TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC

—————————o0o—————————

BÁO CÁO GIẢI TÍCH SỐ

HỌC PHẦN MI3040

Chủ đề 20: Nội suy Newton

Giảng viên hướng dẫn: **Hà Thị Ngọc Yến**

Lớp: **CTTN Toán Tin - K65**

Sinh viên: **Nguyễn Văn Nghiêm – MSSV 20206206**

Hà Nội, Tháng 1 Năm 2022

Mục lục

1 Bài toán nội suy

1.1 Vấn đề nội suy

– Trong thực tế, ta thường gặp những hàm số không biết cụ thể biểu thức giải tích của chúng hoặc quá phức tạp. Bằng việc đo đạc, thực nghiệm thu được bảng số với các y_i tại các điểm x_i tương ứng thuộc đoạn $[a, b]$ nào đó, trong khi ta muốn biết giá trị y tại các điểm $x \neq x_i$.
– Việc tìm chính xác biểu thức của hàm f là điều không thể. Tuy nhiên chúng ta có thể dựa vào các bộ điểm (x, y) đã có để "xấp xỉ" hàm f với sai số chấp nhận được. Tư tưởng này của bài toán chính là các bài toán nội suy - xấp xỉ hàm.
– Nội suy là một công cụ toán học cơ bản được ứng dụng rộng rãi trong nhiều ngành thực nghiệm như công nghệ thông tin, kinh tế, tài chính, dầu khí, xây dựng, y học, truyền hình, điện ảnh và những ngành cần xử lý dữ liệu số khác...
– Có nhiều phương pháp nội suy khác nhau: nội suy tuyến tính, nội suy đa thức, nội suy tam tuyến ... Ở báo cáo này trình bày về nội suy đa thức cụ thể là phương pháp nội suy đa thức Newton.

1.2 Đa thức nội suy

1.2.1 Khái niệm đa thức nội suy

Cho bảng số:

$$f(x_i) = y_i \ (x_i \in [a, b], i = \overline{0, n}) \ (1)$$

Ta cần tìm giá trị của y tại điểm $\overline{x} \neq x_i \ (i = \overline{0, n})$
Từ bảng số xây dựng đa thức $P_n(x_i) = y_i$ với $i = \overline{0, n}$ ($P_n(x)$ bậc n) (2)
Việc thay hàm $f(x)$ bằng một đa thức $P_n(x)$ đơn giản hơn sao cho sự sai lệch của $f(x)$ và $P_n(x)$ là chấp nhận được gọi là xấp xỉ hàm
Đa thức $P_n(x)$ sinh ra từ bảng số (1) thỏa mãn (2) gọi là đa thức nội suy
Các điểm $x_i \in [a, b], i = \overline{0, n}$ là các mốc nội suy
Đặt $\overline{y} = P_n(\overline{x}) \approx f(\overline{x})$ với $\overline{x} \neq x_i, i = \overline{0, n}$ là giá trị nội suy nếu $\overline{x} \in [a, b]$; gọi là giá trị ngoại suy nếu $\overline{x} \notin [a, b]$

1.2.2 Sự duy nhất của đa thức nội suy

Với bộ điểm $x_i, y_i = f(x_i)$ $i = \overline{0, n}$, $x_i \neq x_j$ $\forall i \neq j$ cho trước, đa thức nội suy tồn tại và duy nhất.
**Chứng minh:*

$$P_n(x) = a_0 + a_1x + a_2x^2 + ... + a_nx^n$$

Ta có: $P_n(x_i) = y_i$ *với* $i = \overline{0,n}$

$$\leftrightarrow \begin{cases} a_0 + a_1x_0 + a_2x_0^2 + ... + a_nx_0^n = y_0 \\ a_0 + a_1x_1 + a_2x_1^2 + ... + a_nx_1^n = y_1 \\ ... \\ a_0 + a_1x_n + a_2x_n^2 + ... + a_nx_n^n = y_n \end{cases}$$

hay

$$\begin{pmatrix} 1 & x_0 & x_0^2 & ... & x_0^n \\ 1 & x_1 & x_1^2 & ... & x_1^n \\ ... & ... & ... & ... & ... \\ 1 & x_n & x_n^2 & ... & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ ... \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ ... \\ y_n \end{pmatrix}$$

Ma trận vuông vế trái là ma trận Vandermonde, có
$det = \prod_{0 \le i < j \le n}(x_i - x_j) \neq 0$
Do đó hệ phương trình tồn tại nghiệm duy nhất.

2 Đa thức nội suy Newton

2.1 Ý tưởng phương pháp

Xuất phất từ khai triển Taylor:

$$P_n(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k$$

Với $x_0 = 0$, khai triển Maclaurin:

$$P_n(x) = \sum_{k=0}^{n} \frac{f^{(k)}(0)}{k!}x^k$$

Đạo hàm của hàm số tại điểm x_i được định nghĩa:

$$f^{'}(x_i) = \frac{f(x_i) - f(x_i + h)}{h}$$

Newton nghĩ ra ý tưởng thay thế đạo hàm bằng thương giữa hiệu các giá trị với hiệu các x tương ứng trên bảng giá trị. Từ ta có định nghĩa tỷ hiệu (tỷ sai phân).

2.2 Tỷ hiệu (tỷ sai phân)

2.2.1 Định nghĩa

Xét hàm số: $y = f(x)$; $x \in [a, b]$
Từ bảng số $y_i = f(x_i)$ $i = \overline{0, n}$ trong đó các mốc nội suy là: $a \equiv x_0 < x_1 < ... < x_n \equiv b$
Ta gọi:

$$f[x_{i-1}, x_i] = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} \qquad i = \overline{1, n}$$

là tỷ hiệu cấp 1 của hàm $f(x)$
Tỷ hiệu của tỷ hiệu cấp 1 là tỷ hiệu cấp 2, ký hiệu là:

$$f[x_{i-1}, x_i, x_{i+1}] = \frac{f[x_i, x_{i+1}] - f[x_{i-1}, x_i]}{x_{i+1} - x_{i-1}} \qquad i = \overline{1, n-1}$$

Tỷ hiệu của tỷ hiệu cấp $n-1$ là tỷ hiệu cấp n và ký hiệu:

$$f[x_0, x_1, ..., x_n] = \frac{f[x_1, ..., x_n] - f[x_0, ..., x_{n-1}]}{x_n - x_0}$$

Tỷ hiệu cấp n cần $n+1$ mốc.
Từ định nghĩa ta xây dựng bảng tỷ hiệu:

x	$f(x)$	$f[...,...]$	$f[...,...,...]$	...	$f[x_0, x_1, ..., x_n]$
x_0	$f(x_0)$			...	
x_1	$f(x_1)$	$f[x_0, x_1]$		...	
x_2	$f(x_2)$	$f[x_1, x_2]$	$f[x_0, x_1, x_2]$	...	
...	...	...	...	...	
x_{n-2}	$f(x_{n-2})$	$f[x_{n-3}, x_{n-2}]$	$f[x_{n-4}, x_{n-3}, x_{n-2}]$	...	
x_{n-1}	$f(x_{n-1})$	$f[x_{n-2}, x_{n-1}]$	$f[x_{n-3}, x_{n-2}, x_{n-1}]$	...	
x_n	$f(x_n)$	$f[x_{n-1}, x_n]$	$f[x_{n-2}, x_{n-1}, x_n]$	...	$f[x_0, ..., x_n]$

2.2.2 Tính chất của tỷ hiệu

• Tính chất 1:

$$f[x_0, x_1, ..., x_k] = \sum_{i=0}^{k} \frac{f(x_i)}{\omega'_{k+1}(x_i)}$$

Trong đó:

$$\omega_{k+1}(x) = (x - x_0)(x - x_1)...(x - x_k)$$

$$\omega'_{k+1}(x_i) = (x_i - x_0)...(x_i - x_{i-1})(x_i - x_{i+1})...(x_i - x_k)$$

**Chứng minh:*

– Với $k = 1$

$$f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{f(x_0)}{x_0 - x_1} + \frac{f(x_1)}{x_1 - x_0} = \sum_{i=0}^{1} \frac{f(x_i)}{\omega'_2(x_i)}$$

– Với $k = 2$

$$f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} = \frac{\frac{f(x_1)}{x_1 - x_2} + \frac{f(x_2)}{x_2 - x_1} - \frac{f(x_0)}{x_0 - x_1} - \frac{f(x_1)}{x_1 - x_0}}{x_2 - x_0}$$

$$= \frac{f(x_0)}{(x_0 - x_1)(x_0 - x_2)} + (\frac{1}{x_1 - x_2} - \frac{1}{x_1 - x_0}).\frac{f(x_1)}{x_2 - x_0} + \frac{f(x_2)}{(x_2 - x_0)(x_2 - x_1)}$$

$$= \frac{f(x_0)}{(x_0 - x_1)(x_0 - x_2)} + \frac{f(x_1)}{(x_1 - x_0)(x_1 - x_2)} + \frac{f(x_2)}{(x_2 - x_0)(x_2 - x_1)} = \sum_{i=0}^{2} \frac{f(x_i)}{\omega'_3(x_i)}$$

– Giả sử đúng với $k = m$

$$f[x_0, x_1, ..., x_m] = \sum_{i=0}^{m} \frac{f(x_i)}{\omega'_{m+1}(x_i)}$$

– Cần chứng minh đúng với $k = m + 1$

$$f[x_0, x_1, ..., x_{m+1}] = \frac{f[x_1, ..., x_{m+1}] - f[x_0, ..., x_m]}{x_{m+1} - x_0}$$

$$= \frac{\sum_{i=1}^{m+1}(x_i - x_0)\frac{f(x_i)}{\omega'_{m+2}(x_i)} - \sum_{i=0}^{m}\frac{f(x_i)}{\omega'_{m+1}(x_i)}}{x_{m+1} - x_0}$$

$$= \frac{f(x_0)}{\omega'_{m+2}(x_0)} + \frac{\sum_{i=1}^{m}(\frac{x_i f(x_i) - x_0 f(x_i)}{\omega'_{m+1}(x_i)(x_i - x_{m+1})} - \frac{f(x_i)}{\omega'_{m+1}(x_i)})}{x_{m+1} - x_0} + \frac{f(x_{m+1})}{\omega'_{m+2}(x_{m+1})}$$

$$= \frac{f(x_0)}{\omega'_{m+2}(x_0)} + \sum_{i=1}^{m}\frac{f(x_i)}{\omega'_{m+2}(x_i)} + \frac{f(x_{m+1})}{\omega'_{m+2}(x_i)} = \sum_{i=0}^{m+1}\frac{f(x_i)}{\omega'_{m+2}(x_i)}(dpcm)$$

• Tính chất 2: Tính chất tuyến tính

$$(\alpha f + \beta g)[x_0, x_1, ..., x_k] = \alpha f[x_0, x_1, ..., x_k] + \beta g[x_0, x_1, ..., x_k]$$

**Chứng minh:*

$$(\alpha f + \beta g)[x_0, x_1, ..., x_k] = \sum_{i=0}^{k} \frac{(\alpha f + \beta g)(x_i)}{\omega'_{k+1}(x_i)} = \sum_{i=0}^{k} \frac{\alpha f(x_i) + \beta g(x_i)}{\omega'_{k+1}(x_i)}$$

$$= \alpha \sum_{i=0}^{k} \frac{f(x_i)}{\omega'_{k+1}(x_i)} + \beta \sum_{i=0}^{k} \frac{g(x_i)}{\omega'_{k+1}(x_i)} = \alpha f[x_0, x_1, ..., x_k] + \beta g[x_0, x_1, ..., x_k](dpcm)$$

• Tính chất 3: Tính chất đối xứng
Từ tính chất số 1 có thể thấy rằng tỷ hiệu cấp k của f không phụ thuộc vào trình tự sắp xếp các mốc nội suy. Nên ta có:

$$f[x_0, x_1, ..., x_k] = f[x_{\sigma(0)}, x_{\sigma(1)}, ..., x_{\sigma(k)}]$$

• Tính chất tỷ hiệu của đa thức:

1) Tỷ hiệu của hằng số thì bằng "0"
**Chứng minh:*
Nếu $f(x) = C(const)$ thì

$$f[x_i, x_{i+1}] = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} = \frac{C - C}{x_{i+1} - x_i} = 0$$

2) Tỷ hiệu cấp m của đa thức bậc n có tính chất:

+ Nếu $m = n$ thì tỷ hiệu cấp n là hằng số
+ Nếu $m > n$ thì tỷ hiệu cấp m bằng "0"

**Chứng minh:*
Xét hàm: $f(x) = P(x) = a_0 + a_1x + a_2x^2 + ... + a_nx^n \ (a_n \neq 0)$
Xét: $h(x) = x^k \ k \in \mathbb{N}$

$$h[x_i, x_{i+1}] = \frac{h(x_{i+1}) - h(x_i)}{x_{i+1} - x_i} = \frac{x_{i+1}^k - x_i^k}{x_{i+1} - x_i} = x_{i+1}^{k-1} + x_{i+1}^{k-2}x_i + ... + x_i^{k-1}$$

Là đa thức cấp $k-1$. Tiếp tục xét tỷ hiệu cấp 2, ta được đa thức bậc $k-2$ ... tỷ hiệu cấp k của $h(x)$ là hằng số, tỷ hiệu cấp $k+1$ thì bằng "0".
Như vậy, tỷ hiệu cấp n của $f(x)$ là hằng số và tỷ hiệu cấp $m > n$ của $f(x)$ bằng "0".

2.3 Đa thức nội suy Newton

• Định lý 1:

$$P_n(x) = f(x_0) + (x - x_0)f[x_0, x_1] + (x - x_0)(x - x_1)f[x_0, x_1, x_2]$$
$$+... + (x - x_0)(x - x_1)...(x - x_{n-1})f[x_0, ..., x_n] \qquad (1)$$

Là đa thức nội suy tương ứng với các mốc nội suy $y_i = f(x_i)$ $i = \overline{0, n}$
Chứng minh:
Gọi $L_k(x)$ là đa thức nội suy Lagrange bậc k có các mốc nội suy x_i, $i = \overline{0, k}$
Ta có: $P_k(x) = L_k(x) - L_{k-1}(x)$ $(*)$ và $P_k(x_i) = 0$, $i = \overline{0, k-1}$
$\rightarrow P_k(x) = A(x - x_0)(x - x_1)...(x - x_{k-1}) = A\omega_k(x)$
Thay $x = x_k$ ta được:
$P_k(x_k) = L_k(x_k) - L_{k-1}(x_k) = A\omega_k(x_k)$

$$\rightarrow A = \frac{L_k(x_k) - L_{k-1}(x_k)}{\omega_k(x_k)} = \frac{f(x_k) - L_{k-1}(x_k)}{\omega_k(x_k)}$$
$$= \frac{f(x_k)}{\omega_k(x_k)} + \frac{\omega_k(x_k)\sum_{i=0}^{k-1}\frac{f(x_i)}{(x_k - x_i)\omega_k'(x_i)}}{\omega_k(x_k)} = \frac{f(x_k)}{\omega_k(x_k)} + \sum_{i=0}^{k-1}\frac{f(x_i)}{\omega_{k+1}'(x_i)}$$

Có: $\omega_k(x_k) = (x_k - x_0)(x_k - x_1)...(x_k - x_{k-1}) = \omega_{k+1}'(x_k)$

$$\rightarrow A = \frac{f(x_k)}{\omega_{k+1}'(x_k)} + \sum_{i=0}^{k-1}\frac{f(x_i)}{\omega_{k+1}'(x_i)} = \sum_{i=0}^{k}\frac{f(x_i)}{\omega_{k+1}'(x_i)} = f[x_0, x_1, ..., x_k]$$

Từ $(*)$ ta có:

$$L_k(x) = P_k(x) + L_{k-1}(x) = A\omega_k(x) + L_{k-1}(x)$$
$$= f[x_0, x_1, ..., x_k](x - x_0)(x - x_1)...(x - x_{k-1}) + L_{k-1}(x)$$
$$= f[x_0, x_1, ..., x_k](x - x_0)(x - x_1)...(x - x_{k-1})$$
$$+f[x_0, x_1, ..., x_{k-1}](x - x_0)(x - x_1)...(x - x_{k-2}) + ... + f[x_0, x_1](x - x_0) + f(x_0)$$

Như vậy, đa thức (1) chính là đa thức nội suy Lagrang $L_n(x)$ • Định nghĩa:
Công thức (1) được gọi là công thức nội suy Newton, đa thức trong (1) được gọi là đa thức nội suy Newton
Khi $x_0 < x_1 < ... < x_n$ thì (1) được gọi là đa thức nội suy Newton tiến
Khí $x_0 > x_1 > ... > x_n$ thì (1) được gọi là đa thức nội suy Newton lùi
Công thức (1) sử dụng các yếu tố đầu bảng của bảng tỷ hiệu còn được gọi là đa thức nội suy Newton tiến xuất phát từ x_0
Nếu các mốc nội suy đánh số lại theo thứ tự:

$$x_n, x_{n-1}, ..., x_1, x_0$$

Khi đó đa thức nội suy Newton có dạng:

$$P_n(x) = f(x_n) + (x - x_n)f[x_n, x_{n-1}] + (x - x_n)(x - x_{n-1})f[x_n, x_{n-1}, x_{n-2}]$$
$$+... + (x - x_n)(x - x_{n-1})...(x - x_1)f[x_n, x_{n-1}, ..., x_0] \qquad (2)$$

Công thức (2) sử dụng các yếu tố cuối bảng của bảng tỷ hiệu được gọi là đa thức nội suy Newton lùi xuất phát từ x_n
Hai đa thức nội suy Newton tiến và lùi đều sử dụng chung bảng tỷ hiệu.
• Định lý 2:

$$R_n(x) = f(x) - P_n(x)$$
$$x \neq x_i, i = \overline{0, n} \rightarrow R_n(x) = f[x, x_0, x_1, ..., x_n]\omega_{n+1}(x)$$

Chứng minh:

$$f[x, x_0] = \frac{f(x) - f(x_0)}{x - x_0} \rightarrow f(x) = f(x_0) + (x - x_0)f[x, x_0]$$
$$f[x, x_0, x_1] = \frac{f[x, x_0] - f[x_0, x_1]}{x - x_1} \rightarrow f[x, x_0] = f[x_0, x_1] + (x - x_1)f[x, x_0, x_1]$$
$$\rightarrow f(x) = f(x_0) + (x - x_0)f[x_0, x_1] + (x - x_0)(x - x_1)f[x, x_0, x_1]$$
$$...$$
$$f(x) = f(x_0)+(x-x_0)f[x_0, x_1]+...+(x-x_0)(x-x_1)...(x-x_{n-1})f[x_0, x_1, ..., x_n]$$
$$+(x - x_0)(x - x_1)...(x - x_n)f[x, x_0, x_1, ..., x_n]$$
$$\rightarrow f(x) = P_n(x) + (x - x_0)(x - x_1)...(x - x_n)f[x, x_0, x_1, ..., x_n]$$
$$\rightarrow R_n(x) = f(x) - P_n(x) = (x - x_0)(x - x_1)...(x - x_n)f[x, x_0, x_1, ..., x_n]$$

hay

$$R_n(x) = f[x, x_0, x_1, ..., x_n]\omega_{n+1}(x)$$

Và $R_n(x)$ chính là công thức đánh giá sai số của đa thức nội suy Newton
Mà ta lại có công thức đánh giá sai số chung của đa thức nội suy:

$$R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(x)$$

$$\rightarrow \frac{f^{(n+1)}(\xi)}{(n+1)!} = R_n(x) = f[x, x_0, x_1, ..., x_n]$$

• Nhận xét:

– Chỉ nên sử dụng đa thức nội suy để tính gần đúng $x \in [a, b]$, nằm ngoài $[a, b]$ sẽ có sai số rất lớn.

- Độ dài nội suy lý tưởng để có được sai số bé là $|a-b| \leq 1$ (hoặc ≤ 2).
- Với $n+1$ mốc nội suy cho trước, ta sẽ chỉ thu được một đa thức nội suy và một sai số cố đinh. Để có được đa thức nội suy với sai số chấp nhận được thì chúng ta chỉ có cách là xử lí dữ liệu đầu vào, tối ưu mốc nội suy.
- Với đa thức nội suy Newton, chúng ta không cần phải sắp xếp các mốc nội suy, cũng như chúng ta có thể xuất phát từ mốc x_k bất kì.
- Đa thức nội suy Newton cũng chính là đa thức nội suy Lagrange chỉ khác về cách trình bày.
- Nếu thêm mốc nội suy (x_{n+1}, y_{n+1}) đa thức $P_{n+1}(x)$ được tính như sau:

$$P_{n+1}(x) = P_n(x) + (x-x_0)(x-x_1)...(x-x_n)f[x_0, x_1, ..., x_{n+1}]$$

2.4 Thuật toán và ví dụ

2.4.1 Thuật toán

Các gói sử dụng:
1. *main*: thuật toán chính
2. *Input*: nhập x, y từ file
3. $BuildBTH$: kết nạp mốc $x[i]$ vào bảng tỷ hiệu
4. $BuildBTT$: kết nạp mốc $x[i]$ vào bảng tính tích theo lược đồ Hoocner
5. fx: kết nạp mốc $x[i]$ vào đa thức f
6. $Newton\_interpolation\_Forward$: xây dựng đa thức nội suy Newton tiến
7. $Newton\_interpolation\_Backward$: xây dựng đa thức nội suy Newton lùi
8. $fx0$: ước tính giá trị tại $x0$ theo lược đồ Hoocner
9. add_Newton : thêm mốc nội suy
10. *plot*: vẽ đồ thị

main

Input: flie các mốc nội suy, điểm cần ước lượng giá trị: $x0$, số lượng thêm mốc: k
Output: hệ số đa thức f, giá trị ước lượng tại $x0$: $value$, đồ thị đa thức f

begin
$x, y \longleftarrow Input()$
$n \longleftarrow \text{len}(x) - 1$
In các mốc nội suy x
Khởi tạo các ma trận không: $BTH(n+1), BTT(n+1), f(n+1)$
$BTH[0], BTT[0], f[0] \longleftarrow y[0], 1, y[0]$
$f, BTH, BTT \longleftarrow Newton\_interpolation\_Forward(BTH, BTT, f, x, y, n)$
$value \longleftarrow fx0(f, n, x0)$
In hệ số đa thức f và giá trị $value$
$plot(x, y, f, n)$
if $k <= 0$ **then**: Dừng chương trình
end if
for $i = 1 \rightarrow k$ **do**
Nhập mốc và giá trị nội suy $\{xt, yt\}$
if xt **not in** x **then**
$n \longleftarrow n + 1$
append $\{xt, yt\}$ vào $\{x, y\}$
$f, BTH, BTT \longleftarrow add\_Newton(BTH, BTT, f, x, y, n)$
end if
end for
In hệ số đa thức f và $value$
$plot(x, y, f, n)$
end

Gói **BuildBTH**

Input: BTH, x, y, i
Output: TH_row_i

begin
    Khởi tạo ma trận không $TH\_row\_i(\text{len}(x))$
    $TH\_row\_i[0] \longleftarrow y[i]$
    for $j = 1 \rightarrow i$ **do**
        $TH\_row\_i[j] \longleftarrow \frac{TH\_row\_i[j-1]-BTH[j-1]}{x[i]-x[i-j]}$
    end for
    return TH_row_i
end

Gói **BuildBTT**

Input: BTT, x, i
Output: TT_row_i

begin
    Khởi tạo ma trận không $TT\_row\_i(\text{len}(x))$
    $TT\_row\_i[i+1] \longleftarrow 1$
    for $j = i \rightarrow 1$ **do**
        $TT\_row\_i[j] \longleftarrow BTT[j-1] - x[i] * BTT[j]$
    end for
    $TT\_row\_i[0] \longleftarrow -i * BTT[0]$
    return TT_row_i
end

Gói **fx**

Input: BTH, BTT, i, f
Output: f

begin
  for $j = 0 \to i$ **do**
    $f[j] \longleftarrow f[j] + BTH[i] * BTT[j]$
  end for
  return f
end

Gói **Newton_interpolation_Forward**

Input: BTH, BTT, f, x, y, n
Output: f, BTH, BTT

begin
  for $i = 1 \to n$ **do**
    $BTH \longleftarrow BuildBTH(BTH, x, y, i)$
    $BTT \longleftarrow BuildBTT(BTH, x, i-1)$
    $f \longleftarrow fx(BTH, BTT, i, f)$
  end for
  return f, BTH, BTT
end

Gói **Newton_interpolation_Backward**

Input: BTH, BTT, f, x, y, n
Output: f, BTH, BTT

begin
  x.reverse()
  y.reverse()
  $BTH \longleftarrow y[0]$
  $f[0] \longleftarrow y[0]$
  return Newton_interpolation_Forward
end

Gói **fx0**

Input: $f, n, x0$
Output: $value$

begin
    $value \longleftarrow f[n]$
    for $i = n - 1 \to 0$ **do**
        $value \longleftarrow value * x0 + f[j]$
    end for
    return $value$
end

Gói **add_Newton**

Input: BTH, BTT, f, x, y, n
Output: f, BTH, BTT

begin
    $BTH \longleftarrow BuildBTH(BTH, x, y, n)$
    $BTT \longleftarrow BuildBTT(BTT, x, n - 1)$
    Tạo thêm 1 cột cho f
    $f \longleftarrow fx(BTH, BTT, n, f)$
    return f, BTH, BTT
end

- Khối lượng tính toán:
Gói $BuildBTH$: $3i$
Gói $BuildBTT$: $2i + 2$
Gói fx: $2(i + 1)$
Gói $Newton\_interpolation\_Forward$:
$\sum_1^n(3i + 2i + 2i + 2) = \sum_1^n(7i + 2) = 9 + 16 + 23 + ... + 7n + 2$
$= \frac{(9+7n+2)n}{2} = \frac{7}{2}n^2 + \frac{11}{2}n$

Tổng quát: $O(\frac{7}{2}n^2)$

2.4.2 Ví dụ

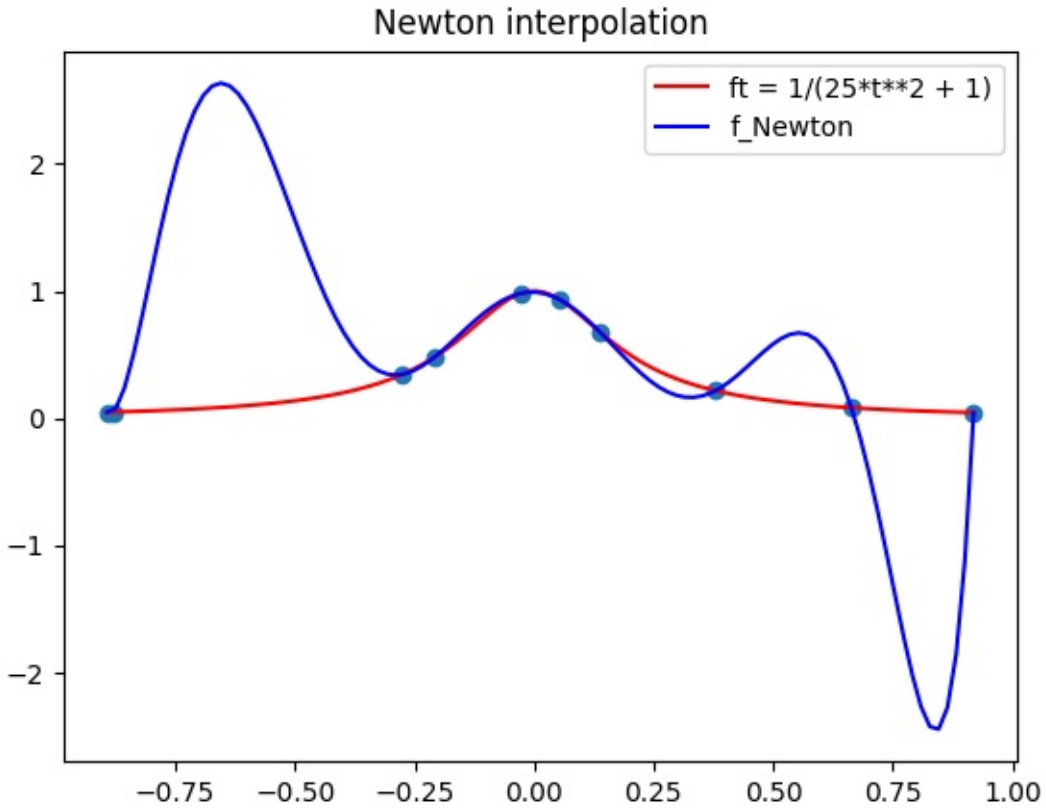
Ví dụ 1:
Để kiểm tra xem thuật toán cùng với phương pháp Newton này có nội suy tốt hay không, ta xấp xỉ dựa trên 1 hàm đã biết.
Chọn hàm

$$f(x) = \frac{1}{25x^2 + 1}$$

Với 10 mốc dữ liệu trên đoạn $[-1, 1]$:

x	-0.8929	-0.881	-0.2753	-0.2089	-0.0276
y	0.0478	0.0490	0.3455	0.4782	0.9813
x	0.0539	0.1371	0.3797	0.6637	0.9194
y	0.9323	0.6803	0.2172	0.0832	0.0452

Thực hiên xấp xỉ hàm và ước lượng tại 3 điểm -0.75, 0, 0.75. Dưới đây là đồ thị thu được:



Đường màu đỏ là hàm $f(x) = \frac{1}{25x^2+1}$ còn màu xanh là đồ thị đa thức nội suy Newton.

Kết quả hệ số đa thức nội suy, và giá trị ước lượng tại các điểm thu được như sau:

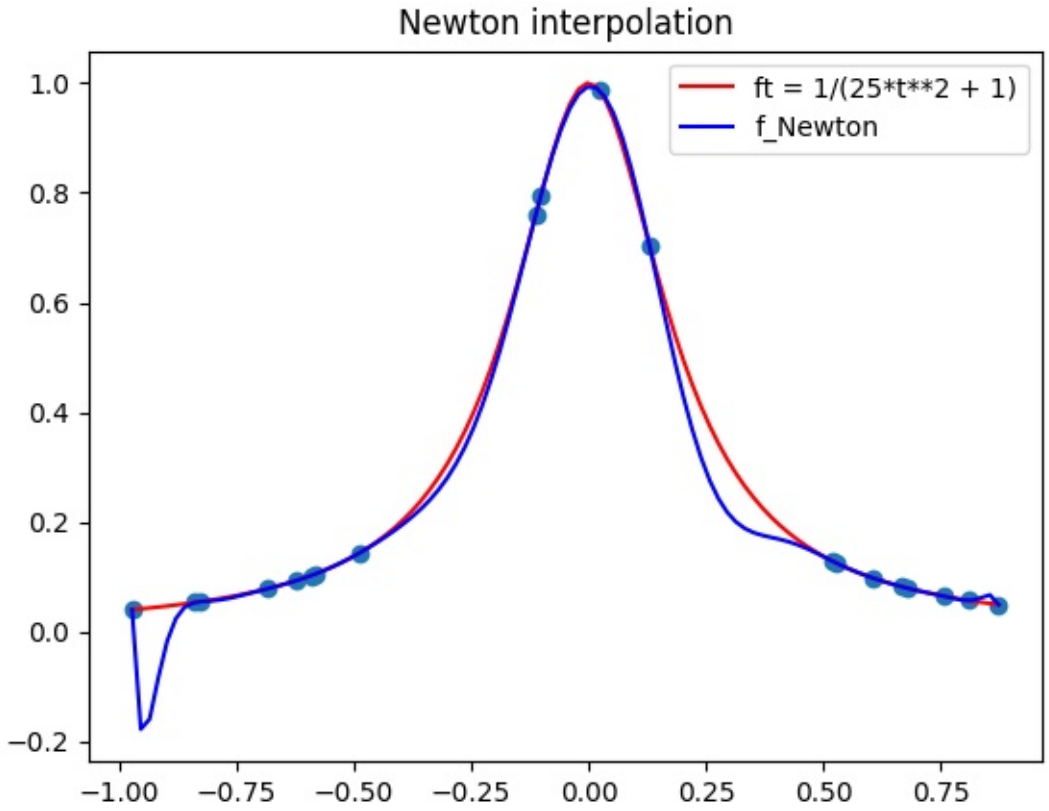
```
Các mốc nội suy:
[-0.8929, -0.881, -0.2753, -0.2089, -0.0276, 0.0539, 0.1371, 0.3797, 0.6637, 0.9194]
Hệ số đa thức nội suy Newton là: (bắt đầu từ hệ số tự do)
[   0.99074   -0.14618  -17.78761    0.14817  131.07465  -19.50782 -277.97317   22.80235  173.31222    0.03034]
Giá trị cần tính tại -0.75  là: 1.965990521427645
Giá trị cần tính tại 0  là: 0.9907421156959585
Giá trị cần tính tại 0.75  là: -1.2948068583413153
Nhập số lượng thêm mốc nội suy: 0
PS C:\Users\Admin\.vscode>
```

Đa thức nội suy thu được:
$P(x) = 0.03034x^9 + 173.31222x^8 + 22.80235x^7 - 277.97317x^6 - 19.50782x^5 + 131.07465x^4 + 0.14817x^3 - 17.78761x^2 - 0.14618x + 0.99074$
Giá trị ước lượng tại 0: 0.9907421156959585
Giá trị ước lượng tại -0.75: 1.965990521427645
Giá trị ước lượng tại 0.75: -1.2948068583413153
Đa thức tìm được có hình dáng tương đối giống với $f(x)$.
Giá trị tại 0 được xấp xỉ tốt ($f(0) = 1 \approx 0.9907$). Nhưng giá trị xấp xỉ tại -0.75 và 0.75 lại xấu, sai lệch lớn ($f(-0.75) = f(0.75) \approx 0.0664$), thậm chí giá trị tại 0.75 còn nhỏ 0 (hàm $f(x)$ luôn không âm).
Lí do bởi vì do các mốc nội suy ta dùng phân bố không được đồng đều, tập trung chủ yếu ở trung tâm nên xấp xỉ ở gần 2 đầu mút sai lệch nhiều và số lượng chưa đủ nhiều.

Ví dụ 2:
Bây giờ, với 20 mốc nội suy xem liệu kết quả có tốt hơn không.

x	-0.9719	-0.8282	-0.8262	-0.6835	-0.6227
y	0.0406	0.0539	0.0554	0.0789	0.0935
x	-0.5911	-0.5833	-0.4867	-0.1122	-0.1014
y	0.1027	0.1052	0.1445	0.7606	0.7955
x	0.0235	0.1298	0.5185	0.5256	0.6066
y	0.9864	0.7036	0.1295	0.1265	0.0980
x	0.6681	0.6799	0.7547	0.8101	0.8731
y	0.0822	0.0796	0.0656	0.0574	0.0499

Thu được đồ thị:




```
Các mốc nội suy:
[-0.9719, -0.8382, -0.8262, -0.6835, -0.6227, -0.5911, -0.5833, -0.4867, -0.1122, -0.1014, 0.0235, 0.1298, 0.5185, 0.5256, 0.6066, 0.668
1, 0.6799, 0.7547, 0.8101, 0.8731]
Hệ số đa thức nội suy Newton là: (bắt đầu từ hệ số tự do)
[     0.9934       0.19873    -20.9183     -15.08071    243.43603    245.9185   -1623.95086  -1899.50123   6675.36742   8456.50216 -1757
4.19759 -23347.50461  29785.53501  40694.47491 -31475.09826 -43618.61913  18878.38625  26256.83615  -4907.89054  -6793.37277]
Giá trị cần tính tại -0.75  là: 0.06436310410342849
Giá trị cần tính tại 0  là: 0.9934020433750895
Giá trị cần tính tại 0.75  là: 0.06649565545061997
Nhập số lượng thêm mốc nội suy: 0
```

Giá trị ước lượng tại 0: 0.99340204337508954
Giá trị ước lượng tại -0.75:0.064363104103428495
Giá trị ước lượng tại 0.75: 0.066495655450619973
Rõ ràng trong trường hợp này dùng 20 mốc nội suy đã có độ chính xác cao hơn so với dùng 10 mốc. Nhưng bù lại đa thức nội suy thu được lại khá phức tạp (bậc 19).
Ngoài ra, từ đồ thị mặc dù chỉ có 4 mốc ở đoạn trung tâm nhưng dáng đồ thị vẫn tốt điều đó cho thấy tại các điểm gần trung tâm được nội suy với độ chính xác cao hơn so với các điểm gần 2 đầu mút. Điều này được lý giải bởi công thức Sai số, tại các điểm trung tâm sai số sẽ bé hơn.

Ví dụ 3:
Giờ ta sẽ xem liệu gói thêm mốc nội suy của ta có chạy tốt hay không. Để dễ dàng kiểm tra, thực hiện xấp xỉ 1 đa thức và hiển nhiên khi đó kết quả trả về phải là đa thức đó. Chọn:

$$f(x) = x^4 + x^3 + x^2 + x + 1$$

Để ra được đa thức trên cần 5 mốc nội suy. Nhưng ở đây bắt đầu từ 2 mốc nội suy:

x	1	2
y	5	31

Thực hiện thêm 3 mốc nội suy bất kì sinh ra từ $f(x)$ khác 2 mốc trên và tính tại điểm $x0 = 10$.
Kiểm tra lần 1 thêm 3 mốc:

x	0	-1	3
y	1	1	121

Kiểm tra lần 2 thêm 4 mốc trong đó có 1 mốc trùng:

x	0	-1	-2	1
y	1	1	11	5

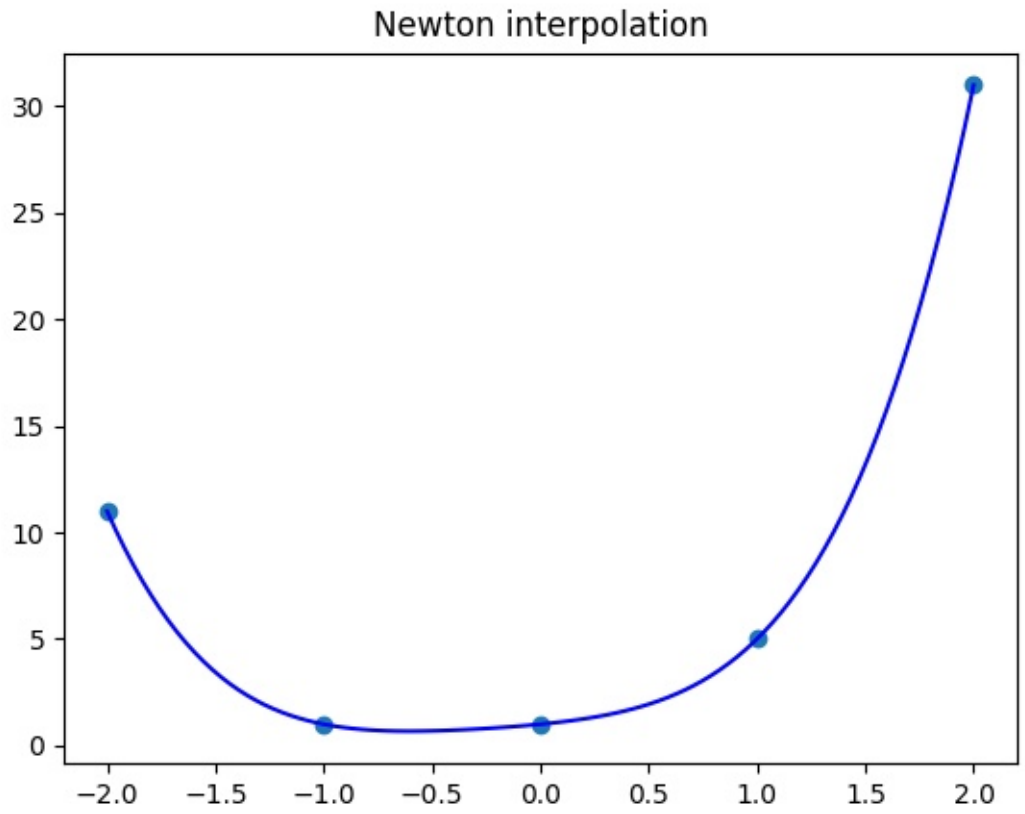
Lần 1:

```
PS C:\Users\Admin\.vscode> & C:/Users/Admin/AppData/Local/Programs/Python/Python39/python.exe
n.py
Mời nhập giá trị cần tính: 10
Các mốc nội suy:
[1.0, 2.0]
Hệ số đa thức nội suy Newton là: (bắt đầu từ hệ số tự do)
[-21.  26.]
Giá trị cần tính tại 10.0  là: 239.0
Nhập số lượng thêm mốc nội suy: 4
Nhập mốc và giá trị nội suy: 0 1
Nhập mốc và giá trị nội suy: -1 1
Nhập mốc và giá trị nội suy: -2 11
Nhập mốc và giá trị nội suy: 1 5
Hệ số đa thức nội suy Newton là: (bắt đầu từ hệ số tự do)
[1. 1. 1. 1. 1.]
Giá trị cần tính tại 10.0  là: 11111.0
```

Lần 2:

```
PS C:\Users\Admin\.vscode> & C:/Users/Admin/AppData/Local/Programs/Python/Python39/python.exe
n.py
Mời nhập giá trị cần tính: 10
Các mốc nội suy:
[1.0, 2.0]
Hệ số đa thức nội suy Newton là: (bắt đầu từ hệ số tự do)
[-21.  26.]
Giá trị cần tính tại 10.0  là: 239.0
Nhập số lượng thêm mốc nội suy: 3
Nhập mốc và giá trị nội suy: 0 1
Nhập mốc và giá trị nội suy: -1 1
Nhập mốc và giá trị nội suy: 3 121
Hệ số đa thức nội suy Newton là: (bắt đầu từ hệ số tự do)
[1. 1. 1. 1. 1.]
Giá trị cần tính tại 10.0  là: 11111.0
```

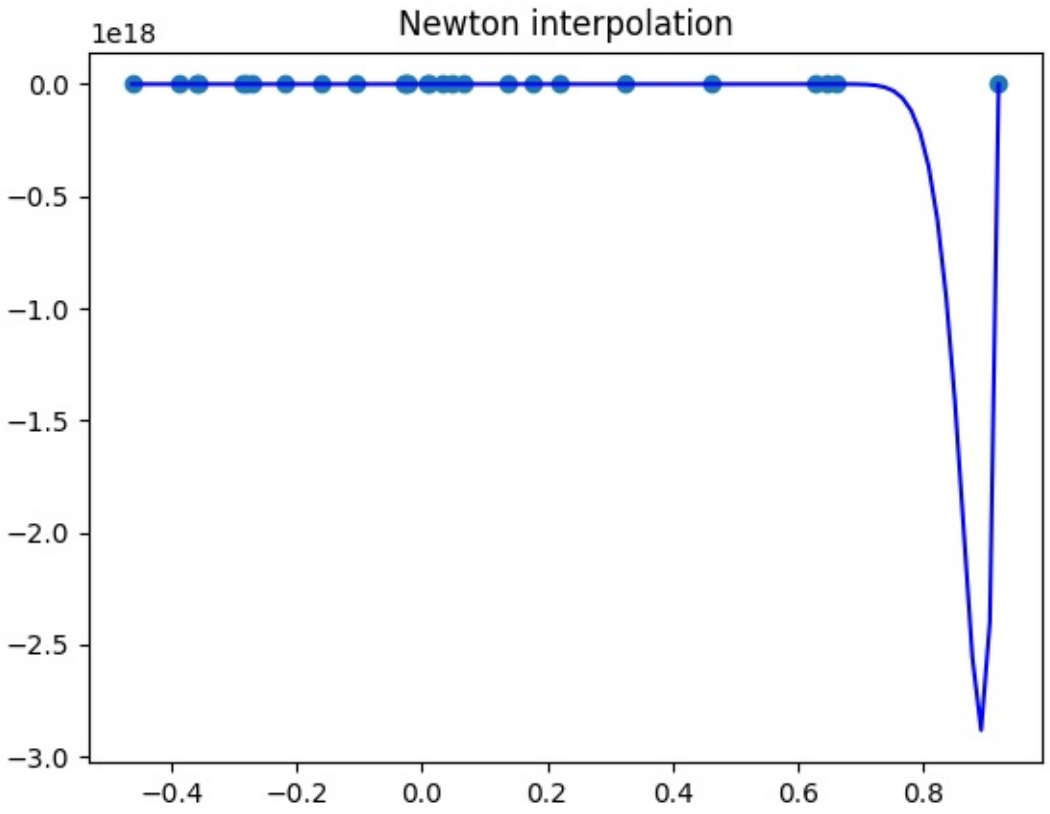
Qua 2 lần kiểm tra, kết quả cuối cùng trả về đều đúng với đa thức ban đầu $f(x)$. Chứng tỏ gói thêm mốc nội suy chạy ổn định.
Đồ thị $f(x)$:



Ví dụ 4:
Xét bộ dữ liệu Mực nước biển trung bình của trái đất phụ thuộc theo nhiệt độ trung bình[1]:

x	y	x	y	x	y
-0.463	-133.8307292	-0.105	-191.5807292	0.138	-33.00572917
-0.387	-150.7140625	-0.028	-97.58072917	0.177	-46.98072917
-0.360	-144.0890625	-0.027	-118.2473958	0.220	-17.9640625
-0.357	-176.7640625	-0.023	-69.2890625	0.323	-34.74739583
-0.288	-132.5557292	0.008	-49.2640625	0.462	-2.780729167
-0.280	-148.8723959	0.010	-84.0390625	0.628	0.8859375
-0.270	-168.6557292	0.032	-66.4390625	0.645	42.341628
-0.218	-130.4973958	0.048	-73.88072917	0.663	13.76927083
-0.160	-124.7307292	0.066	-103.5223958	0.921	55.11239131

Với 27 mốc trên ta sẽ ước tính tại điểm $x0 = -0.199$
Đồ thị thu được:

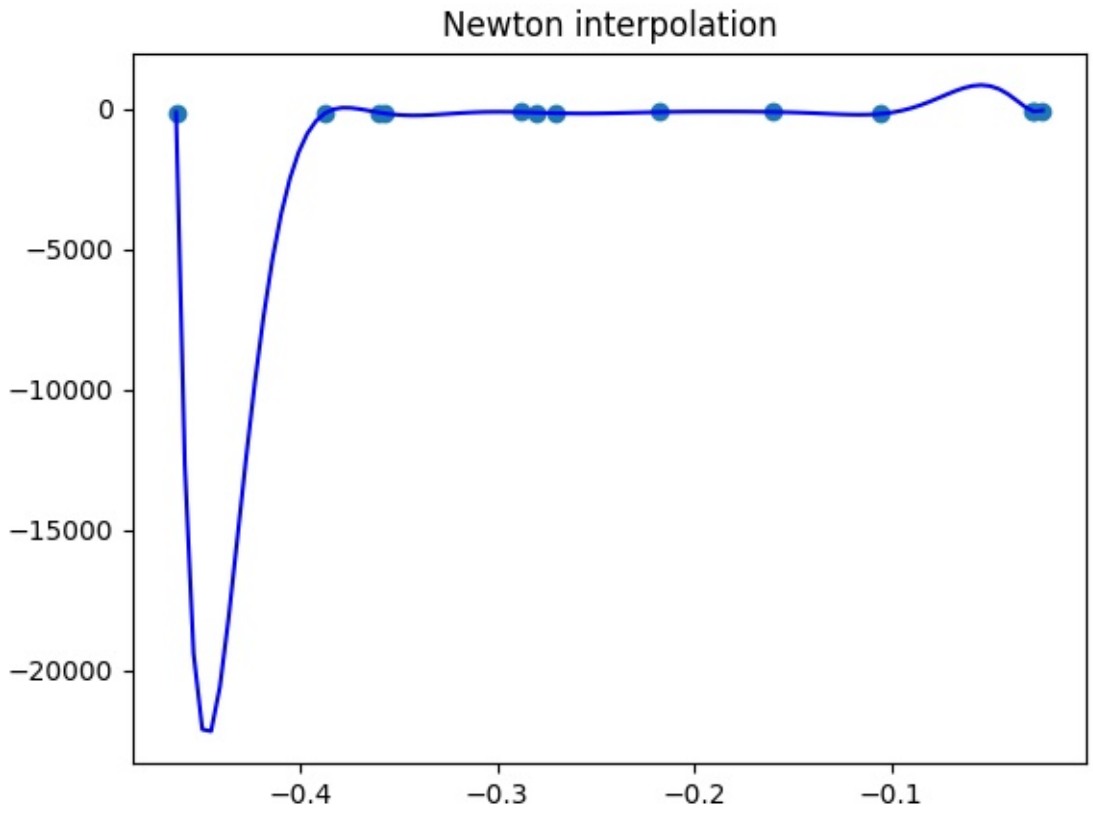


[1]Bộ dữ liệu được cung cấp bởi giảng viên Hà Thị Ngọc Yến - Đại học Bách khoa Hà Nội

Kết quả thu được:

```
PS C:\Users\Admin\.vscode> & C:/Users/Admin/AppData/Local/Programs/Python/Python39/python.exe c:/Users/Admin/.vscode/Newton_interpolatio
n.py
Mời nhập giá trị cần tính: -0.199
Các mốc nội suy:
[-0.463, -0.387, -0.36, -0.357, -0.288, -0.28, -0.27, -0.218, -0.16, -0.105, -0.028, -0.027, -0.023, 0.008, 0.01, 0.032, 0.048, 0.066, 0
.138, 0.177, 0.22, 0.323, 0.462, 0.628, 0.645, 0.663, 0.921]
Hệ số đa thức nội suy Newton là: (bắt đầu từ hệ số tự do)
[ 1.61427e+02 -2.84061e+04 -3.96186e+05  9.18519e+07 -6.73449e+08 -8.23667e+10  1.18758e+12  1.72378e+13 -2.53611e+14 -1.92038e+15  1.99
606e+16  1.23510e+17 -7.35388e+17 -4.47557e+18  1.37142e+19  9.07814e+19 -1.32438e+20 -1.04114e+21  6.96219e+20  6.91483e+21 -2.48320e+2
1 -2.68704e+22  9.00444e+21  5.79359e+22 -2.77580e+22 -5.43887e+22  3.83133e+22]
Giá trị cần tính tại -0.199  là: 453975.8655364534
Nhập số lượng thêm mốc nội suy: 0
PS C:\Users\Admin\.vscode>
```

Với ví dụ này nội suy Newton đã ước tính tại nhiệt độ -0.199 mực nước biển trung bình là: 453975.8655364534
Rõ ràng kết quả này không được hợp lí với thực tế mặc dù độ dài nội suy ≤ 2.
Để lí giải tại sao, giờ chúng ta sẽ chỉ dùng với các mốc nhiệt độ âm. (Từ -0.463 $\rightarrow$ -0.023)
Đồ thị thu được:



Kết quả thu được:

```
PS C:\Users\Admin\.vscode> & C:/Users/Admin/AppData/Local/Programs/Python/Python39/python.exe c:/Users/Admin/.vscode/Newton_interpolatio
n.py
Mời nhập giá trị cần tính: -0.199
Các mốc nội suy:
[-0.463, -0.387, -0.36, -0.357, -0.288, -0.28, -0.27, -0.218, -0.16, -0.105, -0.028, -0.027, -0.023]
Hệ số đa thức nội suy Newton là: (bắt đầu từ hệ số tự do)
[1.62153e+04 2.02478e+06 9.86057e+07 2.50613e+09 3.83373e+10 3.81844e+11 2.58476e+12 1.21264e+13 3.94615e+13 8.73953e+13 1.25608e+14 1.0
5563e+14 3.93431e+13]
Giá trị cần tính tại -0.199  là: -112.98373794126564
Nhập số lượng thêm mốc nội suy: 0
PS C:\Users\Admin\.vscode>
```

Tại nhiệt độ -0.199 mực nước biển trung bình là: -112.98373794126564
Rõ ràng so với kết quả trên (453975.8655364534) thì kết quả này hợp lí với thực tế hơn.
Mặc dù ở đây độ dài nội suy gần đạt mức lí tưởng (≤ 2) nhưng do sử quá nhiều mốc nội suy (27 mốc) nên dẫn đến Newton tìm ra đa thức phức tạp (bậc 26) và 1 phần do điểm -0.199 cách khá xa vị trí trung tâm cùng với đó là phân bố không đồng đều. Nên khi chỉ còn sử dụng đến 13 mốc (không dùng đến các mốc dương) thì kết quả trở lên hợp lí hơn so với dùng toàn bộ mốc.

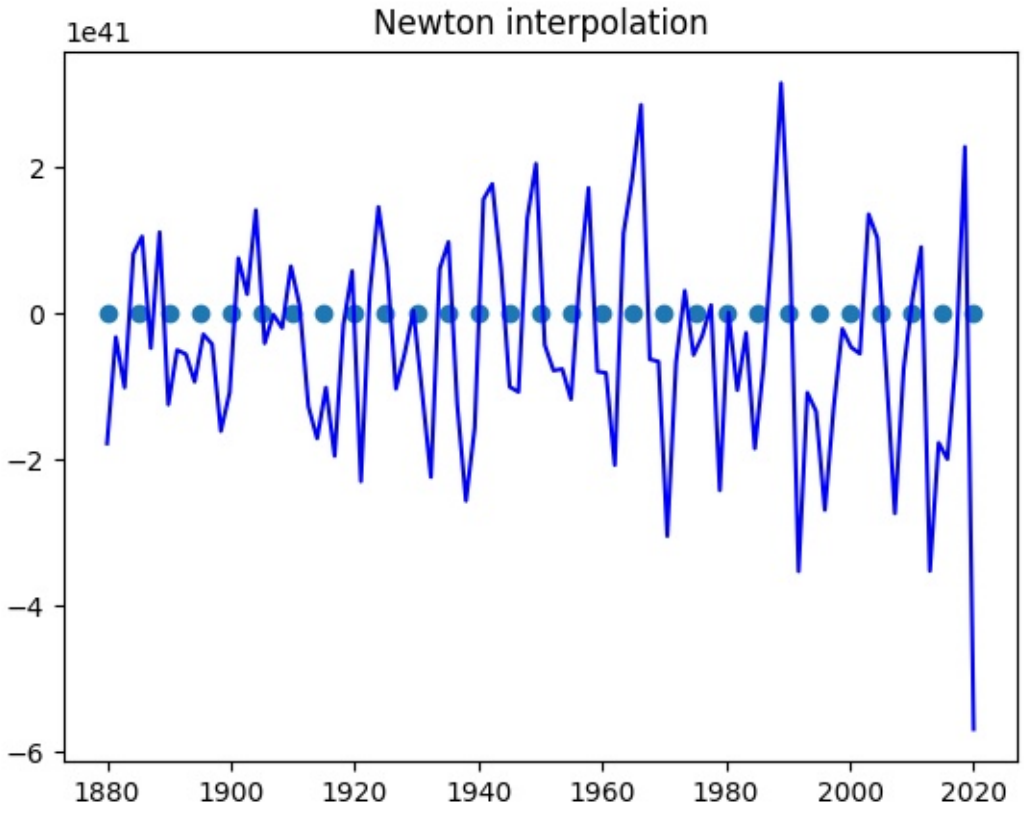
Ví dụ 5:
Xét bộ dữ liệu Nhiệt độ trung bình của Trái Đất qua các năm[2]:

x	y	x	y	x	y
1880	-0.166	1930	-0.158	1980	0.259
1885	-0.328	1935	-0.199	1985	0.119
1890	-0.348	1940	0.124	1997	0.45
1895	-0.227	1945	0.09	1995	0.446
1900	-0.08	1950	-0.174	2000	0.393
1905	-0.263	1955	-0.141	2005	0.675
1910	-0.436	1960	-0.025	2010	0.723
1915	-0.145	1965	-0.107	2015	0.894
1920	-0.276	1970	0.025	2020	1.021
1925	-0.223	1975	-0.013		

Bộ dữ liệu trên là 1 bộ dữ liệu cách đều với khoảng cách 5 năm.
Giờ thực hiện chương trình ước lượng tại $x0 = 1938$.

[2]Bộ dữ liệu được cung cấp bởi giảng viên Hà Thị Ngọc Yến - Đại học Bách khoa Hà Nội

Đồ thị thu được:

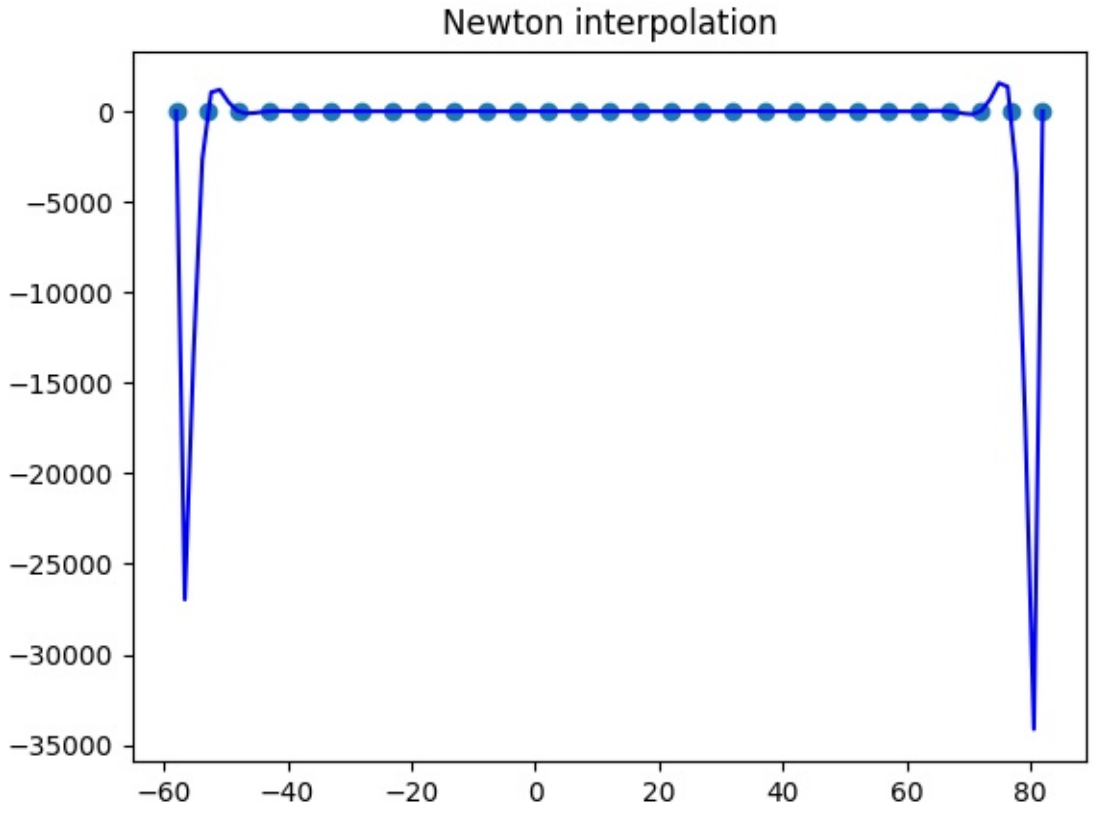


Kết quả thu được:

```
PS C:\Users\Admin\.vscode> & C:/Users/Admin/AppData/Local/Programs/Python/Python39/python.exe c:/Users/Admin/.vscode/Newton_Interpolatio
n.py
Mời nhập giá trị cần tính: 1938
Các mốc nội suy:
[1880.0, 1885.0, 1890.0, 1895.0, 1900.0, 1905.0, 1910.0, 1915.0, 1920.0, 1925.0, 1930.0, 1935.0, 1940.0, 1945.0, 1950.0, 1955.0, 1960.0,
 1965.0, 1970.0, 1975.0, 1980.0, 1985.0, 1990.0, 1995.0, 2000.0, 2005.0, 2010.0, 2015.0, 2020.0]
Hệ số đa thức nội suy Newton là: (bắt đầu từ hệ số tự do)
[ 5.02927e+49 -7.22579e+47  5.00535e+45 -2.22584e+43  7.13795e+40 -1.75795e+38  3.45755e+35 -5.57533e+32  7.50879e+29 -8.56091e+26  8.34
503e+23 -7.00576e+20  5.09170e+17 -3.21493e+14  1.76709e+11 -8.46083e+07  3.52652e+04 -1.27697e+01  4.00309e-03 -1.08076e-06  2.49471e-1
0 -4.87487e-14  7.95615e-18 -1.06459e-21  1.13760e-25 -9.33584e-30  5.52504e-34 -2.09908e-38  3.84494e-43]
Giá trị cần tính tại 1938.0  là: -1.1544109413114617e+41
```

Từ đồ thị trên rõ ràng thấy nó không đi qua các mốc nội suy và giá trị tại 1938 cũng rất lớn: -1.1544109413114617e+41
Đây là điều không mong muốn xảy ra. Lí do bởi vì đa thức tìm ra phức tạp cùng với đó là hệ số đa thức có bậc rất lớn (10^{49}) và có bậc rất nhỏ (10^{-43}), điều này làm cho gói $fx0$ tính toán không còn chính xác.
Để giải quyết điều này chúng ta thực 1 phép đổi biến: $t = x - x0$, $t0 = x0 - x0 = 0$ và thực hiện chương trình:

Đồ thị thu được:



Kết quả thu được:

```
Mời nhập giá trị cần tính: 1938
Các mốc nội suy:
[1880.0, 1885.0, 1890.0, 1895.0, 1900.0, 1905.0, 1910.0, 1915.0, 1920.0, 1925.0, 1930.0, 1935.0, 1940.0, 1945.0, 1950.0, 1955.0, 1960.0,
 1965.0, 1970.0, 1975.0, 1980.0, 1985.0, 1990.0, 1995.0, 2000.0, 2005.0, 2010.0, 2015.0, 2020.0]
Các mốc nội suy sau khi đổi biến:
[-58.0, -53.0, -48.0, -43.0, -38.0, -33.0, -28.0, -23.0, -18.0, -13.0, -8.0, -3.0, 2.0, 7.0, 12.0, 17.0, 22.0, 27.0, 32.0, 37.0, 42.0, 4
7.0, 52.0, 57.0, 62.0, 67.0, 72.0, 77.0, 82.0]
Hệ số đa thức nội suy Newton là: (bắt đầu từ hệ số tự do)
[-0.00129  0.07319 -0.00241 -0.00158  0.00005  0.00001 -0.      -0.       0.       0.      -0.       0.       0.      -0.      -0.
  0.       0.      -0.       0.       0.      -0.      -0.       0.       0.      -0.       0.       0.      -0.       0.     ]
Giá trị cần tính tại 0.0  là: -0.0012858466303425561
Nhập số lượng thêm mốc nội suy: 0
```

*__NOTE__: các giá trị 0, -0 ở hệ số đa thức chỉ gần bằng 0. Mở rộng số chữ số sau dấu phẩy sẽ rõ. Ví dụ kết quả 20 chữ số sau dấu phẩy:

```
Mời nhập giá trị cần tính: 1938
Các mốc nội suy:
[1880.0, 1885.0, 1890.0, 1895.0, 1900.0, 1905.0, 1910.0, 1915.0, 1920.0, 1925.0, 1930.0, 1935.0, 1940.0, 1945.0, 1950.0, 1955.0, 1960.0,
 1965.0, 1970.0, 1975.0, 1980.0, 1985.0, 1990.0, 1995.0, 2000.0, 2005.0, 2010.0, 2015.0, 2020.0]
Các mốc nội suy sau khi đổi biến:
[-58.0, -53.0, -48.0, -43.0, -38.0, -33.0, -28.0, -23.0, -18.0, -13.0, -8.0, -3.0, 2.0, 7.0, 12.0, 17.0, 22.0, 27.0, 32.0, 37.0, 42.0, 4
7.0, 52.0, 57.0, 62.0, 67.0, 72.0, 77.0, 82.0]
Hệ số đa thức nội suy Newton là: (bắt đầu từ hệ số tự do)
[-0.0012858466303425561   0.07319279379195456    -0.002412308857458938   -0.0015779431030000454   0.00005234336708277293  0.000011908873
39906292 -0.00000065401891873114 -0.000000359650194162   0.00000000360712308534  0.00000000001700950403 -0.00000000001015409001  0.0000
0000000016717721  0.00000000000001574135 -0.00000000000000048066 -0.00000000000000001316  0.00000000000000000063  0.
 -0.                0.                0.                0.                -0.                -0.                0.
         0.                -0.                0.                0.                -0.                0.
       ]
Giá trị cần tính tại 0.0  là: -0.0012858466303425561
Nhập số lượng thêm mốc nội suy: 0
PS C:\Users\Admin\.vscode>
```

Với phép đổi biến này thì chương trình chúng ta đã chạy đúng, đồ thị đã đi qua các mốc nội suy và giá trị ước tính tại 1938 là: -0.0012858466303425561.
1 điều thú vị là theo earthobservatory[3] nhiệt độ trung bình Trái Đất năm 1938 gần bằng 0. Điều đó chứng tỏ kết quả chúng ta thu được khá sát với thực tế.

Từ ví dụ trên có thể thấy, phép đổi biến có thể giúp số liệu tính toán không bị quá lớn mà tính đúng đắn của phương pháp không thay đổi. Cùng với việc các mốc cách đều nhau $\rightarrow x_{i+1} - x_i = h \ \forall \ i \in [0, n-1]$ thì liệu có giúp ích gì trong việc tìm đa thức nội suy hay không, chúng ta đến với phần tiếp theo của báo cáo.

3 Đa thức nội suy Newton với mốc cách đều

3.1 Ý tưởng phương pháp

Trong trường hợp các mốc cách đều nhau 1 khoảng bằng h. Khi đó trên Bảng tỷ hiệu, với mỗi cột các phần tử sẽ có mẫu số giống nhau. Nhưng khi tính toán chúng ta chỉ cần dùng giá trị đầu bảng (Newton tiến) hoặc cuối bảng (Newton lùi) để tìm đa thức nội suy Newton. Do vậy, ta có thể giảm số lượng tính toán bằng cách lược bớt phép chia, và thực hiện phép chia sau cùng.
Trường hợp các mốc cách đều: $x_k = x_0 + kh = x_n - (n-k)h$
Với $h = x_{i+1} - x_i \ \forall i = \overline{0, n-1}$. h được gọi là bước lưới.
Xuất phát từ x_0 đến x_k là bước tiến k bước.
Xuất phát từ x_k đến x_0 là bước lùi k bước.
Khi lược bớt phép chia ở công thức tỷ hiệu ta có định nghĩa sai phân.

[3] https://earthobservatory.nasa.gov/world-of-change/global-temperatures

3.2 Sai phân

3.2.1 Định nghĩa

Sai phân tiến cấp 1 xuất phát từ x_k: $\Delta y_k = y_{k+1} - y_k$
Sai phân lùi cấp 1 xuất phát từ x_k: $\nabla y_k = y_k - y_{k-1}$
Sai phân tiến cấp 2 xuất phát từ x_k:

$\Delta^2 y_k = \Delta(\Delta y_k) = \Delta y_{k+1} - \Delta y_k = (y_{k+2} - y_{k+1}) - (y_{k+1} - y_k) = y_{k+2} - 2y_{k+1} + y_k$

Sai phân tiến lùi 2 xuất phát từ x_k:

$$\nabla^2 y_k = \nabla(\nabla y_k) = \nabla y_k - \nabla y_{k-1} = y_k - 2y_{k-1} + y_{k-2} = \Delta^2 y_{k-2}$$

Sai phân tiến cấp s xuất phát từ x_k:

$$\Delta^s y_k = \Delta(\Delta^{s-1} y_k)$$

Sai phân lùi cấp s xuất phát từ x_k:

$$\nabla^s y_k = \nabla(\nabla^{s-1} y_k) = \Delta^s y_{k-s}$$

Từ định nghĩa ta xây dựng bảng sai phân tiến và bảng sai phân lùi:
Bảng sai phân tiến:

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$	...	$\Delta^n y$
x_0	y_0				...	
x_1	y_1	Δy_0			...	
x_2	y_2	Δy_1	$\Delta^2 y_0$		...	
x_3	y_3	Δy_2	$\Delta^2 y_1$	$\Delta^3 y_0$	...	
...	...	...	...	...	...	...
x_n	y_n	Δy_{n-1}	$\Delta^2 y_{n-2}$	$\Delta^3 y_{n-3}$	...	$\Delta^n y_0$

Bảng sai phân lùi:

x	y	∇y	$\nabla^2 y$	$\nabla^3 y$	...	$\nabla^{n-1} y$	$\nabla^n y$
x_0	y_0	∇y_1	$\nabla^2 y_2$	$\nabla^3 y_3$	...	$\nabla^{n-1} y_{n-1}$	$\nabla^n y_n$
x_1	y_1	∇y_2	$\nabla^2 y_3$	$\nabla^3 y_4$	...	$\nabla^{n-1} y_n$	
x_2	y_2	∇y_3	$\nabla^2 y_4$	...	...		
x_3	y_3	∇y_4	...	$\nabla^3 y_n$			
...	...	...	$\nabla^2 y_n$				
x_{n-1}	y_{n-1}	∇y_n					
x_n	y_n						

Do sai phân tiến, lùi có thể biểu diễn qua lại với nhau. Nên các tính chất của sai phân sử dụng sai phân tiến để thể hiện.

3.2.2 Tính chất của sai phân

• Tính chất 1: Liên hệ giữa sai phân và tỷ sai phân:

$$f[x_0, ..., x_k] = \frac{\Delta^k y_0}{h^k k!} = \frac{\nabla^k y_k}{h^k k!}$$

Chứng minh:

$$f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{\Delta y_0}{h}$$
$$f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} = \frac{\Delta y_1 - \Delta y_0}{2h^2} = \frac{1}{2!h^2}\Delta^2 y_0$$
$$...$$
$$\rightarrow f[x_0, x_1, ..., x_k] = \frac{1}{k!h^2}\Delta^k y_0 = \frac{1}{h^k k!}\nabla^k y_k (dpcm)$$

• Tính chất 2: Δ là toán tử tuyến tính

$$\Delta(\alpha y + \beta z) = \alpha \Delta y + \beta \Delta z$$

$$\Delta C = 0, C - const$$
$$\Delta^n(x^n) = n!h^n$$
$$\Delta^m(x^n) = 0, m > n$$
$$\Delta^m(\Delta^k f) = \Delta^{k+m} f$$
$$\Delta^0 f = f$$

Các công thức này dễ dàng được suy ra từ định nghĩa • Tính chất 3: Giá trị của hàm $f(x)$ được biểu diễn qua sai phân các cấp của nó

$$y_m = f(x_m) = f(x + mh) = \sum_{k=0}^{m} C_m^k \Delta^k f(x)$$

Trong đó: $C_m^k = \frac{m!}{k!(m-k)!}$
Chứng minh:

$$\Delta f(x) = f(x + h) - f(x) \rightarrow f(x + h) = (1 + \Delta) f(x)$$
$$f(x + 2h) = f(x + h + h) = (1 + \Delta) f(x + h) = (1 + \Delta)^2 f(x)$$
$$...$$
$$f(x + mh) = (1 + \Delta)^m f(x)$$

Theo Khai triển nhị thức Newton ta được:

$$f(x+mh) = (1+\Delta)^m f(x) = \sum_{k=0}^{m} C_m^k \Delta^k f(x)(dpcm)$$

• Tính chất 4: Sai phân cấp n của $f(x)$ được biểu diễn qua các giá trị liên tiếp của nó

$$\Delta^n f(x) = \sum_{i=0}^{n} (-1)^i C_n^i f(x+(n-i)h)$$

Chứng minh:

$$\Delta^n f(x) = [(1+\Delta)-1]^n f(x) = \sum_{i=0}^{n} (-1)^n C_n^i (1+\Delta)^{n-i} f(x)$$

Từ tính chất số 3 của sai phân:

$$\rightarrow \Delta^n f(x) = \sum_{i=0}^{n} (-1)^i C_n^i f(x+(n-i)h)(dpcm)$$

• Tính chất 5: Giá trị $f(x)$ có đạo hàm liên tục đến cấp n trên đoạn $[x, x+nh]$ thì ta có

$$\Delta^n f(x) = h^n f^{(n)}(x+\theta nh) \qquad 0<\theta<1$$

Chứng minh:

– Khi $n=1$: $\Delta f(x) = f(x+h) - f(x) = hf^{'}(x+h\theta^{'})$ (CT số gia giới nội)
– Giả sử đúng với $n=k$:

$$\Delta^k f(x) = h^k f^{(k)}(x+\theta^{'}kh) \qquad 0<\theta^{'}<1$$

– Ta cần chứng minh đúng với $n=k+1$:

$$\Delta^{k+1} f(x) = \Delta(\Delta^k f(x)) = \Delta(h^k f^{(k)}(x+\theta^{'}kh))$$
$$= h^k[f^{(k)}(x+kh\theta^{'}+h) - f^{(k)}(x+kh\theta^{'})]$$

Áp dụng công thức số gia giới nội:

$$\Delta^{k+1} f(x) = h^k h f^{(k+1)}(x+kh\theta^{'}+\theta^{''}h) \qquad 0<\theta^{''}<1$$

Đặt: $\frac{\theta^{'}k+\theta^{''}}{k+1} = \theta$

$$\rightarrow \Delta^{k+1} f(x) = h^{k+1} f^{(k+1)}(x+\theta(k+1)h)(dpcm)$$

3.3 Đa thức nội suy Newton với mốc cách đều

• Đa thức nội suy Newton tiến cách đều:
Với mốc bất kì ta có:

$$P_n(x) = f(x_0) + (x - x_0)f[x_0, x_1] + (x - x_0)(x - x_1)f[x_0, x_1, x_2] + ...+$$
$$(x - x_0)(x - x_1)...(x - x_{n-1})f[x_0, ..., x_n]$$

Đặt: $x = x_0 + th$
Từ tính chất 1 của sai phân ta có:

$$f[x_0, ..., x_k] = \frac{\Delta^k y_0}{h^k k!}$$

Và

$$x - x_0 = th$$
$$x - x_1 = x - x_0 + x_0 - x_1 = th - h = h(t - 1)$$
$$...$$
$$x - x_{k-1} = x - x_0 + x_0 - x_{k-1} = th - (k - 1)h = h(t - k + 1)$$
$$\rightarrow (x - x_0)(x - x_1)...(x - x_{k-1}) = h^k t(t - 1)(t - 2)...(t - k + 1)$$
$$\rightarrow f[x_0, ..., x_k](x - x_0)(x - x_1)...(x - x_{k-1}) = \frac{\Delta^k y_0}{k!}t(t - 1)(t - 2)...(t - k + 1)$$
$$\rightarrow P_n(t) = y_0 + \frac{\Delta y_0}{1!}t + \frac{\Delta^2 y_0}{2!}t(t - 1) + ... + \frac{\Delta^n y_0}{n!}t(t - 1)...(t - n + 1)$$

Đa thức trên chính là đa thức nội suy Newton tiến với mốc cách đều.
• Đa thức nội suy Newton lùi cách đều:
Với mốc bất kì ta có:

$$P_n(x) = f(x_n) + (x - x_n)f[x_n, x_n] + (x - x_n)(x - x_{n-1})f[x_n, x_{n-1}, x_{n-2}] + ...+$$
$$(x - x_n)(x - x_{n-1})...(x - x_1)f[x_n, x_{n-1}, ..., x_0]$$

Đặt: $x = x_n + th$
Từ tính chất 1 của sai phân ta có:

$$f[x_0, ..., x_k] = \frac{\nabla^k y_k}{h^k k!}$$

Và:

$$x - x_k = x - x_n + x_n - x_k = th + (n - k)h = h(t + n - k)$$
$$x - x_{k-1} = x - x_n + x_n - x_{k-1} = th + (n - k + 1)h = h(t + n - k + 1)$$

$$...$$

$$x - x_1 = x - x_n + x_n - x_1 = th + (n-1)h = h(t+n-1)$$

$$\rightarrow (x-x_k)(x-x_{k-1})...(x-x_1) = h^k(t+n-k)(t+n-k+1)...(t+n-1)$$

$$\rightarrow f[x_0,...,x_k](x-x_k)(x-x_{k-1})...(x-x_1) = \frac{\nabla^k y_k}{k!}(t+n-k)(t+n-k+1)...(t+n-1)$$

$$\rightarrow P_n(t) = y_n + \frac{\nabla y_n}{1!}t + \frac{\nabla^2 y_n}{2!}t(t+1) + ... + \frac{\nabla^n y_n}{n!}t(t+1)...(t+n-1)$$

Đa thức trên chính là đa thức nội Newton lùi với mốc cách đều.

• Sai số:

Công thức đánh giá sai số:

$$R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(x)$$

Đa thức nội suy Newton tiến.Từ $x = x_0 + th$ ta được:

$$\omega_{n+1}(x) = \prod_{k=0}^{n}(x - x_k) = \prod_{k=0}^{n}(t-k)$$

Và do

$$f^{(n+1)}(x) = \lim_{h\to 0}\frac{\Delta^{n+1}y_0}{h^{n+1}}$$

nên nếu xem:

$$f^{(n+1)}(\xi) \approx \frac{\Delta^{n+1}y_0}{h^{n+1}}$$

Thì:

$$R_n(x) \approx \frac{\Delta^{n+1}y_0}{(n+1)!}\prod_{k=0}^{n}(t-k)$$

• Nhận xét:

- Với mốc cách đều và với phép đổi biến ta đã giảm bớt được 1 lượng phép toán.
- Nếu thêm mốc nội suy (x_{n+1}, y_{n+1}) đa thức $P_{n+1}(x)$ được tính như sau:
  + Đối với Newton tiến:

$$P_{n+1}(t) = P_n(t) + \frac{\Delta^{n+1}y_0}{(n+1)!}t(t-1)(t-2)...(t-n)$$

  + Đối với Newton lùi:

$$P_{n+1}(t) = P_n(t) + \frac{\nabla^{n+1}y_{n+1}}{(n+1)!}t(t+1)(t+2)...(t+n)$$

3.4 Thuật toán và ví dụ

3.4.1 Thuật toán

Thuật toán: Nội suy đa thức Newton tiến mốc cách đều

Các gói sử dụng:

1. *main*: thuật toán chính
2. *Input*: nhập x, y từ file và kiểm tra điều kiện cách đều
3. *pickPoint*: chọn ra *num* điểm gần $x0$ nhất 4. *BuildBSP*: kết nạp mốc $x[i]$ vào bảng sai phân tiến
5. *BuildBTT*: kết nạp mốc $x[i]$ vào bảng tính tích theo lược đồ Hoocner
6. fx: kết nạp mốc $x[i]$ vào đa thức f
7. *Newton_interpolation_Forward*: xây dựng đa thức nội suy Newton tiến cách đều
8. $fx0$: ước tính giá trị tại $x0$ theo lược đồ Hoocner
9. *add_Newton* : thêm mốc nội suy
10. *plot*: vẽ đồ thị

main

Input: flie các mốc nội suy, điểm cần ước lượng giá trị: $x0$, số lượng thêm mốc: k, số lượng mốc nội suy: num
Output: hệ số đa thức f, giá trị ước lượng tại $x0$: $value$, đồ thị đa thức f

begin
$x, y \longleftarrow Input()$
$h \longleftarrow x[1] - x[0]$
$x1, y1 \longleftarrow pickPoint(x0, num, x, y)$
In các mốc nội suy $x1$
$n \longleftarrow \text{len}(x1) - 1$
Khởi tạo các ma trận không: $BSP(n+1), BTT(n+1), f(n+1)$
$BSP[0], BTT[0], f[0] \longleftarrow y1[0], 1, y1[0]$
$f, BSP, BTT \longleftarrow Newton\_interpolation\_Forward(BSP, BTT, f, x1, y1, n)$
$value \longleftarrow fx0(f, x0, x1)$
In hệ số đa thức f và giá trị $value$
$plot(x1, y1, f)$
if $k <= 0$ **then**: Dừng chương trình
end if
for $i = 1 \rightarrow k$ **do**
Nhập mốc và giá trị nội suy $\{xt, yt\}$
if xt **not in** $x1$ **and** $\text{abs}(xt - x1[n] - h) < 1e-6$ **then**
$n \longleftarrow n + 1$
append $\{xt, yt\}$ vào $\{x1, y1\}$
$f, BSP, BTT \longleftarrow add\_Newton(BSP, BTT, f, x1, y1, n)$
end if
end for
In hệ số đa thức f và $value$
$plot(x1, y1, f)$
end

Gói **pickPoint**

Input: $x0, num, x, y$
Output: $x1, y1$

begin
    if $num >$ len(x) **then**
        Báo lỗi và dừng chương trình
    end if
    $i \longleftarrow$ int$(\frac{x0-x[0]}{x[1]-x[0]})$
    $left \longleftarrow$ min(len(x) $n - num$, max$(0, i + 1 -$ int$(num/2)))$
    $right \longleftarrow left + num - 1$
    Khởi tạo list $x1, y1$
    $x1 \longleftarrow x[left : right + 1]$
    $y1 \longleftarrow y[left : right + 1]$
    return $x1, y1$
end

Gói **BuildBSP**

Input: BSP, x, y, i
Output: SP_row_i

begin
    Khởi tạo ma trận không SP_row_i(len(x))
    $SP\_row\_i[0] \longleftarrow y[i]$
    for $j = 1 \rightarrow i$ **do**
        $SP\_row\_i[j] \longleftarrow SP\_row\_i[j-1] - BSP[j-1]$
    end for
    return SP_row_i
end

Gói **BuildBTT**

Input: BTT, x, i
Output: TT_row_i

begin
    Khởi tạo ma trận không $TT\_row\_i(\text{len}(x))$
    $TT\_row\_i[i] \longleftarrow 1$
    for $j = i \to 1$ **do**
        $TT\_row\_i[j] \longleftarrow BTT[j-1] - i * BTT[j]$
    end for
    if $i! = 0$ **then**
        $TT\_row\_i[0] \longleftarrow -i * BTT[0]$
    return TT_row_i
end

Gói **fx**

Input: BSP, BTT, i, f
Output: f

begin
    for $j = 1 \to i$ **do**
        $f[j] \longleftarrow f[j] + \frac{BSP[i]*BTT[j-1]}{i!}$
    end for
    return f
end

Gói **Newton_interpolation_Forward**

Input: $BSP, BTT, f, x1, y1, n$
Output: f, BSP, BTT

begin
  for $i = 1 \to n$ **do**
    $BSP \longleftarrow BuildBSP(BSP, x1, y1, i)$
    $BTT \longleftarrow BuildBTT(BTH, x1, i-1)$
    $f \longleftarrow fx(BSP, BTT, i, f)$
  end for
  return f, BSP, BTT
end

Gói **fx0**

Input: $f, x0, x1$
Output: $value$

begin
  $t \longleftarrow \frac{x0 - x1[0]}{x1[1] - x1[0]}$
  $value \longleftarrow f[\text{len}(x1) - 1]$
  for $i = \text{len}(x1) - 2 \to 0$ **do**
    $value \longleftarrow value * t + f[j]$
  end for
  return $value$
end

Gói **add_Newton**

Input: $BSP, BTT, f, x1, y1, n$
Output: f, BSP, BTT

begin
$BSP \longleftarrow BuildBSP(BSP, x1, y1, n)$
$BTT \longleftarrow BuildBTT(BTT, x1, n-1)$
Tạo thêm 1 cột cho f
$f \longleftarrow fx(BSP, BTT, n, f)$
return f, BSP, BTT
end

- Khối lượng tính toán:
Gói $BuildBSP$: i
Gói $BuildBTT$: $2i + 2$
Gói fx: $3i$
Gói $Newton\_interpolation\_Forward$:
$\sum_1^n (i + 2i + 3i) - 2 = \sum_1^n (6i) = 6 + 12 + 18 + ... + 6n - 2$
$= \frac{(6+6n)n}{2} - 2 = 3n^2 + 3n - 2$

Tổng quát: $O(3n^2)$

Thuật toán: Nội suy đa thức Newton lùi mốc cách đều
Các gói sử dụng:
1. *main*: thuật toán chính
2. *Input*: nhập x, y từ file và kiểm tra điều kiện cách đều
3. *pickPoint*: chọn ra num điểm gần $x0$ nhất 4. $BuildBSP$: kết nạp mốc $x[i]$ vào bảng sai phân lùi
5. $BuildBTT$: kết nạp mốc $x[i]$ vào bảng tính tích theo lược đồ Hoocner
6. fx: kết nạp mốc $x[i]$ vào đa thức f
7. $Newton\_interpolation\_Backward$: xây dựng đa thức nội suy Newton lùi cách đều
8. $fx0$: ước tính giá trị tại $x0$ theo lược đồ Hoocner
9. add_Newton : thêm mốc nội suy
10. *plot*: vẽ đồ thị
*__NOTE__: Các gói 2, 3, 6, 10 giống với Nội suy đa thức Newton tiến cách đều

main

Input: flie các mốc nội suy, điểm cần ước lượng giá trị: $x0$, số lượng thêm mốc: k, số lượng mốc nội suy: num
Output: hệ số đa thức f, giá trị ước lượng tại $x0$: $value$, đồ thị đa thức f

begin
$x, y \longleftarrow Input()$
$h \longleftarrow x[1] - x[0]$
$x1, y1 \longleftarrow pickPoint(x0, num, x, y)$
In các mốc nội suy $x1$
$n \longleftarrow \text{len}(x1) - 1$
Khởi tạo các ma trận không: $BSP(n+1), BTT(n+1), f(n+1)$
$BSP[0], BTT[0], f[0] \longleftarrow y1[n], 1, y1[n]$
$f, BSP, BTT \longleftarrow Newton\_interpolation\_Backward(BSP, BTT, f, x1, y1, n)$
$value \longleftarrow fx0(f, x0, x1)$
In hệ số đa thức f và giá trị $value$
$plot(x1, y1, f)$
if $k <= 0$ **then**: Dừng chương trình
end if
for $i = 1 \rightarrow k$ **do**
Nhập mốc và giá trị nội suy $\{xt, yt\}$
if xt **not in** $x1$ **and** $\text{abs}(x1[0] - xt - h) < 1e-6$ **then**
$n \longleftarrow n + 1$
insert $\{xt, yt\}$ vào vị trí đầu $\{x1, y1\}$
$f, BSP, BTT \longleftarrow add\_Newton(BSP, BTT, f, x1, y1, n)$
end if
end for
In hệ số đa thức f và $value$
$plot(x1, y1, f)$
end

Gói **BuildBSP**

Input: BSP, x, y, i
Output: SP_row_i

begin
    Khởi tạo ma trận không $SP\_row\_i(\text{len}(x))$
    $SP\_row\_i[0] \longleftarrow y[i]$
    for $j = 1 \to \text{len}(x) - i - 1$ **do**
        $SP\_row\_i[j] \longleftarrow BSP[j-1] - SP\_row\_i[j-1]$
    end for
    return SP_row_i
end

Gói **BuildBTT**

Input: BTT, x, i
Output: TT_row_i

begin
    Khởi tạo ma trận không $TT\_row\_i(\text{len}(x))$
    $TT\_row\_i[-i] \longleftarrow 1$
    for $j = -i \to 1$ **do**
        $TT\_row\_i[j] \longleftarrow BTT[j-1] - i * BTT[j]$
    end for
    if $i! = 0$ **then**
        $TT\_row\_i[0] \longleftarrow -i * BTT[0]$
    return TT_row_i
end

Gói **Newton_interpolation_Backward**

Input: $BSP, BTT, f, x1, y1, n$
Output: f, BSP, BTT

begin
  for $i = 1 \to n$ **do**
    $BSP \longleftarrow BuildBSP(BSP, x1, y1, n - i)$
    $BTT \longleftarrow BuildBTT(BTH, x1, 1 - i)$
    $f \longleftarrow fx(BSP, BTT, i, f)$
  end for
  return f, BSP, BTT
end

Gói **fx0**

Input: $f, x0, x1$
Output: $value$

begin
  $t \longleftarrow \frac{x0 - x1[len(x1)-1]}{x1[1]-x1[0]}$
  $value \longleftarrow f[\text{len}(x1) - 1]$
  for $i = \text{len}(x1) - 2 \to 0$ **do**
    $value \longleftarrow value * t + f[j]$
  end for
  return $value$
end

Gói **add_Newton**

Input: $BSP, BTT, f, x1, y1, n$
Output: f, BSP, BTT

begin
$BSP \longleftarrow BuildBSP(BSP, x1, y1, 0)$
$BTT \longleftarrow BuildBTT(BTT, x1, 1-n)$
Tạo thêm 1 cột cho f
$f \longleftarrow fx(BSP, BTT, n, f)$
return f, BSP, BTT
end

- Khối lượng tính toán:
Gói $BuildBSP$: $n - i$
Gói $BuildBTT$: $2i + 2$
Gói fx: $3i$
Gói $Newton\_interpolation\_Backward$:
$\sum_1^n (i + 2i + 3i) - 2 = \sum_1^n 6i - 2 = 3n^3 + 3n - 2$

Tổng quát: $O(3n^2)$

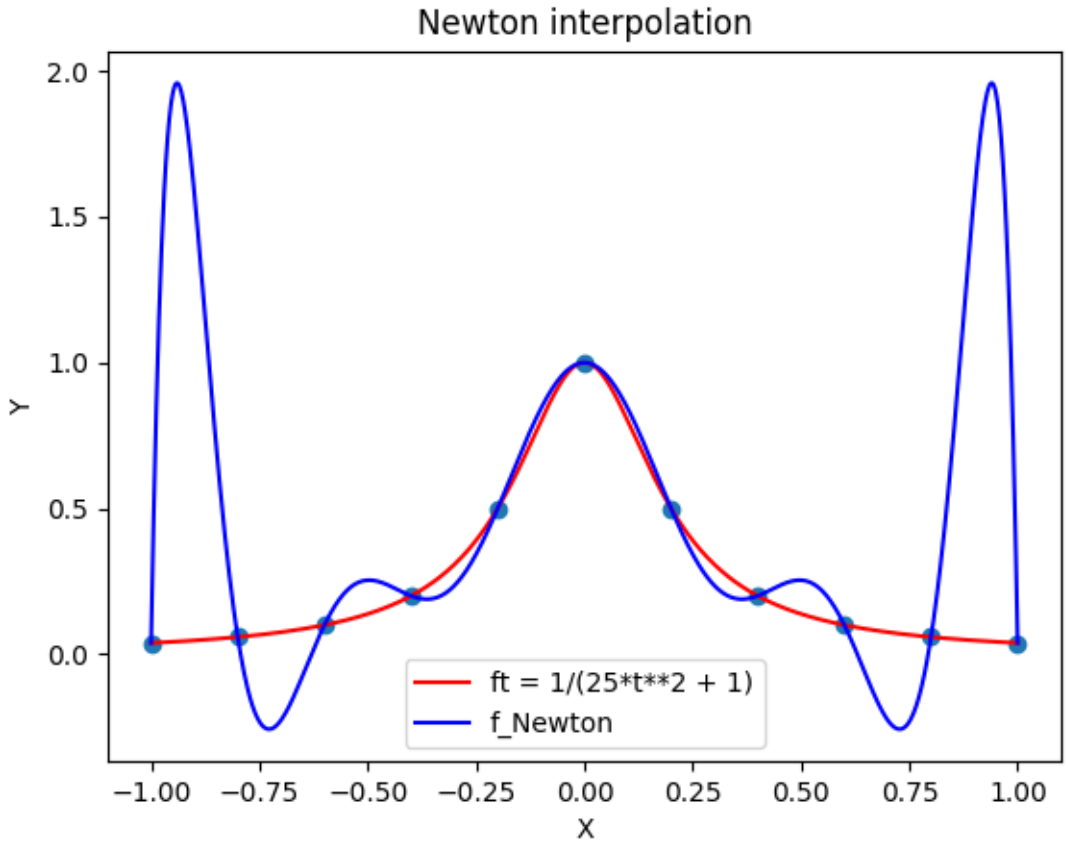
3.4.2 Ví dụ

Ví dụ 1:
Xét bộ dữ liệu 11 mốc được sinh từ hàm $f(x) = \frac{1}{25x^2+1}$

x	y
-1.0	0.0385
-0.8	0.0588
-0.6	0.1000
-0.4	0.2000
-0.2	0.5000
0.0	1
0.2	0.5000
0.4	0.2000
0.6	0.100

0.8	0.0588
1.0	0.0385

Chạy chương trình sử dụng hết 11 mốc, ước tính tại $x0 = 0.75$
Đồ thị thu được:



Kết quả Newton tiến thu được:

```
PS C:\Users\Admin\.vscode> & C:/Users/Admin/AppData/Local/Programs/Python/Python39/python.exe c:/Users/Admin/.vscode/Newton_Interpolatio
n_Forward.py
Mời nhập giá trị cần tính: 0.75
Chọn số lượng mốc nội suy tính (<= 11): 11
Các mốc nội suy:
[-1.0, -0.8, -0.6, -0.4, -0.2, 0.0, 0.2, 0.4, 0.6, 0.8, 1.0]
Hệ số của đa thức nội suy Newton: (bắt đầu từ hệ số tự do)
[  0.0385   15.80279 -43.32399  47.46596 -27.76333   9.68024  -2.10698   0.28869  -0.02419   0.00113  -0.00002]
t = (x - -1.0)/0.2
Giá trị cần tính tại 0.75  là: -0.23147540676853232
Nhập số lượng thêm mốc nội suy: 0
PS C:\Users\Admin\.vscode>
```

Kết quả Newton lùi thu được:

```
PS C:\Users\Admin\.vscode> & C:/Users/Admin/AppData/Local/Programs/Python/Python39/python.exe c:/Users/Admin/.vscode/Newton_interpolatio
n_Backward.py
Mời nhập giá trị cần tính: 0.75
Chọn số lượng mốc nội suy tính (<= 11): 11
Các mốc nội suy:
[-1.0, -0.8, -0.6, -0.4, -0.2, 0.0, 0.2, 0.4, 0.6, 0.8, 1.0]
Hệ số của đa thức nội suy Newton: (bắt đầu từ hệ số tự do)
[  0.0385  -15.80279 -43.32399 -47.46596 -27.76333  -9.68024  -2.10698  -0.28869  -0.02419  -0.00113  -0.00002]
t = (x - 1.0)/-1.8
Giá trị cần tính tại 0.75  là: -0.2314754068143712
Nhập số lượng thêm mốc nội suy: 0
PS C:\Users\Admin\.vscode>
```

Ở đây, do phép đổi biến khác nhau nên 2 đa thức Newton tiến và lùi ra khác nhau nhưng khi đổi lại về ẩn x thì sẽ giống nhau.
Giá trị tại 0.75 theo Newton tiến: -0.23147540676853232
Giá trị tại 0.75 theo Newton lùi: -0.2314754068143712
Có sự sai lệch kết quả giữa Newton tiến và lùi ở đây là do sai số tính toán (độ lệch $\approx 10^{-11}$)
Do sử dụng số lượng mốc không đủ nhiều nên giá trị ước tính ở đây có sai số tương đối lớn ($f(0.75) \approx 0.0664$). Tuy nhiên, dựa vào đồ thị, cũng như mốc bất kì, đoạn trung tâm được xấp xỉ khá tốt.

Ví dụ 2
Giờ ta sẽ kiểm tra gói thêm mốc nội suy. Chọn bộ dữ liệu sinh ra từ đa thức $f(x) = 3x^3 + x^2 + 2$. Đa thức trên cần 4 mốc để tìm ra. Nhưng ban đầu chỉ có 2 mốc:

x	y
1	6
2	30

Cần thêm 2 mốc 3 và 4 đối với Newton tiến, -1 và 0 đối với Newton lùi:

x	y
3	92
4	210

x	y
-1	0
0	2

Chạy chương trình tính tại $x0 = 10$ ($f(10) = 3102$). Kết quả thu được:

Newton tiến:

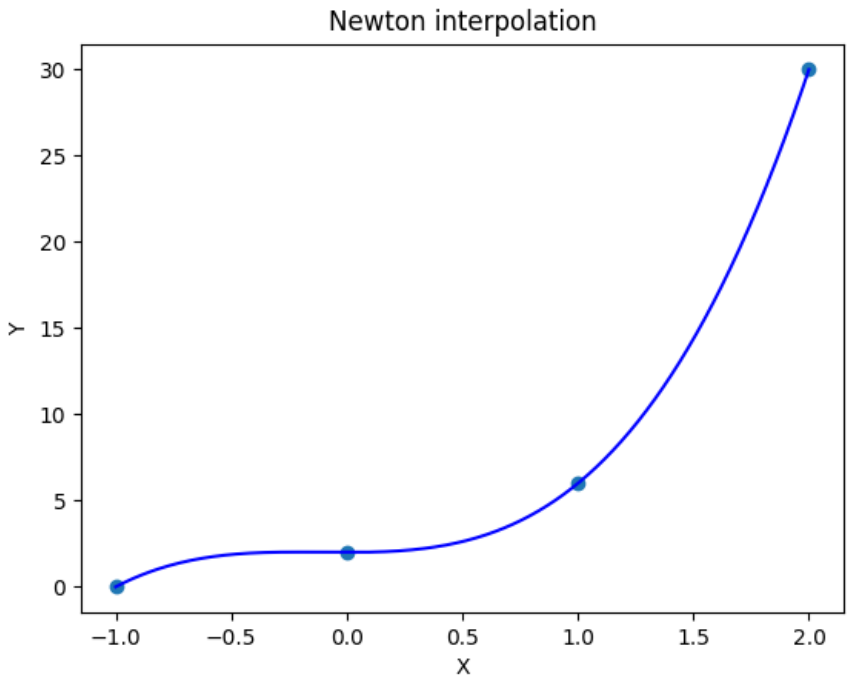
```
PS C:\Users\Admin\.vscode> & C:/Users/Admin/AppData/Local/Programs/Python/Python39/python.exe c:/Users/Admin/.vscode/Newton_Interpolatio
n_Forward.py
Mời nhập giá trị cần tính: 10
Chọn số lượng mốc nội suy tính (<= 2): 2
Các mốc nội suy:
[1.0, 2.0]
Hệ số của đa thức nội suy Newton: (bắt đầu từ hệ số tự do)
[ 6. 24.]
t = (x - 1.0)/1.0
Giá trị cần tính tại 10.0  là: 222.0
Nhập số lượng thêm mốc nội suy: 2
Nhập mốc và giá trị nội suy: 3 92
Nhập mốc và giá trị nội suy: 4 210
Hệ số của đa thức nội suy Newton: (bắt đầu từ hệ số tự do)
[ 6. 11. 10.  3.]
t = (x - 1.0)/1.0
Giá trị cần tính tại 10.0  là: 3102.0
PS C:\Users\Admin\.vscode>
```

Tại $x0 = 10$, kết quả ra đúng 3102
Đa thức theo ẩn t thu được: $P(t) = 6 + 11t + 10t^2 + 3t^3$
Đổi về ẩn x:
$P(x) = 6 + 11\frac{x-1}{1} + 10\frac{(x-1)^2}{1} + 3\frac{(x-1)^3}{1} = 3x^3 + x^2 + 2$
Newton lùi:

```
PS C:\Users\Admin\.vscode> & C:/Users/Admin/AppData/Local/Programs/Python/Python39/python.exe c:/Users/Admin/.vscode/Newton_interpolatio
n_Backward.py
Mời nhập giá trị cần tính: 10
Chọn số lượng mốc nội suy tính (<= 2): 2
Các mốc nội suy:
[1.0, 2.0]
Hệ số của đa thức nội suy Newton: (bắt đầu từ hệ số tự do)
[30. 24.]
t = (x - 2.0)/1.0
Giá trị cần tính tại 10.0  là: 222.0
Nhập số lượng thêm mốc nội suy: 2
Nhập mốc và giá trị nội suy: 0 2
Nhập mốc và giá trị nội suy: -1 0
Hệ số của đa thức nội suy Newton: (bắt đầu từ hệ số tự do)
[30. 40. 19.  3.]
t = (x - 2.0)/1.0
Giá trị cần tính tại 10.0  là: 3102.0
PS C:\Users\Admin\.vscode>
```

Tại $x0 = 10$, kết quả ra đúng 3102
Đa thức theo ẩn t thu được: $P(t) = 30 + 40t + 19t^2 + 3t^3$
Đổi về ẩn x:
$P(x) = 30 + 40\frac{x-2}{1} + 19\frac{(x-2)^2}{1} + 3\frac{(x-2)^3}{1} = 3x^3 + x^2 + 2$
Như vậy, cả 2 gói thêm mốc nội suy của cả Newton tiến và lùi đều ra đúng kết quả.

Đồ thị của $f(x)$ thu được:



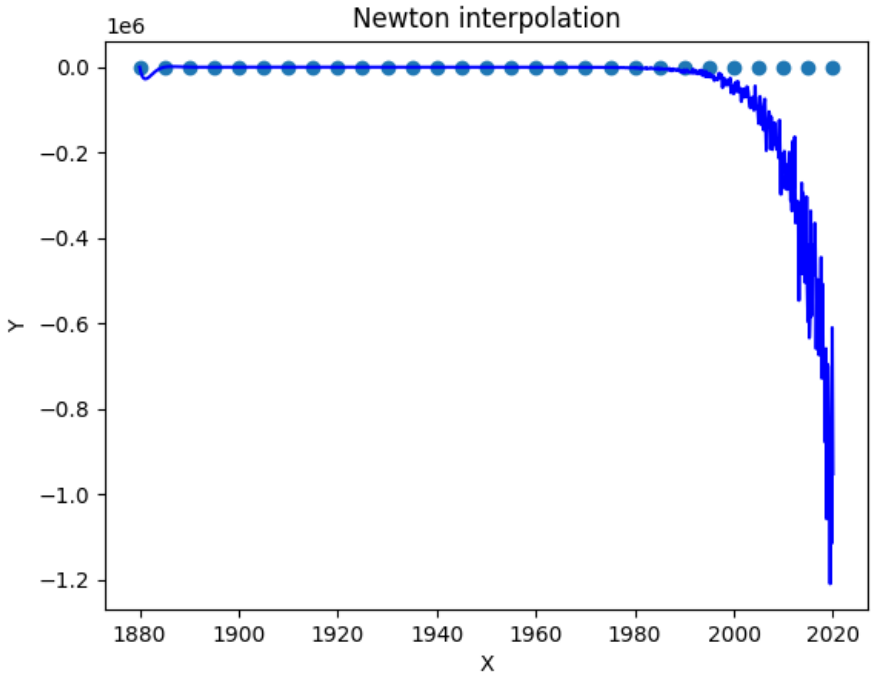
*__NOTE__ khi thêm mốc nội suy cần đảm bảo điều kiện cách đều (cách đều x_n đối với Newton tiến, cách đều x_0 đối với Newton lùi), nếu không mốc đó sẽ không được kết nạp thêm. Ví dụ:

```
PS C:\Users\Admin\.vscode> & C:/Users/Admin/AppData/Local/Programs/Python/Python39/python.exe c:/Users/Admin/.vscode/Newton_Interpolatio
n_Forward.py
Mời nhập giá trị cần tính: 10
Chọn số lượng mốc nội suy tính (<= 2): 2
Các mốc nội suy:
[1.0, 2.0]
Hệ số của đa thức nội suy Newton: (bắt đầu từ hệ số tự do)
[ 6. 24.]
t = (x - 1.0)/1.0
Giá trị cần tính tại 10.0  là: 222.0
Nhập số lượng thêm mốc nội suy: 5
Nhập mốc và giá trị nội suy: 3 92
Nhập mốc và giá trị nội suy: 0 2
Nhập mốc và giá trị nội suy: -5 -348
Nhập mốc và giá trị nội suy: 4 210
Nhập mốc và giá trị nội suy: -4 -174
Hệ số của đa thức nội suy Newton: (bắt đầu từ hệ số tự do)
[ 6. 11. 10.  3.]
t = (x - 1.0)/1.0
Giá trị cần tính tại 10.0  là: 3102.0
PS C:\Users\Admin\.vscode>
```

Mặc dù thêm 5 mốc nhưng thực chất chỉ kết nạp thêm 2 mốc 3 và 4.

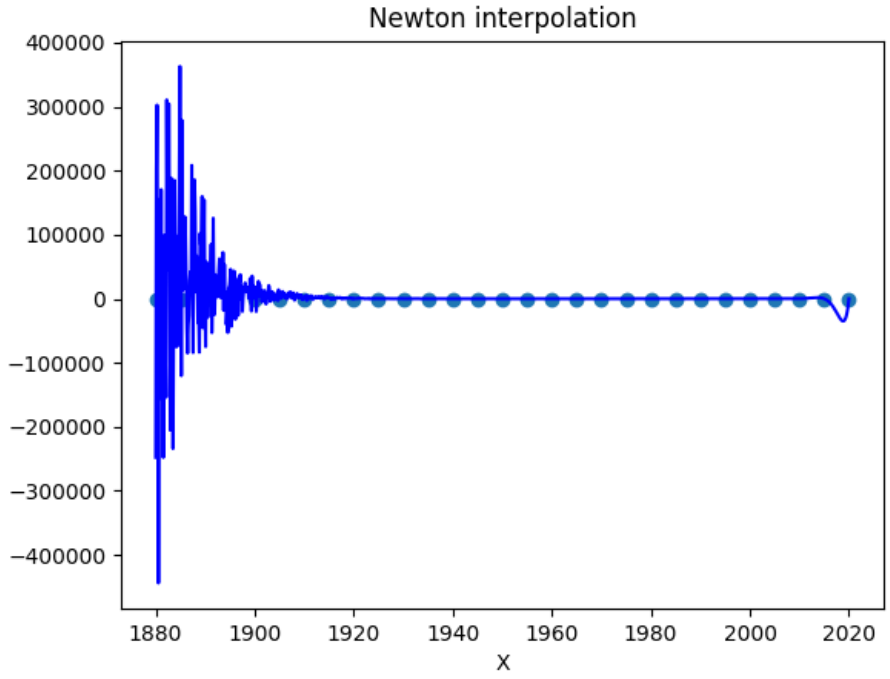
Ví dụ 4:
Chọn bộ dữ liệu Nhiệt độ Trái Đất (Ví dụ 5 trang 22).
Lần 1, ước tính tại 1938, sử dụng cả 29 mốc.
Newton tiến:

```
PS C:\Users\Admin\.vscode> & C:/Users/Admin/AppData/Local/Programs/Python/Python39/python.exe c:/Users/Admin/.vscode/Newton_Interpolatio
n_Forward.py
Mời nhập giá trị cần tính: 1938
Chọn số lượng mốc nội suy tính (<= 29): 29
Các mốc nội suy:
[1880.0, 1885.0, 1890.0, 1895.0, 1900.0, 1905.0, 1910.0, 1915.0, 1920.0, 1925.0, 1930.0, 1935.0, 1940.0, 1945.0, 1950.0, 1955.0, 1960.0,
 1965.0, 1970.0, 1975.0, 1980.0, 1985.0, 1990.0, 1995.0, 2000.0, 2005.0, 2010.0, 2015.0, 2020.0]
Hệ số của đa thức nội suy Newton: (bắt đầu từ hệ số tự do)
[      -0.166    -305834.9956   1178622.89303 -2025715.99797  2094289.70371 -1476322.37446   760749.87767  -299512.86349    92868.53125
  -23177.70457     4731.81881     -799.83184      112.95973      -13.41846        1.347         -0.1146         0.00827       -0.00051
      0.00003       -0.            0.            -0.             0.            -0.             0.            -0.             0.
     -0.            0.     ]
t = (x - 1880.0)/5.0
Giá trị cần tính tại 1938.0  là: -0.26878962708543985
Nhập số lượng thêm mốc nội suy: 0
PS C:\Users\Admin\.vscode>
```



Newton lùi:

```
PS C:\Users\Admin\.vscode> & C:/Users/Admin/AppData/Local/Programs/Python/Python39/python.exe c:/Users/Admin/.vscode/Newton_interpolatio
n_Backward.py
Mời nhập giá trị cần tính: 1938
Chọn số lượng mốc nội suy tính (<= 29): 29
Các mốc nội suy:
[1880.0, 1885.0, 1890.0, 1895.0, 1900.0, 1905.0, 1910.0, 1915.0, 1920.0, 1925.0, 1930.0, 1935.0, 1940.0, 1945.0, 1950.0, 1955.0, 1960.0,
 1965.0, 1970.0, 1975.0, 1980.0, 1985.0, 1990.0, 1995.0, 2000.0, 2005.0, 2010.0, 2015.0, 2020.0]
Hệ số của đa thức nội suy Newton: (bắt đầu từ hệ số tự do)
[      1.021    384438.9556  1472702.69056 2513391.25839 2578135.37176 1802048.59852  920324.38197  358990.54158  110257.26686   27253.5
0625    5510.20644     922.41584     129.02123      15.18053       1.50955       0.12724       0.0091        0.00055       0.00003
 0.            0.            0.            0.            0.            0.            0.            0.            0.            0.     ]
t = (x - 2020.0)/5.0
Giá trị cần tính tại 1938.0  là: 34.83267628176044
Nhập số lượng thêm mốc nội suy: 0
PS C:\Users\Admin\.vscode>
```



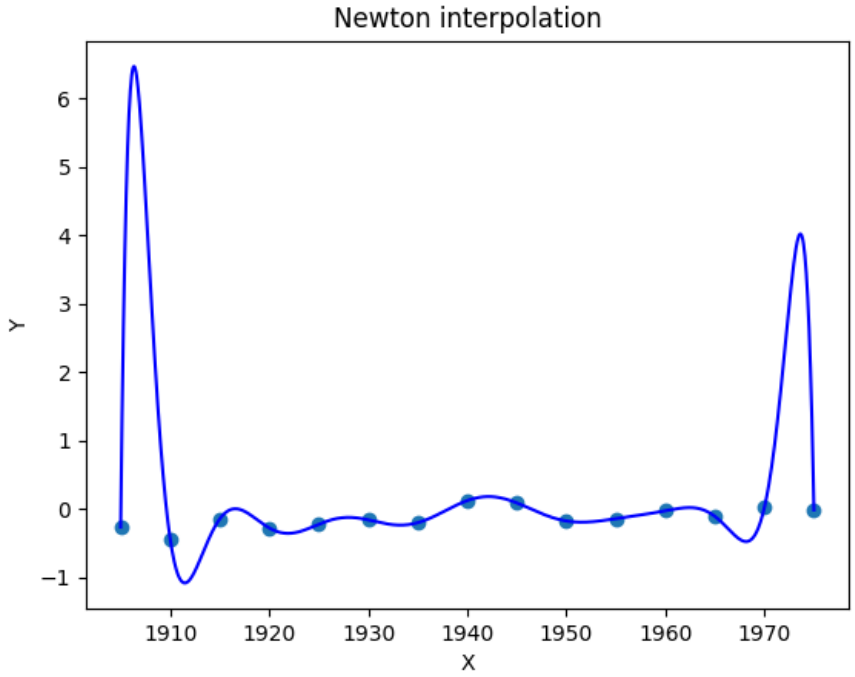
Qua kết quả, cả 2 đồ thị Newton tiến và lùi đều không đi qua các mốc nội suy (Lỗi này đã được đề cập ở Ví dụ 5 trang 22).
Lần 2, ước tính tại 1938, sử dụng 15 mốc.
Newton tiến:

```
PS C:\Users\Admin\.vscode> & C:/Users/Admin/AppData/Local/Programs/Python/Python39/python.exe c:/Users/Admin/.vscode/Newton_Interpolatio
n_Forward.py
Mời nhập giá trị cần tính: 1938
Chọn số lượng mốc nội suy tính (<= 29): 15
Các mốc nội suy:
[1905.0, 1910.0, 1915.0, 1920.0, 1925.0, 1930.0, 1935.0, 1940.0, 1945.0, 1950.0, 1955.0, 1960.0, 1965.0, 1970.0, 1975.0]
Hệ số của đa thức nội suy Newton: (bắt đầu từ hệ số tự do)
[  -0.263     62.43515 -195.91598  255.99103 -187.78763   87.41885  -27.54219    6.08953   -0.9628     0.10944   -0.00887    0.0005    -
0.00002    0.        -0.     ]
t = (x - 1905.0)/5.0
Giá trị cần tính tại 1938.0  là: -0.008203071998446387
Nhập số lượng thêm mốc nội suy: 0
PS C:\Users\Admin\.vscode>
```

Newton lùi:

```
PS C:\Users\Admin\.vscode> & C:/Users/Admin/AppData/Local/Programs/Python/Python39/python.exe c:/Users/Admin/.vscode/Newton_interpolatio
n_Backward.py
Mời nhập giá trị cần tính: 1938
Chọn số lượng mốc nội suy tính (<= 29): 15
Các mốc nội suy:
[1905.0, 1910.0, 1915.0, 1920.0, 1925.0, 1930.0, 1935.0, 1940.0, 1945.0, 1950.0, 1955.0, 1960.0, 1965.0, 1970.0, 1975.0]
Hệ số của đa thức nội suy Newton: (bắt đầu từ hệ số tự do)
[  -0.013    -36.98496 -115.45072 -152.37355 -114.56828  -55.3263   -18.24297   -4.24562   -0.70873   -0.08514   -0.00729   -0.00043   -
0.00002   -0.        -0.     ]
t = (x - 1975.0)/5.0
Giá trị cần tính tại 1938.0  là: -0.008203072212352423
Nhập số lượng thêm mốc nội suy: 0
PS C:\Users\Admin\.vscode>
```

Đồ thị:



Với 15 mốc, đồ thị đã đi qua hết các mốc.
Giá trị tại 1938 ≈ -0.008203072 (tương đối sát so với kết quả Ví dụ 5 trang 22)
Qua đây, ta thấy gói *pickPoint* đã giúp cho chương trình tối ưu hơn, đưa mốc $x0$ về thành điểm trung tâm.

4 Đánh giá phương pháp

- Đa thức nội suy Newton đã giải quyết được nhược điểm của đa thức nội suy Lagrange: Khi bổ sung mốc nội suy ta phải tính toán lại từ đầu chứ không thể dùng kết quả cũ để tính toán với số lượng ít hơn.
- Nội suy tại các điểm trung tâm sẽ có độ chính xác tốt hơn so với nội suy tại các điểm gần 2 đầu mút. Điều đó dẫn đến, vấn đề ngoại suy bằng đa thức nội suy sẽ có sai số rất lớn.
- Với độ dài đoạn nội suy nhỏ hơn bằng 1 (hoặc 2) thì nội suy sẽ tốt hơn. Tuy nhiên nếu sử dụng quá nhiều mốc và phân bố không đồng đều thì tính chính xác vẫn không được tốt.
- Việc sử dụng càng nhiều mốc nội suy không đồng nghĩa với việc độ chính xác càng tăng. Vì vậy ta cần xem xét kĩ lưỡng trong việc chọn số lượng và các mốc nội suy.

– Phương pháp nội suy đa thức Newton đã giảm được khối lượng tính toán đáng kể so với giải hệ phương trình để tìm đa thức (phương pháp Gauss giải hệ phương trình tuyến tính : $O(\frac{1}{3}n^3)$) .

5 Tài liệu tham khảo

1. Lê Trọng Vinh. *Giáo trình Giải tích số*. Nhà xuất bản Khoa học và kỹ thuật, 2007.

2. Phạm Kỳ Anh. *Giải tích số*. Nhà xuất bản Đại học Quốc gia Hà Nội, 1996.