作品四、以蒙地卡羅實驗驗證 **J-B** 檢定統計量

姓名：黃冠翔
學號：410978040

1.

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

$$G_1 = \sqrt{\frac{n}{6}}\hat{s}$$

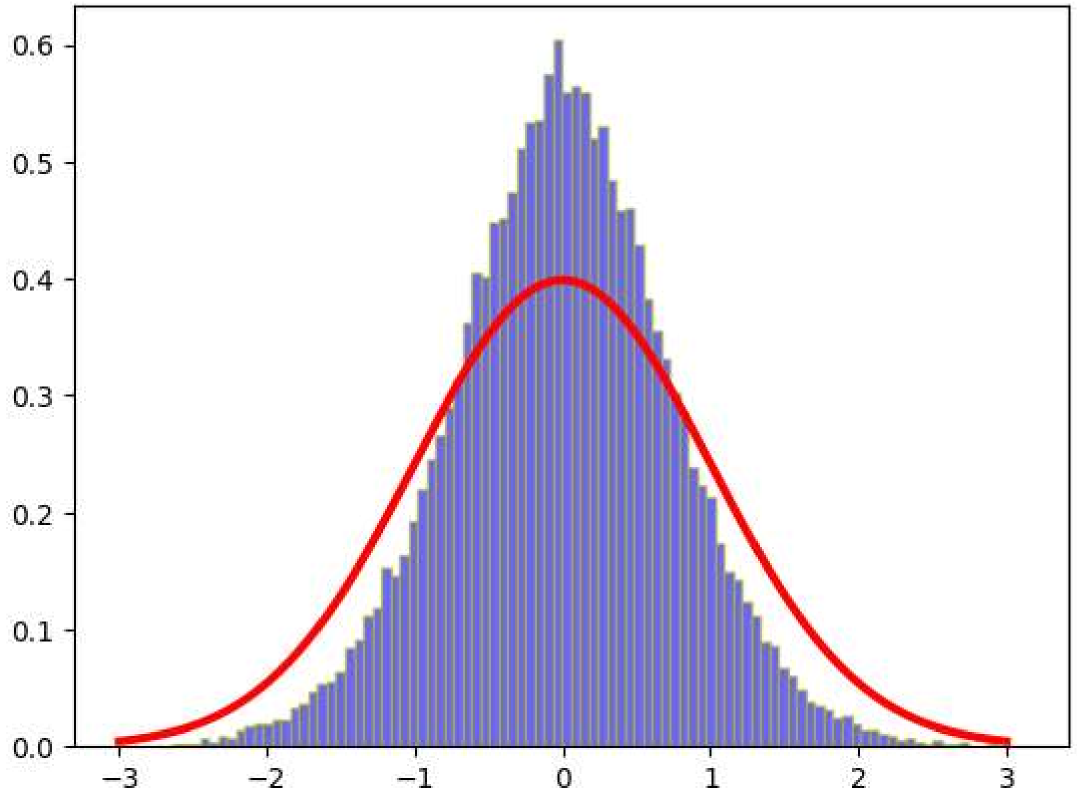
，

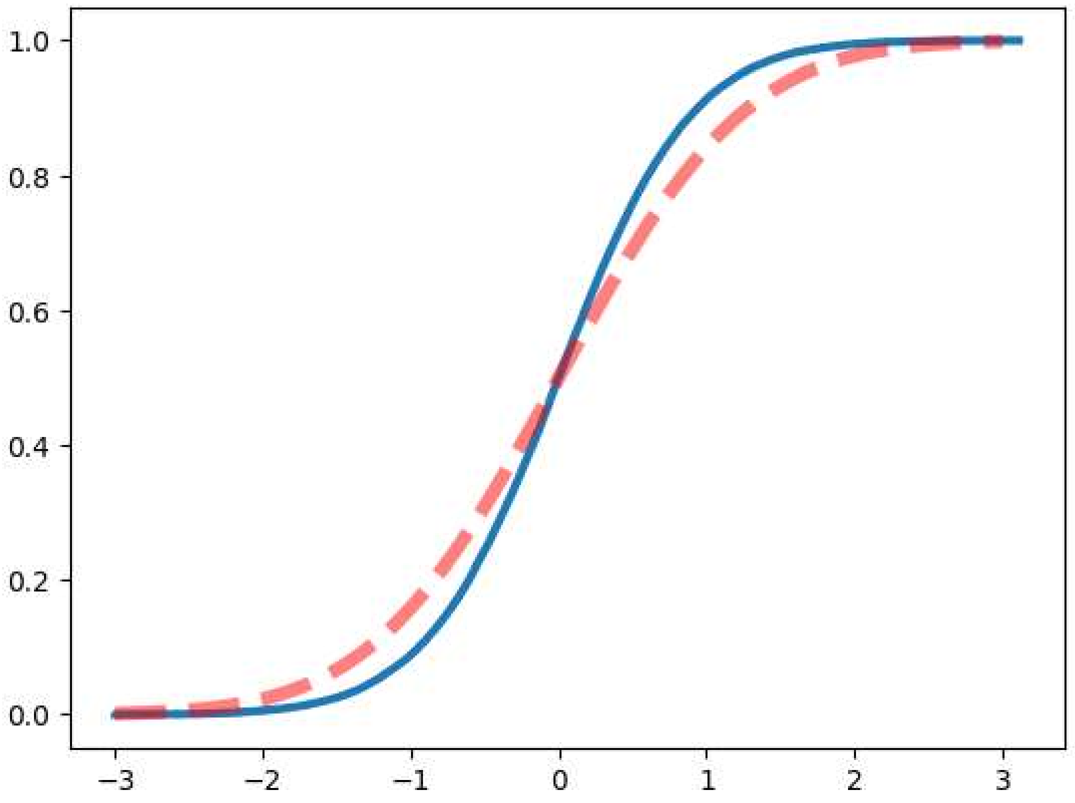
其中 $\hat{s}$ 為偏態係數（skewness）的估計值（參考指令 scipy.stats.skew）。請利用蒙地卡羅模擬（Monte Carlo Simulation）驗證統計量 G_1 服從標準常態 $N(0,1)$。其中蒙地卡羅模擬的環境設定（scenarios）為：

- 樣本數 $n = 10, 20, 30, 50, 100, 300, 500, 1000$。
- 針對每個樣本數 n，模擬次數皆為 $N = 50,000$。
- 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。

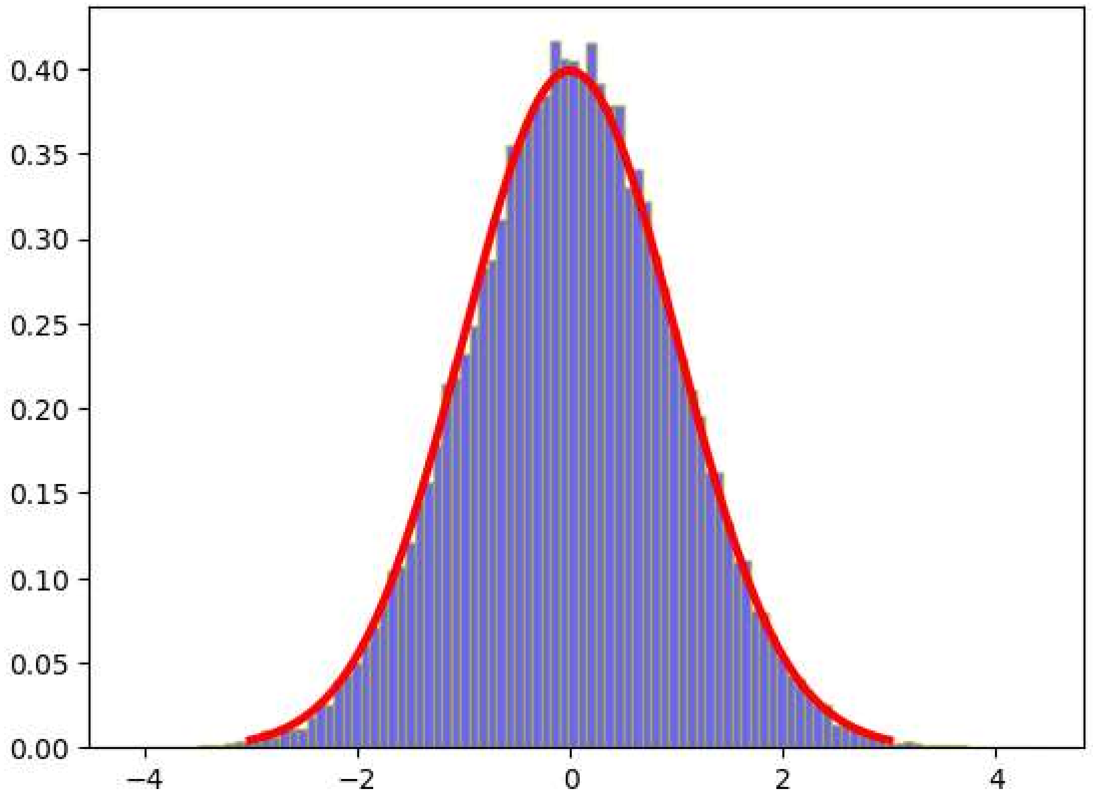
In [ ]:
```python
from scipy.stats import norm, skew
import matplotlib.pyplot as plt
import numpy as np

n = 10 #樣本數
N = 50000 #實驗次數

xx = np.linspace(-3, 3, 200)
norm_pdf = norm.pdf(xx)
plt.plot(xx, norm_pdf, lw=3, color='r')

x1 = norm.rvs(loc = 0, scale = 1, size = (n, N))
G1 = np.sqrt(n / 6)*skew(x1)
plt.hist(G1, bins = 100, alpha = 0.6, color = 'b', edgecolor = 'y', density = True)
plt.show()

x_sort = np.sort(G1)
ecdf = np.arange(1, len(G1)+1) / len(G1)
```

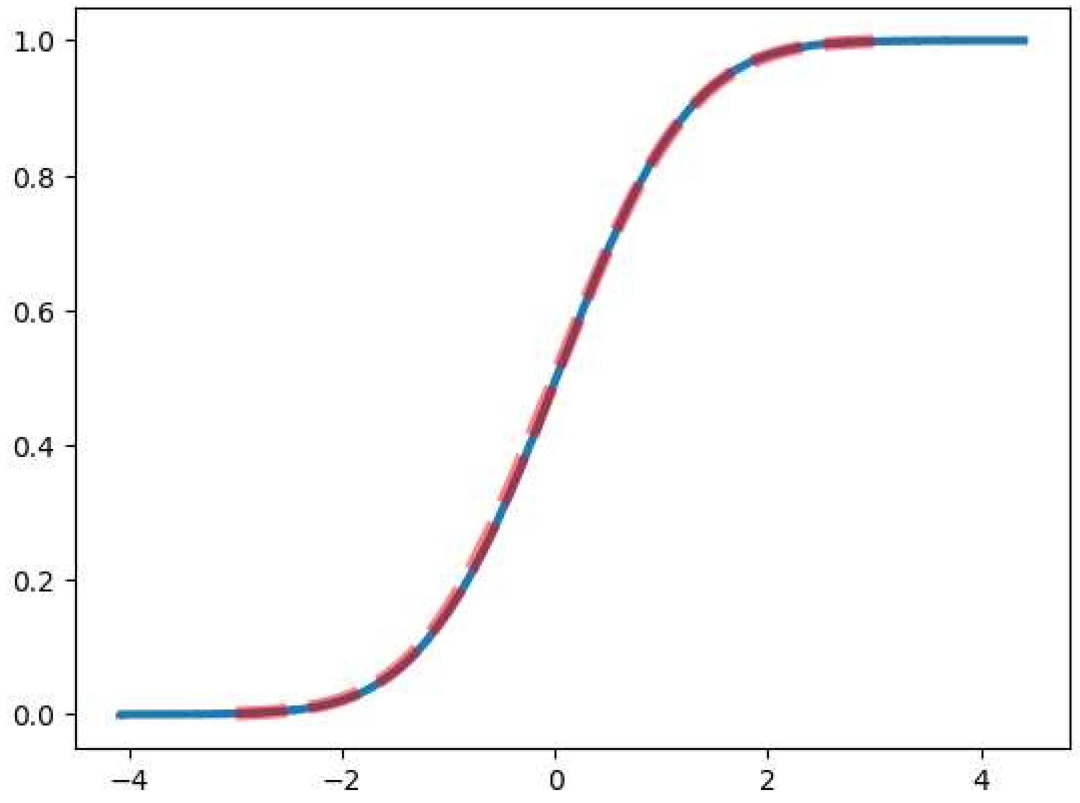
```
plt.plot(x_sort, ecdf, drawstyle = 'steps-pre', lw=3)
norm_cdf = norm.cdf(xx)
plt.plot(xx, norm_cdf, linestyle = '--', lw=5, color='r', alpha=0.5)
plt.show()

n1 = 500 #樣本數
N1 = 50000 #實驗次數

xx = np.linspace(-3, 3, 200)
norm_pdf = norm.pdf(xx)
plt.plot(xx, norm_pdf, lw=3, color='r')

x1 = norm.rvs(loc = 0, scale = 1, size = (n1, N1))
G1 = np.sqrt(n1 / 6)*skew(x1)
plt.hist(G1, bins = 100, alpha = 0.6, color = 'b', edgecolor = 'y', density = True)
plt.show()

x_sort1 = np.sort(G1)
ecdf1 = np.arange(1, len(G1)+1) / len(G1)
plt.plot(x_sort1, ecdf1, drawstyle = 'steps-pre', lw=3)
norm_cdf = norm.cdf(xx)
plt.plot(xx, norm_cdf, linestyle = '--', lw=5, color='r', alpha=0.5)
plt.show()
```


0.6
0.5
0.4
0.3
0.2
0.1
0.0
−3
−2
−1
0
1
2
3



1.0
0.8
0.6
0.4
0.2
0.0
−3
−2
−1
0
1
2
3


0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−4
−2
0
2
4



討論：

- 當樣本數較大時，G_1會愈趨近標準常態 $N(0,1)$。。

2.

同上，但令統計量為

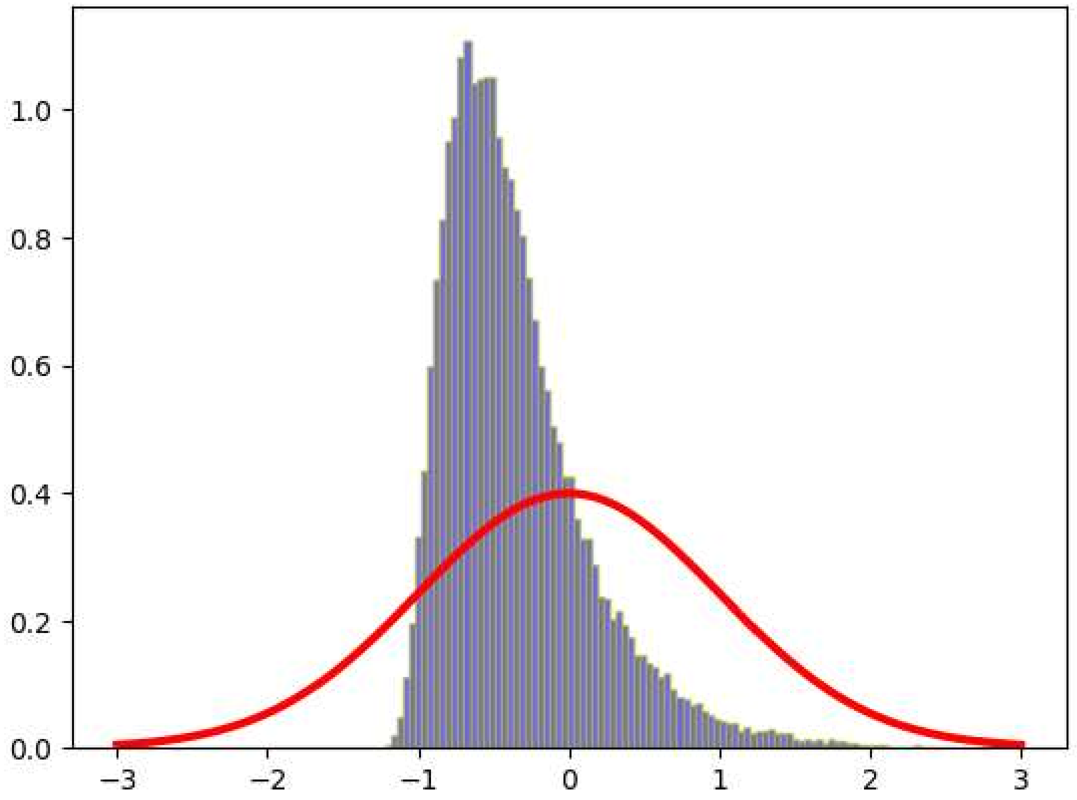
$$G_2 = \sqrt{\frac{n}{24}}(\hat{k} - 3)$$

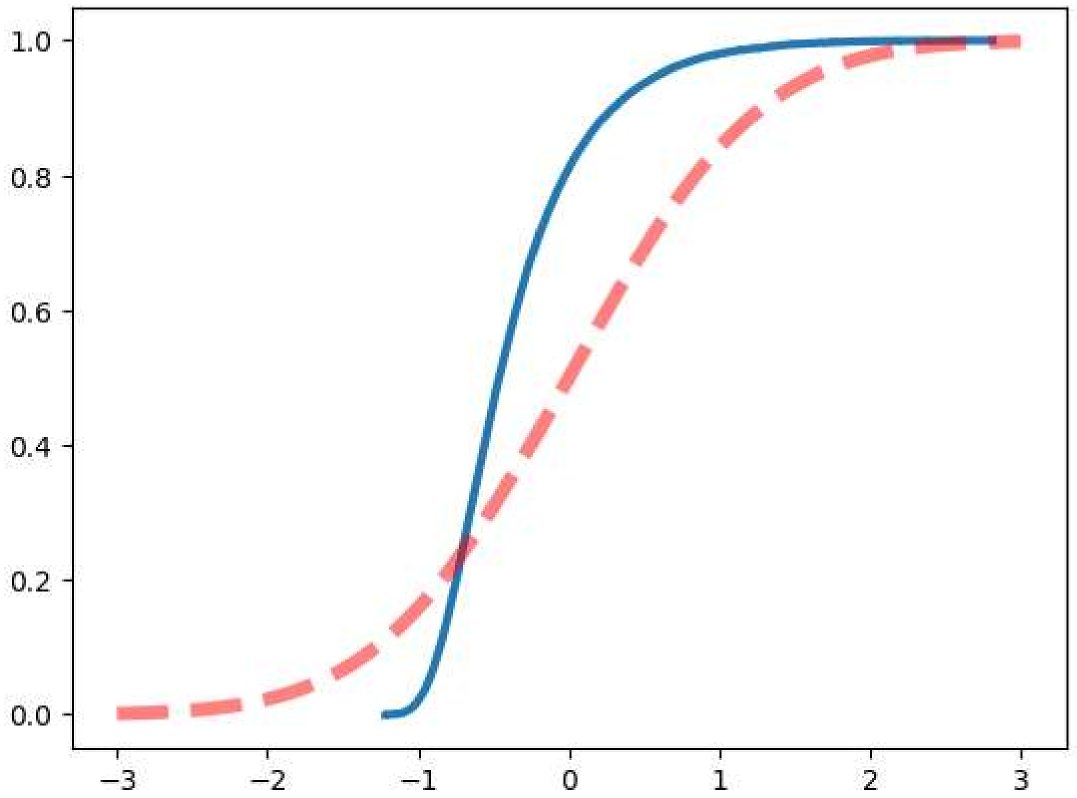
，

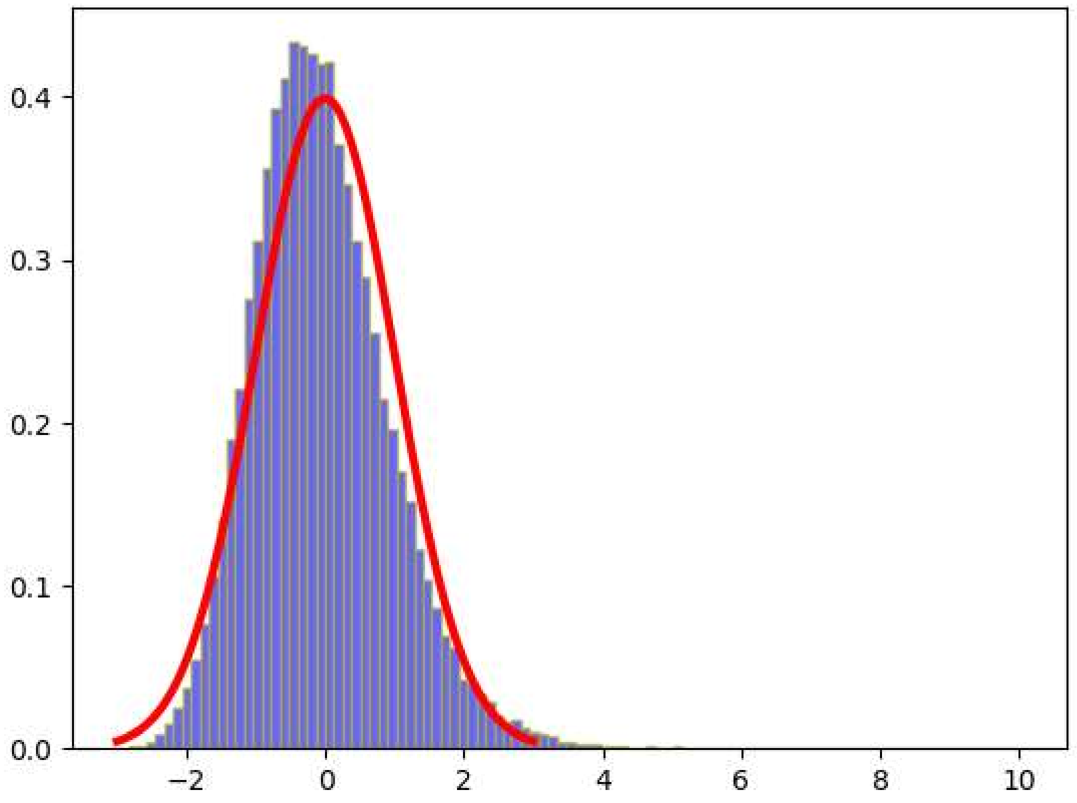
其中 $\hat{k}$ 為峰態係數（Kurtosis）的估計值（參考指令 scipy.stats.kurtosis）。同樣利用蒙地卡羅模擬，驗證統計量 G_2 服從標準常態 $N(0,1)$。蒙地卡羅模擬的環境設定同上。

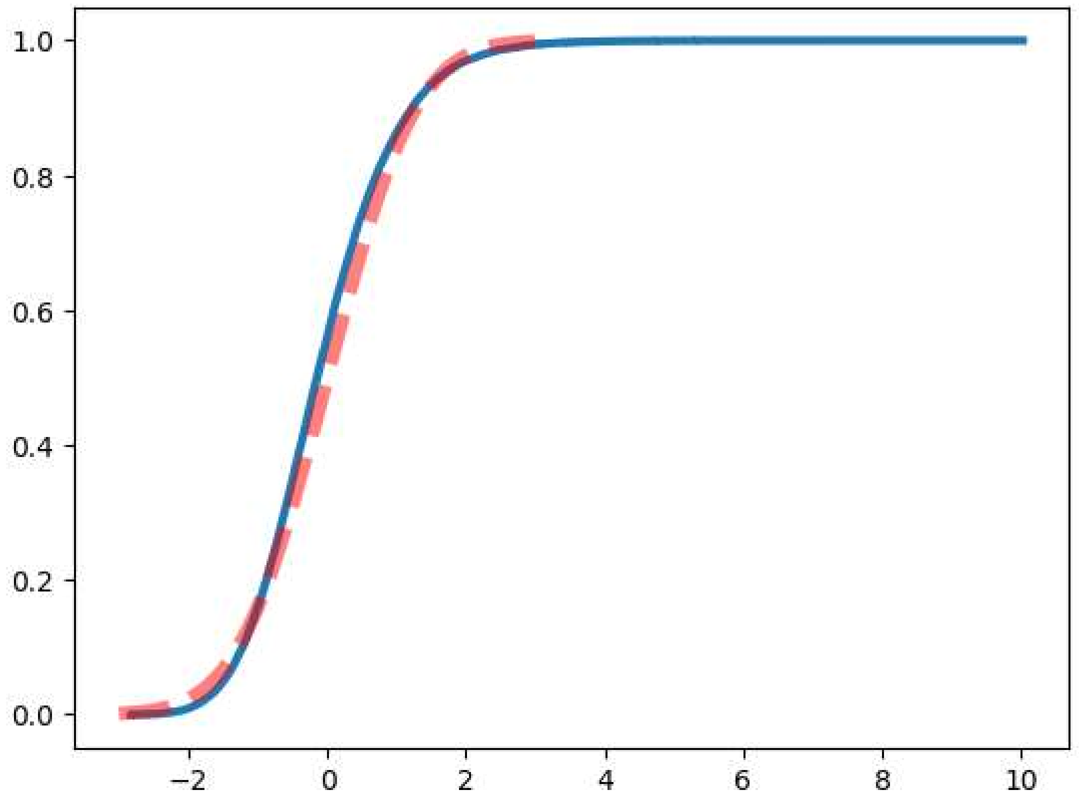
In [ ]:
```python
from scipy.stats import norm, kurtosis
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt

n = 10 #樣本數
N = 50000 #實驗次數

xx = np.linspace(-3, 3, 200)
norm_pdf = norm.pdf(xx)
plt.plot(xx, norm_pdf, lw=3, color='r')

x1 = norm.rvs(loc = 0, scale = 1, size = (n, N))
G2 = np.sqrt(n / 24)*kurtosis(x1)
plt.hist(G2, bins = 100, alpha = 0.6, color = 'b', edgecolor = 'y', density = True)
plt.show()

x_sort = np.sort(G2)
ecdf = np.arange(1, N+1) / N
plt.plot(x_sort, ecdf, drawstyle = 'steps-pre', lw=3)
norm_cdf = norm.cdf(xx)
plt.plot(xx, norm_cdf, linestyle = '--', lw=5, color='r', alpha=0.5)
plt.show()

n1 = 500 #樣本數
N1 = 50000 #實驗次數

xx = np.linspace(-3, 3, 200)
norm_pdf = norm.pdf(xx)
plt.plot(xx, norm_pdf, lw=3, color='r')

x1 = norm.rvs(loc = 0, scale = 1, size = (n1, N1))
G2 = np.sqrt(n1 / 24)*kurtosis(x1)
plt.hist(G2, bins = 100, alpha = 0.6, color = 'b', edgecolor = 'y', density = True)
plt.show()

x_sort1 = np.sort(G2)
ecdf1 = np.arange(1, N1+1) / N1
plt.plot(x_sort1, ecdf1, drawstyle = 'steps-pre', lw=3)
norm_cdf = norm.cdf(xx)
```

```
plt.plot(xx, norm_cdf, linestyle = '--', lw=5, color='r', alpha=0.5)
plt.show()
```




1.0
0.8
0.6
0.4
0.2
0.0
−3
−2
−1
0
1
2
3


0.0
0.1
0.2
0.3
0.4
−2
0
2
4
6
8
10



討論：

- 當樣本數較大時，G_2會愈趨近標準常態 $N(0,1)$。

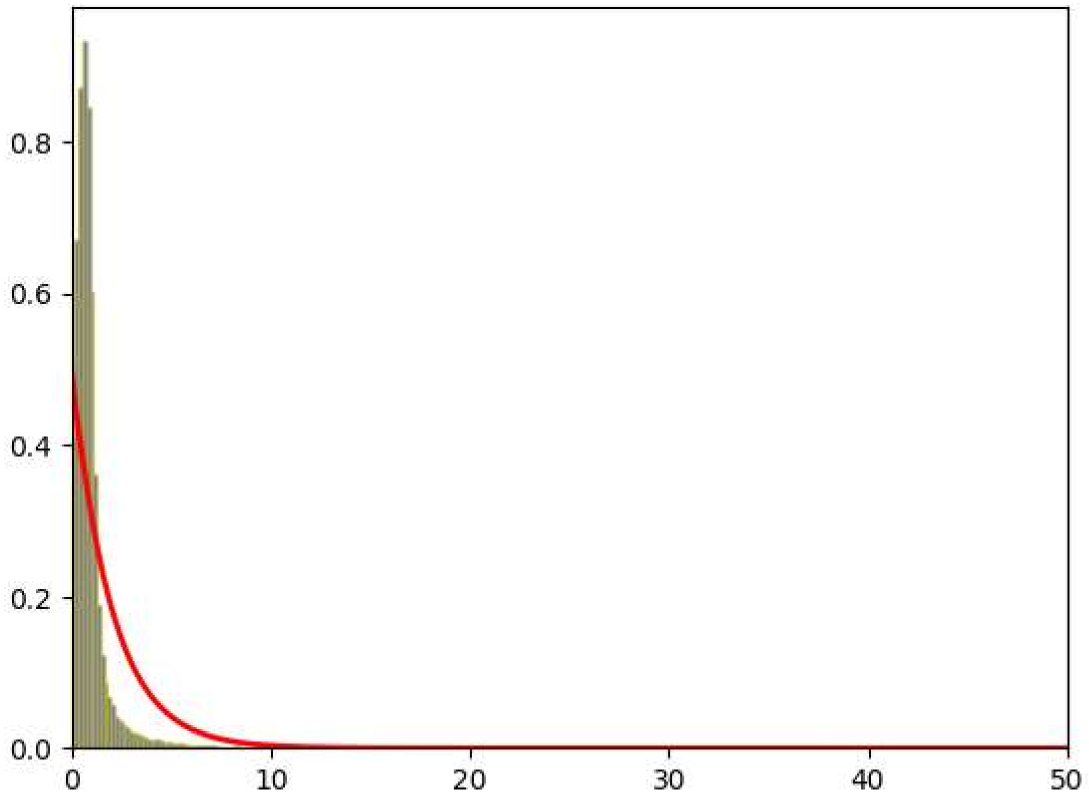
3.

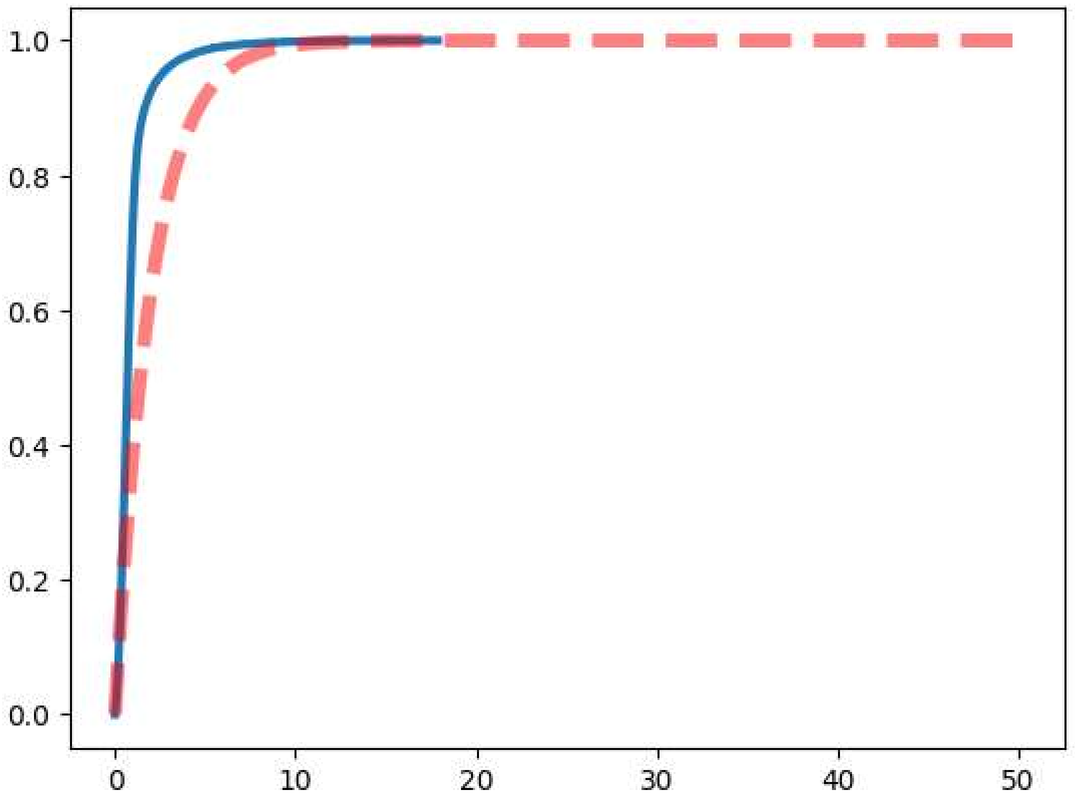
同上，但統計量為

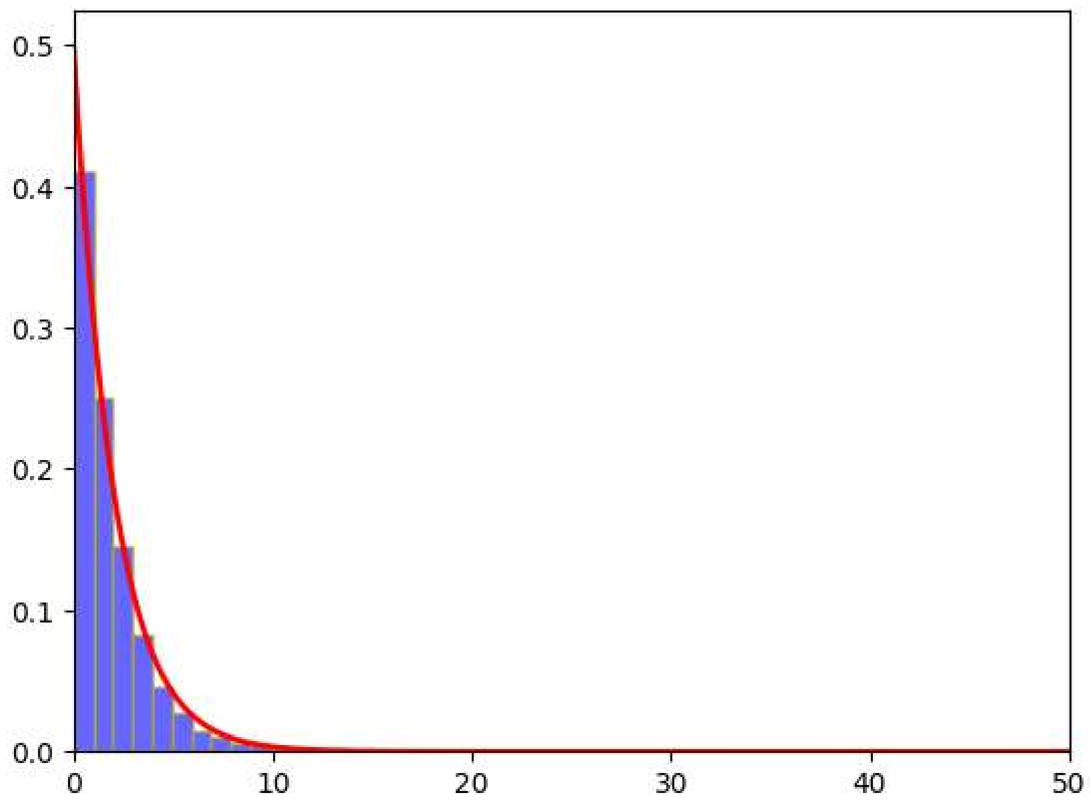
$$G_3 = G_1^2 + G_2^2 = \sqrt{\frac{n}{6}}\left(\hat{s}^2 + \frac{(\hat{k}-3)^2}{4}\right)$$ ，

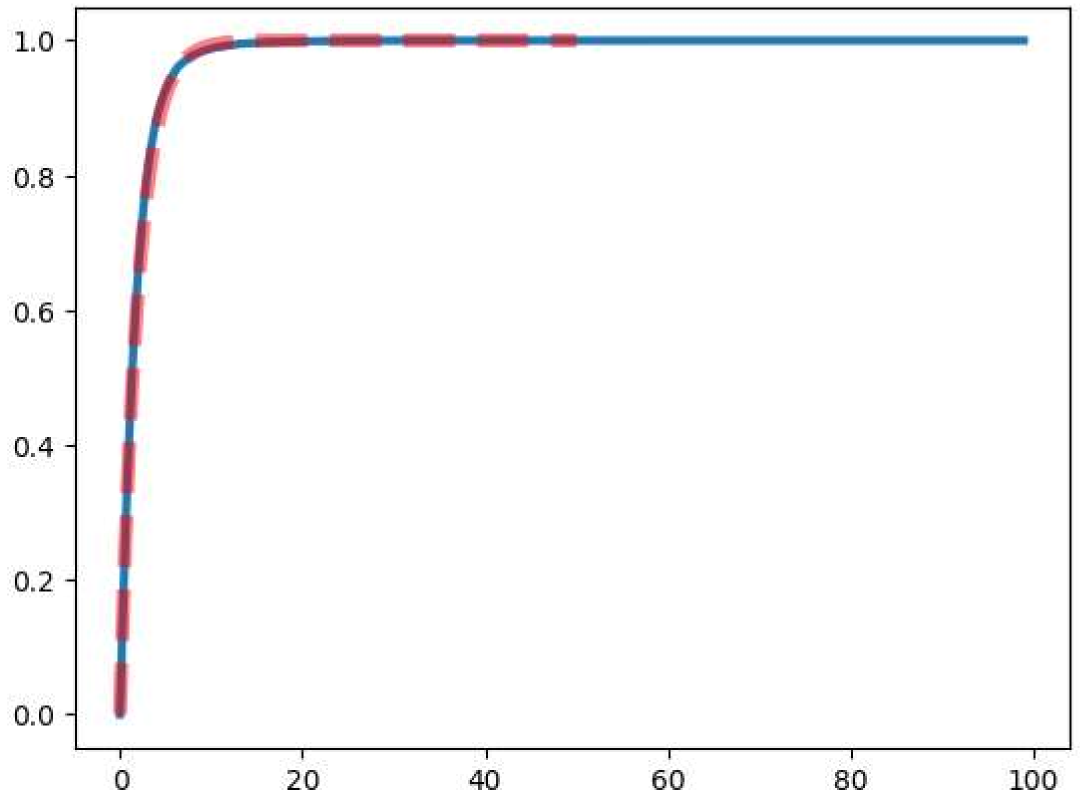
同樣利用上述的蒙地卡羅模擬，驗證統計量 G_3 服從卡方分配 $\chi^2(2)$。G_3 為著名的 J-B (Jarque-Bera) 常態檢定統計量。

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

        n = 10 #樣本數
        N = 50000 #實驗次數

        xlim = [0, 50]
        x = np.linspace(xlim[0], xlim[1], 1000)
        df = 2
        y=chi2.pdf(x, df)
        plt.plot(x,y, lw=2, color='r')

        x1 = norm.rvs(loc = 0, scale = 1, size = (n, N))
        G1 = np.sqrt(n / 6)*skew(x1)
        G2 = np.sqrt(n / 24)*kurtosis(x1)
        G3 = G1 ** 2 + G2 ** 2
        plt.hist(G3, bins = 100, alpha = 0.6, color = 'b', edgecolor = 'y', density = True)
        plt.xlim(xlim[0],xlim[1])
        plt.show()

        x_sort = np.sort(G3)
        ecdf = np.arange(1, N+1) / N
        plt.plot(x_sort, ecdf, drawstyle = 'steps-pre', lw=3)
        chi2_cdf = chi2.cdf(x, df)
        plt.plot(x, chi2_cdf, linestyle = '--', lw=5, color='r', alpha=0.5)
        plt.show()

        n1 = 500 #樣本數
        N1 = 50000 #實驗次數

        xlim1 = [0, 50]
        x1 = np.linspace(xlim1[0], xlim1[1], 1000)
        df = 2
        y1 = chi2.pdf(x1, df)
        plt.plot(x1, y1, lw=2, color='r')

        x1 = norm.rvs(loc = 0, scale = 1, size = (n1, N1))
        G1 = np.sqrt(n1 / 6)*skew(x1)
        G2 = np.sqrt(n1 / 24)*kurtosis(x1)
        G3 = G1**2 + G2**2
        plt.hist(G3, bins = 100, alpha = 0.6, color = 'b', edgecolor = 'y', density = True)
plt.xlim(xlim[0],xlim[1])
plt.show()

x_sort1 = np.sort(G3)
ecdf1 = np.arange(1, N1+1) / N1
plt.plot(x_sort1, ecdf1, drawstyle = 'steps-pre', lw=3)
chi2_cdf1 = chi2.cdf(x, df)
plt.plot(x, chi2_cdf1, linestyle = '--', lw=5, color='r', alpha=0.5)
plt.show()
```




1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50


0.5
0.4
0.3
0.2
0.1
0.0
0
10
20
30
40
50



討論：

- 當樣本數較大時，G_3會愈趨近卡方分配 $\chi2(2)$。

4.

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 [ ]:
```
from scipy.stats import norm, kurtosis, chi2, skew
import numpy as np

def JB_test(x):
```

```python
    n = x.shape[0]
    G1 = np.sqrt(n / 6)*skew(x)
    G2 = np.sqrt(n / 24)*kurtosis(x)
    JB = G1 ** 2 + G2 ** 2
    p_value = 1 - chi2.cdf(JB, df=2)
    return JB, p_value
```

5.

接著檢驗檢定統計量 G_3 的檢定力。採蒙地卡羅模擬方式，步驟如下：

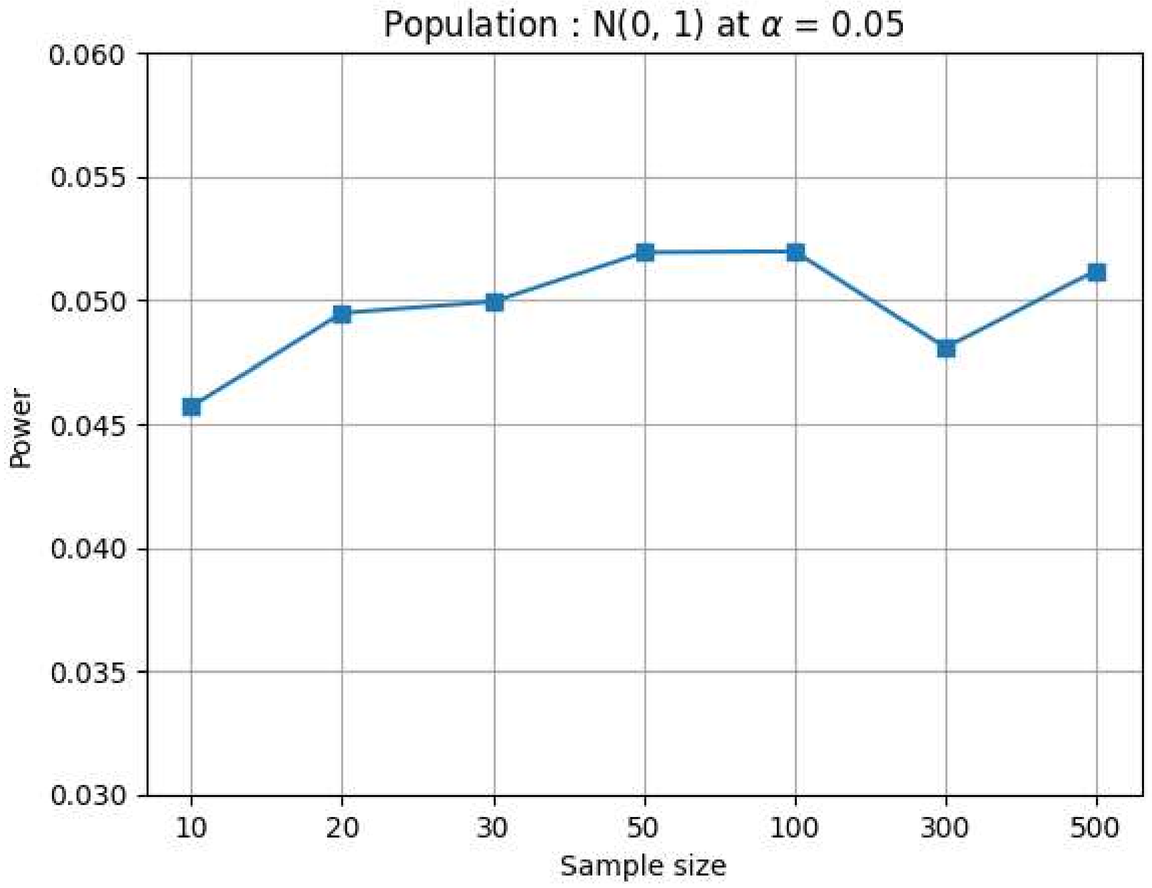
- 從下列的分配母體中抽樣：$N(0,1), T(3), T(10), T(30), U(0,1), \chi^2(8)$。
- 抽樣數 $n = 10, 20, 30, 50, 100, 300, 500$。
- 實驗次數 $N = 50000$。
- 型一誤 $\alpha = 0.05$。
- 對每個分配母體與樣本數，分別計算檢定力： $Power = P(Reject\ H_0 \mid H_a)$，其中 H_0: 資料來自常態；H_a: 資料來自其他分配。 最後針對每個母體，繪製如下圖的 Power vs. sample size。觀察檢定力受樣本數與母體來源（與常態的相似度）的影響。其中 Y 軸必須選擇合適的範圍，方能呈現出清楚的 power 值。
- 其實，當 H_a 來自常態時（也就是資料來自 H_0 的意思），此時的 Power 又稱為顯著水準，且理論上， Power 應該維持在所設定的型一誤 α，即 0.05，如下圖左。因此，當檢定統計量無法維持既定的顯著水準時，後續的檢定力也不用做了。因為檢定統計量是根據 H_0 為真的條件下得到的，連「自己的出身」都維持不住，則檢定力也失去意義。
- 一般而言，檢定力會隨著樣本數增加而變大，亦即，樣本數大有利於辨認資料的「真偽」。

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

n = [10, 20, 30, 50, 100, 300, 500] #樣本數
N = 50000 #實驗次數
alfa = 0.05
power = np.zeros(len(n))

def JB_test(x):
    n = x.shape[0]
    s1 = skew(x,bias=False)
    G1 = np.sqrt(n/6)*s1
    k1 = kurtosis(x,bias = False)
    G2 = np.sqrt(n/24)*(k1)
    G3 = G1**2 + G2**2
    p_value = 1-chi2.cdf(G3,df = 2)
    return G3, p_value

#N(0, 1)
for i in range(len(n)):
    x1 = norm.rvs(loc = 0, scale = 1, size = (n[i], N))
    G3, p_value = JB_test(x1)
    power[i] = (p_value <= alfa).mean()

plt.plot(np.arange(len(n)), power, marker='s')
plt.xticks(range(len(n)), n)
plt.ylim(0.03, 0.06, 0.005)
plt.title("Population : N(0, 1) at $\\alpha$ = 0.05")
plt.xlabel("Sample size")
plt.ylabel("Power")
plt.grid(True)
plt.show()

#T(3)
power1 = np.zeros(len(n))
for i in range(len(n)):
    x2 = t.rvs(df=3, size = (n[i], N))
    stats, p_value = JB_test(x2)
    power1[i] = (p_value <= alfa).mean()

plt.plot(np.arange(len(n)), power1, marker='s')
plt.xticks(range(len(n)), n)
plt.ylim(0, 1.1, 0.2)
plt.title("Population : T(3) at $\\alpha$ = 0.05")
plt.xlabel("Sample size")
plt.ylabel("Power")
plt.grid(True)
plt.show()

#T(10)
power2 = np.zeros(len(n))
for i in range(len(n)):
    x3 = t.rvs(df=10, size = (n[i], N))
    stats, p_value = JB_test(x3)
    power2[i] = (p_value <= alfa).mean()
```

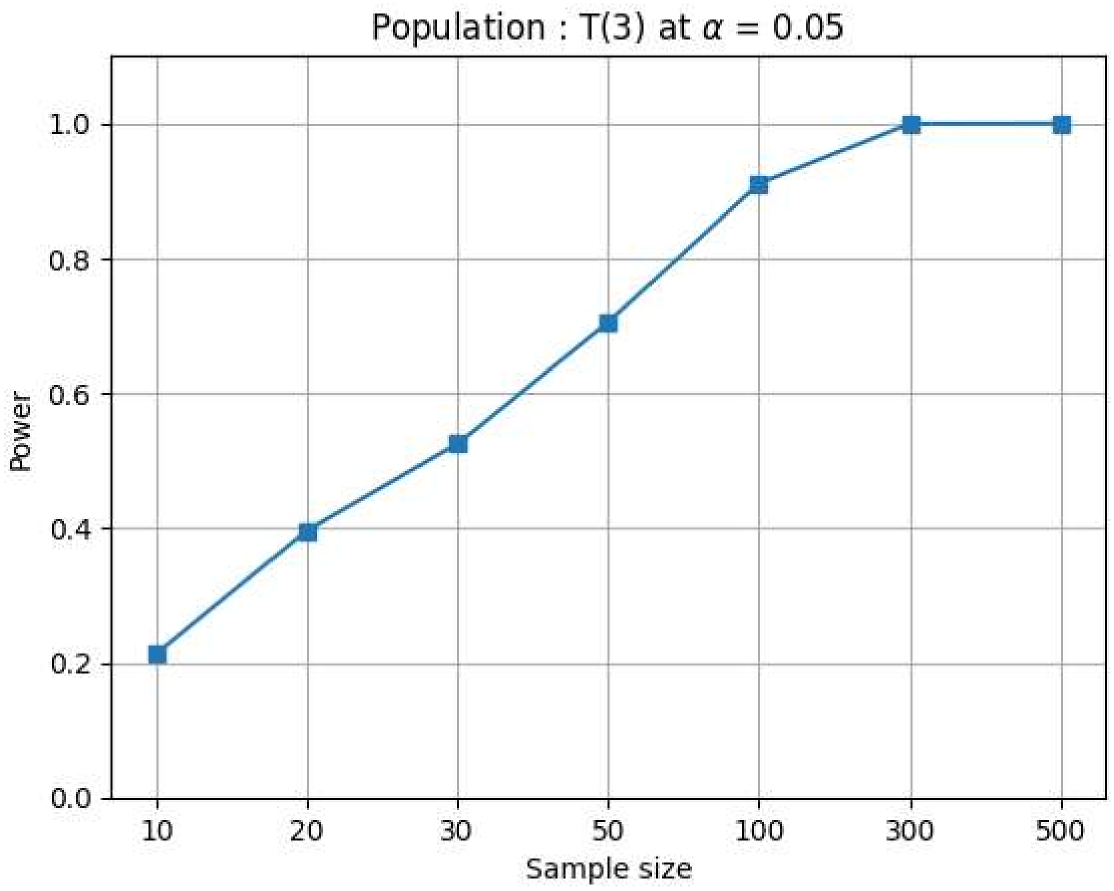
```python
plt.plot(np.arange(len(n)), power2, marker='s')
plt.xticks(range(len(n)), n)
plt.ylim(0, 1.1, 0.2)
plt.title("Population : T(10) at $\\alpha$ = 0.05")
plt.xlabel("Sample size")
plt.ylabel("Power")
plt.grid(True)
plt.show()

#T(30)
power3 = np.zeros(len(n))
for i in range(len(n)):
    x4 = t.rvs(df=30, size = (n[i], N))
    stats, p_value = JB_test(x4)
    power3[i] = (p_value <= alfa).mean()

plt.plot(np.arange(len(n)), power3, marker='s')
plt.xticks(range(len(n)), n)
plt.ylim(0, 1.1, 0.2)
plt.title("Population : T(30) at $\\alpha$ = 0.05")
plt.xlabel("Sample size")
plt.ylabel("Power")
plt.grid(True)
plt.show()

#U(0, 1)
power4 = np.zeros(len(n))
for i in range(len(n)):
    x5 = uniform.rvs(loc = 0, scale = 1, size = (n[i], N))
    stats, p_value = JB_test(x5)
    power4[i] = (p_value <= alfa).mean()

plt.plot(np.arange(len(n)), power4, marker='s')
plt.xticks(range(len(n)), n)
plt.ylim(0, 1.1, 0.2)
plt.title("Population : U(0, 1) at $\\alpha$ = 0.05")
plt.xlabel("Sample size")
plt.ylabel("Power")
plt.grid(True)
plt.show()
```

```python
#chi2(8)
power5 = np.zeros(len(n))
for i in range(len(n)):
    x6 = chi2.rvs(df = 8, size = (n[i], N))
    stats, p_value = JB_test(x6)
    power5[i] = (p_value <= alfa).mean()

plt.plot(np.arange(len(n)), power5, marker='s')
plt.xticks(range(len(n)), n)
plt.ylim(0, 1.1, 0.2)
plt.title("Population : $\chi2(8)$ at $\\alpha$ = 0.05")
plt.xlabel("Sample size")
plt.ylabel("Power")
plt.grid(True)
plt.show()
```

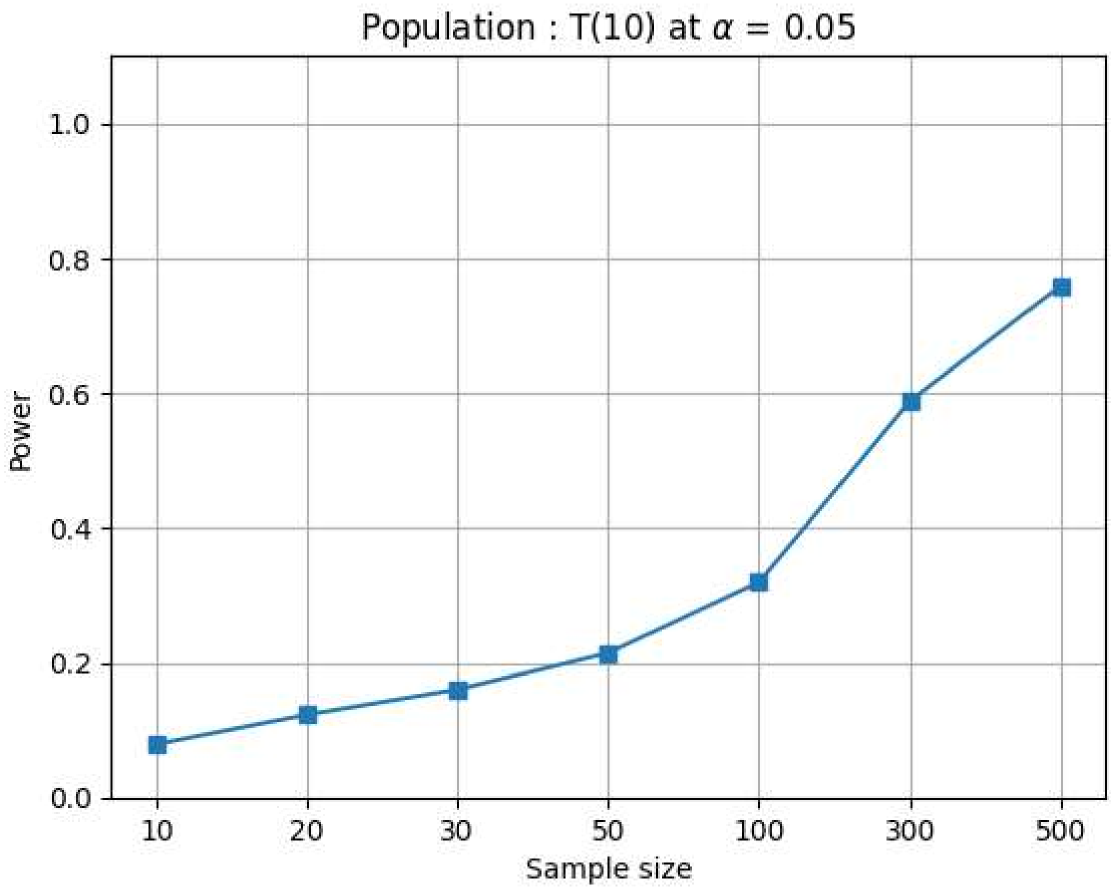
```
C:\Users\guanx\AppData\Local\Temp\ipykernel_8256\1396010389.py:28: MatplotlibDeprecationWarning: Passing the emit parameter of s
et_ylim() positionally is deprecated since Matplotlib 3.6; the parameter will become keyword-only two minor releases later.
  plt.ylim(0.03, 0.06, 0.005)
```



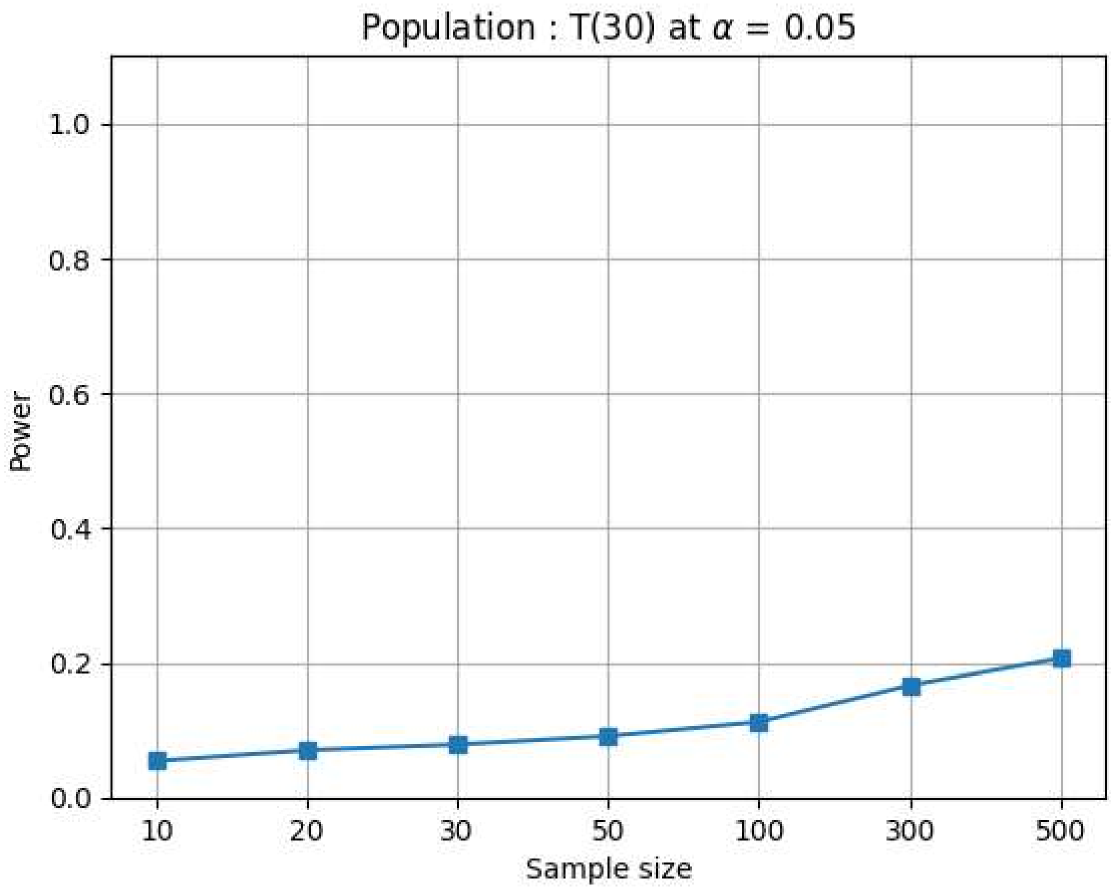
```
C:\Users\guanx\AppData\Local\Temp\ipykernel_8256\1396010389.py:44: MatplotlibDeprecationWarning: Passing the emit parameter of s
et_ylim() positionally is deprecated since Matplotlib 3.6; the parameter will become keyword-only two minor releases later.
  plt.ylim(0, 1.1, 0.2)
```



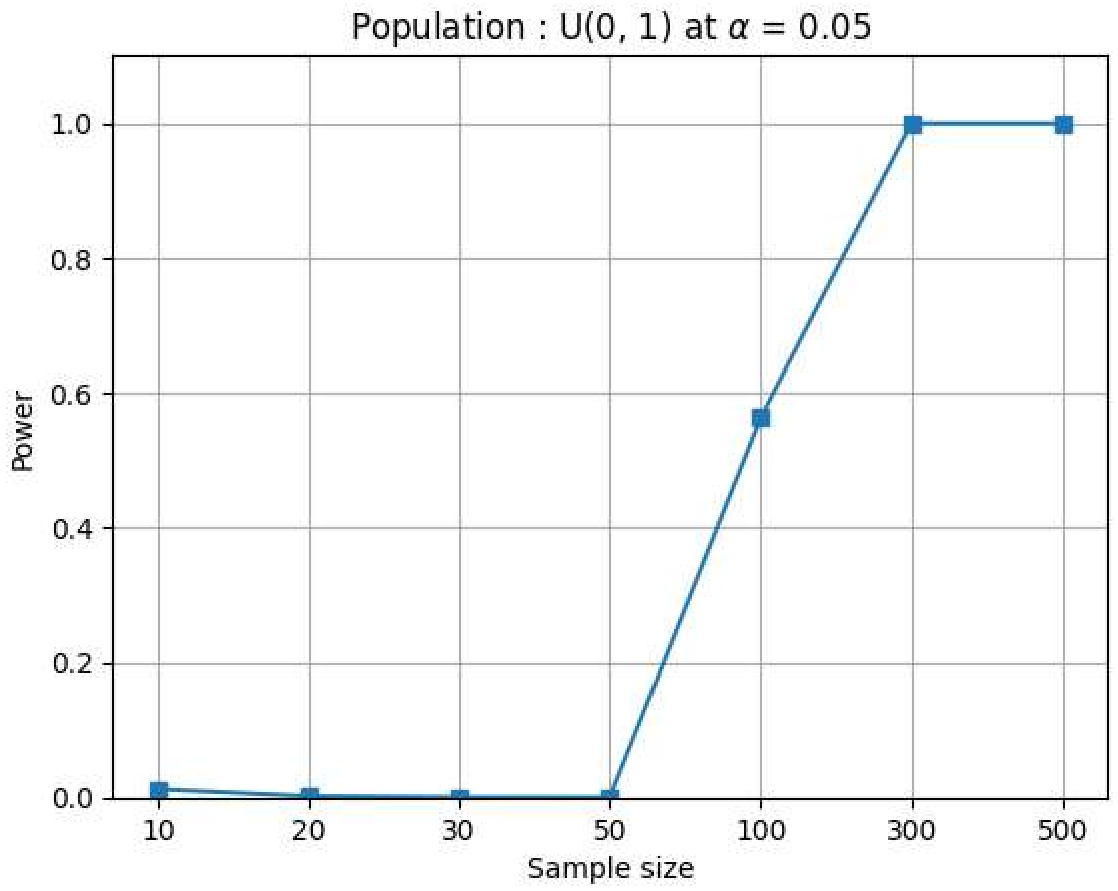
```
C:\Users\guanx\AppData\Local\Temp\ipykernel_8256\1396010389.py:60: MatplotlibDeprecationWarning: Passing the emit parameter of s
et_ylim() positionally is deprecated since Matplotlib 3.6; the parameter will become keyword-only two minor releases later.
  plt.ylim(0, 1.1, 0.2)
```



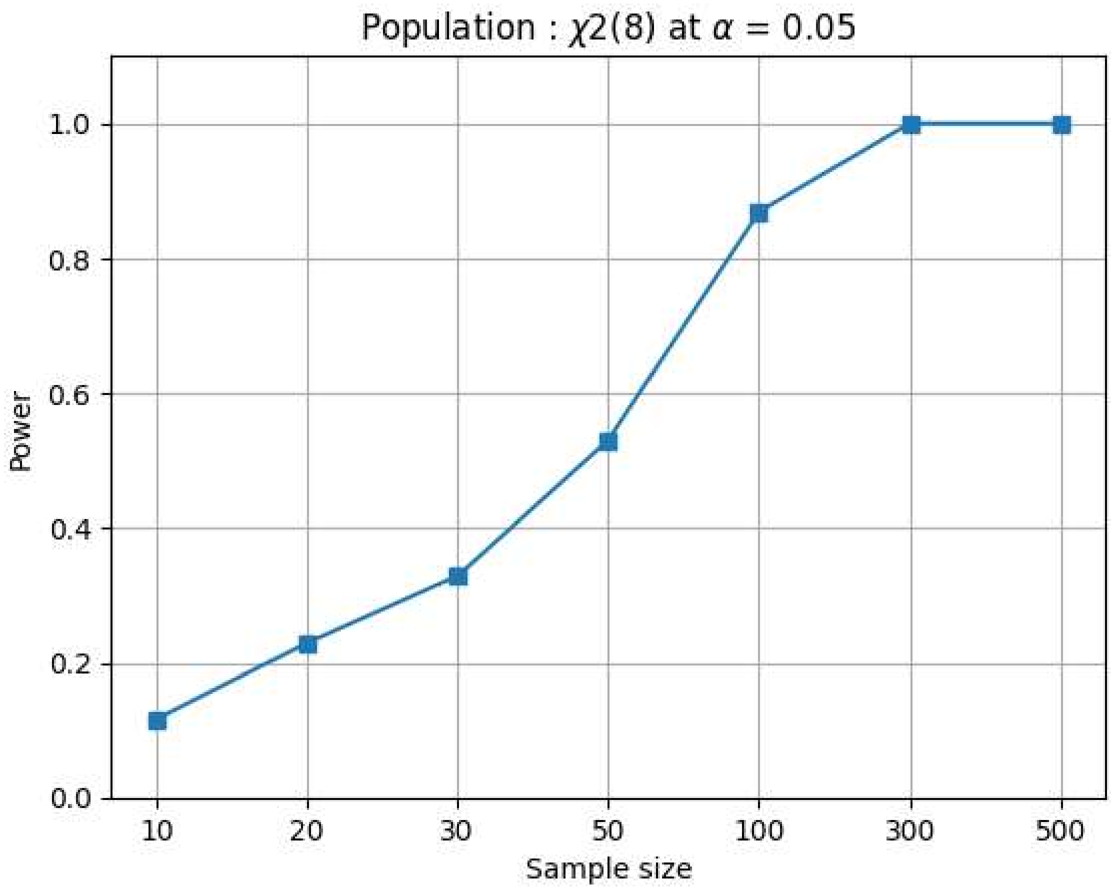
```
C:\Users\guanx\AppData\Local\Temp\ipykernel_8256\1396010389.py:76: MatplotlibDeprecationWarning: Passing the emit parameter of s
et_ylim() positionally is deprecated since Matplotlib 3.6; the parameter will become keyword-only two minor releases later.
  plt.ylim(0, 1.1, 0.2)
```



```
C:\Users\guanx\AppData\Local\Temp\ipykernel_8256\1396010389.py:92: MatplotlibDeprecationWarning: Passing the emit parameter of s
et_ylim() positionally is deprecated since Matplotlib 3.6; the parameter will become keyword-only two minor releases later.
  plt.ylim(0, 1.1, 0.2)
```



```
C:\Users\guanx\AppData\Local\Temp\ipykernel_8256\1396010389.py:108: MatplotlibDeprecationWarning: Passing the emit parameter of
set_ylim() positionally is deprecated since Matplotlib 3.6; the parameter will become keyword-only two minor releases later.
  plt.ylim(0, 1.1, 0.2)
```



討論：

- (1)當樣本來自於常態分配時，無論樣本的大小，檢定力α會大致趨近於0.05。
- (2)T分配中，可以觀察到，當自由度較小時，樣本數的增加能使檢定力有明顯的提升。當自由度逐漸增加時，樣本數的增加雖然能使檢定力有所提升，但效果很有限。
- (3)Uniform分配中，當樣本數小於50時，檢定力趨近於0;當樣本數逐漸增加至100時，檢定力大幅提升;當樣本數增加至300時，檢定力趨近於1。
- (4)卡方分配中，檢定力隨著樣本數的增加而增加。
