Chandler Justice - A02313187

MATH5620: Homework #1
Due: January 22, 2024 at 23:59

1. Write out the details for the accuracy analysis for

$$D_-f(x) = \frac{f(x) - f(x-h)}{h}$$

Compute an expression for the error in terms of h and a constant. What restrictions must be satisfied in order to use this difference.?

Solution: We can utilize the following statement from the textbook to better understand how we can analyze the accuracy of our difference quotient

$$u(x+h) = u(x) + hu'(x) + \frac{1}{2}h^2u''(x) + \frac{1}{6}h^3u'''(x) + O(h^4)$$

We can adapt this function to get a function that computes the error of our approximation by subtracting the derivative of this function from our finite difference. This will allow us to see the error in our approximation.

Recall $D_-f(x)$ has the following Taylor series expansion

$$D_-f(x) = u'(x) + \frac{1}{2}hu''(x) + \frac{1}{6}h^2u'''(x) + \frac{1}{24}h^3u''''(x) + O(h^4)$$

We are using the $O(h^k)$ term to symbolize the remaining unaccounted for terms in the Taylor series. So to compute the error we need to subtract the Taylor series of our exact equation from our approximation. We want

$$|u(x+h) - D_-u(x)| \leq Ch$$

When we compute this difference we get the result

$$\boxed{|u(x+h) - D_-u(x)| = \frac{1}{2}hf''(\zeta) \leq Ch}$$

The restrictions we must set on this error is that h must be representable with the precision we are working with (IE, floats, doubles, etc), and u must be differentiable up to the number of terms we require.

2. Write a code that returns the coefficients for a difference quotient approximating the first derivative of a function at a specified point x given an input array of points.

3. Write a code that will return the coefficients of a derivative of a given order specified at a minimal number of points specified by the user.

Solution: For both questions 2 and 3, I was able to use the same function since my function lets me specify the order of derivative I would like the constants for

```
import numpy as np
import math
def difference_coefficients(k: int, xbar: float, x: np.array) -> np.array:
    # determine size of input points
    n = len(x)

    # initialize the Vandermonde matrix with 1s
    A = np.ones((n, n))
    # subtract xbar from each element and reshape to a row
    xrow = (np.array(x) - xbar).reshape(1, -1)

    # construct the Vandermonde matrix
    for i in range(2, n + 1):
        A[i-1, :] = (xrow ** (i-1)) / math.factorial(i-1)

    # initialize the solution set
    b = np.zeros((n, 1))
    b[k] = 1

    # solve the linear system to determine the coefficients
    c = np.linalg.solve(A, b)
    print(b)
    # return the coefficients as an array
    return c.flatten()
```

4. Write a code that will determine the accuracy of a specified difference quotient. That is, instead of computing the coefficients, input the coefficients and determine the number of equations that should be satisfied.

   Solution: I wrote the following function to validate the coefficients of a difference quotient

```
def check_coefficients(coefficients: np.array, step: float, x: float) -> np.array:
    n = len(coefficients)
    x_values = np.array([x + i * step for i in range(n)])
    Vandr = np.vander(x_values, increasing=True)
    res = np.linalg.solve(Vandr, x_values)
    return res # should return array of all 0s and 1 one.
```

   Looking at the results of this validation, it looks like there is exactly 1 equation that is satisfied.