Numerical Methods Lab 4

Hermite and Newton's Divided Difference Interpolation [C02]

i. Open the colab file shared in BUX.

ii. Create a copy of that shared file in your drive.

iii. Rename the colab filename using the format **Name-ID-Lab Section**

Lab Introduction

Part 1: Hermite Interpolation

For the case of Hermite Interpolation, we look for a polynomial that matches both f′(xi) and f(xi) at the nodes $x_i=x_0,\ldots,x_n$. Say you have n+1 data points, $(x_0,y_0),(x_1,y_1),x_2,y_2),\ldots,(x_n,y_n)$ and you happen to know the first-order derivative at all of these points, namely, $(x_0,y'_0),(x_1,y'_1),(x_2,y'_2),\ldots,(x_n,y'_n)$. According to hermite interpolation, since there are 2n+2 conditions; n+1 for $f(x_i)$ plus n+1 for $f'(x_i)$; you can fit a polynomial of order 2n+1 .

General form of a $2n+1$ degree Hermite polynomial:

$$p_{2n+1} = \sum_{k=0}^{n} \left(f(x_k)h_k(x) + f'(x_k)\hat{h}_k(x) \right),$$

where h_k and $\hat{h}_k$ are defined using Lagrange basis functions by the following equations:

$$h_k(x) = (1 - 2(x - x_k)l'_k(x_k))l^2_k(x_k),$$

and

$$\hat{h}_k(x) = (x - x_k)l^2_k(x_k),$$

where the Lagrange basis function being:

$$l_k(x) = \prod_{j=0, j\neq k}^{n} \frac{x - x_j}{x_k - x_j}.$$

Part 2: Newton's Divided Difference Interpolation

Newton form of a n degree polynomial:

$$p_n(x) = \sum_{k=0}^{n} a_k n_k(x),$$

where the basis is:

$$n_k(x) = \prod_{j=0}^{k-1} (x - x_j),$$
$$n_0(x) = 1,$$

and the coefficients are:

$$a_k = f[x_0, x_1, \ldots, x_k],$$

where the notation $f[x_0, x_1, \ldots, x_k]$ denotes the divided difference.

By expanding the Newton form, we get:

$$p(x) = f[x_0] + (x - x_0)f[x_0, x_1] + (x - x_0)(x - x_1)f[x_0, x_1, x_2] + \cdots + (x - x_0)(x - x_1)\ldots(x - x_{k-1})f[x_0, x_1, \ldots, x_k]$$

[Task 1] – 4 marks

Function l(k, x) has already been defined for you.

You have to implement the functions: **h(k, x) and h_hat(k, x) and hermit(x, y, y_prime)** First two methods implement the Hermit Basis to be used for interpolation using Hermite Polynomials and third method calculates the Hermite polynomial from a set of given nodes and their corresponding derivatives.

You will have to remove the “raise NotImplementedError()”.

[Task 2] – 3 marks

1. You have to implement the **calc_div_diff(x,y)** function which takes input x and y, and calculates all the divided differences. You may use the lambda function difference() inside the calc_div_diff(x,y) function to calculate the divided differences.
2. You have to implement the **__call__()** function which takes an input x, and calculates y using all the difference coefficients. x can be a single value or a numpy. In this case, it is a numpy array. You will have to remove the “raise NotImplementedError()” .

[Task 3]- 1.5 marks

Problem related Newton’s Divided Difference interpolation:

Suppose, you have three nodes (−1, 3.5), (0, 1.2), (1, 2.8). Using Newton's

Divided Difference method, print out the value of the interpolating polynomial at x = 0.5.

You have to solve the given problem using **Newtons_Divided_Differences class**.

[Task 4]- 1.5 marks

Problem related Hermite interpolation:

Suppose, consider the following data set:

x	*f(x)*	*f'(x)*
0	2	1
1	4	-1
3	5	-2

Using Hermit basis, print out the interpolating polynomial and find the value at

x = [0.75,1.65,2.50].

You have to solve the given problem using **hermit function**.