Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Курсовой проект по курсу «Численные методы»

Тема: «Интерполяция экспоненциальными сплайнами»

Студент: В. В. Бирюков
Преподаватель: Д. Л. Ревизников
Группа: М8О-307Б-19
Дата:
Оценка:
Подпись:

Москва, 2022

Постановка задачи

Задача: Построить экспоненциальный сплайн для функции, заданной в узлах интерполяции. Осуществить выбор оптимальных параметров натяжения. Сравнить полученные результаты с интерполяцией кубическими сплайнами.

1 Описание

Экспоненциальный сплайн

Кубический сплайн имеет прямую аналогию в теории упругости, а именно описывает поведение гибкого стрежня, закрепленного в некоторых точках. С этой точки зрения, кубический сплайн s на каждом интервале $[x_i, x_{i+1}]$, $i = 1, \ldots, N$ является решением следующей краевой задачи:

$$[D^4]s = 0,\ s(x_i) = f_i,\ s(x_{i+1}) = f_{i+1},\ s''(x_i) = s''_i,\ s''(x_{i+1}) = s''_{i+1}$$

Где s''_i и s''_{i+1} подобраны так, чтобы $s \in C^2[a, b]$, и получаются как решение системы с трехдиагональной матрицей.

Полученный таким образом сплайн имеет склонность к образованию точек перегиба независимо от того, соответствует ли это исходным данным или нет. Точки перегиба, возникающие вследствие нежелательных колебаний интерполяционной кривой, называют ложными точками перегиба.

Во избежании таких точек, наложим постоянное «натяжение» на интервалы, где они появляется. Определим экспоненциальный сплайн τ как решение совокупности краевых задач на интервалах $[x_i, x_{i+1}]$, $i = 1, \ldots, N$ вида:

$$[D^4 - p_i^2 D^2]\tau = 0,\ \tau(x_i) = f_i,\ \tau(x_{i+1}) = f_{i+1},\ \tau''(x_i) = \tau''_i,\ \tau''(x_{i+1}) = \tau''_{i+1}$$

Где p_i, $i = 1, \ldots, N$ — параметр натяжения.

При этом имеется два предельных случая:

1. $p_i \to 0 \ \Rightarrow\ [D^4 - p_i^2 D^2]\tau = 0 \ \Rightarrow\ [D^4]\tau = 0$. Экспоненциальный сплайн вырождается в кубический сплайн.
2. $p_i \to \infty \ \Rightarrow\ [D^4 - p_i^2 D^2]\tau = 0 \ \Rightarrow\ [(1/p_i^2)D^4 - D^2]\tau = 0 \ \Rightarrow\ [D^2]\tau = 0$. Экспоненциальный сплайн вырождается в сплайн первого порядка — ломаную линию.

На интервале $[x_i, x_{i+1}]$, $i = 1, \ldots, N$ экспоненциальный сплайн задается формулой

$$\tau(x) = \frac{1}{p_i^2 S_i}\left[\tau''_i \operatorname{sh}(p_i(x_{i+1}-x)) + \tau''_{i+1} \operatorname{sh}(p_i(x-x_i))\right] + \left(f_i - \frac{\tau''_i}{p_i^2}\right)\frac{x_{i+1}-x}{h_i} + \left(f_{i+1} - \frac{\tau''_{i+1}}{p_i^2}\right)\frac{x-x_i}{h_i}$$

Где f_i, $i = 1, \ldots, N+1$ — значения функции в узлах интерполяции;
$h_i = x_{i+1} - x_i$, $S_i = \sinh(p_i h_i)$, $i = 1, \ldots, N$;
τ_i'' определяются решением системы уравнений с трехдиагональной матрицей:

$$\begin{cases} d_1\tau_1'' = b_1 \\ e_{i-1}\tau_{i-1}'' + (d_{i-1} + d_i)\tau_i'' + e_i\tau_{i+1}'' = b_i, \ (i = 2, \ldots, N) \\ d_N\tau_{N+1}'' = b_{N+1} \end{cases}$$

Где:

$$\begin{aligned} e_i &= \left(\frac{1}{h_i} - \frac{p_i}{S_i}\right)/p_i^2 \\ d_i &= \left(p_i\frac{C_i}{S_i} - \frac{1}{h_i}\right)/p_i^2 && i = 1, \ldots, N \\ C_i &= \cosh(p_i h_i) \end{aligned}$$

$$\begin{aligned} b_1 &= b_{N+1} = 0 \\ b_i &= \frac{f_{i+1} - f_i}{h_i} - \frac{f_i - f_{i-1}}{h_{i-1}} && i = 2, \ldots, N \end{aligned}$$

Данная система соответствует сплайну с естественными граничными условиями ($\tau_1'' = \tau_{N+1}'' = 0$).

Выбор параметров натяжения

Отдельный интерес представляет задача выбора параметра натяжения для каждого из интервалов интерполяции, достаточного для устранения ложных точек перегиба. Ложные точки перегиба однозначно устраняются при «достаточно больших» значениях параметра, но это также приводит к появлению областей большой кривизны вблизи узлов.

Отсутствие точек перегиба на интервале $[x_i, x_{i+1}]$ означает постоянство знака второй производной τ''. Тогда условия $\tau_i'' b_i > 0$, $\tau_{i+1}'' b_{i+1} > 0$ являются необходимыми и достаточными для этого. Таким образом появляется возможность итеративно изменять параметр p_i до достижения условия $\tau_i'' b_i > 0$ $i = 1, \ldots, N+1$.

Так как рассматривается только естественный сплайн ($\tau_1'' = \tau_{N+1}'' = 0$), ограничим рассмотрение точками $i = 2, \ldots, N$. Пусть для некоторого набора параметров $p_i^{(n)}$, $i = 1, \ldots, N$, выполняется $\tau_k'' b_k < 0$, $k \in [2, N]$. Обозначим

$$\overline{\lambda} = \frac{\max(|b_k|, (d_{k-1} + d_k)|\tau_k''|)}{2\max(|\tau_{k-1}''|, |\tau_{k+1}''|)}$$

и положим $\tilde{p}_i = (\overline{\lambda} h_i)^{-1/2}$, $i = k-1, k$ (или возможно $\tilde{p}_i = \max(p_i^{(n)}, (\overline{\lambda} h_i)^{-1/2})$, если необходимо только увеличивать параметр).

Для получения следующих значений параметров натяжения применим механизм релаксации:

$$p^{(n+1)} = p^{(n)} + \omega(\tilde{p} - p^{(n)})$$

Данный итеративный процесс позволяет дополнительно «натянуть» сплайн в интервалах с ложными точками перегиба, при этом он не всегда помогает бороться с ложными экстремумами.

2 Исходный код

Реализация экспоненциального сплайна. Для решения системы сплайна используется метод прогонки, реализованный ранее в лабораторных работах.

```cpp
#pragma once

#include <vector>
#include <cmath>
#include <iostream>

#include "../linear/tridiagonal_matrix.hpp"
#include "../linear/vector.hpp"

template <class T>
class ExponentialSpline {
public:
  class Segment {
  public:
    T p, h, t1, t2, x1, x2, f1, f2;

    Segment(): p(0), h(0), t1(0), t2(0), x1(0), x2(0), f1(0), f2(0) { }

    Segment(T p, T x1, T x2, T f1, T f2, T t1, T t2):
      p(p), h(std::abs(x1 - x2)), t1(t1), t2(t2), x1(x1), x2(x2), f1(f1), f2(f2) { }

    T operator()(T x) {
      return (t1 * std::sinh(p * (x2 - x)) + t2 * std::sinh(p * (x - x1))) / (p * p *
          std::sinh(p * h)) +
             (f1 - t1 / (p * p)) * (x2 - x) / h +
             (f2 - t2 / (p * p)) * (x - x1) / h;
    }

    friend std::ostream& operator<<(std::ostream& os, const Segment& seg) {
      os << "(" << seg.t1 << " * sinh(" << seg.p << " * (" << seg.x2 << " - x)) + " <<
          seg.t2 << " * sinh(" << seg.p << " * (x - " << seg.x1 << "))) / " << seg.p *
          seg.p * std::sinh(seg.p * seg.h) << " + "
         << (seg.f1 - seg.t1 / (seg.p * seg.p)) << " * (" << seg.x2 << " - x) / " <<
             seg.h << " + "
         << (seg.f2 - seg.t2 / (seg.p * seg.p)) << " * (x - " << seg.x1 << ") / " <<
             seg.h;
      return os;
    }
  };

  std::vector<T> x, y, p;
  std::vector<T> h, d, e;
  Vector<T> b, t;
  std::vector<Segment> segment;
```

```
40
41  ExponentialSpline(const std::vector<T>& x, const std::vector<T>& y, const std::
        vector<T>& p) : x(x), y(y), p(p), h(x.size() - 1), d(x.size() - 1), e(x.size() -
        1), b(x.size()), t(x.size()), segment(x.size() - 1) {
42    if (x.size() != y.size() && x.size() - 1 != p.size()) {
43      throw std::runtime_error("Incompatible arrays");
44    }
45
46    Solve();
47  }
48
49  void Solve() {
50    size_t n = h.size();
51    for (size_t i = 0; i < x.size() - 1; ++i) {
52      h[i] = std::abs(x[i + 1] - x[i]);
53      d[i] = (p[i] * std::cosh(p[i] * h[i]) / std::sinh(p[i] * h[i]) - 1 / h[i]) / (p[i
            ] * p[i]);
54      e[i] = (1 / h[i] - p[i] / std::sinh(p[i] * h[i])) / (p[i] * p[i]);
55    }
56
57    b[0] = 0;
58    b[n] = 0;
59    for (size_t i = 1; i < n; ++i) {
60      b[i] = (y[i+1] - y[i]) / h[i] - (y[i] - y[i-1]) / h[i-1];
61    }
62
63    TDMatrix<T> matrix(n+1);
64    matrix.b[0] = 1;
65    for (size_t i = 1; i < n; ++i) {
66      matrix.a[i] = e[i-1];
67      matrix.b[i] = d[i-1] + d[i];
68      matrix.c[i] = e[i];
69    }
70    matrix.b[n] = 1;
71
72    t = matrix.Solve(b);
73
74    for (size_t i = 0; i < n; ++i) {
75      segment[i] = Segment(p[i], x[i], x[i+1], y[i], y[i+1], t[i], t[i+1]);
76    }
77  }
78
79  void Tense(T relax) {
80    for (size_t k = 1; k < t.Size()-1; ++k) {
81      if (t[k] * b[k] < 0) {
82        T lambda = std::max(std::abs(b[k]), (d[k-1] + d[k]) * std::abs(t[k])) / 2 / std
              ::max(std::abs(t[k-1]), std::abs(t[k+1]));
83        for (size_t i = k-1; i <= k; ++i) {
84          T new_p = 1 / std::sqrt(lambda * h[i]);
```

```
85          p[i] = p[i] + relax * (new_p - p[i]);
86        }
87      }
88    }
89
90    Solve();
91  }
92
93  size_t Size() const {
94    return segment.size();
95  }
96
97  const Segment& operator[](size_t index) {
98    return segment[index];
99  }
100
101 T operator()(const T& value) {
102   if (value < x[0] || value > x.back()) {
103     throw std::runtime_error("Out of range");
104   }
105
106   for (size_t i = 0; i < x.size()-1; ++i) {
107     if (value >= x[i] && value <= x[i+1] ) {
108       return segment[i](value);
109     }
110   }
111
112   return 0; // just for silence the warning
113 }
114 };
115
116 template <class T>
117 std::ostream& operator<<(std::ostream& os, const ExponentialSpline<T>& spline) {
118   for (size_t i = 0; i < spline.x.size() - 1; ++i) {
119     os << '[' << spline.x[i] << ':' << spline.x[i+1] << "] " << spline.segment[i] << '\
        n';
120   }
121   return os;
122 }
```

3 Результаты работы

Программа реализована как консольная утилита, в параметрах запуска можно задать значение параметра натяжения для всех интервалов (если они не указаны во входных данных), число итераций по изменению натяжения, значение параметра релаксации (по умолчанию — 1), формат вывода — формулы сегментов (по умолчанию) или скрипт для `gnuplot`.
Демонстрация текстового вывода:

```
$ ./main -p 10 -tense 1 -relax 0.5 <tests/sin.txt
[0.3927:1.1781] (0 * sinh(10 * (1.1781 -x)) + 15.8202 * sinh(10 * (x -0.3927))) / 128801
+ 0.5 * (1.1781 -x) / 0.7854 + -0.658202 * (x -0.3927) / 0.7854
[1.1781:1.9635] (15.8202 * sinh(10 * (1.9635 -x)) + -16.9673 * sinh(10 * (x -1.1781))) /
128801 + -0.658202 * (1.9635 -x) / 0.7854 + 0.669673 * (x -1.1781) / 0.7854
[1.9635:2.7489] (-16.9673 * sinh(10 * (2.7489 -x)) + 16.9673 * sinh(10 * (x -1.9635))) /
128801 + 0.669673 * (2.7489 -x) / 0.7854 + -0.669673 * (x -1.9635) / 0.7854
[2.7489:3.5343] (16.9673 * sinh(10 * (3.5343 -x)) + -15.8202 * sinh(10 * (x -2.7489))) /
128801 + -0.669673 * (3.5343 -x) / 0.7854 + 0.658202 * (x -2.7489) / 0.7854
[3.5343:4.3197] (-15.8202 * sinh(10 * (4.3197 -x)) + 0 * sinh(10 * (x -3.5343))) / 128801
+ 0.658202 * (4.3197 -x) / 0.7854 + -0.5 * (x -3.5343) / 0.7854
```

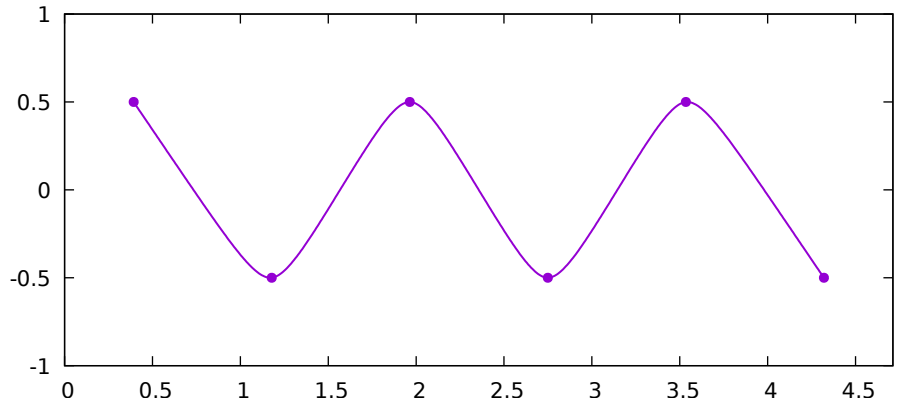
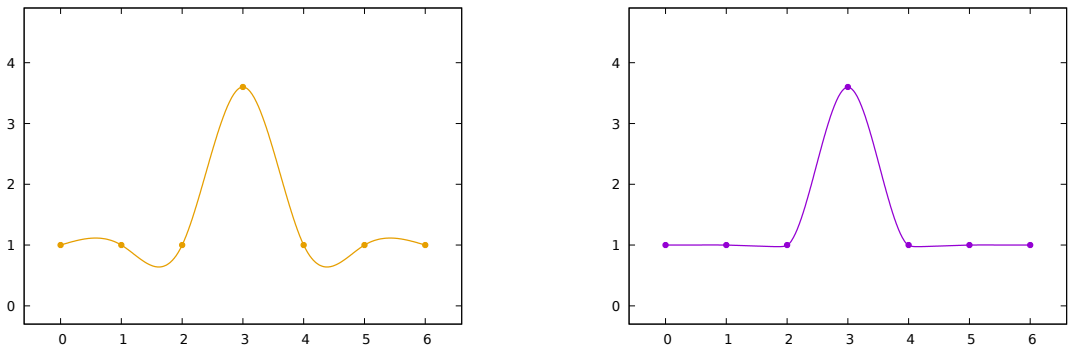
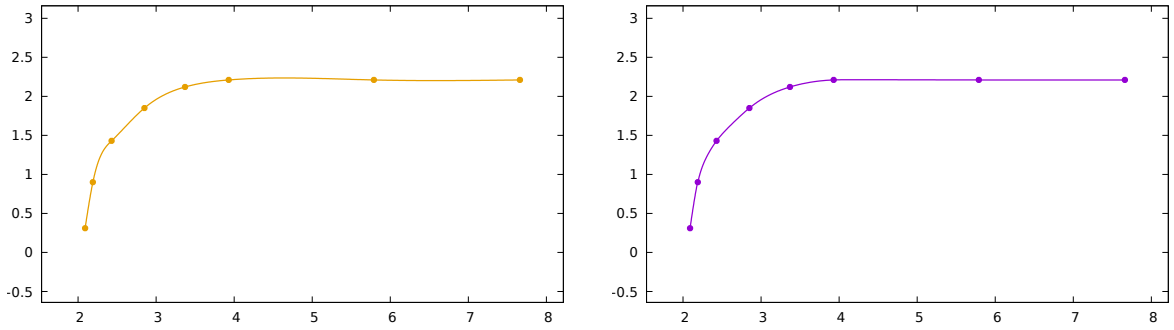


Рис. 1: Предыдущий тестовый пример



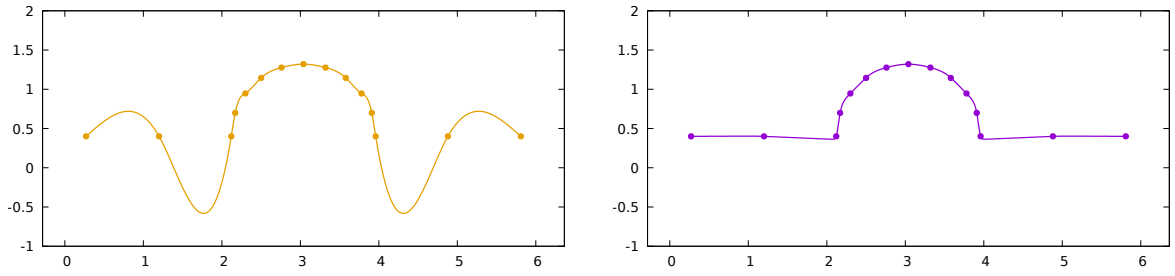
(a) (b)

Рис. 2: Пример с выпавшей точкой. (a) Кубический сплайн. (b) Экспоненциальный сплайн (параметры натяжения подобраны вручную)



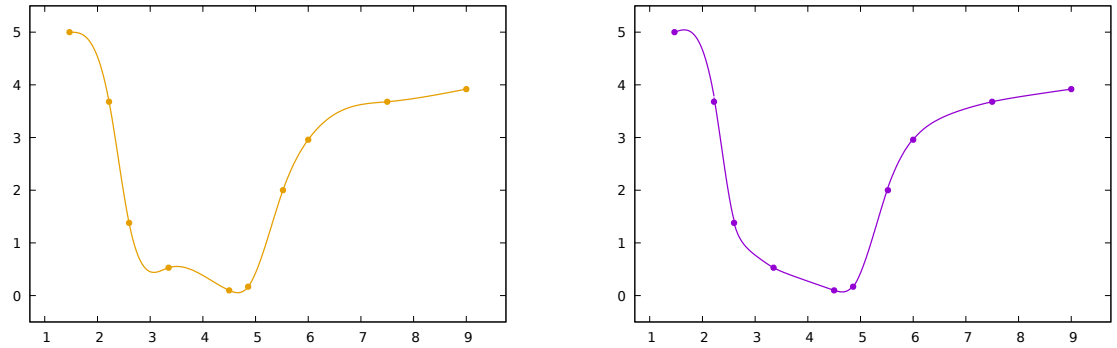
(a) (b)

Рис. 3: Пример с вертикальным наклоном. (a) Кубический сплайн. (b) Экспоненциальный сплайн (параметры натяжения подобраны частично вручную, частично итеративным методом)



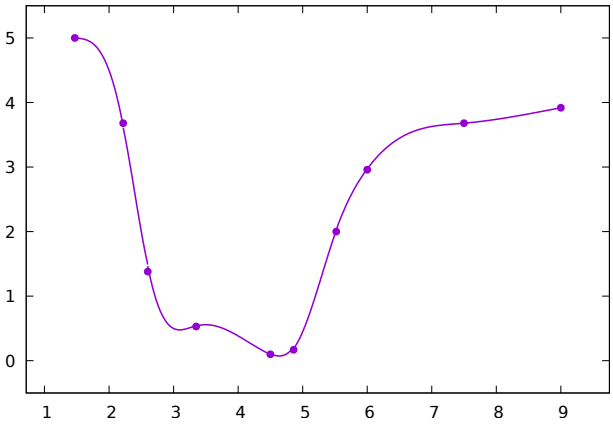
(a) (b)

Рис. 4: Пример с разрывом наклона. (a) Кубический сплайн. (b) Экспоненциальный сплайн (параметры натяжения подобраны частично вручную, частично итеративным методом)

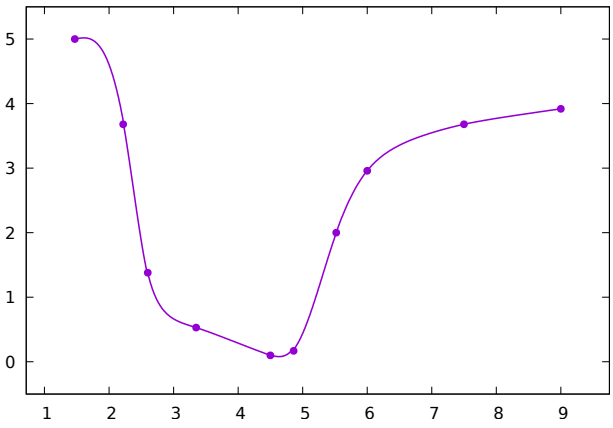


(a) (b)

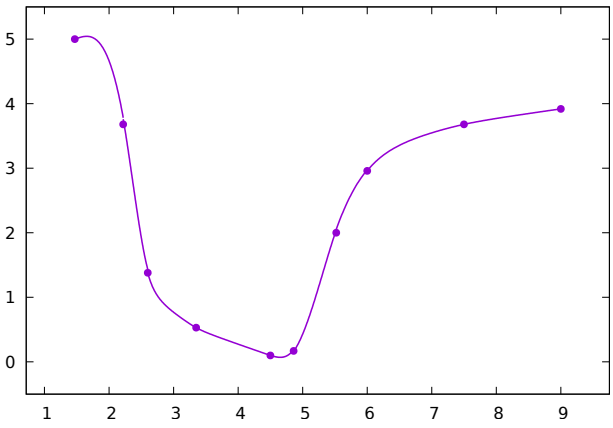
Рис. 5: Пример с ложными точками перегиба. (a) Кубический сплайн. (b) Экспоненциальный сплайн (параметры натяжения подобраны итеративным методом)



(a)



(b)



(c)

Рис. 6: Пример с ложными точками перегиба. (a) Нулевая итерация. (b) Первая итерация. (c) Вторая итерация

4 Выводы

В ходе выполнения курсового проекта я познакомился с экспоненциальными сплайнами. Они имеют схожее происхождение, что и кубические сплайны, но получаются значительно сложнее. При практичеком использовании на некоторых данных крайне сложно подобрать параметры сплайна, приводящие к удовлетворительным результатам. Тогда как на других достаточно встроенного итеративного процесса, при этом полученная интерполяционная кривая более однородна, чем при использовании кубических сплайнов.

Список литературы

[1] B. J. McCartin. *Theory, Computation, and Application of Exponential Splines.* — New York: Courant Institute of Mathematical Sciences, New York University. 1981. 296 c.

[2] Б. Дж. Маккартин. *Применение экспоненциальных сплайнов в вычислительной гидродинамике.* // Аэрокосмическая техника. 1984. №4. Т. 2. С. 13—19.