1 River Flow with Discrete Calculus Integration

1.0.1 (a) Analytic Area Computation

See handwritten homework. Analytical cross-sectional river area was computed as:

$$A_{\text{analytical}} = 21.6\bar{6}\,\mathrm{m}^2$$

1.0.2 (b) Numerical Area Computation

Using given table:

i	**0**	**1**	**2**	**3**	**4**	**5**	**6**	$n = 7$
y_i [m]	0	1	3	5	7	8	9	10
H_i [m]	0	1	1.2	3	3.5	3.2	2	0
U_i $\left[\frac{\mathrm{m}}{\mathrm{s}}\right]$	0	0.1	0.15	0.2	0.25	0.3	0.15	0

[1]:
```python
## Imports
import numpy as np

## Allocate Discrete Data
y = np.array([0,1,3,5,7,8,9,10])
H = np.array([0,1,1.2,3,3.5,3.2,2,0])
U = np.array([0,0.1,0.15,0.2,0.25,0.3,0.15,0])
```

Use trapezoidal method:

$$A = \sum_{i=0}^{n-1} \frac{(y_{i+1} - y_i)}{2} \cdot (H_{i+1} + H_i)$$

[2]:
```python
def trapezoidal_filter(y_current: float, y_next:float) -> np.ndarray:
    """Returns a 1D trapezoidal integration filter"""
    assert( y_next > y_current )
    return np.array([0.5,0.5])*(y_next - y_current)

def trapezoidal_integration(x: np.ndarray, F: np.ndarray) -> float:
    """Integrated a function F over a given discrete domain x using trapezoidal␣
↪integration"""
    stride = 1
    area_trapezoidal = 0

    ## Perform convolution:
    for ii in range(0, len(x)-1, stride):
```

```python
        area_trapezoidal += np.dot( trapezoidal_filter(x[ii],x[ii+1]), F[ii:ii+1+stride] )
    return area_trapezoidal

print("Area computed using trapezoidal rule in m^2: ", trapezoidal_integration(y, H))
```

```
Area computed using trapezoidal rule in m^2:  20.35
```

Use simpson method:

Using the classical Simpson rule is only possible for even data spacing. Since the data provided in unevenly spaced, we must use the alternative Composite Simpson's rule.

[3]:
```python
def simpson_integration(x: np.ndarray, F: np.ndarray) -> float:
    """Integrated a function F over a given discrete domain x using composite Simpson integration.
    Using modified code from https://en.wikipedia.org/wiki/Simpson%27s_rule"""
    stride = 2
    area_trapezoidal = 0

    h = [x[i + 1] - x[i] for i in range(0, len(x)-1)]
    assert len(x)-1 > 0

    for i in range(1, len(x)-1, stride):
        h0, h1 = h[i - 1], h[i]
        hph, hdh, hmh = h1 + h0, h1 / h0, h1 * h0
        area_trapezoidal += (hph / 6) * (
            (2 - hdh) * F[i - 1] + (hph**2 / hmh) * F[i] + (2 - 1 / hdh) * F[i + 1]
        )

    if (len(x)-1) % 2 == 1:
        h0, h1 =  h[len(x)-3], h[len(x)-2]
        area_trapezoidal += F[len(x)-1] * (2 * h1 ** 2 + 3 * h0 * h1) / (6 * (h0 + h1))
        area_trapezoidal += F[len(x)-2] * (h1 ** 2 + 3 * h1 * h0)     / (6 * h0)
        area_trapezoidal -= F[len(x)-3] * h1 ** 3                     / (6 * h0 * (h0 + h1))
    return area_trapezoidal

print("Area computed using simpsons rule in m^2: ", simpson_integration(y, H))
```

```
Area computed using simpsons rule in m^2:  21.45
```

We can compare our results as follows and see that Simpson's rule is more accurate than Trapezoidal:

$$A_{\text{analytical}} = 21.6\bar{6}\,\text{m}^2$$

$$A_{\text{trapezoidal}} = 20.35\,\text{m}^2$$
$$A_{\text{simpson}} = 21.45\,\text{m}^2$$

We do not perform Gauss' method due to missing exact quadrature points.

1.0.3 (c) Numerical Flow Rate Computation

Use previous methods again with modified argument:

$$Q = \int_0^{10} H(y) \cdot U(y) dy \approx \sum_{i=0}^{n-1} \frac{(y_{i+1} - y_i)}{2} \cdot ((H_{i+1} \cdot U_{i+1}) + (H_i \cdot U_i))$$

[4]:
```
print("Flow rate computed using trapezoidal rule: ", trapezoidal_integration(y,␣
 ↪np.multiply(H,U)))
print("Flow rate computed using simpsons rule: ", simpson_integration(y, np.
 ↪multiply(H,U)))
```

```
Flow rate computed using trapezoidal rule:  4.282500000000001
Flow rate computed using simpsons rule:  4.455
```

$$Q_{\text{trapezoidal}} = 4.2825$$
$$Q_{\text{simpson}} = 4.455$$