exercise2 (Score: 20.0 / 22.0)

1. Task (Score: 2.0 / 2.0)
2. Test cell (Score: 3.0 / 3.0)
3. Task (Score: 3.0 / 3.0)
4. Test cell (Score: 2.0 / 2.0)
5. Test cell (Score: 2.0 / 2.0)
6. Test cell (Score: 1.0 / 3.0)
7. Test cell (Score: 3.0 / 3.0)
8. Task (Score: 4.0 / 4.0)

Lab 4

1. 提交作業之前，建議可以先點選上方工具列的**Kernel**，再選擇**Restart & Run All**，檢查一下是否程式跑起來都沒有問題，最後記得儲存。
2. 請先填上下方的姓名(name)及學號(stduent_id)再開始作答，例如：

```
name = "我的名字"
student_id= "B06201000"
```

3. 演算法的實作可以參考[lab-4 (https://yuanyuyuan.github.io/itcm/lab-4.html)](https://yuanyuyuan.github.io/itcm/lab-4.html), 有任何問題歡迎找助教詢問。
4. **Deadline: 11/20(Wed.)**

In [1]:

```
name = "李澤諺"
student_id = "B05902023"
```

Exercise 2

Let $I(f)$ be a define integral defined by

$$I(f) = \int_0^1 f(x)dx,$$

and consider the quadrature formula

$$\hat{I}(f) = \alpha_1 f(0) + \alpha_2 f(1) + \alpha_3 f'(0) \qquad (*)$$

for approximation of $I(f)$.

Part 1.

Determine the coefficients α_j for $j = 1, 2, 3$ in such a way that $\hat{I}$ has the degree of exactness $r = 2$. Here the degree of exactness r is to find r such that

$$\hat{I}(x^k) = I(x^k) \quad \text{for} \quad k = 0, 1, \ldots, r \quad \text{and} \quad \hat{I}(x^j) \neq I(x^j) \quad \text{for} \quad j > r,$$

where x^j denote the j-th power of x.

(Top)

Derive the values of $\alpha_1, \alpha_2, \alpha_3$ in $(*)$. You need to write down the detail in the cell below with Markdown/LaTeX.

若degree of exactness為2，則必須有

$$\hat{I}(1) = I(1)$$
$$\hat{I}(x) = I(x)$$
$$\hat{I}(x^2) = I(x^2)$$

即

$$\alpha_1 \cdot 1|_{x=0} + \alpha_2 \cdot 1|_{x=1} + \alpha_3 \cdot \frac{d}{dx}1|_{x=0} = \int_0^1 1dx$$
$$\alpha_1 \cdot x|_{x=0} + \alpha_2 \cdot x|_{x=1} + \alpha_3 \cdot \frac{d}{dx}x|_{x=0} = \int_0^1 xdx$$
$$\alpha_1 \cdot x^2|_{x=0} + \alpha_2 \cdot x^2|_{x=1} + \alpha_3 \cdot \frac{d}{dx}x^2|_{x=0} = \int_0^1 x^2dx$$

因此可得

$$\alpha_1 + \alpha_2 = 1$$
$$\alpha_2 + \alpha_3 = \frac{1}{2}$$
$$\alpha_2 = \frac{1}{3}$$

所以

$$\alpha_1 = \frac{2}{3}\ ,\ \alpha_2 = \frac{1}{3}\ ,\ \alpha_3 = \frac{1}{6}$$

Fill in the tuple variable `alpha_1` , `alpha_2` , `alpha_3` with your answer above.

In [2]:

(Top)

```
alpha_1 = 2 / 3
alpha_2 = 1 / 3
alpha_3 = 1 / 6
```

In [3]:

part_1 (Top)

```python
print("alpha_1 =", alpha_1)
print("alpha_2 =", alpha_2)
print("alpha_3 =", alpha_3)
### BEGIN HIDDEN TESTS
assert abs(alpha_1 - 2/3) <= 1e-7, 'alpha_1 is wrong!'
assert abs(alpha_2 - 1/3) <= 1e-7, 'alpha_2 is wrong!'
assert abs(alpha_3 - 1/6) <= 1e-7, 'alpha_3 is wrong!'
### END HIDDEN TESTS
```

```
alpha_1 = 0.6666666666666666
alpha_2 = 0.3333333333333333
alpha_3 = 0.16666666666666666
```

(Top)

Part 2.

Find an apppropriate expression for the error $E(f) = I(f) - \hat{I}(f)$, and write your process in the below cell with Markdown/LaTeX.

利用Peano kernel theorem

$$E(f) = \frac{1}{2}\int_0^1 f^{(3)}(t)K(t)dt$$

其中，由mean value theorem for integrals，可知 $\exists\, \xi \in [0, 1]$，使得

$$E(f) = \frac{1}{2}f^{(3)}(\xi)\int_0^1 K(t)dt$$

此外，當 $t \in [0, 1]$ 時

$$\begin{aligned}
K(t) &= E((x-t)_+^2) \\
&= I((x-t)_+^2) - \hat{I}((x-t)_+^2) \\
&= \int_0^1 (x-t)_+^2 dx - (\frac{2}{3}\cdot(x-t)_+^2|_{x=0} + \frac{1}{3}\cdot(x-t)_+^2|_{x=1} + \frac{1}{6}\cdot\frac{d}{dx}(x-t)_+^2|_{x=0}) \\
&= \int_t^1 (x-t)^2 dx - (\frac{2}{3}\cdot 0|_{x=0} + \frac{1}{3}\cdot(x-t)^2|_{x=1} + \frac{1}{6}\cdot\frac{d}{dx}0|_{x=0}) \\
&= \frac{1}{3}(1-t)^3 - (\frac{2}{3}\cdot 0 + \frac{1}{3}\cdot(1-t)^2 + \frac{1}{6}\cdot 0) \\
&= -\frac{t}{3}(1-t)^2
\end{aligned}$$

因此可得

$$\begin{aligned}
E(f) &= \frac{1}{2}f^{(3)}(\xi)\int_0^1 -\frac{t}{3}(1-t)^2 dt \\
&= \frac{1}{2}f^{(3)}(\xi)\cdot(-\frac{1}{36}) \\
&= -\frac{1}{72}f^{(3)}(\xi)
\end{aligned}$$

Part 3.

Compute

$$\int_0^1 e^{-\frac{x^2}{2}} dx$$

using quadrature formulas $(*)$, the Simpson's rule and the Gauss-Legendre formula in the case $n = 1$. Compare the obtained results.

Part 3.1

Import necessary libraries

In [4]:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.special.orthogonal import p_roots
```

Part 3.2

Define the function $f(x) = e^{-\frac{x^2}{2}}$ and its derivative.

In [5]:

(Top)

```
def f(x):
    return np.exp(-x**2 / 2)

def d_f(x):
    return -x * np.exp(-x**2 / 2)
```

Print and check your functions.

In [6]:

part_3_1_1 (Top)

```
print('f(0) =', f(0))
print("f'(0) =", d_f(0))
### BEGIN HIDDEN TESTS
assert abs(f(5) - np.exp(-5**2/2)) <= 1e-7, 'f(5) is wrong!'
assert abs(f(10) - np.exp(-10**2/2)) <= 1e-7, 'f(10) is wrong!'
assert abs(d_f(5) - -5*np.exp(-5**2/2)) <= 1e-7, "f'(5) is wrong!"
assert abs(d_f(10) - -10*np.exp(-10**2/2)) <= 1e-7, "f'(10) is wrong!"
### END HIDDEN TESTS
```

```
f(0) = 1.0
f'(0) = 0.0
```

Part 3.3

Compute

$$\int_0^1 e^{-\frac{x^2}{2}} dx$$

with the formula $(*)$.

Fill your answer into the variable `approximation`.

In [7]:

(Top)

```
approximation = alpha_1 * f(0) + alpha_2 * f(1) + alpha_3 * d_f(0)
```

Run and check your answer.

In [8]:

part_3_2 (Top)

```
print("The result of the integral is", approximation)
### BEGIN HIDDEN TESTS
assert abs(approximation - 0.8688435532375445) < 1e-3, "wrong approximation!"
### END HIDDEN TESTS
```

```
The result of the integral is 0.8688435532375445
```

Part 3.4

Compute

$$\int_0^1 e^{-\frac{x^2}{2}} dx$$

with Simpson's rule.

Implement Simpson's rule

In [9]:

(Top)

```
def simpson(
    f,
    a,
    b,
    N=50
):
    '''
    Parameters
    ----------
    f : function
        Vectorized function of a single variable
    a , b : numbers
        Interval of integration [a,b]
    N : (even) integer
        Number of subintervals of [a,b]

    Returns
    -------
    S : float
        Approximation of the integral of f(x) from a to b using
        Simpson's rule with N subintervals of equal length.
    '''
    delta_x = (b - a) / N
    return (delta_x / 3) * sum([f(a + (2 * i - 2) * delta_x) + 4 * f(a + (2 * i - 1) * delta_x) + f(a + (
2 * i) * delta_x) for i in range(1 , N // 2)])
```

Run and check your function.

In [10]:

```
        simpson                                                                                    (Top)
```

```
S = simpson(f, 0, 1, N=50)
print("The result from Simpson's rule is", S)
### BEGIN HIDDEN TESTS
assert abs(S - 0.8556243929705796) < 1e-7, "Wrong answer!"
### END HIDDEN TESTS
```

```
The result from Simpson's rule is 0.8308780725021094

---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-10-cf44ad25f30e> in <module>
      2 print("The result from Simpson's rule is", S)
      3 ### BEGIN HIDDEN TESTS
----> 4 assert abs(S - 0.8556243929705796) < 1e-7, "Wrong answer!"
      5 ### END HIDDEN TESTS

AssertionError: Wrong answer!
```

Part 3.5

Compute

$$\int_0^1 e^{-\frac{x^2}{2}} dx$$

with the Gauss-Legendre formula using $n = 1$.

In [11]:

```
                                                                                                   (Top)
```

```
def gauss(
    f,
    n,
    a,
    b
):
    '''
    Parameters
    ----------
    f : function
        Vectorized function of a single variable
    n : integer
        Number of points
    a , b : numbers
        Interval of integration [a,b]

    Returns
    -------
    G : float
        Approximation of the integral of f(x) from a to b using the
        Gaussian–Legendre quadrature rule with N points.
    '''
    [x , w] = p_roots(n)
    m = (b - a) / 2
    n = (b + a) / 2
    return m * sum(w * f(m * x + n))
```

Run and check your function.

In [12]:

```
          Gauss-Legendre
```
(Top)

```
G = gauss(f, 1, 0, 1)
print("The result from Gauss-Legendre is", G)
### BEGIN HIDDEN TESTS
assert abs(G - 0.88) <= 1e-1, "Wrong answer!"
### END HIDDEN TESTS
```

```
The result from Gauss-Legendre is 0.8824969025845955
```

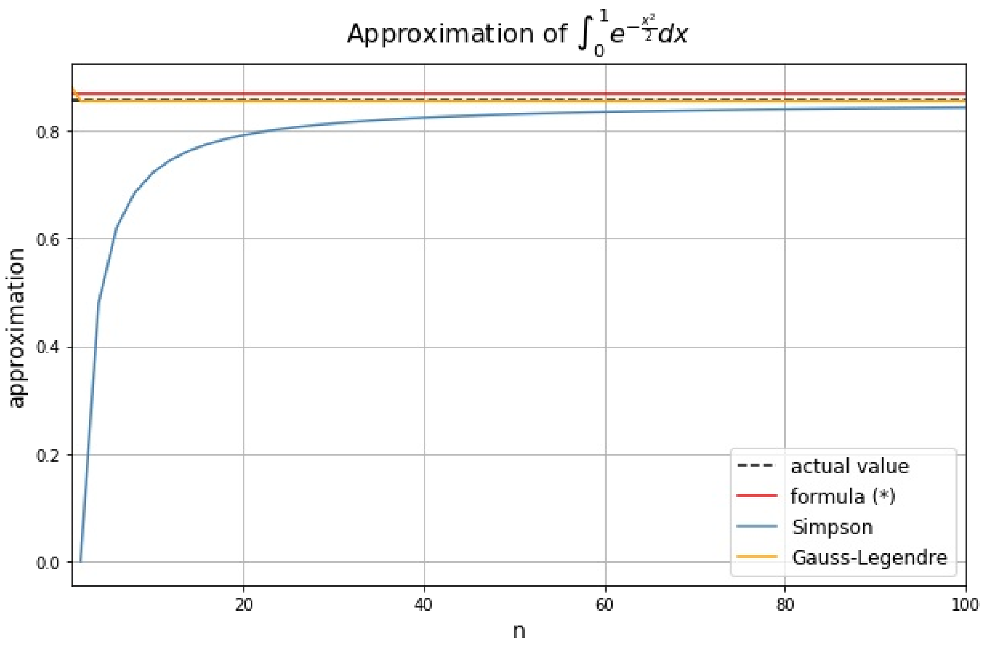
(Top)

Part 3.6

Compare the obtained results of three methods above and write down your observation. You can use either code or markdown to depict.

In [13]:

```
plt.figure(figsize = (10 , 6))
plt.title(r'Approximation of $\int_{0}^{1} e^{-\frac{x^2}{2}} dx$' , fontsize = 16 , y = 1.02)
plt.axhline(y = 0.855624 , linestyle = '--' , color = 'black' , label = 'actual value')
plt.axhline(y = approximation , color = 'red' , label = 'formula (*)')
n_range = [n for n in range(2 , 101 , 2)]
plt.plot(n_range , [simpson(f , 0 , 1 , n) for n in n_range] , color = 'steelblue' , label = 'Simpson')
n_range = [n for n in range(1 , 101 , 1)]
plt.plot(n_range , [gauss(f , n , 0 , 1) for n in n_range] , color = 'orange' , label = 'Gauss-Legendre')
plt.legend(loc = 'lower right' , fontsize = 12)
plt.xlabel('n' , fontsize = 14)
plt.ylabel('approximation' , fontsize = 14)
plt.xlim([1 , 100])
plt.grid(True)
plt.show()
```



(上圖中 $\int_0^1 e^{-\frac{x^2}{2}} dx$ 的值是將WolframAlpha所計算出來的估計值當作實際值而得)
由上圖可以看出，Gaussian-Legendre formula的估計值在區間數目n很少時就已經收斂，非常接近實際值，因此隨著n變大時，估計值幾乎沒什麼變化。
而Simpson formula的估計值隨著點的數目n變大時，會逐漸收斂至實際值。
而formula (*)是直接計算出估計值而未進行迭代，因此估計值為常數，其誤差較前兩種方法來得大。

In [ ]: