MA 322 (2021) Scientific Computing Lab Lab 03

**Name:** Udandarao Sai Sandeep

**Roll Number:** 180123063

**Dept.:** Mathematics and Computing

**Note:** For all questions, the values have been rounded off till the **6th** digit.

**Q1.** The Newton **forward-difference** formula was used to estimate the specified value using interpolating polynomials of **degree 1, degree 2** and **degree 3**.

1. The function is to be estimated at **x = 0.43**. For each case, the initial set of nodes have been chosen in such a manner such that value at which the function is to be estimated lies between the node values (preferably in the **forward** region). The results are as follows:

**Part1**

Nodes Used: **[0.25, 0.5]**

Using Interpolating polynomial of **degree 1**, estimate at x = 0.43: **2.418803**

Nodes Used: **[0.25, 0.5, 0.75]**

Using Interpolating polynomial of **degree 2**, estimate at x = 0.43: **2.348863**

Nodes Used: **[0, 0.25, 0.5, 0.75]**

Using Interpolating polynomial of **degree 3**, estimate at x = 0.43: **2.360605**

1. The function is to be estimated at **x = 0.18**. For each case, the initial set of nodes have been chosen in such a manner such that value at which the function is to be estimated lies between the node values (preferably in the **forward** region). The results are as follows:

**Part2**

Nodes Used: **[0.1, 0.2]**

Using Interpolating polynomial of **degree 1**, estimate at x = 0.18: **-0.506648**

Nodes Used: **[0.1, 0.2, 0.3]**

Using Interpolating polynomial of **degree 2**, estimate at x = 0.18: **-0.508050**

Nodes Used: **[0.1, 0.2, 0.3, 0.4]**

Using Interpolating polynomial of **degree 3**, estimate at x = 0.18: **-0.508143**

**Q2.** The Newton **backward-difference** formula was used to estimate the specified value using interpolating polynomials of **degree 1, degree 2** and **degree 3**.

1. The function is to be estimated at **x = -1/3**. For each case, the initial set of nodes have been chosen in such a manner such that value at which the function is to be estimated lies between the node values (preferably in the **backward** region). The results are as follows:

**Part1**

Nodes Used: **[-0.5, -0.25]**

Using Interpolating polynomial of **degree 1**, estimate at x = -1/3: **0.215042**

Nodes Used: **[-0.75, -0.5, -0.25]**

Using Interpolating polynomial of **degree 2**, estimate at x = -1/3: **0.180306**

Nodes Used: **[-0.75, -0.5, -0.25, 0]**

Using Interpolating polynomial of **degree 3**, estimate at x = -1/3: **0.174519**

1. The function is to be estimated at **x = 0.25**. For each case, the initial set of nodes have been chosen in such a manner such that value at which the function is to be estimated lies between the node values (preferably in the **backward** region). The results are as follows:

**Part2**

Nodes Used: **[0.2, 0.3]**

Using Interpolating polynomial of **degree 1**, estimate at x = 0.25: **-0.138693**

Nodes Used: **[0.1, 0.2, 0.3]**

Using Interpolating polynomial of **degree 2**, estimate at x = 0.25: **-0.132952**

Nodes Used: **[0.1, 0.2, 0.3, 0.4]**

Using Interpolating polynomial of **degree 3**, estimate at x = 0.25: **-0.132775**

**Q3.**

This question was solved using **SymPy** module of python (used to evaluate algebraic expressions in Python).

Here, P(x) is a **fourth-degree** polynomial. Hence, using **Newton’s forward Interpolation Formula**, the **polynomial P(x)** can be constructed.

where

Since **h = 1** and **x0 = 0**, **u = x**.

So, we have:

Putting in the known values of forward differences, we obtain the following equation:

Now, we are to evaluate at **x = 10**.

Since we know that is **independent** of the **constant** and the **linear** terms of P(x), we therefore do not require the values of and to calculate at x = 10.

So,

So,

The value of 2P(10) = **1140**

**Q4.**

(Actual Value)

1. Here, we **directly** approximate **g(x**) at x = 0.25 using **Newton Forward difference formula**. Following similar methods as in **Q1**(for **degree 3**), we obtain the following results:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| g(x) | 9.9833 | 4.9667 | 3.2836 | 2.4339 | 1.9177 |

**Part1**

Using direct g(x), estimate at x = 0.25: **3.911598**

Error b/w actual and estimated value = **0.04686491057236619**

1. Here, we approximate **xg(x)** at x = 0.25 first using Newton Forward difference formula. Then, we divide the estimated value of **xg(x)** by **0.25** to obtain estimated value of **g(x) at**

**x = 0.25.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| xg(x) | 0.99833 | 0.99334 | 0.98508 | 0.97356 | 0.95885 |

**Part2**

Using xg(x), estimate at x = 0.25: **3.958478**

Error b/w actual and estimated value = **1.4776927633519676e-05**

1. It can be seen that the error between the actual and the estimated value is **very less** in case of **part (ii)**. This can be attributed to the fact that in **part (i)**, the forward differences are **highly fluctuating** andare **significantly high**. In **part** (ii), the forward differences are very **small**, and tend to decrease with further iterations, and therefore in this case, the estimation is more accurate.

Also, if look at the **Taylor expansion** of g(x) and xg(x),



In the Taylor expansion of , we can see that the Taylor expansion involves the term . With small changes in x (within the range **[0,0.5]**), the term changes very **rapidly**. This makes the **estimation of g(x) very error prone** (in part (i)).

**Q5.**

The polynomials **P(x)** and **Q(x)** are as follows:

**P(x) = 3 − 2(x + 1) + 0(x + 1) (x) + (x + 1) (x) (x − 1)**

**Q(x) = −1 + 4 (x + 2) – 3 (x + 2) (x + 1) + (x + 2) (x + 1) (x)**

1. To show that both P(x) and Q(x) interpolate the given data, we will evaluate the values of the polynomials at **x = -2, -1, 0, 1, 2** and show that the value at those nodes match the functional values. The calculated values are as follows:

**P (-2) = -1**

**P (-1) = 3**

**P (0) = 1**

**P (1) = -1**

**P (2) = 3**

**Q (-2) = -1**

**Q (-1) = 3**

**Q (0) = 1**

**Q (1) = -1**

**Q (2) = 3**

Since the value of the polynomials at the specific nodes match the functional data (f(x)), both the cubic polynomials P(x) and Q(x) **interpolate** the given data.

1. Simplifying **P(x)** and **Q(x)**, we obtain:

Since both polynomials simplify to be the **same** polynomial, the uniqueness property of interpolating polynomials is **not violated**.

**Q6.**

Since all the 4th order forward differences are 1, all the **5th order differences will be 0**. Hence, the polynomial is of **4th degree**, and this polynomial can be constructed using Newton’s forward interpolation formula by interpolating at the given nodes, along with using the fact that all 4th order forward differences are 1. The constructed polynomial is as follows:

**Hence, the coefficient of the x3 term is**

The coefficient of x3 is **-11/12**

**Q7.**

Given the following polynomial:



According to the question, the above polynomial is the **interpolating polynomial for f(x)** at the nodes **x0 = 0, x1 = 0.25, x2 = 0.5 and x3 = 0.75**.

We know that if **y** is a **node**, then **P(y) = f(y)**.

Since **x =** **0.75** is a node, hence **P(0.75) = f(0.75)**.

So, **f(0.75) = P(0.75) = 6**

**f(0.75) = 6.000000**