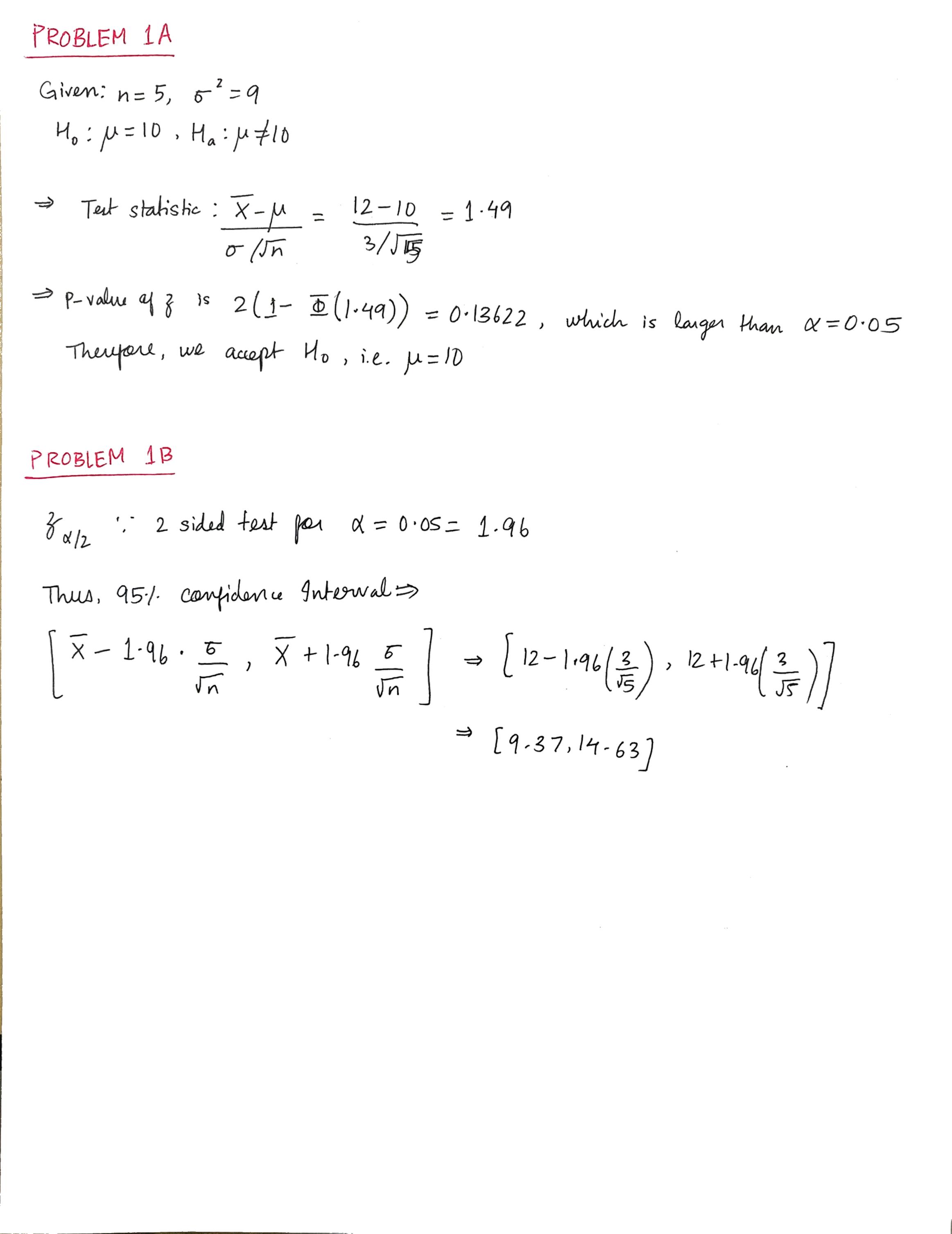
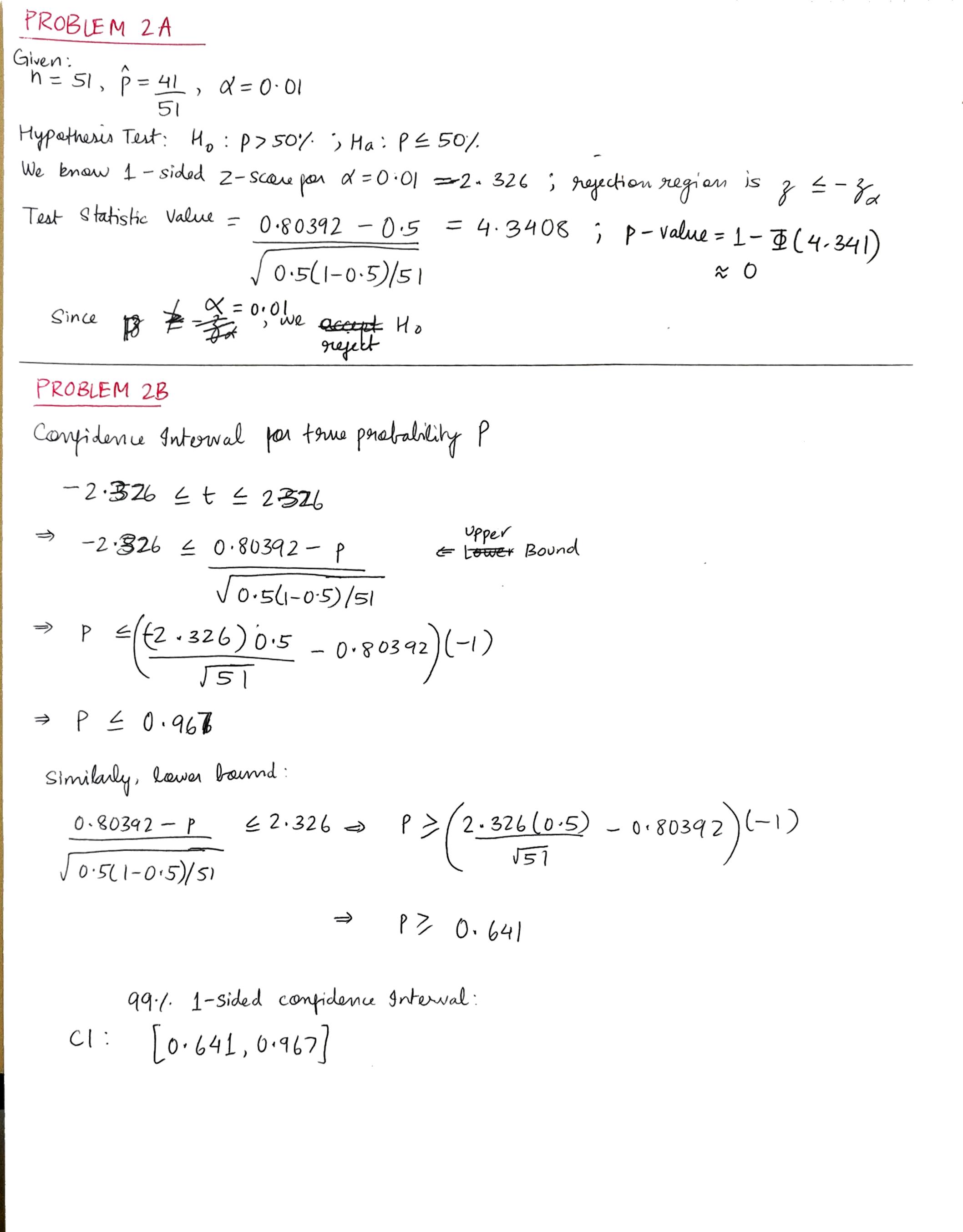
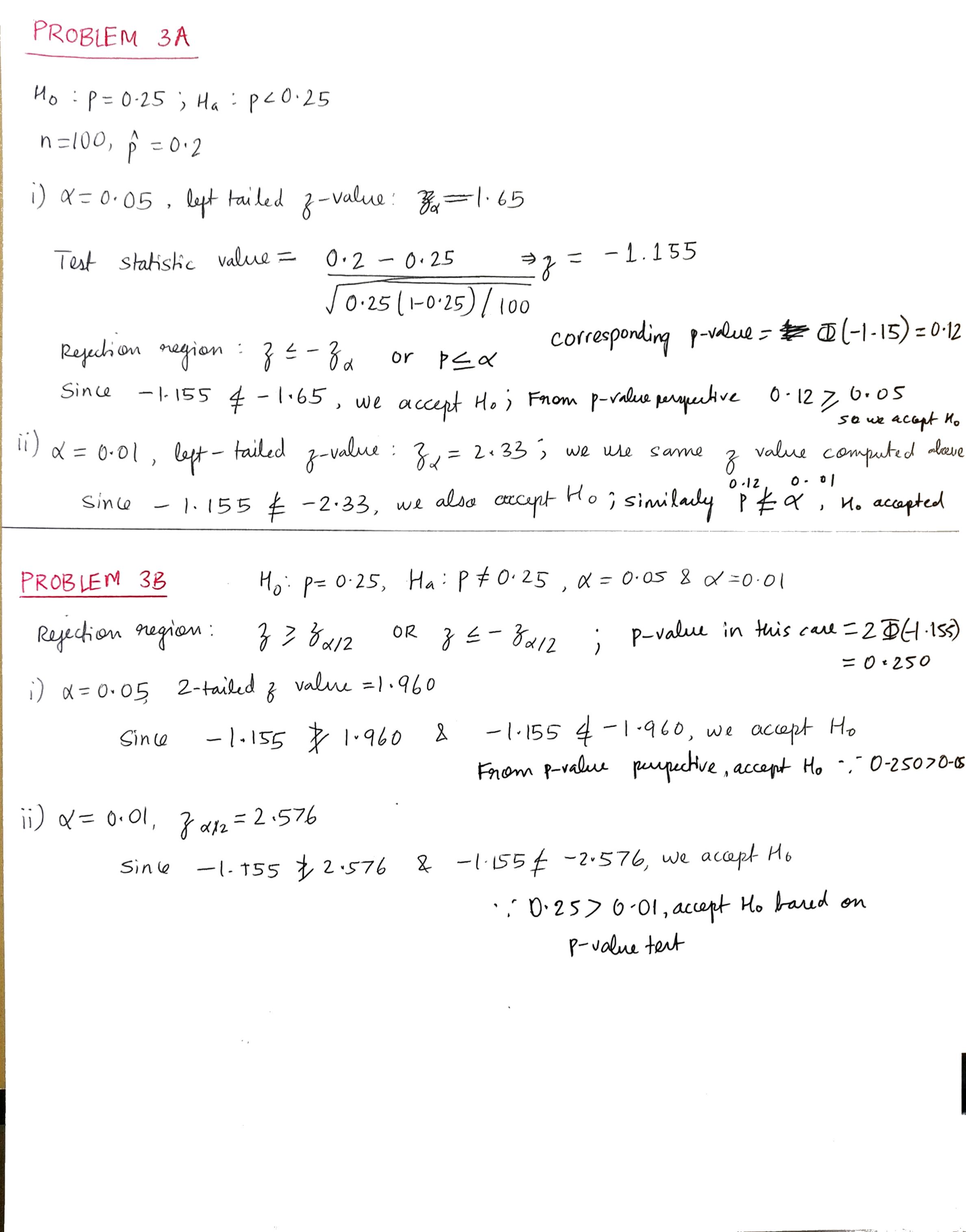
**Problem 1**

****

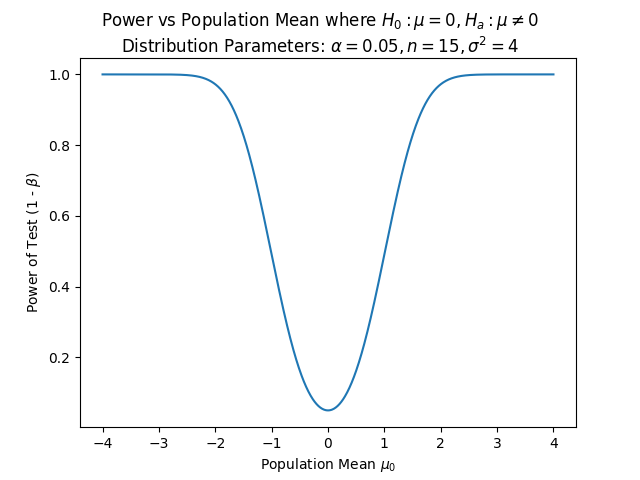
**Problem 2**

****

**Problem 3**

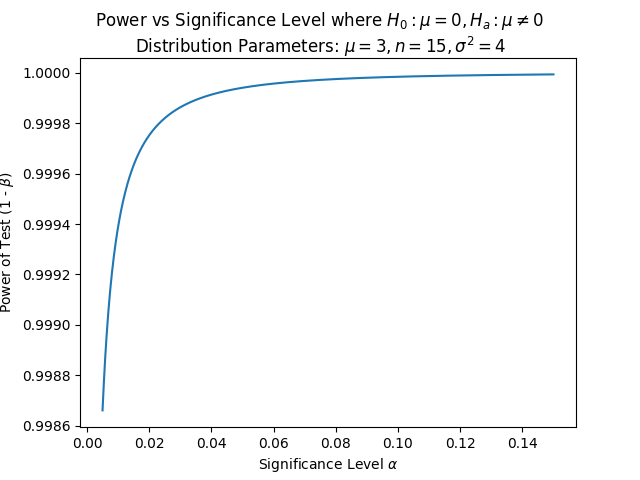
****

**Problem 4A**

****

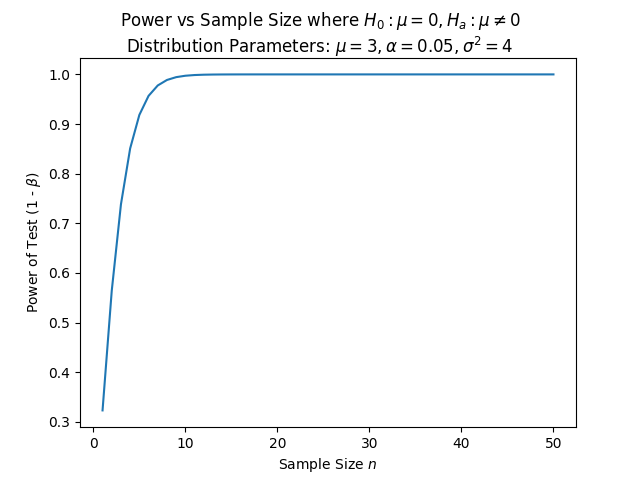
**Takeaway:** Power of the test increases the further the population mean moves from the hypothesized mean in either direction. This is intuitive because the power measures the inverse probability of type 2 errors, i.e. missing accepting the null hypothesis when it is correct, so as population mean moves in either extreme, the chance of the null hypothesis being true decreases, hence chance of type II errors also decrease.

**Problem 4B**

****

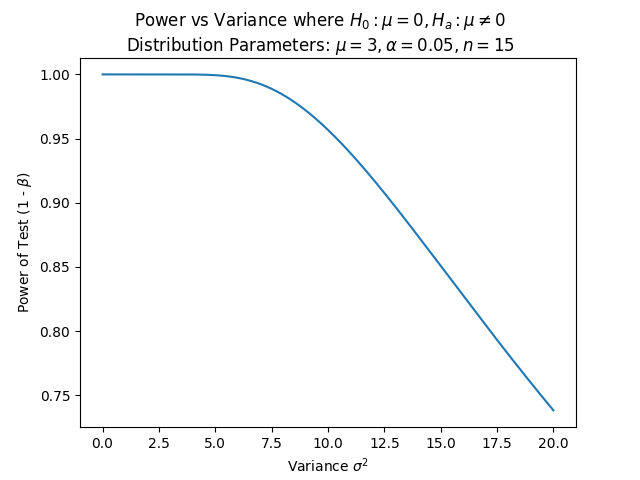
**Takeaway:** The power of the test increases with alpha, asymptotically reaching 100%. This is intuitive, because Type I and Type II error are at odds with each other, by design. Hence, loosening the tolerated Type I error decreases the probability of committing a Type II error.

**Problem 4C**

****

**Takeaway:** Power increases with sample size, asymptotically reaching 100%. This is intuitive because the greater the n, the more closely the sample mean represents the population mean, resulting in a more powerful conclusion.

**Problem 4D**

****

**Takeaway:** Power decreases as variance increases. This makes intuitive sense because in a very spread-out distribution, the sample mean might not closely represent the population mean, so it is easy to make incorrect acceptance or rejection decisions.

**import** numpy **as** np

**import** scipy.special **as** sp

**import** scipy.stats **as** st

**from** matplotlib **import** pyplot **as** plt

**from** time **import** sleep

**def** generate\_mu():

*# Simulate True Mean of Normal Distribution*

    mu\_array = np.linspace(-4, 4, 1000)

**return** mu\_array

**def** generate\_alpha():

*# Simulate Probability of Type 1 Errors*

    alpha\_error = np.linspace(0.005, 0.15, 1000)

**return** alpha\_error

**def** generate\_n():

*# Simulate Various Sample Sizes*

    max\_n = 50

    n\_array = np.linspace(1, max\_n, max\_n)

**return** n\_array

**def** generate\_var():

*# Simulate Different Variances*

    sigma\_squared\_array = np.linspace(0.001, 20, 1000)

**return** sigma\_squared\_array

**def** compute\_power(mu, z\_alpha, n, sigma):

*# Power is probability of rejecting the null hypothesis*

*# when it is wrong*

*# Find acceptance and rejection regions in terms of sample means*

    Xbar\_lower\_bound = -z\_alpha \* (sigma / n \*\* 0.5)

    Xbar\_upper\_bound = z\_alpha \* (sigma / n \*\* 0.5)

*# Find power in terms of Z value*

    Z\_left\_tail = (Xbar\_lower\_bound - mu) / (sigma / n \*\* 0.5)

    Z\_right\_tail = (Xbar\_upper\_bound - mu) / (sigma / n \*\* 0.5)

*# Compute and return power*

    power = abs(sp.ndtr(Z\_left\_tail)) + (1 - abs(sp.ndtr(Z\_right\_tail)))

**return** power

**def** plot\_power(scenario):

**if** (scenario == **'vary\_mu'**):

        mu\_array = generate\_mu()

        z\_alpha = 1.96  *# two-tailed test value for alpha = 0.05*

        n = 15  *# number of observations*

        sigma = 2  *# standard deviation*

        power\_array = []

**for** mu **in** mu\_array:

            power = compute\_power(mu, z\_alpha, n, sigma)

            power\_array.append(power)

        plt.plot(mu\_array, power\_array)

        plt.suptitle(

**r'**Power vs Population Mean where $H\_0 : \mu = 0, H\_a : \mu \neq 0$**'** + **'\n'** +

**r'**Distribution Parameters: $\alpha = 0.05, n = 15, \sigma^2 = 4$**'**)

        plt.xlabel(**r'**Population Mean $\mu\_0$**'**)

**elif** (scenario == **'vary\_alpha'**):

        alpha\_array = generate\_alpha()

        mu = 3.0  *# population mean*

        n = 15  *# number of observations*

        sigma = 2  *# standard deviation*

        power\_array = []

**for** alpha **in** alpha\_array:

            z\_alpha = abs(st.norm.ppf(alpha / 2.0))

            power = compute\_power(mu, z\_alpha, n, sigma)

            power\_array.append(power)

        plt.plot(alpha\_array, power\_array)

        plt.suptitle(

**r'**Power vs Significance Level where $H\_0 : \mu = 0, H\_a : \mu \neq 0$**'** + **'\n'** +

**r'**Distribution Parameters: $\mu = 3, n = 15, \sigma^2 = 4$**'**)

        plt.xlabel(**r'**Significance Level $\alpha$**'**)

**elif** (scenario == **'vary\_n'**):

        n\_array = generate\_n()

        mu = 3.0  *# population mean*

        z\_alpha = 1.96  *# two-tailed test value for alpha = 0.05*

        sigma = 2  *# standard deviation*

        power\_array = []

**for** n **in** n\_array:

            power = compute\_power(mu, z\_alpha, n, sigma)

            power\_array.append(power)

        plt.plot(n\_array, power\_array)

        plt.suptitle(

**r'**Power vs Sample Size where $H\_0 : \mu = 0, H\_a : \mu \neq 0$**'** + **'\n'** +

**r'**Distribution Parameters: $\mu = 3, \alpha = 0.05, \sigma^2 = 4$**'**)

        plt.xlabel(**r'**Sample Size $n$**'**)

**elif** (scenario == **'vary\_sigma\_squared'**):

        sigma\_squared\_array = generate\_var()

        z\_alpha = 1.96  *# two-tailed test value for alpha = 0.05*

        n = 15  *# number of observations*

        mu = 3.0  *# population mean*

        power\_array = []

**for** sigma\_squared **in** sigma\_squared\_array:

            sigma = sigma\_squared \*\* 0.5

            power = compute\_power(mu, z\_alpha, n, sigma)

            power\_array.append(power)

        plt.plot(sigma\_squared\_array, power\_array)

        plt.suptitle(

**r'**Power vs Variance where $H\_0 : \mu = 0, H\_a : \mu \neq 0$**'** + **'\n'** +

**r'**Distribution Parameters: $\mu = 3, \alpha = 0.05, n = 15$**'**)

        plt.xlabel(**r'**Variance $\sigma^2$**'**)

    plt.ylabel(**r'**Power of Test (1 - $\beta$)**'**)

    plt.show()

**def** main():

    scenario\_array = [**'vary\_mu'**,

**'vary\_alpha'**,

**'vary\_n'**,

**'vary\_sigma\_squared'**

                      ]

**for** scenario **in** scenario\_array:

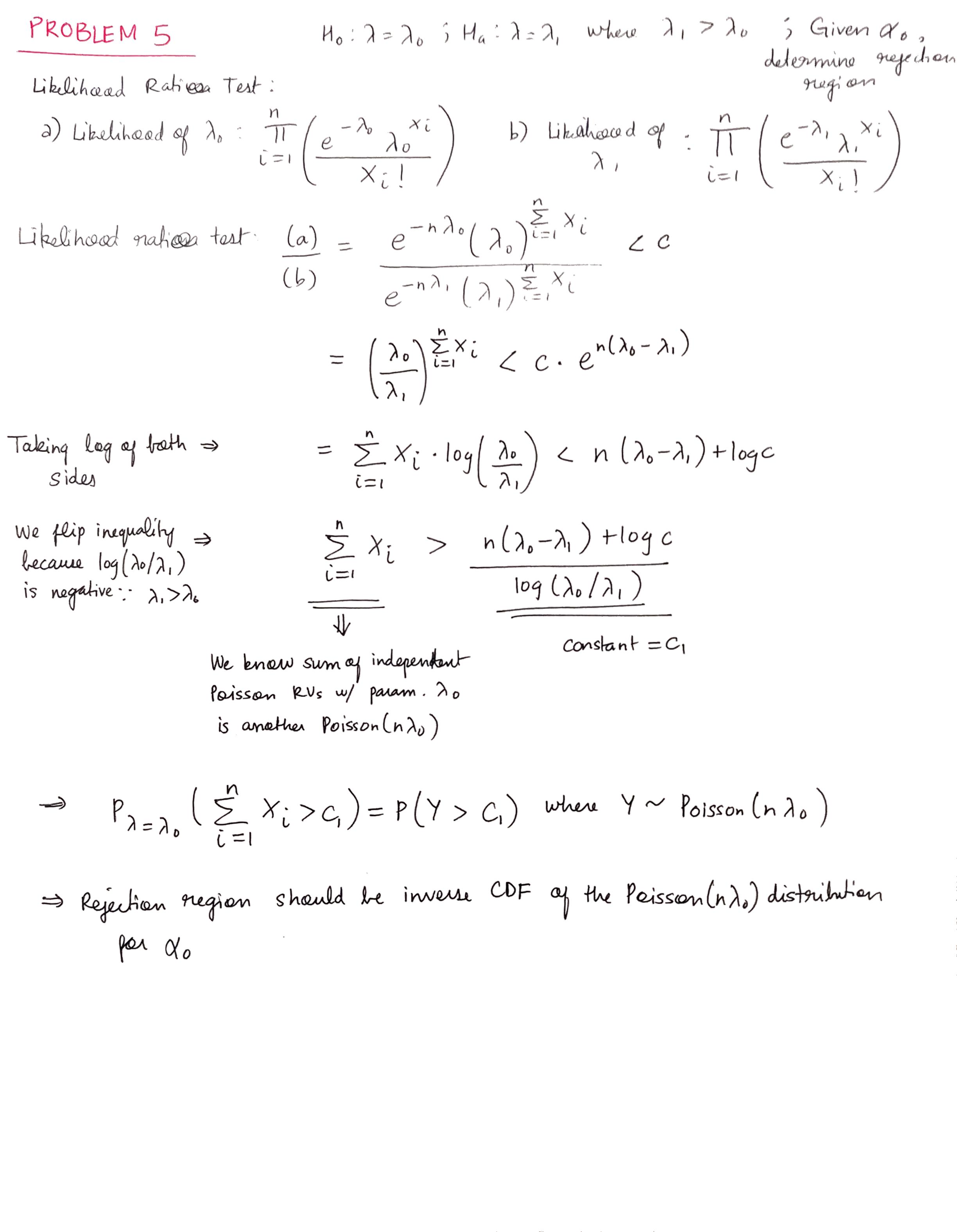
        plot\_power(scenario)

**return**

**if** \_\_name\_\_ == **'\_\_main\_\_'**:

    main()

**Problem 5**

****

**Problem 6**

