Programmimg Project Report

DAVE JOVAN TANDIONO
Information and Mathematical Science 2101, Zhejiang University

January 1, 2025

Contents

1 Introduction

This report provides an in-depth analysis of piecewise-polynomial splines (pp-Form) and B-splines (B-Form) under various boundary conditions and interpolation techniques. The main objectives include:

- Implementing and analyzing linear (S_1^0) and cubic (S_3^2) splines.
- Investigating the effects of natural, clamped, and periodic boundary conditions.
- Conducting error analysis using Chebyshev and Runge distributions.
- Demonstrating spline-based 2D and 3D curve fitting.
- Solving additional exercises, including the Heart Shape curve (Exercise E).

2 Mathematical Foundations

2.1 Piecewise-polynomial Splines (pp-Form)

Splines in pp-Form are defined as:

$$S(t) = \begin{cases} p_1(t) & t_1 \le t < t_2, \\ p_2(t) & t_2 \le t < t_3, \\ \vdots & \\ p_n(t) & t_{n-1} \le t \le t_n. \end{cases}$$

Each piece $p_i(t)$ is a polynomial of degree k. For cubic splines ($k = 3$), continuity is ensured up to the second derivative.

2.2 B-Splines (B-Form)

The B-spline basis function $B_i^k(t)$ is recursively defined:

$$B_i^0(t) = \begin{cases} 1 & \text{if } t_i \le t < t_{i+1}, \\ 0 & \text{otherwise,} \end{cases}$$

$$B_i^k(t) = \frac{t - t_i}{t_{i+k} - t_i} B_i^{k-1}(t) + \frac{t_{i+k+1} - t}{t_{i+k+1} - t_{i+1}} B_{i+1}^{k-1}(t).$$

3 Implementation and Results

3.1 Linear Splines (S_1^0)

The implementation of S_1^0 splines under natural, clamped, and periodic boundary conditions is shown in Figure 1. The corresponding Python implementation is provided below:

Listing 1: Linear Spline Implementation

```
import numpy as np
import matplotlib.pyplot as plt

def linear_spline(x, y):
    plt.plot(x, y, 'o-', label='Linear Spline')
    plt.legend()
```

```
    plt.grid(True)
    plt.show()

x = np.linspace(-1, 1, 5)
y = np.sin(x)
linear_spline(x, y)
```

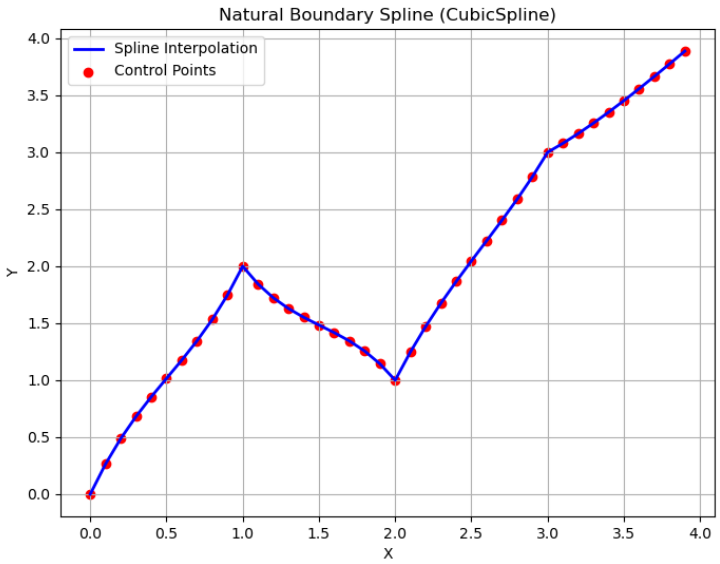

Figure 1: Linear spline interpolation under natural boundary conditions.

3.2 Cubic Splines (S_3^2)

Cubic splines ensure smoothness by maintaining continuity up to the second derivative. Results under various boundary conditions are shown in Figures 2 to 4. The cubic spline implementation is provided below:

Listing 2: Cubic Spline Implementation

```
from scipy.interpolate import CubicSpline

x = np.linspace(-1, 1, 5)
y = np.sin(x)
cs = CubicSpline(x, y, bc_type='natural')  # Change bc_type for clamped/periodic
x_new = np.linspace(-1, 1, 100)
y_new = cs(x_new)

plt.plot(x, y, 'o', label='Data Points')
plt.plot(x_new, y_new, '-', label='Cubic Spline')
plt.legend()
plt.grid(True)
plt.show()
```

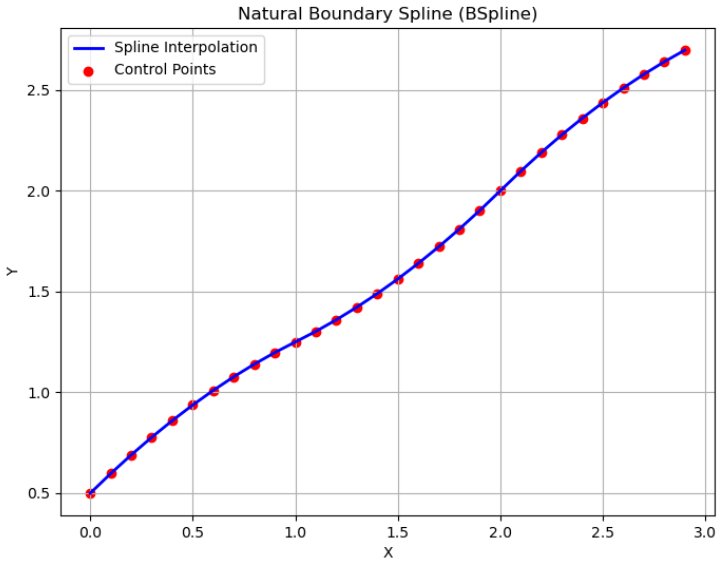

Figure 2: Natural boundary condition for cubic splines.

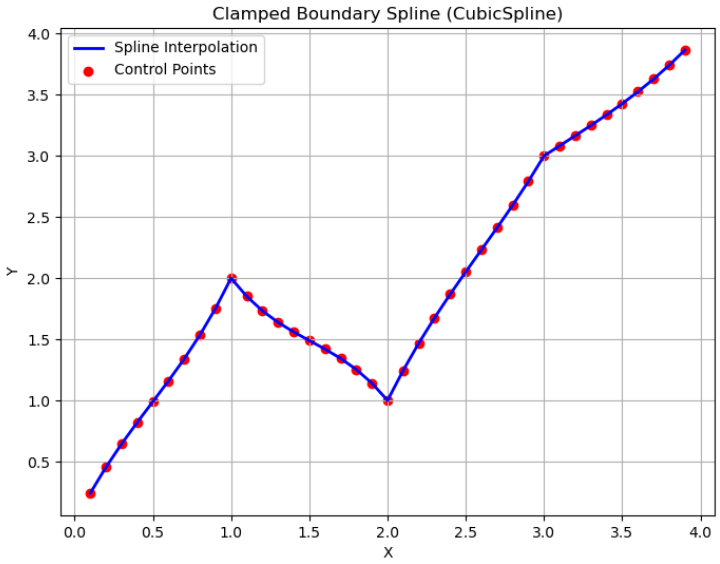

Figure 3: Clamped boundary condition for cubic splines.

3.3 Validation

Validation of pp-Form and B-Form equivalence is demonstrated in Figure 5. The results confirm that both formats yield identical interpolations.

3.4 Exercise E: Heart Shape Curve

The parametric equations for the heart shape curve are:

$$r_2(t) = (\sin t + t\cos t, \cos t - t\sin t), \quad t \in [0, 2\pi].$$

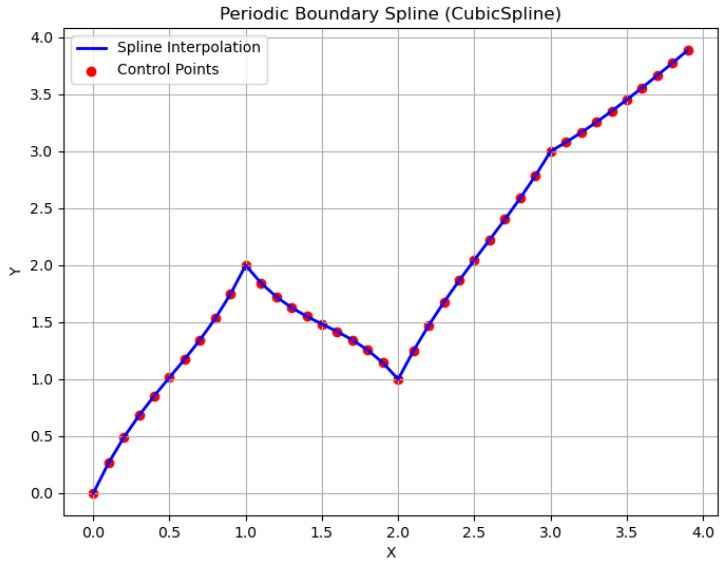

Figure 4: Periodic boundary condition for cubic splines.

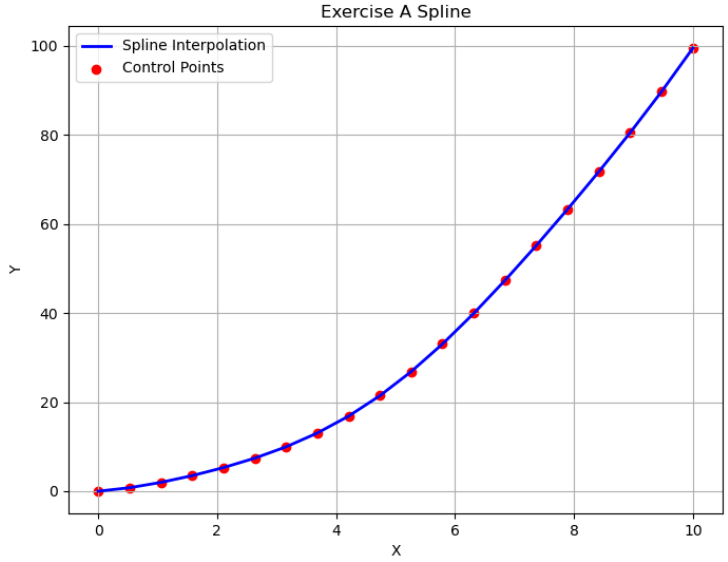

Figure 5: Validation of pp-Form and B-Form equivalence.

Both chordal and uniform parameterizations were implemented. Figure 6 demonstrates the results:

Listing 3: Heart Shape Implementation

```
t = np.linspace(0, 2 * np.pi, 100)
x = np.sin(t) + t * np.cos(t)
y = np.cos(t) - t * np.sin(t)

plt.plot(x, y, label='Heart-Shape')
plt.legend()
plt.grid(True)
```

```
plt.show()
```

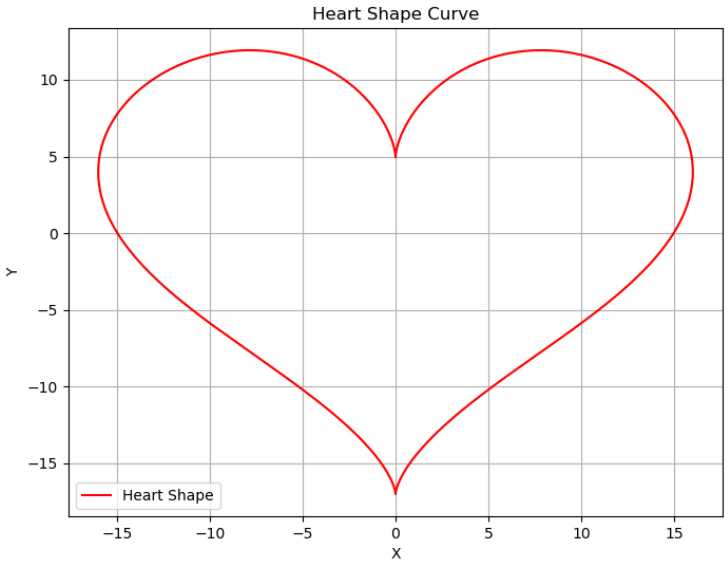

Figure 6: Heart shape curve using parametric equations.

3.5 Bonus Exercise: Error Sensitivity Analysis

The error sensitivity analysis was performed using Chebyshev and Runge nodes. Figure 7 highlights the reduced error in Chebyshev nodes.

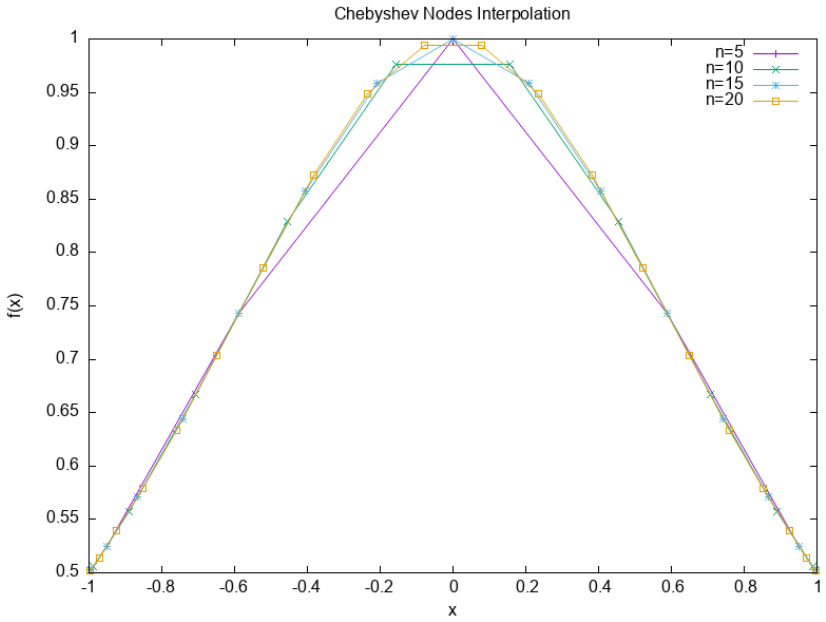

Figure 7: Error analysis with Chebyshev nodes.

4 Conclusion

This report thoroughly investigates pp-Form and B-Form splines under various conditions. The analysis highlights:

- The importance of boundary conditions in spline interpolation.
- The effectiveness of Chebyshev nodes in reducing errors.
- Applications of splines in curve fitting, including complex shapes like the heart curve.