Heaven's Light is Our Guide

Rajshahi University of Engineering and Technology

Course Code
ECE 2214

Course Title
Numerical Methods and Discrete Mathematics Sessional

Experiment Date: December 2, 2023,
Submission Date: December 18, 2023

Lab Report 11, 12 & 13: Implementing Taylor Series & Interpolation(Newton's Forward & Newton's Backward Method) in MATLAB

Submitted to	**Submitted by**
Md. Omaer Faruq Goni	Md. Tajim An Noor
Lecturer	Roll: 2010025
Dept of ECE, Ruet	

Implementing & Plotting Taylor Series for Sine in MATLAB

Introduction

Taylor Series

The Taylor series is a way to represent a function as an infinite sum of terms calculated from the values of the function's derivatives at a specific point.

For a function $f(x)$ that is infinitely differentiable at a point a, the Taylor series expansion around a is given by:

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \dots$$

For sine function, the taylor series is expanded as:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

Tools Used

- MATLAB R2021a - for writing and running code.
- MacTeX -LaTeXcompiler.
- VS Code with LaTeXworkshop extension as a text editor.

Process

Code for Taylor sine series:

```
close all;
clear all;
clc;
syms x;
f =input('Enter the function : ');

theta = -pi:pi/20:pi;

g = f;

%hold on;
y = input('Enter the value for approximation: ');
a = input('Enter the starting point: ');

n = input('Enter the number of order: ');

taylor = eval(subs(f,x,a));

for i = 1:n
    f = diff(f,x);
    taylor = taylor + ( eval(subs(f,x,a)) * power((x-a),i) / factorial(i) );
    disp(taylor);
    plot(theta,subs(taylor,x,theta));
    hold on;
    if i == n
    plot(theta,subs(g,x,theta));
    end
end

disp(eval(subs(taylor,x,y)));
```

Output

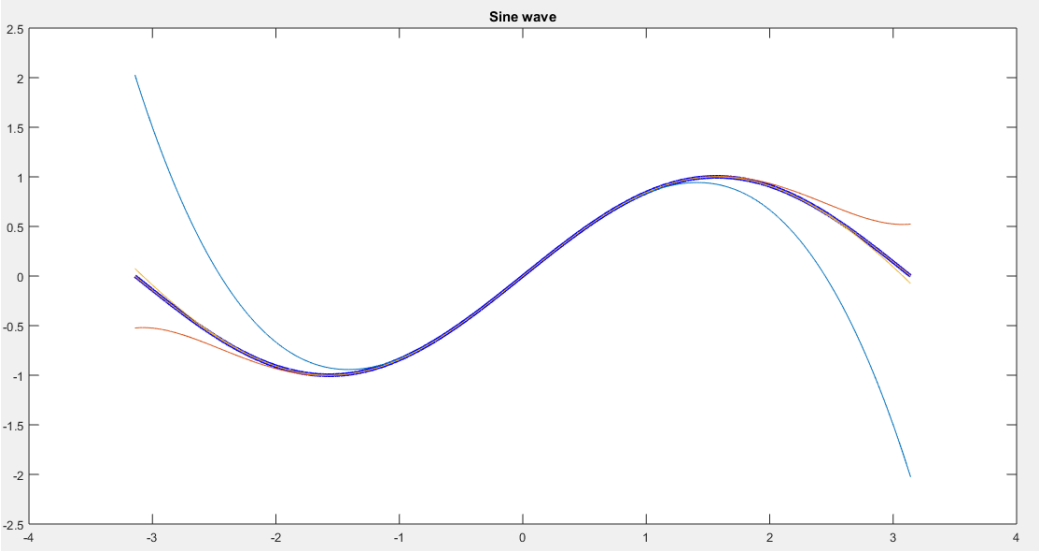

Figure 1: Taylor Sine Series ($-\pi \leq$ x $\leq \pi$)

Implementing Interpolation; Newton's Forward Method in MATLAB

Introduction

Interpolation, Newton's Forward Method

Newton's Forward Interpolation is a numerical technique used for interpolating or estimating the value of a function between given data points. It belongs to the category of polynomial interpolation methods and is particularly useful when dealing with equally spaced data points.

The method involves constructing a polynomial of degree n that passes through $n+1$ equally spaced data points. It approximates the value of the function at an intermediate point within the range of the given data. The general form of the Newton's Forward Interpolation polynomial is:

$$f(x) = P_n(x) = f(x_0) + \Delta y_0 \cdot P_1(u) + \Delta^2 y_0 \cdot P_2(u) + \dots$$

Here, $P_n(x)$ is the polynomial of degree n used for interpolation, $f(x_0)$ represents the value of the function at the starting point x_0, Δy_0 denotes the first forward difference, $\Delta^2 y_0$ signifies the second forward difference, and so on. The symbol u typically represents the normalized difference ratio $\frac{x-x_0}{h}$, where h is the common difference between the data points.

Tools Used

- MATLAB R2021a - for writing and running code.
- MacTeX -LaTeXcompiler.
- VS Code with LaTeXworkshop extension as a text editor.

Process

Code for Newton's Forward Interpolation:

```
clc;
clear all;
close all;
matrix = [];

number = input('How many pair of data is given: ');

for i = 1:number
        matrix(i,1) = input('Enter value(x) : ');
        matrix(i,2) = input('Enter value(y) : ');
end

xo = matrix(1,1);
xn = matrix(number,1);
h = matrix(2,1)-xo;
x = input('Enter the value for which the function is to be
↪  evaluated:');
n = number -1;

for j = 3: (number+1)
    for i = 1:n
        matrix(i,j) = matrix(i+1,j-1)-matrix(i,j-1);
    end
    n = n-1;
end
disp(matrix)
p = (x-xo)/h;
z = p;
c = 3;
y0 = matrix(1,2);
for i = 1:number-1
    fprintf('y0 = %f + (%f * %f)/%d! \n',y0,p,matrix(1,c),i)
    y0 = y0 + (pmatrix(1,c))/factorial(i);
    c = c+1;
    p = p(z-i);
end
disp(y0);
```

Output

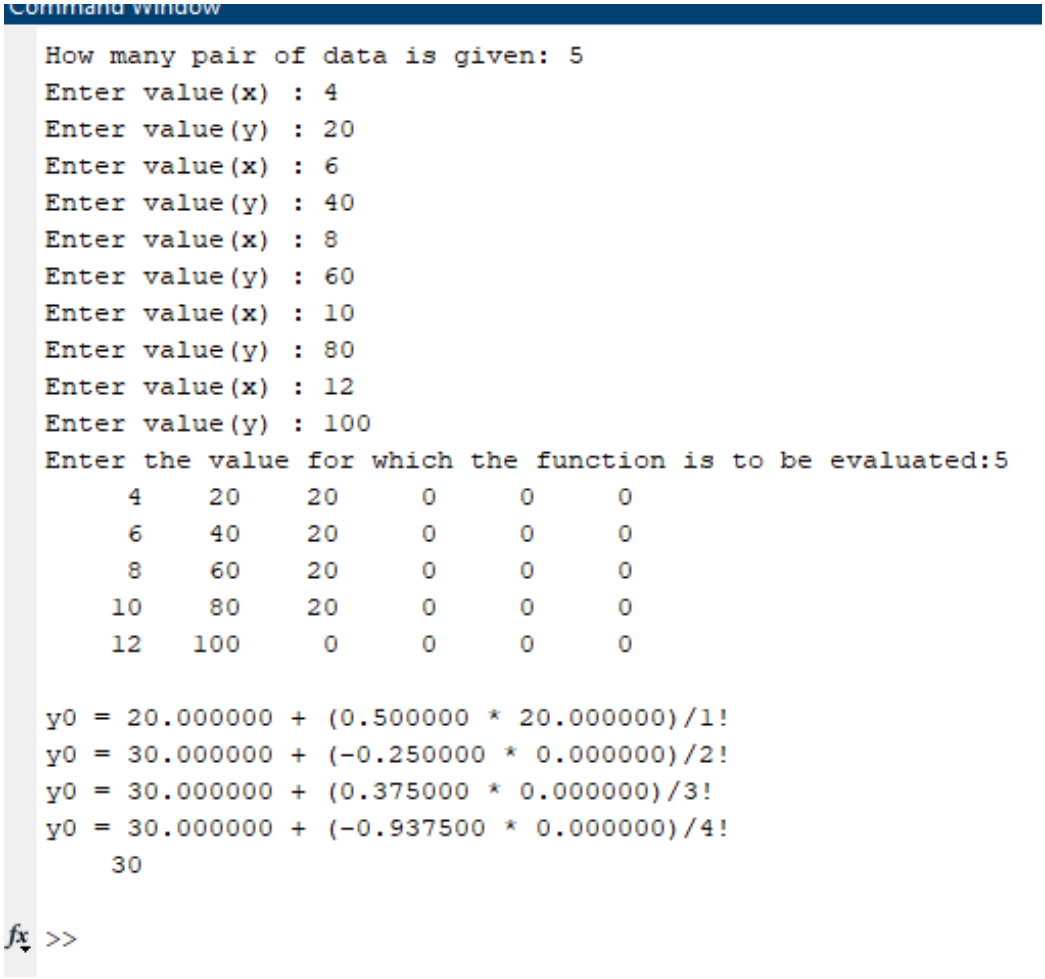

Figure 2: Newton's Forward Method

Implementing Interpolation; Newton's Backward Method in MATLAB

Introduction

Interpolation, Newton's Backward Method

Newton's Backward Interpolation is another numerical technique used for estimating the value of a function between given data points. Similar to Newton's Forward Interpolation, it falls under the category of polynomial interpolation methods and is particularly useful for equally spaced data points.

The general form of the Newton's Backward Interpolation polynomial is:

$$f(x) = P_n(x) = f(x_n) + \Delta y_{n-1} \cdot P_1(u) + \Delta^2 y_{n-2} \cdot P_2(u) + \dots$$

Here, $P_n(x)$ is the polynomial of degree n used for interpolation, $f(x_n)$ represents the value of the function at the last point x_n, Δy_{n-1} denotes the first backward difference, $\Delta^2 y_{n-2}$ signifies the second backward difference, and so on. The symbol u typically represents the normalized difference ratio $\frac{x-x_n}{h}$, where h is the common difference between the data points.

Tools Used

- MATLAB R2021a - for writing and running code.
- MacTeX -LaTeXcompiler.
- VS Code with LaTeXworkshop extension as a text editor.

Process

Code for Newton's Backward Interpolation:

```
clc;
clear all;
close all;
matrix = [];

number = input('How many pair of data is given: ');

for i = 1:number
        matrix(i,1) = input('Enter value(x) : ');
        matrix(i,2) = input('Enter value(y) : ');
end


xo = matrix(1,1);
xn = matrix(number,1);
h = matrix(2,1)-xo;
x = input('Enter the value for which the function is to be
 ↪  evaluated:');

n = number -1;

for j = 3: (number+1)
    for i = 1:n
        matrix(i,j) = matrix(i+1,j-1)-matrix(i,j-1);
    end
    n = n-1;
end

disp(matrix)
p = (x-xn)/h;
z = p;
c = 3;
yn = matrix(number,2);
i = 1;
j = number-1;
while(1)
    fprintf('yn = %f + (%f * %f)/%d! \n',yn,p,matrix(j,c),i)
    yn = yn + (pmatrix(j,c))/factorial(i);
```

```
        if i == number-1
             break
        end
        c = c+1;
        p = p(z+i);
        i = i+1;
        j = j-1;
    end
    disp(yn);
```

Output

```
Command Window
How many pair of data is given: 5
Enter value(x) : 4
Enter value(y) : 20
Enter value(x) : 6
Enter value(y) : 40
Enter value(x) : 8
Enter value(y) : 60
Enter value(x) : 10
Enter value(y) : 80
Enter value(x) : 12
Enter value(y) : 100
Enter the value for which the function is to be evaluated:11
     4    20    20     0     0     0
     6    40    20     0     0     0
     8    60    20     0     0     0
    10    80    20     0     0     0
    12   100     0     0     0     0

yn = 100.000000 + (-0.500000 * 20.000000)/1!
yn = 90.000000 + (-0.250000 * 0.000000)/2!
yn = 90.000000 + (-0.375000 * 0.000000)/3!
yn = 90.000000 + (-0.937500 * 0.000000)/4!
    90

fx >>
```

Figure 3: Newton's Backward Method

Functions

The functions used to do the three methods in MATLAB are as such with brief description of each of them:

for loop The for loop is used to keep the programme running within a specific range. After finishing the process, the final value is passed on to the next programming lines.

plot The Taylor's series of sine waves was plotted using the plot function. In this function, we must provide the x and y values at a specific moment, and the function will plot the diagram based on the values.

hold on, hold off These commands govern how numerous graphs are plotted on the same axes. When we use hold on, we keep the current plot and axis attributes and add any subsequent plots to the existing graph. When we use hold off, the properties of the axes are reset to their defaults, and any new plots replace the current ones.

These function are the newly learned ones for these experiments.[1]

References

[1] “MATLAB Documentation,” Nov. 2023, [Online; accessed 17. Nov. 2023]. [Online]. Available: https://www.mathworks.com/help/matlab/ref