HW_7_code

May 6, 2022

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

0.1 Q3

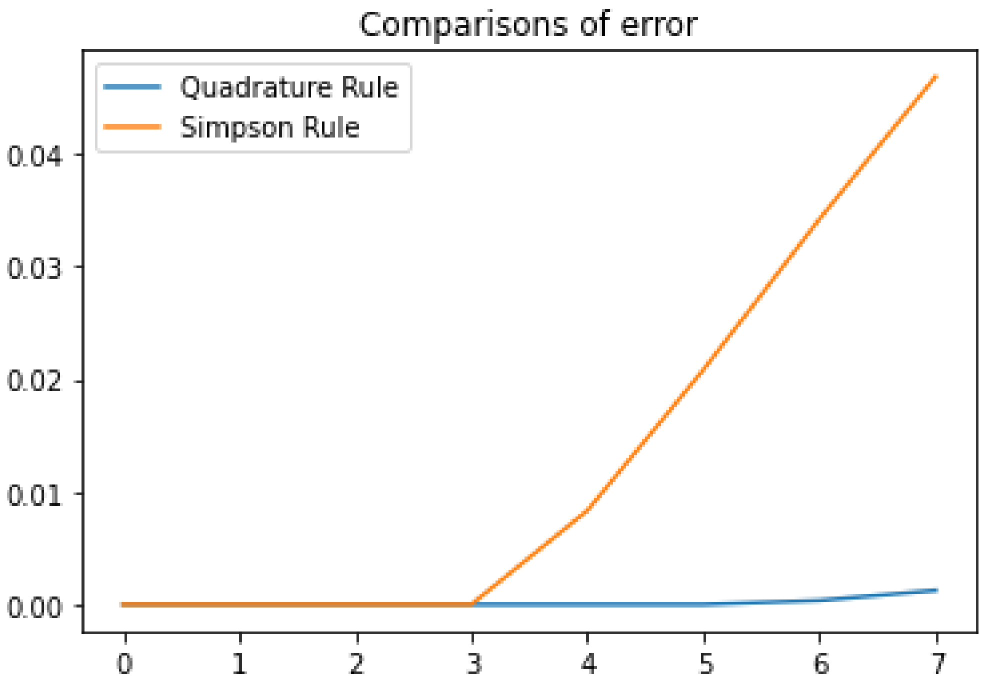
[2]:
```
def quadrature(f):
    X = np.array([1/2, 1/2 + (1/2)*np.sqrt(3/5), 1/2 - (1/2)*np.sqrt(3/5)])
    W = np.array([4/9, 5/18, 5/18])
    y = np.array([f(x) for x in X])

    return np.dot(y, W)


def simpson(f, a=0, b=1):
    c = (a + b) / 2
    return ((b-a)/6) * (f(a) + 4*f(c) + f(b))
```

[3]:
```
error_qua_hist = []
error_sim_hist = []
for k in range(0, 8):
    f = lambda x: x**k
    exact = 1/(k+1)
    error_qua = abs(exact - quadrature(f))
    error_sim = abs(exact - simpson(f))
    error_qua_hist.append(error_qua)
    error_sim_hist.append(error_sim)
```

[4]:
```
k_space = list(range(0, 8))
plt.plot(k_space, error_qua_hist, label='Quadrature Rule')
plt.plot(k_space, error_sim_hist, label='Simpson Rule')
plt.title('Comparisons of error')
plt.legend()
plt.show()
```



- As we can see from the plot, the quadrature rule, which comes from hermite interpolation, is exact up to degree $2n + 1 = 5$.
- In contrast, as for simpson's rule, which comes from Lagrange interpolation, is exact up to degree $n = 2$.
- Note that, the plot reveal that it is exact up to degree $n = 3$. It is merely a coincidence. As the lagrange interpolation of x^3 at the three nodes gives us

$$p(x) = \frac{3}{2}x^2 - \frac{1}{2}x$$

and

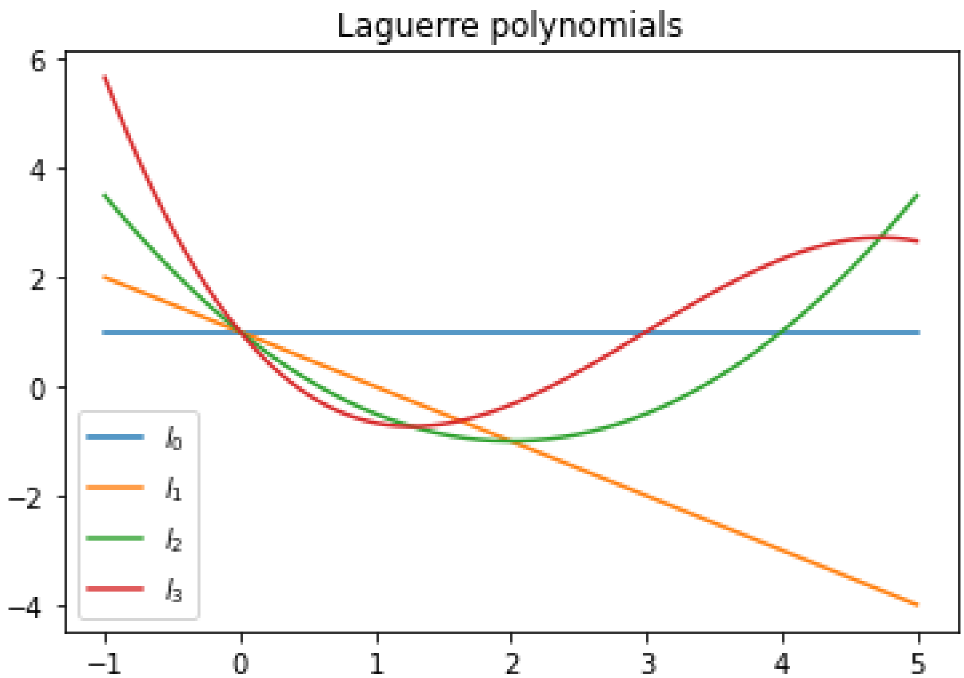
$$\int_0^1 x^3 dx = \int_0^1 p(x)dx = \frac{1}{4}$$

0.2 Q4

[5]:
```
l_0 = lambda x: 1
l_1 = lambda x: -x + 1
l_2 = lambda x: (1/2) * (x**2 - 4*x + 2)
l_3 = lambda x: (1/6) * (-x**3 + 9 * x**2 - 18*x + 6)
```

[6]:
```
x_space = np.linspace(-1, 5, 1001)
plt.plot(x_space, [l_0(x) for x in x_space], label=r'$l_0$')
plt.plot(x_space, [l_1(x) for x in x_space], label=r'$l_1$')
plt.plot(x_space, [l_2(x) for x in x_space], label=r'$l_2$')
plt.plot(x_space, [l_3(x) for x in x_space], label=r'$l_3$')
plt.legend()
```

```
plt.title('Laguerre polynomials')
plt.show()
```



[7]:

```
def quadrature(n, f):
    if n == 2:
        X = np.array([0.58576, 3.41421])
        W = np.array([0.853553, 0.146447])
    if n == 3:
        X = np.array([0.4157745, 2.29428, 6.28995])
        W = np.array([0.711093, 0.278518, 0.0103893])
    if n == 4:
        X = np.array([0.322548, 1.74576, 4.53662, 9.39507])
        W = np.array([0.603154, 0.357419, 0.0388879, 0.000539295])

    y = np.array([f(x) for x in X])

    return np.dot(y, W)
```

[8]:

```
exact = 1/2
f = lambda x: np.exp(-x)
for n in range(2, 5):
    print(f'n = {n}, error = {abs(quadrature(n, f) - exact)}')
```

```
n = 2, error = 0.020023421556548896
n = 3, error = 0.0026968391830128335
```

```
n = 4, error = 0.00034432387742505677
```

[9]:
```python
exact = np.pi ** .5 /2
f = lambda x: np.exp(-x**2 + x)
for n in range(2, 5):
    print(f'n = {n}, error = {abs(quadrature(n, f) - exact)}')
```

```
n = 2, error = 0.20176421812245082
n = 3, error = 0.0346771708037783
n = 4, error = 0.0385477868412718
```