Numerical Methods HW1 - Notes and Plots

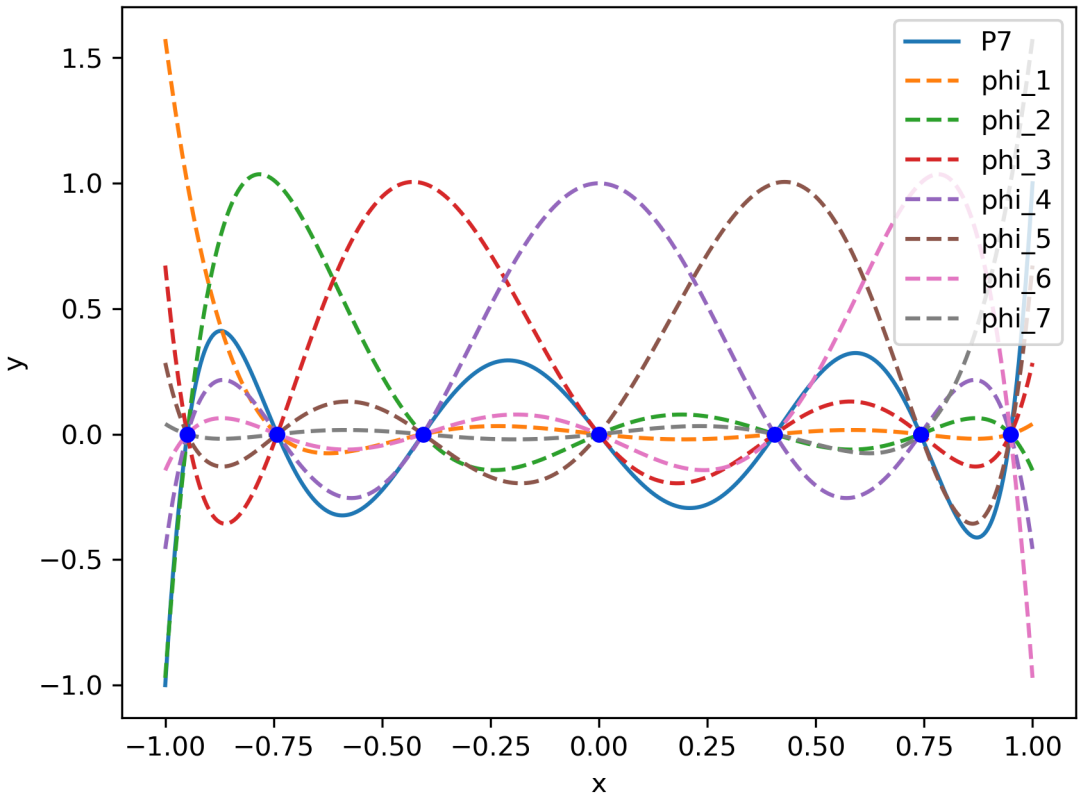
Teresa del Aguila Ferrandis

1 Find the Sample Points x_k

- Gaussian quadrature in a nutshell:
- In this first part, we numerically find all the roots for a given Legendre polynomial. $P_n(x)$ has n distinct roots. We test for the 0 to 7th order Legendre Polynomials.
- To call an nth order Legendre polynomial I will use legendre function from scipy.special library and to find the derivative I will simply do polynomial.der().
- Results for Legendre polynomial roots obtained:
  - **P(0):** none
  - **P(1):** 0
  - **P(2):** -0.5773502691896257, 0.5773502691896257
  - **P(3):** -0.7745966692414835, -1.131318380176525e-19, 0.7745966692414834
  - **P(4):** -0.8611363115940527, -0.33998104358485626, 0.33998104358485626, 0.8611363115940526
  - **P(5):** -0.906179845938664, -0.5384693101056831, 1.9721522630525295e-31, 0.5384693101056831, 0.906179845938664
  - **P(6):** -0.932469514203152, -0.6612093864662646, -0.2386191860831969, 0.23861918608319693, 0.6612093864662646, 0.932469514203152
  - **P(7):** -0.9491079123427585, -0.7415311855993945, -0.4058451513773972, 4.930380657631324e-31, 0.40584515137739724, 0.7415311855993946, 0.9491079123427585

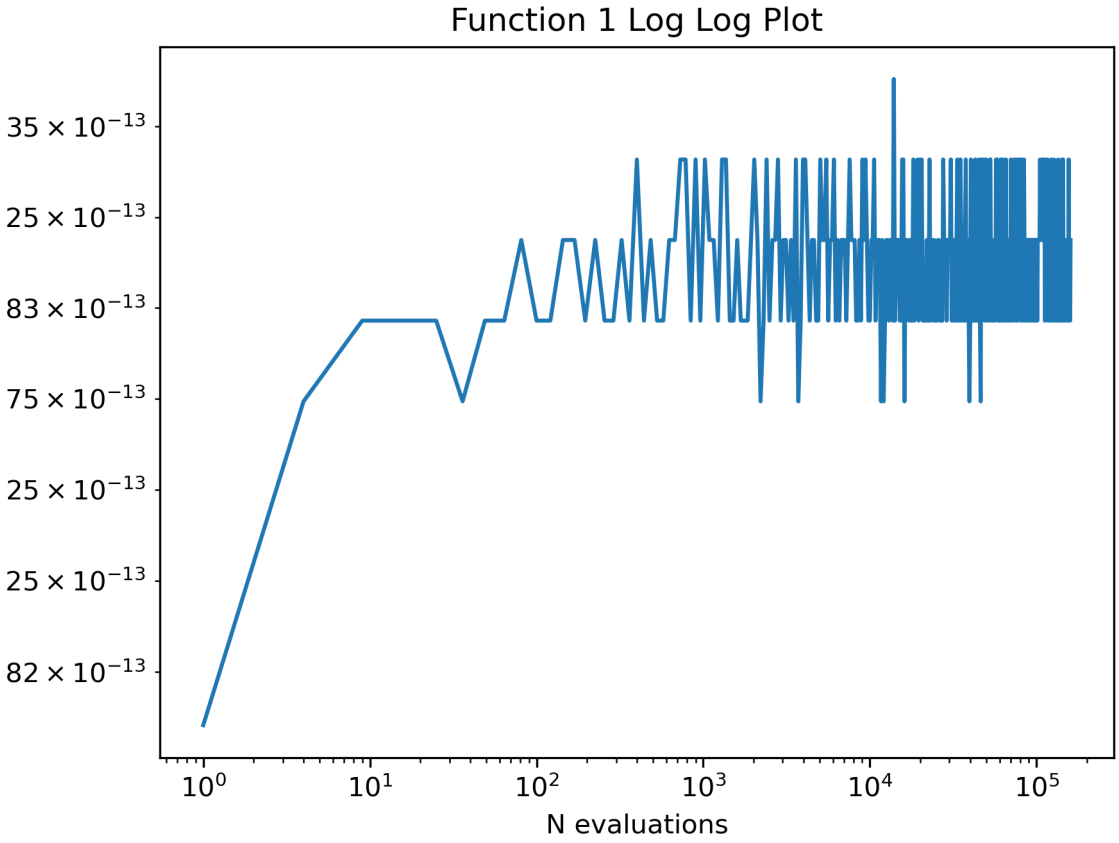
2 Find the weights w_k

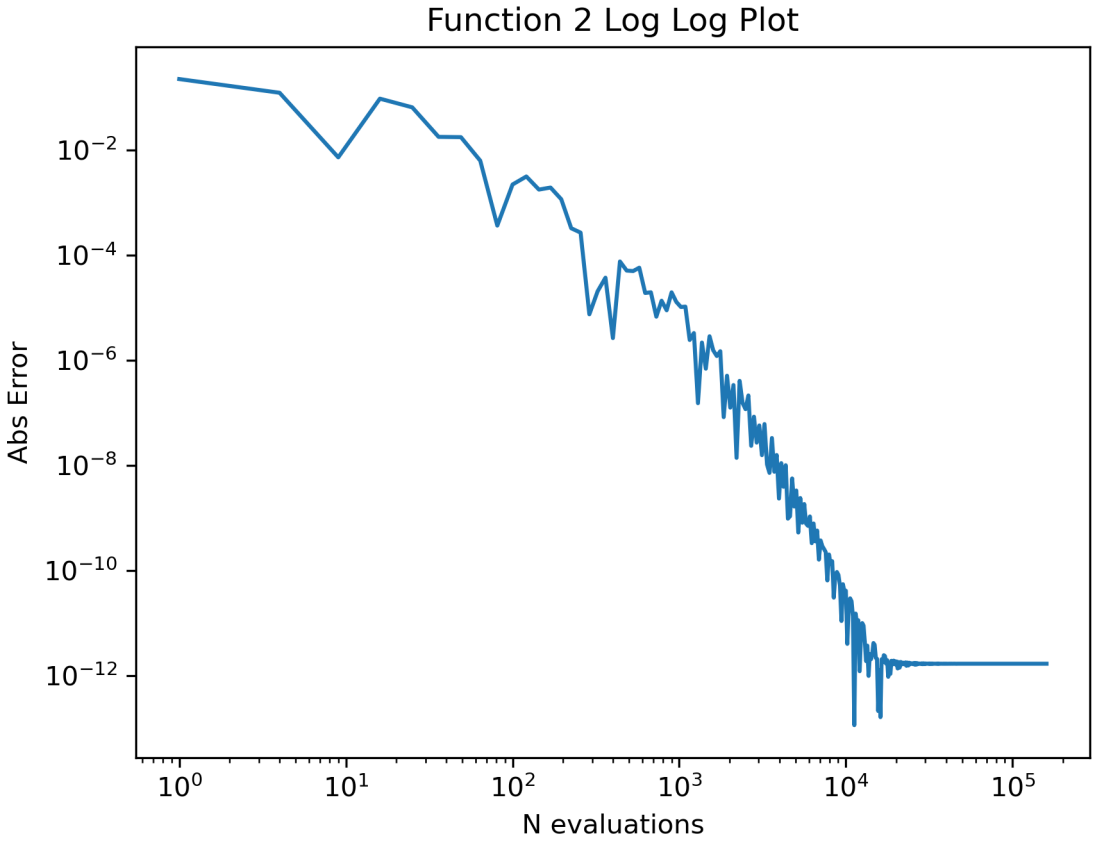
- Gaussian quadrature constructs degree $N-1$ *interpolating polynomial* $\Phi(x)$ for N sample points. These are used to determine the weights w_k.
- We need N *indicator polynomials* $\phi_k(x)$ to determine each $\Phi(x)$. These are degree $N-1$ and satisfy $\phi_m(x_k) = \delta_{mk}$ - i.e. it is 1 at one sample point and 0 elsewhere.
- **task 1:** write function to compute $\phi_k(x)$ for a given set of sample points. Plot $P_N(x)$ and all $\phi_k(x)$ for $N = 7$ and also mark the sample points.
- Plot of all $\phi_k(x)$ for $P_7(x)$ as well as the sample points/roots:

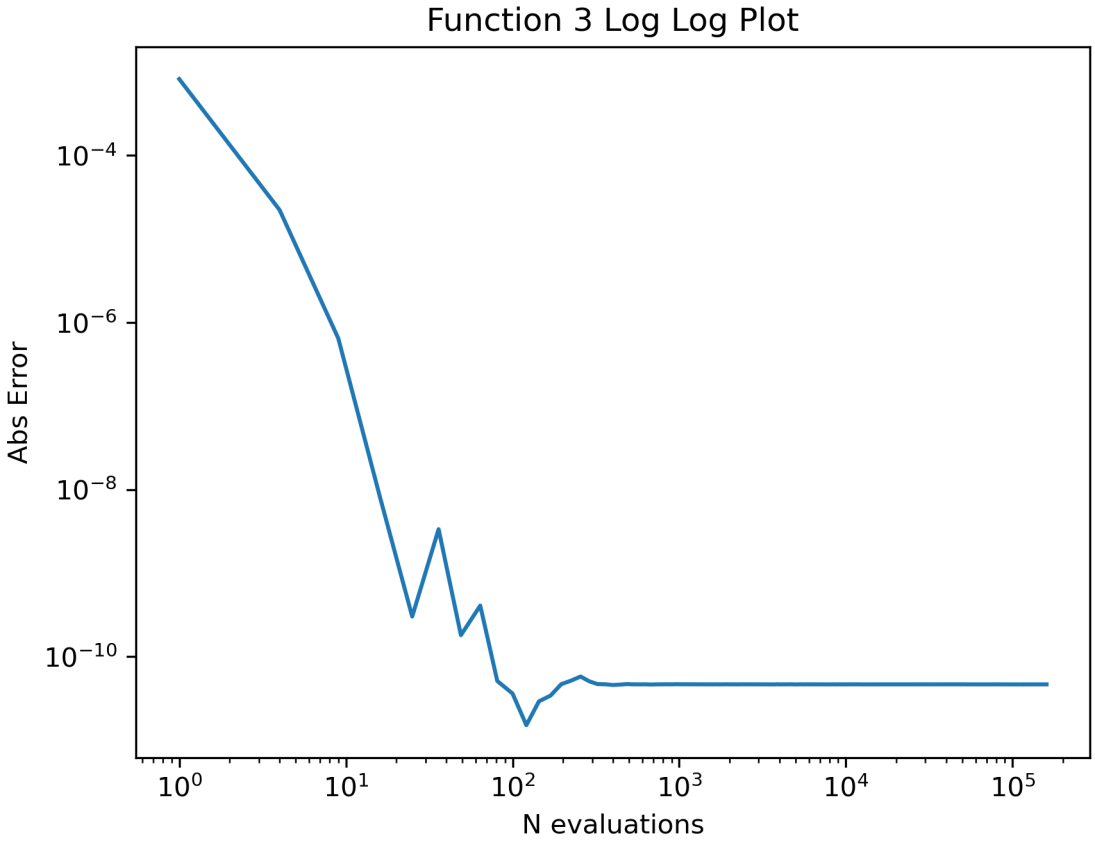


3 Integrating

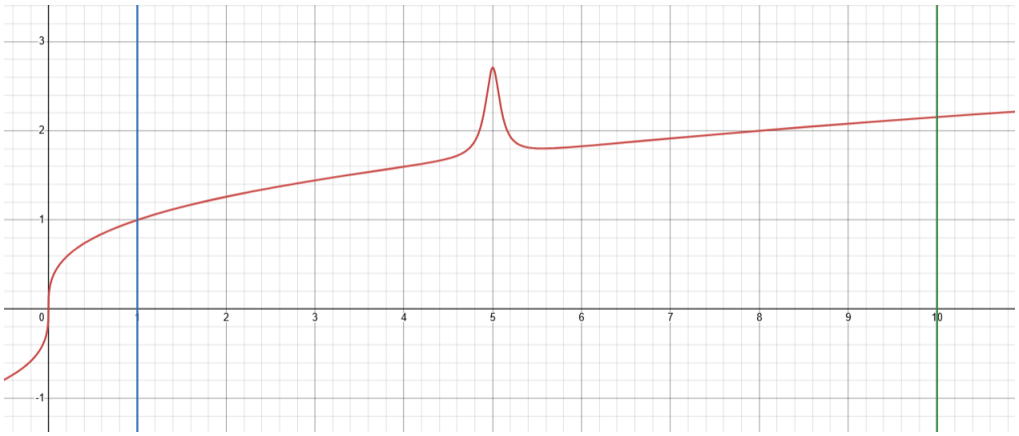
- After implementing the Gaussian Quadrature algorithm we are given specific functions to apply it to and then observe how error changes with increasing number of function evaluations in a log log plot. For the error calculation, I used Mathematica to calculate the true value of the integrals. The results are displayed below.


Function 1 Log Log Plot
35 × 10−13
25 × 10−13
83 × 10−13
75 × 10−13
25 × 10−13
25 × 10−13
82 × 10−13
10^0
10^1
10^2
10^3
10^4
10^5
N evaluations


Function 2 Log Log Plot
Abs Error
10−2
10−4
10−6
10−8
10−10
10−12
10^0
10^1
10^2
10^3
10^4
10^5
N evaluations



- For the second function, we are asked: It takes some evaluations before the schemes start to converge here. Why? What is special about the number of points needed? Looking at the plot of the second function, below, we can see there is a peaked region where the function changes rapidly and is thus difficult to approximate and numerically integrate using Guassian Quadrature. We need enough points to be able to accurately approximate the peak region.



- We are also asked: Some schemes converge at the expected rate, some slower, why might that be? Smoother functions require less function evaluations for the quadrature scheme to converge to a certain error.