TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC

Báo cáo bài số 17:

Nội suy Newton

Người hướng dẫn: *TS. Hà Thị Ngọc Yến*

SV thực hiện : 1. Nguyễn Mạnh Tuấn

2. Phạm Văn Huy

Mục lục

A. Mở đầu

1. Lời mở đầu

Trong thực tế, thường gặp những hàm số y=f(x) mà không biết được biểu thức giải tích cụ thể của chúng. Thông thường bằng đo đạc, thực nghiệm, ta sẽ thu được một bảng số, ở đó ta biết các giá trị y_i tại các điểm x_i tương ứng thuộc đoạn [a,b] nào đó, trong khi ta muốn biết giá trị của y tại các điểm $x \neq x_i$.

Ví dụ: Bài toán Price's House của Kaggle (một bài toán thực tế với các dữ liệu được thu thập từ hơn 1500 ngôi nhà khác nhau). Các ngôi nhà có khoảng hơn 80 thuộc tính(như diện tích, số tầng, số phòng ngủ...) và họ đưa cho bạn các ngôi nhà khác với thuộc tính tương ứng và nhiệm vụ của bạn là dự báo giá nhà cho các ngôi nhà này. Điều này rõ ràng có ý nghĩa kinh tế rất lớn, đơn giản như khi bán một ngôi nhà, bạn có thể dựa vào thông số chung của các ngôi nhà trên thị trường và phán đoán được khoảng giá của ngôi nhà bạn định bán là bao nhiêu sẽ hợp lí...

Quay lại với bài toán chúng ta đang nói ở trên. Rõ ràng do chúng ta không biết dạng của hàm f ra sao và chúng ta chỉ biết các giá trị của f tại các điểm hữu hạn. Vì vậy tìm ra chính xác biểu thức của hàm f là điều không thể. Tuy nhiên chúng ta có thể dựa vào các điểm (x,y) đã biết để " xấp xỉ" hàm f sao cho giá trị tại các điểm x khác có sai số chấp nhận được. Tư tưởng của các bài toán này giống như tư tưởng của các bài toán nội suy- xấp xỉ hàm.

Trong báo cáo này nhóm xin trình bày về phương pháp nội suy Newton

2.Khái niệm về đa thức nội suy

2.1 Bài toán

Bảng số

$$f(x_i)=y_i \qquad , (i = \overline{0,n}) \qquad (1)$$

Nghĩa là biết giá trị y_i tại các điểm x_i tương ứng (i=0,n) thuộc đoạn [a;b].

Trong khi ta muốn biết giá trị của y tại điểm $\overline{x} \neq x_i$ (i=0,n).

Từ bảng số liệu ta cần xây dựng một đa thức $P_n(x)$ có bậc ≤ n sao cho:

$$P_n(x_i)=y_i \quad , (i = \overline{0,n}) \quad (2)$$

Việc thay hàm $f(x)$ bằng một hàm $P_n(x)$ 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 $P(x)$ được sinh ra từ bảng số liệu (1) và thỏa mãn điêu kiện (2) được gọi là đa thức nội suy .

Các điểm $x_i \in$ [a, b], (i = $\overline{0,n}$) là các mốc nội suy, và ta cũng có thể nói $P_n(x) \approx$ f(x) trên đoạn [a, b].

Đặt $\overline{y} = P_n(\overline{x}) \approx f(\overline{x})$ với $\overline{x} \neq x_i$, i = $\overline{0,n}$.

$\overline{y}$ gọi là giá trị nội suy nếu $\overline{x} \in$ (a, b); gọi là giá trị ngoại suy nếu $\overline{x}$ nằm ngoài [a, b].

Ta đặt hiệu $R_n(\overline{x}) = f(\overline{x}) - P_n(\overline{x})$ gọi là sai số của phép tính nội suy tại điểm $\overline{x}$.

2.2 Sự duy nhất của đa thức nội suy

Định lý 1: Nếu có $x_0, x_1, x_2, \ldots, x_n$ là n+1 mốc nội suy khác nhau; và $y_0, y_1, y_2, \ldots, y_n$ là các giá trị tương ứng thì sẽ tồn tại duy nhất một đa thức $P_n(x)$ có bậc nhỏ hơn hoặc bằng n thỏa mãn điều kiện $P_n(x_i)=y_i$

Chứng minh:

Giả sử tồn tại hai đa thức $P_n(x)$ và $Q_n(x)$ cùng thỏa mãn là đa thức nội suy, nghĩa là $P_n(x_i)=Q_n(x_i)$ với i= $\overline{0,n}$.

Đặt $H(x)= P_n(x) - Q_n(x)$ thì rõ ràng H(x) là đa thức có bậc ≤ n, lại có n+1 nghiệm (do $P_n(x_i)=Q_n(x_i)$ với i= $\overline{0,n}$) nên rõ ràng H(x)≡ 0, hay

$$P_n(x_i) \equiv Q_n(x_i) \ .$$

2.3 Ý nghĩa của việc chọn công thức nội suy là đa thức

Ngoài ý nghĩa lịch sử ra, đa thức đại số thường được dùng trong phương pháp nội suy vì lý do đơn giản: các phép toán cộng, trừ,, nhân, đạo, hàm, tích phân dễ dàng thực hiện trên đa thức. Hơn nữa nếu P(x) là đa thức, còn c là hằng số thì P(x+c) và P(cx) cũng là đa thức.

Từ đây trở đi trong báo cáo này chúng ta chỉ xem công thức nội suy là các đa thức, và xem chúng là một.

B.Đa thức nội suy Newton với mốc bất kỳ

1.Ý tưởng phương pháp

Đa thức nội suy Lagrange đơn giản, dễ dàng tính toán, tuy nhiên nó lại có nhược điểm lớn là khi thêm vào một mốc nội suy, quá trình tính toán phải bỏ đi tất cả và làm lại từ đầu. Điều này vô cùng bất tiện.

Các bài toán thực tế rõ ràng chúng ta sẽ không cố định số mốc nội suy đã biết, mà theo thời gian chúng ta sẽ thu thập được thêm các mốc nội suy mới với các giá trị tương ứng, làm phong phú thêm nguồn dữ liệu của mình. Tương tự như vậy với bài toán nội suy chúng ta đang xem xét, khi thêm các mốc nội suy mới lại phải làm lại từ đầu quả là một bất tiện lớn.

Để giải quyết tình trạng trên, Newton đưa ra cách tiếp cận khác thuận lợi hơn, với xuất phát điểm từ ý tưởng khai triển Taylor.

- Theo khai triển Taylor

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

Với x_0=0, khai triển Maclaurin:

P$(x) = \sum_{k=0}^{n} \left(\frac{f^{(k)}(0)}{k!}\, x^k\right)$

Khai triển Taylor cho phép ta xấp xỉ một hàm khả vi tại một điểm thành một đa thức và có thể tính giá trị của đa thức tại các điểm lân cận điểm x_o,

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

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

Từ đây 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

Sau đây chúng ta sẽ đi vào chi tiết phương pháp nội suy mà Newton đưa ra.

2.Tỷ hiệu

2.1 Công thức

Xét hàm số $y = f(x)$ với x thuộc [a;b]:

Định nghĩa:

Từ bảng giá trị $y_i = f(x_i)$ trong đó các mốc nội suy x_i thuộc đoạn [a;b] và đôi một khác nhau ta đặt

- $f[x_{i-1}; x_i] = \frac{f(x_i)-f(x_{i-1})}{x_i-x_{i-1}}$ gọi là tỷ hiệu cấp 1 của $f(x)$
- Tỷ hiệu cấp 2 là tỷ hiêu của tỷ hiệu cấp 1, ký hiệu là:

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

Tiếp tục như vậy ta có tỷ hiệu cấp n là tỷ hiệu của tỷ hiệu cấp n-1 kí hiệu

$$f[x_0, x_1, \dots, x_n] = \frac{f[x_1, x_2, \dots, x_n] - f[x_0, x_1, \dots, x_{n-1}]}{x_n - x_0}$$

Nhận xét: Tỉ hiệu cấp n cần n + 1 mốc nội suy.

x	$f(x)$	$f[\dots,\dots]$	$f[\dots,\dots,\dots]$	…	$f[x_0, x_1, \dots, x_n]$
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_3	$f(x_3)$	$f[x_2, x_3]$	$f[x_1, x_2, x_3]$	…	
…	…	…	…	…	
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	$f(x_n)$	$f[x_{n-1}, x_n]$	$f[x_{n-2}, x_{n-1}, x_n]$	…	$f[x_0, \dots, x_n]$

Tỉ hiệu các cấp được mô tả trong bảng sau đây:

2.2 Tính chất

Tính chất 1:

a) Tỷ hiệu cấp k của tổng hai hàm số f và g bằng tổng các tỷ hiệu cùng cấp:

$$(f + g)[x_i, \dots, x_{i+k}] = f[x_i, \dots, x_{i+k}] + g[x_i, \dots, x_{i+k}]$$

b) Hằng số nhân c được đưa ra ngoài dấu tỷ hiệu:

$$(cf)[x_i, \dots, x_{i+k}] = cf[x_i, \dots, x_{i+k}]$$

Dễ dàng chứng minh được tính chất trên bằng phương pháp quy nạp.

Tính chất 2:

Tỷ hiệu có tính chất đối xứng: $f[x_i, \dots, x_{i+k}] = f[x_{i+k}, \dots, x_i]$

Tính chất này dễ dàng suy ra từ định nghĩa.

Tính chất 3:

a) Tỷ hiệu của hằng số bằng 0

b) Tỷ hiệu cấp m của đa thức bậc n có các tính chất sau:

- Nếu m=n thì tỷ hiệu cấp m là hằng số
- Nếu m>n thì tỷ hiệu cấp m bằng 0

Chứng minh:

a) Nếu f(x) = C(const) thì theo định nghĩa ta có:
$f[x_{i-1}; x_i] = \frac{f(x_i)-f(x_{i-1})}{x_i-x_{i-1}} = \frac{C-C}{x_i-x_{i-1}} = 0$ (đpcm)

b) Nếu f(x)=P(x)=$a_0+a_1x + a_2x^2 + \cdots + a_nx^n \quad (a_n \neq 0)$

Xét hàm số h(x)=x^k,k ∈ N.

Tỉ hiệu cấp 1:

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

Là đa thức cấp k-1. Áp dụng tính chất 1 và xét tỷ hiệu cấp hai, ta được đa thức cấp k-2.

Cứ như vậy sau k lần ta thu được tỷ hiêu cấp k của hàm số h(x)=x^k là hằng số, và tỷ hiệu cấp k+1 thì bằng 0 (Tính chất 3a).

Áp dụng tính chất 1 đối với đa thức P(x) ta được đpcm.

3.Đa thức nội suy Newton có các mốc bất kỳ

3.1 Xây dựng công thức

Từ bảng số (1): Với các mốc nội suy là x_i, i = $\overline{0,n}$, Newton đưa thêm vào mốc x ∈ [a, b] bất kì gần x_0 và tính:

$f[x; x_0] = \frac{f(x)-f(x_0)}{x-x_0} \Rightarrow f(x) = f(x_0) + (x - x_0)f[x, x_0]$ (3)

Ta lại có:

$$f[x, x_0, x_1] = \frac{f[x, x_0] - f[x_0, x_1]}{x - x_1}$$

Hay: $f[x, x_0] = f[x_0, x_1] + (x - x_1)f[x, x_0, x_1]$ (4)

Thay (4) vào (3) ta được:

$f(x) = f(x_0) + (x - x_0)f[x_0, x_1] + (x - x_0)(x - x_1)f[x, x_0, x_1]$

Lặp lại quá trình trên đối với tỷ hiệu cấp 3,4,... cuối cung ta được:

$$f(x) = f(x_0) + (x - x_0)f[x_0, x_1] + (x - x_0)(x - x_1)f[x_0, x_1, x_2] + \cdots$$
$$+(x - x_0)(x - x_1) \dots (x - x_{n-1})f[x_0, x_1, \dots, x_n]$$
$$+(x - x_0)(x - x_1) \dots (x - x_{n-1})(x - x_n)f[x, x_0, x_1, \dots, x_n] \quad (5)$$

Trong đẳng thức (5) nếu đặt

$$P_n(x) = f(x_0) + (x - x_0)f[x_0, x_1] + (x - x_0)(x - x_1)f[x_0, x_1, x_2] + \cdots$$
$$+(x - x_0)(x - x_1) \dots (x - x_{n-1})f[x_0, x_1, \dots, x_n] \quad (6)$$

Và $R_n(x) = (x - x_0)(x - x_1) \dots (x - x_{n-1})(x - x_n)f[x, x_0, x_1, \dots, x_n]$
$= \prod_{k=0}^{n} (x - x_k)f[x, x_0, x_1, \dots, x_n]$ (7)

thì rõ ràng $f(x) = P_n(x) + R_n(x)$ (8)

$P_n(x)$ được gọi là đa thức nội suy Newton và $R_n(x)$ là sai số của đa thức nội suy Newton

3.2 Chứng minh sự đúng đắn của phương pháp

Ta thấy $P_n(x)$ là đa thức bậc n sinh ra từ bảng số liệu (1). Ta chỉ cần chỉ ra rằng $P_n(x)$ thỏa mã điều kiện (2) thì $P_n(x)$ là đa thức nội suy

Tức là ta cần chứng minh $P_n(x_i) = y_i$, (i = $\overline{0,n}$).

Thực vậy, từ (7) ta có $y_i = f(x_i) = P_n(x_i) + R_n(x_i)$

Tại các mốc nội suy ta có:

$R_n(x) = \prod_{k=0}^{n} (x - x_k)f[x, x_0, x_1, \dots, x_n] = 0$

=> $y_i = f(x_i) = P_n(x_i)$ (i = $\overline{0,n}$).

Vậy ta có $P_n(x)$ là đa thức nội suy.

4.Sai số

Theo công thức (8) : $f(x) = P_n(x) + R_n(x)$

Trong đó $R_n(x) = \prod_{k=0}^{n} (x - x_k)f[x, x_0, x_1, \dots, x_n]$

Hay $R_n(x) = w_{n+1}(x)f[x, x_0, x_1, \dots, x_n]$

Do đa thức nội suy $P_n(x)$ là duy nhất nên sai số là bằng nhau dù ta có sử dụng đa thức nội suy Newton hay Lagrange. Như ở phần đa thức nội suy Lagrange ta có công thức đánh giá sai số:

$R_n(x) = kw_{n+1}(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} w_{n+1}(x)$ với $\xi \in [x,b]$.

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

5.*Nhận xét:*

- Nhìn vào công thức tính sai số $R_n(x)$ ta thấy ngay lí do tại sao ta chỉ nên sử dụng đa thức nội suy để tính gần đúng x ∈ [a, b]. Đó là để cho sai số nhỏ.Ngược lại nếu x nằm ngoài đoạn [a, b] thì khi đó hiệu $x - x_i$ sẽ lớn và tích của chúng sẽ lớn nên sai số sẽ lớn.
- Với n+1 mốc nội suy cho trước thì ta sẽ thu được một đa thức nội suy và một sai số cố định. Vì vậy ta chỉ có thể ước lượng được sai số. 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, tức là chúng ta phải chọn các mốc nội suy trước khi tiến hành tìm đa thức nội suy.
- Với đa thức nội suy newton với mốc bất kì, 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.

C.Đa thức nội suy Newton với mốc cách đều

1. Ý tưởng phương pháp

Trường hợp đa thức nội suy Newton có các mốc bất kì, chúng ta đã làm rất rõ ràng. Nhưng trong thực tế, việc chúng ta gặp trường hợp các mốc nội suy cách đều cũng khá là nhiều, và khi đó chúng ta cải biến ý tưởng đi một chút thì tính toán trên trường hợp các mốc bất kì sẽ đơn giản và thuận lợi hơn rất nhiều. Ta thấy việc tính toán đa thức nội suy Newton chỉ sử dụng các tỷ hiệu trên đường chéo chính, mà các giá trị hiệu $x_j - x_i$ trong trường hợp với các mốc cách đều không thay đổi, nên chúng ta có thể không thực hiện ngay phép chia cho hiệu giữa các mốc nội suy mà chỉ cần thực nó khi xuất ra đa thức. Điều đó sẽ khiến khối lương tính toán giảm đi rất nhiều. Dựa trên ý tưởng này chúng ta xây dựng công thức với trường hợp mốc nội suy cách đều

2.Sai phân hữu hạn

2.1 Công thức

Với trường hợp các mốc nội suy x_i(các mốc nội suy được sắp xếp theo thứ tự tăng dần), $i = \underline{0,n}$ có khoảng cách đều nhau một khoảng $h = x_i - x_{i-1}$ $\forall i = \underline{1,n}$

Đặt $\Delta x = x_{i+1} - x_i$ là số gia của đối số và gọi h là bước (h>0)

Biểu thức $\Delta(x) = f(x + h) - f(x)$ (1) là sai phân cấp một của hàm số $f(x)$ tại điểm x.

Tương tự sai phân cấp 2 là sai phân của sai phân cấp 1:

$\Delta^2 f(x) = \Delta(\Delta^1 f(x))=\Delta(f(x+h) - f(x)) = f(x+2h) - 2f(x+\mathrm{h}) + f(x)$

Sai phân cấp m là sai phân của sai phân cấp m-1 kí hiệu:

$\Delta^m(x) = \Delta(\Delta^{m-1} f(x))$; m=2,3,…

Từ đinh nghĩa ta thấy có thể xem Δ như toán tử được lập từ hàm (x) tương đương, nghĩa là Δ tác động lên f sao cho:
$\Delta f(x) = f(x+h) - f(x)$

2.2 Tính chất :

Tính chất 1: Δlà toán tửtuyến tính:

$\forall \alpha, \beta \in R$; $\forall y, z$ thì $\Delta(\alpha y + \beta z)=\alpha\Delta y + \beta\Delta z$

Và $\Delta C = 0$ $(C = const)$

$\Delta^n(x^n) = n!\, h^n$

$\Delta^m(x^n)$ =0 khi m>n

$\Delta^m(\Delta^k f) = \Delta^{m+k} f$

$\Delta^0 f = f$

Chứng minh:

Các công thức đều dễ dàng suy ra từ định nghĩa

Tính chất 2: Giá trị của hàm $f(x)$ được biểu diễn qua sai phân các cấp của nó:

$$f(x+mh) = \sum_{k=0}^{m} C_m^k \Delta^k f(x) \qquad (2)$$

Chứng minh: Từ(1) ta có $\Delta(x) = f(x+h) - f(x)$

Do coi Δ là toán tử, 1 là toán tử đơn vị nên ta có:

$(x+h) = (\Delta+1) f(x)$

Truy hồi ta có $(x + \mathrm{m}h)=(\Delta + 1)^m f(x)$ m=1,2,… (3)

Khai triển $(\Delta + 1)^m$ bằng nhị thức Newton ta có điều phải chứng minh.

Tính chất 3: Sai phân hữu hạn cấp m của hàm $f(x)$ được biểu diễn qua các giá trị liên tiếp của nó:

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

Chứng minh:

Do $\Delta^m(x) = [(1+\Delta) - 1]^m f(x)$

$$= (1+\Delta)^m f(x) - C_m^1(1+\Delta)^{m-1} f(x) + ... + (-1)^m f(x)$$

Sử dụng đẳng thức (3) ta được điều phải chứng minh

Tính chất 4: Giả sử (x) có đạo hàm liên tục đến cấp m trên đoạn [x;x+mh] thì ta có:

$\Delta^m(x) = h^m f^{(m)}(x+\theta mh)$ (5) Trong đó $0<\theta<1$

Chứng minh: Ta chứng minh công thức (5) bằng quy nạp

Khi m=1 ta có $\Delta(x) = f(x+h) - f(x) = hf'(x+\theta h)$ đúng (vì nó là công thức số gia giới nội)

Giả sử (5) đúng với m=k

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

Ta chứng minh (5) cũng đúng với m=k+1

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

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

Áp dụng định lý Lagrange ta được:

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

Đặt $\frac{\theta' k+\theta''}{k+1} = \theta$ thì $0<\theta<1$

Nên $\Delta^{k+1}(x) = h^{k+1} f^{(k+1)}(x+\theta(k+1)h)$ là điều phải chứng minh

Các công thức sai phân và các tính chất của nó sẽ được sử dụng để xây dựng các đa thức nội suy sau này.

2.3 Bảng sai phân hữu hạn:

Giả sử hàm số $y = f(x)$ được cho trong dạng một bảng số

$y_i = f(x_i)$,i = $\overline{0,n}$, các mốc nội suy

$x_i = x_0 + ih$,i = $\overline{0,n}$ (h=$\Delta x = x_{i+1} - x_i \quad \forall i = \underline{0,n}$)

Như vậy sai phân các cấp được xác định theo công thức

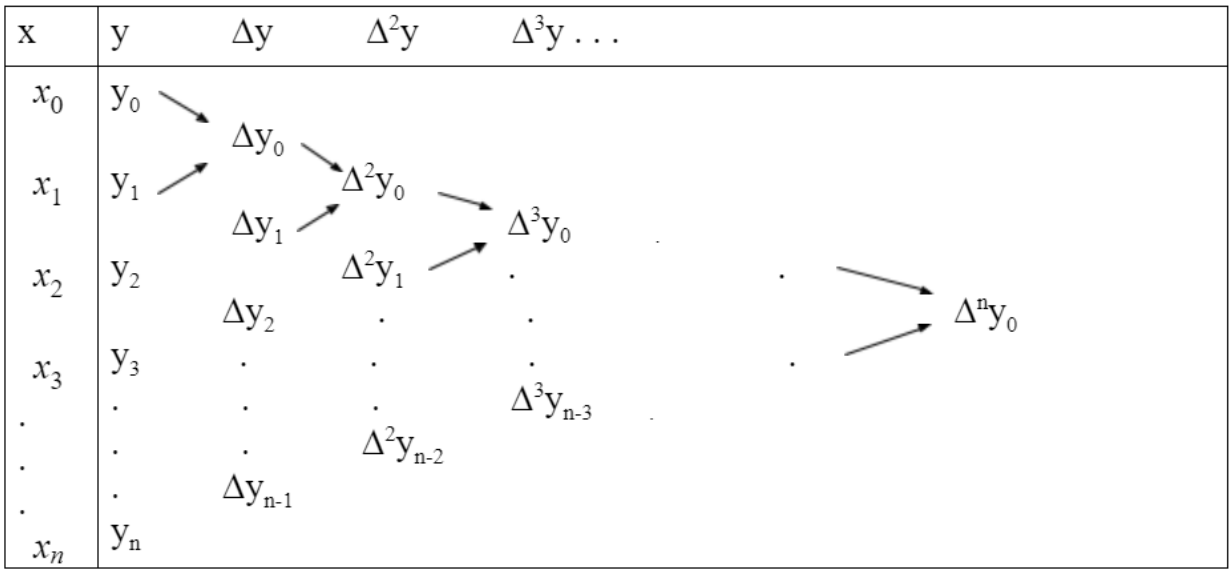
$\Delta y_i = y_{i+1} - y_i \quad \forall i = \underline{0, n-1}$ (sai phân cấp 1)

$\Delta^2 y_i = \Delta(\Delta y_i) = \Delta y_{i+1} - \Delta y_i \quad \forall i = \underline{0, n-2}$ là sai phân cấp 2

Tương tự sai phân cấp m là sai phân của sai phân cấp m-1

$\Delta^m y_i = \Delta(\Delta^{m-1} y_i) = \Delta^{m-1} y_{i+1} - \Delta^{m-1} y_i$ (6)

Công thức 6 được mô tả bằng bảng sau :



Hay:

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$	…	$\Delta^n y$
x_0	y_0	0	0	0	…	0
x_1	y_1	Δy_0	0	0	…	0
x_2	y_2	Δy_1	$\Delta^2 y_0$	0	…	0
x_3	y_3	Δy_2	$\Delta^2 y_1$	$\Delta^3 y_0$	…	0
…	…	…	…	…	…	
x_n	y_n	Δy_n	$\Delta^2 y_n$	$\Delta^3 y_n$	…	$\Delta^n y_0$

Bảng sai phân các cấp

Ta thấy, để có sai phân cấp 1, cần 2 mốc nội suy, sai phân cấp 2 cần 3 mốc nội suy... sai phân cấp n cần n+1 mốc nội suy.

3.Liên hệ giữa sai phân và tỷ hiệu:

$$\text{Ta có } f[x_0, x_1] = \frac{f(x_1)-f(x_0)}{x_1-x_0} = \frac{y_1-y_0}{x_1-x_0} = \frac{\Delta y_0}{h}$$

$$f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} = \frac{1}{2h^2}[\Delta y_1 - \Delta y_0]$$

$$= \frac{1}{2!\,h^2}\Delta^2 y_0$$

Từ đó suy ra: Với k=1,2…

$$f[x_0, x_1, \ldots, x_k] = \frac{1}{n!\,h^k}\Delta^k y_0$$

4.Đa thức nội suy Newton với mốc cách đều:

Ta có đa thức nội suy Newton tiến xuất phát từ x_0 (các mốc đã được sắp xếp theo thứ tự x_i (i= $\underline{1,n}$)

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

Đặt $x - x_k = (t - k)h \quad với \ \ k$=0,1,2,…

Thay vào ta được:

$$f(x) \approx P_n = P_n(x_0 + ht)$$
$$= y_0 + \frac{\Delta y_0}{1!}t + \frac{\Delta^2 y_0}{2!}t(t-1) + \dots + \frac{\Delta^n y_0}{n!}t(t-1) \dots (t-n+1)$$

Công thức trên được gọi là đa thức nội suy Newton tiến có các mốc cách đều (hệ số được sử dụng là hàng đầu của bảng sai phân)

Bây giờ xuất phát từ đa thức nội suy Newton lùi, tương tự như trên ta thu được đa thức:

$$f(x) \approx P_n = P_n(x_n + ht)$$
$$= y_n + \frac{\Delta y_n}{1!}t + \frac{\Delta^2 y_n}{2!}t(t-1) + \dots + \frac{\Delta^n y_n}{n!}t(t-1) \dots (t-n+1)$$

Công thức trên được gọi là đa thức nội suy Newton lùi có các mốc cách đều (hệ số được sử dụng là hàng cuối của bảng sai phân).

5.Sai số của đa thức nội suy Newton có mốc cách đều:

Từ $x = x_0 + ht$ ta được:

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

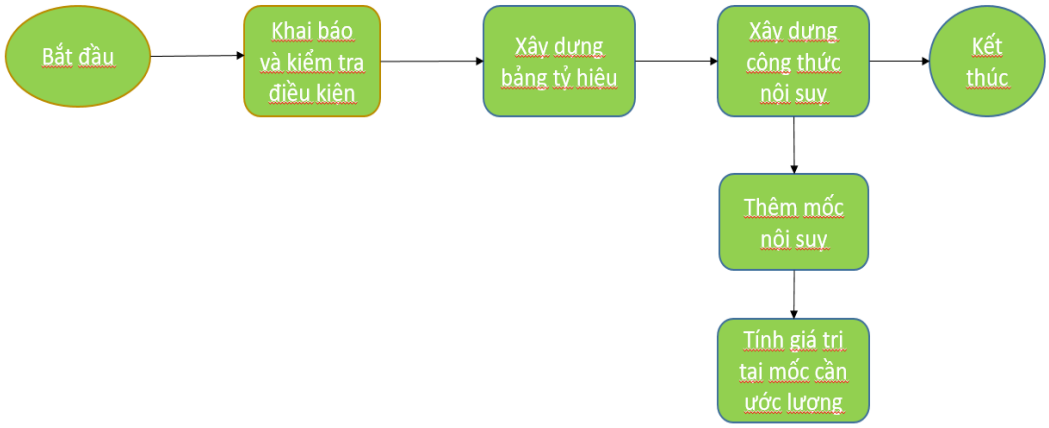
$$và \ do: \frac{\Delta^{n+1} y^0}{h^{n+1}}$$

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

D. Thuật toán và chương trình:

1. Thuật toán xây dựng đa thức nội suy Newton với mốc bất kỳ

Thuật toán xây dựng đa thức nội suy Newton với mốc bất kỳ được miêu tả như sau:



1.1 Khai báo và kiểm tra:

Def inputCheck(x, y):

1. Kiểm tra độ dài của vector x và vector y:

```
if length(x) != length(y):
        Mời nhập lại
else:
```

2. Kiểm tra xem có 2 giá trị nào của x trùng nhau không?, nếu có loại giá trị sau của x và y tương ứng

```
for i from 0 to n1-1:
    if there exist j != i such that x[j] = x[i]:
        delete x[j], y[j]
```

3. Trả về x và y sau khi đã được kiểm tra

```
return x, y
```

1.2 Xây dựng bảng tỷ hiệu:

- Sử dụng cấu trúc dữ liệu ma trận:

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

- Công thức tỷ hiệu:

$$f[x_i, x_{i+1}, \dots, x_{i+k}] = \frac{f[x_{i+1}, x_{i+2}, \dots, x_{i+k}] - f[x_i, x_{i+1}, \dots, x_{i+k-1}]}{x_{i+k} - x_i}$$

Def buildBTH(x, y, n):

1. Khởi tạo ma trận BTH chứa thông tin về bảng tỷ hiệu cỡ (n+1)x(n+1)

```
BTH = zeros(n+1, n+1)
```

2. Gán cột đầu của bảng tỷ hiệu bằng vector y

```
for i from 0 to n:
    BTH[i, 0] = y[i]
```

3. Xây dựng bảng dựa trên công thức tỷ hiệu

```
for j from 1 to n:
    for i from 0 to n-j:
        BTH[i, j] = (BTH[i, j-1] – BTH[i+1, j-1]) / (x[i] – x[i+j])
return BTH
```

1.3 Xây dựng công thức nội suy:

Def nsNewtonTien(x, y, n):

1. Tạo bảng tỷ hiệu

```
BTH = buildBTH(x, y, n)
```

2. Khởi tạo hàm f

```
f = BTH[0,0]  (f(x_0))
```

3. Cộng từng phần tử trong công thức nội suy

```
for i from 1 to n:
    f += (t − x_0) * ⋯ * (t − x_{i−1}) * BTH[0, i]
```

4. Trả về hàm f

```
return f
```

Def nsNewtonLui(x, y, n):

1. Xây dựng bảng tỷ hiệu

```
BTH = buildBTH(x, y, n)
```

2. Khởi tạo hàm f

```
f = BTH[n,0]  (f(x_n))
```

3. Cộng từng phần tử trong công thức nội suy

```
for i from 1 to n:
    f += (t − x_n) * ⋯ * (t − x_{n−i+1}) * BTH[n − i, i]
```

4. Trả về hàm f

```
return f
```

1.4 Thêm mốc nội suy

```
Def addBTH(x, BTH, n, x_add, y_add):
1. Đặt biến x mới
        x_new = x.append(x_add)
        m = length(x_add)
2. Khởi tạo ma trận bảng tỷ hiệu mới newBTH
        newBTH = zeros(n+1+m, n+1+m)
3. Copy BTH vào newBTH theo đúng vị trí
4. Gán những phần tử còn lại của cột đầu thành các giá trị của y
5. Tính các giá trị còn lại của bảng tỉ hiệu mới
        for j from 1 to n+1+m:
                for i from n+1 to n+1+m-j:
                        newBTH[i, j] = (newBTH[i+1, j-1] – newBTH[i, j-1]) / (x[i+j] – x[i])
6. Trả về bảng tỷ hiệu mới
        return newBTH
```

```
Def addBTH(x, BTH, n, x_add, y_add):
1. Đặt biến x mới
        x_new = x.append(x_add)
        m = length(x_add)
2. Khởi tạo ma trận bảng tỷ hiệu mới newBTH
        newBTH = zeros(n+1+m, n+1+m)
3. Copy BTH vào newBTH theo đúng vị trí
4. Gán những phần tử còn lại của cột đầu thành các giá trị của y
5. Tính các giá trị còn lại của bảng tỉ hiệu mới
        for j from 1 to n+1+m:
                for i from n+1 to n+1+m-j:
                        newBTH[i, j] = (newBTH[i+1, j-1] – newBTH[i, j-1]) / (x[i+j] – x[i])
6. Trả về bảng tỷ hiệu mới
        return newBTH
```

1.5 Tính giá trị tại mốc cần ước lượng

```
Def pickPoint(x, x0, m):
1. Kiểm tra điều kiện: m < length(x)
2. Chọn ra  m điểm có khoảng cách tới x nhỏ nhất:
        hieu = abs(x – [x0])
        for i from 1 to num:
                index.add(i): hieu(i) = min(hieu)
                hieu.delete(hieu[i])
3. Trả về mảng index tương ứng với các chỉ số cần lấy
```

Def pickPoint1(x, x0, m):

1. Kiểm tra xem m < length(x).
2. Nếu có:
    1. Tạo mảng hieu = abs(x – [x0])
    2. Sắp xếp mảng hiệu và lấy ra các chỉ số tương ứng
    3. Chọn ra m chỉ số đầu tiên của hiệu hieu
    4. Trả về m chỉ số này.

Def estimate(x, y, x0):

1. Xác định các mốc cần lấy: x = x[pickPoint(x, x0, m)],…
2. Xây dựng bảng tỷ hiệu ứng với x, y
    n = length(x) – 1
    BTH = buildBTH(x, y, n)
3. Xây dựng công thức nội suy tương ứng
    f = nsNewtonTien(x, BTH, n)

Hoặc f = nsNewtonLui(x, BTH, n)

4. Tính và trả về giá trị tại mốc cần ước lượng
    return f(x0)

Chương trình (viết bằng ngôn ngữ Python):

```
###
Import
library


          import numpy as np
          import matplotlib.pyplot as plt
          from sympy import *
          from sympy import init_printing
          init_printing()


          # Nhap input
          def inputData():
              x = []
              y = []
              with open('Newton.txt','r+') as f:
                      for line in f.readlines():
                          xt = float(line.split(' ')[0])
                          yt = float(line.split(' ')[1])
                          check = True
                          for x_check in x:
                              if x_check == xt:
```

```
                            check = False
                            break
                    if check:
                        x.append(xt)
                        y.append(yt)
        return x, y, len(x)-1


# Dung bang ty hieu
def buildBTH(x, y, n):
    ## Khoi tao
    BTH = np.zeros([n+1, n+1])
    ## Gan cot dau
    for i in range(n+1):
        BTH[i, 0] = y[i]
    ## Xay dung ty hieu
    for j in range(1,n+1):
        for i in range(n+1-j):
            BTH[i, j] = (BTH[i+1, j-1] - BTH[i, j-1]) / (x[i+j] - x[i])
    return BTH



# Noi suy Newton tien
def nsNewtonTien(x, y, n):
    BTH = buildBTH(x, y, n)
    t = Symbol('t')
    f = BTH[0, 0]
    var = (t - x[0])
    for i in range(1,n+1):
        f += var * BTH[0, i]
        var = var * (t - x[i])
    return f



# Noi suy Newton lui
def nsNewtonLui(x, y, n):
##    f = Symbol('f')
    BTH = buildBTH(x, y, n)
    t = Symbol('t')
    f = BTH[n, 0]
    var = (t - x[n])
    for i in range(1,n+1):
        f += var * BTH[n-i, i]
        var = var * (t - x[n-i])
    return f



# Xap xi gia tri
def pickPoints(x, x0, num):
    if num > len(x):
        raise Exception('Moi nhap lai')
    else:
        hieu = [abs(x[i] - x0) for i in range(len(x))]
        index = [i[0] for i in sorted(enumerate(hieu), key=lambda t:t[1])]
        return index[:num]


def estimate(x, y, x0, deg):
```

```
            index = pickPoints(x, x0, deg+1)
            x1 = [x[i] for i in index]
            y1 = [y[i] for i in index]
            ## buildBTH
            BTH = buildBTH(x1, y1, deg)
            f = nsNewtonTien(x1, y1, deg)
            value = f.subs(Symbol('t'), x0)
            return f, value


        def main():
            x, y, n = inputData()
            x0 = float(input("Moi nhap gia tri can tinh: "))
            deg = int(input("Moi nhap bac da thuc (< bac lon nhat): "))
            f, v = estimate(x, y, x0, deg)
            print("Da thuc noi suy la: ", simplify(f))
            print("Gia tri can tinh tai ", x0, " la: ", v)
            ## plot
            xx = np.linspace(x[0], x[-1], 100)
            fx = [f.subs(Symbol('t'), xxx) for xxx in xx]


            plt.figure()
            plt.scatter(x, y, marker='*')
            plt.plot(xx, fx)
            plt.xlabel('Points')
            plt.ylabel('Values')


        if __name__=='__main__':
            main()
```

1.6 Độ phức tạp tính toán:

- Không gian: $O(n^2)$
- Thời gian: $O(n^2)$

1.7 Kết quả chạy

Ví dụ với bảng dữ liệu:

x	11	13	14	18	19	21
y	13.42	14.10	17.58	18.50	18.78	22.82

a. Sử dụng thuật toán trên, ta tìm được đa thức nội suy:

$f(t) = 0.34t - 0.00476190476190476(t-19)(t-18)(t-14)(t-13)(t-11) + 0.044047619047619(t-18)(t-14)(t-13)(t-11) - 0.242380952380952(t-14)(t-13)(t-11) + 1.04666666666667(t-13)(t-11) + 9.68$

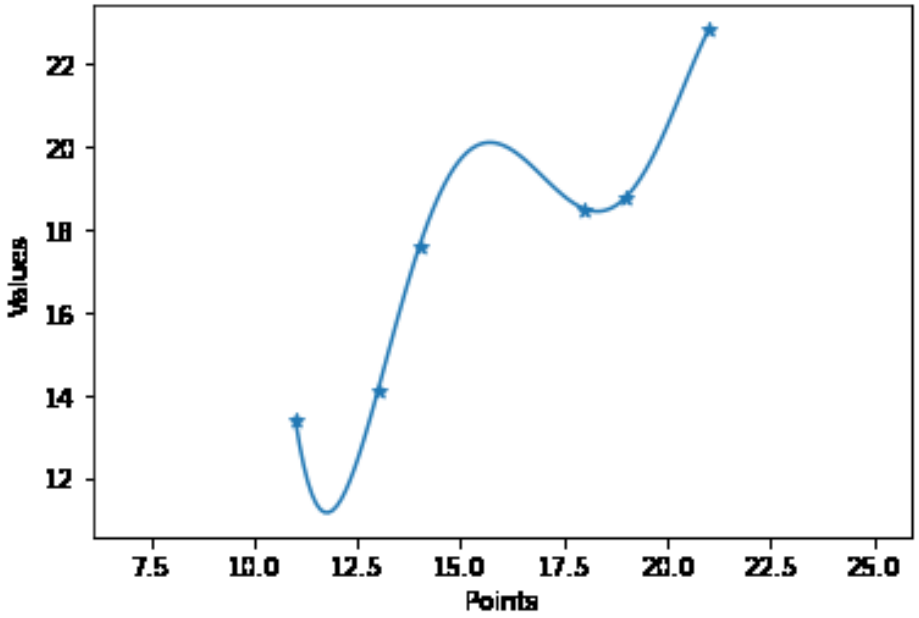
Rút gọn, ta được:

$f(t) = -0.00476190476190476t^5 + 0.4011904761904 76t^4 - 13.3138095238095t^3 + 217.298809523809t^2 - 1741.66142857143t + 5492.3$

b. Xấp xỉ giá trị tại mốc 13.5:

$f(13.5) = 15.9273660714286$

c. Vẽ đồ thị biểu diễn:



d. Thêm mốc nội suy

Xét dữ liệu:

x	23	25
y	23.56	24.74

Thêm 2 mốc dữ liệu này vào bảng dữ liệu, ta được đa thức nội suy mới:

$f1(t) = 0.34t + 9.68 + 1.04666666666667(t-13)(t-11) +0.115650910894661(t-23)(t-21)(t-19)(t-18)(t-14)(t-13)(t-11) - 0.281630555555556(t-21)(t-19)(t-18)(t-14)(t-13)(t-11) - 0.00476190476190476(t-19)(t-18)(t-14)(t-13)(t-11) + 0.044047619047619(t-18)(t-14)(t-13)(t-11) - 0.242380952380952(t-14)(t-13)(t-11)$

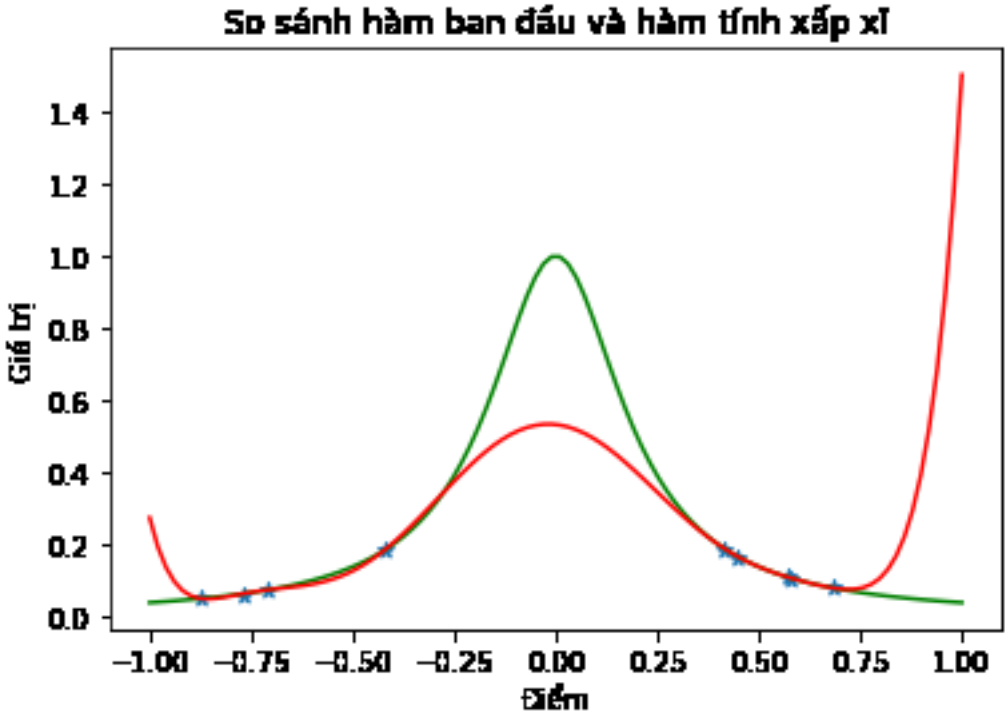
Rút gọn lại, ta được:

$f1(t) = 0.115650910894661t^7 - 14.0440889520202t^6 + 722.093745905484t^5 - 20376.4273410354t^4 + 340789.615767578t^3 - 3377806.45187358t^2 + 18370649.493407t - 42289924.898125$

e. Xấp xỉ hàm bằng đa thức nội suy:

Xét hàm số: $g(x) = \frac{1}{25x^2+1}$

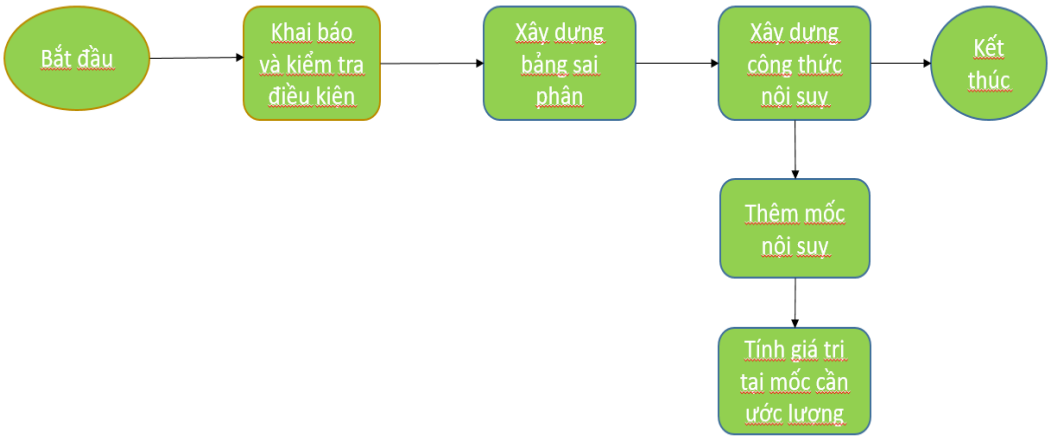
Sau khi xấp xỉ bằng đa thức nội suy Newton với 9 mốc, ta so sánh hàm g với đa thức nội suy của nó bởi đồ thị sau:



Trong đó đường màu xanh thể hiện giá trị thực của g còn đường màu đỏ thể hiện giá trị xấp xỉ bằng đa thức nội suy qua 9 mốc lấy bất kì trên [-1;1]. Ta thấy đa thức nội suy xấp xỉ hàm số tương đối chính xác ở các điểm gần trung tâm và mốc lấy nội suy.

2. ***Thuật toán nội suy Newton với mốc cách đều***

Thuật toán xây dựng đa thức nội suy Newton với mốc cách đều được miêu tả như sau:



2.1 Khai báo và kiểm tra điều kiện

Def inputCheck(x, y):

1. Kiểm tra độ dài của vector x và vector y:

```
if false(length(x) != length(y)):
        Mời nhập lại
else:
```

2. Sắp xếp lại x và y tương ứng

3. Kiểm tra xem có 2 giá trị nào của x trùng nhau không?, nếu có loại giá trị sau của x và y tương ứng

```
for i from 0 to n1-1:
        if x[i+1] == x[i]:
                delete x[i+1], y[i+1]
```

4. Trả về x và y sau khi đã được kiểm tra

```
return x, y
```

2.2 Xây dựng bảng sai phân

Def buildBSP(y, n):

1. Khởi tạo ma trận BSP chứa thông tin về bảng sai phân cỡ (n+1)x(n+1)

```
BSP = zeros(n+1, n+1)
```

2. Gán cột đầu của bảng sai phân bằng vector y

```
for i from 0 to n:
        BSP[i, 0] = y[i]
```

3. Xây dựng bảng dựa trên công thức sai phân

```
for j from 1 to n:
        for i from 0 to n-j:
                BSP[i, j] = BSP[i+1, j-1] – BSP[i, j-1]
return BSP
```

2.3 Xây dựng công thức nội suy

Def nsTien(x, y, n):

1. Tạo bảng sai phân

```
BSP = buildBSP(y, n)
```

2. Khởi tạo hàm f

```
f =  BSP[0,0]  (f(x_0))
```

3. Cộng từng phần tử trong công thức nội suy

```
for i from 1 to n:
```

$$f \mathrel{+}= \frac{t(t-1)\dots(t-i+1)}{i!} * BSP[0,i]$$

4. Trả về hàm f

```
return f
```

Def nsLui(x, y, n):

1. Tạo bảng sai phân

   BSP = buildBSP(y, n)

2. Khởi tạo hàm f

   f = BSP[n,0] (f(x_n))

3. Cộng từng phần tử trong công thức nội suy

   for i from 1 to n:

   f += $\frac{t(t+1)\dots(t+i-1)}{i!} * BSP[n-i, i]$

4. Trả về hàm f

   return f

2.4 Tính giá trị tại mốc cần ước lượng

Def Estimate(x, y, h, x0, deg):

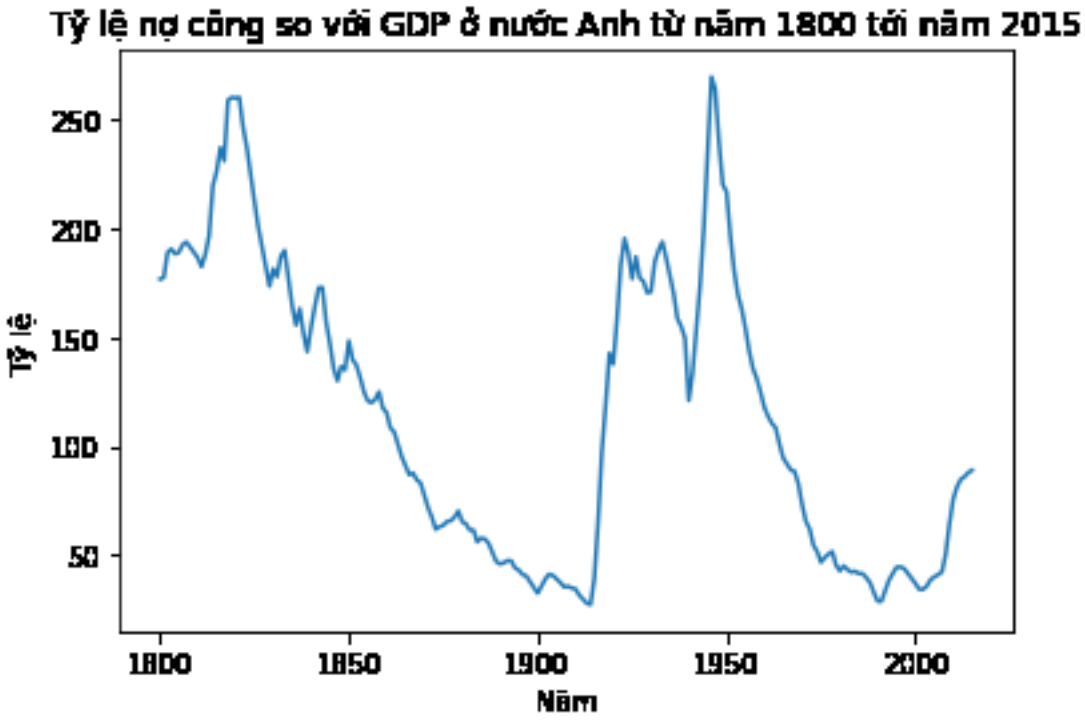
1. Check input.
2. Lấy khoảng cách con [l, u], $l = \left\lfloor \frac{x_0 - x[0]}{h} \right\rfloor, u = l + 1.$
3. Chọn deg + 1 điểm gần x0 nhất làm mốc nội suy, gọi là x1 và y1.
4. Xây dựng bảng sai phân.
5. Xây dựng công thức nội suy tiến f.
6. Tính giá trị tại điểm x0: $f(\frac{x_0 - x_{lower}}{h})$.
7. Trả về giá trị

2.5 Độ phức tạp tính toán:

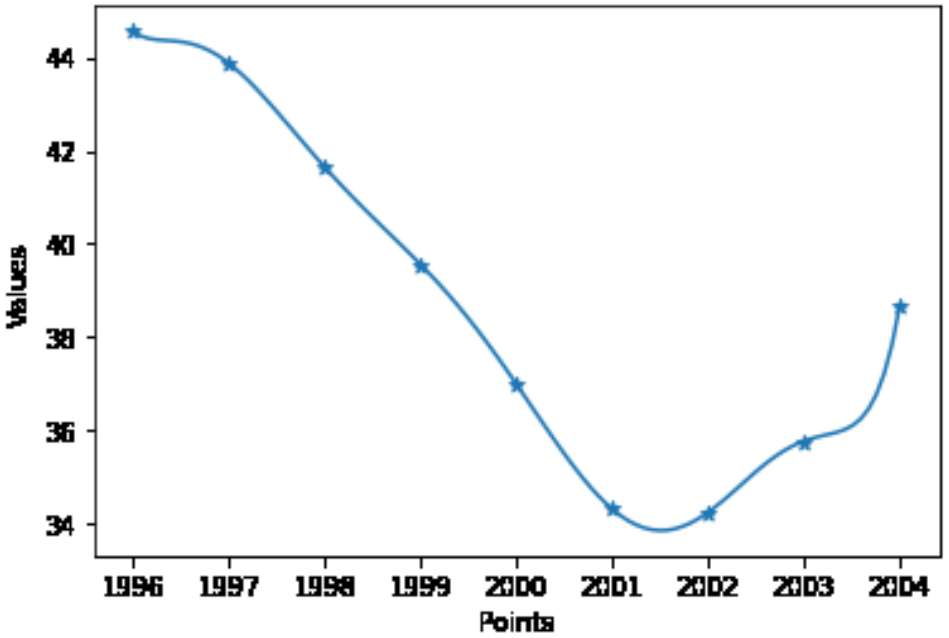
- Độ phức tạp không gian: $O(n^2)$
- Độ phức tạp thời gian: $O(n^2)$

2.6 Kết quả chạy

a. Xét bộ dữ liệu nợ công của Vương Quốc Anh qua từng năm trong khoảng thời gian 1800-2015 được thể hiện bởi đồ thị sau:



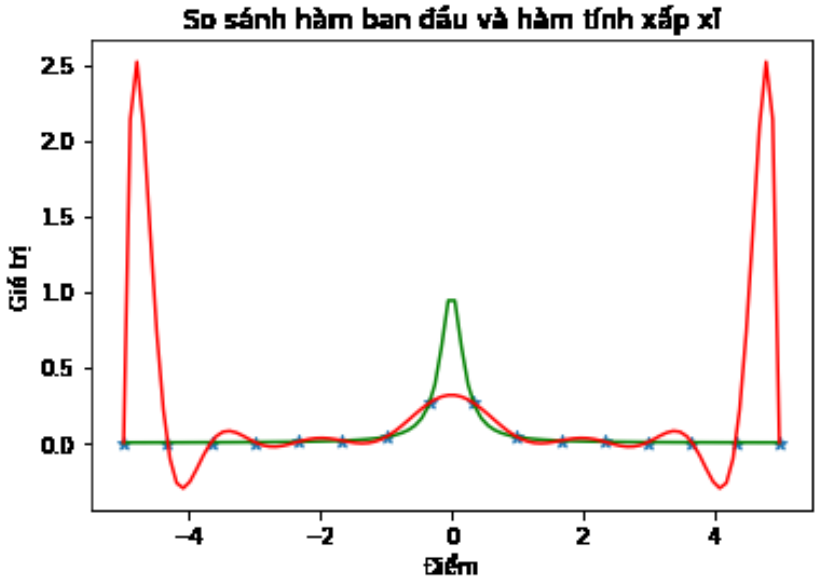
Ta sử dụng đa thức nội suy Newton bậc 8, với 9 điểm nội suy, để ước lượng tỷ lệ nợ công tại thời điểm 1999.5 (giữa năm 1999):



Và được ước lượng tại thời điểm 1999.6:

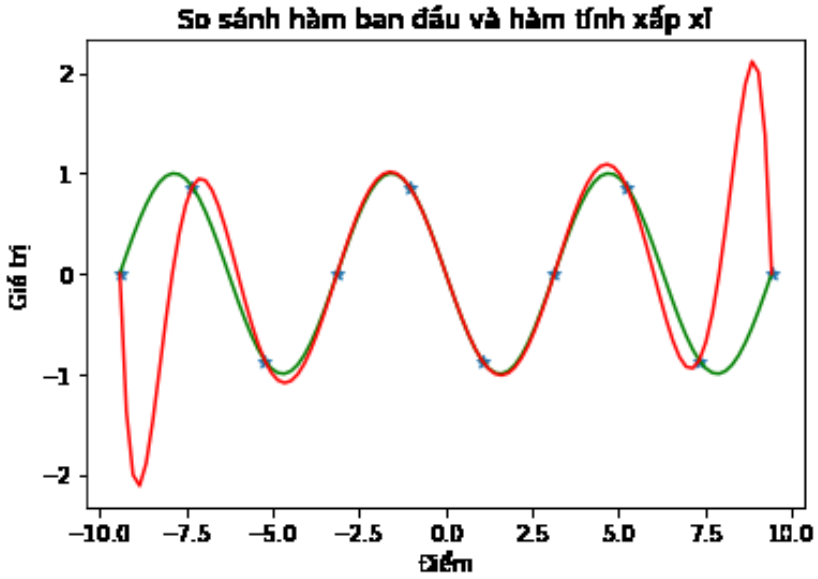
f(1999.5) H 38.3890487670898

b. Xấp xỉ hàm số g(x) = $\frac{1}{1+25x^2}$ trên đoạn [-5; 5] với 16 mốc nội suy cách đều:



Ta thấy đa thức xấp xỉ hàm số khá tốt ở các điểm gần trung tâm và các mốc nội suy; kém tại gần 2 đầu đoạn.

c. Xấp xỉ hàm số g(x) = sin(x) trên đoạn $[-3\pi; 3\pi]$ với 10 mốc nội suy cách đều:



Tương tự, ta cũng thấy đa thức nội suy xấp xỉ hàm số khá tốt ở các điểm gần trung tâm và gần các mốc nội suy; xấp xỉ không tốt ở gần 2 đầu đoạn.

E. Kết luận

Trong báo cáo này, nhóm chúng em (nhóm 17) đã trình bày về tổng quan lý thuyết và cài đặt chương trình thực tế cho phương pháp nội suy đa thức Newton.

Do còn đang trong quá trình học hỏi, chúng em không tránh khỏi những sai sót cần sửa chữa. Do vậy, mong quý thầy cô, các bạn trong lớp và các bạn độc giả giúp đưa ra những đóng góp mang tính xây dựng để chúng em hoàn thiện báo cáo này một cách tốt hơn.

Cuối cùng, chúng em xin chân thành cảm ơn TS. Hà Thị Ngọc Yến, giảng viên hướng dẫn môn học, đã tận tình đưa ra những chỉ dẫn và phản biện thiết thực, giúp chúng em có thể nghiên cứu và hoàn thiện bản báo cáo này. Chúng mình cũng xin cảm ơn các bạn cùng lớp học vì những góp ý chính xác của mình.

Hà Nội, 12/2020