Programming Project Design

DAVE JOVAN TANDIONO 3210300364
Information and Mathematical Science 2101, Zhejiang University
Due time: January 1, 2025

Abstract

This report presents a comprehensive design and implementation of the Splinify system, a computational tool for spline interpolation. The document elaborates on the mathematical principles, software architecture, implementation details, and test results for various types of splines, including cubic splines, B-splines, and piecewise polynomial splines.

Contents

1 Introduction

Spline interpolation is a mathematical technique to approximate functions or data points using piecewise polynomials. The Splinify system is designed to implement and test different spline types, focusing on:

- Natural and clamped cubic splines
- B-splines with different boundary conditions
- Piecewise polynomial splines

The system is implemented in C++ and includes Python scripts for post-processing.

2 Mathematical Formulation

This section describes the mathematical foundation for the splines implemented in the Splinify system.

2.1 Cubic Spline

A cubic spline is a piecewise function $S(x)$, defined as:

$$S(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, \quad x \in [x_i, x_{i+1}]$$

where the coefficients a_i, b_i, c_i, d_i are determined by:

1. Continuity at internal points
2. Smoothness conditions for the first and second derivatives
3. Boundary conditions:
   - **Natural Boundary:** $S''(x_0) = 0$ and $S''(x_n) = 0$
   - **Clamped Boundary:** Specified values for $S'(x_0)$ and $S'(x_n)$

2.2 B-Spline

A B-spline is defined using basis functions $B_{i,k}(x)$:

$$B_{i,1}(x) = \begin{cases} 1 & \text{if } x \in [x_i, x_{i+1}) \\ 0 & \text{otherwise} \end{cases}$$

For $k > 1$, the recursive relation is:

$$B_{i,k}(x) = \frac{x - x_i}{x_{i+k-1} - x_i} B_{i,k-1}(x) + \frac{x_{i+k} - x}{x_{i+k} - x_{i+1}} B_{i+1,k-1}(x)$$

2.3 Piecewise Polynomial

A piecewise polynomial is constructed as:

$$P(x) = \begin{cases} P_1(x) & x \in [x_0, x_1] \\ P_2(x) & x \in [x_1, x_2] \\ \vdots & \\ P_n(x) & x \in [x_{n-1}, x_n] \end{cases}$$

3 Implementation

The Splinify system is implemented in C++ with Python for post-processing and testing. The key modules are:

- **SplineUtils:** Includes core classes like `CubicSpline`, `BSpline`, and `PiecewisePolynomialSpline`.
- **InputValidator:** Ensures valid input data.
- **Visualization:** Python scripts to generate numerical outputs.

3.1 Code Structure

The directory structure is as follows:

```
Splinify/
  |- src/
     |- SplineUtils/
  |- tests/
     |- CppTests/
  |- scripts/
  |- docs/
```

3.2 Code Example

The following code computes a cubic spline:

```
#include <iostream>
#include "CubicSpline.h"

int main() {
    CubicSpline spline({0, 1, 2}, {0, 1, 0});
    double y = spline.interpolate(1.5);
    std::cout << "Interpolated value at x=1.5: " << y << std::endl;
    return 0;
}
```

4 Testing and Results

The Splinify system was tested with multiple datasets.

4.1 Cubic Spline Test

The cubic spline test produced the following results:

x	y
0	0
0.5	1.01562
1	2
1.5	1.48437
2	1

4.2 B-Spline Test

The B-spline test confirmed smoothness at control points, with numerical outputs verifying the accuracy.

4.3 Piecewise Polynomial Test

The piecewise polynomial spline was evaluated for edge cases, ensuring correct interpolation for non-uniform datasets.

5 Future Work

The following enhancements are proposed for the Splinify system:

- Extend to higher-dimensional splines for 3D data visualization.
- Optimize computational efficiency for handling large datasets.
- Integrate machine learning to predict optimal spline parameters.

6 Conclusion

The Splinify system provides a robust framework for implementing and testing spline interpolation techniques. The integration of numerical analysis and computational efficiency makes it a versatile tool for data approximation tasks.