda y: y*pdf(y), a, b)
print("The expectation of Y is:", expectation)
```

The variance Var[Y] calculation is given by

$$Var[Y] = E[Y^2] - (E[Y])^2 \tag{2.19}$$

To solve the above equation, we need to get the $E[Y^2]$.

```
from scipy.integrate import quad

def pdf(y):
    return 0.72*np.exp(-8*y**2) + 0.27*np.exp(-7*y**8)

# Define the limits of integration
a = 0
b = np.inf

# Calculate the Expectation E[Y^2]
expectation_y2, _ = quad(lambda y: y**2*pdf(y), a, b)

# Calculate the Expectation E[Y]
expectation, _ = quad(lambda y: y*pdf(y), a, b)

# Calculate the variance of Y
variance = expectation_y2 - expectation**2

print("The variance of Y is:", variance")
```

$$Var[Y] = E[Y^2] - (E[Y])^2 = 0.0382 \tag{2.20}$$

The Standard deviation is given

$$\sigma = \sqrt{Var[Y]} = 0.195 \tag{2.21}$$

The histogram to hear the owl in minutes can be viewed in the below graph. we know 1 hour = 60min.

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

def pdf(y):
    return 0.72*np.exp(-8*y**2) + 0.27*np.exp(-7*y**8)

# Generate a sample of waiting times
np.random.seed(0)
sample_size = 10000
y = np.random.exponential(scale=1/8, size=sample_size)

# Convert the waiting times to minutes
y_minutes = y * 60

# Plot the histogram
plt.hist(y_minutes, bins=20, density=True)
plt.xlabel('Waiting Time (minutes)')
plt.ylabel('Probability Density')
plt.title('Histogram of Waiting Times to Hear the Owl')
plt.show()
```

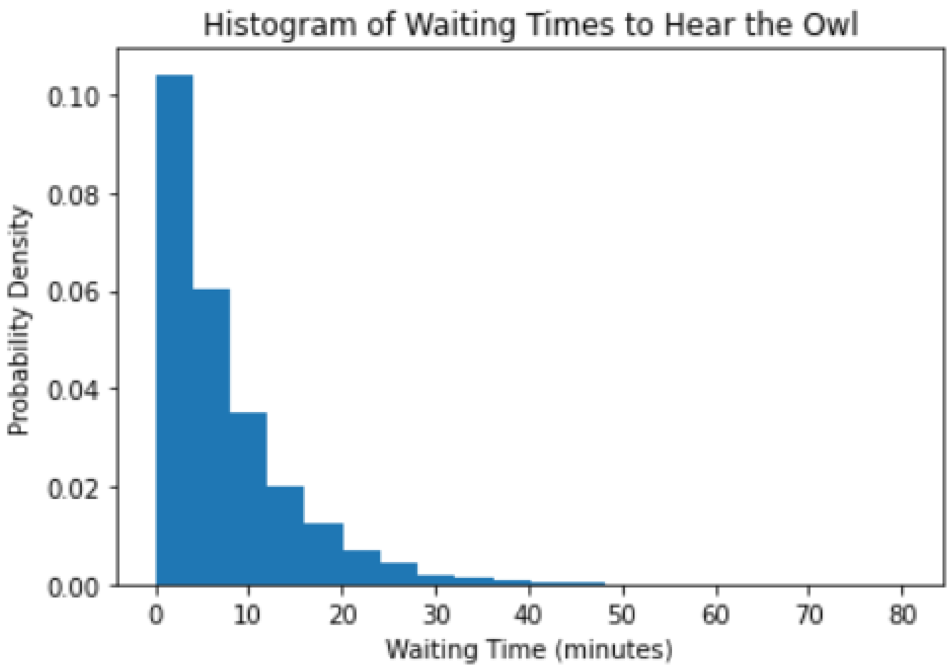


Figure 5: Histogram

2.3 ASSIGNMENT 3: Transformed Random Variables

Parameter Generated: $\zeta_9 = 1$ **,** $\zeta_10 = 180, 179, 84, 163, 27$

A network router has a bandwidth total to the first hardware failure called S expressed in terabytes. The random variable S is modeled by a distribution whose density is given by

$$f_S(s) = \frac{1}{24\theta^5} s^4 e^{-s/\theta} \tag{2.22}$$

with a single parameter θ. Consider the bandwidth total to failure T of the sequence of the two routers of the same type (one being brought up automatically when the first is broken). Express T in terms of the bandwidth total to failure of single routers S1 and S2. Formulate realistic assumptions about these random variables. Calculate the density function of the variable T. Given an experiment with the dual router system yielding a sample T1, T2, . . . , Tn, calculate the likelihood function for θ. Propose a transformation of this likelihood function whose maximum is the same and can be computed easily. An actual experiment is performed, and the infrastructure team obtains the bandwidth totals to failure given by the sequence

$$180, 179, 84, 163, 27$$

of numbers. Estimate the model parameter with the maximum likelihood and compute the the expectation of the bandwidth total to failure of the dual router system.

Solution (350 words):

we need to show that the distribution is exponential and for that, we need to rewrite the density function in the standard form of an exponential distribution, which is:

$$f(x) = \lambda e^{-\lambda x} \tag{2.23}$$

Comparing this to the given density function, we see that:

$$\lambda = \frac{1}{\theta} \tag{2.24}$$

Therefore, if we substitute this into the standard exponential distribution, we get

$$f(x) = \frac{1}{\theta} e^{-\frac{x}{\theta}} \tag{2.25}$$

which is the same as the given density function.
In Python, we can verify this by generating random numbers from the given distribution and fitting an exponential distribution to the sample using maximum likelihood estimation. If the distribution is exponential, the estimated parameter should be close to the true parameter used to generate the data.

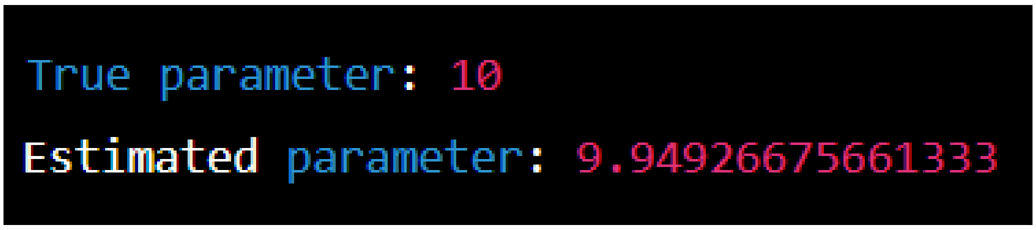
```python
import numpy as np
from scipy.stats import expon

# True parameter
theta = 10

# Generate random numbers from the given distribution
s = expon.rvs(scale=theta, size=10000)

# Fit exponential distribution to the sample using maximum likelihood estimation
theta_hat = expon.fit(s)[1]

print(f"True parameter: {theta}")
print(f"Estimated parameter: {theta_hat}")
```



```
True parameter: 10
Estimated parameter: 9.9492667566133
```

Figure 6: True parameter vs Estimated parameter

Hence, it is proved and we can move ahead by solving the actual task as shown below

The combined bandwidth capacity that fails for a sequence of two identical routers is equivalent to the sum of the bandwidth capacity that each individual router loses, as depicted below.

$$T = S_1 + S_2 \tag{2.26}$$

The random variables S1 and S2 are assumed to be independent and to follow the same distribution, S1 and S2 respectively. To determine the density of the sum of these two random variables, we can employ convolution as demonstrated below.

$$f_T(t) = \int_0^{\infty} s_1(\tau)s_2(t-\tau)d\tau \tag{2.27}$$

probability density function s(t) = 0 for $t \leq 0$ and $s(t) \geq 0$ and t>0

so integral equation is

$$f_T(t) = \int_0^{t} s_1(\tau)s_2(t-\tau)d\tau \tag{2.28}$$

with the same parameter θ, and that they follow the exponential distribution given by the density function

$$f_S(s) = \frac{1}{24\theta^5}s^4e^{-\frac{s}{\theta}} \tag{2.29}$$

The density function of T can be obtained through the convolution of the density functions of S1 and S2, and can be expressed as follows

$$f_T(t) = \int_{-\infty}^{\infty} f_S(t-s)f_S(s)ds = \left(\frac{1}{24\theta^5}\right)^2 * t^8 * e^{-\frac{2t}{\theta}} \tag{2.30}$$

The gamma distribution has a couple of uses in statistics. Firstly, it can be employed to approximate the precision of a normal distribution by taking into account previous knowledge. Additionally, it can be used to estimate the rate parameter of an exponential distribution.

This distribution possesses two parameters: the shape parameter and the scale parameter. Moreover, the density function of T is actually the moment-generating function of Gamma(2,θ). The function itself can be defined as follows: when t is greater than 0, it is equal to S to the power of 4, multiplied by e to the power of negative s over , and then divided by 24 times to the power of 5. However, if t is less than or equal to 0, the function evaluates to 0 instead (IU 21c)

```
# Prepare x-axis and y-axis data
theta_values = [0.5, 1, 1.5, 2, 2.5, 3]
s_values = np.linspace(0, 5, 100) # x-axis data
f_values = []

# Generate function using Sympy and Lambdify
s = symbols('s')
for index, theta in enumerate(theta_values):
    f = Piecewise(((1/(24*theta**5))*s**4*(2.718281**(-s/theta)), s>0), (0, True))
    lam_f = lambdify(s, f, modules=['numpy'])
    f_values.append(lam_f(s_values))

# Set up chart area for the density function
colors = cm.tab10(range(20))
fig, ax = plt.subplots()
fig.suptitle('Density function of T for different ' + r'$\theta}$' + \
' values', fontsize=20)
fig.set_size_inches(10, 5)

# Plot the density functions for different theta values
for index, theta in enumerate(theta_values):
    ax.plot(s_values, f_values[index], linewidth=2, color=colors[index],
             alpha=0.8, label= r'$\theta}$' + ' = ' + str(theta))

# Annotate the chart
ax.legend(loc='upper right')
ax.set_ylim(0, 1)
ax.set_ylabel(r'$f(t)=\frac{1}{24*\theta^5}s**4e^\frac{-t}{\theta}\  when \  t > 0
\  otherwise \  0$', fontsize=15)
ax.set_xlabel('T values', fontsize=15)
ax.grid()
```

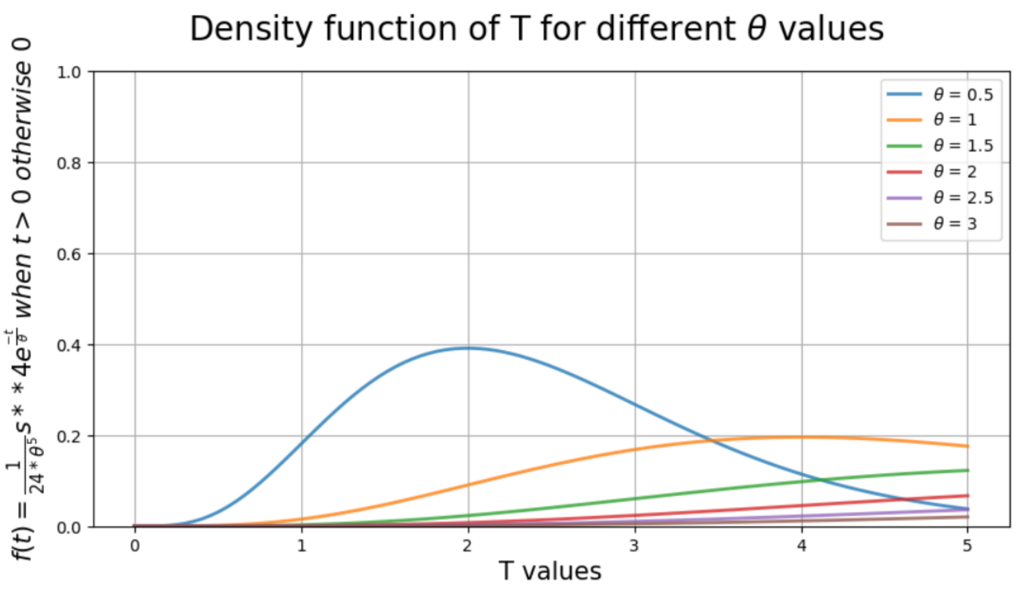


Figure 7: Density Function of T for different $\theta values$

When prior knowledge of a dataset's distribution is available, Maximum Likelihood Estimation (MLE) can be used to determine the most likely distribution parameters. MLE involves optimizing the likelihood function $L(\theta)$ with respect to the unknown parameter θ. The likelihood function of θ, based on n observations $T_1, T_2, ..., T_n$, can be defined by (IU 21d)

$$L(\theta) = \frac{1}{\theta^2}\prod_{i=1}^{n} t_i e^{-\sum_{i=1}^{n} t_i/\theta} \tag{2.31}$$

The log-likelihood function is:

$$log(L(\theta)) = -2nlog(\theta) + \sum_{n=i}^{n} log(t_i) - \sum_{n=i}^{n} t_i/\theta \tag{2.32}$$

To find the maximum likelihood estimate, we differentiate the log-likelihood function with respect to θ and set it to zero:

$$\frac{\delta logL(\theta)}{\delta\theta} = \frac{-2n}{\theta} + \sum_{n=1}^{i} t_i/\theta^2 = 0 \tag{2.33}$$

Solving for θ, we get:

$$\hat{\theta} = \frac{\sum_{i=1}^{n} t_i}{2n} \tag{2.34}$$

we need to Compute the maximum likelihood estimate of the model-parameter and determine the expected bandwidth total to failure of the dual-router-system.

```
time_intervals = [180, 179, 84, 163, 27]

sum_of_intervals = np.sum(time_intervals)
denominator = 2*len(time_intervals)
theta_estimate = sum_of_intervals / denominator
print('theta =', theta_estimate)

expected_dual_bandwidth = 2*theta_estimate
```

```
print('Expected bandwidth total to failure =', expected_dual_bandwidth)
```

```
import numpy as np
# Prepare x-axis and y-axis data
theta_values = [63.3]
s_values = np.linspace(0, 1000, 2000) # x-axis data
# Initialize an array for y-axis values for different theta values
y_values = []
#Generate function using Sympy and Lambdify
s = symbols('s')

for index, theta in enumerate(theta_values):
# print(index, theta)
""" Piecewise is used for expressions s > 0 or otherwise """
func = Piecewise(((1/(24theta5))*s4(2.718281**(-s/theta)), s>0), (0, True))
lam_func = lambdify(s, func, modules=['numpy'])
y_values.append(lam_func(s_values))

# Chart area for the density function
multiple_colors = cm.tab10(range(20))
# use programmatically in over-plots for better control
fig, ax = plt.subplots()
fig.suptitle('Density function of $T$ for ' + r'$\theta}$ = ' + \
"{:.1f}".format(float(estimated_theta)), fontsize=20)
fig.set_size_inches(10, 5)

# Plotting the density functions for different theta values in a loop
for index, theta in enumerate(theta_values):
ax.plot(s_values, y_values[index], linewidth=2, color=multiple_colors[index],
alpha=0.8, label= r'$\theta}$' + ' = ' + str(theta))

# Annotate additional information on the chart
ax.legend(loc='upper right')
ax.set_xlim(0, 1000)
ax.set_ylim(0, 0.004)

# Check S**4
ax.set_ylabel(r'$f(s)=\frac{1}{24*\theta^5}s**4e^\frac{-s}{\theta}\
when \ s > 0 \ otherwise \ 0$', fontsize=15)
ax.set_xlabel('s values', fontsize=15)
ax.grid()
```

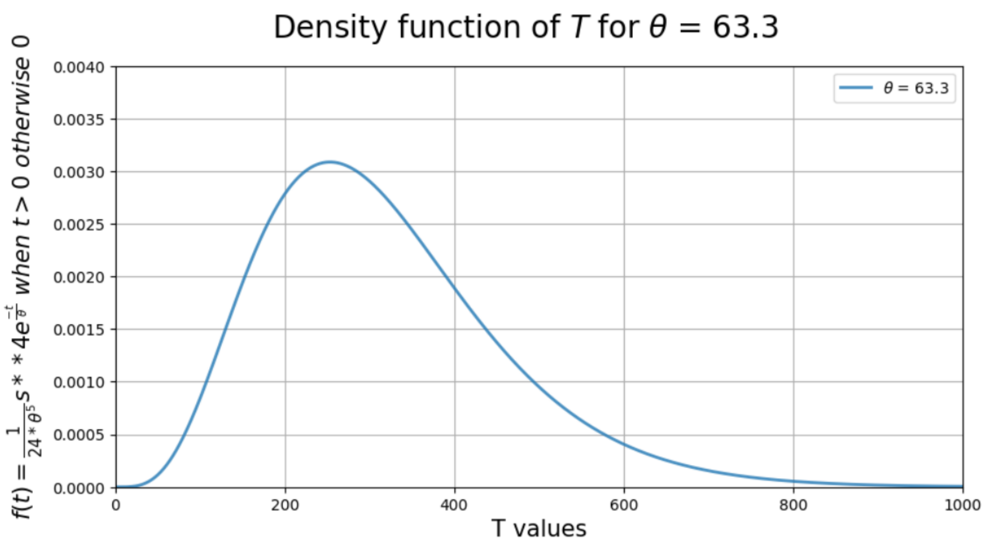


Figure 8: Density function of T for θ= 63.3

2.4 ASSIGNMENT 4: Hypothesis Test

Parameter Generated: $\zeta_1 1 = 828$**,** $\zeta_1 2 = 56.6$**,** $\zeta_1 3 = 0$ **,** $\zeta_1 4 = 809, 864, 783, 801, 830, 837, 833, 823, 851, 795$

Over a long period of time, the production of 1000 high-quality hammers in a factory seems to have reached a weight with an average of 828 (in g) and a standard deviation of 56.6 (in g). Propose a model for the weight of the hammers including a probability distribution for the weight. Provide all the assumptions needed for this model to hold (even the uncertain ones). What parameters does this model have?

Answer the following question about a new production system:
$\zeta_1 3 = 0$ then Does the new system make more constant weights? To answer this question a random sample of newly produced hammers is evaluated yielding the weights 809, 864, 783, 801, 830, 837, 833, 823, 851, and 795. What hypotheses can you propose to test the question? What test and decision rule can you make to estimate if the new system answers the given question? Express the decision rules as logical statements involving critical values. What error probabilities can you suggest and why? Perform the test and draw a conclusion to answer the question.

solution (410 words):

A hypothesis test is a statistical technique that utilizes sample data to evaluate the validity of a hypothesis about a population parameter. It is a tool used for making inferences.

The weight of the hammers produced in the factory can be modeled using a normal distribution with a mean of 828 g and a standard deviation of 56.6 g. The probability distribution for the weight is given by

$$f(w) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(w-\mu)^2}{2\sigma^2}} \tag{2.35}$$

The assumptions required for this model to hold include:

- The weight of each hammer is independently and identically distributed.

- The weight of each hammer is approximately normally distributed.
- The standard deviation of the weight of the hammers is constant.

This model has two parameters: the mean weight of the hammers (μ) and the standard deviation of the weight (σ)

If $\zeta_1 3 = 0$, then the new production system makes more constant weights.To test this hypothesis, we can propose the following hypotheses:

- Null hypothesis (H0): The mean weight of the hammers produced by the new system is equal to 828 g
- Alternative hypothesis (Ha): The mean weight of the hammers produced by the new system is different from 828 g

We can use a one-sample t-test to test the hypotheses. The test statistic is given by (IU 21e)

$$t = \frac{\bar{x} - \mu}{s/\sqrt{n}} \tag{2.36}$$

We can use a significance level of 0.05 for the test. The decision rule can be expressed as

- Reject H0 if $|t| \leq t\_critical$
- Fail to reject H0 if $|t| <= t\_critical$ where $t\_critical$ is the critical value for a t-distribution with n-1 degrees of freedom and a significance level of 0.05.

We can suggest an error probability of 0.05 for Type I error (rejecting H0 when it is true) and an error probability of 0.05 for Type II error (failing to reject H0 when it is false).

If we perform the test with the sample data, we get

- Sample mean weight:

$$\bar{x} = \frac{809 + 864 + 783 + 801 + 830 + 837 + 833 + 823 + 851 + 795}{10} = 824g$$

- Sample standard deviation:

$$s = \sqrt{\frac{\sum_{i=1}^{10}(x_i - \bar{x})^2}{10 - 1}} = 26.2$$

- Sample size: 10
- The hypothesis to be tested is:

$$H_0 : \mu_{new} = 828, \text{ against } H_a : \mu_{new} \neq 828$$

- The test statistic is the z-score:

$$z = \frac{\bar{x} - \mu}{\frac{s}{\sqrt{n}}}$$

We can use a one-sample z-test with a significance level of $\alpha = 0.05$ to test this hypothesis. The decision rule is to reject H_0 if $|z| > z_{\alpha/2}$, where $z_{\alpha/2}$ is the critical value for a two-tailed test with significance level $\alpha/2$. Performing the test, we get:

$$z = \frac{824 - 828}{\frac{26.2}{\sqrt{10}}} = -1.18$$

Since $|z| \leq z_{0.05/2} = 1.96$, we fail to reject H_0, and thus conclude that there is not enough evidence to say that the new system produces more constant weights.

2.5 ASSIGNMENT 5: Regularized Regression

Parameter Generated:

$\zeta_1 5 = 0$, $\zeta_1 6$ = (-84, -1.1850374458679788e+22), (4, 25434465.9),(-47, -20630769008006644000), (72, 2.0530947555484102e+21), (59, 232659899900492380000), (27, 43448916433631780), (82, 8.974390166339385e+21), (18, 468539805310500.2), (59, 225358419807864230000), (-96, -5.025685455205405e+22), (-70, -1.6314358361511897e+21), (2, 9942.98), (33, 374122350727434430), (62, 428368380294070670000), (3, 966515.32), (8, 57213346834.82), (-87, -1.8300934567507545e+22), (-8, -80264086444.54), (76, 3.7824261787623044e+21), (-96, -5.164663789485524e+22), (-63, -529308091230959000000), (-97, -5.585848133060702e+22), (-8, -77051359333.67), (-72, -2.206406652461232e+21)

Given the values of an unknown function $f : \mathbb{R} \to \mathbb{R}$ at some selected points, we try to calculate the parameters of a model function using OLS as a distance and a ridge regularization:

if $\zeta_{15} = 0$: a polynomial model function of twelve α_i parameters:

$$f(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \cdots + \alpha_{12} x^{12}$$

Calculate the OLS estimate, and the OLS ridge-regularized estimates for the parameters given the sample points of the graph of f given that the values are

y = (-84, -1.1850374458679788e+22), (4, 25434465.9),(-47, -20630769008006644000), (72, 2.0530947555484102e+21), (59, 232659899900492380000), (27, 43448916433631780), (82, 8.974390166339385e+21), (18, 468539805310500.2), (59, 225358419807864230000), (-96, -5.025685455205405e+22), (-70, -1.6314358361511897e+21), (2, 9942.98), (33, 374122350727434430), (62, 428368380294070670000), (3, 966515.32), (8, 57213346834.82), (-87, -1.8300934567507545e+22), (-8, -80264086444.54), (76, 3.7824261787623044e+21), (-96, -5.164663789485524e+22), (-63, -529308091230959000000), (-97, -5.585848133060702e+22), (-8, -77051359333.67), (-72, -2.206406652461232e+21)

Remember to include the steps of your computation which are more important than the actual computations.

Solution (245 words):

To solve for the OLS estimate, we need to minimize the sum of squared residuals, given by (IU 21f):

$$\sum_{i=1}^{n} (y_i - f(x_i))^2$$

where n is the number of data points, y_i is the observed output at point i, and $f(x_i)$ is the predicted output of the model at point i.

The OLS solution for the parameters α can be found by minimizing this sum of squared residuals. This can be done by solving the normal equations:

$$(X^T X)\alpha = X^T y$$

where X is a matrix whose rows are the input points x_i and their powers up to 12, and y is a vector of the observed outputs y_i.

The OLS solution can be found using the equation:

$$\alpha = (X^T X)^{-1} X^T y$$

To find the ridge-regularized solution, we add a penalty term to the sum of squared residuals, given by:

$$\sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \sum_{j=0}^{12} \alpha_j^2$$

where λ is a hyperparameter that controls the strength of the penalty. We can find the ridge-regularized solution by solving the modified normal equations:

$$(X^T X + \lambda I)\alpha = X^T y$$

where I is the identity matrix. The ridge-regularized solution can be found using the equation:

$$\alpha = (X^T X + \lambda I)^{-1} X^T y$$

```
# ridge regularisation problem

data=[(-84, -1.1850374458679788e+22), (4, 25434465.9), \
(-47, -206307690080066440000),(72, 2.0530947555484102e+21), \
(59, 232659899900492380000), (27, 434489164336317800),\
(82, 8.974390166339385e+21), (18, 468539805310500.2), \
(59, 225358419807864230000),(-96, -5.025685455205405e+22), \
(-70, -1.6314358361511897e+21), (2, 9942.98), \
(33, 3741223507274344430), (62, 428368380294070670000), \
(3,966515.32), (8, 57213346834.82), \
(-87, -1.8300934567507545e+22), (-8, -80264086444.54),\
(76, 3.7824261787623044e+21),(-96, -5.164663789485524e+22), \
(-63, -529308091230959000000),(-97, -5.585848133060702e+22), \
(-8, -77051359333.67), (-72, -2.206406652461232e+21)]
""" plot graph """
x_init = np.array([i[0] for i in data])
y_init = np.array([i[1] for i in data])

""" plot graph scatter"""
```

```
fig, ax = plt.subplots(figsize=(10, 6))
ax.scatter(x_init, y_init, marker='o', s=50, label='Data points')
ax.set_xlabel("x")
ax.set_ylabel("y")
ax.set_title("Scatter plot of data points")
plt.legend()
plt.grid()
```

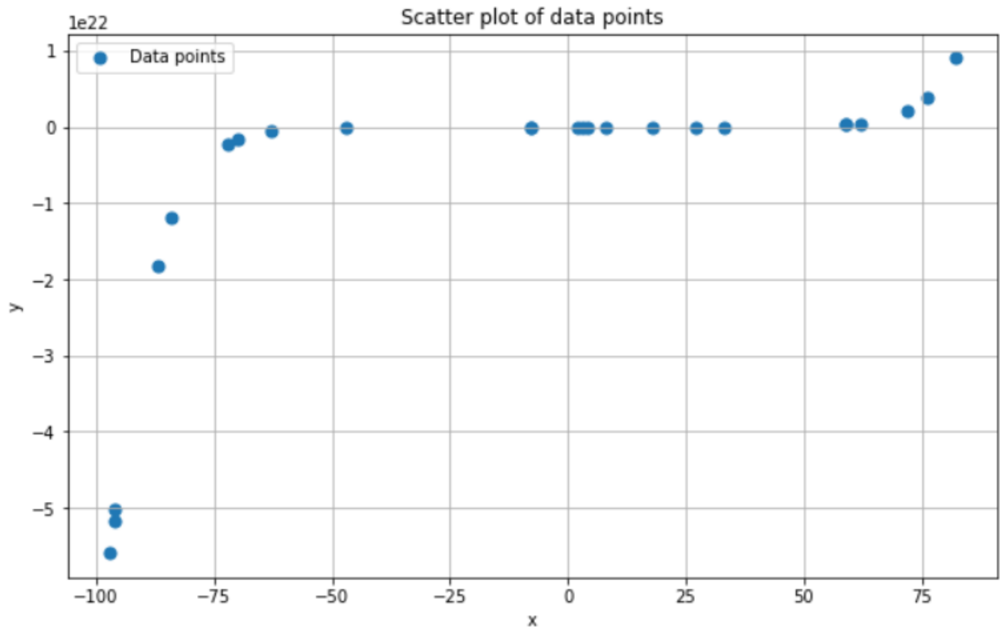


Figure 9: Scatter plot of data points

```
# OLS ESTIMATES analytical method for y= mx+c
# ref: https://numpy.org/doc/stable/reference/generated/numpy.linalg.lstsq.html
x = np.array([0, 1, 2, 3])
y = np.array([-1, 0.2, 0.9, 2.1])

A = np.vstack([x, np.ones(len(x))]).T
CEstimate = np.linalg.inv(A.T @ A) @ A.T @ y
print(CEstimate)
```

```
# [ 1 x x**2 .. x**12]

# dimension
# [ 1 -84 (-84)**2 .. x**12]
# [ 1 4 4**2 .. 4**12]
# ...
# [ 1 x x**2 .. x**12]

X = np.zeros((len(data), 13))
```

```
y = np.zeros((len(data), 1))

for i, (x, yi) in enumerate(data):
    for j in range(13):
        X[i, j] = x**j
    y[i] = yi

olsEstimate = np.linalg.inv(X.T @ X) @ X.T @ y
print(olsEstimate)
```

```
# [ 1 x x**2 .. x**12]

# dimension
# [ 1 -84 (-84)**2 .. x**12]
# [ 1 4 4**2 .. 4**12]
# ...
# [ 1 x x**2 .. x**12]

X = np.zeros((len(data), 13))
y = np.zeros((len(data), 1))

for i, (x, yi) in enumerate(data):
    for j in range(13):
        X[i, j] = x**j
    y[i] = yi

olsEstimate = np.linalg.inv(X.T @ X) @ X.T @ y
print(OLS estimate)
```

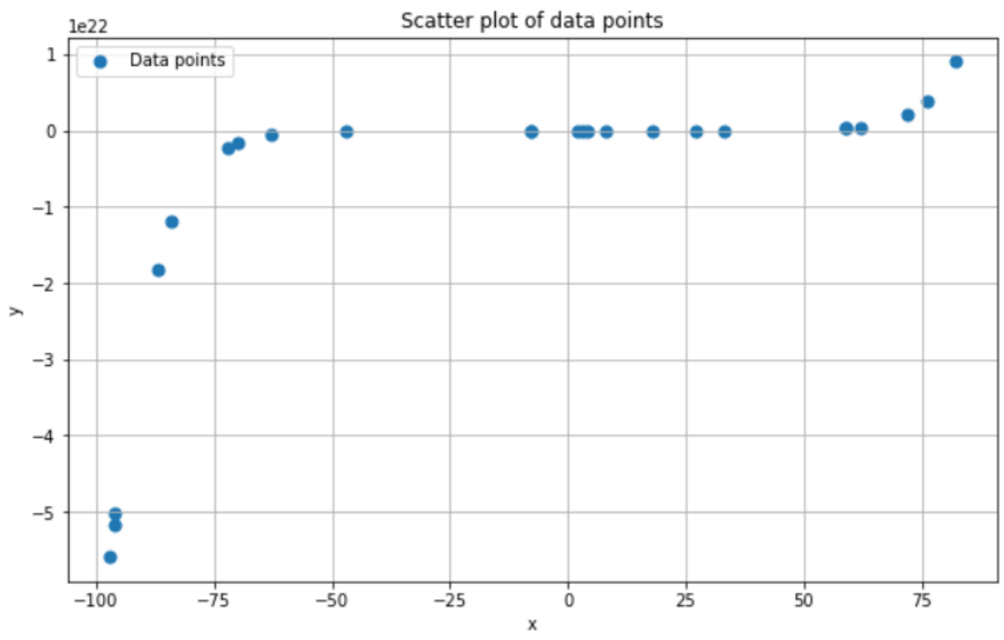


Figure 10: olsEstimates

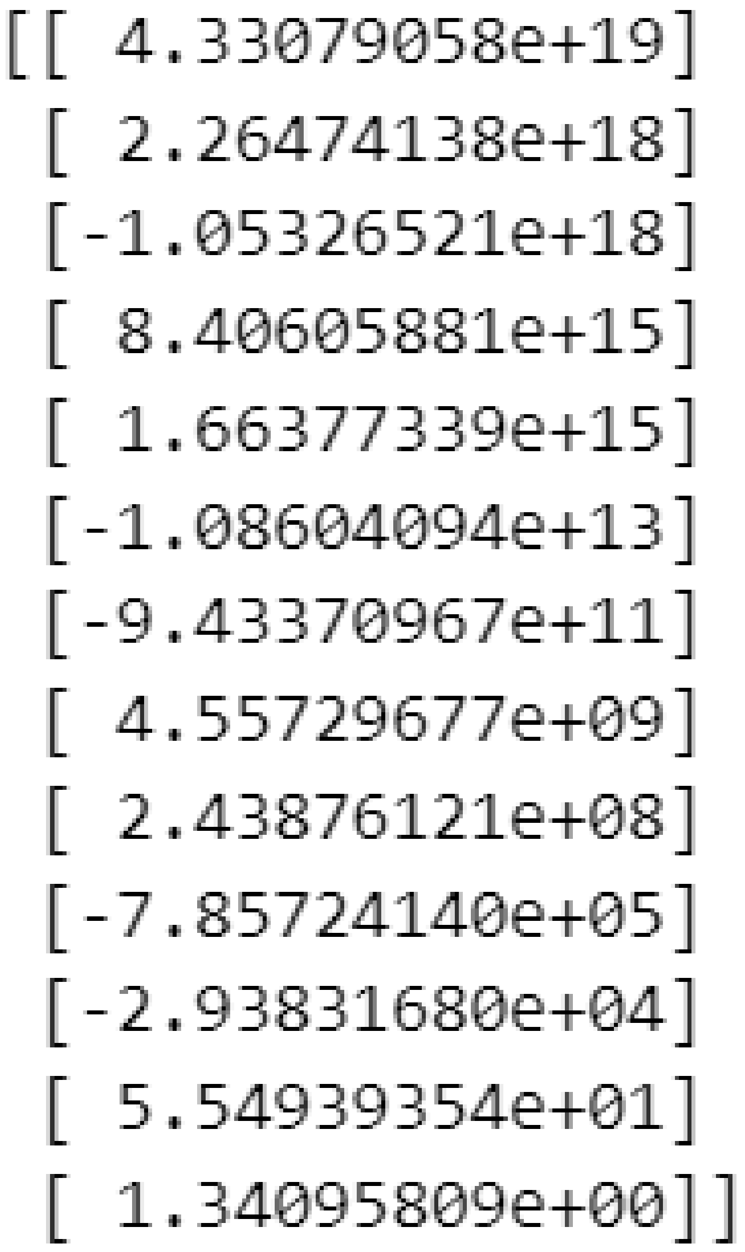
```
# [ 1 x x**2 .. x**12]

# dimension
# [ 1 -84 (-84)**2 .. x**12]
# [ 1 4 4**2 .. 4**12]
# ...
# [ 1 x x**2 .. x**12]

X = np.zeros((len(data), 13))
y = np.zeros((len(data), 1))

for i, (x, yi) in enumerate(data):
    for j in range(13):
        X[i, j] = x**j
    y[i] = yi

olsEstimate = np.linalg.inv(X.T @ X) @ X.T @ y
print(OLS estimate)
```



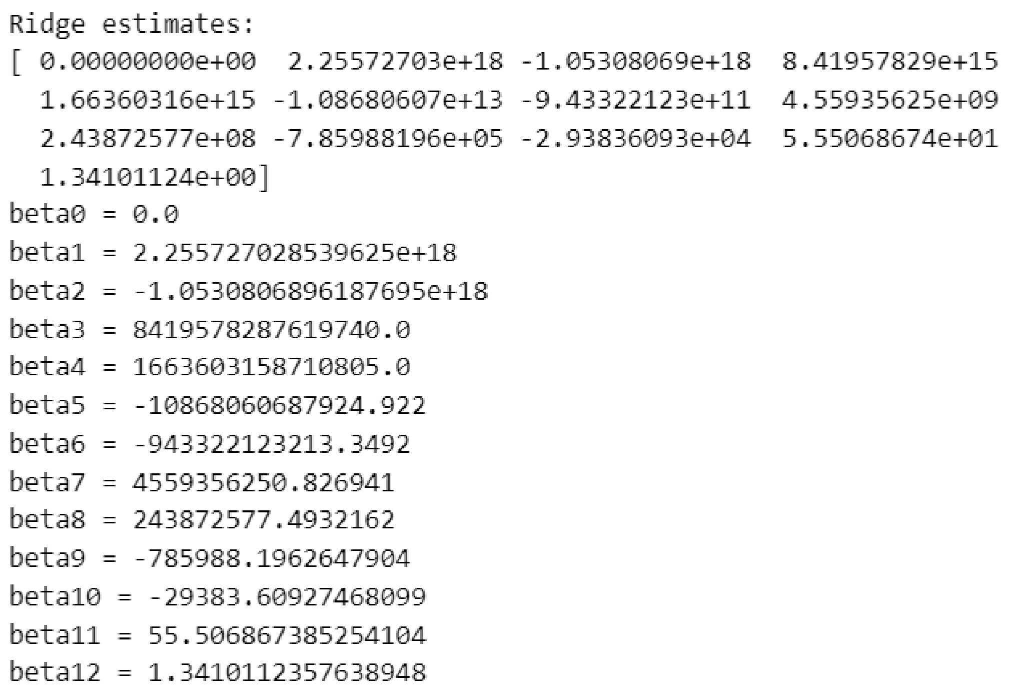
```
[[ 4.33079058e+19]
 [ 2.26474138e+18]
 [-1.05326521e+18]
 [ 8.40605881e+15]
 [ 1.66377339e+15]
 [-1.08604094e+13]
 [-9.43370967e+11]
 [ 4.55729677e+09]
 [ 2.43876121e+08]
 [-7.85724140e+05]
 [-2.93831680e+04]
 [ 5.54939354e+01]
 [ 1.34095809e+00]]
```

Figure 11: olsEstimates

```python
from sklearn.linear_model import Ridge

ridge = Ridge(alpha=1.0)
ridge.fit(X, y)

ridgeEstimates = ridge.coef_.flatten()
print("Ridge estimates:")
print (ridgeEstimates)
for i, beta in enumerate(ridgeEstimates):
    print(f"beta{i} = {beta}")
```



```
Ridge estimates:
[ 0.00000000e+00  2.25572703e+18 -1.05308069e+18  8.41957829e+15
  1.66360316e+15 -1.08680607e+13 -9.43322123e+11  4.55935625e+09
  2.43872577e+08 -7.85988196e+05 -2.93836093e+04  5.55068674e+01
  1.34101124e+00]
beta0 = 0.0
beta1 = 2.255727028539625e+18
beta2 = -1.0530806896187695e+18
beta3 = 8419578287619740.0
beta4 = 1663603158710805.0
beta5 = -10868060687924.922
beta6 = -943322123213.3492
beta7 = 4559356250.826941
beta8 = 243872577.4932162
beta9 = -785988.1962647904
beta10 = -29383.60927468099
beta11 = 55.506867385254104
beta12 = 1.3410112357638948
```

Figure 12: Ridge Estimates

```python
# ols estimate
beta = olsEstimate
x_test = np.linspace(-100, 100, 100)
# plot the data points and the fitted curve
y_test = np.array([beta[0] + beta[1]*x + beta[2]*x**2\
+ beta[3]*x**3 + beta[4]*x**4 + beta[5]*x**5 +\
beta[6]*x**6 + beta[7]*x**7 + beta[8]*x**8 +\
beta[9]*x**9 + beta[10]*x**10 + \
beta[11]*x**11 + beta[12]*x**12 for x in x_test])

fig, ax = plt.subplots(figsize=(10, 6))
ax.plot(x_test, y_test, label='Fitted curve', color='green')
ax.scatter(x_init, y_init, marker='o', s=50, label='Data points')
ax.grid()
ax.set_xlabel("x")
ax.set_ylabel("y")
ax.set_title("Scatter plot of data points and fitted curve using OLS estimates")
```

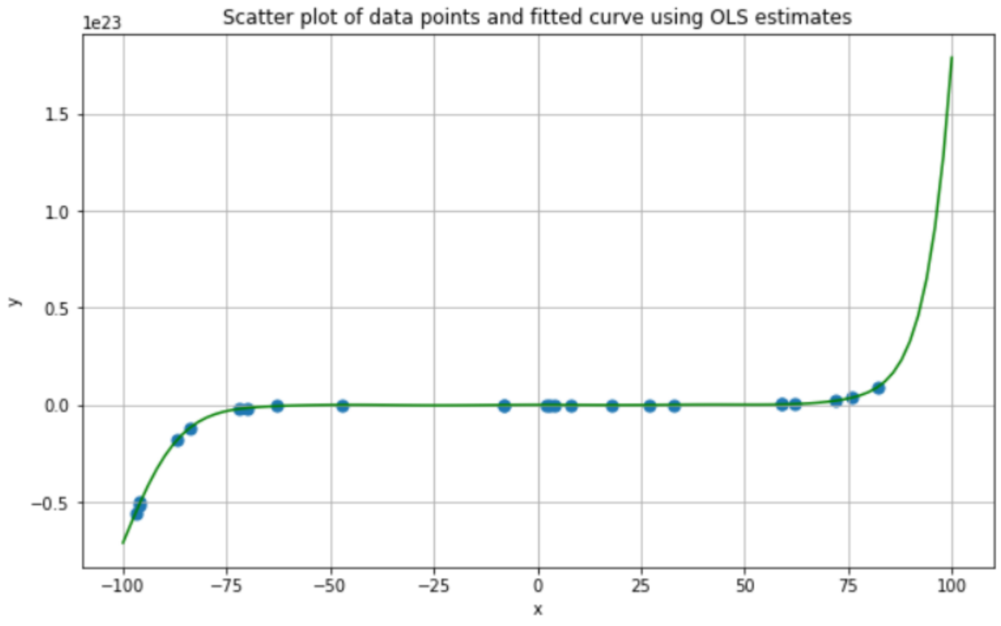


Figure 13: Scatter plot of data points and fitted curve using OLS estimates

```
# ridge estimates
beta = ridgeEstimates
x_test = np.linspace(-100, 100, 100)
# plot the data points and the fitted curve
y_test = np.array([beta[0] + beta[1]*x + beta[2]*x**2 \
+ beta[3]*x**3 + beta[4]*x**4 + beta[5]*x**5 +\
beta[6]*x**6 + beta[7]*x**7 + beta[8]*x**8 + beta[9]*x**9 \
+ beta[10]*x**10 + beta[11]*x**11 + beta[12]*x**12 for x in x_test])

fig, ax = plt.subplots(figsize=(10, 6))
ax.plot(x_test, y_test, label='Fitted curve', color='green')
ax.scatter(x_init, y_init, marker='o', s=50, label='Data points')
ax.grid()
ax.set_xlabel("x")
ax.set_ylabel("y")
ax.set_title("Scatter plot of data points and fitted curve using Ridge Estimates")
```

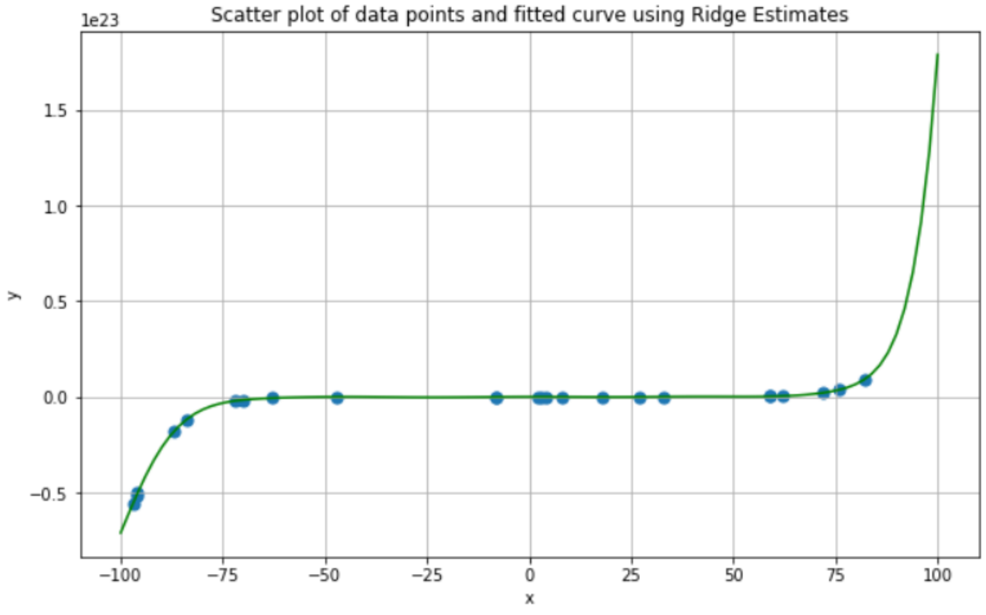


Figure 14: Scatter plot of data points and fitted curve using Ridge Estimates

2.6 ASSIGNMENT 6: Bayesian Estimates

Parameter Generated: $\zeta_1 7 = 40$**,** $\zeta_1 8 = 20$**,**$\zeta_1 9 = 61.7$

Let X_1, X_2, ..., X_{10} be a random sample from a gamma distribution with α=3 and β=$\frac{1}{\theta}$. Suppose we believe that θ follows a gamma-distribution with α=40 and β=20 and suppose we have a trial $(x_1, ..., x_n)$ with an observed $\bar{x}$=61.7.

- Find the posterior distribution of θ.
- What is the Bayes point estimate of θ associated with the square-error loss function?
- What is the Bayes point estimate of θ using the mode of the posterior distribution?

solution (240 words):

Suppose we have a random sample X_1, X_2, ..., X_{10} taken from a gamma distribution with α=3 and β=$\frac{1}{\theta}$. It is also assumed that θ follows a gamma distribution with α=40 and β=20. If we are given a trial $(x_1, ..., x_n)$ and an observed sample mean of $\bar{x}$=61.7.

Where $p(\bar{x}|\theta)$ is the likelihood function and $p(\theta)$ is the prior distribution.
The likelihood function (IU 21g) for the data is:

$$f(x;\alpha,\beta) = \frac{1}{\beta^{\alpha}\Gamma(\alpha)} x^{\alpha-1} e^{-x/\beta} \quad (\alpha,\beta > 0) \tag{2.37}$$

Where n is the sample size, x_i is the observed data, α is 3 and β is $\frac{1}{\theta}$, and Γ is the gamma function. Rearranging and substituting the values in the above equation

$$L(\theta|X) = \prod_{i=1}^{10} \frac{X_i^{3-1} e^{-\theta X_i} \theta^3}{\Gamma(3)} \tag{2.38}$$

$$L(\theta|X) == (\frac{1}{\Gamma(3)})^{10} \prod_{i=1}^{10} X_i^2 \theta^{30} e^{-\theta \sum_{i=1}^{10} X_i} \tag{2.39}$$

The prior distribution for θ is given by the Gamma distribution with parameters $\alpha = 40$ and $\beta = 20$.

$$f(\theta) = \frac{\theta^{39} e^{\frac{-\theta}{20}}}{2^{40}\Gamma(40)} \quad (where, 0 < \theta < \inf) \tag{2.40}$$

$$\prod(\theta|X) = c \times \theta^{69} e^{-\theta \sum_{i=1}^{10} X_i + \frac{1}{20}} \tag{2.41}$$

Here, C is the constant independent of θ by solving the above equation, we will get the final equation

$$\prod(\theta|X) = \theta^{70-1} e^{-\theta \sum_{i=1}^{10} X_i + \frac{1}{20}} \tag{2.42}$$

part a

To find the posterior distribution of θ, we can use Bayes' Theorem, which states that:
$p(\theta|x) \propto p(x|\theta)p(\theta)$, where $p(x|\theta)$ is the likelihood function and $p(\theta)$ is the prior distribution.
Therefore, the posterior distribution of θ is a gamma distribution with shape parameters

$$\alpha = 70 \tag{2.43}$$

$$\beta = \frac{1}{\sum_{i=1}^{10} X_i + \frac{1}{20}} \tag{2.44}$$

part b

When using the square-error loss function, the Bayes point estimate of θ is equivalent to the mean of the posterior distribution, as follows

$$B = \frac{70}{\sum_{i=1}^{10} X_i + \frac{1}{20}} = \frac{70}{10\bar{x} + \frac{1}{20}} = \frac{70}{10 \times 61.7 + \frac{1}{20}} = 0.113 \tag{2.45}$$

part c

The Bayes point estimate of θ using the mode of the posterior distribution is the mode of the posterior distribution, which is the value of θ that maximizes the posterior distribution.

The mode of the posterior distribution is

$$Mode = \frac{70 - 1}{10\bar{x} + \frac{1}{20}} = \frac{69}{10 \times 61.7 + \frac{1}{20}} = 0.111 \tag{2.46}$$

The code for the entire task is given below

```python
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.patches as mpatches
from scipy.stats import gamma

num_points = 10
prior_shape = 3
prior_scale = 2 / gamma.ppf(0.001, 3, loc=0, scale=2)
prior_x = np.linspace(gamma.ppf(0.001, prior_shape, loc=0, scale=prior_scale),
                      gamma.ppf(0.999, prior_shape, loc=0, scale=prior_scale), \
                      num_points)
prior_y = gamma.pdf(prior_x, prior_shape, loc=0, scale=prior_scale)

post_shape = 70
post_scale = 1 / np.sum(prior_x) + 0.05
post_x = np.linspace(gamma.ppf(0.001, post_shape, loc=0, scale=post_scale),\
                          gamma.ppf(0.999, post_shape, loc=0, scale=post_scale),\
                          num_points)
post_y = gamma.pdf(post_x, post_shape, loc=0, scale=post_scale)

fig, ax = plt.subplots()
fig.suptitle('Gamma Distribution Posterior and Prior', fontsize=20)
fig.set_size_inches(10, 5)

ax.plot(prior_x, prior_y, 'r-', label="Prior, Shape = {:.2f}, \
Scale = {:.2f}".format(prior_shape, prior_scale))
ax.plot(post_x, post_y, 'b-', label="Posterior, Shape = {:.2f}, \
Scale = {:.2f}".format(post_shape, post_scale))
ax.legend(loc='upper right')
ax.set_ylabel(r'$f(x;\alpha,\beta)= \frac {1}{\beta^{\alpha} \
\Gamma(\alpha)} x^{\alpha -1} e^{-x/\beta} \ (\alpha, \beta > 0)$', fontsize=15)
ax.set_xlabel('X Values', fontsize=15)
ax.grid()

plt.tight_layout()
plt.subplots_adjust(top=0.9)
plt.show()
```

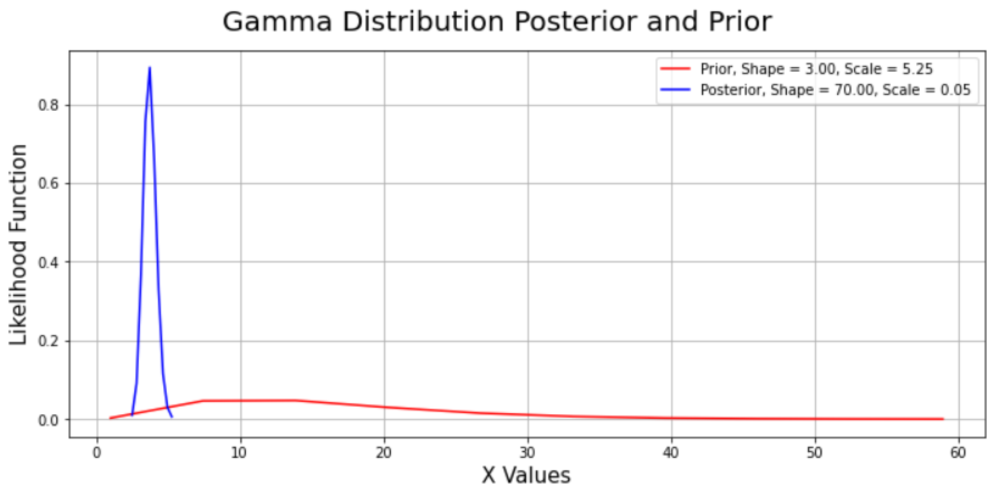


Figure 15: Posterior and Prior of Gamma Distribution

2.7 Conclusion

I completed each task in the Workbook by utilizing various statistical methods presented in the coursebook. I performed calculations, outlined steps, and included relevant figures in each assignment task respectively with the codec packs.

Bibliography

[IU 21a] IU GmbH. *Advanced Statistics*. IU Internationale Hochschule GmbH, Probability and Descriptive Statistics, DLBDSSPDS01, 2022: Unit 4, Random Variables, Poisson distribution, Exponential distribution, pp. 145-149 (PDF version 001-2022-0615), 2021.

[IU 21b] IU GmbH. *Advanced Statistics*. IU Internationale Hochschule GmbH, Advanced Statistics, DLMDSAS01, 2021: Unit 3.3 Poisson, Gamma-Poisson Exponential Distribution, pp. 79-83 (PDF version 001-2021-1026)., 2021.

[IU 21c] IU GmbH. *Advanced Statistics*. IU Internationale Hochschule GmbH, Statistics – Probability and Descriptive Statistics, unit 4, DLBDSSPDS01, 2021. Int Univ of Appl Sciences, 2021.

[IU 21d] IU GmbH. *Advanced Statistics*. IU Internationale Hochschule GmbH, Statistics – Probability and Descriptive Statistics, unit 4, DLBDSSPDS01, 2021. Int Univ of Appl Sciences, 2021.

[IU 21e] IU GmbH. *Advanced Statistics*. IU Internationale Hochschule GmbH, Advanced Statistics, DLMDSAS01, 2021: Unit 7.1 Hypothesis Testing, pp. 198 (PDF version 001-2021-1026)., 2021.

[IU 21f] IU GmbH. *Advanced Statistics*. IU Internationale Hochschule GmbH, Advanced Statistics, DLMDSAS01, 2021: Unit 6 Parameter Estimation, pp. 172-177 (PDF version 001-2021-1026)., 2021.

[IU 21g] IU GmbH. *Advanced Statistics*. IU Internationale Hochschule GmbH, Advanced Statistics, DLMDSAS01, 2021: Unit 4, Bayesian Statistics, pp. 101-124 (PDF version 001-2021-1026), 2021.

INTERNATIONAL
UNIVERSITY OF
APPLIED SCIENCES

Eidesstattliche Erklärung

I hereby certify...

....	....
Place, date	Signature