PROBABILITY AND RANDOM PROCESSES

EE325 PROGRAMMING ASSIGNMENT - 2

Aminesh Gogia – 23B1210
Rushabh Gayakwad – 23B2464
Jaswin Reddy M – 23B1289

Part 1

The objective is to determine the joint probability mass function (PMF) of X and Y, the marginal PMF of Y, and ultimately to calculate a suitable premium α that ensures a low probability of a negative surplus $S = \alpha X - \Omega Y$, where Ω is the insured payout amount. Following the derivation, we approximate distributions using Gaussian approximations for large values of λ and small values of p, as specified in the assignment. We then compute the probability $P(S < a)$ for different threshold values a and determine the minimum α that satisfies the requirement $P(S < a) < 0.01$.

Let $X \sim \text{Poisson}(\lambda)$.
We are given:

$$P(X = x, Y = y) = P(Y = y|X = x) \cdot P(X = x)$$

Also,

$$P(Y = y|X = x) = \binom{x}{y} p^y (1-p)^{x-y}$$

Thus, the joint PMF is:

$$P(X = x, Y = y) = \binom{x}{y} p^y (1-p)^{x-y} \cdot \frac{\lambda^x e^{-\lambda}}{x!}$$

Expanding the binomial coefficient:

$$P(X = x, Y = y) = \frac{x!}{y!(x-y)!} p^y (1-p)^{x-y} \cdot \frac{\lambda^x e^{-\lambda}}{x!}$$

Simplifying, we get:

$$P(X = x, Y = y) = \frac{\lambda^x e^{-\lambda}}{y!(x-y)!} p^y (1-p)^{x-y}$$

To find the marginal PMF of Y, denoted $f_Y(y)$, we sum over all $x \geq y$:

$$f_Y(y) = \sum_{x=y}^{\infty} \frac{\lambda^x e^{-\lambda}}{y!(x-y)!} p^y (1-p)^{x-y}$$

On simplifying, we get

$$f_Y(y) = \frac{(\lambda p)^y e^{-\lambda p}}{y!}$$

i.e. $Y \sim \text{Poisson}(\lambda p)$.

Now, Poisson is a limiting Binomial, with $n \to \infty$. Also, the Binomial variable is a sum of i.i.d. Bernoullis. Applying the Central Limit Theorem to the sum of Bernoullis, we can approximate the Poisson (their sum) to be distributed as:

$$X \approx N(\lambda, \lambda) \quad \text{and} \quad Y \approx N(\lambda p, \lambda p(1-p))$$

The surplus S is defined as:

$$S = \alpha X - \Omega Y$$

where:

- $X \sim \text{Poisson}(\lambda)$, the number of policies sold in a year.
- $Y \sim \text{Poisson}(\lambda p)$, the number of claims in the year.

The **expected value** $E(S)$ is given by:

$$E(S) = E(\alpha X - \Omega Y) = \alpha E(X) - \Omega E(Y)$$

Since $X \sim \text{Poisson}(\lambda)$, we have $E(X) = \lambda$.
For $Y \sim \text{Poisson}(\lambda p)$, the expected value $E(Y) = \lambda p$.
Thus:

$$E(S) = \alpha\lambda - \Omega\lambda p$$

The **variance of** S, $\text{Var}(S)$, is given by:

$$\text{Var}(S) = \text{Var}(\alpha X - \Omega Y) = \alpha^2 \text{Var}(X) + \Omega^2 \text{Var}(Y) - 2\alpha\Omega \,\text{Cov}(X, Y).$$

Since $X \sim \text{Poisson}(\lambda)$, we have $\text{Var}(X) = \lambda$.
For $Y \sim \text{Poisson}(\lambda p)$, we have $\text{Var}(Y) = \lambda p$.
The covariance between X and Y, $\text{Cov}(X, Y)$, is given by:

$$\text{Cov}(X, Y) = E((X - \lambda)(Y - \lambda p)) = p\lambda.$$

Therefore:

$$\text{Var}(S) = \alpha^2\lambda + \Omega^2\lambda p - 2\alpha\Omega p\lambda.$$

To satisfy $\Pr(S < a) < 0.01$, we standardize S by subtracting $E(S)$ and dividing by $\sqrt{\text{Var}(S)}$, and set it equal to the z-score (inverse of the Gaussian CDF) for a probability of 0.01. Let $Z \sim \mathcal{N}(0, 1)$ be a standard normal variable. Then we want:

$$\Pr\left(\frac{S - E(S)}{\sqrt{\text{Var}(S)}} < \frac{a - E(S)}{\sqrt{\text{Var}(S)}}\right) < 0.01.$$

For a probability of 0.01, the z-score is approximately -2.3263. Thus:

$$\frac{a - (\alpha\lambda - \Omega\lambda p)}{\sqrt{\alpha^2\lambda + \Omega^2\lambda p - 2\alpha\Omega p\lambda}} < -2.3263 = Z.$$

This setup can then be solved numerically to find α, the minimum premium:

$$a - \alpha\lambda + \Omega\lambda p < Z\sqrt{\alpha^2\lambda + \Omega^2\lambda p - 2\alpha\Omega p\lambda}.$$

Premiums Calculated for different λ, p

λ	p	a	α
10000	0.01	10	0.12416
10000	0.01	0	0.12316
10000	0.01	-10	0.12216
10000	0.01	-20	0.12116
10000	0.01	-30	0.12016
10000	0.01	-100	0.11315

Table 1: Minimum Premium P for $\lambda = 10000$, $p = 0.01$

λ	p	a	α
10000	0.02	10	0.23358
10000	0.02	0	0.23258
10000	0.02	-10	0.23158
10000	0.02	-20	0.23058
10000	0.02	-30	0.22958
10000	0.02	-100	0.22258

Table 2: Minimum Premium P for $\lambda = 10000$, $p = 0.02$

λ	p	a	α
100000	0.01	10	0.10742
100000	0.01	0	0.10732
100000	0.01	-10	0.10722
100000	0.01	-20	0.10712
100000	0.01	-30	0.10702
100000	0.01	-100	0.10632

Table 3: Minimum Premium P for $\lambda = 100000$, $p = 0.01$

λ	p	a	α
100000	0.02	10	0.21040
100000	0.02	0	0.21030
100000	0.02	-10	0.21020
100000	0.02	-20	0.21010
100000	0.02	-30	0.21000
100000	0.02	-100	0.20930

Table 4: Minimum Premium P for $\lambda = 100000$, $p = 0.02$

Part-2

Cross-subsidy refers to compensating for loss in a certain business activity by another profitable activity. In Part 2, the target is to ensure profitability over the two demographics without allowing too much loss in each. Cross-subsidy targets allow for flexible business planning and price (here, premium) setting, as we do not need to set precise achievement targets for multiple demographics. The idea is to make up for high-risk customers by charging the low-risk customers a higher premium.

Cross-subsidy is, however, slightly risky, as overall profitability might be due to one of the demographics, while the other might be loss-making. The loss being more than a certain value might lead to the company running out of business in the particular demographic, thus narrowing the customer segment. The issue with cross-subsidy might be that a lot more high-risk insurers sign up for their low-premium insurance scheme as compared to the low-risk customers who sign up. We thus risk 'breaking the pot'.

We want to charge the minimum possible premiums for both demographics. We want the minimum α_1, α_2 that satisfy $P(S_1 < a_1) < 0.01$, $P(S_2 < a_2) < 0.01$ and $P(S < a) < 0.01$ simultaneously.

Using the same method as Part 1, we calculate minimum premiums that satisfy $P(S_i < a_i) < 0.01$ for the individual demographics. Now, starting with these values (as minimas for iterating), we try to find a suitable pair (α_1, α_2) that satisfy $P(S < a) < 0.01$.

Coincidentally, the minimum (α_1, α_2) individually satisfying $P(S_i < a_i) < 0.01$ also satisfy $P(S < a) < 0.01$. For $(\alpha_1, \alpha_2) = (0.2093, 2.4578)$, all three conditions are satisfied. Thus, this is the premium we will charge.

If the (α_1, α_2) that satisfied the $P(S_i < a_i) < 0.01$ conditions individually did not satisfy $P(S < a) < 0.01$, we would have to test different (α_1, α_2), starting with $(0.2093, 2.4578)$:

- After considering the propensities of each demographic to pay, we would increment both α_1 and α_2 gradually by suitable (different) amounts to see if the z-score inequality is satisfied.
- For example, we could alternatively increase α_1 by 0.0001 and α_2 by 0.0002 to eventually find a suitable pair (α_1, α_2)
- This is critical, as increasing the premiums for just one of the demographics might reduce the number of insurers from that particular demographic.

Assuming X, Y to be independent would make $\text{Cov}(X, Y) = 0$. This could be problematic as our Gaussian Approximation would go wrong when we apply the Central Limit Theorem with the wrong standard deviation. We would get slightly different values of α_1, α_2, which would be close to those obtained by not assuming independence but might not be the same. The values for premiums obtained (assuming independence) are the same up to the fourth decimal place! Even though assuming independence does not seem right, if we only worry up to the fourth decimal place, it does not exactly cause issues.

Part-3

Let $\hat{\lambda}$ represent the estimated Poisson parameter, calculated as the sample mean of the parameters over the years:

$$\hat{\lambda} = \frac{1}{n} \sum_{i=1}^{n} \lambda_i$$

where λ_i is the parameter for each year and n is the number of years.
The estimated variance, denoted by $\hat{\sigma}^2$, is given by:

$$\hat{\sigma}^2 = \frac{s^2 \cdot n}{n-1}$$

where s^2 is the sample variance, n is the number of years:

$$s^2 = \frac{1}{n}\sum_{i=1}^{n}(\lambda_i - \hat{\lambda})^2.$$

The actual parameter λ is estimated to lie within the interval (with 95% confidence):

$$\hat{\lambda} \pm \frac{2 \cdot \hat{\sigma}}{\sqrt{n}}$$

We similarly estimate the Poisson parameter $(\widehat{\lambda p})$ for the number of claims.

We now find $\hat{p}$ by dividing the estimates of λp by 99.9999% estimates of λ. (Lower Limit/Upper Limit, Upper Limit/Lower Limit)

- For the 20 year samples, the 95% Confidence Interval of λ is $(99838.266, 100130.734)$ and that of p is $(0.019707, 0.020307)$
- For the 100 year samples, the 95% Confidence Interval of λ is $(100001.056, 100119.424)$ and that of p is $(0.019859, 0.020116)$

Discretising the intervals of p and λ, we computed intervals for α:

a	α_{min}	α_{max}
10	0.20739	0.21356
0	0.20729	0.21346
-10	0.20719	0.21336
-20	0.20710	0.21326
-30	0.20700	0.21316
-100	0.20630	0.21246

Table 5: Minimum and Maximum Premium considering 20 year data

a	α_{min}	α_{max}
10	0.20896	0.21160
0	0.20886	0.21150
-10	0.20876	0.21140
-20	0.20866	0.21130
-30	0.20856	0.21120
-100	0.20786	0.21050

Table 6: Minimum and Maximum Premium considering 100 year data

Insurers	Claims
100427	1957
99988	1994
100290	2062
99726	1876
100051	2080
100173	2021
100121	1973
99904	1918
99231	1908
99973	1958
99972	1939
99416	1977
100393	1998
99895	1961
100398	2033
99796	2016
99892	1969
100217	1974
100348	2100
100032	2049
100033	1957
99674	1950
100070	1948
100344	2080
99967	2068
100350	2088
99873	2022
100711	1950
100001	2016
100467	2076
100868	2038
100187	2096
99930	2076
99975	1978
100381	1984
99674	2039
99994	2079
99874	1987
99956	2049
100362	2071
100189	2006
99914	2009
99868	2012
99810	1959
99795	2034
100079	1949
99329	1974
99747	1892
100246	2074
100235	1985
99842	2015
99895	2000

99506	1973
100227	1952
100107	1978
100039	1995
100026	1999
99754	2042
100077	2051
100114	1956
99861	1940
100408	2009
99890	1969
100211	1964
100031	1994
99412	1937
99631	2065
99947	1963
99773	1980
100364	1994
99702	1958
100312	1986
100146	2016
99759	2015
99946	2002
99937	1976
100117	1944
99662	1923
100278	1959
99967	1911
100230	1970
100502	1895
100166	2060
100205	2090
100328	2128
100137	1994
100342	1966
100553	1960
99822	1995
100651	2042
100322	2006
100536	2022
99494	2026
100219	1956
100218	2048
100479	2073
100261	1950
100292	2068
100195	2048
99985	2028

Insurers	Claims
100168	2106
99473	2042
99795	2030
100241	1956
100268	1951
100322	2005
99973	1970
100222	2022
99747	2036
100157	1963
99698	2128
99818	2023
99464	1953
99882	1934
100242	1961
99411	1991
99916	1949
100531	1957
100414	2024
99948	2005

ee325-asgn-4-complete

November 3, 2024

1 EE325 Programming Assignment - 4

1.0.1 Q1. Deciding Premium to reduce probability of breaking the pot

[1]:
```
import numpy as np
import pandas as pd
from scipy.stats import norm

def normal_dist(lam, omega, p, a):
    # Gaussian approximation z-score for Pr(S < a) < 0.01
    z_critical = norm.ppf(0.01)
    results = []

    for lam_val in lam:
        for p_val in p:
            for a_val in a:
                alpha = 0.00001  # A starting value
                while True:
                    mean_S = (alpha * lam_val) - (omega * lam_val * p_val)
                    var_S = (lam_val * alpha**2) + (lam_val * p_val* omega**2)␣
 ↪- (2 * alpha * omega * lam_val * p_val)
                    std_dev_S = np.sqrt(var_S)

                    lhs = z_critical * std_dev_S
                    rhs = a_val - mean_S

                    if lhs >= rhs:
                        results.append((lam_val, p_val, a_val, alpha))
                        break

                    alpha += 0.00001
    return results
```

[4]:
```
# Part 1: Single Demographic Analysis
omega = 10
lam = [10000, 100000]
p = [0.01, 0.02]
a = [10, 0, -10, -20, -30, -100]
```

```
min_premium = normal_dist(lam, omega, p, a)

for result in min_premium:
    lam_val, p_val, a_val, alpha = result
    print(f"Minimum Premium P for  ={lam_val}, p={p_val}, a={a_val}, alpha:␣
 ↪{alpha:.5f}")
```

```
Minimum Premium P for  =10000, p=0.01, a=10, alpha: 0.12416
Minimum Premium P for  =10000, p=0.01, a=0, alpha: 0.12316
Minimum Premium P for  =10000, p=0.01, a=-10, alpha: 0.12216
Minimum Premium P for  =10000, p=0.01, a=-20, alpha: 0.12116
Minimum Premium P for  =10000, p=0.01, a=-30, alpha: 0.12016
Minimum Premium P for  =10000, p=0.01, a=-100, alpha: 0.11315
Minimum Premium P for  =10000, p=0.02, a=10, alpha: 0.23358
Minimum Premium P for  =10000, p=0.02, a=0, alpha: 0.23258
Minimum Premium P for  =10000, p=0.02, a=-10, alpha: 0.23158
Minimum Premium P for  =10000, p=0.02, a=-20, alpha: 0.23058
Minimum Premium P for  =10000, p=0.02, a=-30, alpha: 0.22958
Minimum Premium P for  =10000, p=0.02, a=-100, alpha: 0.22258
Minimum Premium P for  =100000, p=0.01, a=10, alpha: 0.10742
Minimum Premium P for  =100000, p=0.01, a=0, alpha: 0.10732
Minimum Premium P for  =100000, p=0.01, a=-10, alpha: 0.10722
Minimum Premium P for  =100000, p=0.01, a=-20, alpha: 0.10712
Minimum Premium P for  =100000, p=0.01, a=-30, alpha: 0.10702
Minimum Premium P for  =100000, p=0.01, a=-100, alpha: 0.10632
Minimum Premium P for  =100000, p=0.02, a=10, alpha: 0.21040
Minimum Premium P for  =100000, p=0.02, a=0, alpha: 0.21030
Minimum Premium P for  =100000, p=0.02, a=-10, alpha: 0.21020
Minimum Premium P for  =100000, p=0.02, a=-20, alpha: 0.21010
Minimum Premium P for  =100000, p=0.02, a=-30, alpha: 0.21000
Minimum Premium P for  =100000, p=0.02, a=-100, alpha: 0.20930
```

1.0.2 Q2. Cross-Subsidy Analysis and Setting Premiums for two demographics

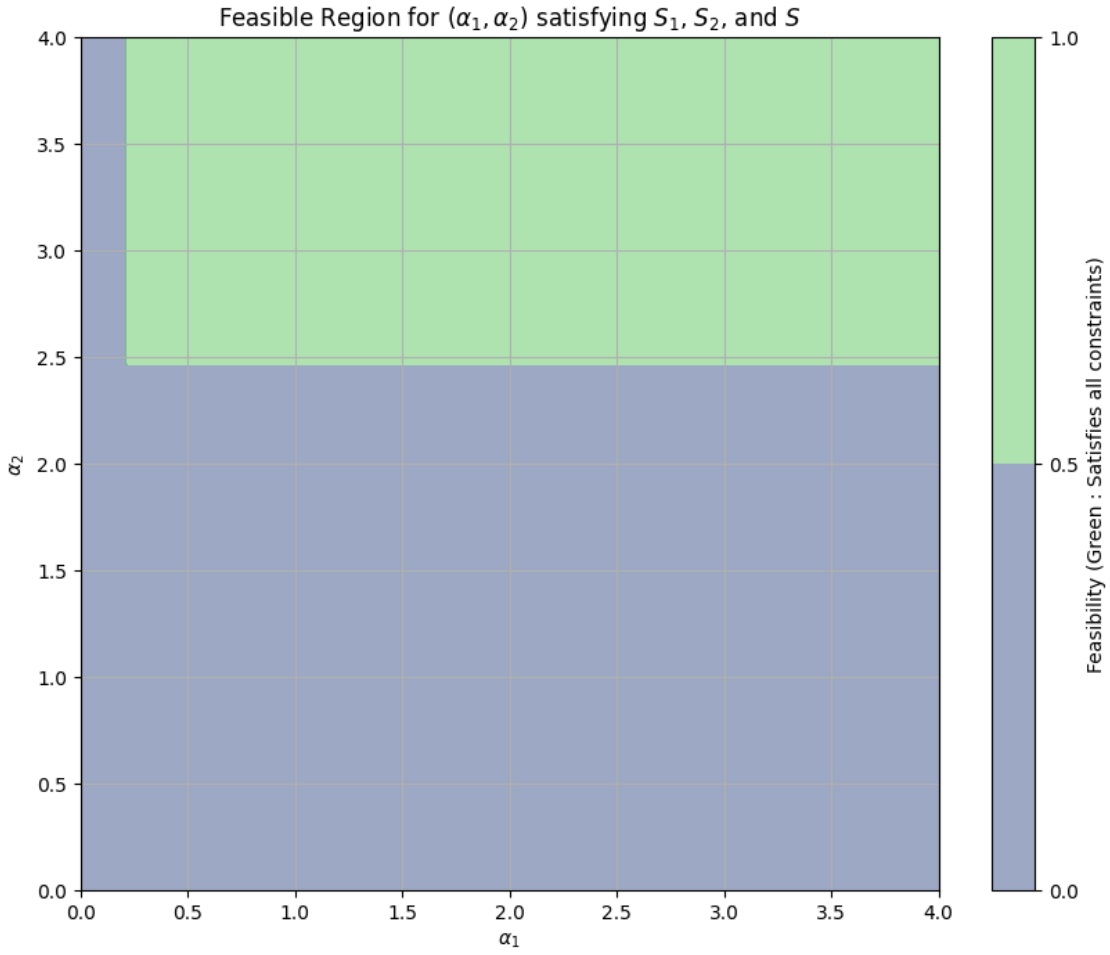
[3]:
```
lambda_1, lambda_2 = 100000, 5000
omega_1, omega_2 = 10, 100
p1, p2 = 0.02, 0.02
a1, a2, a3 = -100, -15, 105

r1 = normal_dist([lambda_1], omega_1, [p1], [a1])[0]
r2 = normal_dist([lambda_2], omega_2, [p2], [a2])[0]

l1, p1, a1, alpha_1 = r1
l2, p2, a2, alpha_2 = r2
```

```
mean_S = (alpha_1 * l1) - (omega_1 * l1 * p1) + (alpha_2 * l2) - (omega_2 * l2␣
 ↪* p2)
var_S = (l1 * alpha_1**2) + (l1 * p1 * omega_1**2) - (2 * alpha_1 * omega_1 *␣
 ↪l1 * p1) + \
       (l2 * alpha_2**2) + (l2 * p2 * omega_2**2) - (2 * alpha_2 * omega_2 *␣
 ↪l2 * p2)
std_dev_S = np.sqrt(var_S)

z_critical = norm.ppf(0.01)
lhs = z_critical * std_dev_S
rhs = a3 - mean_S

print(f"LHS: {lhs}, RHS: {rhs}")
print(f"Probability: {norm.cdf(rhs/std_dev_S)}")
print(f"Minimum premium for both demographics: alpha_1={alpha_1:.5f},␣
 ↪alpha_2={alpha_2:.5f}") ## To satisfy earning conditions for both␣
 ↪demographics separately
```

```
LHS: -2523.9070498670685, RHS: -3114.200000045026
Probability: 0.002049533673634662
Minimum premium for both demographics: alpha_1=0.20930, alpha_2=2.45784
```

[4]:
```
import matplotlib.pyplot as plt


alpha_1_vals = np.linspace(0, 4, 200)
alpha_2_vals = np.linspace(0, 4, 200)
alpha_1_grid, alpha_2_grid = np.meshgrid(alpha_1_vals, alpha_2_vals)


mean_S1 = (alpha_1_grid * lambda_1) - (omega_1 * lambda_1 * p1)
var_S1 = (lambda_1 * alpha_1_grid**2) + (lambda_1 * p1 * omega_1**2) - (2 *␣
 ↪alpha_1_grid * omega_1 * lambda_1 * p1)
std_dev_S1 = np.sqrt(var_S1)
lhs_S1 = z_critical * std_dev_S1
rhs_S1 = a1 - mean_S1

mean_S2 = (alpha_2_grid * lambda_2) - (omega_2 * lambda_2 * p2)
var_S2 = (lambda_2 * alpha_2_grid**2) + (lambda_2 * p2 * omega_2**2) - (2 *␣
 ↪alpha_2_grid * omega_2 * lambda_2 * p2)
std_dev_S2 = np.sqrt(var_S2)
lhs_S2 = z_critical * std_dev_S2
rhs_S2 = a2 - mean_S2

mean_S = mean_S1 + mean_S2
var_S = var_S1 + var_S2
std_dev_S = np.sqrt(var_S)
```

```
lhs_S = z_critical * std_dev_S
rhs_S = a3 - mean_S

# Determining feasible points where all three inequalities hold
feasible = (lhs_S1 >= rhs_S1) & (lhs_S2 >= rhs_S2) & (lhs_S >= rhs_S)

# Plotting the feasible region
plt.figure(figsize=(10, 8))
plt.contourf(alpha_1_grid, alpha_2_grid, feasible, levels=1, alpha=0.5)
plt.xlabel(r'$\alpha_1$')
plt.ylabel(r'$\alpha_2$')
plt.title(r'Feasible Region for $(\alpha_1, \alpha_2)$ satisfying $S_1$, $S_2$,␣
 ↪and $S$')
plt.grid(True)
plt.colorbar(label='Feasibility (Green : Satisfies all constraints)')
plt.show()
```



1.0.3 Q3.Parameter Estimation and recalculating the Premium

[5]:
```
def est_params(df):
    n = df['Insurers'].size
    l_mean = df['Insurers'].mean()
    l_var = df['Insurers'].var() * n / (n - 1)

    up_l = norm.ppf(1 - 0.025) * np.sqrt(l_var) / np.sqrt(n) + l_mean
    lo_l = -norm.ppf(1 - 0.025) * np.sqrt(l_var) / np.sqrt(n) + l_mean

    c_num = df['Claims'].size
    c_var = df['Claims'].var() * n / (n - 1)
    c_mean = df['Claims'].mean()

    up_lp = norm.ppf(1 - 0.025) * np.sqrt(c_var) / np.sqrt(c_num) + c_mean
    lo_lp = -norm.ppf(1 - 0.025) * np.sqrt(c_var) / np.sqrt(c_num) + c_mean

    sure_l_up = norm.ppf(1 - 0.00001) * np.sqrt(l_var) / np.sqrt(n) + l_mean
    sure_l_low = -norm.ppf(1 - 0.00001) * np.sqrt(l_var) / np.sqrt(n) + l_mean
    p_lo = lo_lp / sure_l_up
    p_up = up_lp / sure_l_low

    return (lo_l, up_l), (lo_lp, up_lp), (p_lo, p_up)
```

[6]:
```
df_20 = pd.read_csv('generated_dataframe_20.csv')
df_100 = pd.read_csv('generated_dataframe_100.csv')

print("N = 20:")
l_ci_20, lp_ci_20, p_ci_20 = est_params(df_20)
print(f"Lambda 95% CI: {l_ci_20}")
print(f"Lambda*p 95% CI: {lp_ci_20}")
print(f"p 95% CI: {p_ci_20}")

print("\nN = 100:")
l_ci_100, lp_ci_100, p_ci_100 = est_params(df_100)
print(f"Lambda 95% CI: {l_ci_100}")
print(f"Lambda*p 95% CI: {lp_ci_100}")
print(f"p 95% CI: {p_ci_100}")
```

```
N = 20:
Lambda 95% CI: (99838.26630150972, 100130.73369849028)
Lambda*p 95% CI: (1976.6774563721715, 2023.9225436278284)
p 95% CI: (0.019707120084168285, 0.020306990923310966)

N = 100:
Lambda 95% CI: (100001.0564309468, 100119.42356905321)
Lambda*p 95% CI: (1989.689869925697, 2010.310130074303)
p 95% CI: (0.019859359798009738, 0.020116890166207747)
```

[8]:
```
# Updated minimum premium calculations
print("Updated Minimum Premiums (N=20):")
new_min_20 = normal_dist([l_ci_20[0], l_ci_20[1]], omega, [p_ci_20[0],␣
 ↪p_ci_20[1]], a=[10, 0, -10, -20, -30, -100])
for r in new_min_20:
    l, p, a_val, alpha = r
    print(f"Minimum P for  ={l:.2f}, p={p:.5f}, a={a_val},  : {alpha:.4f}")

print("\nUpdated Minimum Premiums (N=100):")
new_min_100 = normal_dist([l_ci_100[0], l_ci_100[1]], omega, [p_ci_100[0],␣
 ↪p_ci_100[1]], a=[10, 0, -10, -20, -30, -100])
for r in new_min_100:
    l, p, a_val, alpha = r
    print(f"Minimum P for  ={l:.2f}, p={p:.5f}, a={a_val},  : {alpha:.4f}")
```

```
Updated Minimum Premiums (N=20):
Minimum P for  =99838.27, p=0.01971, a=10,  : 0.2074
Minimum P for  =99838.27, p=0.01971, a=0,  : 0.2073
Minimum P for  =99838.27, p=0.01971, a=-10,  : 0.2072
Minimum P for  =99838.27, p=0.01971, a=-20,  : 0.2071
Minimum P for  =99838.27, p=0.01971, a=-30,  : 0.2070
Minimum P for  =99838.27, p=0.01971, a=-100,  : 0.2063
Minimum P for  =99838.27, p=0.02031, a=10,  : 0.2136
Minimum P for  =99838.27, p=0.02031, a=0,  : 0.2135
Minimum P for  =99838.27, p=0.02031, a=-10,  : 0.2134
Minimum P for  =99838.27, p=0.02031, a=-20,  : 0.2133
Minimum P for  =99838.27, p=0.02031, a=-30,  : 0.2132
Minimum P for  =99838.27, p=0.02031, a=-100,  : 0.2125
Minimum P for  =100130.73, p=0.01971, a=10,  : 0.2074
Minimum P for  =100130.73, p=0.01971, a=0,  : 0.2073
Minimum P for  =100130.73, p=0.01971, a=-10,  : 0.2072
Minimum P for  =100130.73, p=0.01971, a=-20,  : 0.2071
Minimum P for  =100130.73, p=0.01971, a=-30,  : 0.2070
Minimum P for  =100130.73, p=0.01971, a=-100,  : 0.2063
Minimum P for  =100130.73, p=0.02031, a=10,  : 0.2135
Minimum P for  =100130.73, p=0.02031, a=0,  : 0.2134
Minimum P for  =100130.73, p=0.02031, a=-10,  : 0.2133
Minimum P for  =100130.73, p=0.02031, a=-20,  : 0.2132
Minimum P for  =100130.73, p=0.02031, a=-30,  : 0.2132
Minimum P for  =100130.73, p=0.02031, a=-100,  : 0.2125

Updated Minimum Premiums (N=100):
Minimum P for  =100001.06, p=0.01986, a=10,  : 0.2090
Minimum P for  =100001.06, p=0.01986, a=0,  : 0.2089
Minimum P for  =100001.06, p=0.01986, a=-10,  : 0.2088
Minimum P for  =100001.06, p=0.01986, a=-20,  : 0.2087
Minimum P for  =100001.06, p=0.01986, a=-30,  : 0.2086
```

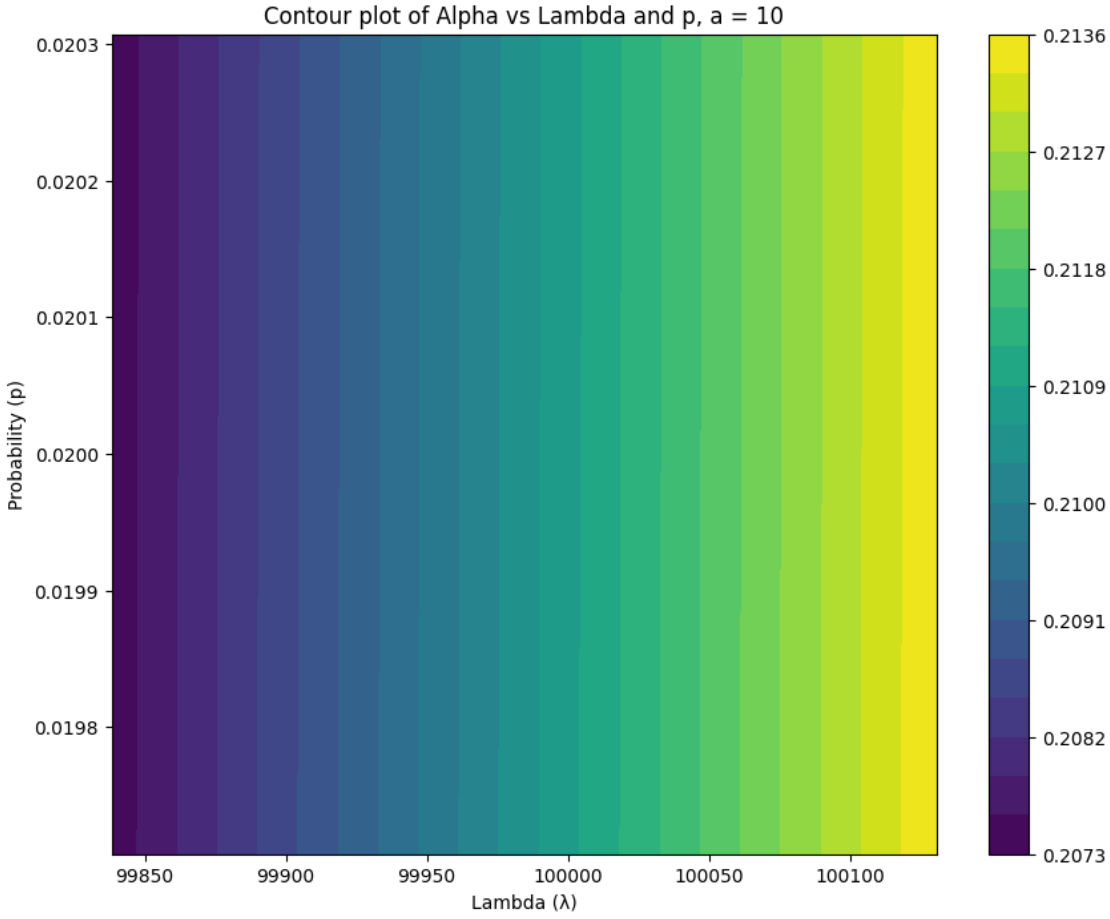
```
Minimum P for  =100001.06, p=0.01986, a=-100,  : 0.2079
Minimum P for  =100001.06, p=0.02012, a=10,  : 0.2116
Minimum P for  =100001.06, p=0.02012, a=0,  : 0.2115
Minimum P for  =100001.06, p=0.02012, a=-10,  : 0.2114
Minimum P for  =100001.06, p=0.02012, a=-20,  : 0.2113
Minimum P for  =100001.06, p=0.02012, a=-30,  : 0.2112
Minimum P for  =100001.06, p=0.02012, a=-100,  : 0.2105
Minimum P for  =100119.42, p=0.01986, a=10,  : 0.2090
Minimum P for  =100119.42, p=0.01986, a=0,  : 0.2089
Minimum P for  =100119.42, p=0.01986, a=-10,  : 0.2088
Minimum P for  =100119.42, p=0.01986, a=-20,  : 0.2087
Minimum P for  =100119.42, p=0.01986, a=-30,  : 0.2086
Minimum P for  =100119.42, p=0.01986, a=-100,  : 0.2079
Minimum P for  =100119.42, p=0.02012, a=10,  : 0.2116
Minimum P for  =100119.42, p=0.02012, a=0,  : 0.2115
Minimum P for  =100119.42, p=0.02012, a=-10,  : 0.2114
Minimum P for  =100119.42, p=0.02012, a=-20,  : 0.2113
Minimum P for  =100119.42, p=0.02012, a=-30,  : 0.2112
Minimum P for  =100119.42, p=0.02012, a=-100,  : 0.2105
```

[34]:
```python
import matplotlib.pyplot as plt

def contour_plot(lam_range, p_range, a, omega):

    # lam_range = np.linspace(99838.26630150972, 100130.73369849028, 50)  # Example range for lambda
    # p_range = np.linspace(0.019859359798009738, 0.020116890166207747, 50)  # Example range for p
    # omega = 10
    a_val = [a]

    results = normal_dist(lam_range, omega, p_range, a_val)
    alpha_vals = np.array([result[3] for result in results])

    alpha_min, alpha_max = alpha_vals.min(), alpha_vals.max()

    lam_grid, p_grid = np.meshgrid(lam_range, p_range)
    alpha_grid = alpha_vals.reshape(len(p_range), len(lam_range))
    plt.figure(figsize=(10, 8))
    contour = plt.contourf(lam_grid, p_grid, alpha_grid, levels=20, cmap='viridis')
    plt.colorbar(contour)
    plt.xlabel('Lambda ( )')
    plt.ylabel('Probability (p)')
    plt.title(f'Contour plot of Alpha vs Lambda and p, a = {a}')
    plt.show()
```

```
    return (alpha_min, alpha_max)
```

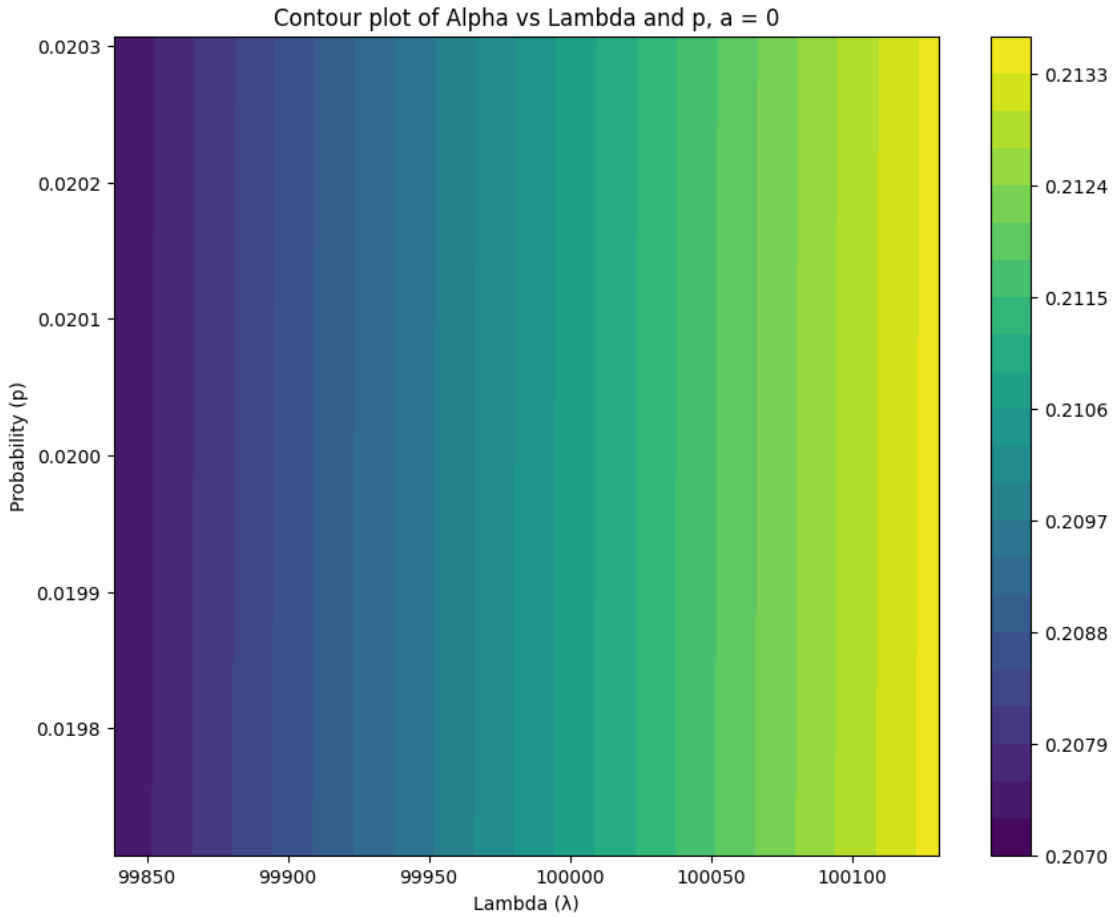
[35]:
```
lam_range_20 = np.linspace(99838.26630150972, 100130.73369849028, 50)  #␣
 ↪Example range for lambda
p_range_20 = np.linspace(0.019707120084168285, 0.020306990923310966, 50)  #␣
 ↪Example range for p
omega = 10
```

[36]:
```
contour_plot(lam_range_20, p_range_20, 10, omega)
```



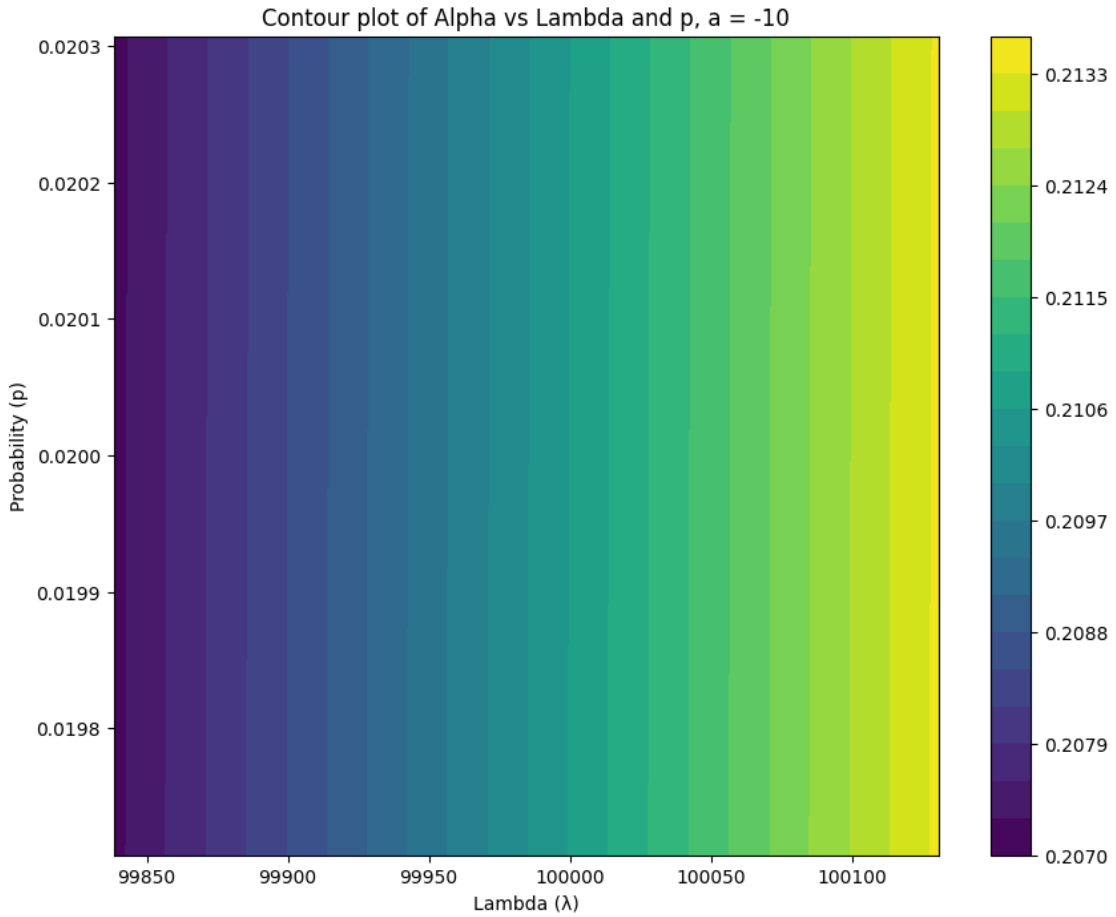
[36]:
```
(0.20739000000006663, 0.2135600000000728)
```

[37]:
```
contour_plot(lam_range_20, p_range_20, 0, omega)
```



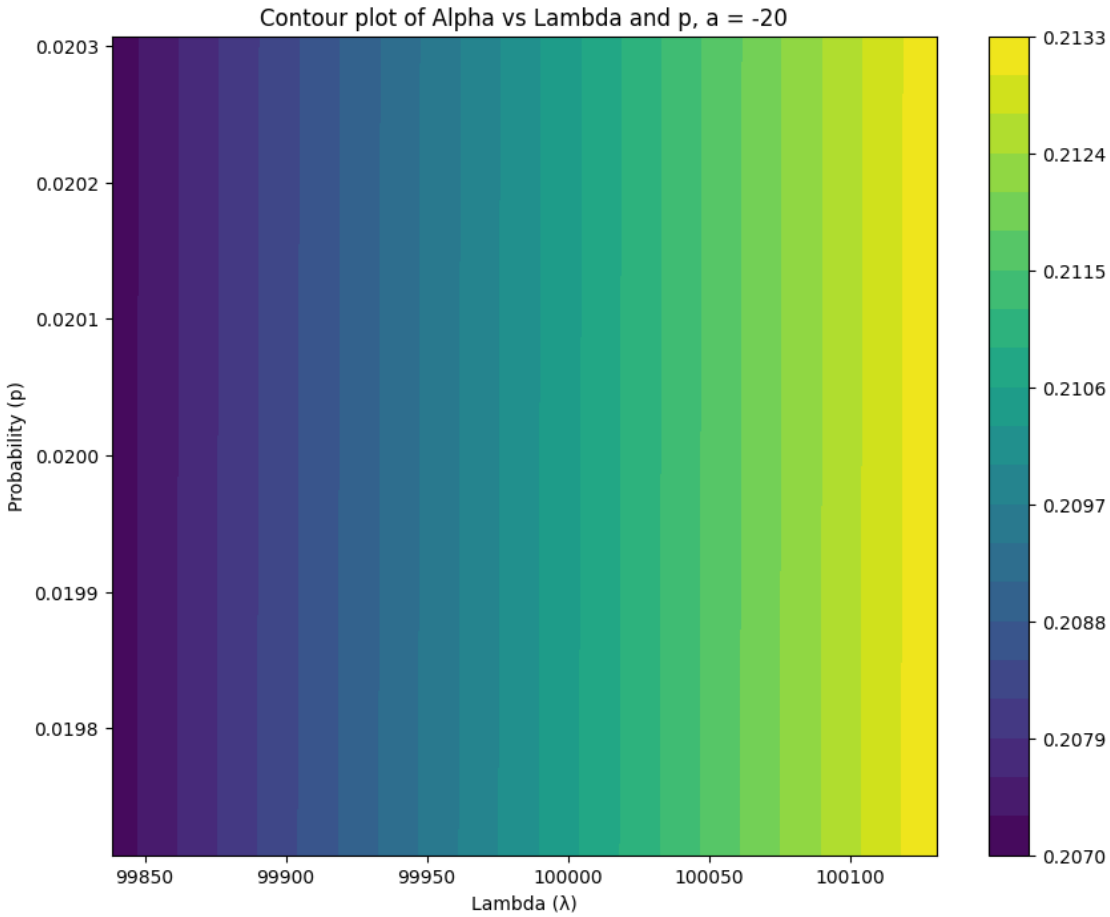
[37]: (0.20729000000006653, 0.2134600000000727)

[38]:
```
contour_plot(lam_range_20, p_range_20, -10, omega)
```



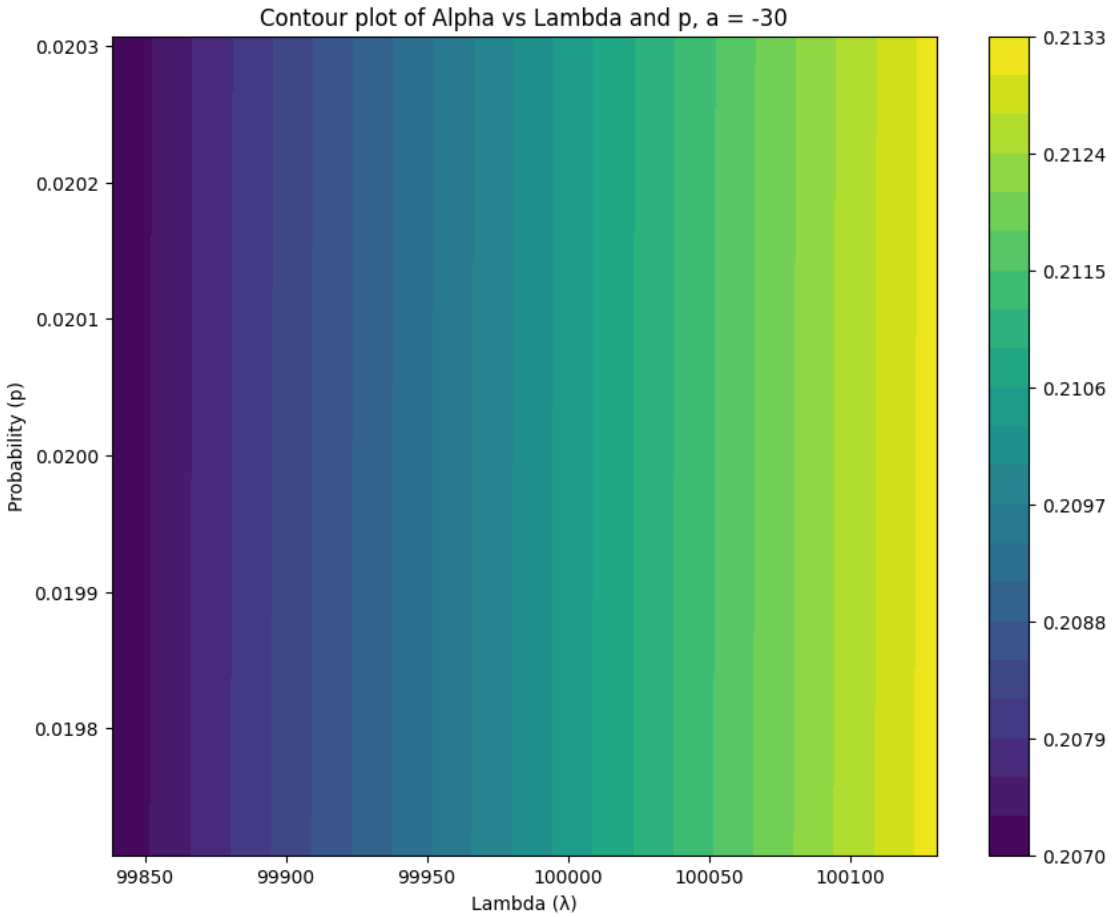
[38]: (0.20719000000006643, 0.2133600000000726)

[39]:
```
contour_plot(lam_range_20, p_range_20, -20, omega)
```



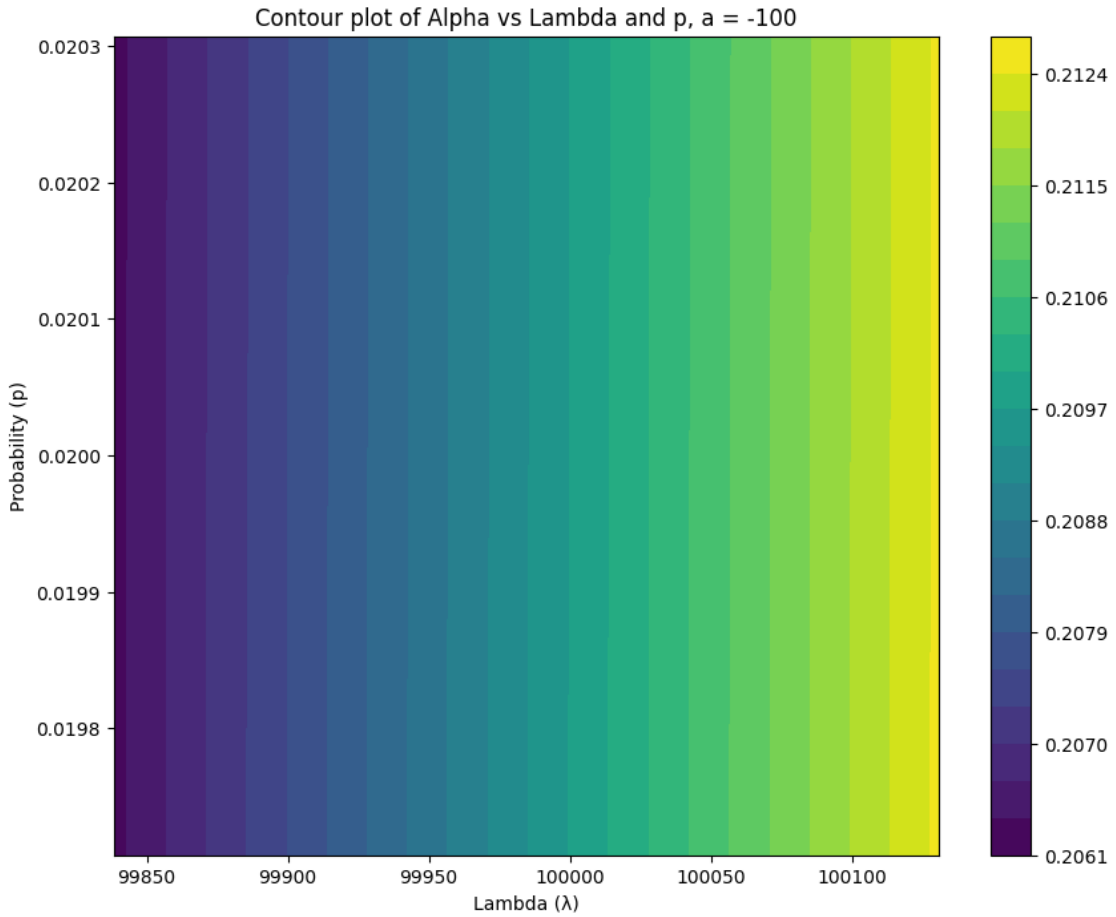
[39]: (0.20710000000006634, 0.2132600000000725)

[40]:
```
contour_plot(lam_range_20, p_range_20, -30, omega)
```



[40]: (0.20700000000006624, 0.2131600000000724)

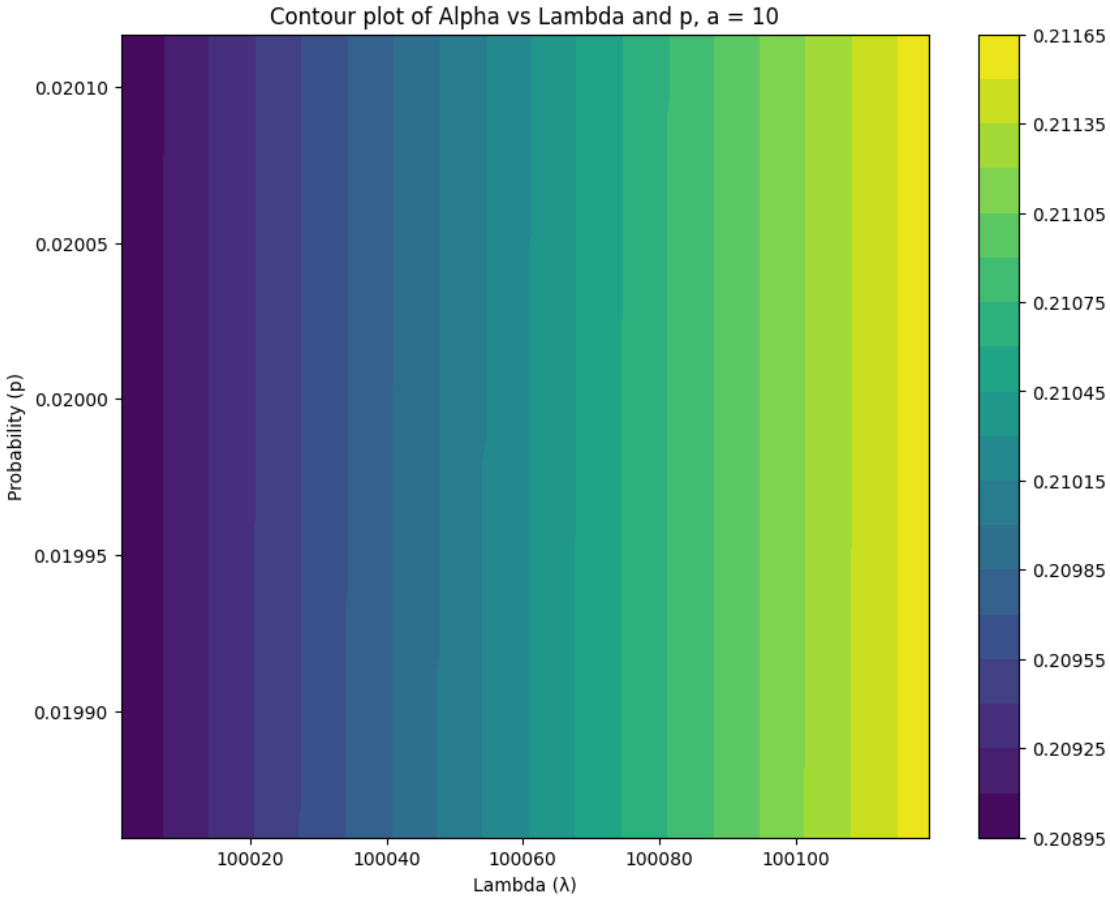
[41]:
```
contour_plot(lam_range_20, p_range_20, -100, omega)
```



[41]: (0.20630000000006554, 0.2124600000000717)

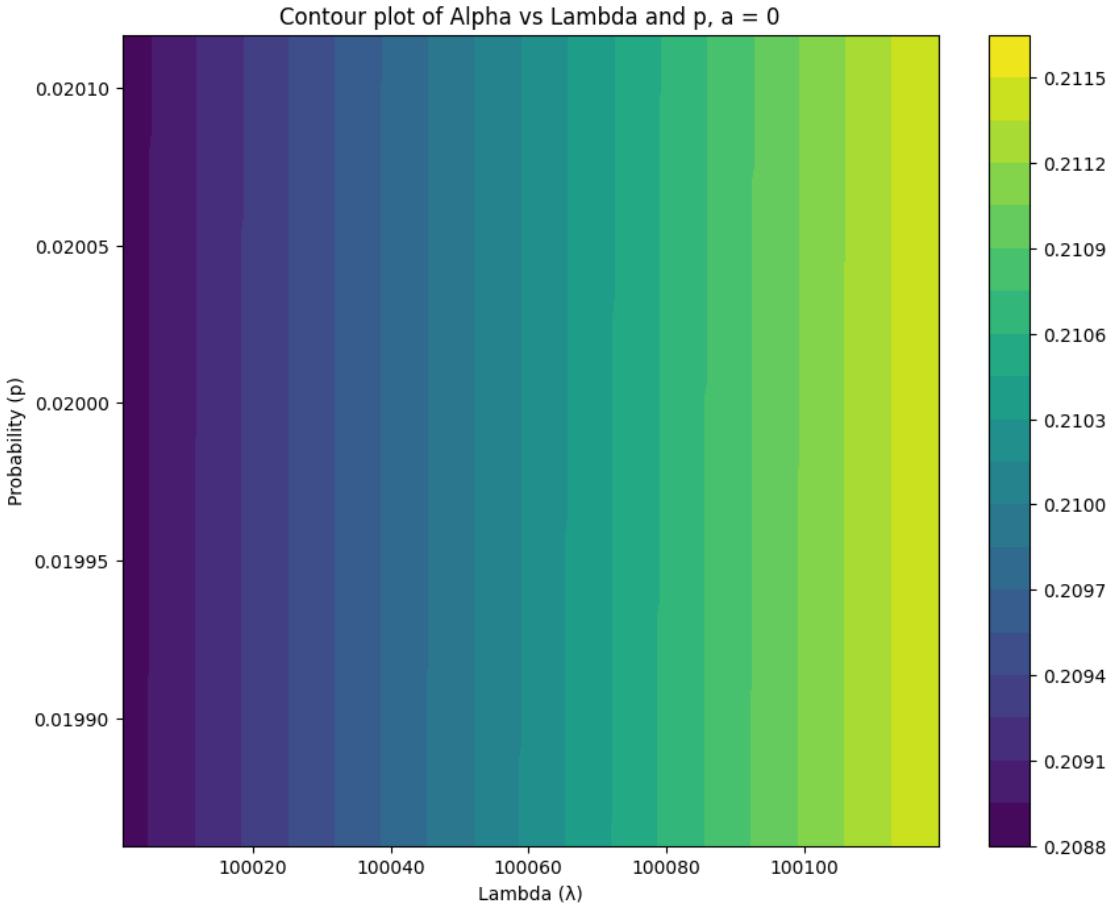
[42]:
```
lam_range_100 = np.linspace(100001.0564309468, 100119.42356905321, 50)  #␣
 ↪Example range for lambda
p_range_100 = np.linspace(0.019859359798009738, 0.020116890166207747, 50)  #␣
 ↪Example range for p
omega = 10
```

[43]:
```
contour_plot(lam_range_100, p_range_100, 10, omega)
```



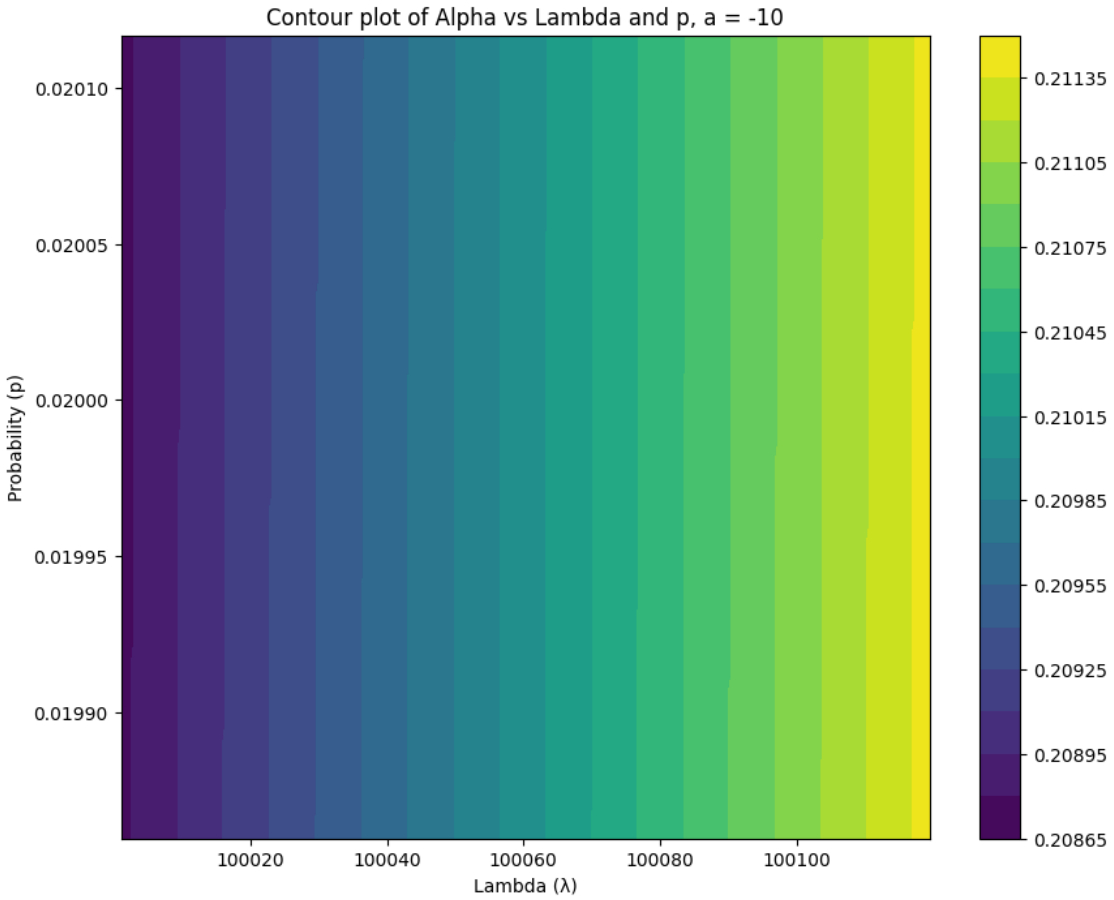
[43]: (0.2089600000000682, 0.21160000000007084)

[44]:
```
contour_plot(lam_range_100, p_range_100, 0, omega)
```



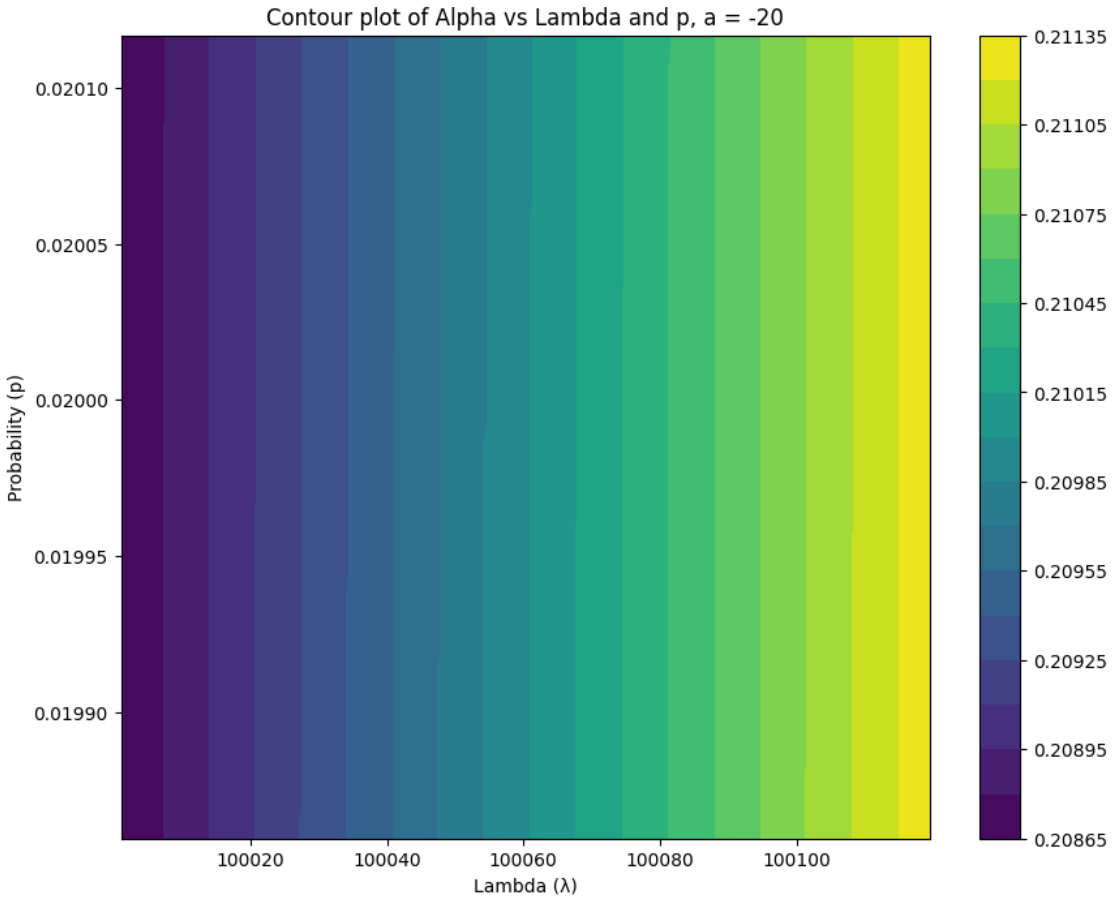
[44]: (0.2088600000000681, 0.21150000000007074)

[45]:
```
contour_plot(lam_range_100, p_range_100, -10, omega)
```



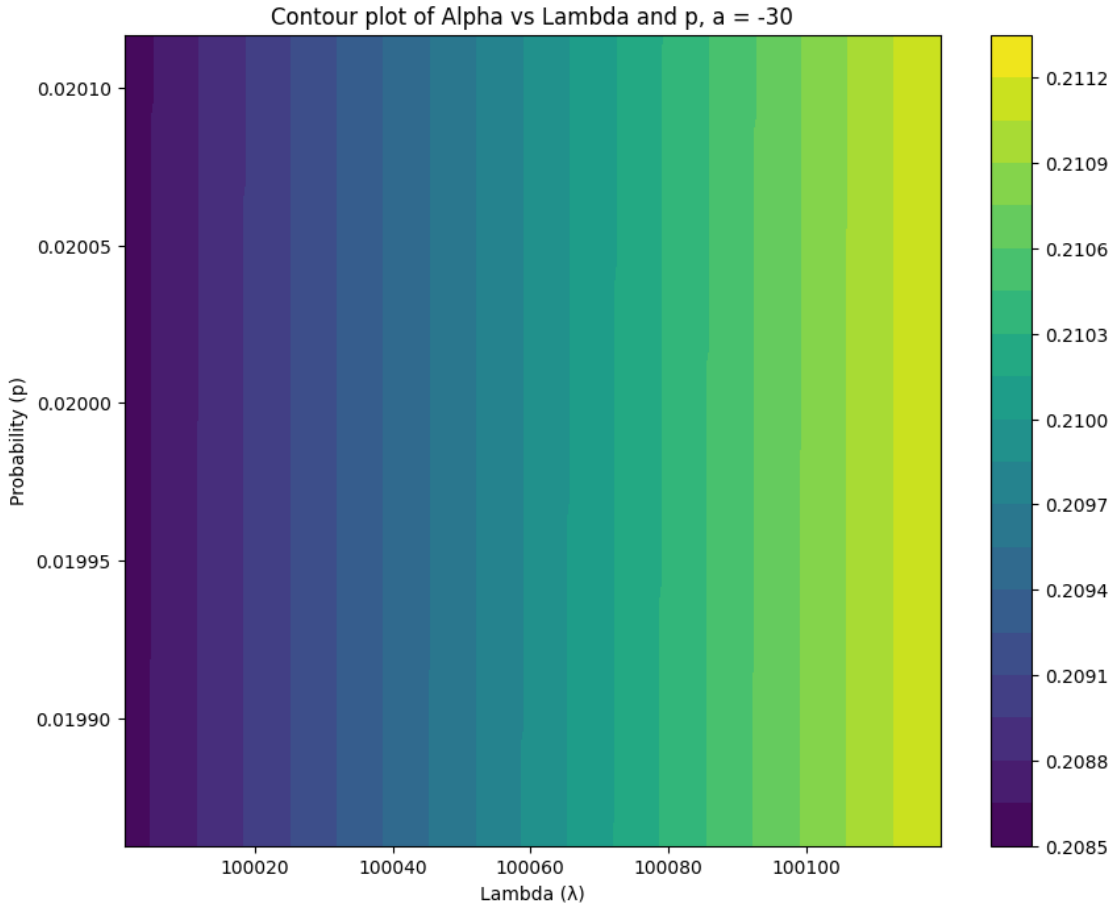
[45]: (0.208760000000068, 0.21140000000007064)

[46]:
```
contour_plot(lam_range_100, p_range_100, -20, omega)
```



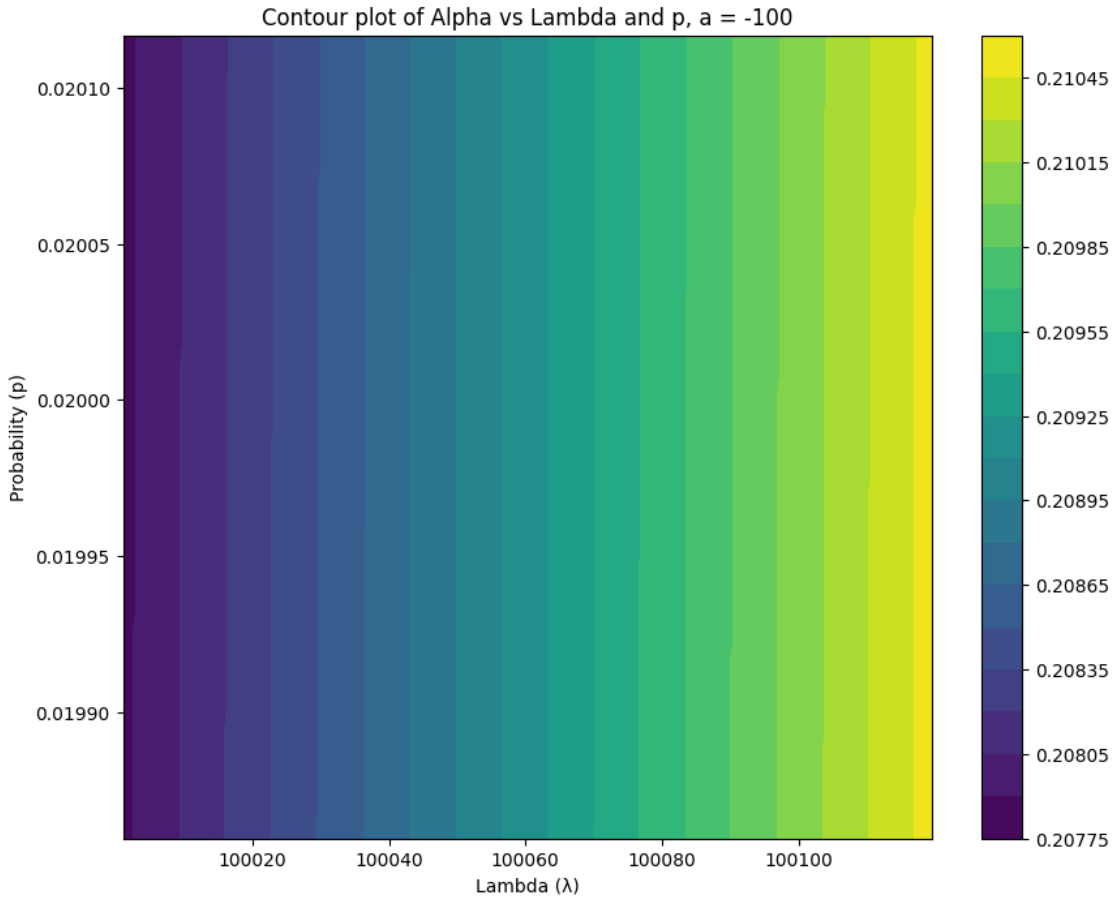
[46]: (0.2086600000000679, 0.21130000000007054)

[47]:
```
contour_plot(lam_range_100, p_range_100, -30, omega)
```



[47]: (0.2085600000000678, 0.21120000000007044)

[48]:
```
contour_plot(lam_range_100, p_range_100, -100, omega)
```



[48]: (0.2078600000000671, 0.21050000000006974)

[ ]:
```
import numpy as np
import pandas as pd
from scipy.stats import norm
## Checking what difference assuming X and Y to be independent makes
def normal_dist_indep(lam, omega, p, a):
    # Gaussian approximation z-score for Pr(S < a) < 0.01
    z_critical = norm.ppf(0.01)
    results = []

    for lam_val in lam:
        for p_val in p:
            for a_val in a:
                alpha = 0.00001  # A starting value
                while True:
                    mean_S = (alpha * lam_val) - (omega * lam_val * p_val)
                    var_S = (lam_val * alpha**2) + (lam_val * p_val* omega**2)␣
↪# - (2 * alpha * omega * lam_val * p_val)
```

```
                    std_dev_S = np.sqrt(var_S)

                    lhs = z_critical * std_dev_S
                    rhs = a_val - mean_S

                    if lhs >= rhs:
                        results.append((lam_val, p_val, a_val, alpha))
                        break

                    alpha += 0.00001
    return results
```

[11]:
```
omega = 10
lam = [10000, 100000]
p = [0.01, 0.02]
a = [10, 0, -10, -20, -30, -100]

min_premium_ind = normal_dist(lam, omega, p, a)

for result in min_premium_ind:
    lam_val, p_val, a_val, alpha = result
    print(f"Minimum Premium P for  ={lam_val}, p={p_val}, a={a_val}, alpha:␣
 ↪{alpha:.5f}")
```

```
Minimum Premium P for  =10000, p=0.01, a=10, alpha: 0.12416
Minimum Premium P for  =10000, p=0.01, a=0, alpha: 0.12316
Minimum Premium P for  =10000, p=0.01, a=-10, alpha: 0.12216
Minimum Premium P for  =10000, p=0.01, a=-20, alpha: 0.12116
Minimum Premium P for  =10000, p=0.01, a=-30, alpha: 0.12016
Minimum Premium P for  =10000, p=0.01, a=-100, alpha: 0.11315
Minimum Premium P for  =10000, p=0.02, a=10, alpha: 0.23358
Minimum Premium P for  =10000, p=0.02, a=0, alpha: 0.23258
Minimum Premium P for  =10000, p=0.02, a=-10, alpha: 0.23158
Minimum Premium P for  =10000, p=0.02, a=-20, alpha: 0.23058
Minimum Premium P for  =10000, p=0.02, a=-30, alpha: 0.22958
Minimum Premium P for  =10000, p=0.02, a=-100, alpha: 0.22258
Minimum Premium P for  =100000, p=0.01, a=10, alpha: 0.10742
Minimum Premium P for  =100000, p=0.01, a=0, alpha: 0.10732
Minimum Premium P for  =100000, p=0.01, a=-10, alpha: 0.10722
Minimum Premium P for  =100000, p=0.01, a=-20, alpha: 0.10712
Minimum Premium P for  =100000, p=0.01, a=-30, alpha: 0.10702
Minimum Premium P for  =100000, p=0.01, a=-100, alpha: 0.10632
Minimum Premium P for  =100000, p=0.02, a=10, alpha: 0.21040
Minimum Premium P for  =100000, p=0.02, a=0, alpha: 0.21030
Minimum Premium P for  =100000, p=0.02, a=-10, alpha: 0.21020
Minimum Premium P for  =100000, p=0.02, a=-20, alpha: 0.21010
Minimum Premium P for  =100000, p=0.02, a=-30, alpha: 0.21000
```

```
Minimum Premium P for  =100000, p=0.02, a=-100, alpha: 0.20930
```