COSC 343: homework 2

Micah Sherry

March 30, 2024

1 four point Gaussian quadrature rule on [-1, 1]

1.1 defining four point Gaussian quadrature

Define a four point Gaussian Quadrature rule on the interval [-1,1].
To define the four point Gaussian quadrature rule I solved (using sage) the system:

$$w_1 + w_2 + w_3 + w_4 = 2$$

$$w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 = 0$$
$$w_1x_1^2 + w_2x_2^2 + w_3x_3^2 + w_4x_4^2 = \frac{2}{3}$$
$$w_1x_1^3 + w_2x_2^3 + w_3x_3^3 + w_4x_4^3 = 0$$
$$w_1x_1^4 + w_2x_2^4 + w_3x_3^4 + w_4x_4^4 = \frac{2}{5}$$
$$w_1x_1^5 + w_2x_2^5 + w_3x_3^5 + w_4x_4^5 = 0$$
$$w_1x_1^6 + w_2x_2^6 + w_3x_3^6 + w_4x_4^6 = \frac{2}{7}$$
$$w_1x_1^7 + w_2x_2^7 + w_3x_3^7 + w_4x_4^7 = 0$$

I got the solution (rounded to 4 decimal places for readability):

$$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \approx \begin{pmatrix} 0.3479 \\ 0.6521 \\ 0.6521 \\ 0.3479 \\ -0.8611 \\ -0.34 \\ 0.34 \\ 0.8611 \end{pmatrix}$$

To use these points and weights to integrate numerically use the formula:

$$\int_{-1}^{1} f(x)dx \approx \sum_{i=1}^{4} w_i \times f(x_i)$$

Code:

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

def fourPointGaussianQuad(f):
    """
    the points and weighs in this method come from solving the system :
    w1*x1^0 + w2*x2^0 + w3*x3^0 + w4*x4^0 = 2
```

```
    w1*x1^1 + w2*x2^1 + w3*x3^1 + w4*x4^1 = 0
    w1*x1^2 + w2*x2^2 + w3*x3^2 + w4*x4^2 = 2/3
    w1*x1^3 + w2*x2^3 + w3*x3^3 + w4*x4^3 = 0
    w1*x1^4 + w2*x2^4 + w3*x3^4 + w4*x4^4 = 2/5
    w1*x1^5 + w2*x2^5 + w3*x3^5 + w4*x4^5 = 0
    w1*x1^6 + w2*x2^6 + w3*x3^6 + w4*x4^6 = 2/7
    w1*x1^7 + w2*x2^7 + w3*x3^7 + w4*x4^7 = 0
    """
    w = [0.347854845137454,    0.652145154862546,
         0.652145154862546, 0.347854845137454]

    x = [-0.861136311594053, -0.339981043584856,
         0.339981043584856, 0.861136311594053]
    area = 0
    for i in range(len(w)):
        area += w[i]*f(x[i])
    return area
```

1.2 highest exactly integrable degree polynomial

What is the highest degree polynomial function that your rule can integrate exactly?
Gaussian quadrature with n points should be able to integrate all polynomials of degree $2n-1$ exactly. This comes from the fact that Gaussian quadrature with n point has $2n$ unknown variables and needs $2n$ equations to solve and the last equation in the system

$$w_1x_1^{2n-1} + ... + w_nx_n^{2n-1} = \int_{-1}^{1} x^{2n-1}dx$$

So a four point Gaussian quadrature routine should be able to integrate all polynomials of degree 7 or less.

1.3 test of four point Gaussian quadrature on [-1,1]

Write code that shows your four point Gaussian Quadrature rule on [-1,1] quadrature rule will in fact integrate polynomial of this degree exactly.
To verify my code works i will be using the polynomial

$$f_1(x) = 2x^7 - 3x^6 + 5x^5 - 4x^4 + x^3 - 2x^2 + 3x - 1$$

and will be using the polynomial

$$f_2(x) = x^6$$

Code:

```
import numpy as np
import matplotlib.pyplot as plt
from GaussianQuadrature import fourPointGaussianQuad

def f1(x):
    return (2*x**7 - 3*x**6 + 5*x**5 -
            4*x**4 + x**3 - 2*x**2 + 3*x - 1)

def F1(x):
    # antiderivative of f1(x)
    return  (2/8*x**8 - 3/7*x**7 + 5/6*x**6 - 4/5*x**5 +
             1/4*x**4 - 2/3*x**3 + 3/2*x**2 - x)
def f2(x):
    return x**6
```

```python
def F2(x):
    # antiderivative of f2(x)
    return x**7/7

if __name__=="__main__":
    trueVal_f1 = F1(1) - F1(-1)
    print(trueVal_f1)
    print(fourPointGaussianQuad(f1))

    trueVal_f2 = F2(1) - F2(-1)
    print(trueVal_f2)
    print(fourPointGaussianQuad(f2))
```

1.3.1 output:

For the first polynomial's true value I got:

$$\int_{-1}^{1} f_1(x)dx = F_1(1) - F_1(-1) = -5.7904761904761894$$

and using Gaussian quadrature I got a value of -5.790476190476198

For the second polynomial's true value I got:

$$\int_{-1}^{1} f_2(x)dx = F_2(1) - F_2(-1) = 0.2857142857142857$$

and using Gaussian quadrature I got a value of 0.2857142857142867