

## 數值 hw4

1.

```
main.py
1 import numpy as np
2
3 def f(x):
4     return np.exp(x) * np.sin(4*x)
5
6 def composite_trapezoidal(a, b, h):
7     n = int((b - a) / h)
8     x = np.linspace(a, b, n + 1)
9     y = f(x)
10    return (h / 2) * (y[0] + 2 * np.sum(y[1:n]) + y[n])
11
12 def composite_simpson(a, b, h):
13     n = int((b - a) / h)
14     if n % 2 != 1:
15         n += 1 # Simpson's rule requires an even number of intervals
16     x = np.linspace(a, b, n + 1)
17     y = f(x)
18     return (h / 3) * (y[0] + 4 * np.sum(y[1:n:2]) + 2 * np.sum(y[2:n-1:2]) + y[n])
19
20 def composite_midpoint(a, b, h):
21     n = int((b - a) / h)
22     x = np.linspace(a, b, n + 1)
23     y = f(x)
24     return h * np.sum(y[1:n])
```

input

Composite Trapezoidal Rule: 0.396148  
Composite Simpson's Rule: 0.385664  
Composite Midpoint Rule: 0.380805

...Program finished with exit code 0

2.

```
main.py
1 import numpy as np
2 import scipy.integrate as spi
3
4 def f(x):
5     return x**2 * np.log(x)
6
7 def gauss_quadrature(a, b, n):
8     # 取得勒讓德多項式的根與權重
9     x, w = np.polynomial.legendre.leggauss(n)
10    # 變換積分區間到 [a, b]
11    t = 0.5 * (x + 1) * (b - a) + a
12    return 0.5 * (b - a) * np.sum(w * f(t))
13
14 # 設定積分區間
15 a, b = 1, 1.5
16
17 # 計算 Gaussian Quadrature 近似值
18 I_gauss_3 = gauss_quadrature(a, b, 3)
19 I_gauss_4 = gauss_quadrature(a, b, 4)
20
21 # 計算精確值
22 I_exact = spi.integrate.quad(f, a, b)
```

input

Gaussian Quadrature (n=3): 0.192259  
Gaussian Quadrature (n=4): 0.192259  
Exact Integral Value: 0.192259

...Program finished with exit code 0

3.

```

1 import numpy as np
2 import scipy.integrate as spi
3
4 def f(x, y):
5     return 2 * y * np.sin(x) + np.cos(x)**2
6
7 def simpsons_double_integral(a, b, n, m):
8     x = np.linspace(a, b, n + 1)
9     hx = (b - a) / n
10    integral = 0
11
12    for i in range(n + 1):
13        xi = x[i]
14        ya, yb = np.sin(xi), np.cos(xi)
15        y = np.linspace(ya, yb, m + 1)
16        hy = (yb - ya) / m
17
18        fy = f(xi, y)
19
20        simpson_u = hy / 3 * ((fy[0] + fy[-1]) + np.sum((fy[1:-1]) * 4) - np.sum((fy[2:-2]) * 2))

```

Simpson's Rule: 0.511988  
Gaussian Quadrature: 0.511866  
Exact Integral Value: 0.404120

...Program finished with exit code 0

4.

```

1 import numpy as np
2 import scipy.integrate as spi
3
4 def simpsons_rule(f, a, b, n):
5     if n % 2 == 1:
6         raise ValueError("n must be even for Simpson's rule")
7
8     h = (b - a) / n
9     x = np.linspace(a, b, n + 1)
10    y = f(x)
11
12    integral = (h / 3) * (y[0] + 4 * sum(y[1:n:2]) + 2 * sum(y[2:n-1:2]) + y[n])
13    return integral
14
15 def f1(x):
16     return x**(-1/4) * np.sin(x)
17
18 def f2(x):
19     return x**(-4) * np.sin(x)
20

```

Approximate integral for (a): 6.94708410137921  
Approximate integral for (b): 0.27465825002428546

...Program finished with exit code 0  
Press ENTER to exit console.