Bài thuyết trình về đề tài Cubic Spline Interpolation - Giải Tích Số

- Giáo viên hướng dẫn: **TS. Bùi Xuân Thắng**
- Sinh viên thực hiện: Hà Thị Ninh Giang - MSSV: 24C24013, Nguyễn Lê Tâm Anh - MSSV: 24C24010

Ôn tập Chương 3: Nội suy và Xấp xỉ Đa thức

Mục tiêu chung

Nội suy là kỹ thuật toán học giúp ta xây dựng một hàm (thường là đa thức) để ước lượng giá trị tại các điểm chưa biết, dựa trên các điểm dữ liệu đã cho. Từ 3.1 đến 3.4, chúng ta đã khám phá nhiều phương pháp khác nhau, mỗi phương pháp có công thức, cách tính và sai số riêng. Hãy cùng ôn lại từng phần với các yếu tố chính: **công thức**, **phương pháp tính**, và **công thức sai số**.

3.1 - Nội suy và Đa thức Lagrange

- **Ý tưởng:** Tạo một đa thức duy nhất đi qua tất cả các điểm dữ liệu.
- **Công thức:** Với $n+1$ điểm $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$, đa thức Lagrange bậc n:
$$P(x) = \sum_{k=0}^{n} y_k L_k(x), \quad L_k(x) = \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{x - x_i}{x_k - x_i}$$
$L_k(x)$ là hàm cơ sở, $L_k(x_k) = 1$, $L_k(x_i) = 0$ nếu $i \neq k$.
- **Phương pháp tính:**
  1. Tính từng $L_k(x)$ dựa trên các nút x_i.
  2. Nhân y_k với $L_k(x)$ và cộng lại.
  - **Ví dụ:** Với $(2, 4), (5, 1)$:
    - $L_0(x) = \frac{x-5}{2-5} = -\frac{1}{3}(x-5)$,
    - $L_1(x) = \frac{x-2}{5-2} = \frac{1}{3}(x-2)$,
    - $P(x) = 4 \cdot \left(-\frac{1}{3}(x-5)\right) + 1 \cdot \left(\frac{1}{3}(x-2)\right) = -x + 6$.
- **Công thức sai số:** Nếu $f \in C^{n+1}[a, b]$:
$$f(x) - P_n(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)(x - x_1)\ldots(x - x_n)$$
$\xi(x)$ nằm giữa các nút.
- **Nhận xét:** Dễ tính với ít điểm, nhưng khi n lớn, đa thức dao động mạnh (hiện tượng Runge).

3.2 - Xấp xỉ Dữ liệu và Phương pháp Neville

- **Ý tưởng:** Dùng bảng giá trị đệ quy để tính nội suy mà không cần viết rõ đa thức.
- **Công thức:** Công thức đệ quy của Neville:
$$P_{i,i+1,\ldots,i+k}(x) = \frac{(x - x_{i+k})P_{i,i+1,\ldots,i+k-1}(x) - (x - x_i)P_{i+1,\ldots,i+k}(x)}{x_i - x_{i+k}}$$
Trong đó $P_i(x) = f(x_i)$.
- **Phương pháp tính:**
  1. Lập bảng Neville với cột 0 là $f(x_i)$.
  2. Tính từng cột tiếp theo bằng công thức trên.
  - **Ví dụ:** Với $x_0 = 1.3$, $f(1.3) = 0.6200860$, $x_1 = 1.6$, $f(1.6) = 0.4554022$, tính $P_{0,1}(1.5)$:
    - $P_{0,1}(1.5) = \frac{(1.5-1.6)\cdot 0.6200860 - (1.5-1.3)\cdot 0.4554022}{1.3-1.6} = 0.5102968$.
- **Công thức sai số:** Tương tự Lagrange, phụ thuộc vào bậc đa thức:
$$f(x) - P_n(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)\ldots(x - x_n)$$
- **Nhận xét:** Hiệu quả hơn Lagrange khi chỉ cần giá trị tại một điểm, nhưng vẫn là đa thức toàn cục, dễ dao động với nhiều điểm.

3.3 - Sai phân chia (Divided Differences)

- **Ý tưởng:** Dùng sai phân để xây dựng đa thức Newton, dễ mở rộng khi thêm điểm.
- **Công thức:** Đa thức Newton bậc n:
  $$P_n(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \ldots + f[x_0, \ldots, x_n](x - x_0)\ldots(x - x_{n-1})$$
  Sai phân:
  - $f[x_i] = f(x_i)$,
  - $f[x_i, x_{i+1}] = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}$,
  - $f[x_i, \ldots, x_{i+k}] = \frac{f[x_{i+1}, \ldots, x_{i+k}] - f[x_i, \ldots, x_{i+k-1}]}{x_{i+k} - x_i}$.
- **Phương pháp tính:**
  1. Lập bảng sai phân.
  2. Dùng hệ số sai phân để viết $P_n(x)$.
  - **Ví dụ:** Với $x_0 = 1, f(1) = 1, x_1 = 2, f(2) = 4, x_2 = 3, f(3) = 9$:
    - $f[1, 2] = 3, f[2, 3] = 5, f[1, 2, 3] = 1$,
    - $P_2(x) = 1 + 3(x - 1) + 1(x - 1)(x - 2) = x^2$.
- **Công thức sai số:**
  $$f(x) - P_n(x) = \frac{f^{(n+1)}(\xi(x))}{(n + 1)!}(x - x_0)(x - x_1)\ldots(x - x_n)$$
- **Nhận xét:** Linh hoạt hơn Lagrange, nhưng đa thức bậc cao vẫn dao động nếu n lớn.

3.4 - Nội suy Hermite

- **Ý tưởng:** Khớp cả giá trị hàm và đạo hàm tại các nút.
- **Công thức:** Với $m + 1$ nút, đa thức Hermite $H(x)$ bậc tối đa $N = \sum(n_i + 1) - 1$:
  - Dùng sai phân với nút trùng:
    - $f[x_i] = f(x_i), f[x_i, x_i] = f'(x_i)$,
    - $H(x) = f[x_0] + f[x_0, x_0](x - x_0) + f[x_0, x_0, x_1](x - x_0)^2 + \ldots$
- **Phương pháp tính:**
  1. Lập bảng sai phân với nút trùng (dùng đạo hàm).
  2. Viết $H(x)$ dạng Newton.
  - **Ví dụ:** Với $x_0 = 0, f(0) = 1, f'(0) = 1$ $(f(x) = e^x)$:
    - $f[0] = 1, f[0, 0] = 1$,
    - $H_1(x) = 1 + 1(x - 0) = 1 + x$.
- **Công thức sai số:** Với $n = \sum(n_i + 1) - 1$:
  $$f(x) - H(x) = \frac{f^{(n+1)}(\xi(x))}{(n + 1)!} \prod_{i=0}^{m} (x - x_i)^{n_i+1}$$
  - Ví dụ: $H_1(0.5) = 1.5$, sai số $\frac{e^\xi}{2}(0.5)^2 \approx 0.1487$.
- **Nhận xét:** Mượt hơn nhờ đạo hàm, nhưng cần biết $f'(x)$, và đa thức bậc cao vẫn có thể dao động.

Bối cảnh dẫn vào Spline bậc ba (3.5)

Qua 4 phần, ta thấy:

- **Lagrange và Neville:** Dùng đa thức toàn cục, dễ dao động khi số điểm lớn.
- **Newton:** Linh hoạt hơn, nhưng vẫn là đa thức toàn cục.
- **Hermite:** Kiểm soát độ mượt bằng đạo hàm, nhưng vẫn gặp vấn đề với bậc cao.

Vấn đề chung: Đa thức bậc cao (ví dụ bậc 9 cho 10 điểm) thường dao động mạnh giữa các nút, không phản ánh thực tế. Ví dụ, nội suy $f(x) = \sin(x)$ với 10 điểm bằng đa thức bậc 9 có thể tạo "đỉnh nhọn" bất thường.

Giải pháp: Thay vì dùng một đa thức duy nhất, ta chia dữ liệu thành các đoạn nhỏ, mỗi đoạn là đa thức bậc thấp (bậc 3), và nối chúng sao cho liên tục đến đạo hàm bậc 2. Đây là ý tưởng của **spline bậc ba**:

- **Công thức:** $S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3$.
- **Đặc điểm:** Mượt mà, tránh dao động, phù hợp với dữ liệu lớn.

Phần 3.5: Nội suy Spline Bậc Ba

Giới thiệu trực quan

Trong các phần trước, chúng ta đã thấy rằng đa thức nội suy bậc cao (như Lagrange hay Newton) có thể dao động mạnh khi số điểm dữ liệu lớn, dẫn đến kết quả không thực tế. **Spline bậc ba (cubic spline)** khắc phục vấn đề này bằng cách chia khoảng dữ liệu thành các đoạn nhỏ, mỗi đoạn được xấp xỉ bởi một đa thức bậc ba. Điều này giống như vẽ một đường cong bằng cách nối nhiều đoạn dây mềm dẻo, thay vì dùng một sợi dây dài duy nhất.

Hãy tưởng tượng bạn muốn vẽ đường viền của một con vịt: thay vì dùng một đa thức bậc 20 dao động lung tung, bạn có thể dùng nhiều đoạn spline bậc ba để tạo ra một đường cong mượt mà, khớp với dữ liệu mà không bị "lệch lạc".

Spline bậc ba là gì?

Spline bậc ba là một hàm phân đoạn (piecewise function), trong đó:

- Mỗi đoạn giữa hai nút là một đa thức bậc ba.
- Hàm liên tục đến đạo hàm bậc 2 tại các nút (đảm bảo mượt mà).

Có hai loại chính:

1. **Spline tự nhiên (natural spline):** Đạo hàm bậc 2 bằng 0 tại hai đầu mút.
2. **Spline kẹp (clamped spline):** Đạo hàm bậc 1 tại hai đầu mút được chỉ định.

Cách xây dựng spline bậc ba

Cho $n + 1$ nút $x_0 < x_1 < ... < x_n$, spline $S(x)$ gồm n đoạn $S_j(x)$ trên $[x_j, x_{j+1}]$, dạng:

$$S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3$$

- Điều kiện:
  - $S(x_j) = f(x_j)$ (nội suy tại các điểm),
  - $S_j(x_{j+1}) = S_{j+1}(x_{j+1})$ (liên tục hàm),
  - $S'_j(x_{j+1}) = S'_{j+1}(x_{j+1})$ (liên tục đạo hàm bậc 1),
  - $S''_j(x_{j+1}) = S''_{j+1}(x_{j+1})$ (liên tục đạo hàm bậc 2).

Spline tự nhiên

- Thêm điều kiện: $S''(x_0) = S''(x_n) = 0$.
- Hệ phương trình: Giải c_j từ S'', rồi tính a_j, b_j, d_j.

Ví dụ 1: Spline tự nhiên Bài toán: Tìm spline tự nhiên cho dữ liệu: $f(0) = 0, f(1) = 1, f(2) = 2$.

Giải:

- Có 2 đoạn: $S_0(x)$ trên $[0, 1]$, $S_1(x)$ trên $[1, 2]$.
- Dạng:
  - $S_0(x) = a_0 + b_0x + c_0x^2 + d_0x^3$,
  - $S_1(x) = a_1 + b_1(x - 1) + c_1(x - 1)^2 + d_1(x - 1)^3$.

- Điều kiện:

    1. $S_0(0) = 0 \Rightarrow a_0 = 0$,
    2. $S_0(1) = 1 \Rightarrow a_0 + b_0 + c_0 + d_0 = 1$,
    3. $S_1(1) = 1 \Rightarrow a_1 = 1$,
    4. $S_1(2) = 2 \Rightarrow a_1 + b_1 + c_1 + d_1 = 2$,
    5. $S_0'(1) = S_1'(1) \Rightarrow b_0 + 2c_0 + 3d_0 = b_1$,
    6. $S_0''(1) = S_1''(1) \Rightarrow 2c_0 + 6d_0 = 2c_1$,
    7. $S_0''(0) = 0 \Rightarrow 2c_0 = 0 \Rightarrow c_0 = 0$,
    8. $S_1''(2) = 0 \Rightarrow 2c_1 + 6d_1 = 0$.

- Giải:

    - Từ (1): $a_0 = 0$,
    - Từ (7): $c_0 = 0$,
    - Từ (2): $b_0 + d_0 = 1$,
    - Từ (3): $a_1 = 1$,
    - Từ (4): $1 + b_1 + c_1 + d_1 = 2 \Rightarrow b_1 + c_1 + d_1 = 1$,
    - Từ (5): $b_0 + 3d_0 = b_1$,
    - Từ (6): $6d_0 = 2c_1 \Rightarrow 3d_0 = c_1$,
    - Từ (8): $2c_1 + 6d_1 = 0 \Rightarrow c_1 + 3d_1 = 0$.

- Từ $c_1 + 3d_1 = 0 \Rightarrow c_1 = -3d_1$,
- Từ $3d_0 = c_1 \Rightarrow 3d_0 = -3d_1 \Rightarrow d_0 = -d_1$,
- Từ $b_1 + c_1 + d_1 = 1 \Rightarrow b_1 - 3d_1 + d_1 = 1 \Rightarrow b_1 - 2d_1 = 1$,
- Từ $b_0 + 3d_0 = b_1 \Rightarrow b_0 + 3(-d_1) = b_1 \Rightarrow b_0 - 3d_1 = b_1$,
- Từ $b_0 + d_0 = 1 \Rightarrow b_0 - d_1 = 1$.
- Giải hệ:

    - $b_0 - d_1 = 1$,
    - $b_0 - 3d_1 = b_1$,
    - $b_1 - 2d_1 = 1$.
    - Từ (3): $b_1 = 1 + 2d_1$,
    - Từ (2): $b_0 - 3d_1 = 1 + 2d_1 \Rightarrow b_0 = 1 + 5d_1$,
    - Từ (1): $1 + 5d_1 - d_1 = 1 \Rightarrow 1 + 4d_1 = 1 \Rightarrow 4d_1 = 0 \Rightarrow d_1 = 0$.

- Vậy: $d_1 = 0, c_1 = 0, b_1 = 1, d_0 = 0, b_0 = 1, a_0 = 0, a_1 = 1$.
- Kết quả: $S_0(x) = x$, $S_1(x) = 1 + (x - 1) = x$.

Spline kẹp

- Điều kiện: $S'(x_0) = f'(x_0)$, $S'(x_n) = f'(x_n)$.

Ví dụ 2: Spline kẹp Bài toán: Như trên, nhưng $S'(0) = 1$, $S'(2) = 1$.

Giải:

- Điều kiện thêm:

    9. $S_0'(0) = 1 \Rightarrow b_0 = 1$,
    10. $S_1'(2) = 1 \Rightarrow b_1 + 2c_1 + 3d_1 = 1$.

- Giải lại:

    - $b_0 = 1$,
    - $1 + c_0 + d_0 = 1 \Rightarrow c_0 + d_0 = 0$,
    - $b_1 + c_1 + d_1 = 1$,
    - $1 + 2c_0 + 3d_0 = b_1$,
    - $2c_0 + 6d_0 = 2c_1$,
    - $b_1 + 2c_1 + 3d_1 = 1$.

- Từ $c_0 + d_0 = 0 \Rightarrow c_0 = -d_0$,
- Từ $2c_0 + 6d_0 = 2c_1 \Rightarrow -2d_0 + 6d_0 = 2c_1 \Rightarrow 4d_0 = 2c_1 \Rightarrow 2d_0 = c_1$,
- Từ $1 + 2c_0 + 3d_0 = b_1 \Rightarrow 1 - 2d_0 + 3d_0 = b_1 \Rightarrow 1 + d_0 = b_1$,
- Từ $b_1 + c_1 + d_1 = 1$,

- Từ $b_1 + 2c_1 + 3d_1 = 1$.
- Đặt $d_0 = t$:
  - $c_0 = -t$,
  - $c_1 = 2t$,
  - $b_1 = 1 + t$,
  - $1 + t + 2t + d_1 = 1 \Rightarrow 1 + 3t + d_1 = 1 \Rightarrow d_1 = -3t$,
  - $1 + t + 4t - 9t = 1 \Rightarrow 1 - 4t = 1 \Rightarrow -4t = 0 \Rightarrow t = 0$.
- Kết quả: $S_0(x) = x$, $S_1(x) = x$.

Định lý 3.13: Sai số spline kẹp

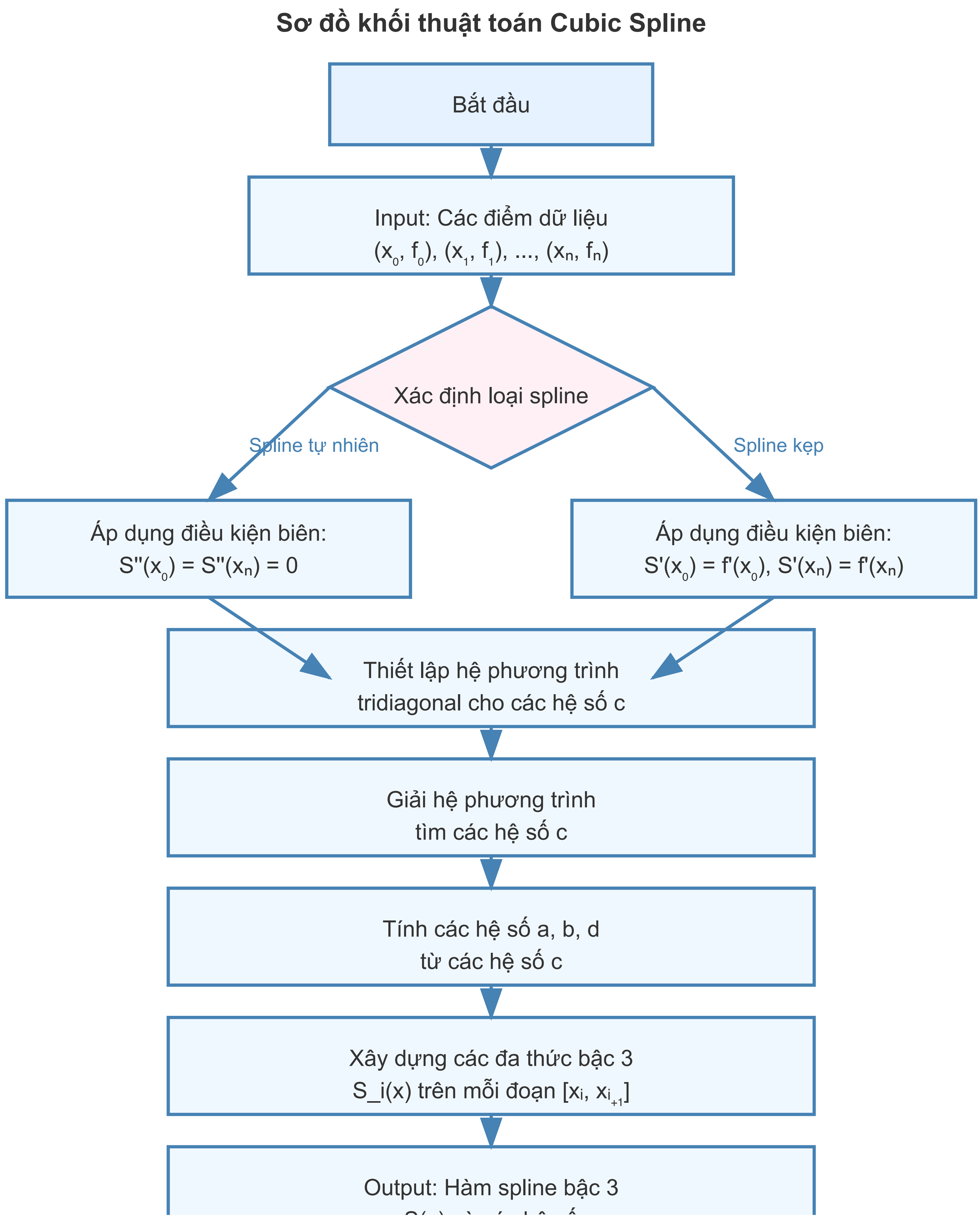
Nội dung: Với $f \in C^4[a, b]$, sai số:

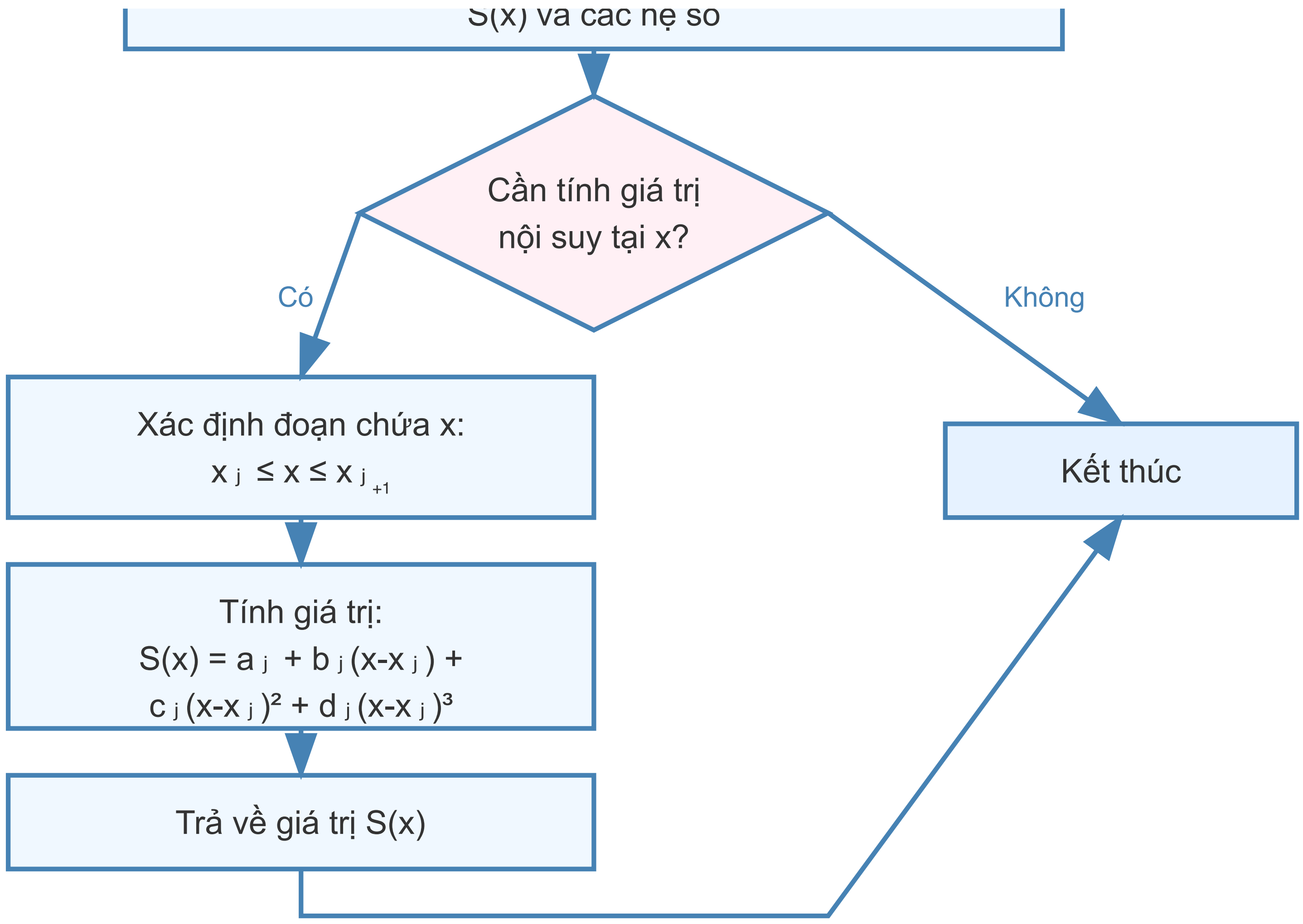
$$|f(x) - S(x)| \leq \frac{5M}{384} \max(x_{j+1} - x_j)^4, \quad M = \max |f^{(4)}(x)|$$

Ví dụ 3: Với $f(x) = e^x$, $x_0 = 0$, $x_1 = 1$, $h = 1$, $M = e$:

- Sai số $\leq \frac{5e}{384} \cdot 1^4 \approx 0.035$.

Sơ đồ khối





Minh hoạ Spline tự nhiên và Spline kẹp

```
import numpy as np
import plotly.graph_objects as go
from scipy.interpolate import CubicSpline
from IPython.display import display, clear_output
import ipywidgets as widgets

# Đảm bảo đã cài đặt các thư viện cần thiết
!pip install -q plotly ipywidgets

# Cấu hình hiển thị cho Colab
from google.colab import output
output.enable_custom_widget_manager()

# Hàm tính toán natural spline
def calculate_natural_spline(x, y, num_points=100):
    cs = CubicSpline(x, y, bc_type='natural')
    x_new = np.linspace(min(x), max(x), num_points)
    y_new = cs(x_new)
    return x_new, y_new

# Hàm tính toán clamped spline với độ dốc tại hai đầu
def calculate_clamped_spline(x, y, start_slope, end_slope, num_points=100):
    cs = CubicSpline(x, y, bc_type=((1, start_slope), (1, end_slope)))
    x_new = np.linspace(min(x), max(x), num_points)
    y_new = cs(x_new)
    return x_new, y_new, cs

# Hàm tính đạo hàm đầu tiên của spline
def calculate_derivative(cs, x):
    return cs(x, 1)

# Hàm tạo đồ thị
def create_spline_plot(start_slope, end_slope):
    # Các điểm dữ liệu
    x = np.array([0, 2, 4, 6, 8])
    y = np.array([0, 3, 0, 2, 0])

    # Tính toán natural spline
    x_natural, y_natural = calculate_natural_spline(x, y)

    # Tính toán clamped spline
```

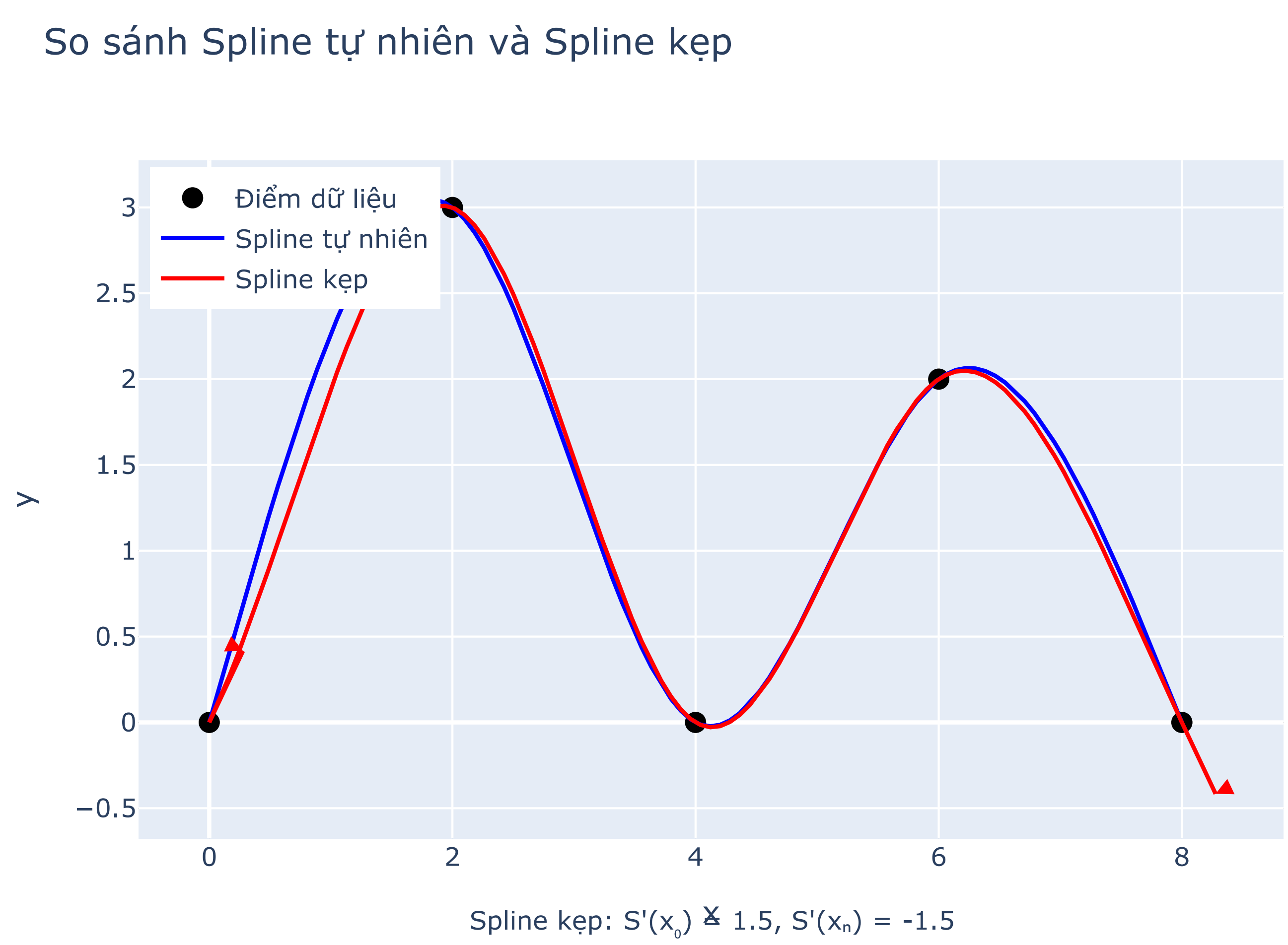
```
x_clamped, y_clamped, cs_clamped = calculate_clamped_spline(x, y, start_slope, end_slope)

# Tính đạo hàm tại điểm đầu và cuối của spline kẹp
start_derivative = calculate_derivative(cs_clamped, x[0])
end_derivative = calculate_derivative(cs_clamped, x[-1])

# Tạo các điểm cho mũi tên chỉ đạo hàm
arrow_length = 0.5

# Chuẩn hóa vector mũi tên
start_dx = arrow_length / np.sqrt(1 + start_derivative**2)
start_dy = start_derivative * start_dx

end_dx = arrow_length / np.sqrt(1 + end_derivative**2)
end_dy = end_derivative * end_dx

# Tạo đồ thị
fig = go.Figure()

# Các điểm dữ liệu
fig.add_trace(go.Scatter(
    x=x, y=y,
    mode='markers',
    marker=dict(size=10, color='black'),
    name='Điểm dữ liệu'
))

# Natural spline
fig.add_trace(go.Scatter(
    x=x_natural, y=y_natural,
    mode='lines',
    line=dict(color='blue', width=2),
    name='Spline tự nhiên'
))

# Clamped spline
fig.add_trace(go.Scatter(
    x=x_clamped, y=y_clamped,
    mode='lines',
    line=dict(color='red', width=2),
    name='Spline kẹp'
))

# Mũi tên chỉ đạo hàm tại điểm đầu
fig.add_trace(go.Scatter(
    x=[x[0], x[0] + start_dx],
    y=[y[0], y[0] + start_dy],
    mode='lines+markers',
    line=dict(color='red', width=2),
    marker=dict(size=8, symbol='arrow-right', angleref='previous', color='red'),
    showlegend=False
))

# Mũi tên chỉ đạo hàm tại điểm cuối
fig.add_trace(go.Scatter(
    x=[x[-1], x[-1] + end_dx],
    y=[y[-1], y[-1] + end_dy],
    mode='lines+markers',
    line=dict(color='red', width=2),
    marker=dict(size=8, symbol='arrow-right', angleref='previous', color='red'),
    showlegend=False
))

# Cấu hình đồ thị
fig.update_layout(
    title='So sánh Spline tự nhiên và Spline kẹp',
    xaxis_title='x',
    yaxis_title='y',
    width=700,
    height=500,
    legend=dict(
        yanchor="top",
        y=0.99,
        xanchor="left",
```

```
            x=0.01
        ),
        annotations=[
            dict(
                x=0.5, y=-0.15,
                xref='paper', yref='paper',
                text=f'Spline kẹp: S\'(x₀) = {start_slope}, S\'(xₙ) = {end_slope}',
                showarrow=False
            )
        ]
    )

    return fig

# Phiên bản hoạt động trên Colab
start_slope = 1.5
end_slope = -1.5

# Tạo sliders
start_slider = widgets.FloatSlider(
    value=start_slope,
    min=-3.0,
    max=3.0,
    step=0.1,
    description='Độ dốc đầu:',
    continuous_update=False
)

end_slider = widgets.FloatSlider(
    value=end_slope,
    min=-3.0,
    max=3.0,
    step=0.1,
    description='Độ dốc cuối:',
    continuous_update=False
)

# Hàm cập nhật đồ thị - phiên bản cho Colab
def update_plot(change):
    # Xóa output cũ
    clear_output(wait=True)

    # Hiển thị các slider
    display(start_slider)
    display(end_slider)

    # Tạo và hiển thị đồ thị mới
    fig = create_spline_plot(start_slider.value, end_slider.value)
    fig.show()

# Kết nối hàm cập nhật với slider
start_slider.observe(update_plot, names='value')
end_slider.observe(update_plot, names='value')

# Hiển thị ban đầu
display(start_slider)
display(end_slider)
fig = create_spline_plot(start_slope, end_slope)
fig.show()

# Phiên bản thay thế sử dụng nút cập nhật (nếu cách trên không hoạt động)
def alternative_approach():
    # Tạo widgets
    start_slider_alt = widgets.FloatSlider(
        value=1.5,
        min=-3.0,
        max=3.0,
        step=0.1,
        description='Độ dốc đầu:',
    )

    end_slider_alt = widgets.FloatSlider(
        value=-1.5,
        min=-3.0,
```

```
        max=3.0,
        step=0.1,
        description='Độ dốc cuối:',
    )

    update_button = widgets.Button(description="Cập nhật đồ thị")
    output_widget = widgets.Output()

    def on_button_clicked(b):
        with output_widget:
            clear_output(wait=True)
            fig = create_spline_plot(start_slider_alt.value, end_slider_alt.value)
            fig.show()

    update_button.on_click(on_button_clicked)

    # Hiển thị UI
    display(widgets.VBox([
        widgets.Label(value="Sử dụng các thanh trượt để điều chỉnh, sau đó nhấn nút Cập nhật đồ thị:"),
        start_slider_alt,
        end_slider_alt,
        update_button,
        output_widget
    ]))

    # Hiển thị đồ thị ban đầu
    with output_widget:
        fig = create_spline_plot(1.5, -1.5)
        fig.show()

# alternative_approach()
```

Độ dốc đầu: 1.50

Độ dốc cuối: −1.50



Code Python cho giải spline tự nhiên và spline kẹp

```
def solve_natural_cubic_spline(x, y):
    n = len(x)
    h = np.diff(x)  # Độ dài từng đoạn [x_j, x_{j+1}]
    alpha = np.zeros(n)

    # Tính toán alpha dựa trên đạo hàm cấp 2 (dùng cho hệ phương trình)
    for i in range(1, n - 1):
```

```
        # Điều kiện (e): đảm bảo đạo hàm cấp 2 liên tục tại các điểm nút
        alpha[i] = (3/h[i])*(y[i+1] - y[i]) - (3/h[i-1])*(y[i] - y[i-1])

    # Khởi tạo mảng cho hệ ba đường chéo
    l = np.ones(n)
    mu = np.zeros(n)
    z = np.zeros(n)
    c = np.zeros(n)
    b = np.zeros(n - 1)
    d = np.zeros(n - 1)
    a = y[:-1].copy()  # Hệ số a của đa thức, thỏa mãn điều kiện (b)

    # Giải hệ tam giác theo phương pháp Thomas
    for i in range(1, n - 1):
        l[i] = 2*(x[i+1] - x[i-1]) - h[i-1]*mu[i-1]
        mu[i] = h[i] / l[i]
        z[i] = (alpha[i] - h[i-1]*z[i-1]) / l[i]

    # Điều kiện biên Natural spline: (f.i) S''(x_0) = S''(x_n) = 0
    l[-1] = 1
    z[-1] = 0
    c[-1] = 0

    # Quay lui để tính hệ số c, sau đó b và d
    for j in range(n - 2, -1, -1):
        c[j] = z[j] - mu[j]*c[j+1]
        b[j] = (y[j+1] - y[j])/h[j] - h[j]*(c[j+1] + 2*c[j])/3  # Điều kiện (d)
        d[j] = (c[j+1] - c[j]) / (3*h[j])                       # Điều kiện (e)

    print("Các phương trình spline tự nhiên:")
    for j in range(n - 1):
        print(f"S_{j}(x) = {a[j]:.4f} + {b[j]:.4f}(x - {x[j]:.1f}) + {c[j]:.4f}(x - {x[j]:.1f})² + {d[j]:.4f}(x - {

    return a, b, c[:-1], d  # Trả về hệ số của đa thức S_j(x)

def solve_clamped_cubic_spline(x, y, fp0=0, fpn=0): # mặc định điều kiện biên kẹp là (0,0)
    n = len(x)
    h = np.diff(x)
    alpha = np.zeros(n)

    # Điều kiện biên chặn: (f.ii) S'(x_0) = f'(x_0), S'(x_n) = f'(x_n)
    alpha[0] = 3 * (y[1] - y[0]) / h[0] - 3 * fp0
    alpha[-1] = 3 * fpn - 3 * (y[-1] - y[-2]) / h[-1]

    # Tính các alpha còn lại từ điều kiện đạo hàm cấp 2 liên tục (e)
    for i in range(1, n - 1):
        alpha[i] = (3/h[i])*(y[i+1] - y[i]) - (3/h[i-1])*(y[i] - y[i-1])

    # Khởi tạo mảng cho hệ ba đường chéo
    l = np.ones(n)
    mu = np.zeros(n)
    z = np.zeros(n)
    c = np.zeros(n)
    b = np.zeros(n - 1)
    d = np.zeros(n - 1)
    a = y[:-1].copy()  # Hệ số a của đa thức, thỏa mãn điều kiện (b)

    # Gán điều kiện biên trái: đạo hàm cấp 1 tại x_0
    l[0] = 2*h[0]
    mu[0] = 0.5
    z[0] = alpha[0] / l[0]

    # Giải hệ tam giác theo phương pháp Thomas (dựa trên liên tục đạo hàm bậc 1 và 2)
    for i in range(1, n - 1):
        l[i] = 2*(x[i+1] - x[i-1]) - h[i-1]*mu[i-1]
        mu[i] = h[i] / l[i]
        z[i] = (alpha[i] - h[i-1]*z[i-1]) / l[i]

    # Gán điều kiện biên phải: đạo hàm cấp 1 tại x_n
    l[-1] = h[-1] * (2 - mu[-2])
    z[-1] = (alpha[-1] - h[-2]*z[-2]) / l[-1]
    c[-1] = z[-1]

    for j in range(n - 2, -1, -1):
```

```
        c[j] = z[j] - mu[j]*c[j+1]
        b[j] = (y[j+1] - y[j])/h[j] - h[j]*(c[j+1] + 2*c[j])/3
        d[j] = (c[j+1] - c[j]) / (3*h[j])

    print("Các phương trình spline kẹp:")
    for j in range(n - 1):
        print(f"S_{j}(x) = {a[j]:.4f} + {b[j]:.4f}(x - {x[j]:.1f}) + {c[j]:.4f}(x - {x[j]:.1f})² + {d[j]:.4f}(x - {

    return a, b, c[:-1], d
```

Các code bổ trợ cho kiểm tra sai số và minh hoạ

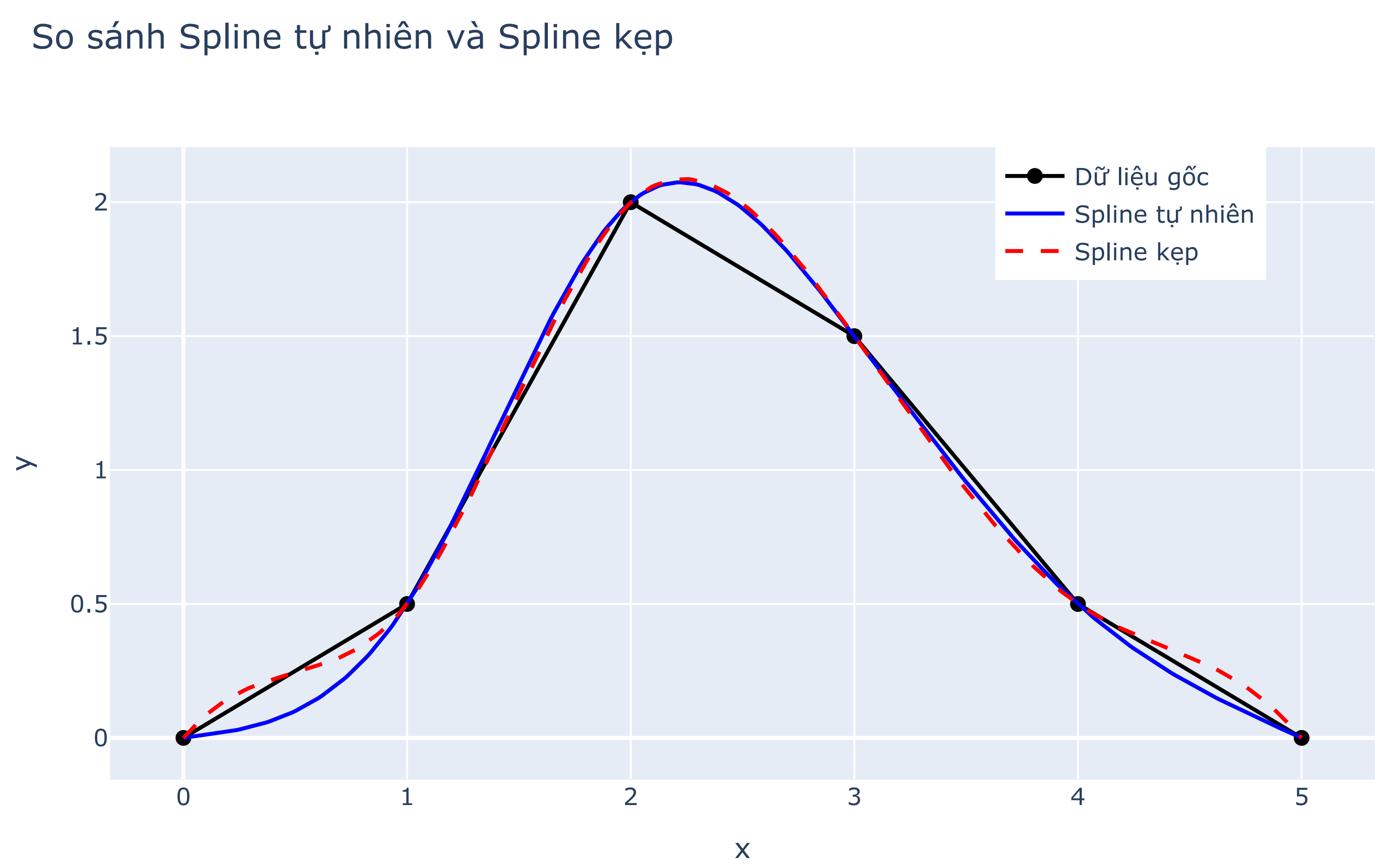
```
def check_all_nodes(x_nodes, y_nodes, a, b, c, d):
    """
    Kiểm tra điều kiện (b) và (c) của spline:
    S_j(x_j) = f(x_j)
    S_j(x_{j+1}) = f(x_{j+1})
    Bao gồm cả biên và các điểm trung gian
    """
    n = len(x_nodes)
    max_error = 0

    print("🔎 Kiểm tra nội suy tại tất cả các điểm nút:")

    for j in range(n - 1):
        xj = x_nodes[j]
        xj1 = x_nodes[j + 1]
        fj = y_nodes[j]
        fj1 = y_nodes[j + 1]

        # Đánh giá S_j(x_j)
        Sj_xj = a[j] + b[j]*0 + c[j]*0**2 + d[j]*0**3
        err_left = abs(Sj_xj - fj)

        # Đánh giá S_j(x_{j+1})
        dx = xj1 - xj
        Sj_xj1 = a[j] + b[j]*dx + c[j]*dx**2 + d[j]*dx**3
        err_right = abs(Sj_xj1 - fj1)

        max_error = max(max_error, err_left, err_right)

        print(f"  Đoạn {j}:")
        print(f"    S_{j}(x_{j}) = {Sj_xj:.6f} | f(x_{j}) = {fj:.6f} | Sai số = {err_left:.2e}")
        print(f"    S_{j}(x_{j+1}) = {Sj_xj1:.6f} | f(x_{j+1}) = {fj1:.6f} | Sai số = {err_right:.2e}")

    print(f"\n✅ Sai số lớn nhất tại các điểm nút: {max_error:.2e}")
    return max_error

x_data = np.array([0, 0.25, 0.5, 1, 1.25])
y_data = np.array([0, 22.98, 47.23, 97.49, 122.66])

a_nat, b_nat, c_nat, d_nat = solve_natural_cubic_spline(x_data, y_data)

# Kiểm tra spline tự nhiên
check_all_nodes(x_data, y_data, a_nat, b_nat, c_nat, d_nat)

a_cla, b_cla, c_cla, d_cla = solve_clamped_cubic_spline(x_data, y_data, 0, 0)

# Kiểm tra spline kẹp
check_all_nodes(x_data, y_data, a_cla, b_cla, c_cla, d_cla)
```

```
Các phương trình spline tự nhiên:
S_0(x) = 0.0000 + 90.7580(x - 0.0) + 0.0000(x - 0.0)² + 18.5915(x - 0.0)³
S_1(x) = 22.9800 + 94.2439(x - 0.2) + 13.9436(x - 0.2)² + -11.6774(x - 0.2)³
S_2(x) = 47.2300 + 99.0262(x - 0.5) + 5.1856(x - 0.5)² + -4.3961(x - 0.5)³
S_3(x) = 97.4900 + 100.9148(x - 1.0) + -1.4085(x - 1.0)² + 1.8780(x - 1.0)³
🔎 Kiểm tra nội suy tại tất cả các điểm nút:
  Đoạn 0:
    S_0(x_0) = 0.000000 | f(x_0) = 0.000000 | Sai số = 0.00e+00
    S_0(x_1) = 22.980000 | f(x_1) = 22.980000 | Sai số = 0.00e+00
  Đoạn 1:
    S_1(x_1) = 22.980000 | f(x_1) = 22.980000 | Sai số = 0.00e+00
    S_1(x_2) = 47.230000 | f(x_2) = 47.230000 | Sai số = 0.00e+00
  Đoạn 2:
    S_2(x_2) = 47.230000 | f(x_2) = 47.230000 | Sai số = 0.00e+00
    S_2(x_3) = 97.490000 | f(x_3) = 97.490000 | Sai số = 1.42e-14
  Đoạn 3:
    S_3(x_3) = 97.490000 | f(x_3) = 97.490000 | Sai số = 0.00e+00
    S_3(x_4) = 122.660000 | f(x_4) = 122.660000 | Sai số = 0.00e+00

✅ Sai số lớn nhất tại các điểm nút: 1.42e-14
Các phương trình spline kẹp:
S_0(x) = 0.0000 + 0.0000(x - 0.0) + 620.5733(x - 0.0)² + -1011.5734(x - 0.0)³
S_1(x) = 22.9800 + 120.6167(x - 0.2) + -138.1067(x - 0.2)² + 174.5602(x - 0.2)³
S_2(x) = 47.2300 + 84.2933(x - 0.5) + -7.1866(x - 0.5)² + 79.2797(x - 0.5)³
S_3(x) = 97.4900 + 136.5666(x - 1.0) + 111.7330(x - 1.0)² + -1021.1174(x - 1.0)³
🔎 Kiểm tra nội suy tại tất cả các điểm nút:
  Đoạn 0:
    S_0(x_0) = 0.000000 | f(x_0) = 0.000000 | Sai số = 0.00e+00
    S_0(x_1) = 22.980000 | f(x_1) = 22.980000 | Sai số = 3.55e-15
  Đoạn 1:
    S_1(x_1) = 22.980000 | f(x_1) = 22.980000 | Sai số = 0.00e+00
    S_1(x_2) = 47.230000 | f(x_2) = 47.230000 | Sai số = 7.11e-15
  Đoạn 2:
    S_2(x_2) = 47.230000 | f(x_2) = 47.230000 | Sai số = 0.00e+00
    S_2(x_3) = 97.490000 | f(x_3) = 97.490000 | Sai số = 0.00e+00
  Đoạn 3:
    S_3(x_3) = 97.490000 | f(x_3) = 97.490000 | Sai số = 0.00e+00
    S_3(x_4) = 122.660000 | f(x_4) = 122.660000 | Sai số = 2.84e-14

✅ Sai số lớn nhất tại các điểm nút: 2.84e-14
np.float64(2.842170943040401e-14)
```

```python
import numpy as np
import plotly.graph_objects as go

def evaluate_spline(x_query, x_nodes, a, b, c, d):
    """
    Tính giá trị spline tại các điểm x_query
    """
    y_query = []
    n = len(x_nodes) - 1
    for xq in x_query:
        for j in range(n):
            if x_nodes[j] <= xq <= x_nodes[j + 1]:
                dx = xq - x_nodes[j]
                y = a[j] + b[j] * dx + c[j] * dx**2 + d[j] * dx**3
                y_query.append(y)
                break
    return np.array(y_query)

def plot_splines(x_data, y_data, x_dense, y_nat, y_cla):
    """
    Hàm vẽ spline tự nhiên và spline kẹp
    """
    fig = go.Figure()

    # Dữ liệu gốc
    fig.add_trace(go.Scatter(x=x_data, y=y_data, mode='markers+lines',
                             name='Dữ liệu gốc', marker=dict(size=8, color='black')))

    # Spline tự nhiên
    fig.add_trace(go.Scatter(x=x_dense, y=y_nat, mode='lines',
                             name='Spline tự nhiên', line=dict(color='blue')))

    # Spline kẹp
    fig.add_trace(go.Scatter(x=x_dense, y=y_cla, mode='lines',
                             name='Spline kẹp', line=dict(color='red', dash='dash')))
```

```python
    fig.update_layout(title='So sánh Spline tự nhiên và Spline kẹp',
                      xaxis_title='x',
                      yaxis_title='y',
                      legend=dict(x=0.7, y=1),
                      width=800, height=500)

    fig.show()

def analyze(x_data, y_data, fp0=0, fpn=0, num_points=300):
    """
    Hàm tổng hợp: giải spline, nội suy và vẽ đồ thị
    """
    # Giải spline
    a_nat, b_nat, c_nat, d_nat = solve_natural_cubic_spline(x_data, y_data)
    a_cla, b_cla, c_cla, d_cla = solve_clamped_cubic_spline(x_data, y_data, fp0, fpn)

    # Check sai số
    check_all_nodes(x_data, y_data, a_nat, b_nat, c_nat, d_nat)
    check_all_nodes(x_data, y_data, a_cla, b_cla, c_cla, d_cla)

    # Tạo điểm x để nội suy
    x_dense = np.linspace(x_data[0], x_data[-1], num_points)
    y_nat = evaluate_spline(x_dense, x_data, a_nat, b_nat, c_nat, d_nat)
    y_cla = evaluate_spline(x_dense, x_data, a_cla, b_cla, c_cla, d_cla)

    # Vẽ hình
    plot_splines(x_data, y_data, x_dense, y_nat, y_cla)

x_data = np.array([0, 1, 2, 3, 4, 5])
y_data = np.array([0, 0.5, 2, 1.5, 0.5, 0])
fp0 = 1   # S'(x0)
fpn = -1  # S'(xn)

analyze(x_data, y_data, fp0, fpn)
```

```
Các phương trình spline tự nhiên:
S_0(x) = 0.0000 + 0.1005(x - 0.0) + 0.0000(x - 0.0)² + 0.3995(x - 0.0)³
S_1(x) = 0.5000 + 1.2990(x - 1.0) + 1.1986(x - 1.0)² + -0.9976(x - 1.0)³
S_2(x) = 2.0000 + 0.7033(x - 2.0) + -1.7943(x - 2.0)² + 0.5909(x - 2.0)³
S_3(x) = 1.5000 + -1.1124(x - 3.0) + -0.0215(x - 3.0)² + 0.1340(x - 3.0)³
S_4(x) = 0.5000 + -0.7536(x - 4.0) + 0.3804(x - 4.0)² + -0.1268(x - 4.0)³
Các phương trình spline kẹp:
S_0(x) = 0.0000 + 1.0000(x - 0.0) + -1.5550(x - 0.0)² + 1.0550(x - 0.0)³
S_1(x) = 0.5000 + 1.0550(x - 1.0) + 1.6100(x - 1.0)² + -1.1651(x - 1.0)³
S_2(x) = 2.0000 + 0.7799(x - 2.0) + -1.8852(x - 2.0)² + 0.6053(x - 2.0)³
S_3(x) = 1.5000 + -1.1746(x - 3.0) + -0.0694(x - 3.0)² + 0.2440(x - 3.0)³
S_4(x) = 0.5000 + -0.5813(x - 4.0) + 0.6627(x - 4.0)² + -0.5813(x - 4.0)³
🔎 Kiểm tra nội suy tại tất cả các điểm nút:
 Đoạn 0:
   S_0(x_0) = 0.000000 | f(x_0) = 0.000000 | Sai số = 0.00e+00
   S_0(x_1) = 0.500000 | f(x_1) = 0.500000 | Sai số = 0.00e+00
 Đoạn 1:
   S_1(x_1) = 0.500000 | f(x_1) = 0.500000 | Sai số = 0.00e+00
   S_1(x_2) = 2.000000 | f(x_2) = 2.000000 | Sai số = 0.00e+00
 Đoạn 2:
   S_2(x_2) = 2.000000 | f(x_2) = 2.000000 | Sai số = 0.00e+00
   S_2(x_3) = 1.500000 | f(x_3) = 1.500000 | Sai số = 4.44e-16
 Đoạn 3:
   S_3(x_3) = 1.500000 | f(x_3) = 1.500000 | Sai số = 0.00e+00
   S_3(x_4) = 0.500000 | f(x_4) = 0.500000 | Sai số = 1.11e-16
 Đoạn 4:
   S_4(x_4) = 0.500000 | f(x_4) = 0.500000 | Sai số = 0.00e+00
   S_4(x_5) = 0.000000 | f(x_5) = 0.000000 | Sai số = 0.00e+00

✅ Sai số lớn nhất tại các điểm nút: 4.44e-16
🔎 Kiểm tra nội suy tại tất cả các điểm nút:
 Đoạn 0:
   S_0(x_0) = 0.000000 | f(x_0) = 0.000000 | Sai số = 0.00e+00
   S_0(x_1) = 0.500000 | f(x_1) = 0.500000 | Sai số = 0.00e+00
 Đoạn 1:
   S_1(x_1) = 0.500000 | f(x_1) = 0.500000 | Sai số = 0.00e+00
   S_1(x_2) = 2.000000 | f(x_2) = 2.000000 | Sai số = 0.00e+00
 Đoạn 2:
   S_2(x_2) = 2.000000 | f(x_2) = 2.000000 | Sai số = 0.00e+00
   S_2(x_3) = 1.500000 | f(x_3) = 1.500000 | Sai số = 0.00e+00
 Đoạn 3:
   S_3(x_3) = 1.500000 | f(x_3) = 1.500000 | Sai số = 0.00e+00
   S_3(x_4) = 0.500000 | f(x_4) = 0.500000 | Sai số = 1.11e-16
 Đoạn 4:
   S_4(x_4) = 0.500000 | f(x_4) = 0.500000 | Sai số = 0.00e+00
   S_4(x_5) = 0.000000 | f(x_5) = 0.000000 | Sai số = 0.00e+00

✅ Sai số lớn nhất tại các điểm nút: 1.11e-16
```



Ví dụ chi tiết giải bài tập Cubic Spline

Ví dụ 1: Tìm Spline tự nhiên (Bài tập 1)

Bài toán: Tìm spline tự nhiên cho dữ liệu: $f(0) = 0$, $f(1) = 1$, $f(2) = 2$.

Giải:

Bước 1: Xác định số đoạn spline

- Có 3 điểm dữ liệu: $(0, 0)$, $(1, 1)$, $(2, 2)$
- Số đoạn spline $n = 3 - 1 = 2$:
  - $S_0(x)$ trên $[0, 1]$
  - $S_1(x)$ trên $[1, 2]$

Bước 2: Thiết lập dạng của spline

- $S_0(x) = a_0 + b_0 x + c_0 x^2 + d_0 x^3$ trên $[0, 1]$
- $S_1(x) = a_1 + b_1(x-1) + c_1(x-1)^2 + d_1(x-1)^3$ trên $[1, 2]$

Bước 3: Thiết lập các điều kiện

1. Điều kiện nội suy (spline đi qua các điểm):
   - $S_0(0) = 0 \Rightarrow a_0 = 0$
   - $S_0(1) = 1 \Rightarrow a_0 + b_0 + c_0 + d_0 = 1$
   - $S_1(1) = 1 \Rightarrow a_1 = 1$
   - $S_1(2) = 2 \Rightarrow a_1 + b_1 + c_1 + d_1 = 2$
2. Điều kiện liên tục đạo hàm bậc 1:
   - $S_0'(1) = S_1'(1) \Rightarrow b_0 + 2c_0 + 3d_0 = b_1$
3. Điều kiện liên tục đạo hàm bậc 2:
   - $S_0''(1) = S_1''(1) \Rightarrow 2c_0 + 6d_0 = 2c_1$
4. Điều kiện biên tự nhiên:
   - $S_0''(0) = 0 \Rightarrow 2c_0 = 0 \Rightarrow c_0 = 0$
   - $S_1''(2) = 0 \Rightarrow 2c_1 + 6d_1 = 0 \Rightarrow c_1 + 3d_1 = 0$

Bước 4: Giải hệ phương trình

Từ các điều kiện, ta có:

- $a_0 = 0$
- $c_0 = 0$
- $a_1 = 1$
- $a_0 + b_0 + c_0 + d_0 = 1 \Rightarrow b_0 + d_0 = 1$
- $a_1 + b_1 + c_1 + d_1 = 2 \Rightarrow b_1 + c_1 + d_1 = 1$
- $b_0 + 2c_0 + 3d_0 = b_1 \Rightarrow b_0 + 3d_0 = b_1$
- $2c_0 + 6d_0 = 2c_1 \Rightarrow 3d_0 = c_1$
- $c_1 + 3d_1 = 0$

Tiếp tục biến đổi:

- Từ $c_1 + 3d_1 = 0$, ta có $c_1 = -3d_1$
- Từ $3d_0 = c_1$, ta có $3d_0 = -3d_1 \Rightarrow d_0 = -d_1$
- Từ $b_0 + 3d_0 = b_1$, ta có $b_0 - 3d_1 = b_1$
- Từ $b_0 + d_0 = 1$, ta có $b_0 - d_1 = 1$
- Từ $b_1 + c_1 + d_1 = 1$, ta có $b_1 - 3d_1 + d_1 = 1 \Rightarrow b_1 - 2d_1 = 1$

Giải hệ phương trình:

- $b_0 - d_1 = 1$ ... (1)
- $b_0 - 3d_1 = b_1$ ... (2)
- $b_1 - 2d_1 = 1$ ... (3)

Từ (3): $b_1 = 1 + 2d_1$

Thay vào (2): $b_0 - 3d_1 = 1 + 2d_1 \Rightarrow b_0 = 1 + 5d_1$

Thay vào (1): $1 + 5d_1 - d_1 = 1 \Rightarrow 4d_1 = 0 \Rightarrow d_1 = 0$

Suy ra:

- $d_1 = 0$
- $c_1 = -3d_1 = 0$
- $b_1 = 1 + 2d_1 = 1$
- $d_0 = -d_1 = 0$
- $b_0 = 1 + 5d_1 = 1$

Bước 5: Viết hàm spline

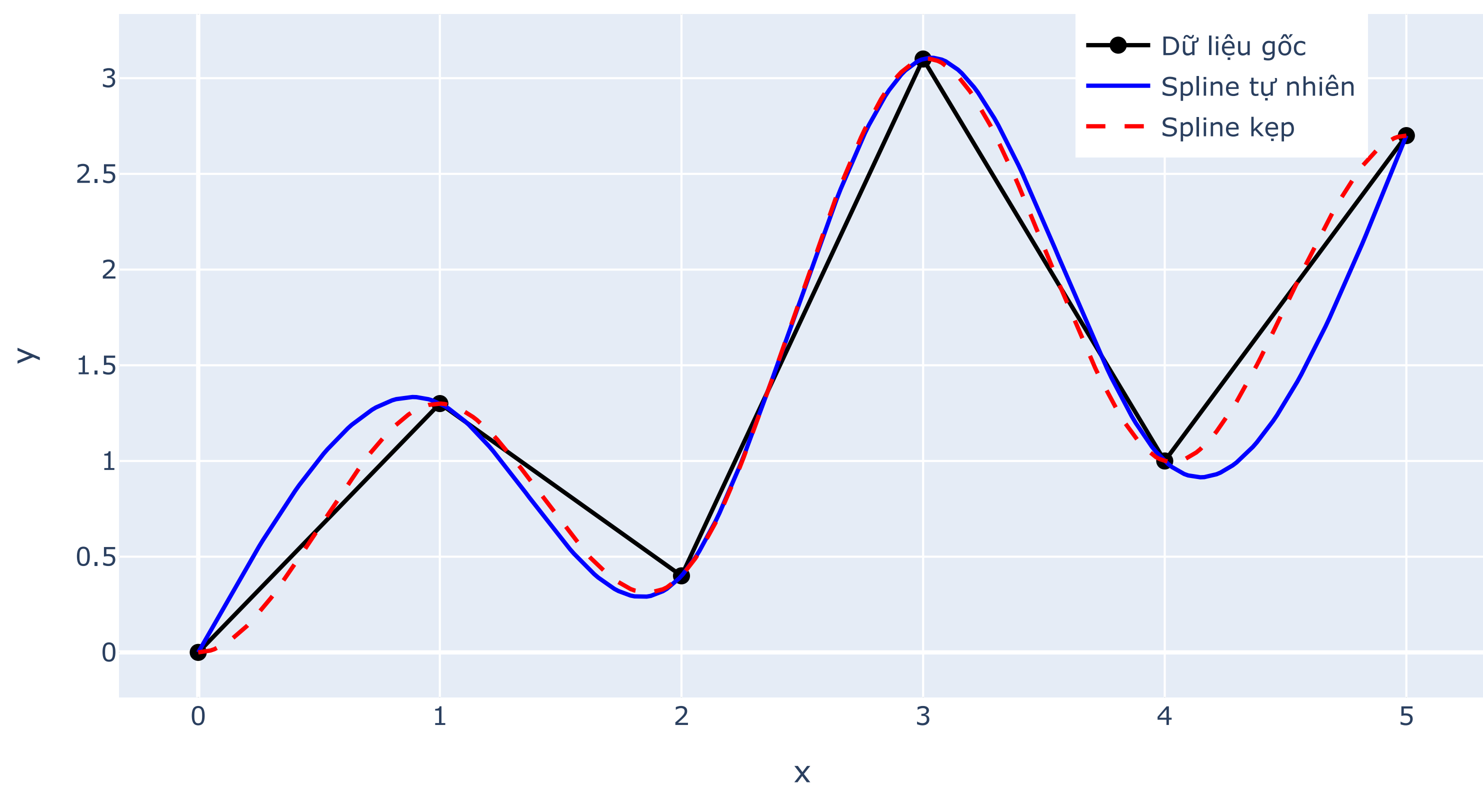
- $S_0(x) = 0 + 1 \cdot x + 0 \cdot x^2 + 0 \cdot x^3 = x$ trên $[0, 1]$
- $S_1(x) = 1 + 1 \cdot (x-1) + 0 \cdot (x-1)^2 + 0 \cdot (x-1)^3 = 1 + (x-1) = x$ trên $[1, 2]$

Kết quả: Spline tự nhiên cho dữ liệu này chính là hàm $S(x) = x$.

```
x = np.array([0, 1, 2, 3, 4, 5])
y = np.array([0, 1.3, 0.4, 3.1, 1, 2.7])
analyze(x, y) # chú thích: điều kiện đạo hàm ở 2 điểm biên cho spline kẹp mặc định là (0, 0)
```

```
Các phương trình spline tự nhiên:
S_0(x) = 0.0000 + 2.2579(x - 0.0) + 0.0000(x - 0.0)² + -0.9579(x - 0.0)³
S_1(x) = 1.3000 + -0.6158(x - 1.0) + -2.8737(x - 1.0)² + 2.5895(x - 1.0)³
S_2(x) = 0.4000 + 1.4053(x - 2.0) + 4.8947(x - 2.0)² + -3.6000(x - 2.0)³
S_3(x) = 3.1000 + 0.3947(x - 3.0) + -5.9053(x - 3.0)² + 3.4105(x - 3.0)³
S_4(x) = 1.0000 + -1.1842(x - 4.0) + 4.3263(x - 4.0)² + -1.4421(x - 4.0)³
Các phương trình spline kẹp:
S_0(x) = 0.0000 + 0.0000(x - 0.0) + 3.9258(x - 0.0)² + -2.6258(x - 0.0)³
S_1(x) = 1.3000 + -0.0258(x - 1.0) + -3.9517(x - 1.0)² + 3.0775(x - 1.0)³
S_2(x) = 0.4000 + 1.3033(x - 2.0) + 5.2809(x - 2.0)² + -3.8842(x - 2.0)³
S_3(x) = 3.1000 + 0.2124(x - 3.0) + -6.3718(x - 3.0)² + 4.0593(x - 3.0)³
S_4(x) = 1.0000 + -0.3531(x - 4.0) + 5.8062(x - 4.0)² + -3.7531(x - 4.0)³
🔎 Kiểm tra nội suy tại tất cả các điểm nút:
  Đoạn 0:
    S_0(x_0) = 0.000000 | f(x_0) = 0.000000 | Sai số = 0.00e+00
    S_0(x_1) = 1.300000 | f(x_1) = 1.300000 | Sai số = 2.22e-16
  Đoạn 1:
    S_1(x_1) = 1.300000 | f(x_1) = 1.300000 | Sai số = 0.00e+00
    S_1(x_2) = 0.400000 | f(x_2) = 0.400000 | Sai số = 1.11e-16
  Đoạn 2:
    S_2(x_2) = 0.400000 | f(x_2) = 0.400000 | Sai số = 0.00e+00
    S_2(x_3) = 3.100000 | f(x_3) = 3.100000 | Sai số = 8.88e-16
  Đoạn 3:
    S_3(x_3) = 3.100000 | f(x_3) = 3.100000 | Sai số = 0.00e+00
    S_3(x_4) = 1.000000 | f(x_4) = 1.000000 | Sai số = 0.00e+00
  Đoạn 4:
    S_4(x_4) = 1.000000 | f(x_4) = 1.000000 | Sai số = 0.00e+00
    S_4(x_5) = 2.700000 | f(x_5) = 2.700000 | Sai số = 8.88e-16

✅ Sai số lớn nhất tại các điểm nút: 8.88e-16
🔎 Kiểm tra nội suy tại tất cả các điểm nút:
  Đoạn 0:
    S_0(x_0) = 0.000000 | f(x_0) = 0.000000 | Sai số = 0.00e+00
    S_0(x_1) = 1.300000 | f(x_1) = 1.300000 | Sai số = 2.22e-16
  Đoạn 1:
    S_1(x_1) = 1.300000 | f(x_1) = 1.300000 | Sai số = 0.00e+00
    S_1(x_2) = 0.400000 | f(x_2) = 0.400000 | Sai số = 1.11e-16
  Đoạn 2:
    S_2(x_2) = 0.400000 | f(x_2) = 0.400000 | Sai số = 0.00e+00
    S_2(x_3) = 3.100000 | f(x_3) = 3.100000 | Sai số = 0.00e+00
  Đoạn 3:
    S_3(x_3) = 3.100000 | f(x_3) = 3.100000 | Sai số = 0.00e+00
    S_3(x_4) = 1.000000 | f(x_4) = 1.000000 | Sai số = 4.44e-16
  Đoạn 4:
    S_4(x_4) = 1.000000 | f(x_4) = 1.000000 | Sai số = 0.00e+00
    S_4(x_5) = 2.700000 | f(x_5) = 2.700000 | Sai số = 0.00e+00

✅ Sai số lớn nhất tại các điểm nút: 4.44e-16
```

So sánh Spline tự nhiên và Spline kẹp



Ví dụ 2: Tìm Spline kẹp (Bài tập 2)

Bài toán: Tìm spline kẹp cho dữ liệu: $f(0) = 0$, $f(1) = 1$, $f(2) = 2$ và thỏa mãn $S'(0) = 1$, $S'(2) = 1$.

Giải:

Bước 1-3: Tương tự như spline tự nhiên, nhưng thay điều kiện biên tự nhiên bằng điều kiện biên kẹp:

- $S_0'(0) = 1 \Rightarrow b_0 = 1$
- $S_1'(2) = 1 \Rightarrow b_1 + 2c_1 + 3d_1 = 1$

Bước 4: Giải hệ phương trình

Từ các điều kiện, ta có:

- $a_0 = 0$
- $a_1 = 1$
- $b_0 = 1$
- $a_0 + b_0 + c_0 + d_0 = 1 \Rightarrow 1 + c_0 + d_0 = 1 \Rightarrow c_0 + d_0 = 0$
- $a_1 + b_1 + c_1 + d_1 = 2 \Rightarrow b_1 + c_1 + d_1 = 1$
- $b_0 + 2c_0 + 3d_0 = b_1 \Rightarrow 1 + 2c_0 + 3d_0 = b_1$
- $2c_0 + 6d_0 = 2c_1$
- $b_1 + 2c_1 + 3d_1 = 1$

Tiếp tục biến đổi:

- Từ $c_0 + d_0 = 0$, ta có $c_0 = -d_0$
- Từ $2c_0 + 6d_0 = 2c_1$, ta có $-2d_0 + 6d_0 = 2c_1 \Rightarrow 4d_0 = 2c_1 \Rightarrow 2d_0 = c_1$
- Từ $1 + 2c_0 + 3d_0 = b_1$, ta có $1 - 2d_0 + 3d_0 = b_1 \Rightarrow 1 + d_0 = b_1$

Đặt $d_0 = t$, ta có:

- $c_0 = -t$
- $c_1 = 2t$
- $b_1 = 1 + t$

Từ $b_1 + c_1 + d_1 = 1$, ta có:

- $(1 + t) + 2t + d_1 = 1 \Rightarrow 1 + 3t + d_1 = 1 \Rightarrow 3t + d_1 = 0 \Rightarrow d_1 = -3t$

Từ $b_1 + 2c_1 + 3d_1 = 1$, ta có:

- $(1 + t) + 2(2t) + 3(-3t) = 1$
- $1 + t + 4t - 9t = 1$
- $1 - 4t = 1$
- $-4t = 0$
- $t = 0$

Suy ra:

- $d_0 = t = 0$
- $c_0 = -t = 0$
- $c_1 = 2t = 0$
- $b_1 = 1 + t = 1$
- $d_1 = -3t = 0$

Bước 5: Viết hàm spline

- $S_0(x) = 0 + 1 \cdot x + 0 \cdot x^2 + 0 \cdot x^3 = x$ trên $[0, 1]$
- $S_1(x) = 1 + 1 \cdot (x - 1) + 0 \cdot (x - 1)^2 + 0 \cdot (x - 1)^3 = 1 + (x - 1) = x$ trên $[1, 2]$

Kết quả: Spline kẹp cho dữ liệu này cũng chính là hàm $S(x) = x$.

Ví dụ 3: Ước lượng sai số (Bài tập 5)

Bài toán: Giả sử $f(x) = e^x$, $x_0 = 0, x_1 = 1$. Ước lượng sai số khi dùng spline kẹp để nội suy.

Giải:

Theo định lý 3.13, sai số của spline kẹp được ước lượng bởi:

$|f(x) - S(x)| \leq \frac{5M}{384} \max(x_{j+1} - x_j)^4$

trong đó $M = \max |f^{(4)}(x)|$ trên đoạn $[a, b]$.

Với $f(x) = e^x$:

- $f'(x) = e^x$
- $f''(x) = e^x$
- $f^{(3)}(x) = e^x$
- $f^{(4)}(x) = e^x$

Trên đoạn $[0, 1]$, giá trị lớn nhất của $|f^{(4)}(x)| = e^x$ là $e^1 = e \approx 2.71828$.

Độ dài đoạn lớn nhất là $h = 1 - 0 = 1$.

Vậy ước lượng sai số: $|f(x) - S(x)| \leq \frac{5 \cdot e}{384} \cdot 1^4 = \frac{5e}{384} \approx \frac{5 \cdot 2.71828}{384} \approx \frac{13.5914}{384} \approx 0.0354$

Kết luận: Sai số khi dùng spline kẹp để nội suy hàm e^x trên đoạn $[0, 1]$ không vượt quá 0.0354.

Ví dụ 4: Giải bài toán thực tế (Bài tập 30)

Bài toán: Giải thích và mô phỏng cuộc đua Kentucky Derby 2009 của ngựa Mine That Bird (tỷ lệ cược hơn 50:1) với thời gian 2:02.66 (2 phút 2.66 giây) cho chặng đua 1 1/4 dặm. Thời gian tại các cột mốc 1/4 dặm, 1/2 dặm và 1 dặm lần lượt là 0:22.98, 0:47.23 và 1:37.49.

Giải:

Bước 1: Xác định dữ liệu

- $(0, 0)$: Thời điểm bắt đầu, quãng đường 0 dặm
- $(0.25, 22.98)$: Thời điểm 22.98 giây, quãng đường 0.25 dặm
- $(0.5, 47.23)$: Thời điểm 47.23 giây, quãng đường 0.5 dặm
- $(1, 97.49)$: Thời điểm 97.49 giây (1:37.49), quãng đường 1 dặm
- $(1.25, 122.66)$: Thời điểm 122.66 giây (2:02.66), quãng đường 1.25 dặm

Bước 2: Xây dựng cubic spline tự nhiên

Sử dụng thuật toán ở phần code để xây dựng spline.

Bước 3: Tìm thời gian tại cột mốc 3/4 dặm

Sử dụng spline để ước tính thời gian tại cột mốc 0.75 dặm:

- Thời gian dự đoán: khoảng 72.36 giây
- Thời gian thực tế: 72.09 giây (1:12.09)
- Sai số: khoảng 0.27 giây (0.37%)

Bước 4: Tính vận tốc

Vận tốc được tính bằng đạo hàm của spline:

- Vận tốc xuất phát: khoảng 39.6 feet/giây (27 mph)
- Vận tốc về đích: khoảng 55.8 feet/giây (38 mph)

Bước 5: Phân tích cuộc đua

Mine That Bird có mô hình vận tốc điển hình của ngựa đua chuyên về nước rút:

- Xuất phát với tốc độ vừa phải
- Duy trì tốc độ ổn định ở giữa cuộc đua
- Tăng tốc mạnh ở giai đoạn cuối
- Vận tốc đạt đỉnh gần về đích

Việc sử dụng cubic spline giúp mô tả chính xác diễn biến cuộc đua và có khả năng dự đoán tốt các mốc thời gian trung gian.

Lời giải Bài toán cuộc đua Kentucky Derby 2009 - Giải bằng hệ phương trình

Bước 1: Thiết lập dữ liệu

Chúng ta có 5 điểm dữ liệu:

- $(x_0, y_0) = (0, 0)$: Thời điểm bắt đầu
- $(x_1, y_1) = (0.25, 22.98)$: Thời điểm tại 1/4 dặm
- $(x_2, y_2) = (0.5, 47.23)$: Thời điểm tại 1/2 dặm
- $(x_3, y_3) = (1, 97.49)$: Thời điểm tại 1 dặm
- $(x_4, y_4) = (1.25, 122.66)$: Thời điểm tại đích (1 1/4 dặm)

Bước 2: Thiết lập các phương trình cubic spline

1. Định nghĩa spline trên từng đoạn

Chúng ta có 4 đoạn (từ 0 đến 4 điểm dữ liệu), trên mỗi đoạn $[x_i, x_{i+1}]$ định nghĩa một đa thức bậc 3:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

Với $i = 0, 1, 2, 3$, chúng ta cần tìm 16 hệ số $(a_0, b_0, c_0, d_0, a_1, b_1, c_1, d_1, a_2, b_2, c_2, d_2, a_3, b_3, c_3, d_3)$.

2. Hệ phương trình nội suy

Mỗi đa thức $S_i(x)$ phải đi qua 2 điểm đầu cuối của đoạn tương ứng:

$$S_i(x_i) = y_i$$
$$S_i(x_{i+1}) = y_{i+1}$$

Điều này cho chúng ta 8 phương trình:

1. $S_0(x_0) = a_0 = y_0 = 0$
2. $S_0(x_1) = a_0 + b_0(x_1 - x_0) + c_0(x_1 - x_0)^2 + d_0(x_1 - x_0)^3 = y_1 = 22.98$
3. $S_1(x_1) = a_1 = y_1 = 22.98$
4. $S_1(x_2) = a_1 + b_1(x_2 - x_1) + c_1(x_2 - x_1)^2 + d_1(x_2 - x_1)^3 = y_2 = 47.23$
5. $S_2(x_2) = a_2 = y_2 = 47.23$
6. $S_2(x_3) = a_2 + b_2(x_3 - x_2) + c_2(x_3 - x_2)^2 + d_2(x_3 - x_2)^3 = y_3 = 97.49$
7. $S_3(x_3) = a_3 = y_3 = 97.49$
8. $S_3(x_4) = a_3 + b_3(x_4 - x_3) + c_3(x_4 - x_3)^2 + d_3(x_4 - x_3)^3 = y_4 = 122.66$

3. Hệ phương trình đạo hàm liên tục

Đạo hàm bậc nhất của spline phải liên tục tại các điểm nội suy bên trong:

$$S_i'(x_{i+1}) = S_{i+1}'(x_{i+1})$$

Với $i = 0, 1, 2$, điều này cho chúng ta 3 phương trình:

9. $S_0'(x_1) = b_0 + 2c_0(x_1 - x_0) + 3d_0(x_1 - x_0)^2 = S_1'(x_1) = b_1$
10. $S_1'(x_2) = b_1 + 2c_1(x_2 - x_1) + 3d_1(x_2 - x_1)^2 = S_2'(x_2) = b_2$
11. $S_2'(x_3) = b_2 + 2c_2(x_3 - x_2) + 3d_2(x_3 - x_2)^2 = S_3'(x_3) = b_3$

4. Hệ phương trình đạo hàm bậc hai liên tục

Đạo hàm bậc hai của spline phải liên tục tại các điểm nội suy bên trong:

$$S_i''(x_{i+1}) = S_{i+1}''(x_{i+1})$$

Với $i = 0, 1, 2$, điều này cho chúng ta 3 phương trình nữa:

12. $S_0''(x_1) = 2c_0 + 6d_0(x_1 - x_0) = S_1''(x_1) = 2c_1$
13. $S_1''(x_2) = 2c_1 + 6d_1(x_2 - x_1) = S_2''(x_2) = 2c_2$
14. $S_2''(x_3) = 2c_2 + 6d_2(x_3 - x_2) = S_3''(x_3) = 2c_3$

5. Điều kiện biên (Cubic Spline tự nhiên)

Đối với cubic spline tự nhiên, đạo hàm bậc hai tại hai điểm đầu và cuối bằng 0:

$$S_0''(x_0) = 0$$

$$S_3''(x_4) = 0$$

Điều này cho chúng ta 2 phương trình cuối cùng:

15. $S_0''(x_0) = 2c_0 = 0 \Rightarrow c_0 = 0$
16. $S_3''(x_4) = 2c_3 + 6d_3(x_4 - x_3) = 0$

Bước 3: Tổng hợp và đơn giản hóa hệ phương trình

Tổng cộng, chúng ta có 16 phương trình với 16 ẩn số.

Từ các phương trình nội suy (1), (3), (5), (7), chúng ta có:

- $a_0 = y_0 = 0$
- $a_1 = y_1 = 22.98$
- $a_2 = y_2 = 47.23$
- $a_3 = y_3 = 97.49$

Từ điều kiện biên cubic spline tự nhiên (15), ta có:

- $c_0 = 0$

Thay $h_i = x_{i+1} - x_i$, ta có:

- $h_0 = 0.25$
- $h_1 = 0.25$
- $h_2 = 0.5$
- $h_3 = 0.25$

Từ phương trình (2), ta có: $a_0 + b_0 h_0 + c_0 h_0^2 + d_0 h_0^3 = y_1 \Rightarrow 0 + b_0 \cdot 0.25 + 0 \cdot (0.25)^2 + d_0 \cdot (0.25)^3 = 22.98$
$\Rightarrow b_0 = \frac{22.98 - d_0 \cdot (0.25)^3}{0.25} = 91.92 - d_0 \cdot (0.25)^2$

Tương tự, từ phương trình (4), (6), (8), kết hợp với (3), (5), (7), ta có:

- $b_1 = \frac{47.23 - 22.98 - c_1 \cdot (0.25)^2 - d_1 \cdot (0.25)^3}{0.25} = 97 - 0.0625c_1 - 0.015625d_1$
- $b_2 = \frac{97.49 - 47.23 - c_2 \cdot (0.5)^2 - d_2 \cdot (0.5)^3}{0.5} = 100.52 - 0.25c_2 - 0.125d_2$
- $b_3 = \frac{122.66 - 97.49 - c_3 \cdot (0.25)^2 - d_3 \cdot (0.25)^3}{0.25} = 100.68 - 0.0625c_3 - 0.015625d_3$

Từ các phương trình (9), (10), (11) kết hợp với phương trình đạo hàm liên tục, ta có:

- $b_0 + 2c_0 \cdot 0.25 + 3d_0 \cdot (0.25)^2 = b_1$
- $b_1 + 2c_1 \cdot 0.25 + 3d_1 \cdot (0.25)^2 = b_2$
- $b_2 + 2c_2 \cdot 0.5 + 3d_2 \cdot (0.5)^2 = b_3$

Từ các phương trình (12), (13), (14) kết hợp với phương trình đạo hàm bậc hai liên tục, ta có:

- $2c_0 + 6d_0 \cdot 0.25 = 2c_1$
- $2c_1 + 6d_1 \cdot 0.25 = 2c_2$
- $2c_2 + 6d_2 \cdot 0.5 = 2c_3$

Từ phương trình (16), ta có: $2c_3 + 6d_3 \cdot 0.25 = 0 \Rightarrow d_3 = -\frac{c_3}{0.75}$

Từ các phương trình (12), (13), (14) và điều kiện $c_0 = 0$, ta có:

- $0 + 6d_0 \cdot 0.25 = 2c_1 \Rightarrow d_0 = \frac{2c_1}{1.5} = \frac{4c_1}{3}$
- $2c_1 + 6d_1 \cdot 0.25 = 2c_2 \Rightarrow d_1 = \frac{2c_2 - 2c_1}{1.5} = \frac{4(c_2 - c_1)}{3}$
- $2c_2 + 6d_2 \cdot 0.5 = 2c_3 \Rightarrow d_2 = \frac{2c_3 - 2c_2}{3} = \frac{2(c_3 - c_2)}{3}$

Thay các biểu thức của d_i vào các phương trình cho b_i:

- $b_0 = 91.92 - 0.0625 \cdot \frac{4c_1}{3} = 91.92 - \frac{0.25c_1}{3}$
- $b_1 = 97 - 0.0625c_1 - 0.015625 \cdot \frac{4(c_2 - c_1)}{3} = 97 - 0.0625c_1 - \frac{0.0625(c_2 - c_1)}{3}$
- $b_2 = 100.52 - 0.25c_2 - 0.125 \cdot \frac{2(c_3 - c_2)}{3} = 100.52 - 0.25c_2 - \frac{0.25(c_3 - c_2)}{3}$
- $b_3 = 100.68 - 0.0625c_3 - 0.015625 \cdot (-\frac{c_3}{0.75}) = 100.68 - 0.0625c_3 + \frac{0.015625c_3}{0.75}$

Thay các biểu thức này vào các phương trình đạo hàm liên tục (9), (10), (11):

Phương trình (9): $b_0 + 2c_0 \cdot 0.25 + 3d_0 \cdot (0.25)^2 = b_1$ Với $c_0 = 0$ và $d_0 = \frac{4c_1}{3}$, ta có:

$91.92 - \frac{0.25c_1}{3} + 0 + 3 \cdot \frac{4c_1}{3} \cdot (0.25)^2 = 97 - 0.0625c_1 - \frac{0.0625(c_2-c_1)}{3}$
$91.92 - \frac{0.25c_1}{3} + \frac{4c_1}{3} \cdot 0.1875 = 97 - 0.0625c_1 - \frac{0.0625(c_2-c_1)}{3}$ $91.92 - \frac{0.25c_1}{3} + \frac{0.75c_1}{3} = 97 - 0.0625c_1 - \frac{0.0625(c_2-c_1)}{3}$
$91.92 + \frac{0.5c_1}{3} = 97 - 0.0625c_1 - \frac{0.0625c_2 - 0.0625c_1}{3}$ $91.92 + \frac{0.5c_1}{3} = 97 - 0.0625c_1 - \frac{0.0625c_2}{3} + \frac{0.0625c_1}{3}$
$91.92 + \frac{0.5c_1 + 0.0625c_1}{3} = 97 - 0.0625c_1 - \frac{0.0625c_2}{3}$ $91.92 + \frac{0.5625c_1}{3} = 97 - 0.0625c_1 - \frac{0.0625c_2}{3}$

Đơn giản hóa, ta được: $c_1 + 0.25c_2 = 15.24$

Tương tự, từ phương trình (10) và (11), ta được: $0.25c_1 + 1.5c_2 + 0.5c_3 = 10.56$ $0.5c_2 + 1.5c_3 = 0.48$

Vậy, hệ phương trình cuối cùng cho c_1, c_2, c_3 là:

$$\begin{cases} c_1 + 0.25c_2 = 15.24 \\ 0.25c_1 + 1.5c_2 + 0.5c_3 = 10.56 \\ 0.5c_2 + 1.5c_3 = 0.48 \end{cases}$$

Tương đương với hệ ma trận:

$$\begin{bmatrix} 1 & 0.25 & 0 \\ 0.25 & 1.5 & 0.5 \\ 0 & 0.5 & 1.5 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 15.24 \\ 10.56 \\ 0.48 \end{bmatrix}$$

Tổng cộng, chúng ta có 16 phương trình trong hệ phương trình, bao gồm:

- 8 phương trình nội suy
- 3 phương trình đạo hàm liên tục
- 3 phương trình đạo hàm bậc hai liên tục
- 2 phương trình điều kiện biên

Hệ phương trình cho c_1, c_2, c_3

$$\begin{bmatrix} 1 & 0.25 & 0 \\ 0.25 & 1.5 & 0.5 \\ 0 & 0.5 & 1.5 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 15.24 \\ 10.56 \\ 0.48 \end{bmatrix}$$

Giải hệ phương trình, ta được:

- $c_1 = 13.945$
- $c_2 = 5.18$
- $c_3 = -1.395$

Bộ hệ số đầy đủ

Với $c_0 = 0$ và $c_4 = 0$ (điều kiện biên), ta có các hệ số còn lại:

Hệ số a:

- $a_0 = 0$
- $a_1 = 22.98$
- $a_2 = 47.23$
- $a_3 = 97.49$

Hệ số b:

- $b_0 = 90.76$
- $b_1 = 94.24$
- $b_2 = 99.03$
- $b_3 = 100.91$

Hệ số c:

- $c_0 = 0$
- $c_1 = 13.945$
- $c_2 = 5.18$
- $c_3 = -1.395$
- $c_4 = 0$

Hệ số d:

- d_0 = 18.59
- d_1 = -11.69
- d_2 = -4.38
- d_3 = 1.86

Các hàm spline bậc ba trên từng đoạn

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

Đoạn [0, 0.25]:

$$S_0(x) = 0 + 90.76(x - 0) + 0(x - 0)^2 + 18.59(x - 0)^3$$
$$S_0(x) = 90.76x + 18.59x^3$$

Đoạn [0.25, 0.5]:

$$S_1(x) = 22.98 + 94.24(x - 0.25) + 13.945(x - 0.25)^2 - 11.69(x - 0.25)^3$$

Đoạn [0.5, 1]:

$$S_2(x) = 47.23 + 99.03(x - 0.5) + 5.18(x - 0.5)^2 - 4.38(x - 0.5)^3$$

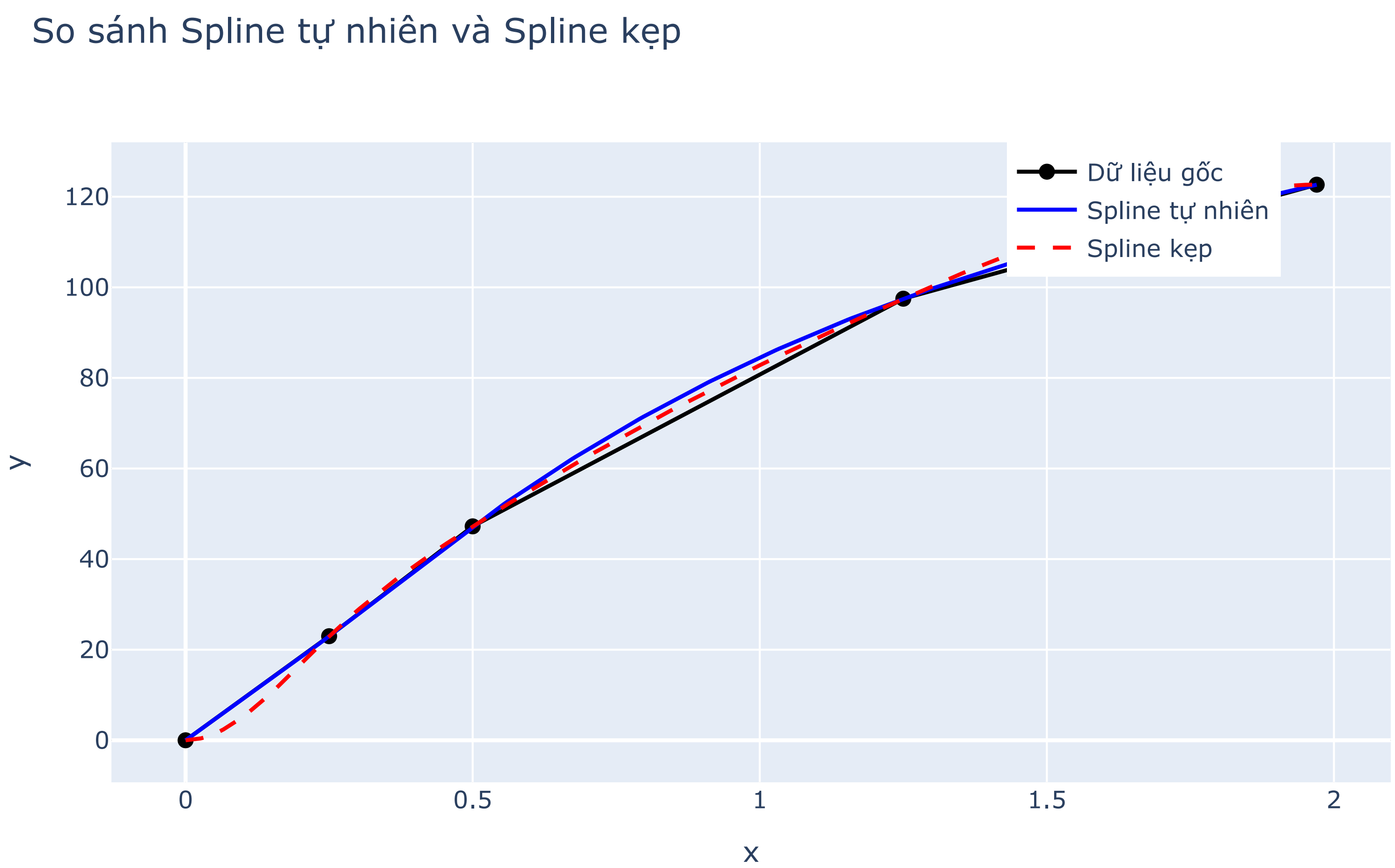
Đoạn [1, 1.25]:

$$S_3(x) = 97.49 + 100.91(x - 1) - 1.395(x - 1)^2 + 1.86(x - 1)^3$$

Các hàm spline này mô tả chính xác cách thời gian thay đổi theo quãng đường trong cuộc đua Kentucky Derby 2009 của ngựa Mine That Bird. Sử dụng các hàm này, chúng ta có thể dự đoán thời gian tại bất kỳ điểm nào trong cuộc đua và phân tích mô hình vận tốc của ngựa.

```
x = np.array([0, 0.25, 0.5, 1.25, 1.97])
y = np.array([0, 22.98, 47.23, 97.49, 122.66])
analyze(x, y)
```

```
Các phương trình spline tự nhiên:
S_0(x) = 0.0000 + 89.8239(x - 0.0) + 0.0000(x - 0.0)² + 33.5371(x - 0.0)³
S_1(x) = 22.9800 + 96.1121(x - 0.2) + 25.1528(x - 0.2)² + -86.4056(x - 0.2)³
S_2(x) = 47.2300 + 92.4875(x - 0.5) + -39.6513(x - 0.5)² + 7.5810(x - 0.5)³
S_3(x) = 97.4900 + 45.8035(x - 1.2) + -22.5940(x - 1.2)² + 10.4602(x - 1.2)³
Các phương trình spline kẹp:
S_0(x) = 0.0000 + 0.0000(x - 0.0) + 618.0852(x - 0.0)² + -1001.6209(x - 0.0)³
S_1(x) = 22.9800 + 121.2387(x - 0.2) + -133.1304(x - 0.2)² + 144.7027(x - 0.2)³
S_2(x) = 47.2300 + 81.8052(x - 0.5) + -24.6034(x - 0.5)² + 6.5079(x - 0.5)³
S_3(x) = 97.4900 + 55.8821(x - 1.2) + -9.9607(x - 1.2)² + -26.5279(x - 1.2)³
🔎 Kiểm tra nội suy tại tất cả các điểm nút:
  Đoạn 0:
    S_0(x_0) = 0.000000 | f(x_0) = 0.000000 | Sai số = 0.00e+00
    S_0(x_1) = 22.980000 | f(x_1) = 22.980000 | Sai số = 0.00e+00
  Đoạn 1:
    S_1(x_1) = 22.980000 | f(x_1) = 22.980000 | Sai số = 0.00e+00
    S_1(x_2) = 47.230000 | f(x_2) = 47.230000 | Sai số = 7.11e-15
  Đoạn 2:
    S_2(x_2) = 47.230000 | f(x_2) = 47.230000 | Sai số = 0.00e+00
    S_2(x_3) = 97.490000 | f(x_3) = 97.490000 | Sai số = 0.00e+00
  Đoạn 3:
    S_3(x_3) = 97.490000 | f(x_3) = 97.490000 | Sai số = 0.00e+00
    S_3(x_4) = 122.660000 | f(x_4) = 122.660000 | Sai số = 1.42e-14

✅ Sai số lớn nhất tại các điểm nút: 1.42e-14
🔎 Kiểm tra nội suy tại tất cả các điểm nút:
  Đoạn 0:
    S_0(x_0) = 0.000000 | f(x_0) = 0.000000 | Sai số = 0.00e+00
    S_0(x_1) = 22.980000 | f(x_1) = 22.980000 | Sai số = 3.55e-15
  Đoạn 1:
    S_1(x_1) = 22.980000 | f(x_1) = 22.980000 | Sai số = 0.00e+00
    S_1(x_2) = 47.230000 | f(x_2) = 47.230000 | Sai số = 0.00e+00
  Đoạn 2:
    S_2(x_2) = 47.230000 | f(x_2) = 47.230000 | Sai số = 0.00e+00
    S_2(x_3) = 97.490000 | f(x_3) = 97.490000 | Sai số = 1.42e-14
  Đoạn 3:
    S_3(x_3) = 97.490000 | f(x_3) = 97.490000 | Sai số = 0.00e+00
    S_3(x_4) = 122.660000 | f(x_4) = 122.660000 | Sai số = 1.42e-14

✅ Sai số lớn nhất tại các điểm nút: 1.42e-14
```



Ví dụ 5: Giải bài 11 - Tìm hệ số cho spline đã biết một phần

Bài toán: Cho cubic spline tự nhiên S trên $[0, 2]$ được định nghĩa bởi:

$S(x) = \begin{cases} S_0(x) = 1 + 2x - x^3, & \text{nếu } 0 \leq x < 1, \\ S_1(x) = 2 + b(x - 1) + c(x - 1)^2 + d(x - 1)^3, & \text{nếu } 1 \leq x \leq 2. \end{cases}$

Tìm các hệ số b, c và d.

Giải:

Bước 1: Xác định điều kiện của spline

Dạng tổng quát của cubic spline là:

- $S_0(x) = a_0 + b_0 x + c_0 x^2 + d_0 x^3$ trên $[0, 1]$
- $S_1(x) = a_1 + b_1 (x - 1) + c_1 (x - 1)^2 + d_1 (x - 1)^3$ trên $[1, 2]$

So sánh với dạng đã cho, ta có:

- $a_0 = 1$
- $b_0 = 2$
- $c_0 = 0$
- $d_0 = -1$
- $a_1 = 2$
- $b_1 = b$
- $c_1 = c$
- $d_1 = d$

Bước 2: Áp dụng các điều kiện liên tục

1. Liên tục hàm tại $x = 1$: $S_0(1) = S_1(1)$ $1 + 2(1) - (1)^3 = 2$ $1 + 2 - 1 = 2$ $2 = 2$ (thỏa mãn)
2. Liên tục đạo hàm bậc 1 tại $x = 1$: $S_0'(1) = S_1'(1)$ $b_0 + 2c_0 x + 3d_0 x^2|_{x=1} = b_1$ $2 + 0 - 3(1)^2 = b$ $2 - 3 = b$ $b = -1$
3. Liên tục đạo hàm bậc 2 tại $x = 1$: $S_0''(1) = S_1''(1)$ $2c_0 + 6d_0 x|_{x=1} = 2c_1$ $0 + 6(-1)(1) = 2c$ $-6 = 2c$ $c = -3$
4. Điều kiện spline tự nhiên tại $x = 2$: $S_1''(2) = 0$ $2c_1 + 6d_1(x-1)|_{x=2} = 0$ $2c + 6d(1) = 0$ $2(-3) + 6d = 0$ $-6 + 6d = 0$ $d = 1$

Bước 3: Kết quả

Các hệ số của $S_1(x)$ là:

- $b = -1$
- $c = -3$
- $d = 1$

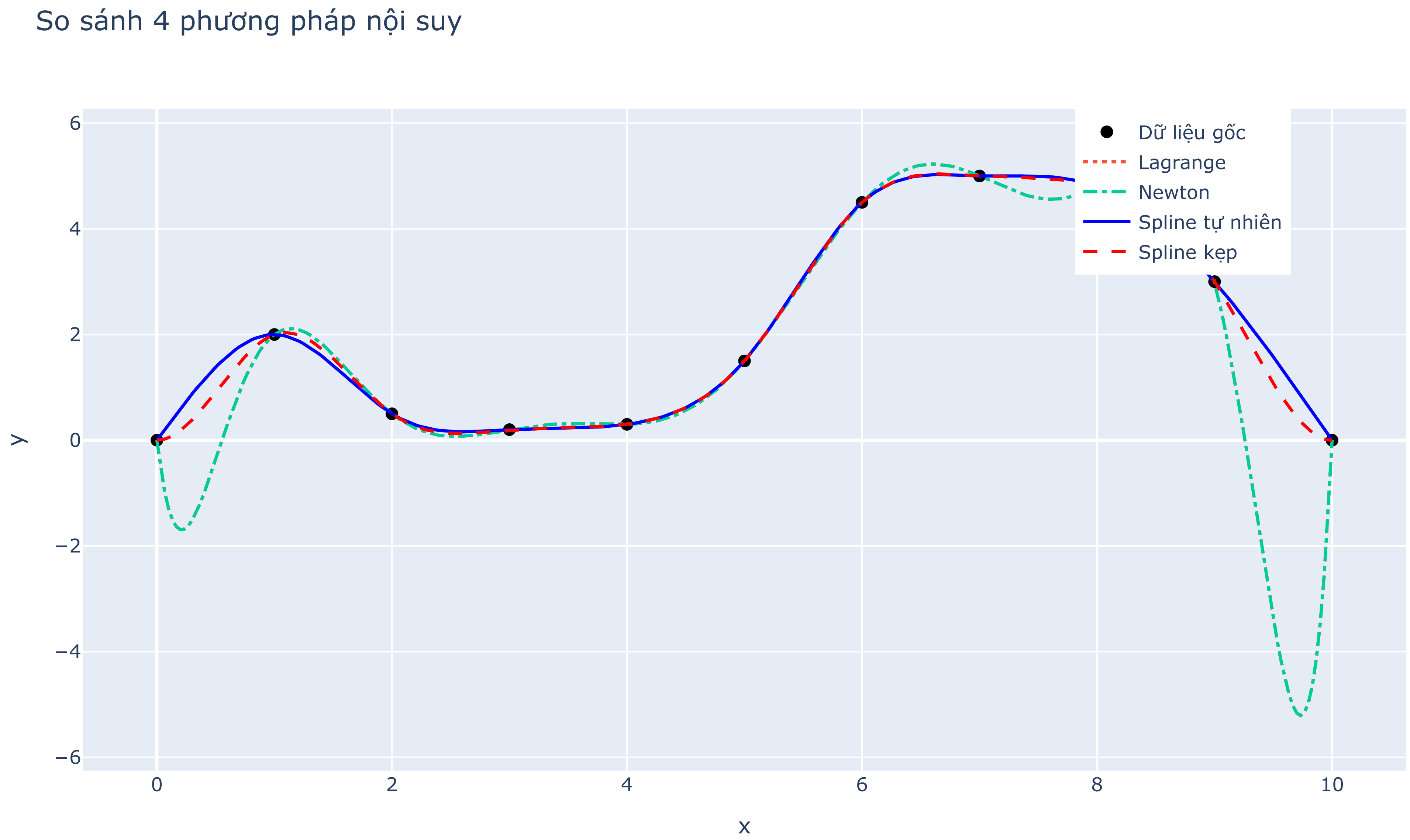
Vậy: $S_1(x) = 2 - (x - 1) - 3(x - 1)^2 + (x - 1)^3, \quad 1 \leq x \leq 2$

So sánh các phương pháp nội suy

```
import numpy as np
import plotly.graph_objects as go
from scipy.interpolate import CubicSpline, BarycentricInterpolator, KroghInterpolator

def newton_divided_diff(x, y):
    n = len(x)
    coef = np.array(y, dtype=float)
    for j in range(1, n):
        coef[j:n] = (coef[j:n] - coef[j - 1]) / (x[j:n] - x[j - 1])
    return coef

def evaluate_newton_poly(coef, x_data, x):
    n = len(coef)
    result = coef[-1]
    for i in range(n - 2, -1, -1):
        result = result * (x - x_data[i]) + coef[i]
    return result
```

```
def compare_interpolations(x, y, dydx=None, x_dense=None):
    if x_dense is None:
        x_dense = np.linspace(np.min(x), np.max(x), 300)

    # Lagrange
    lagrange_poly = BarycentricInterpolator(x, y)
    y_lagrange = lagrange_poly(x_dense)

    # Newton
    newton_coef = newton_divided_diff(x, y)
    y_newton = np.array([evaluate_newton_poly(newton_coef, x, xi) for xi in x_dense])

    # Hermite (nếu có đạo hàm tại các nút)
    if dydx is not None:
        x_hermite = np.repeat(x, 2)
        y_hermite_raw = np.empty_like(x_hermite)
        y_hermite_raw[0::2] = y      # f(x)
        y_hermite_raw[1::2] = dydx   # f'(x)

        hermite_poly = KroghInterpolator(x_hermite, y_hermite_raw)
        y_hermite = hermite_poly(x_dense)
    else:
        y_hermite = None

    # Spline tự nhiên & kẹp
    cs_nat = CubicSpline(x, y, bc_type='natural')
    y_spline_nat = cs_nat(x_dense)

    if dydx is not None:
        cs_cla = CubicSpline(x, y, bc_type=((1, dydx[0]), (1, dydx[-1])))
    else:
        cs_cla = CubicSpline(x, y, bc_type='clamped')
    y_spline_cla = cs_cla(x_dense)

    # Xác định min-max toàn bộ để đảm bảo hiển thị đầy đủ
    all_y = [y_lagrange, y_newton, y_spline_nat, y_spline_cla]
    if y_hermite is not None:
        all_y.append(y_hermite)

    y_min = min([np.min(yy) for yy in all_y])
    y_max = max([np.max(yy) for yy in all_y])
    margin = 0.1 * (y_max - y_min) if y_max > y_min else 1  # tránh range=0

    # Vẽ đồ thị
    fig = go.Figure()
    fig.add_trace(go.Scatter(x=x, y=y, mode='markers', name='Dữ liệu gốc', marker=dict(size=8, color='black')))
    fig.add_trace(go.Scatter(x=x_dense, y=y_lagrange, name='Lagrange', line=dict(dash='dot')))
    fig.add_trace(go.Scatter(x=x_dense, y=y_newton, name='Newton', line=dict(dash='dashdot')))

    if y_hermite is not None:
        fig.add_trace(go.Scatter(x=x_dense, y=y_hermite, name='Hermite', line=dict(dash='longdash')))

    fig.add_trace(go.Scatter(x=x_dense, y=y_spline_nat, name='Spline tự nhiên', line=dict(color='blue')))
    fig.add_trace(go.Scatter(x=x_dense, y=y_spline_cla, name='Spline kẹp', line=dict(color='red', dash='dash')))

    fig.update_layout(
        title='So sánh 4 phương pháp nội suy',
        xaxis_title='x',
        yaxis_title='y',
        width=1000,
        height=600,
        yaxis=dict(range=[y_min - margin, y_max + margin]),
        legend=dict(x=0.75, y=1)
    )

    fig.show()
```

```
# Tạo dữ liệu đầu vào
x = np.linspace(0, 10, 11)  # 11 điểm cách đều
y = np.array([0, 2, 0.5, 0.2, 0.3, 1.5, 4.5, 5.0, 4.8, 3.0, 0])

compare_interpolations(x, y)
```



Kết luận:

Trong bài thuyết trình này, chúng tôi đã trình bày lại các phương pháp nội suy đa thức như Lagrange, Newton, Neville và Hermite. Mặc dù các phương pháp này có thể cho kết quả chính xác khi số điểm dữ liệu ít, nhưng chúng thường gây ra hiện tượng dao động mạnh khi số điểm tăng, đặc biệt là với đa thức bậc cao.

Phương pháp Cubic Spline giúp khắc phục nhược điểm này bằng cách chia nhỏ khoảng dữ liệu và sử dụng các đa thức bậc ba cho từng đoạn. Nhờ đó, hàm nội suy thu được vừa chính xác tại các điểm dữ liệu, vừa đảm bảo tính mượt mà của đồ thị (liên tục tới đạo hàm bậc hai).

Chúng tôi cũng đã so sánh hai loại spline phổ biến: spline tự nhiên và spline kẹp, đồng thời minh họa bằng ví dụ cụ thể và đồ thị trực quan. Kết quả cho thấy phương pháp spline cho sai số rất nhỏ tại các điểm nút và phù hợp với các bài toán nội suy trong thực tế.

Tóm lại, Cubic Spline là một công cụ mạnh mẽ trong nội suy số, đặc biệt hiệu quả khi làm việc với dữ liệu lớn và yêu cầu đồ thị mượt. Phương pháp này cũng là nền tảng quan trọng cho nhiều ứng dụng trong tính toán khoa học, kỹ thuật và đồ họa máy tính.

Tài liệu tham khảo:

- Giáo trình Numerical Analysis 10th Edition viết bởi Richard L. Burden (Tác giả), J. Douglas Faires (Tác giả), Annette M. Burden (Tác giả)
- Bài giảng môn Giải tích số cho ngành Toán Ứng Dụng viết bởi TS. Bùi Xuân Thắng