作品五、 單變量函數的根與最小值

姓名：黃冠翔

學號：410978040

目標：

- 以數值與符號的方式計算單變量函數的根 $f(x) = 0$。
- 以數值與符號的方式計算單變量函數的最小值 $min_x f(x)$。
- 熟悉在函數圖形中標示根或極值位置。

---

1. 畫出目標函數的圖形並標示出關鍵值的位置（最小值、最大值或實數根）

$$\min_x \sqrt{\frac{x^2+1}{x+1}}$$

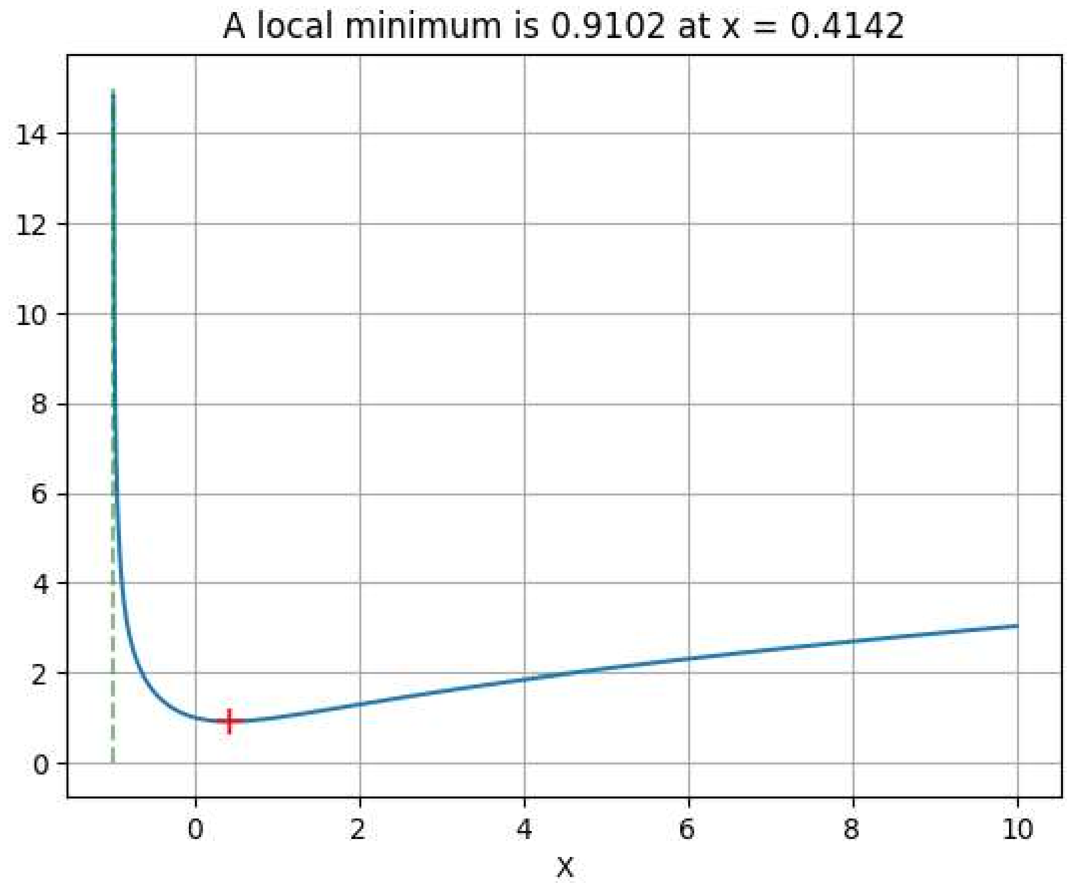
注意事項：

- 畫出目標函數的圖形。
- 使用單行函數設定的 lambda。
- 使用 scipy.optimize.minimize_scalar 計算函數的區域最小值（Local minimum）。
- 指令 scipy.optimize.minimize_scalar 一次只能計算一個區域最小值，決定區域的參數是 bracket。
- 本題採 bracket = $[0.1, 0.2, 1]$, 意思是 $f(0.2) < f(0.1)$ 及 $f(0.2) < f(1)$。
- 使用 plt.text 標示區域最小值（Local minimum）的位置。

In [ ]:
```
import numpy as np
import scipy.optimize as opt
import matplotlib.pyplot as plt

f = lambda x : np.sqrt((x**2 + 1)/(x + 1))
x = np.linspace(-2, 10, 1000)
plt.plot(x, f(x))
plt.xlabel('X')
plt.grid(True)
plt.vlines(-1, 0, 15, color = 'g', linestyle='--', alpha = 0.5)
```

```python
res = opt.minimize_scalar(f, bracket=[0.1, 0.2, 1])
#print(res) # find out the return values
print('The function has a local minimum at x = {:.4f}'.format(res.x))
print('The corresponding function value is {:.4f}'.format(res.fun))

plt.text(res.x, res.fun, '+', color = 'r', fontsize = 16,
    horizontalalignment='center',
    verticalalignment='center')
plt.title('A local minimum is 0.9102 at x = 0.4142')
plt.show()
```

```
C:\Users\guanx\AppData\Local\Temp\ipykernel_4156\3238345579.py:5: RuntimeWarning: invalid value encountered in sqrt
  f = lambda x : np.sqrt((x**2 + 1)/(x + 1))
The function has a local minimum at x = 0.4142
The corresponding function value is 0.9102
```



討論：描述上述程式與執行結果，有甚麼值得說明或強調的。

- 目標函數在 $x = 0.4142$ 時會有區域最小值（Local minimum）為 0.9102。
- 目標函數有漸進線為 $x = -1$。

---

2. 畫出目標函數的圖形並標示出關鍵值的位置（最小值、最大值或實數根）

$$\underset{-4\le x\le 3}{min}\ (x+1)^5 \sin(x-3)$$

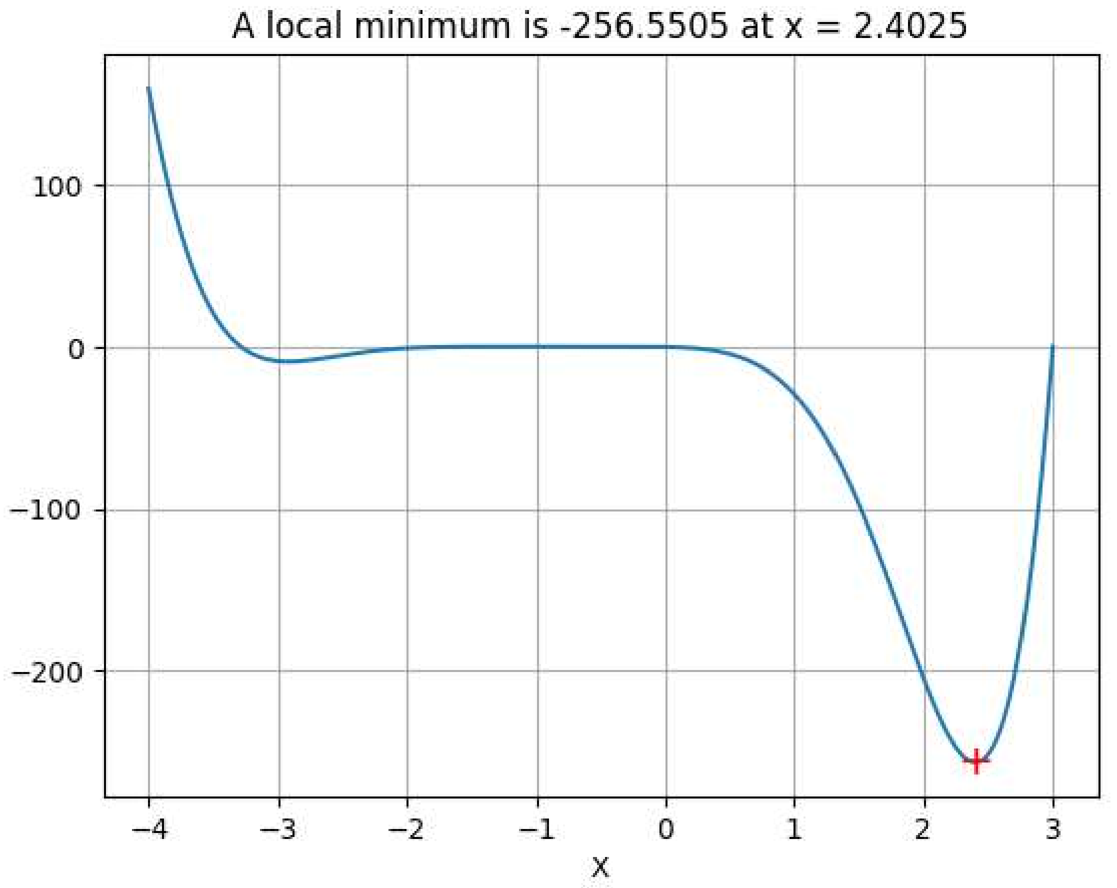
注意事項：

- 畫出目標函數的圖形。
- 使用單行函數設定的 lambda。
- 使用 scipy.optimize.minimize_scalar 計算函數的區域最小值（Local minimum）。
- 指令 scipy.optimize.minimize_scalar 一次只能計算一個區域最小值，決定區域的參數是 bracket。
- 本題採 bracket = $[0, 2, 3]$, 意思是 $f(2) < f(0)$ 及 $f(2) < f(3)$。
- 使用 plt.text 標示區域最小值（Local minimum）的位置。

In [ ]:

```python
import numpy as np
import scipy.optimize as opt
import matplotlib.pyplot as plt

f = lambda x : (x + 1)**5*np.sin(x - 3)
x = np.linspace(-4, 3, 1000)
plt.plot(x, f(x))
plt.xlabel('X')
plt.grid(True)

res = opt.minimize_scalar(f, bracket=[0, 2, 3])
#print(res) # find out the return values
print('The function has a local minimum at x = {:.4f}'.format(res.x))
print('The corresponding function value is {:.4f}'.format(res.fun))

plt.text(res.x, res.fun, '+', color = 'r', fontsize = 16,
    horizontalalignment='center',
    verticalalignment='center')
plt.title('A local minimum is -256.5505 at x = 2.4025')
plt.show()
```

```
The function has a local minimum at x = 2.4025
The corresponding function value is -256.5505
```



討論：描述上述程式與執行結果，有甚麼值得說明或強調的。

- 目標函數在 $x = 2.4025$ 時會有區域最小值（Local minimum）為 -256.5505。

---

3. 畫出目標函數的圖形並標示出關鍵值的位置（最小值、最大值或實數根）

計算 $L(x) = 10$ 的解 x, 其中

$L(x) = \int_a^x \sqrt{1 + (f'(t))}dt$, for $f(t) = t^2/2$ and $a = 0$.

注意事項：

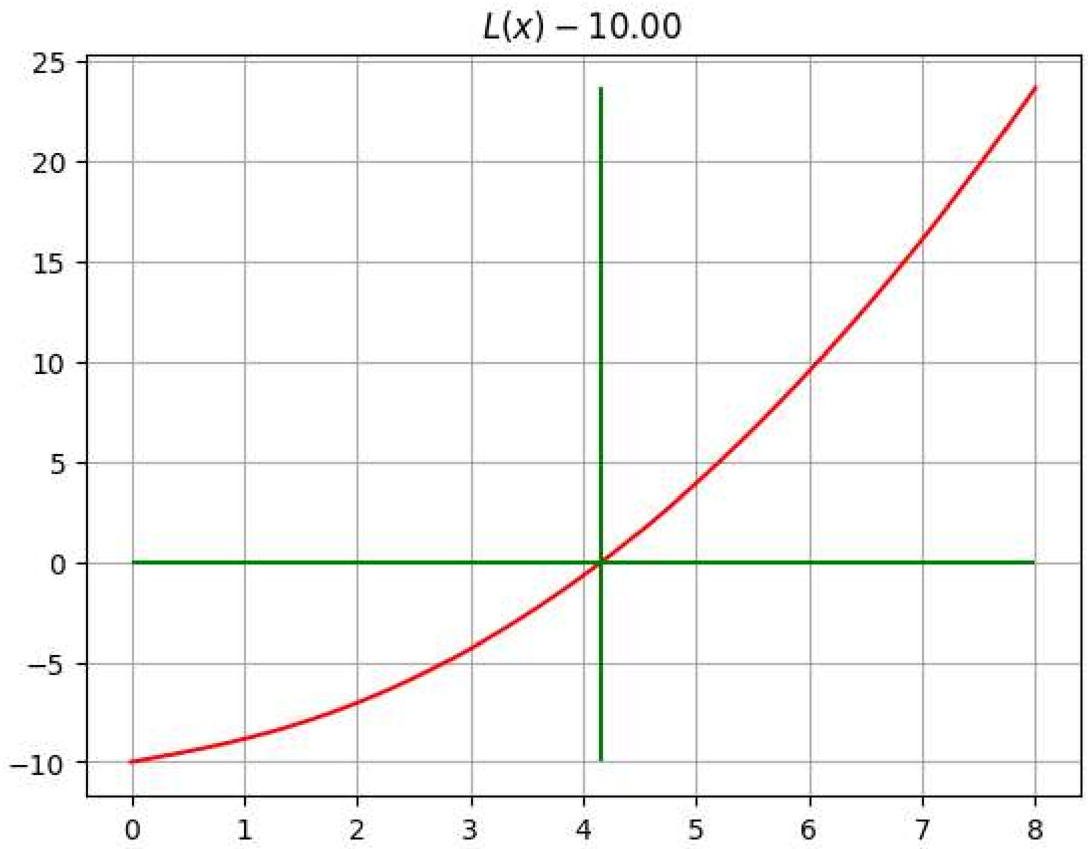
- 使用 sympy 套件計算符號微分 symbolic differentiation。
- 當使用 sympy 套件最符號運算時，必須使用專屬的數學函數，譬如，sympy.sqrt()，不再使用原來的np.sqrt()。
- 使用 sym.lambdify 將符號函數轉為數值型的 lambda 函數。
- 本題的函數較為複雜，因此採 def 函數設定方式。
- 這個問題可以改寫為較明顯的方式： $f(x) = 0$, where $f(x) = F(x) - 10$。
- 在計算複雜函數的根之前，可以試著畫出函數來觀察根的大略位置，以便與計算出來的根做比對。這裡採用向量函數的方式來計算帶著積分的函數。
- 使用 sicpy.optimize.root_scalre 時，必須加入 args 參數，作為函數額外的常數資料，在此就是 prob=10。
- 使用 plt.hlines 和 plt.vlines 繪製水平和垂直線段，兩線段交點即為根。
- 第二張圖畫出 $F(x)$ 的意義，使用 plt.fill_between 繪製出積分結果(面積)，並使用 plt.text 標示面積大小。

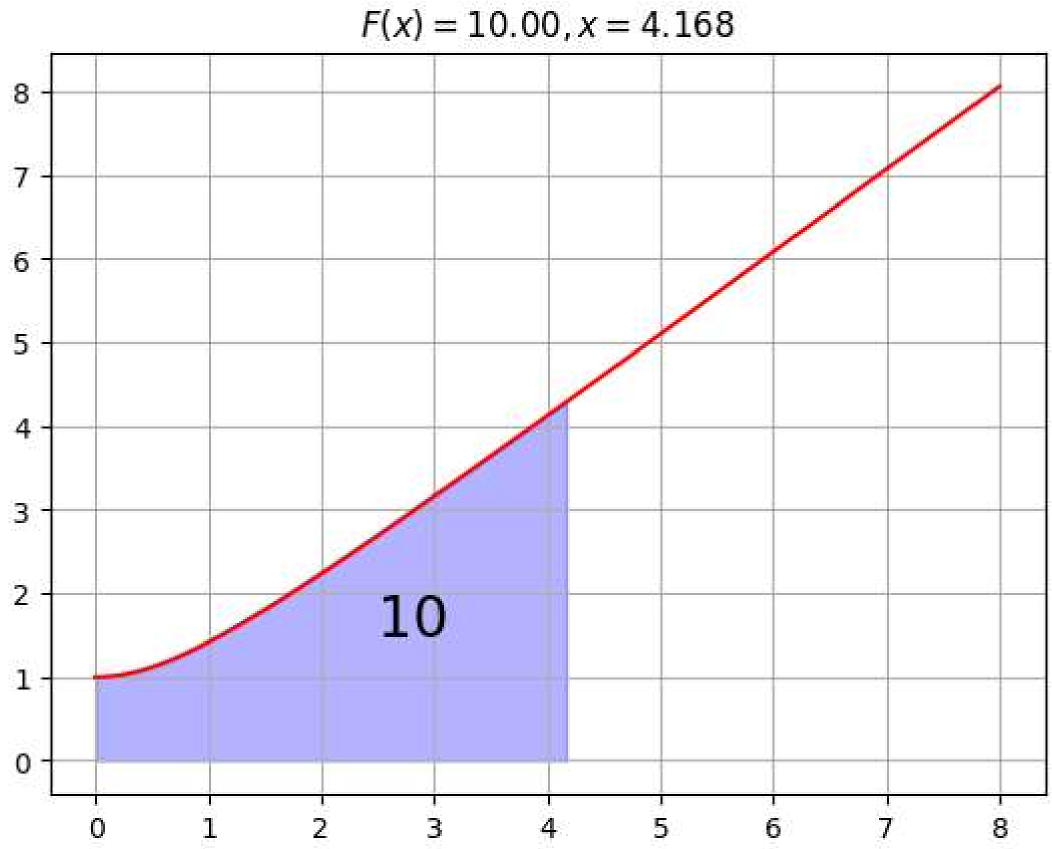
In [ ]:

```python
import numpy as np
import sympy as sym
import scipy.optimize as opt
import matplotlib.pyplot as plt
import scipy.integrate as integral

t = sym.Symbol('t')
f_t = (t**2)/2
f_prime = f_t.diff(t)
feval = sym.lambdify(t, sym.sqrt(1+(f_prime**2)))

def f(x, prob):
    L = integral.quad(feval, a, x)[0]
    return L - prob

n = 200
a = 0
prob = 10
x = np.linspace(0, 8, n)
t = np.linspace(0, 5, n)
vec_f = np.vectorize(f)
L = vec_f(x, prob)

sol = opt.root_scalar(f, args = prob, bracket = [0, 5])
print('The root is x = {:.3f}'.format(sol.root))

plt.plot(x, L, c='r')
```

```
plt.grid(True)
plt.hlines(0, x.min(), x.max(), color = 'g')
plt.vlines(sol.root, L.min(), L.max(), color = 'g')
plt.title('$L(x) - {:.2f}$'.format(prob))
plt.show()

plt.plot(x, feval(x), c='r')
plt.grid(True)
x_area = np.linspace(x.min(), sol.root, 100)
y_area = feval(x_area)
plt.fill_between(x_area, y_area, 0, color = 'b', alpha = 0.3)
plt.text(2.5, 1.5, prob, fontsize = 20)
plt.title('$F(x)={:.2f}, x = {:.3f}$'.format(prob, sol.root))
plt.show()
```

```
The root is x = 4.168
```





討論：描述上述程式與執行結果，有甚麼值得說明或強調的。

- 第一張圖標示目標函數的根 $x = 4.168$。
- 第二張圖繪製積分結果(面積)為10。

---

4. 最大概似函數估計（**MLE**）：

計算$\underset{\lambda}{max} \ln \prod_{i=1}^{N} f(x_i;\lambda)$,

其中 $f(x_i;\lambda)$ 代表指數分配（參數 λ）的概似函數，即 $f(x_i;\lambda) = \lambda e^{-\lambda x_i}$。 令樣本數 $N= 10, 20 ,30, 50, 100, 300, 500，**分別生成樣本**x_i**（令真實**\lambda = 2**，或自己設定）**，\$ 並採最大概似估計法(log MLE)估計 λ。

注意事項：

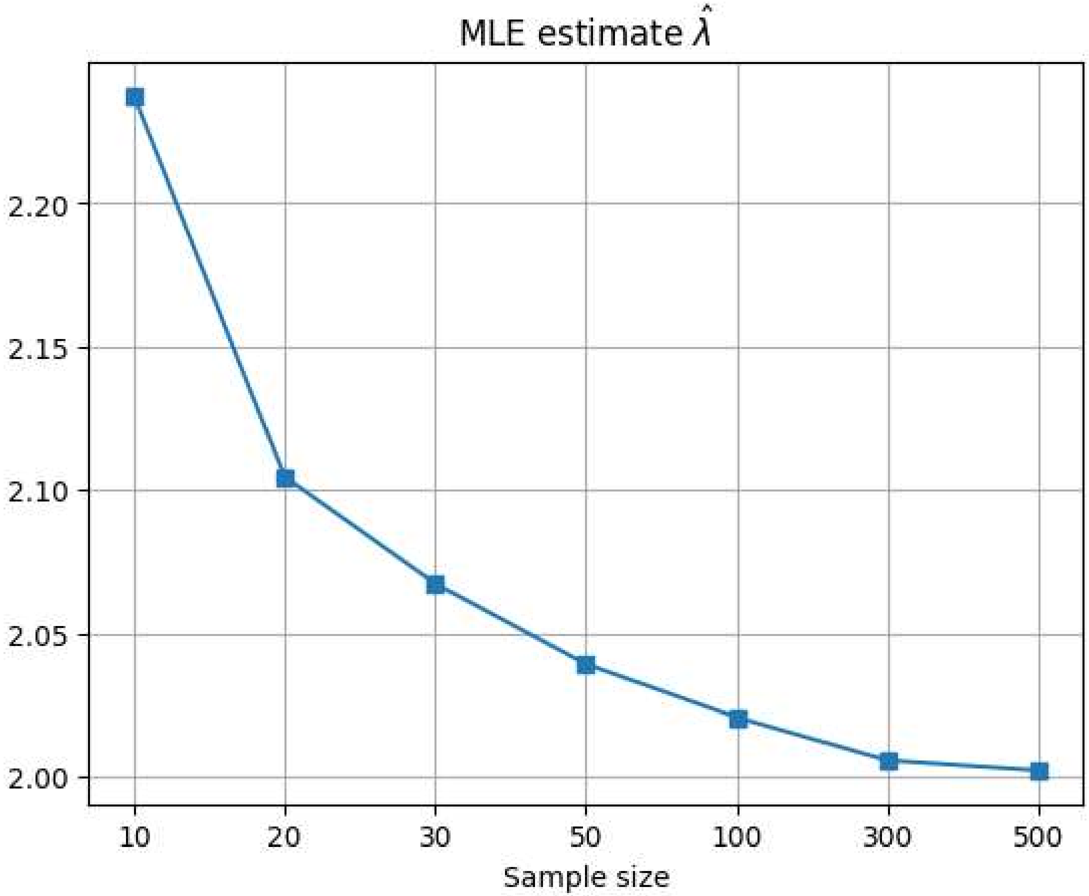
- 任取一組樣本，繪製目標函數 $\ln\prod_{i=1}^{N} f(x_i;\lambda)$，並標示出最大值的位置。
- 畫一張折線圖，呈現每個樣本數的 λ 估計值。為得到在每一個樣本數下較穩定的估計值，對每個樣本數皆執行 $10,000$ 次，最後取其平均數作為估計值，如第一張圖。也因為執行了 $10,000$ 次，因此可以得出估計值的標準差，同樣也為每個樣本數的標準差畫一張折線圖，如第二張圖。

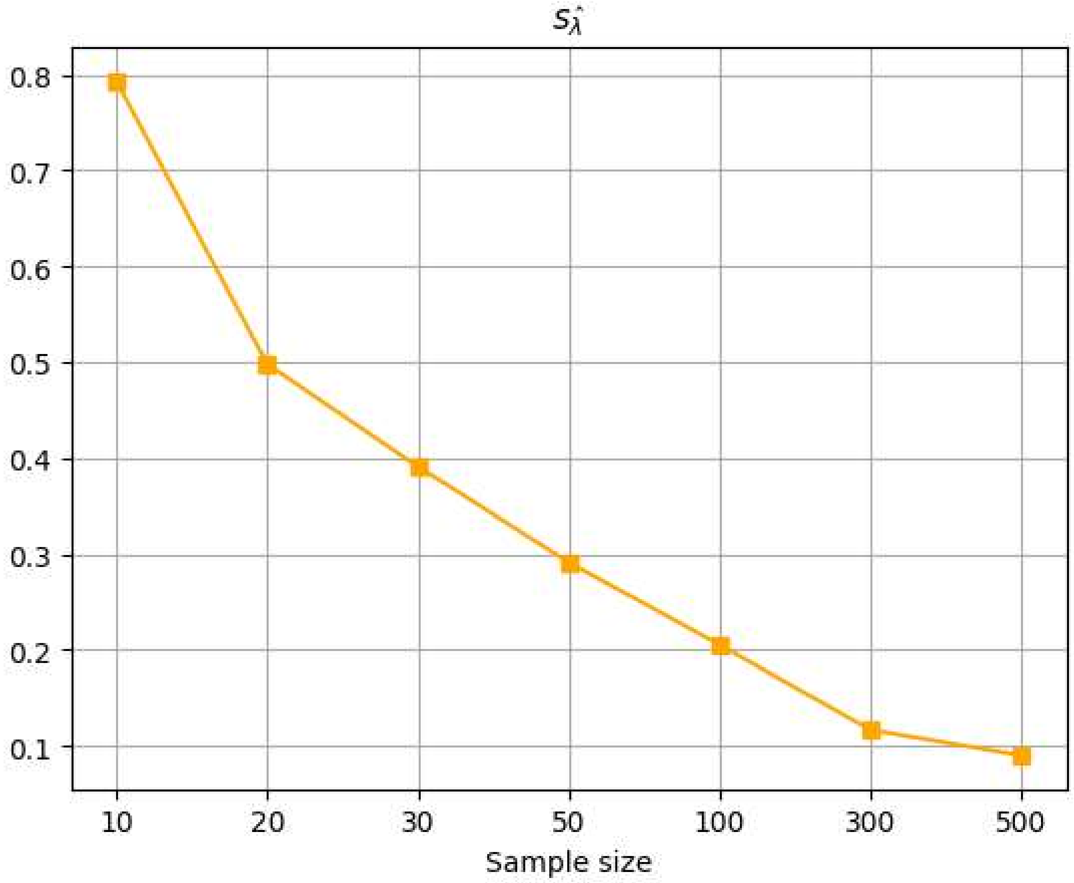
In [ ]:

```
import numpy as np
import scipy.optimize as opt
import matplotlib.pyplot as plt
from scipy.stats import expon

N = [10, 20, 30, 50, 100, 300, 500] #樣本數
T = 10000 #實驗次數
t_lambda = 2
e_lambda = np.zeros(len(N))
e_lambda_std = np.zeros(len(N))
e = np.zeros(T)

for i in range(len(N)):
    n = N[i]
    x = expon.rvs(loc = 0, scale = 1 / t_lambda, size = (T, n))
    for j in range(T):
        x1 = x[j, :]
        f = lambda y : -(n*np.log(y) - y*np.sum(x1))
        res = opt.minimize_scalar(f)
        e[j] = res.x
    e_lambda[i] = e.mean()
    e_lambda_std[i] = e.std()

plt.plot(e_lambda, marker = 's')
plt.xticks(np.arange(len(N)), labels=N)
plt.xlabel('Sample size')
plt.grid(True)
plt.title('MLE estimate $\hat{\lambda}$')
plt.show()

plt.plot(e_lambda_std, marker = 's', c = 'orange')
plt.xticks(np.arange(len(N)), labels=N)
plt.xlabel('Sample size')
```

```
plt.grid(True)
plt.title('$s_\hat{\lambda}$')
plt.show()
```

```
C:\Users\guanx\AppData\Local\Temp\ipykernel_7388\394935140.py:18: RuntimeWarning: divide by zero encountered in log
  f = lambda y : -(n*np.log(y) - y*np.sum(x1))
d:\vs\venv_name\lib\site-packages\scipy\optimize\_optimize.py:2769: RuntimeWarning: invalid value encountered in double_scalars
  w = xb - ((xb - xc) * tmp2 - (xb - xa) * tmp1) / denom
```





討論：描述上述程式與執行結果，有甚麼值得說明或強調的。

- 第一張圖顯示當樣本數增加，λ平均的估計值會愈接近真實的$\lambda = 2$。
- 第二張圖顯示當樣本數增加，λ估計值的標準差會逐漸下降。
