Lab02 Exercises Template

December 6, 2024

0.1 Example 1

Evaluate the integral of $f(x) = x^4 - 2x^3 + 1$ on the interval $[0, 2]$: first symbolically, and then numerically using the trapezoidal rule.

Note, we can evaluate the integral symbolically as long as the function has an analytical expression that SymPy (or Maple, Mathematica, Wolfram Alpha…) knows how to solve. Otherwise, numerical integration is required.

[20]:
```python
from sympy import *  # the symbolic math package

def f(x):
    return x**4 - 2*x**3 + 1

a = 0.0  # beginning of interval
b = 2.0  # end of interval
```

[21]:
```python
# symbolic integration
xs = Symbol('xs', real=True)  # the variable of integration
integrate(f(xs), (xs, a, b))
```

[21]: 0.4

[43]:
```python
import numpy as np

# trapezoidal integration
def trapezoidal_rule(f, a, b, n):
    x = np.linspace(a, b, n+1)
    y = f(x)
    h = (b - a) / n
    integral = h * (0.5 * y[0] + 0.5 * y[-1] + np.sum(y[1:-1]))
    return integral

n = 1000  # number of trapezoids
integral = trapezoidal_rule(f, a, b, n)
integral
```

[43]: 1.02053999051608

0.2 Example 2

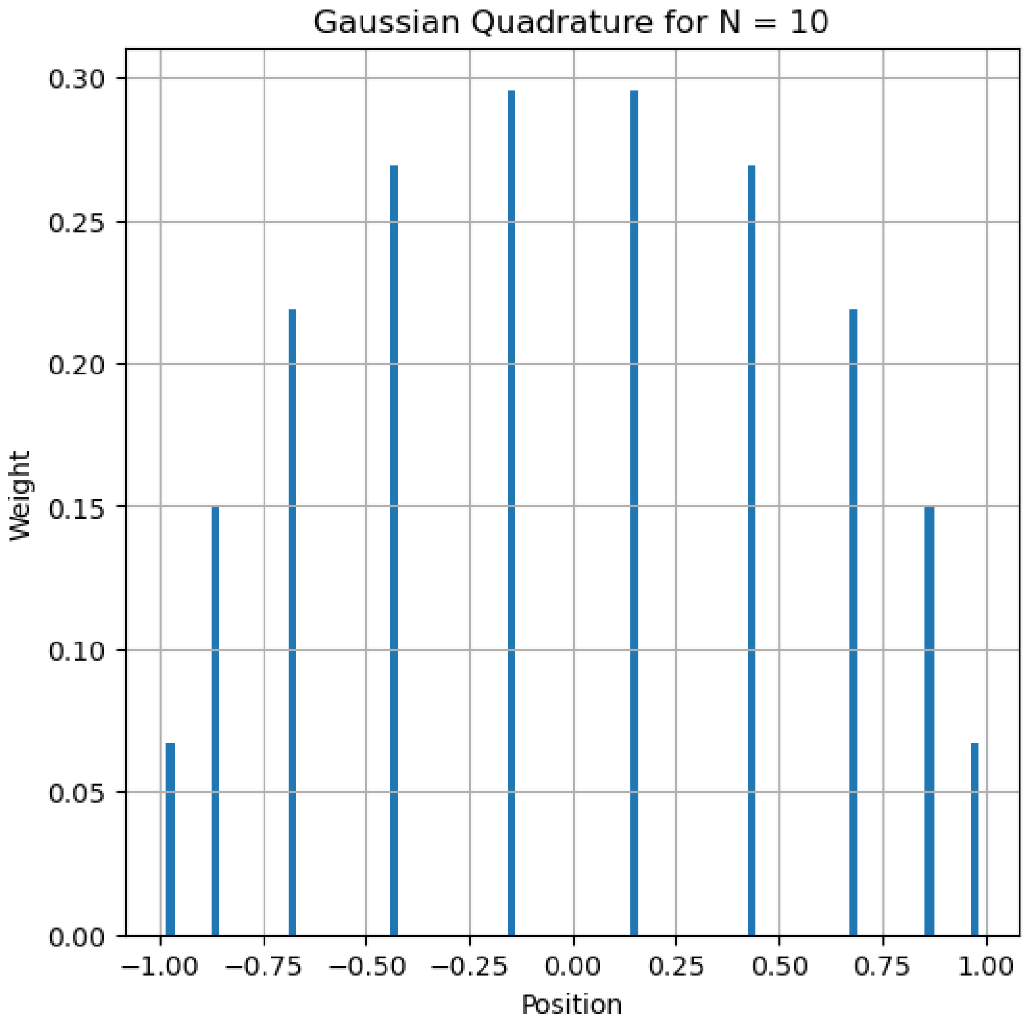
Let's have a look at Newman's gaussxw and gausswab code. Recall: * Use `gaussxw` for integration limits from −1 to +1, * Use `gaussxwab` for integration limits from a to b. * Since the calculation of weights and points is expensive, use `gaussxw.py` if you are going to change the limits repeatedly (see textbook pages 167-168 for how).

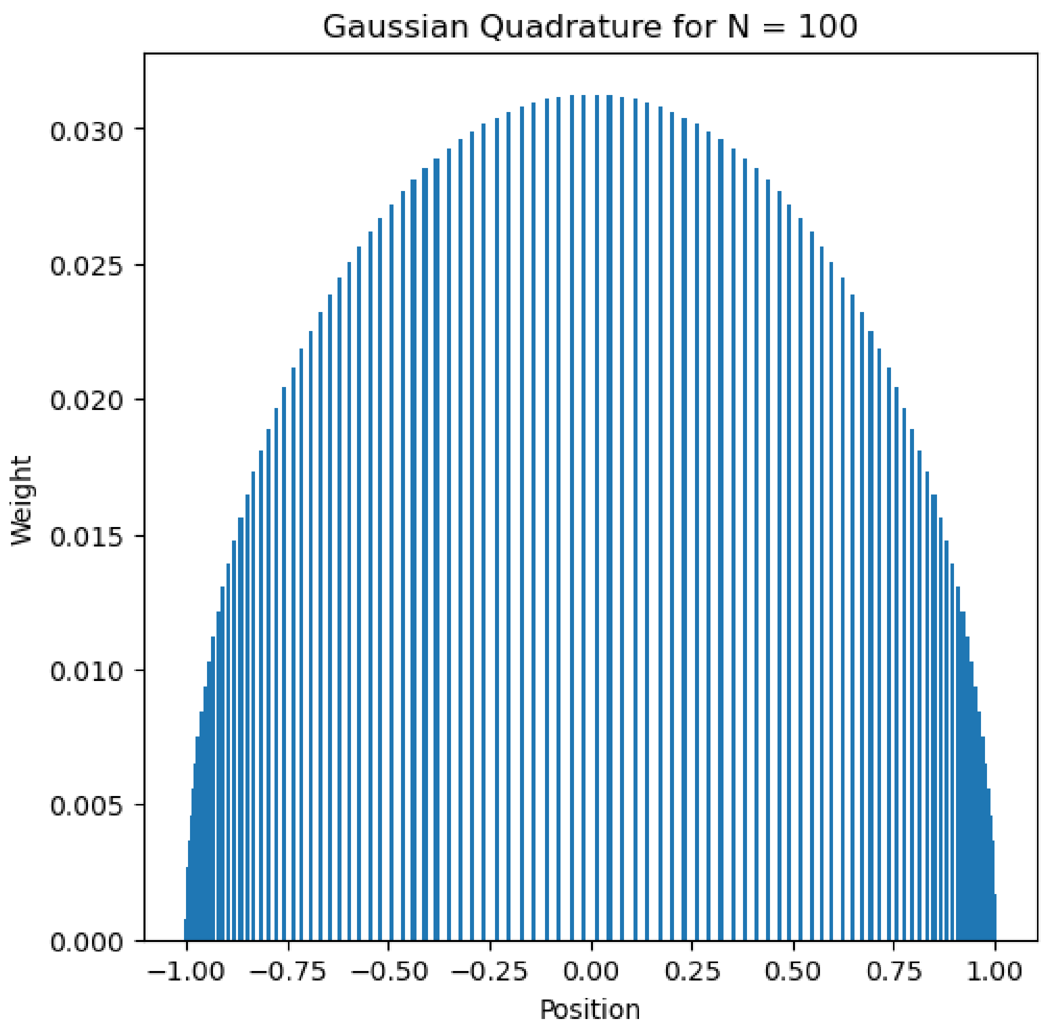
[23]:
```
# %load gaussxw
from pylab import *
def gaussxw(N):

    # Initial approximation to roots of the Legendre polynomial
    a = linspace(3,4*N-1,N)/(4*N+2)
    x = cos(pi*a+1/(8*N*N*tan(a)))

    # Find roots using Newton's method
    epsilon = 1e-15
    delta = 1.0
    while delta>epsilon:
        p0 = ones(N,float)
        p1 = copy(x)
        for k in range(1,N):
            p0,p1 = p1,((2*k+1)*x*p1-k*p0)/(k+1)
        dp = (N+1)*(p0-x*p1)/(1-x*x)
        dx = p1/dp
        x -= dx
        delta = max(abs(dx))

    # Calculate the weights
    w = 2*(N+1)*(N+1)/(N*N*(1-x*x)*dp*dp)

    return x,w

def gaussxwab(N,a,b):
    x,w = gaussxw(N)
    return 0.5*(b-a)*x+0.5*(b+a),0.5*(b-a)*w
```

Reproduce Newman's Figure 5.4 with the help of the above code.

[44]:
```
import numpy as np
import matplotlib.pyplot as plt

# Parameters for Gaussian quadrature, N = 10
N_1 = 10
a = -1.0
b = 1.0

x, w = gaussxwab(N_1, a, b)
```

```python
# Plot for N = 10
plt.figure(figsize=(6, 6))
plt.bar(x, w, width=0.02)
plt.xlabel('Position')
plt.ylabel('Weight')
plt.title('Gaussian Quadrature for N = 10')
plt.grid(True)
plt.show()

# Parameters for Gaussian quadrature, N = 100
N_2 = 100
a = -1.0
b = 1.0

x, w = gaussxwab(N_2, a, b)

# Plot for N = 100
plt.figure(figsize=(6, 6))
plt.bar(x, w, width=0.01)
plt.xlabel('Position')
plt.ylabel('Weight')
plt.title('Gaussian Quadrature for N = 100')
plt.show()
```

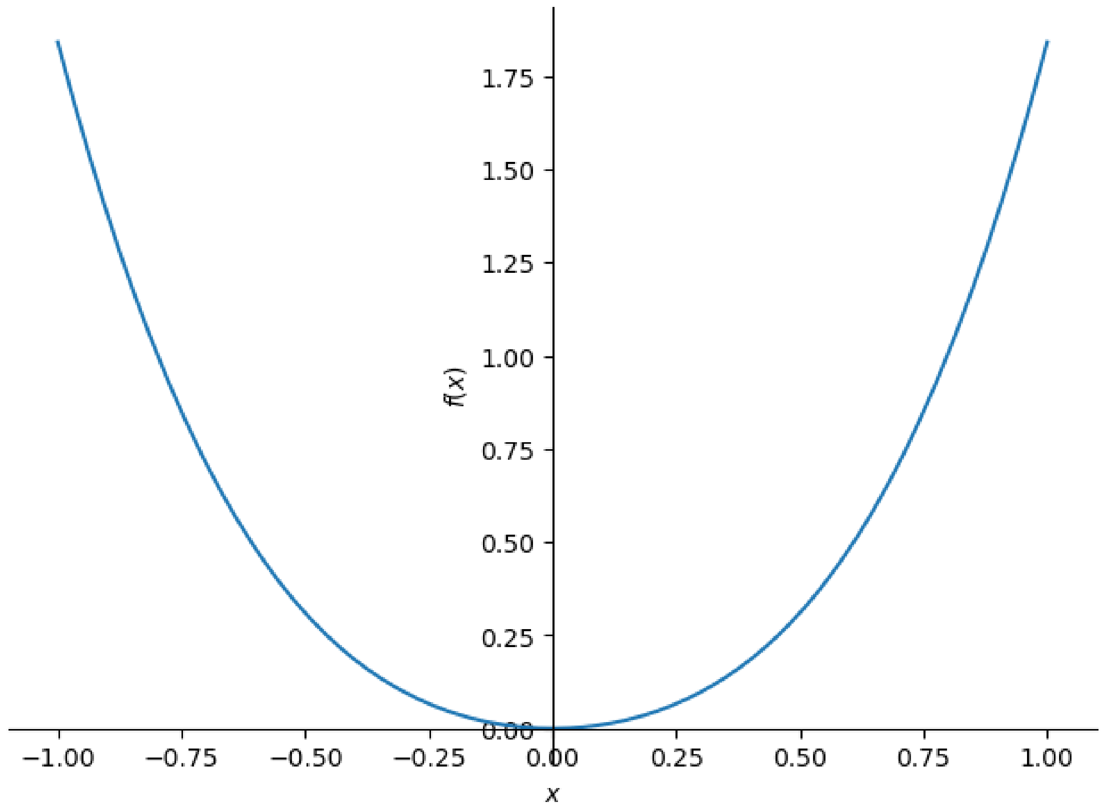

Gaussian Quadrature for N = 10
Weight
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
Position



0.3 Example 3

Compute $\int_{-1}^{1} [x^4 + \sin(x^2)]\, dx$: first symbolically, and then using Gaussian quadrature.

[25]:
```
# first, plot the function using symbolic math package
from sympy import *
init_printing()
x = symbols('x', real=True)
f = x**4 + sin(x**2)
plotting.plot(f, (x, -1, 1))
```



[25]:
```
<sympy.plotting.plot.Plot at 0x7f0d38f4c390>
```

[26]:
```
# integrate using symbolic math package
res=integrate(f, (x, -1, +1))
res
```

[26]:

$$\frac{2}{5} + \frac{3\sqrt{2}\sqrt{\pi}S\left(\frac{\sqrt{2}}{\sqrt{\pi}}\right)\Gamma\left(\frac{3}{4}\right)}{4\Gamma\left(\frac{7}{4}\right)}$$

[27]:
```
# Numerical evaluation
N(res)
```

[27]:

1.02053660344676

[46]:
```
# Define function
N=100
def f(x):
    return x**4 + np.sin(x**2)

x, w = gaussxw(N)
```

```python
# Numerical integration
integral_numerical = np.sum(w * f(x))
integral_numerical
```

[46]: 1.02053660344676