HW 3

Author: Gavin DeBrun

1a)

We compute the value of the function $f(x) = x^{1.5}cos(x^2)$ at the following points $(x_j, x_{j+1}) = \{(0, \frac{\pi}{8}), (\frac{\pi}{8}, \frac{\pi}{4}), (\frac{\pi}{4}, \frac{3\pi}{8}), (\frac{3\pi}{8}, \frac{\pi}{2})\}$ and use the trapezoid area formula $A_{trap} = \frac{h}{2}[f_j + f_{j+1}]$ to approximate the value of the integral I_j over the subinterval. Then, we sum the areas of the trapezoids to approximate the value of the integral I over the entire interval $[0, \frac{\pi}{2}]$.

```python
import numpy as np


def f(x):
    return x**1.5 * np.cos(x**2)


h = np.pi / 8
xs = np.arange(0, np.pi / 2 + h, h)
fs = f(xs)
N = fs.shape[0]

I = 0
for i in range(1, N - 1):
    I += h * fs[i]

I += h / 2 * (fs[0] + fs[-1])

print(I)
```

```
0.1077949713288272
```

Using the trapezoid rule, we get $I \approx .10779$

1b)

Now instead of using trapezoids, we approximate the integral by calculating the sum of the areas of rectangles with width h whose height is given by the value of the function at the midpoint of the subinterval.

Concretely, $x_i = \{\frac{\pi}{16}, \frac{3\pi}{16}, \frac{5\pi}{16}, \frac{7\pi}{16}\}$

```
xs = np.arange(np.pi / 16, 7 * np.pi / 16 + h, h)
fs = f(xs)

I = np.sum(fs * h)
print(I)
```

```
0.22094613205800323
```

Using the midpoint rule, we get $I \approx .22029$

1c)

The end corrected trapezoid rule is given by $I \approx \sum_{j=0}^{N-1} \frac{h}{2}[f_j + f_{j+1}] - \frac{h^2}{12}[f'(b) - f'(a)]$ where $f'(x) = 1.5x^{.5}cos(x^2) - 2x^{2.5}sin(x^2)$

```
h = np.pi / 8
xs = np.arange(0, np.pi / 2 + h, h)
fs = f(xs)
N = fs.shape[0]

I = 0
for i in range(1, N - 1):
    I += h * fs[i]

I += h / 2 * (fs[0] + fs[-1])


def fp(x):
    return 1.5 * x**0.5 * np.cos(x**2) - 2 * x**2.5 * np.sin(x**2)


I -= h**2 / 12 * (fp(np.pi / 2) - fp(0))
print(I)
```

```
0.1762865784049138
```

Using the end corrected trapezoid rule, we get $I \approx .1763$

1d)

The Simpson approximation is given by $I \approx \sum_j I_j$ wjere $I_j = \frac{h}{3}[f(x_j) + 4f(x_{j+1}) + f(x_{j+2})]$

```
h = np.pi / 8
xs = np.arange(0, np.pi / 2 + h, h)
fs = f(xs)
N = fs.shape[0]

I = 0

for i in range(0, N - 2, 2):
    I += h / 3 * (fs[i] + 4 * fs[i + 1] + fs[i + 2])

print(I)
```

```
0.1964065881916026
```

Using the Simpson approximation, we get $I \approx .1964$

1e)

```
from scipy.integrate import quad

I, err = quad(f, 0, np.pi / 2)

print(I)
```

```
0.18213710308632183
```

The closest approximation to scipy.integrate.quad is the end corrected trapezoid rule.

2)

$f'_{i,h} = \frac{f_i - f_{i-1}}{h} - hc_1 + h^2 c_2 + O(h^3)$

$f'_{i,2h} = \frac{f_i - f_{i-2}}{2h} - 2hc_1 + 4h^2 c_2 + O(h^3)$

$f'_{i,3h} = \frac{f_i - f_{i-3}}{3h} - 3hc_1 + 9h^2 c_2 + O(h^3)$

$af'_{i,h} + bf'_{i,2h} + cf'_{i,3h} = (a+b+c)f'_i - hc_1(a+2b+3c) + h^2c_2(a+4b+9c) + O(h^3)$

We want:

$a + b + c = 1$

$a + 2b + 3c = 0$

$a + 4b + 9c = 0$

```
import sympy as sym

a = sym.Symbol("a")
b = sym.Symbol("b")
c = sym.Symbol("c")

sym.solve([a + b + c - 1, a + 2 * b + 3 * c, a + 4 * b + 9 * c], [a, b, c])
```

```
{a: 3, b: -3, c: 1}
```

Thus,

$f'_i \approx \frac{3f_i - 3f_{i-1}}{h} - \frac{3f_i - 3f_{i-2}}{2h} + \frac{f_i - f_{i-3}}{3h} + O(h^3)$

$f'_i \approx \frac{18f_i - 18f_{i-1}}{6h} - \frac{9f_i - 9f_{i-2}}{6h} + \frac{2f_i - 2f_{i-3}}{6h} + O(h^3)$

$$\boxed{f'_i \approx \frac{11f_i - 18f_{i-1} + 9f_{i-2} - 2f_{i-3}}{6h} + O(h^3)}$$

3)

Romberg integration with two stepsizes h and $\frac{h}{2}$ gives us:

$I = \frac{4I_2 - I_1}{3}$ where $I_1 = 12.045$ and $I_2 = 11.801$

```
print((4 * 11.801 - 12.045) / 3)
```

```
11.719666666666667
```

Thus, a better approximation is $I \approx 11.7197$

4)

```
def simpson_integrate(x, f):
    I = 0
    dxj = x[2] - x[0]
    n = x.shape[0]
    if n % 2 == 0:
        # do trapezoid at end
        for i in range(0, n - 2, 2):
            I += dxj / 6 * (f[i] + 4 * f[i + 1] + f[i + 2])

        I += (dxj / 2) * (f[n - 2] + f[n - 1])

    else:
        for i in range(0, n - 2, 2):
            I += dxj / 6 * (f[i] + 4 * f[i + 1] + f[i + 2])

    return I
```