Lab Report: Chebyshev Polynomial

Introduction

I'll begin by writing a function *cheby_coeff(f,N)* that takes input f, function, and N, number of coefficients, and return the Chebycheff coefficients. Using the computed coefficients, I would then be able to multiply them with the respected Chevyshev polynomial to get an approximation of f(x). To get the maximum error, compute the absolute difference between the evaluation of the polynomial from the exact function with x = linspace(-1,1,100). In addition for the function f(x) = exp(x), the code computes the Chebyshev polynomial of f(x) over the interval with N large enough so that the error is less than 1e-6. As for the piece function g(x) = sgn(x), I'll be observing the behavior of the error function by computing at N = 2, 3, 4, 5.

Algorithm Method

Part A: To write the function to find Chebycheff coefficients, I start with a for loops for i that goes from 1 to N+1 (since Matlab begins at 1 we need N+1 to make up for index 0) and then implement the transformation of $x = cos(\theta)$. For the first a_0, the computation would be different than other since it equals to the product of 1/pi and the integration of f(x) from 0 to pi. For i = 1,...,N, a_i would then be the product of 2/pi and the integration of $f(x)cos(i\theta)$ from 0 to pi. I use the built-in integration to solve for these coefficients. The Chebycheff coefficients for f(x) = exp(x) are a = [1.2661, 1.1303, 0.2715, 0.0443, 0.0055, 0.0005, 0.0000, 0.0000]. In order to get the Chebyshev polynomial approximation, I created a for loop in the main script and solve for $T_i = cos(icos^{-1}(x))$ and multiply it with its respected a_i for i = 0,...N and add them all up to get the approximation. Evaluating x on the Chebyshev polynomial with N = 7 and compared them to the exact f(x) would lead to finding the maximum error of 2.1083e-07, which is smaller than 1e-6.

Part B: To calculate the coefficients for g(x) = sgn(x), I run the *cheby_coeff(g,5)* (N = 5 since I want to check up to 5 coefficients) and get the output b = [0, 1.2732, 0, -0.4244, 0, 0.2546]. Using the same algorithm as above with the addition of changing N values, I get the maximum error of each N to be *max_error_2* = [0.9871, 0.9743, 0.9743, 0.9614, 0.9357, 0.8973]. While the maximum error seems be decreasing as N increases, it is unlikely that I can make the error as small as I like by increasing N. This is because the function itself is not smooth so using Chebyshev polynomial to approximate it would never be exact and thus the error can't be as small as I would like it to be.

Conclusion

Chebyshev coefficients is an efficient way to approximate a function as observe from Part A. As it takes only a few coefficients to get to the error one would like with an N large enough. However, this accuracy also depends on the function one would like to approximate. If the function is not smooth as the example in Part B, it would be difficult to get a small error even with a larger N.