TMA373 Assignment 1

Edoardo Mangia
edoardom@student.chalmers.se

Bibiana Farinha
bibiana@student.chalmers.se

February 2025

Contents

1 Task 1: Polynomial Interpolation in 1D

In this task, we consider the data set

$$\{(2,2),\ (3,6),\ (4,5),\ (5,5),\ (6,6)\}$$

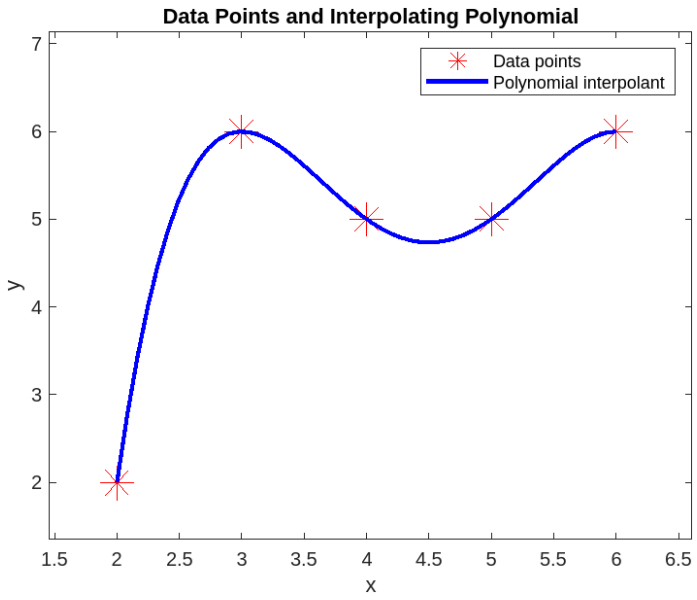
and find the unique polynomial interpolant of degree 4 using MATLAB's `polyfit` and `polyval` functions. The code below plots both the data points and the interpolating polynomial.

Implementation

```
% Define the data points
x = [2, 3, 4, 5, 6];      % nodes
y = [2, 6, 5, 5, 6];      % function values

% Plot the data points
figure;
plot(x, y, 'r*', 'MarkerSize', 16);
xlabel('x');
ylabel('y');
title('Data Points and Interpolating Polynomial');
hold on;  % Keep current plot for the polynomial

% Compute the polynomial interpolant of degree 4
n = 4;
c = polyfit(x, y, n);

```

```
% Generate points for plotting the polynomial
xx = linspace(min(x), max(x), 50);
yy = polyval(c, xx);

% Plot the interpolating polynomial
plot(xx, yy, 'b-', 'LineWidth', 2);
legend('Data points', 'Polynomial interpolant');
hold off;
```

Visualization



2 Task 2: Life Expectancy Data Interpolation

The second task involves interpolating life expectancy data for two European regions. The data given are:

- **Western Europe:** Years: $[1975, 1980, 1985, 1990]$, Life Expectancy: $[72.8, 74.2, 75.2, 76.4]$
- **Eastern Europe:** Years: $[1975, 1980, 1985, 1990]$, Life Expectancy: $[70.2, 70.2, 70.3, 71.2]$

We use a degree 3 polynomial interpolant to estimate the life expectancy in the years 1970, 1983, and 1988, which were chosen arbitrarily.

Implementation

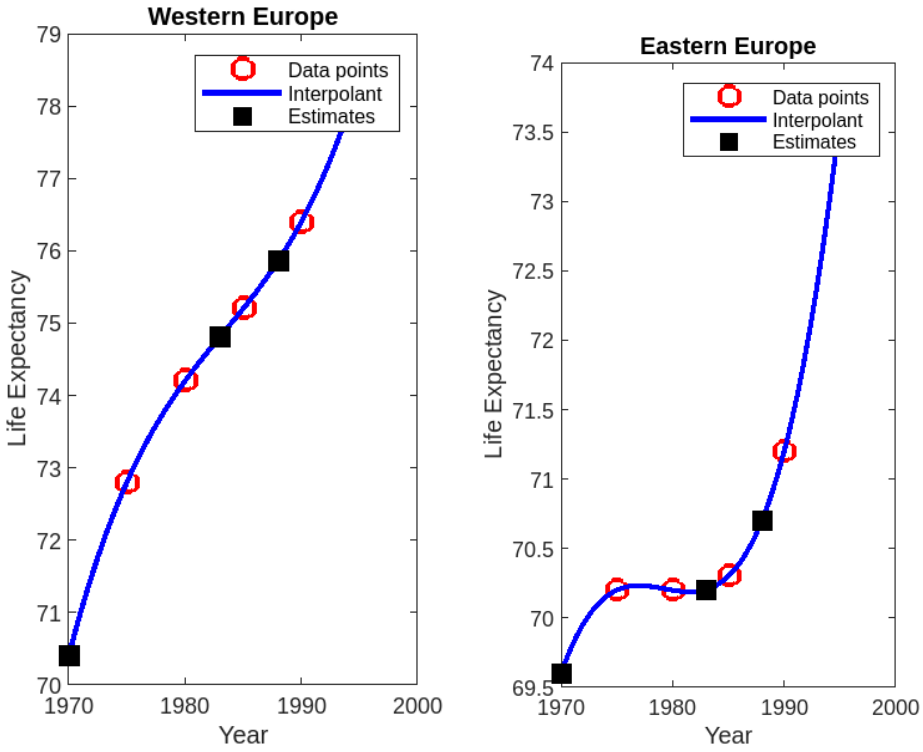
```
% Define the years and corresponding life expectancy data
years = [1975, 1980, 1985, 1990];

```

```
% Western Europe data
LE_west = [72.8, 74.2, 75.2, 76.4];
% Eastern Europe data
LE_east = [70.2, 70.2, 70.3, 71.2];

% Years to estimate
years_est = [1970, 1983, 1988];

% Interpolation for Western Europe
coeff_west = polyfit(years, LE_west, 3);
LE_west_est = polyval(coeff_west, years_est);

% Interpolation for Eastern Europe
coeff_east = polyfit(years, LE_east, 3);
LE_east_est = polyval(coeff_east, years_est);

% Plot results for Western Europe
figure;
subplot(1,2,1);
plot(years, LE_west, 'ro', 'MarkerSize', 8, 'LineWidth', 2);
hold on;
xx_west = linspace(min(years)-5, max(years)+5, 100);
yy_west = polyval(coeff_west, xx_west);
plot(xx_west, yy_west, 'b-', 'LineWidth', 2);
plot(years_est, LE_west_est, 'ks', 'MarkerSize', 10,
    'MarkerFaceColor', 'k');
xlabel('Year');
ylabel('Life Expectancy');
title('Western Europe');
legend('Data points', 'Interpolant', 'Estimates');
hold off;

% Plot results for Eastern Europe
subplot(1,2,2);
plot(years, LE_east, 'ro', 'MarkerSize', 8, 'LineWidth', 2);
hold on;
xx_east = linspace(min(years)-5, max(years)+5, 100);
yy_east = polyval(coeff_east, xx_east);
plot(xx_east, yy_east, 'b-', 'LineWidth', 2);
plot(years_est, LE_east_est, 'ks', 'MarkerSize', 10,
    'MarkerFaceColor', 'k');
xlabel('Year');
ylabel('Life Expectancy');
title('Eastern Europe');
legend('Data points', 'Interpolant', 'Estimates');
hold off;

% Display estimated values in the command window
fprintf('Western Europe estimates:\n');
fprintf('Year 1970: %.2f (Actual: 71.8)\n', LE_west_est(1));
fprintf('Year 1983: %.2f\n', LE_west_est(2));
fprintf('Year 1988: %.2f\n', LE_west_est(3));

fprintf('\nEastern Europe estimates:\n');
fprintf('Year 1970: %.2f (Actual: 69.6)\n', LE_east_est(1));
fprintf('Year 1983: %.2f\n', LE_east_est(2));
fprintf('Year 1988: %.2f\n', LE_east_est(3));
```

Output

```
Western Europe estimates:
Year 1970: 70.40 (Actual: 71.8)
Year 1983: 74.81
Year 1988: 75.86

Eastern Europe estimates:
Year 1970: 69.60 (Actual: 69.6)
Year 1983: 70.20
Year 1988: 70.70
```

Visualization



(a) (b)

Discussion

The interpolation for the year 1970 involves extrapolation, since 1970 lies outside the range of the data.

For Western Europe, the known life expectancy in 1970 is 71.8, the estimated value from the cubic polynomial was 70.4. Similarly, for Eastern Europe the known value is 69.6, value which is identical to the estimated one.

The values obtained by extrapolation are more prone to be incorrect, since the interpolation is based on the given data points.

3 Task 3: Implementation of Lagrange Interpolation

In this task, we implement our own function for Lagrange interpolation without using built-in functions like `polyval`. The MATLAB function `MyLagrangeInterpol` computes the interpolation polynomial based on the Lagrange basis functions.

The internal for loop is used to compute the Lagrange basis polynomial function described as:

$$L_k(xx(i)) = \prod_{\substack{j=1 \\ j \neq k}}^{n} \frac{xx(i) - x_j}{x_k - x_j}$$

Where:

x: Vector of given interpolation nodes (length n).

y: Vector of function values at nodes (length n).

xx: Vector of evaluation points where we want to compute the interpolated function values (length m).

yy: A vector of interpolated function values at each $xx(i)$.

Implementation

```
function yy = MyLagrangeInterpol(x, y, xx)

n = length(x);
m = length(xx);
yy = zeros(1, m);  % initialize result

for i = 1:m
    % Evaluate the interpolating polynomial at xx(i)
    L_val = zeros(1, n);  % values of the Lagrange basis
        polynomials at xx(i)
    for k = 1:n
        % Compute Lagrange basis polynomial L_k(xx(i))
        numerator = 1;
        denominator = 1;
        for j = 1:n
            if j ~= k
                numerator = numerator * (xx(i) - x(j));
                denominator = denominator * (x(k) - x(j));
            end
        end
        L_val(k) = numerator / denominator;
    end
    % Combine the contributions from each basis polynomial
    yy(i) = sum(y .* L_val);
```

```
end

end
```

And comparing it with the built-in function:

```
% Data points
x = [2, 3, 4, 5, 6];
y = [2, 6, 5, 5, 6];

% Evaluate using built-in polyfit/polyval
c = polyfit(x, y, 4);
xx = linspace(min(x), max(x), 50);
yy_polyval = polyval(c, xx);

% Evaluate using the custom Lagrange interpolation function
yy_lagrange = MyLagrangeInterpol(x, y, xx);

% Compute the error between the two methods
error = norm(yy_polyval - yy_lagrange, inf);  % max norm error
fprintf('Maximum difference between polyfit/polyval and
    MyLagrangeInterpol: %e\n', error);

figure;
plot(xx, yy_polyval, 'b-', 'LineWidth', 2);
hold on;
plot(xx, yy_lagrange, 'r--', 'LineWidth', 2);
plot(x, y, 'ko', 'MarkerSize', 8, 'MarkerFaceColor', 'k');
xlabel('x');
ylabel('Interpolated y');
title('Comparison: Built-in vs. Lagrange Interpolation');
legend('polyval', 'MyLagrangeInterpol', 'Data points');
hold off;
```

Output

```
Maximum difference between polyfit/polyval and MyLagrangeInterpol:
    2.344791e-13
```

The maximum difference computed between the result of this function and MATLAB's `polyval` is negligible, standing at 2.344791e-13.

Visualization