期中專題：蒙地卡羅

學號：411073088

姓名：陳敬翰

作品目標：

- 以蒙地卡羅實驗驗證 J-B 檢定統計量，透過不同的樣本數，觀察檢定力的變化。
- 觀察不同分配下，使用J-B統計量檢定力的變化。
- 測試不同檢定量的檢定力。

以蒙地卡羅實驗驗證 **J-B** 檢定統計量，透過不同的樣本數，觀察檢定力的變化。

令 $\{x_i, i = 1, \cdots, n\}$ 代表來自標準常態 N(0,1) 的 n 個隨機樣本。統計量 G_1 表示為

G_1 = $\sqrt{\frac{n}{6}}\hat{s}$，

其中 $\hat{s}$ 為偏態係數（skewness）的估計值。請利用蒙地卡羅模擬（Monte Carlo Simulation）驗證統計量 G_1 服從標準常態 N(0,1)。其中蒙地卡羅模擬的環境設定（scenarios）為： 樣本數 n = 10, 20, 30, 50, 100, 300, 500。

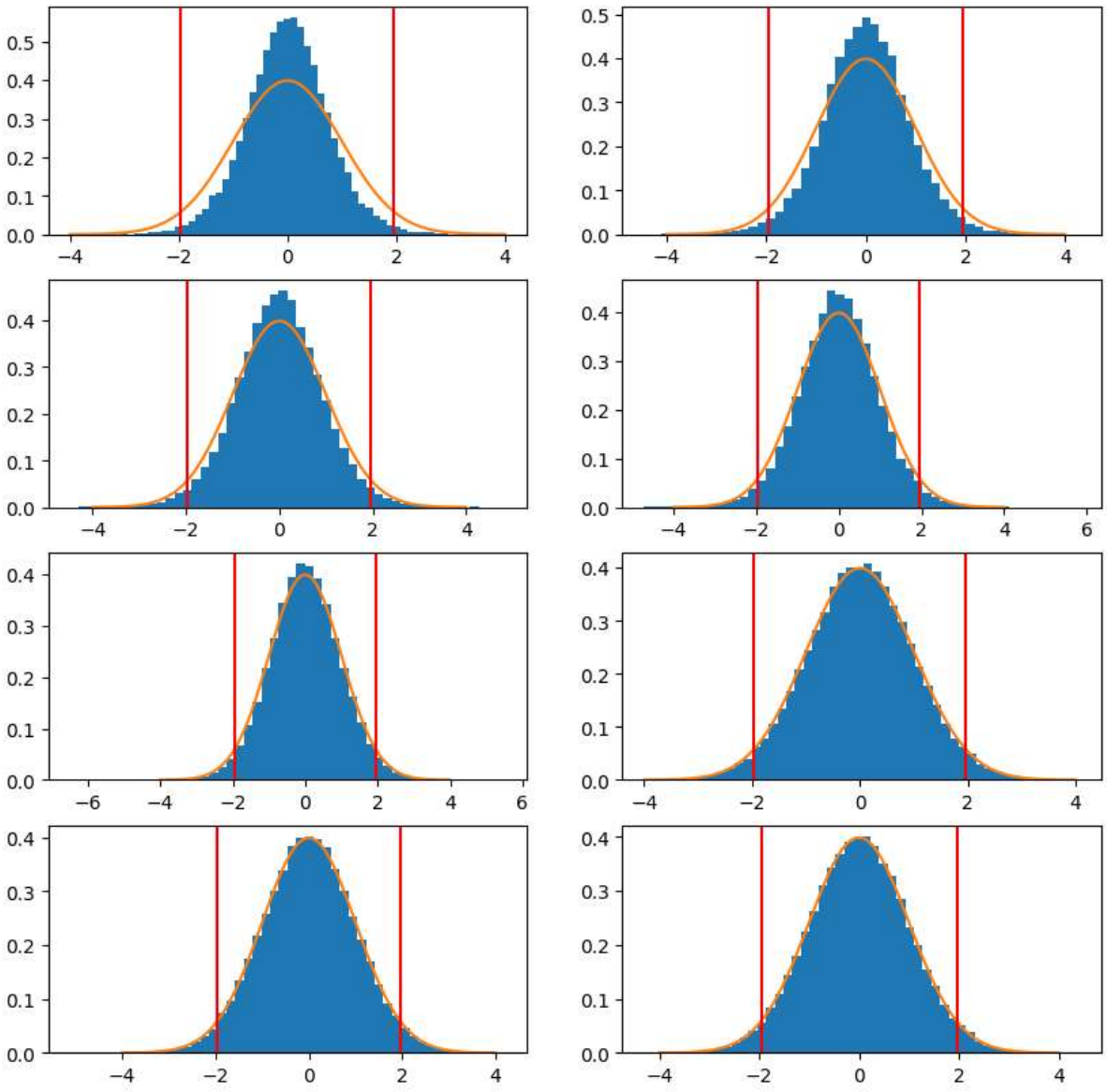
針對每個樣本數 n，模擬次數皆為 N=10000。

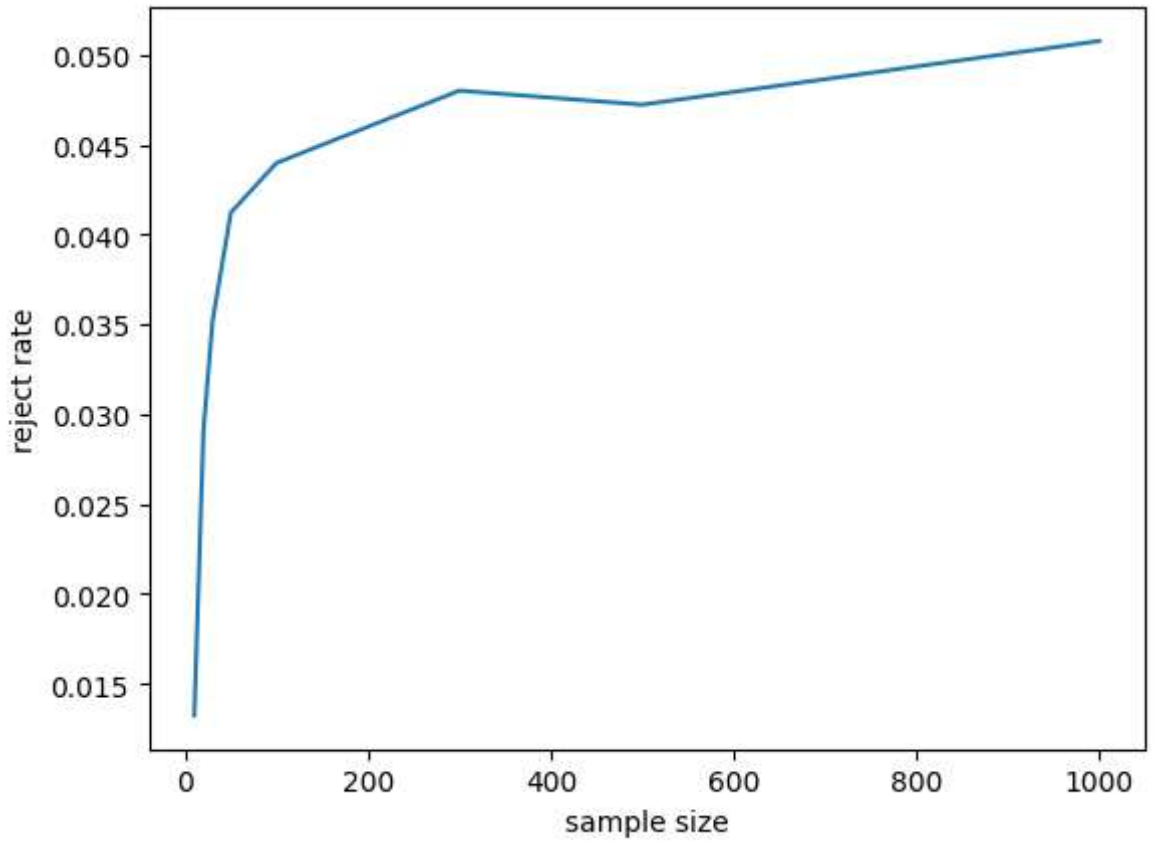
繪製 n = 10 與 n = 500 時，統計量 G_1 的直方圖與 ECDF 圖。並分別畫上對應的標準常態 PDF 與 CDF 圖。

In [ ]:
```
from scipy.stats import norm
from scipy.stats import skew, kurtosis
from matplotlib import pyplot as plt
import numpy as np


fig, ax = plt.subplots(4, 2, figsize=(10, 10))
N = 50000
n_size = [10, 20, 30, 50, 100, 300, 500, 1000]


xx = np.linspace(-4,4,1000)
yy = norm.pdf(xx)


fail_list = []

for i in range(8):
    n = n_size[i]
    X = norm.rvs(size=(N,n))
    g1 = np.sqrt(n / 6) * skew(X,axis=1)
    ax[i//2, i%2].hist(g1, bins=50, density=True, label='n='+str(n))
    ax[i//2, i%2].plot(xx, yy, label='N(0,1)')
    ax[i//2, i%2].axvline(x=1.96, color='r')
    ax[i//2, i%2].axvline(x=-1.96, color='r')
```

```
    # ax[i//2, i%2].set_title('n='+str(n))

    # 計算型一誤差
    fail = np.sum(np.abs(g1)>1.96)/N
    fail_list.append(fail)
print(fail_list)
plt.show()

plt.plot(n_size, fail_list)
plt.xlabel('sample size')
plt.ylabel('reject rate')
plt.show()
```

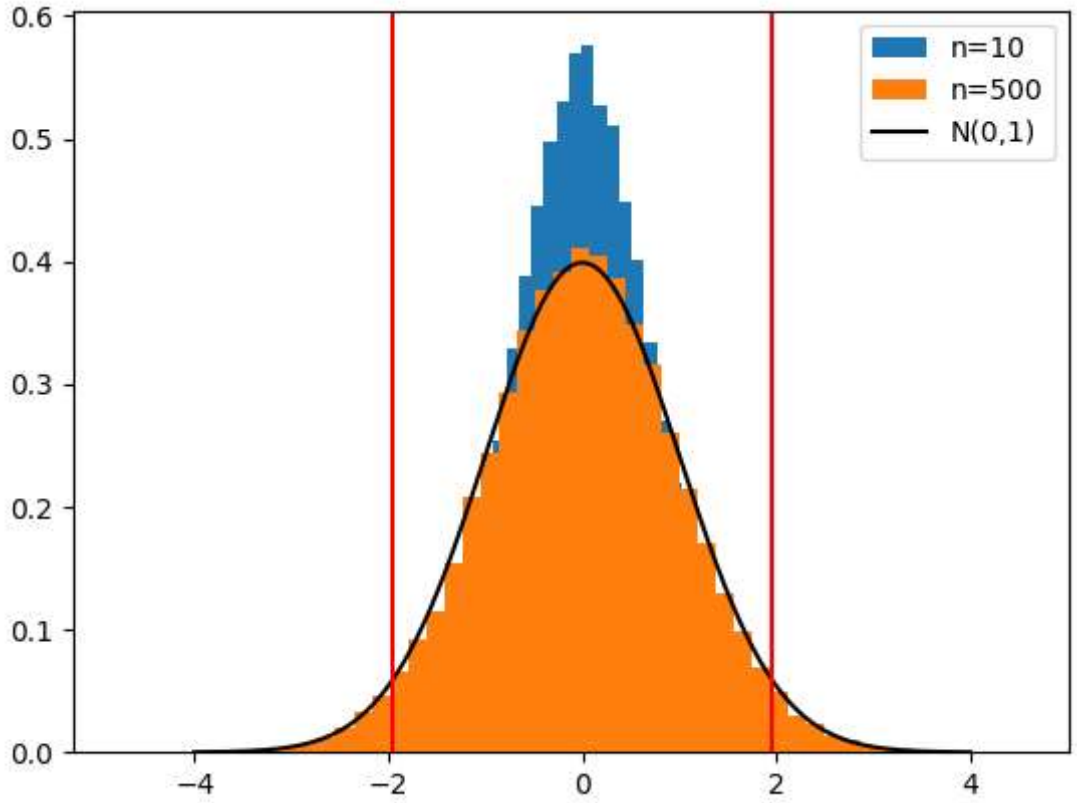
```
[0.01324, 0.0291, 0.0352, 0.04122, 0.04398, 0.048, 0.04722, 0.05076]
```





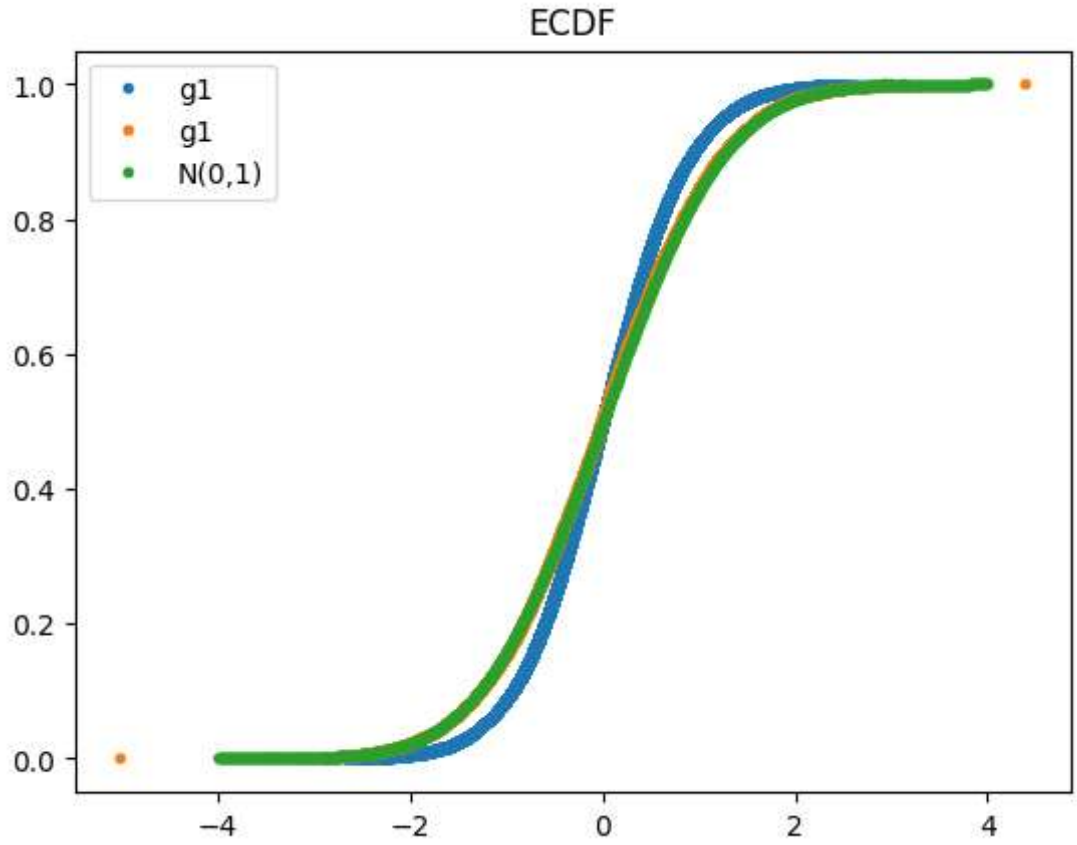
In [ ]:

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

color = ['r','g','b','y','c','m','k','w']

n = 10
N = 50000
X = norm.rvs(size=(N,n))
g1 = np.sqrt(n / 6) * skew(X,axis=1)
plt.hist(g1, bins=50, density=True, label='n=10')

n = 500
N = 50000
X = norm.rvs(size=(N,n))
g1 = np.sqrt(n / 6) * skew(X,axis=1)
plt.hist(g1, bins=50, density=True, label='n=500')


xx = np.linspace(-4,4,1000)
plt.plot(xx, norm.pdf(xx), label='N(0,1)', color='k')
plt.legend()
plt.axvline(x=1.96, color='r')
plt.axvline(x=-1.96, color='r')
plt.show()
```



In [ ]:

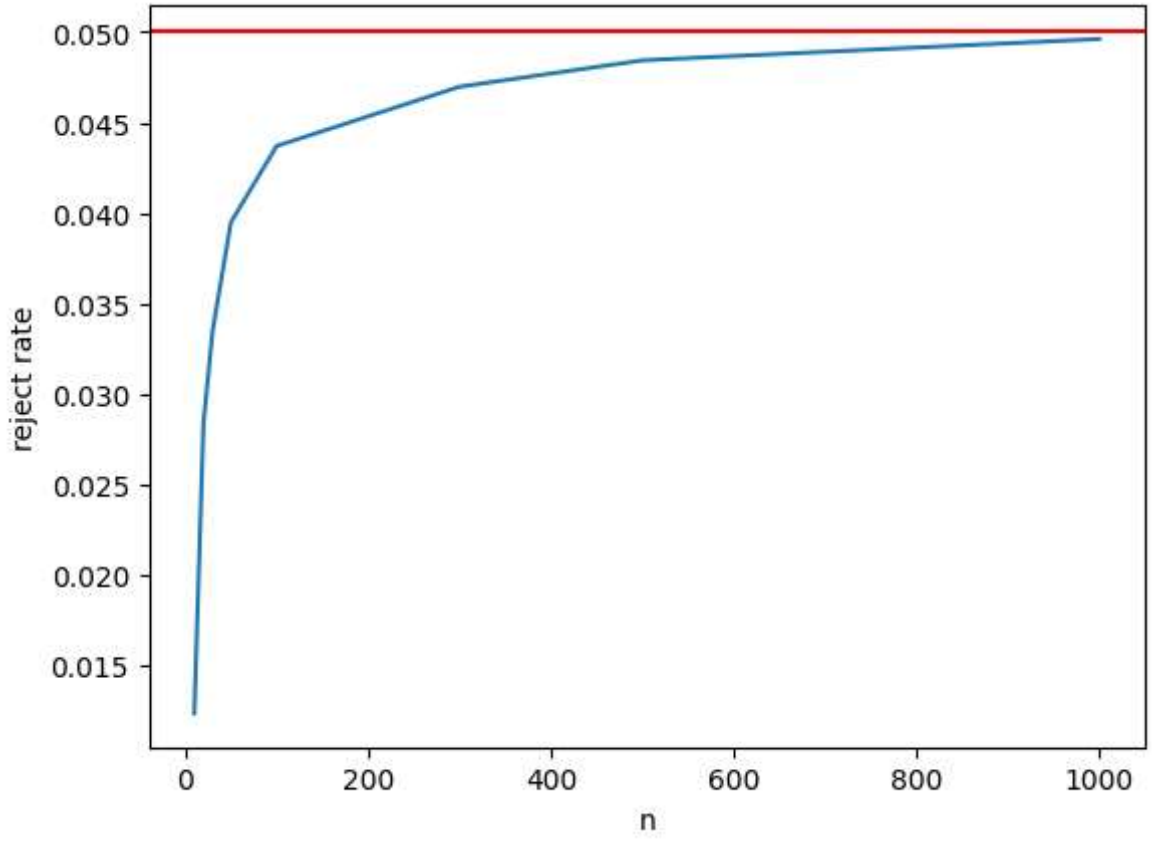
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm, skew

# Generate the data
n = 10
N = 50000
X = norm.rvs(size=(N,n))
g1 = np.sqrt(n / 6) * skew(X,axis=1)

# Calculate the ECDF
x = np.sort(g1)
y = np.arange(1, len(x)+1) / len(x)
# Plot the ECDF
plt.plot(x, y, label = 'g1', marker='.', linestyle='none')

n = 500
N = 10000
X = norm.rvs(size=(N,n))
g1 = np.sqrt(n / 6) * skew(X,axis=1)

# Calculate the ECDF
x = np.sort(g1)
y = np.arange(1, len(x)+1) / len(x)
# Plot the ECDF
plt.plot(x, y, label = 'g1', marker='.', linestyle='none')


x = np.linspace(-4,4,1000)
plt.plot(x, norm.cdf(x), label='N(0,1)',marker='.', linestyle='none')


plt.title('ECDF')
plt.legend()
plt.show()
```



In [ ]:

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

N = 50000
n_size = [10, 20, 30, 50, 100, 300, 500, 1000]

xx = np.linspace(-4,4,1000)
yy = norm.pdf(xx)


fail_list = []

for i in range(8):
    n = n_size[i]
    X = norm.rvs(size=(N,n))
    g1 = np.sqrt(n / 6) * skew(X,axis=1)

    # 計算型一誤差
    fail = np.sum(np.abs(g1)>1.96)/N
    fail_list.append(fail)
print(fail_list)

plt.plot(n_size, fail_list)
plt.xlabel('n')
plt.ylabel('reject rate')
plt.axhline(y=0.05, color='r')
plt.show()
```

```
[0.01232, 0.02838, 0.03348, 0.03948, 0.0437, 0.04698, 0.04844, 0.04962]
```



作品結論：

- 在設定常態alpha=0.05,拒絕域為正負1.96的狀況下，當reject_rate越接近alpha，則檢定力越高，代表這個分配越接近常態分配，由上圖可知，當n越來越大時，reject_rate越接近alpha，檢定力越高，代表這個分配越接近常態分配。
- 另外，g1適用於檢定常態分配的偏態，當n越大時，g1的分配越接近常態分配，因此g1適用於檢定常態分配的偏態。

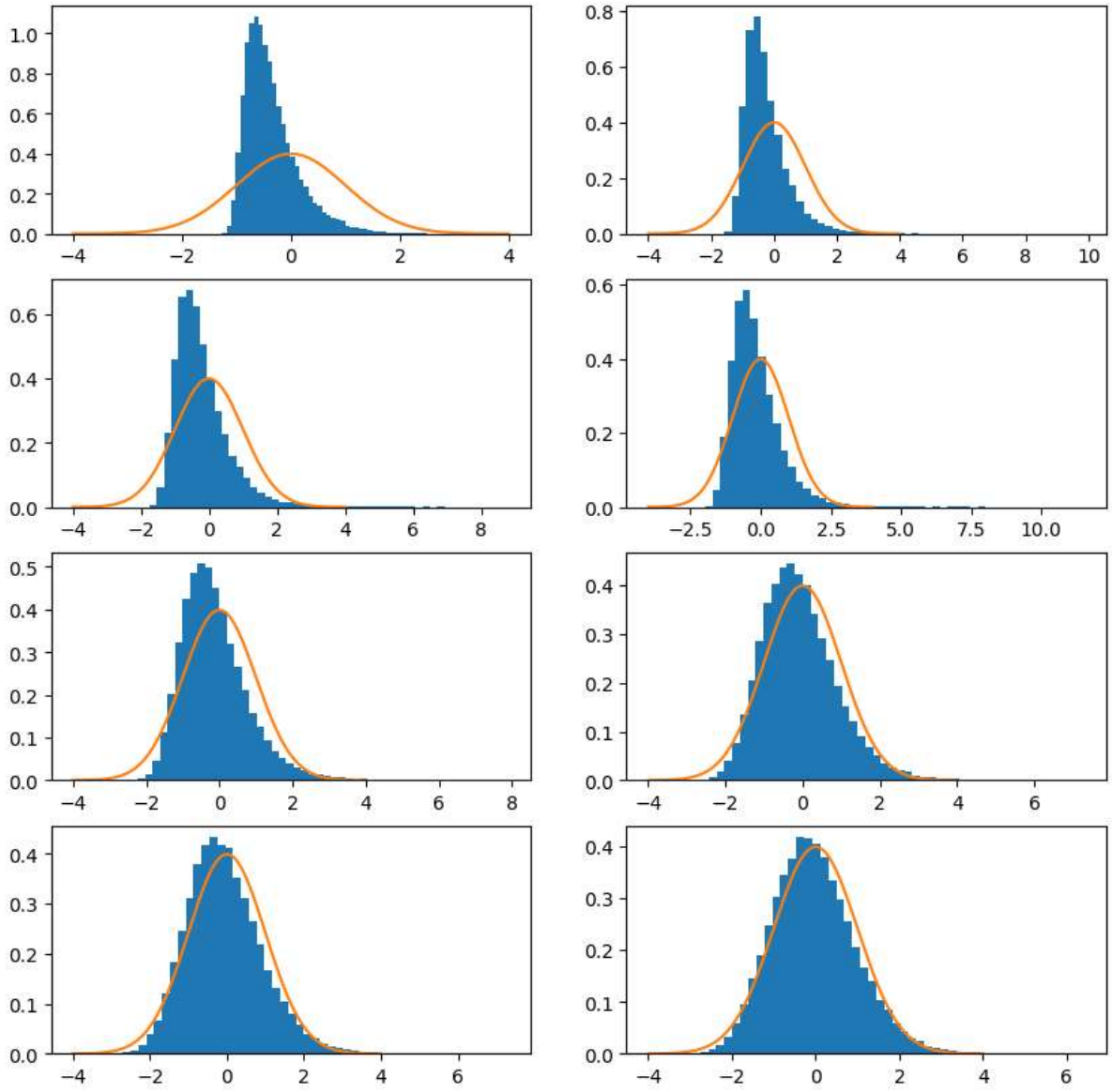
同上，但令統計量為

G_2 = $\sqrt{\frac{n}{24}}(\hat{k}-3)$ ，

其中 $\hat{k}$ 為峰態係數（Kurtosis）的估計值。同樣利用蒙地卡羅模擬，驗證統計量 G_2 服從標準常態 N(0,1)。蒙地卡羅模擬的環境設定同上。

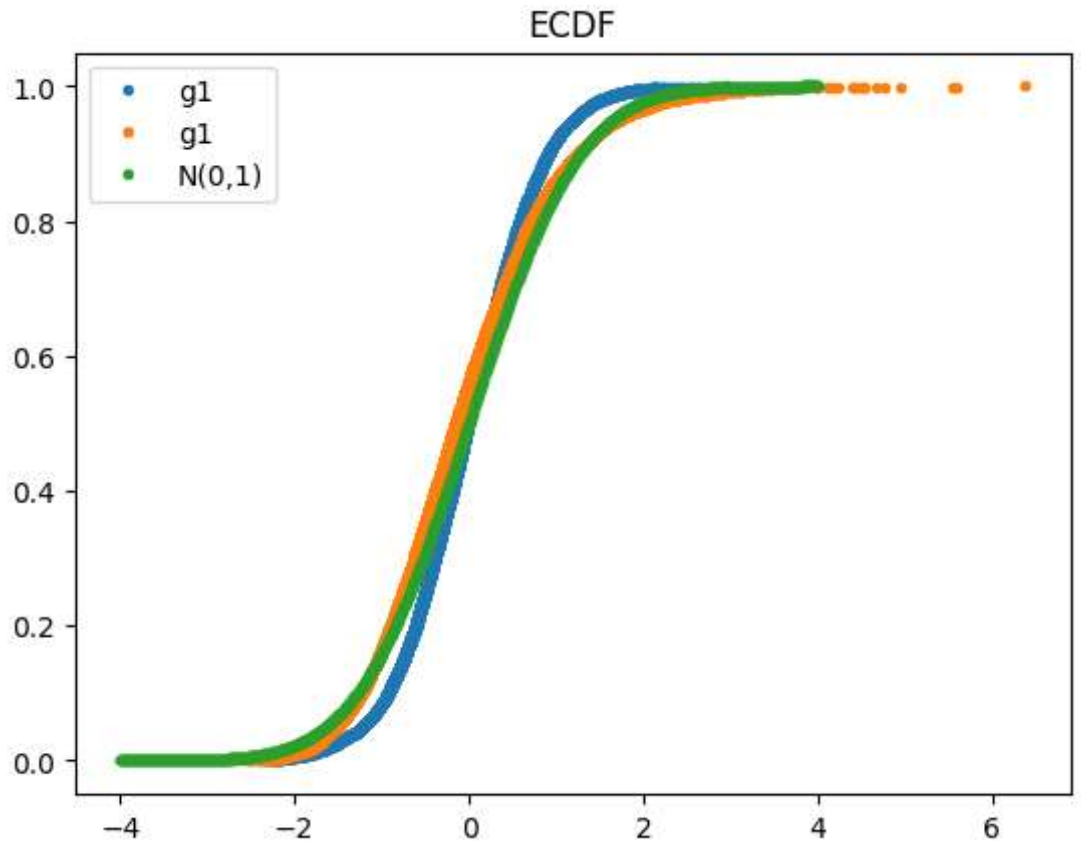
In [ ]:
```
from scipy.stats import norm
from scipy.stats import skew, kurtosis
from matplotlib import pyplot as plt
import numpy as np


fig, ax = plt.subplots(4, 2, figsize=(10, 10))
N = 50000
n_size = [10, 20, 30, 50, 100, 300, 500, 1000, 10000]


xx = np.linspace(-4,4,1000)
yy = norm.pdf(xx)
```

```
for i in range(8):
    n = n_size[i]
    X = norm.rvs(size=(N,n))
    g1 = np.sqrt(n / 24) * kurtosis(X,axis=1)
    ax[i//2, i%2].hist(g1, bins=50, density=True, label='n='+str(n))
    ax[i//2, i%2].plot(xx, yy, label='N(0,1)')
    # ax.flat[i].set_title('n='+str(n))
plt.show()
```



In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm, skew

# Generate the data
n = 10
N = 10000
X = norm.rvs(size=(N,n))
g1 = np.sqrt(n / 6) * skew(X,axis=1)

# Calculate the ECDF
x = np.sort(g1)
y = np.arange(1, len(x)+1) / len(x)
# Plot the ECDF
plt.plot(x, y, label = 'g1', marker='.', linestyle='none')

n = 500
N = 10000
```

```
X = norm.rvs(size=(N,n))
g2 = np.sqrt(n / 24) * kurtosis(X,axis=1)

# Calculate the ECDF
x = np.sort(g2)
y = np.arange(1, len(x)+1) / len(x)
# Plot the ECDF
plt.plot(x, y, label = 'g1', marker='.', linestyle='none')


x = np.linspace(-4,4,1000)
plt.plot(x, norm.cdf(x), label='N(0,1)',marker='.', linestyle='none')


plt.title('ECDF')
plt.legend()
plt.show()
```

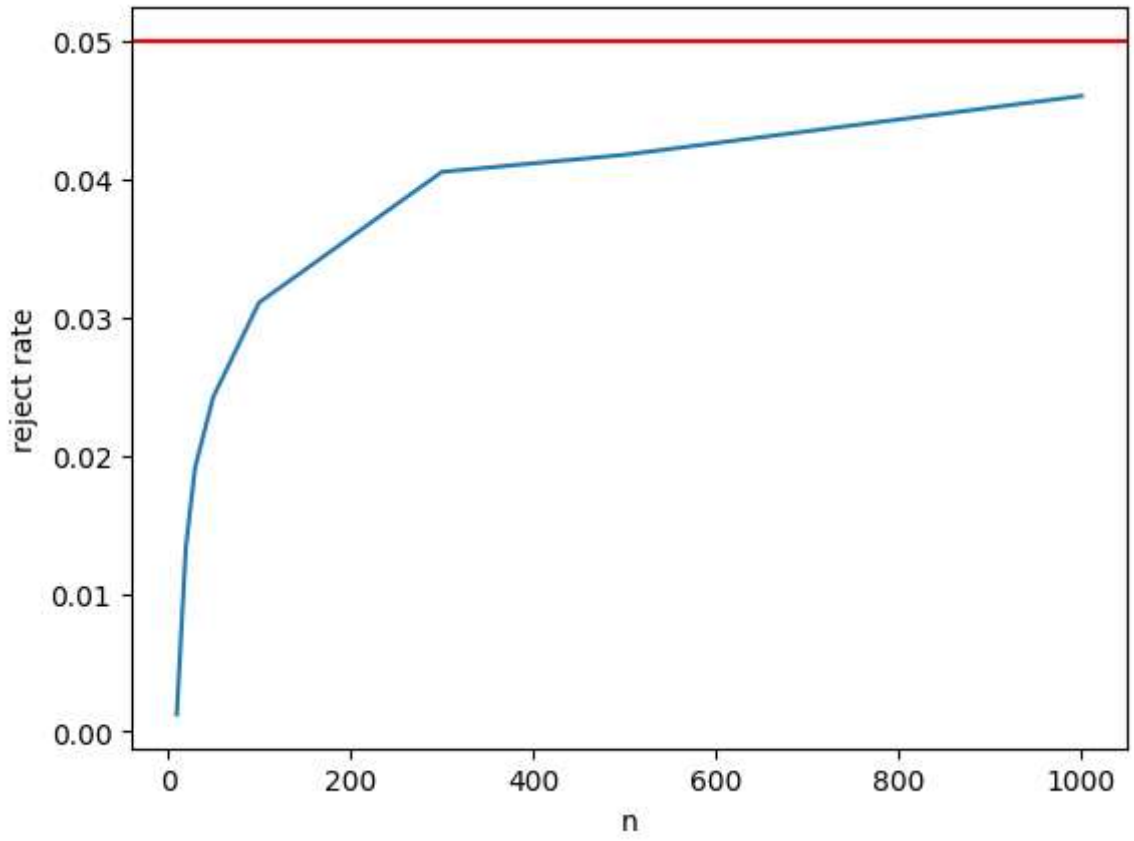


In [ ]:

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

N = 50000
n_size = [10, 20, 30, 50, 100, 300, 500, 1000]

xx = np.linspace(-4,4,1000)
yy = norm.pdf(xx)


fail_list = []

for i in range(8):
    n = n_size[i]
    X = norm.rvs(size=(N,n))
    g2 = np.sqrt(n / 24) * kurtosis(X,axis=1)
```

```
    # 計算型一誤差
    fail = np.sum(np.abs(g2)>1.96)/N
    fail_list.append(fail)
print(fail_list)

plt.plot(n_size, fail_list)
plt.xlabel('n')
plt.ylabel('reject rate')
plt.axhline(y=0.05, color='r')
plt.show()
```

```
[0.00124, 0.01342, 0.0191, 0.02424, 0.03106, 0.04052, 0.04176, 0.04602]
```



作品結論：

- 在設定常態alpha=0.05,拒絕域為正負1.96的狀況下，當reject_rate越接近alpha，則檢定力越高，代表這個分配越接近常態分配，由上圖可知，當n越來越大時，reject_rate越接近alpha，檢定力越高，代表這個分配越接近常態分配。
- 另外，g2適用於檢定常態分配的峰態，當n越大時，g2的分配越接近常態分配，因此g2適用於檢定常態分配的峰態。

同上，但統計量為

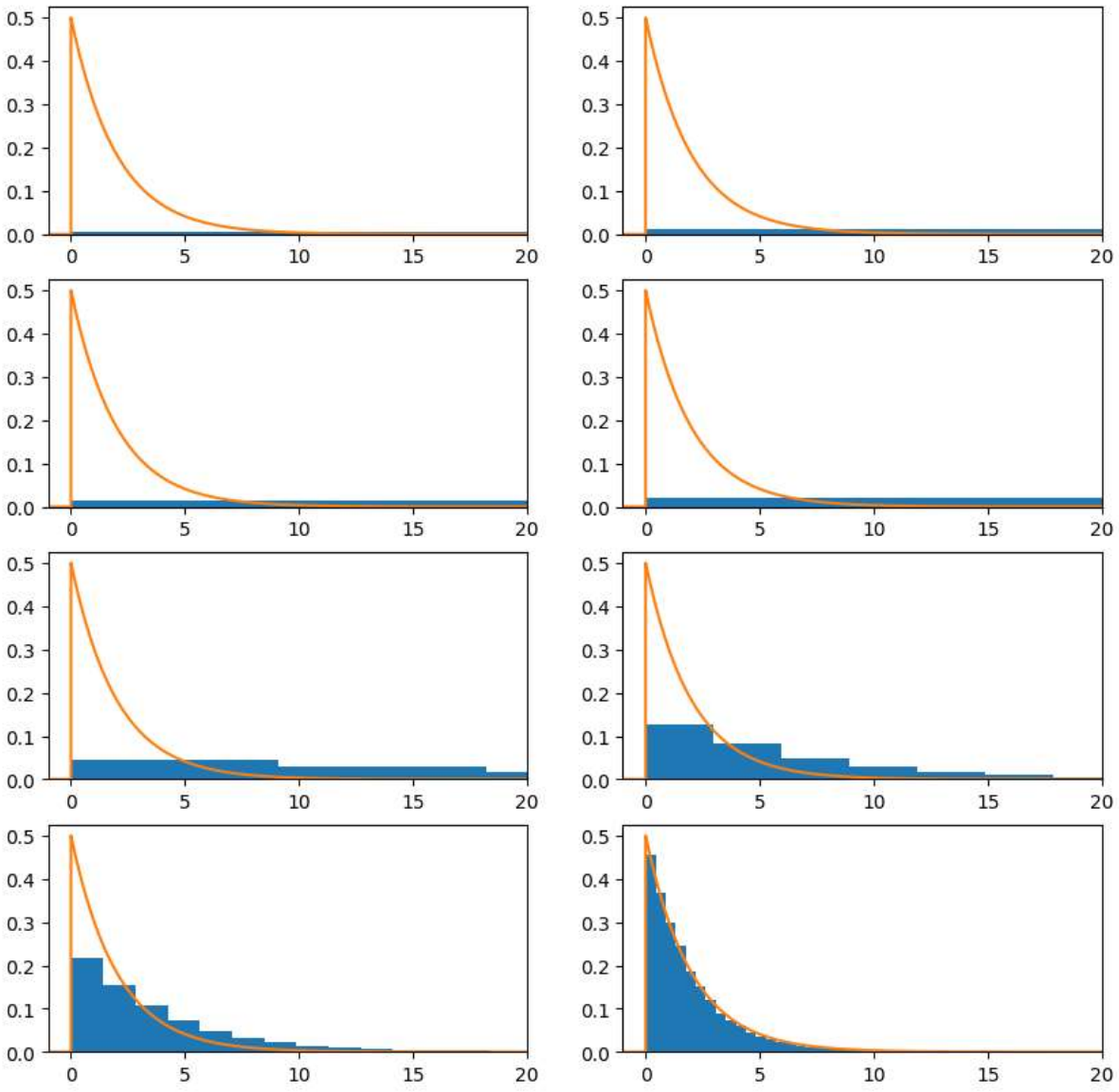
$$G_3 = G_1^2 + G_2^2 = \frac{n}{6}\left(\hat{s}^2 + \frac{(\hat{k}-3)^2}{4}\right) \ ,$$

同樣利用上述的蒙地卡羅模擬，驗證統計量 G_3 服從卡方分配 $\chi^2(2)$。

In [ ]:

```
from scipy.stats import norm, chi2
from scipy.stats import skew, kurtosis
from matplotlib import pyplot as plt
import numpy as np
```

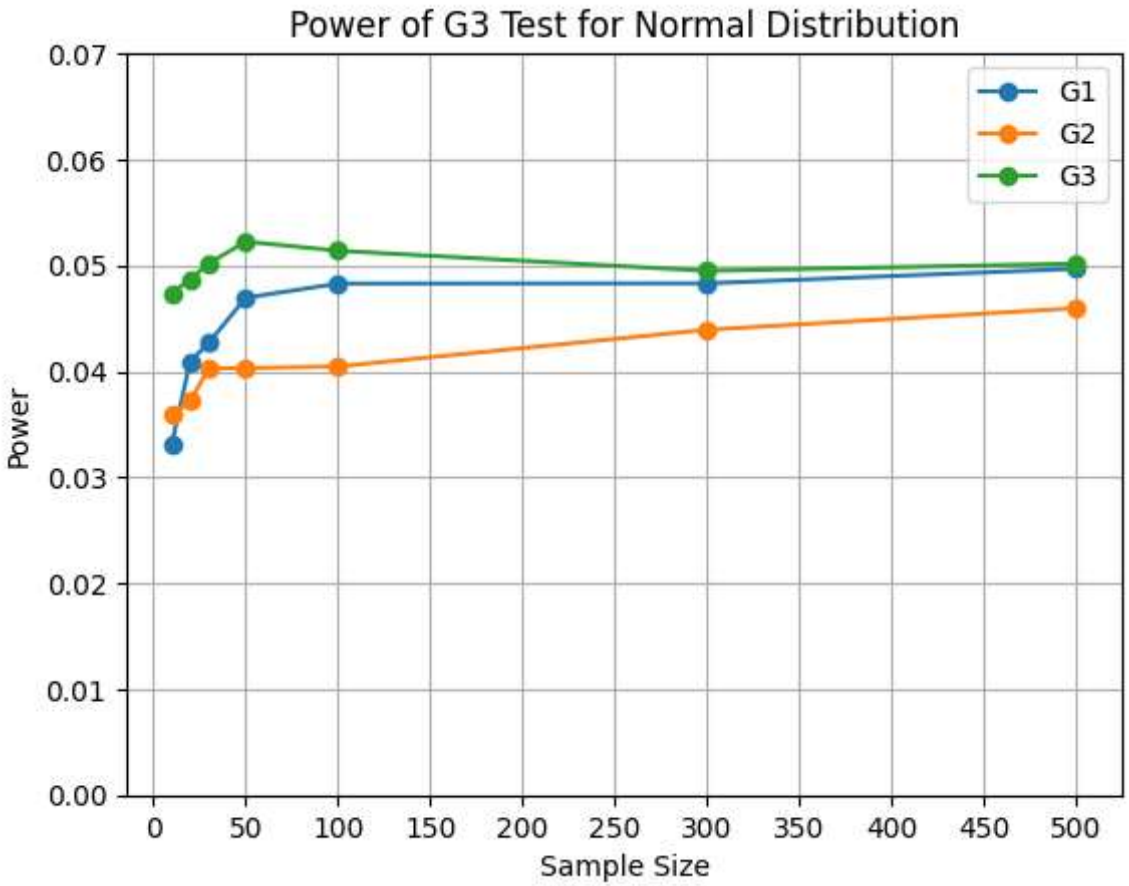
```
fig, ax = plt.subplots(4, 2, figsize=(10, 10))
n_size = [10, 20, 30, 50, 100, 300, 500, 1000]
N = 50000
for i in range(8):
    X = norm.rvs(size=(N,n_size[i]))
    g1 = np.sqrt(n / 6) * skew(X,axis=1)
    g2 = np.sqrt(n / 24) * (kurtosis(X, axis=1))
    ax[i//2, i%2].hist(g1**2+g2**2, bins=100, density=True, label='g3')

    xx = np.linspace(-1, 50, 100000)

    # Calculate the pdf values
    pdf_values = chi2.pdf(xx, df=2)

    # Plot the pdf
    ax[i//2, i%2].plot(xx, pdf_values, label='chi2')
    # 設定x軸的範圍
    ax[i//2, i%2].set_xlim([-1, 20])
    # ax[i//2, i%2].set_title('n='+str(n_size[i]))
plt.show()
```



In [ ]:

```
from cProfile import label
from matplotlib import pyplot as plt
import numpy as np
from scipy.stats import norm, skew,kurtosis
```

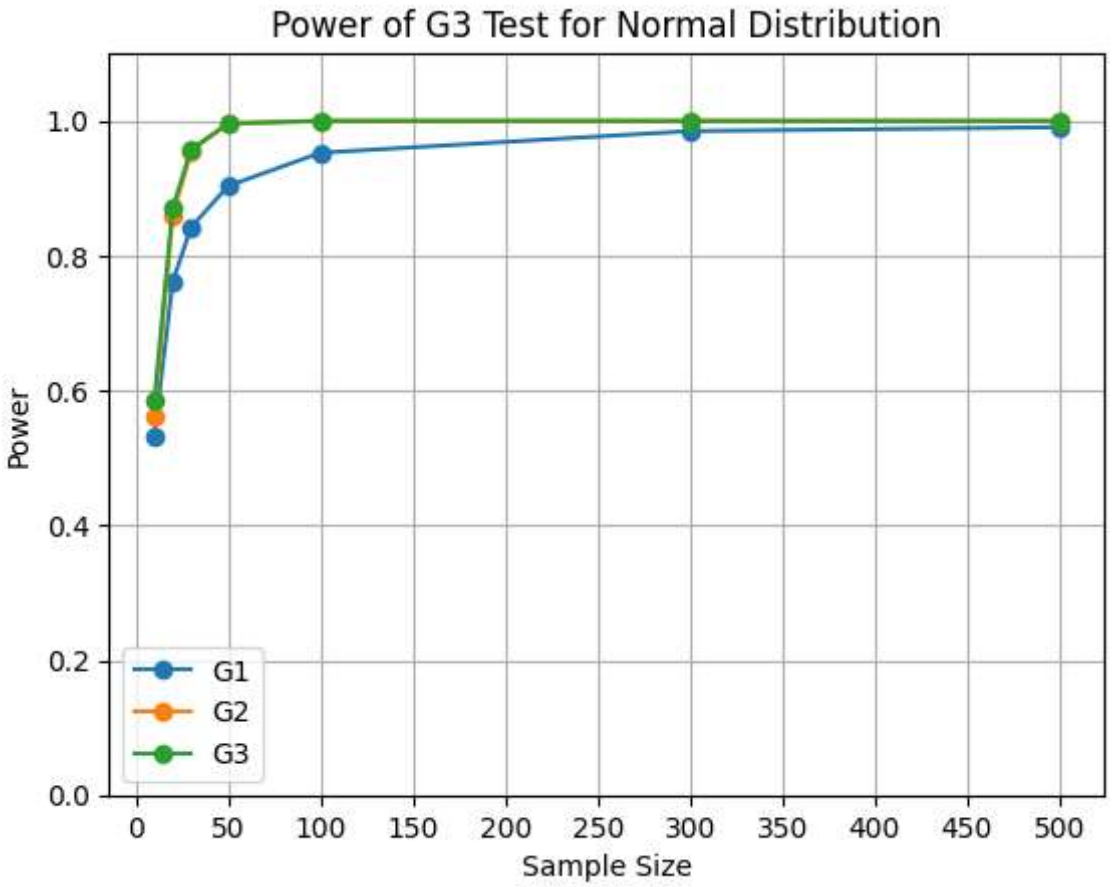
```
sample_sizes = [10, 20, 30, 50, 100,300,500]
N = 50000
alpha = 0.05
cv1=norm.ppf(1-alpha/2)
cv = chi2.ppf(1 - alpha,df=2)
powers = []
powers1=[]
powers2=[]

for n in sample_sizes:
    x = norm.rvs(0,1,size=(N, n))
    g1 = np.sqrt(n / 6) * skew(x, axis=1,bias=False)
    g2= np.sqrt(n/24)*(kurtosis(x,fisher=True,bias=False,axis=1))
    g3= g1**2+g2**2
    reject = np.abs(g3) > cv
    reject1 = np.abs(g1) > cv1
    reject2 = np.abs(g2) > cv1
    reject_rate1 = np.mean(reject1)
    reject_rate2 = np.mean(reject2)
    reject_rate = np.mean(reject)
    powers1.append(reject_rate1)
    powers2.append(reject_rate2)
    powers.append(reject_rate)


plt.ylim(0,0.07)
plt.xticks( np.arange(0,550,50))
plt.plot(sample_sizes, powers1, marker='o',label='G1')
plt.plot(sample_sizes, powers2, marker='o',label='G2')
plt.plot(sample_sizes, powers, marker='o',label='G3')
plt.xlabel('Sample Size')
plt.legend()
plt.grid(True)
plt.ylabel('Power')
plt.title('Power of G3 Test for Normal Distribution')
plt.show()
```



In [ ]:

```
from cProfile import label
from matplotlib import pyplot as plt
import numpy as np
from scipy.stats import norm, skew,kurtosis, t, chi2
sample_sizes = [10, 20, 30, 50, 100,300,500]
N = 50000
alpha = 0.05
cv1=norm.ppf(1-alpha/2)
cv = chi2.ppf(1 - alpha,df=2)
powers = []
powers1=[]
powers2=[]

for n in sample_sizes:
    x = t.rvs(1,size=(N, n))

    g1 = np.sqrt(n / 6) * skew(x, axis=1,bias=False)
    g2= np.sqrt(n/24)*(kurtosis(x,fisher=True,bias=False,axis=1))
    g3= g1**2+g2**2
    reject = np.abs(g3) > cv
    reject1 = np.abs(g1) > cv1
    reject2 = np.abs(g2) > cv1
    reject_rate1 = np.mean(reject1)
    reject_rate2 = np.mean(reject2)
    reject_rate = np.mean(reject)
    powers1.append(reject_rate1)
    powers2.append(reject_rate2)
    powers.append(reject_rate)


plt.ylim(0,1.1)
plt.xticks(np.arange(0,550,50))
plt.plot(sample_sizes, powers1, marker='o',label='G1')
plt.plot(sample_sizes, powers2, marker='o',label='G2')
plt.plot(sample_sizes, powers, marker='o',label='G3')
```

```
plt.xlabel('Sample Size')
plt.legend()
plt.grid(True)
plt.ylabel('Power')
plt.title('Power of G3 Test for Normal Distribution')
plt.show()
```



作品結論：

- g1是檢定常態分配的偏態，g2是檢定常態分配的峰態，g3是檢定常態分配的偏態與峰態，由上圖可知，當n越來越大時，reject_rate越接近alpha，檢定力越接近alpha，代表這個分配越接近常態分配。

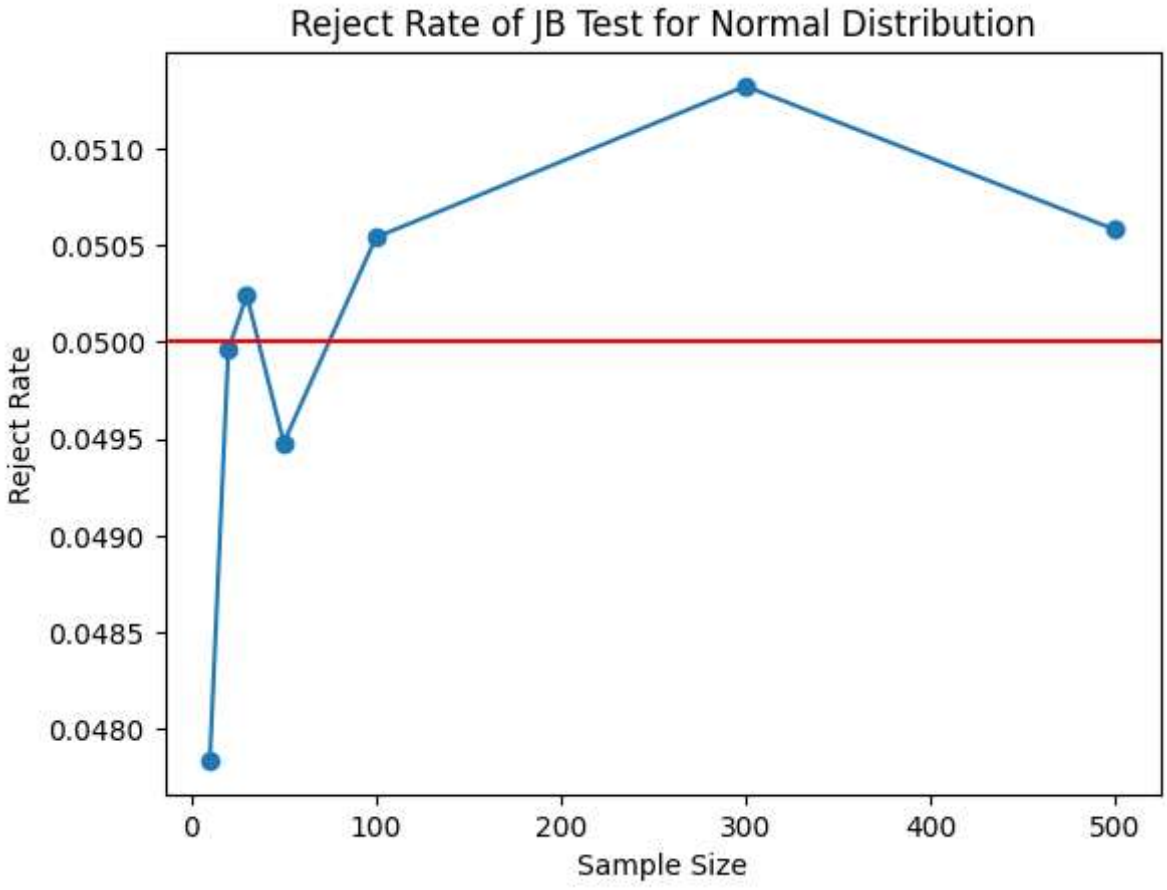
觀察不同分配下，使用J-B統計量檢定力的變化。

將上述驗證程式改寫為一個副程式，假設取名為 stats, p_value = JB_test(x)，輸入參數 x 代表欲檢定是否為常態的一組資料。 輸出兩個結果，stats 為 G_3 檢定統計量的值，p_value 為檢定的 p-value。

In [ ]:
```
def jb_test(x):
    n = x.shape[1]
    s = skew(x, bias=False,axis=1)
    G1 = np.sqrt(n / 6) * s
    k = kurtosis(x, bias=False,axis=1)
    G2 = np.sqrt(n / 24) * (k)
    stats = G1**2 + G2**2
    p_value = 1 - chi2.cdf(stats, df=2)
    return [stats, p_value]
```

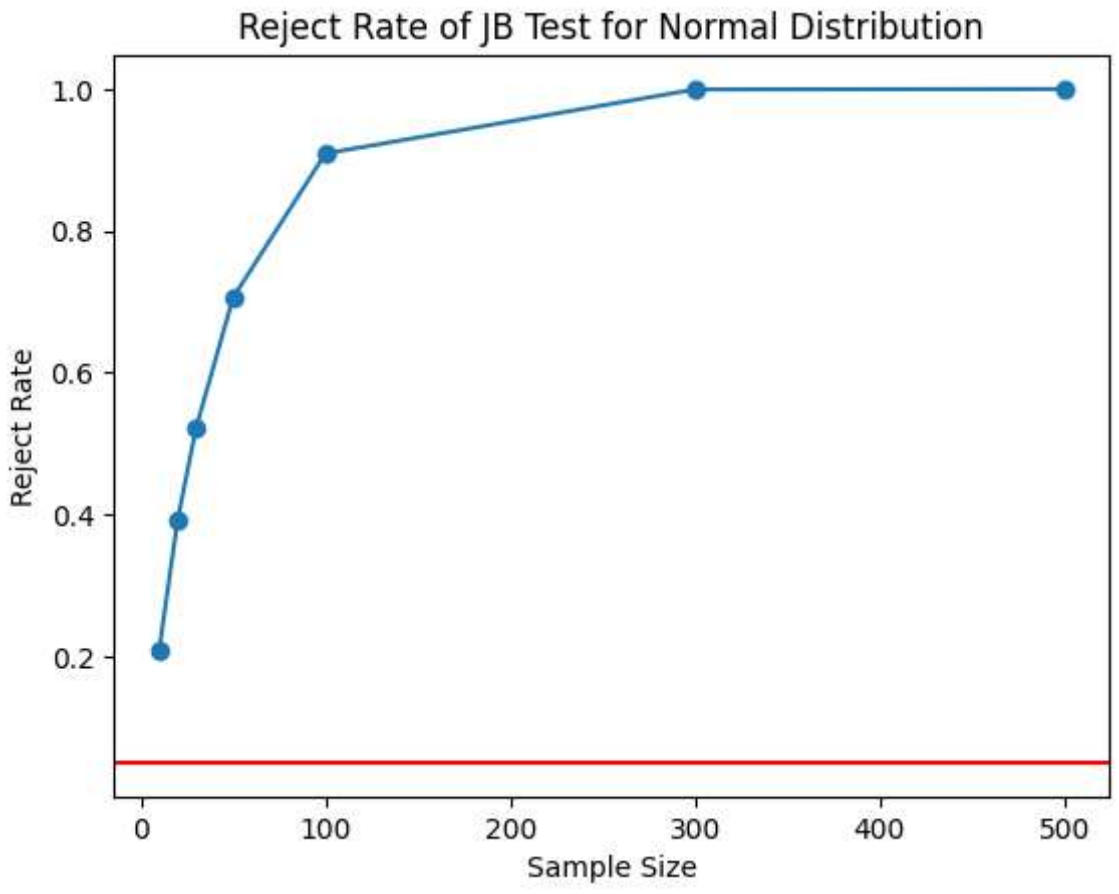
In [ ]:
```
from cProfile import label
from matplotlib import pyplot as plt
import numpy as np
from scipy.stats import norm, skew,kurtosis,chi2

n = [10, 20, 30, 50, 100,300,500]
N = 50000
reject_rate_list = []
alpha = 0.05
cv = chi2.ppf(1 - alpha,df=2)
for i in range(7):
    x = norm.rvs(0,1,size=(N, n[i]))
    stats, p_value = jb_test(x)
    reject = np.abs(stats) > cv

    reject_rate = np.mean(reject)
    print('n=',n[i],'reject_rate=',reject_rate)
    reject_rate_list.append(reject_rate)

plt.plot(n, reject_rate_list, marker='o')
plt.xlabel('Sample Size')
plt.ylabel('Reject Rate')
plt.title('Reject Rate of JB Test for Normal Distribution')
plt.axhline(y=0.05, color='r')
plt.show()
```

```
n= 10 reject_rate= 0.04784
n= 20 reject_rate= 0.04996
n= 30 reject_rate= 0.05024
n= 50 reject_rate= 0.04948
n= 100 reject_rate= 0.05054
n= 300 reject_rate= 0.05132
n= 500 reject_rate= 0.05058
```



In [ ]:
```
from cProfile import label
from matplotlib import pyplot as plt
import numpy as np
from scipy.stats import norm, skew,kurtosis,chi2,t

n = [10, 20, 30, 50, 100,300,500]
N = 50000

reject_rate_list = []
for i in range(7):
    x = t.rvs(df=3,size=(N, n[i]))
    stats, p_value = jb_test(x)
    reject = np.abs(stats) > cv
    reject_rate = np.mean(reject)
    print('n=',n[i],'reject_rate=',reject_rate)
    reject_rate_list.append(reject_rate)

plt.plot(n, reject_rate_list, marker='o')
plt.xlabel('Sample Size')
plt.ylabel('Reject Rate')
plt.title('Reject Rate of JB Test for Normal Distribution')
plt.axhline(y=0.05, color='r')
plt.show()
```

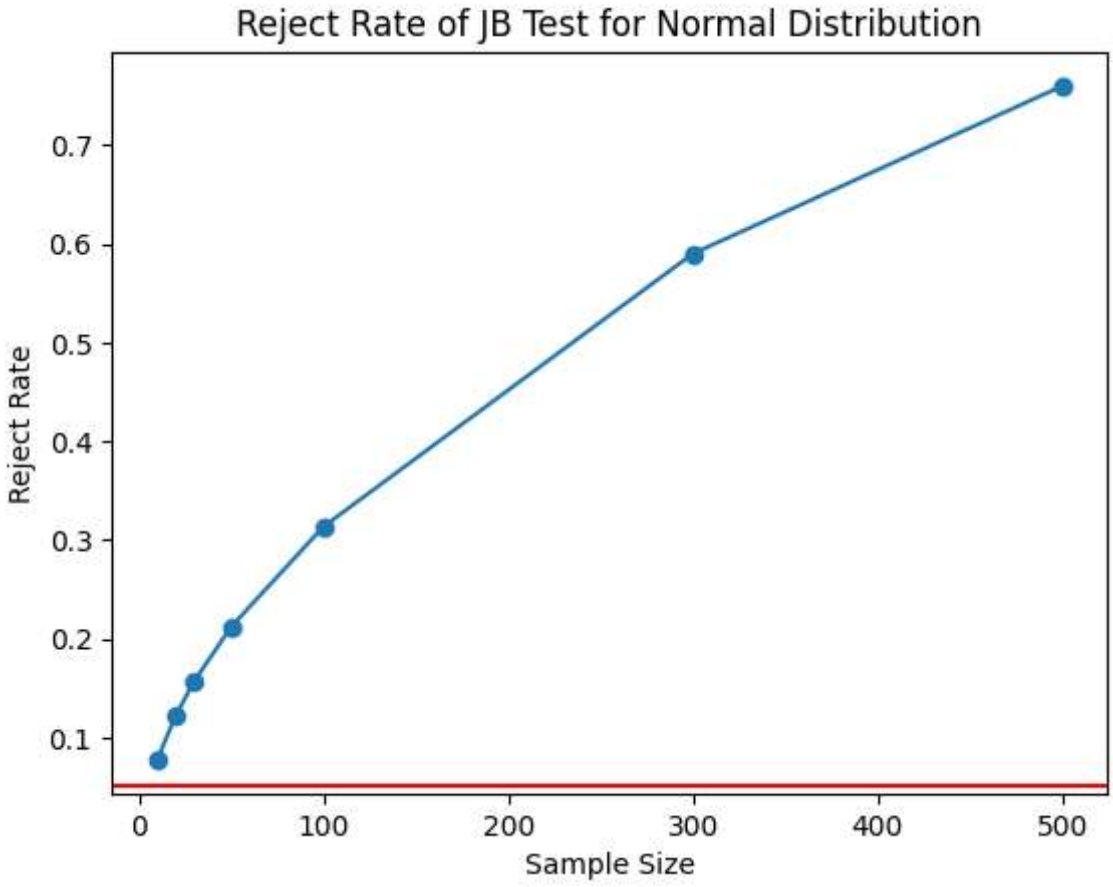
```
n= 10 reject_rate= 0.20922
n= 20 reject_rate= 0.39296
n= 30 reject_rate= 0.52246
n= 50 reject_rate= 0.70594
n= 100 reject_rate= 0.90908
n= 300 reject_rate= 0.99946
n= 500 reject_rate= 1.0
```



In [ ]:
```
from cProfile import label
from matplotlib import pyplot as plt
import numpy as np
```

```
from scipy.stats import norm, skew,kurtosis,chi2,t

n = [10, 20, 30, 50, 100,300,500]
N = 50000

reject_rate_list = []
for i in range(7):
    x = t.rvs(df=10,size=(N, n[i]))
    stats, p_value = jb_test(x)
    reject = p_value < 0.05
    reject_rate = np.mean(reject)
    print('n=',n[i],'reject_rate=',reject_rate)
    reject_rate_list.append(reject_rate)

plt.plot(n, reject_rate_list, marker='o')
plt.xlabel('Sample Size')
plt.ylabel('Reject Rate')
plt.title('Reject Rate of JB Test for Normal Distribution')
plt.axhline(y=0.05, color='r')
plt.show()
```

```
n= 10 reject_rate= 0.0775
n= 20 reject_rate= 0.1221
n= 30 reject_rate= 0.1568
n= 50 reject_rate= 0.21196
n= 100 reject_rate= 0.31356
n= 300 reject_rate= 0.59016
n= 500 reject_rate= 0.76006
```

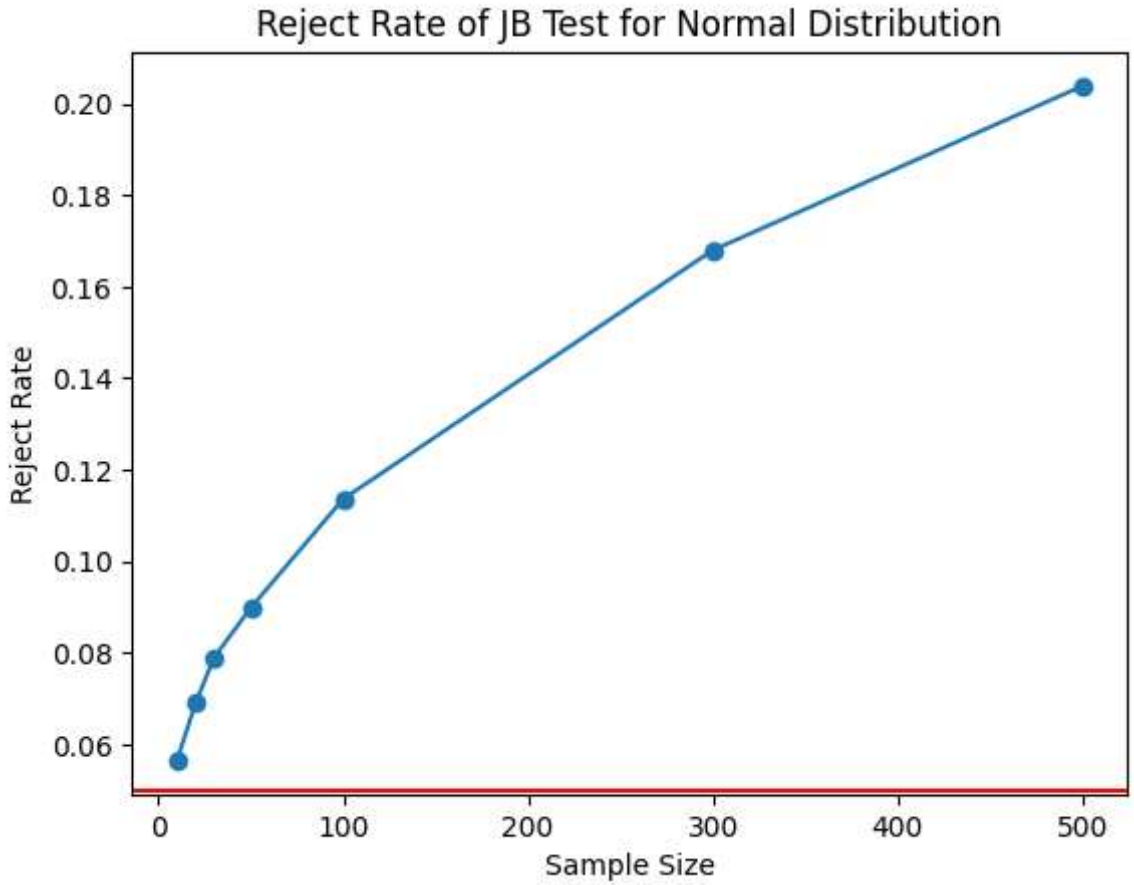


In [ ]:

```
from cProfile import label
from matplotlib import pyplot as plt
import numpy as np
from scipy.stats import norm, skew,kurtosis,chi2,t

n = [10, 20, 30, 50, 100,300,500]
N = 50000
```

```
reject_rate_list = []
for i in range(7):
    x = t.rvs(df=30,size=(N, n[i]))
    stats, p_value = jb_test(x)
    reject = p_value < 0.05
    reject_rate = np.mean(reject)
    print('n=',n[i],'reject_rate=',reject_rate)
    reject_rate_list.append(reject_rate)

plt.plot(n, reject_rate_list, marker='o')
plt.xlabel('Sample Size')
plt.ylabel('Reject Rate')
plt.title('Reject Rate of JB Test for Normal Distribution')
plt.axhline(y=0.05, color='r')
plt.show()
```

```
n= 10 reject_rate= 0.0565
n= 20 reject_rate= 0.06898
n= 30 reject_rate= 0.07894
n= 50 reject_rate= 0.08986
n= 100 reject_rate= 0.11362
n= 300 reject_rate= 0.16794
n= 500 reject_rate= 0.2038
```

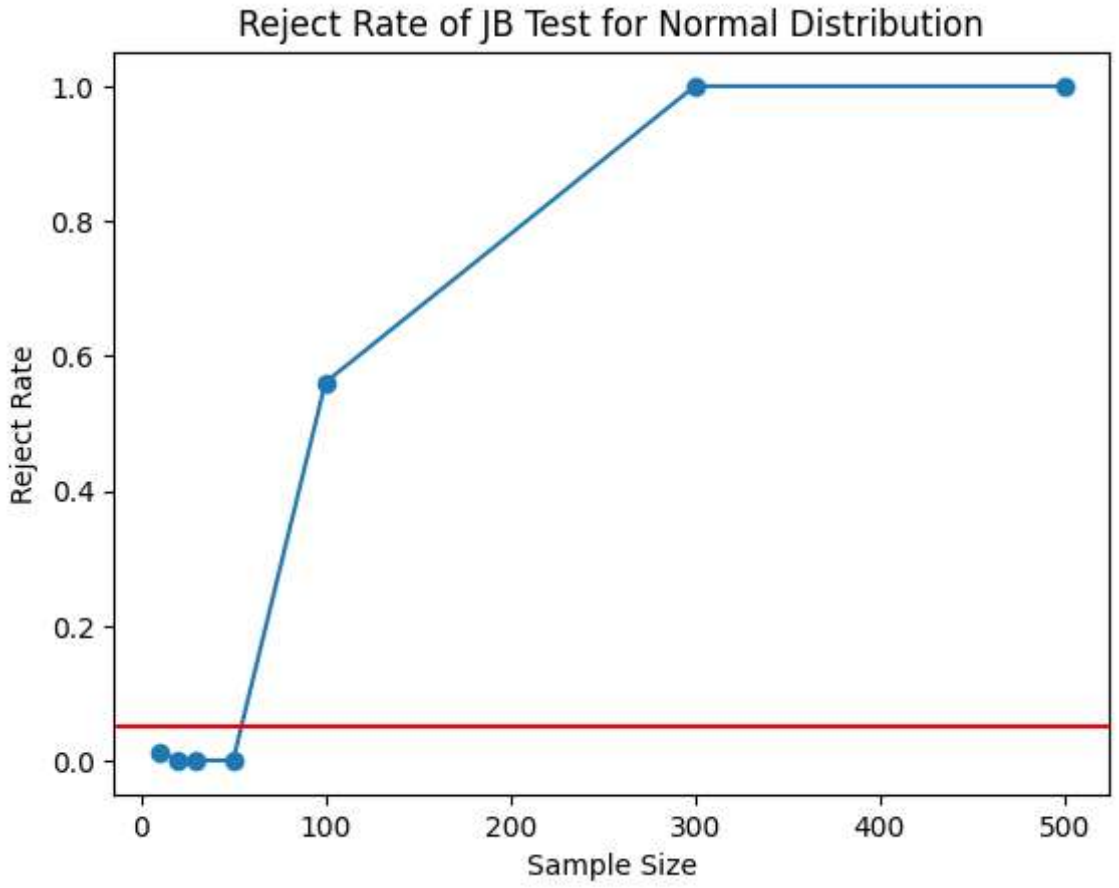


In [ ]:

```
from cProfile import label
from matplotlib import pyplot as plt
import numpy as np
from scipy.stats import norm, skew,kurtosis,chi2,t,uniform

n = [10, 20, 30, 50, 100,300,500]
N = 50000

reject_rate_list = []
for i in range(7):
    x = uniform.rvs(0,1,size=(N, n[i]))
```

```
    stats, p_value = jb_test(x)
    reject = p_value < 0.05
    reject_rate = np.mean(reject)
    print('n=',n[i],'reject_rate=',reject_rate)
    reject_rate_list.append(reject_rate)

plt.plot(n, reject_rate_list, marker='o')
plt.xlabel('Sample Size')
plt.ylabel('Reject Rate')
plt.title('Reject Rate of JB Test for Normal Distribution')
plt.axhline(y=0.05, color='r')
plt.show()
```

```
n= 10 reject_rate= 0.01288
n= 20 reject_rate= 0.00138
n= 30 reject_rate= 0.00046
n= 50 reject_rate= 0.00042
n= 100 reject_rate= 0.56092
n= 300 reject_rate= 1.0
n= 500 reject_rate= 1.0
```

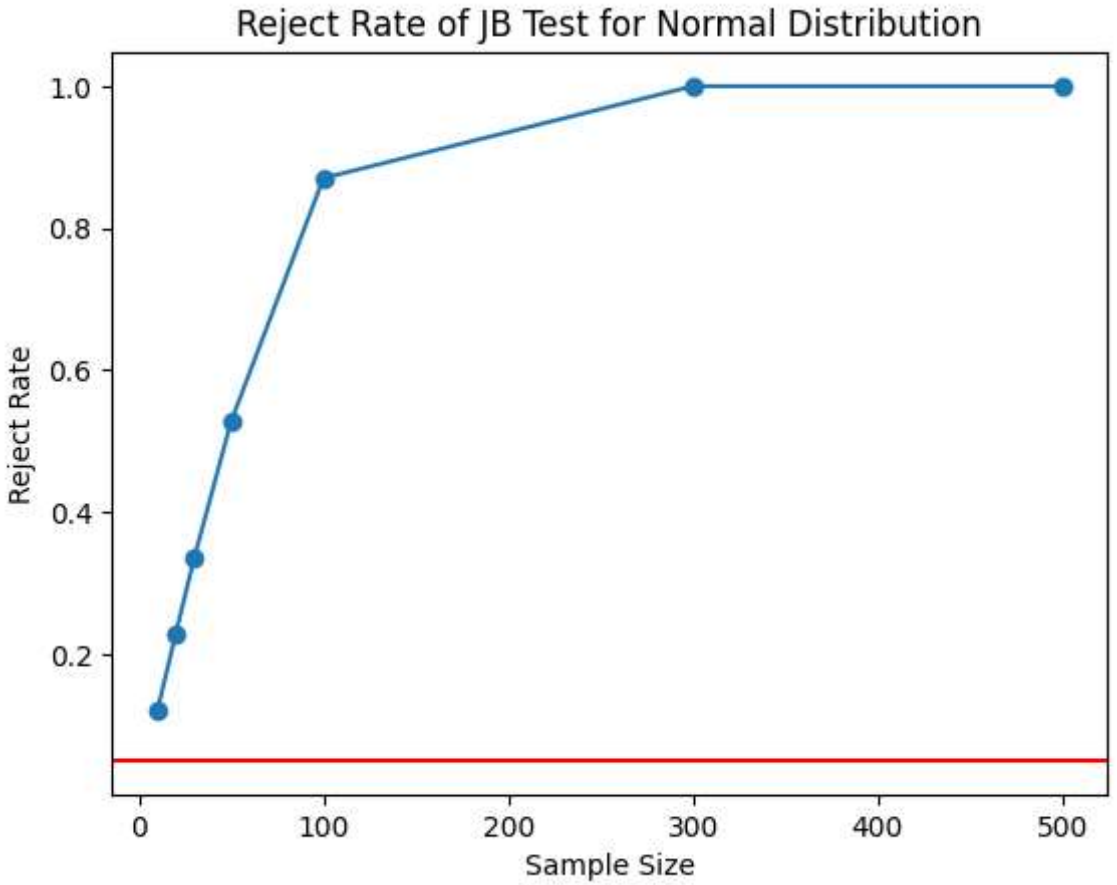


In [ ]:

```
from cProfile import label
from matplotlib import pyplot as plt
import numpy as np
from scipy.stats import norm, skew,kurtosis,chi2,t,uniform

n = [10, 20, 30, 50, 100,300,500]
N = 50000

reject_rate_list = []
for i in range(7):
    x = chi2.rvs(df=8,size=(N, n[i]))
    stats, p_value = jb_test(x)
    reject = p_value < 0.05
    reject_rate = np.mean(reject)
    print('n=',n[i],'reject_rate=',reject_rate)
```

```
    reject_rate_list.append(reject_rate)

plt.plot(n, reject_rate_list, marker='o')
plt.xlabel('Sample Size')
plt.ylabel('Reject Rate')
plt.title('Reject Rate of JB Test for Normal Distribution')
plt.axhline(y=0.05, color='r')
plt.show()
```

```
n= 10 reject_rate= 0.1192
n= 20 reject_rate= 0.22804
n= 30 reject_rate= 0.33504
n= 50 reject_rate= 0.52748
n= 100 reject_rate= 0.87046
n= 300 reject_rate= 1.0
n= 500 reject_rate= 1.0
```



作品結論：

- $Power = P(Reject\ H_0 \mid H_a)$，以上面的情況來說，$H_0$為常態分配，$H_a$為非常態分配，當檢定力越高，代表$P(Reject\ H_0 \mid H_a)$越高，也就是說，當檢定力越高，代表我們越能夠拒絕$H_0$，當n越大時，power會越高

測試不同檢定量的檢定力。

其他統計量的檢定，例如 Shapiro-Wilk test, Kolmogorov-Smirnov test, Anderson-Darling test 等，也可以利用蒙地卡羅模擬來驗證其檢定力。

In [ ]:
```
from scipy.stats import jarque_bera,anderson,normaltest

from cProfile import label
```

```python
from matplotlib import pyplot as plt
import numpy as np
from scipy.stats import norm, skew,kurtosis,chi2,t,uniform

n = [10, 20, 30, 50, 100,300,500]
N = 50000

reject_rate_list = []
jb_reject_rate_list = []
k2_reject_rate_list = []

for i in range(7):
    x = t.rvs(df=3,size=(N, n[i]))
    stats, p_value = jb_test(x)
    reject = p_value < 0.05
    reject_rate = np.mean(reject)
    print('n=',n[i],'reject_rate=',reject_rate)

    jb_stat, jb_p_value = jarque_bera(x,axis=1)
    jb_reject = jb_p_value < 0.05
    jb_reject_rate = np.mean(jb_reject)
    print('n=',n[i],'jb_reject_rate=',jb_reject_rate)

    k2, p = normaltest(x,axis=1)
    k2_reject = p < 0.05
    k2_reject_rate = np.mean(k2_reject)
    print('n=',n[i],'k2_reject_rate=',k2_reject_rate)



    reject_rate_list.append(reject_rate)
    jb_reject_rate_list.append(jb_reject_rate)
    k2_reject_rate_list.append(k2_reject_rate)

    print()


plt.plot(n, reject_rate_list, marker='o',label='G3')
plt.plot(n, jb_reject_rate_list, marker='o',label='JB')
plt.plot(n, k2_reject_rate_list, marker='o',label='K2')
plt.xlabel('Sample Size')
plt.ylabel('Reject Rate')
plt.title('Reject Rate of JB Test for Normal Distribution')
plt.axhline(y=0.05, color='r')
plt.legend()
plt.show()
```

```
n= 10 reject_rate= 0.21236
n= 10 jb_reject_rate= 0.10002
```

```
c:\Users\asd11\Documents\GitHub\Statistical-Computing\myenv\Lib\site-packages\scipy\stats\_stats_py.py:1806: UserWarning: kurtosistest only valid for n>=20 ... continuing anyway, n=10
  warnings.warn("kurtosistest only valid for n>=20 ... continuing "
```

```
n= 10 k2_reject_rate= 0.2321

n= 20 reject_rate= 0.38954
n= 20 jb_reject_rate= 0.30816
n= 20 k2_reject_rate= 0.38378

n= 30 reject_rate= 0.52628
n= 30 jb_reject_rate= 0.4607
n= 30 k2_reject_rate= 0.50082

n= 50 reject_rate= 0.70858
n= 50 jb_reject_rate= 0.66686
n= 50 k2_reject_rate= 0.66598

n= 100 reject_rate= 0.9115
n= 100 jb_reject_rate= 0.89786
n= 100 k2_reject_rate= 0.87876

n= 300 reject_rate= 0.99942
n= 300 jb_reject_rate= 0.99928
n= 300 k2_reject_rate= 0.99866

n= 500 reject_rate= 0.99998
n= 500 jb_reject_rate= 0.99998
n= 500 k2_reject_rate= 0.99998
```