Student Details

Table of Contents

%Roll number : AM25M009 %Name : Mohamed Mafaz %Assignment : Newton's Interpolation %Depatment : Applied Mechanics

Part 1 (Preprocessing / Writing Functions)

Function that finds slope

```
clc;
clear;

function [slope] = divided_difference(y2, y1, x2, x1)
    slope = (y2 - y1) / (x2 - x1);

end

function [sum] = NI(x, y, number)

    % The idea:
    % Intead of using matrix to store all the data, we use a single vector
    % and overwrite it, since the non diagonol hold no value to us for this
    % problem, I overwrite the y array itself


    as = [];          % a's are array of the coefficients
    as = [as, y(1)]; % First coefficient is y's first value itself

    temp_y = y;       % temp_y is a copy of y, but temp_y keeps shrinking its
size, see line 32

    for order = 1: length(x)-1       % Number of Columns
        for i = 1: length(temp_y)-1  % Number of Rows
            temp_y(i) = divided_difference(temp_y(i+1), temp_y(i), x(i +
order), x(i));
                                    % Finding Slope, tricky part is the x's
where we need to skip ith order of x
        end

        temp_y = temp_y(1: end-1); % Shrinking
        as = [as, temp_y(1)];      % Appending to the as array
    end
```

```
    % This is to compute a0 + a1(x-x0) + a2(x-x0)(x-x1) .....
    sum = 0;
    for i = 1: length(as)
        mul = 1;
        for j = 1:i-1
            mul = mul * (number - x(j));
        end
        sum = sum + (as(i)*mul);
    end
end
```

Newton's Basis

```
function [prod] = Newton_Basis(xs, basis, number)
    prod = 1;
    for i = 1: basis
        prod = prod * (number - xs(i));
    end
end
```

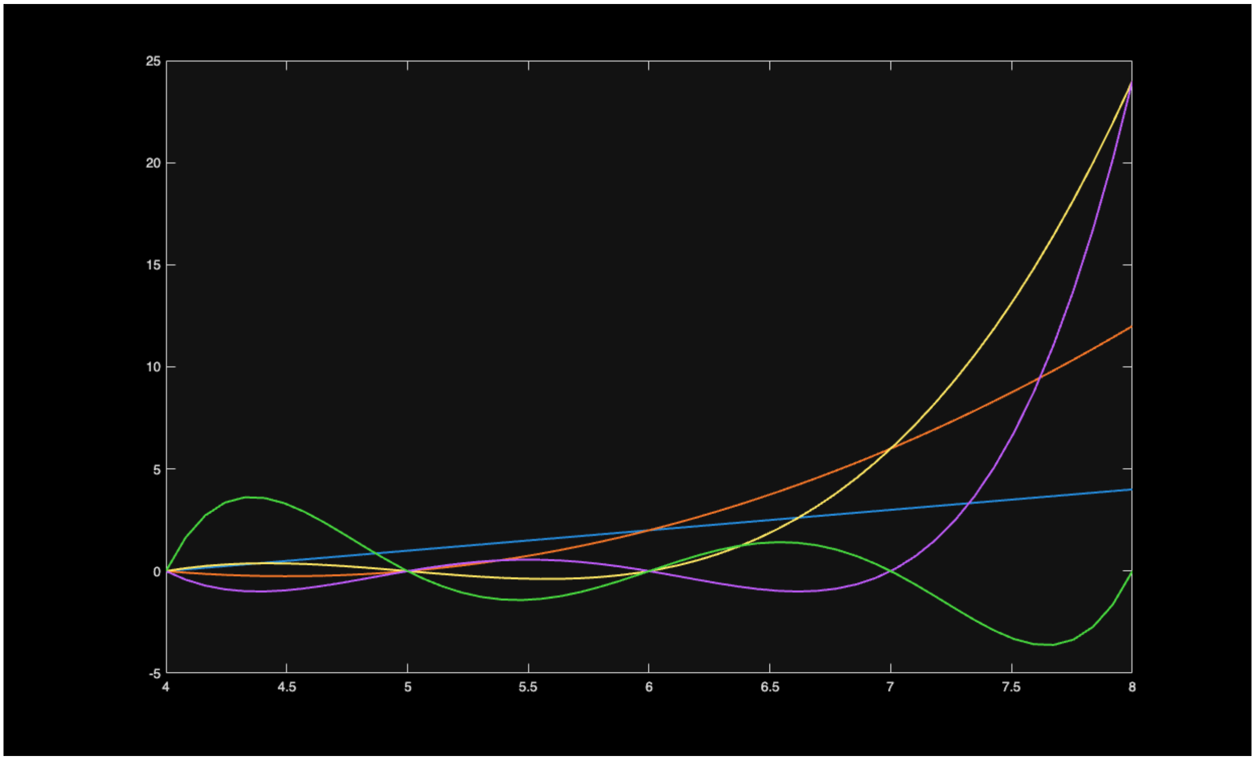
Part 2 (Processing / Using the function)

```
x = [4.0, 5.0, 6.0, 7.0, 8.0];
y = [1.58740105, 1.709976, 1.81712059, 1.912931, 2.0];

sample_points = 50;

% Predicting
test_xs = linspace(min(x), max(x), sample_points);
test_ys = [];
for i = 1: sample_points
    test_ys = [test_ys, NI(x, y, test_xs(i)) ];
end

% Processing and Plotting Newton Polynomial)
for j = 1: length(x)
    test_ys_poly = zeros(1, sample_points);
    for i = 1: sample_points
        test_ys_poly(i) = Newton_Basis(x, j, test_xs(i));
    end
     plot(test_xs, test_ys_poly, 'LineWidth', 1.5, 'DisplayName',
sprintf('P_{%d}(x)', j));
    hold on
end
```



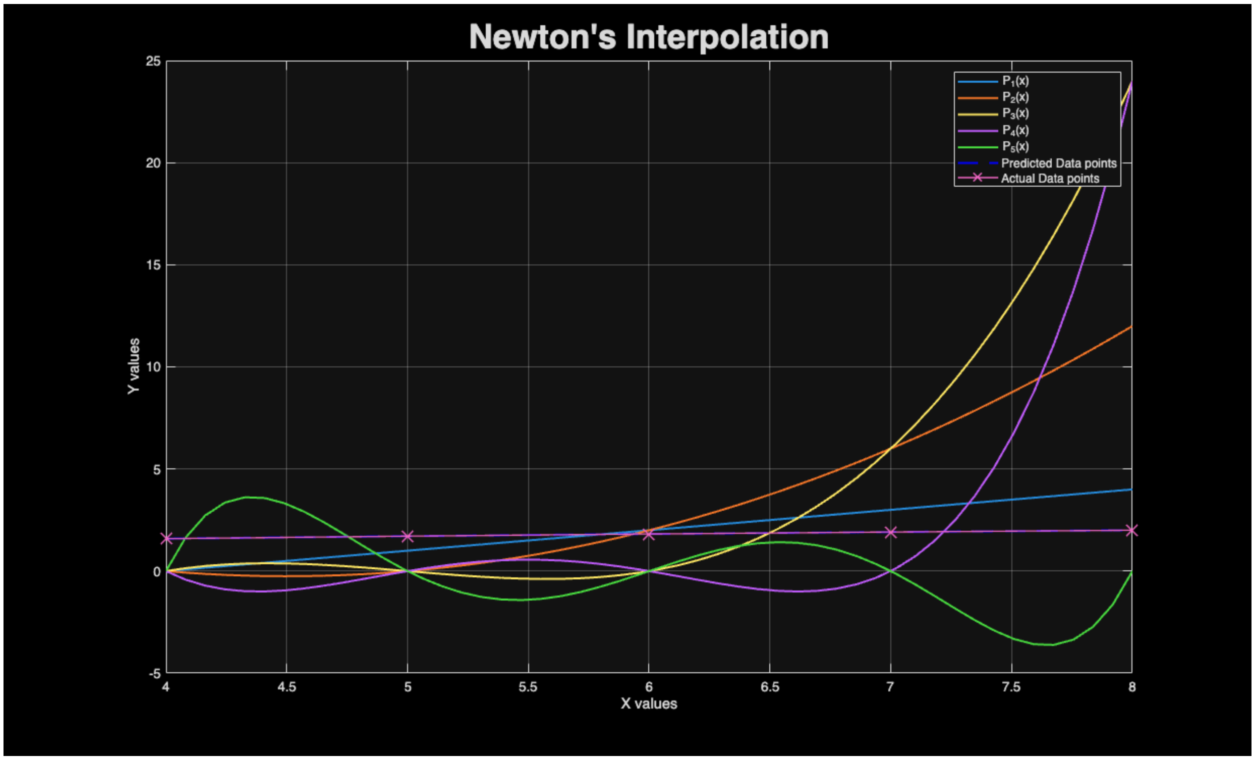
Part 3 (post processing or plots or results)

```
% Plotting predicted Data
plot(test_xs, test_ys, '--b', 'LineWidth', 1.5, 'DisplayName', 'Predicted
Data points');
xlabel('X values');
ylabel('Y values');
title("Newton's Interpolation", 'FontSize', 25);

hold on;

% Plotting actual Data
plot(x, y, 'LineWidth', 1, 'DisplayName', 'Actual Data points', Marker='x',
MarkerSize=12);

legend show
grid on;
```



Published with MATLAB® R2025a