Assignment-5

Shivanshu Shekhar

December 31, 2023

1 Assignments

1. Write a C program to compute the root using this formula and its complex form for a range of α values (given α_{start}, α_{end} and N). Note that the range of values will be over many orders of magnitude. So the samples should be spaced geometrically.

2. Also calculate the roots using "float" precision. For this, you have to have vari- ables declared as float.

3. Also calculate the roots using "float" precision with the more accurate formula.

4. Determine the error between the expressions in parts 2 and 3 (taking part 1 as the exact answer) and plot the magnitude of error vs α in a log log plot. Discuss.

5. Write a Python function that uses the built in routines to compute the sum. This is our exact solution.

6. Code the series with a forward series computing the function using the recursion as you update the sum. Obtain the error and plot the error vs n for x = 1.5 and for x = 15. Explain the difference.

7. Code the series with a backward series assuming that J60 (x) = 1 and J61 (x) = 0. Normalize the value of J0 (x) to 1. Determine the error in this code for x = 1.5 and for x = 15.

8. Implement Clenshaw algorithm in Python for arbitrary coefficients in the recursion.

9. Implement the Chebyshev sum for $\exp\{x\}$ between -1 and 1 using both Clenshaw and using the direct method. Compare the errors.

10. Repeat for Fourier.

2 Clenshaw Summation

The Clenshaw algorithm is a numerical technique used for evaluating a polynomial in a numerically stable and efficient manner, particularly when the polynomial is expressed in a special form known as the Chebyshev series. The Chebyshev series representation of a function $f(x)$ on the interval $[-1, 1]$ is given by:

$$f(x) = \sum_{k=0}^{N} a_k T_k(x),$$

where a_k are the Chebyshev coefficients, and $T_k(x)$ are the Chebyshev polynomials of the first kind. The Chebyshev polynomials are defined as:

$$T_0(x) = 1,$$
$$T_1(x) = x,$$
$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x) \quad \text{for } k \geq 2.$$

The Clenshaw algorithm efficiently evaluates the Chebyshev series at a specific point x using the recurrence relation:

$$b_k = a_k + 2x \cdot b_{k+1} - b_{k+2} \quad \text{for } k = N, N-1, \ldots, 0.$$

Here, b_k are the Clenshaw values. The algorithm iterates through the series coefficients in reverse order, updating the Clenshaw values until the final result is obtained. The final result is given by:

$$f(x) \approx 0.5 \cdot (b_0 - b_2).$$

This process avoids explicitly computing and storing the intermediate values of the Chebyshev polynomials, resulting in a numerically stable and efficient evaluation of the polynomial at the given point x.

3 Solutions

3.1 Errox in Quadratic roots Approximations

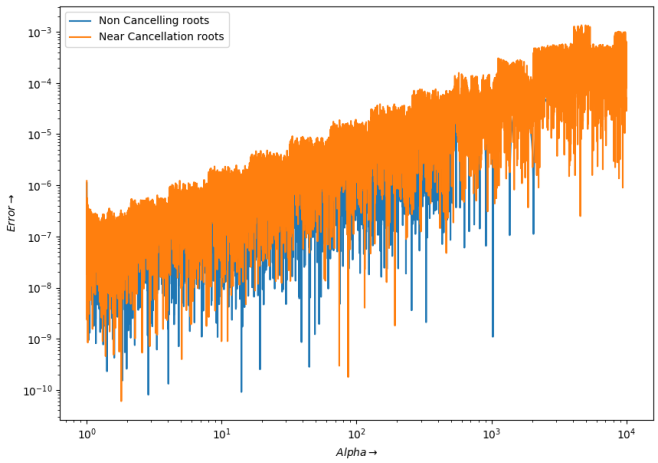

Figure 1: Precision error

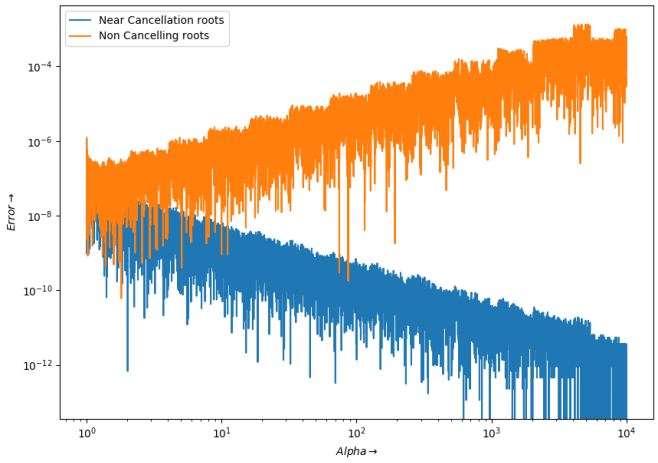

Figure 2: Approximation error

In Figure: 1 we can see that as the magnitude of α is increasing the error of both non and near-cancelling roots are increasing which is expected as in the exact formula we are not optimizing for near-cancelling roots. So, as α's size increases, the truncation error dues to floats also increase. While in Figure

:2 the error for the non-cancelling the error increases like the exact case but the error in near cancellation roots are getting better as alpha increases.

3.2 Estimating the Sum

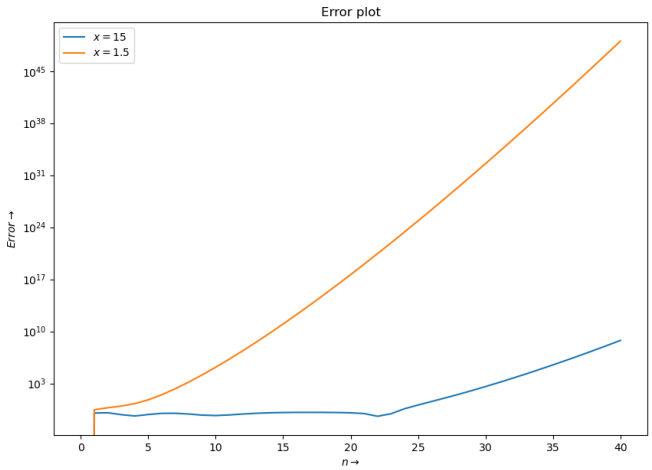

Figure 3: Forward Series error

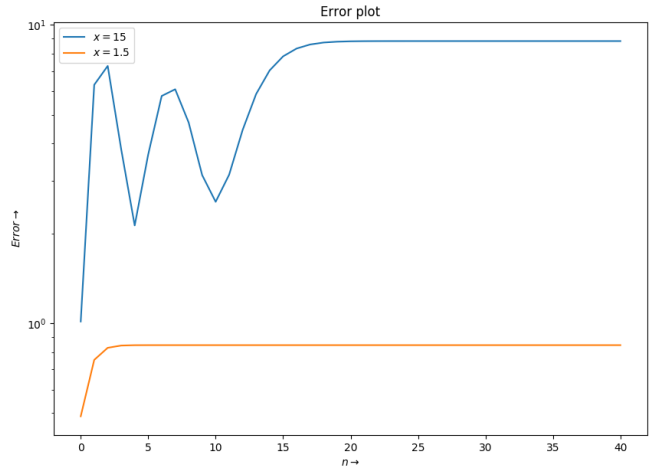

Figure 4: Backward Series error

3.3 Clenshaw Implementation

Below is the code for Clenshaw algorithm for arbitrary coefficients in the recursion.

```
def part3(func, a, n, alpha, beta):
b = np.zeros(n+2)

for i in range(n-1, -1, -1):
    b[i] = a(i) + alpha(i)*b[i+1] + beta(i+1)*b[i+2]

return func(0)*a(0) + func(1)*b[1] + beta(1)*func(0) + b[2]
```

3.4 Using Clenshaw in Chebshev and Fourier

In this section, we compare the estimation accuracy of using raw summation and using Clenshaw recursive summation.

Below are the Clenshaw codes for Fourier and Chebyshev respectively:

```
def fourierClen(a0, an, bn, x):
theta = np.pi*x

d, dd = 0.0, 0.0
n = len(an)

for i in range(n, 0, -1):
    tmp = d
    d = bn[i-1] + 2*np.cos(theta)*d - dd
    dd = tmp

sine_sum = d*np.sin(theta)

d, dd= 0.0, 0.0

for i in range(n, 0, -1):
    tmp = d
    d = an[i-1] + 2*np.cos(theta)*d - dd
    dd = tmp

cosine_sum = 0.5*a0 + np.cos(theta)*d -dd

return cosine_sum + sine_sum

def chebyClen(c, m, a, b, x):
y = (x - 0.5*(b+a))/(0.5*(b-a))
y2 = 2*y

d, dd = 0.0, 0.0
for j in range(m-1, 0, -1):
    tmp = d
    d = y2*d-dd+c[j]
    dd = tmp

return y*d - dd + 0.5*c[0]
```

The Mean error using raw implementation for Chebyshev: 3.6794506401529264e-29 with a standard deviation of 4.968722697693055e-29. While we get a

mean error of 3.465385837950218e-29 and a standard deviation of 4.6298758051334315e-29 when using Clenshaw summation. We have almost the same values for the Fourier series.