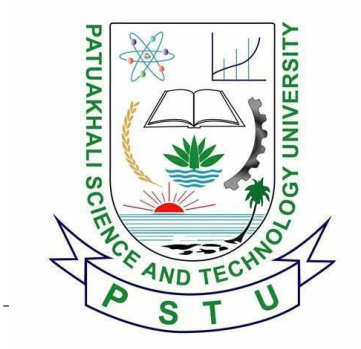


PATUAKHALI SCIENCE AND TECHNOLOGY UNIVERSITY

COURSE CODE CCE 312
Numerical Methods Sessional

SUBMITTED TO:

Prof. Dr. Md Samsuzzaman
Department of Computer and Communication Engineering
Faculty of Computer Science and Engineering

SUBMITTED BY:

Md. Sharafat Karim
ID: **2102024**,
Registration No: **10151**
Faculty of Computer Science and Engineering

Assignment 09
Assignment title: Simpsons Rule
Date of submission: 24 Sun, Aug 2025

Integrals

Sharafat Karim

CONTENTS

Let's approximate the integral of a function using numerical methods.

Trapezoidal Rule

At first our necessary libs,

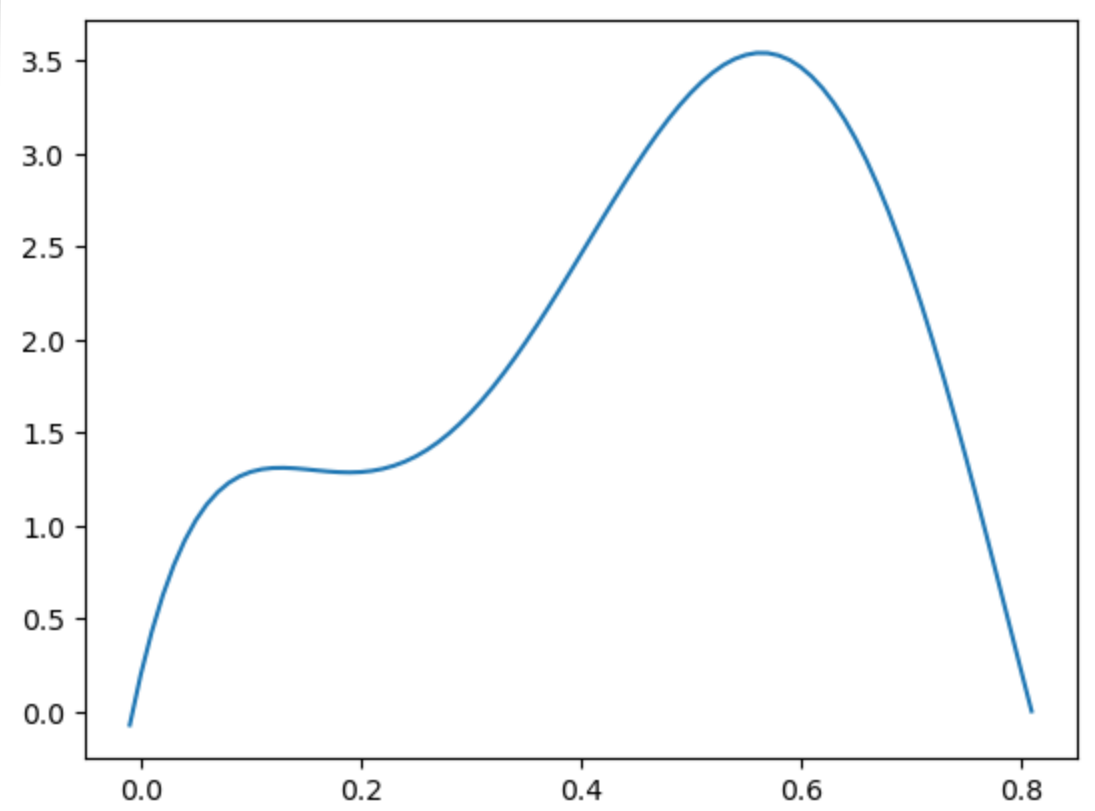
```
import numpy as np
import matplotlib.pyplot as plt
```

Let's define a sample function to integrate.

```
def fun(x):
  return 0.2 + 25 * x - 200 * x**2 + 675 * x**3 - 900 * x**4 + 400 * x**5
```

Let's plot it!

```
array = np.arange(-0.01, 0.82, 0.01)
plt.plot(array, fun(array))
```



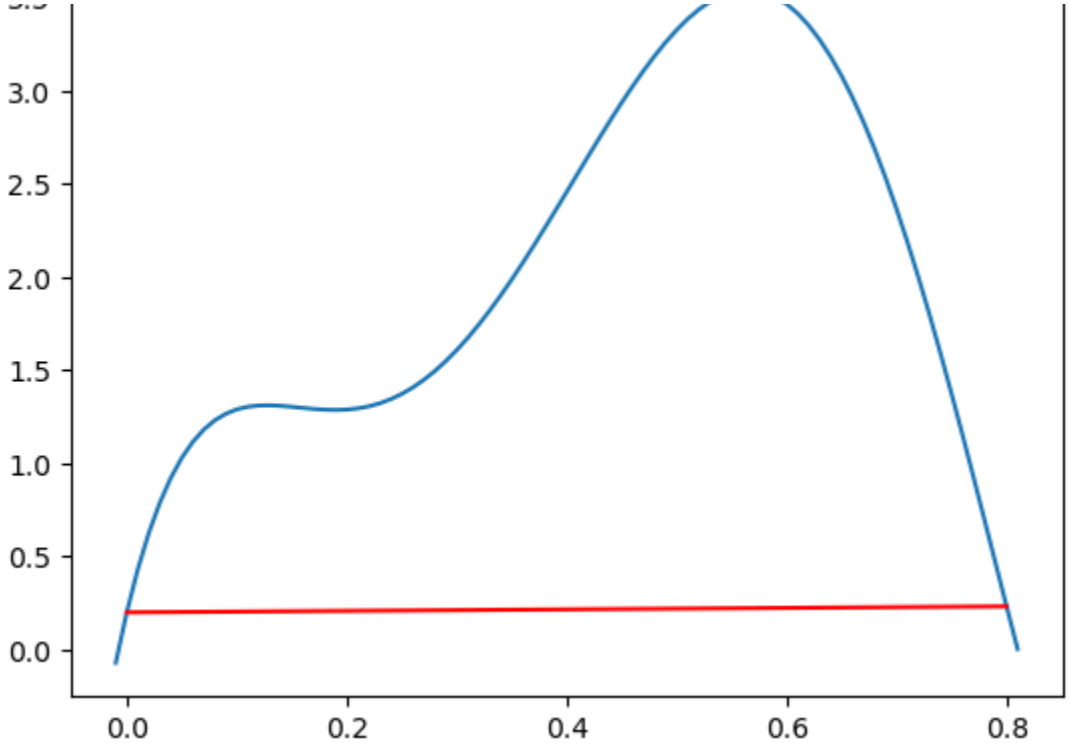
Let's define our Trapezoidal Rule!

```
def trapezoidal_rule(fun, a, b):
  return (b - a) * (fun(a) + fun(b)) / 2
```

Our trapezoidal rule function is ready to use!

```
print(trapezoidal_rule(fun, 0, 0.8))
plt.plot(array, fun(array))
plt.plot([0, 0.8], [fun(0), fun(0.8)], color='red')
```

```
0.1728000000000225
```



Simpon's 1/3 Rule

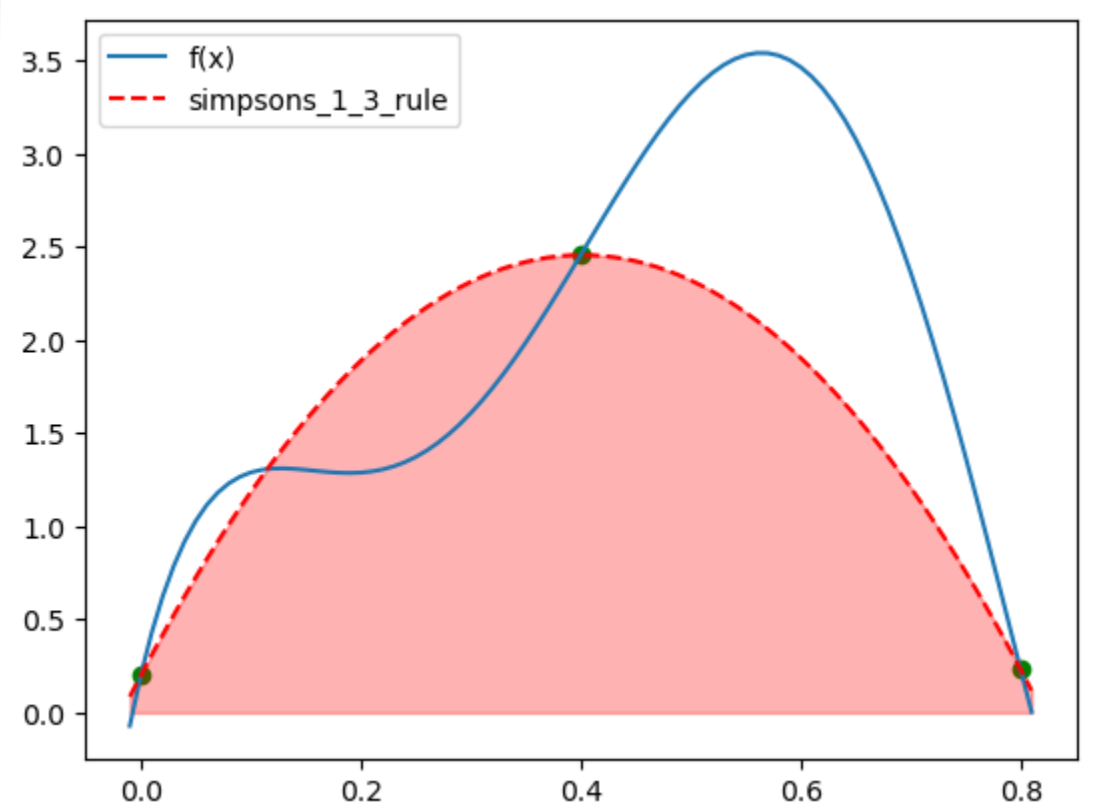
Let's implement it,

```
def simpsons_1_3_rule(fun, a, b):
    return (b - a) / 6 * (fun(a) + 4 * fun((a + b) / 2) + fun(b))
```

Let's plot it,

```
from scipy.interpolate import CubicSpline

print(simpsons_1_3_rule(fun, 0, 0.8))

plt.plot(array, fun(array), label="f(x)")
plt.scatter([0, 0.4, 0.8], [fun(0), fun(0.4), fun(0.8)], color='green')
plt.plot(array, CubicSpline([0, 0.4, 0.8], [fun(0), fun(0.4), fun(0.8)])(array), '--',
plt.fill_between(array, CubicSpline([0, 0.4, 0.8], [fun(0), fun(0.4), fun(0.8)])(array)
plt.legend()
```

```
1.3674666666666742
```



Simpon's 3/8 Rule

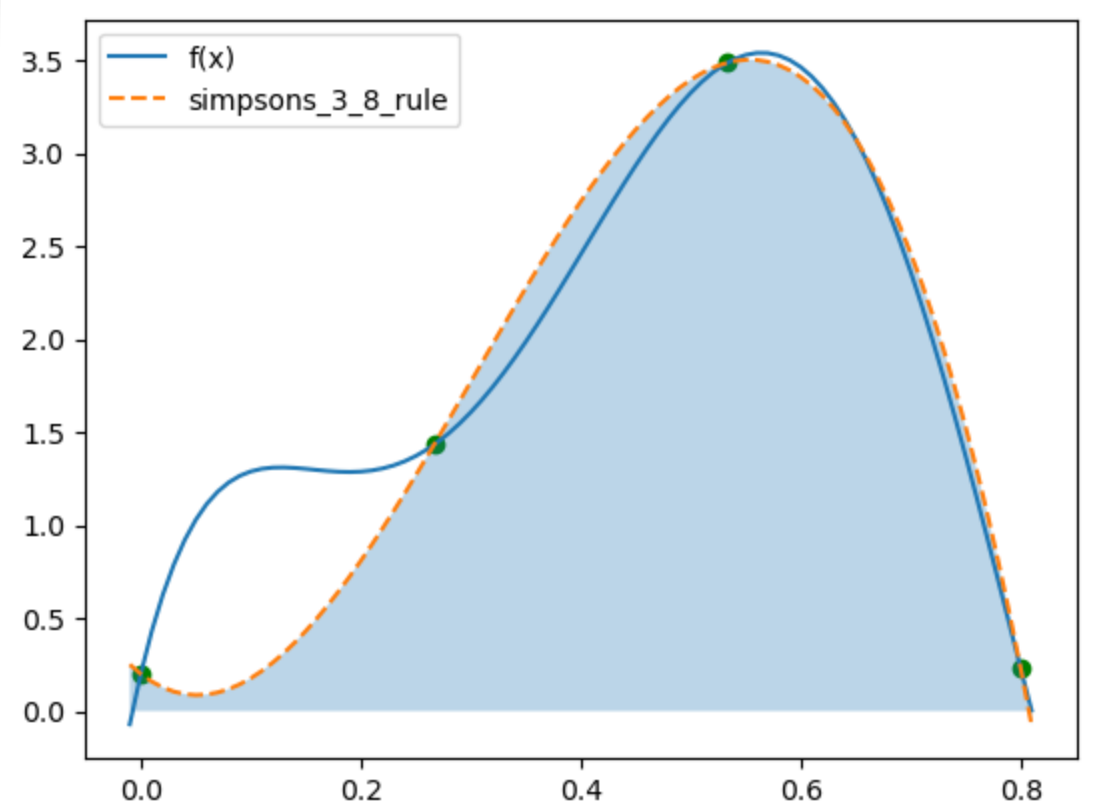
```
def simpsons_3_8_rule(fun, a, b):
    a, b = min(a, b), max(a, b)
    h = (b - a) / 3
    return (3 * h / 8) * (fun(a) + 3 * fun(a + h) + 3 * fun(a + 2 * h) + fun(b))
```

Let's plot it,

```
from scipy.interpolate import CubicSpline

a, b, c, d = 0, 0.8/3, 0.8/3*2, 0.8
print(simpsons_3_8_rule(fun, a, d))

plt.plot(array, fun(array), label="f(x)")
plt.plot(array, CubicSpline([a, b, c, d], [fun(a), fun(b), fun(c), fun(d)])(array), '--
plt.scatter([a, b, c, d], [fun(a), fun(b), fun(c), fun(d)], color='green')
plt.fill_between(array, CubicSpline([a, b, c, d], [fun(a), fun(b), fun(c), fun(d)])(arr
```

```
plt.legend()
```

```
1.519170370370378
```

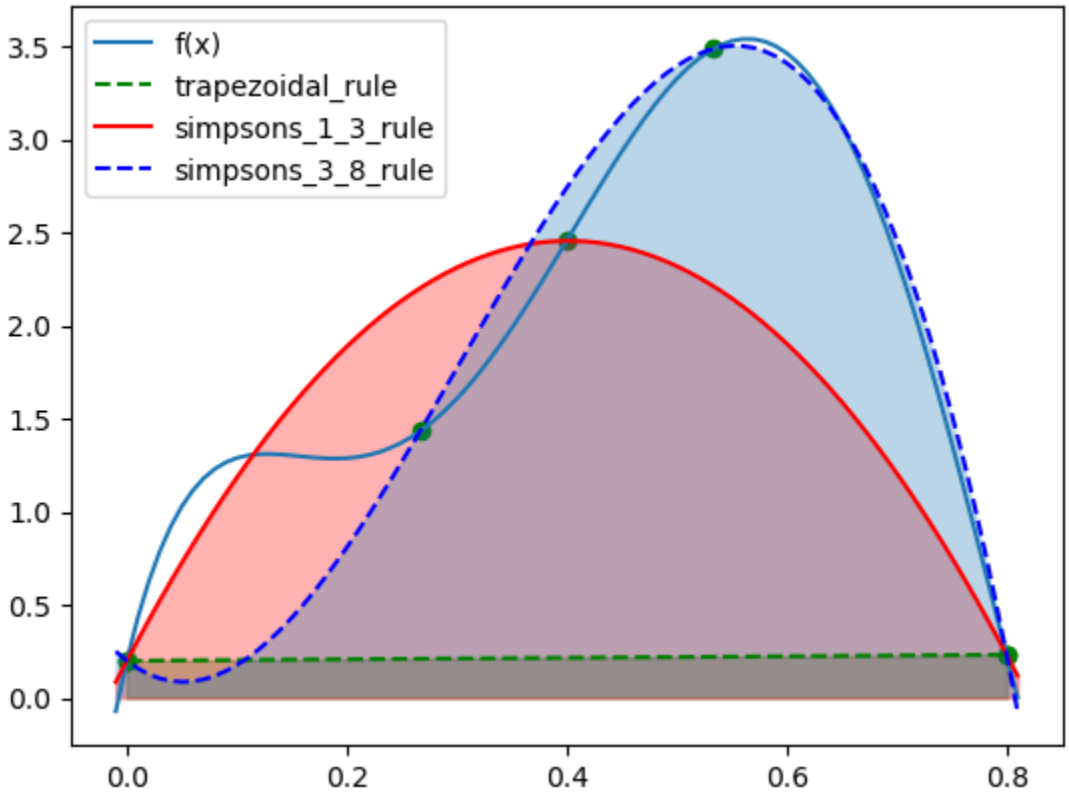


Let's plot all of them altogether,

```
# Main function
plt.plot(array, fun(array), label="f(x)")

# Trapezoidal rule
print("Trapezoidal rule   -> ", trapezoidal_rule(fun, 0, 0.8))
plt.plot([0, 0.8], [fun(0), fun(0.8)], '--', color='green', label='trapezoidal_rule')
plt.fill_between([0, 0.8], [fun(0), fun(0.8)], color='green', alpha=0.3)

# Simpson's 1/3 rule
print("Simpson's 1/3 rule -> ", simpsons_1_3_rule(fun, 0, 0.8))

plt.scatter([0, 0.4, 0.8], [fun(0), fun(0.4), fun(0.8)], color='green')
plt.plot(array, CubicSpline([0, 0.4, 0.8], [fun(0), fun(0.4), fun(0.8)])(array), color=
plt.fill_between(array, CubicSpline([0, 0.4, 0.8], [fun(0), fun(0.4), fun(0.8)])(array)

a, b, c, d = 0, 0.8/3, 0.8/3*2, 0.8

# Simpson's 3/8 rule
print("Simpson's 3/8 rule -> ", simpsons_3_8_rule(fun, a, d))

plt.scatter([a, b, c, d], [fun(a), fun(b), fun(c), fun(d)], color='green')
```

```
plt.plot(array, CubicSpline([a, b, c, d], [fun(a), fun(b), fun(c), fun(d)])(array), '--
plt.fill_between(array, CubicSpline([a, b, c, d], [fun(a), fun(b), fun(c), fun(d)])(arr

plt.legend()
```

```
Trapezoidal rule   ->  0.1728000000000225
Simpson's 1/3 rule ->  1.3674666666666742
Simpson's 3/8 rule ->  1.519170370370378
```



Numerical Methods
Polynomial Regression

Data Analysis
Data Analysis