BÁO CÁO MÔN GIẢI TÍCH SỐ

NỘI SUY LAGRANGE

Nguyễn Đình Huy 20200277

Hà Nội 2022

MI-TN K65

Giảng viên hướng dẫn: Hà Thị Ngọc Yến

Mục lục

1 Giới thiệu

Cho hàm số $y = f(x)$ trên đoạn $[a, b]$, được biểu diễn qua bảng các giá trị

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

Cụ thể hơn

x	x_0	x_1	x_2	...	x_n
$f(x)$	y_0	y_1	y_2	...	y_n

Với $\{(x_i, y_i)\}_{i=0}^{n}$ là $n+1$ bộ số cho trước, việc xây dựng 1 hàm số xấp xỉ $f(x)$ có $n+1$ điểm giá trị trên trùng với $f(x)$, được gọi là nội suy.

Trên thực tế có rất nhiều phương pháp nội suy nhiều biến phức tạp, xử lý những hàm số biến thiên khó đoán. Ví dụ như phép nội suy *Krigin* là công cụ quan trọng trong địa chất học, dùng để xác định độ ẩm, nhiệt độ,... của địa hình; hay nội suy *IDW (Inverse Distance Weighting)* được sử dụng nhiều trong Digital Elevation Model và dự báo thời tiết.

Tuy nhiên mô hình đơn giản nhất cho nội suy là nội suy đa thức, và bài báo cáo này sẽ giới thiệu công thức nội suy đa thức: nội suy Lagrange.

2 Kiến thức chung về nội suy đa thức

2.1 Các định nghĩa

Các đa thức $P_n(x)$ được gọi là đa thức nội suy.
Các điểm $x_i \in [a, b]$ với $i = \overline{0, n}$ gọi là các mốc nội suy.
Giá trị $\overline{y} = P_n(\overline{x}) \approx f(\overline{x})\ (\overline{x} \neq x_i), i = \overline{0, n}$ được gọi là giá trị nội suy khi $\overline{x} \in (a, b)$.
Hiệu số $R_n(\overline{x}) = f(\overline{x}) - P_n(\overline{x})$ gọi là sai số của phép tính nội suy tại điểm $\overline{x}$.

2.2 Tính chất

1. $P_n(x_i) = f(x_i) \quad , i = \overline{0, n}$.
2. Bậc của đa thức $P_n(x) \leq n$.
3. Đa thức nội suy tìm được từ một bộ điểm là duy nhất.

Tính chất thứ nhất và thứ hai có thể thấy hiển nhiên, bạn đọc tự chứng minh, ta sẽ đi chứng minh tính chất thứ ba.

Chứng minh: Giả sử ta tìm được hai đa thức $P_n(x)$ và $Q_n(x)$ có bậc $\leq n$ thỏa mãn:

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

Đặt $H_n(x) = P_n(x) - Q_n(x)$. Do đó $H_n(x)$ có bậc $\leq n$.
Mặt khác, $H_n(x_i) = P_n(x_i) - Q_n(x_i) = 0, i = \overline{0,n}$.
$\Rightarrow H_n(x)$ có bậc $\leq n$ và có ít nhất $n+1$ nghiệm.
$\Rightarrow H_n(x) \equiv 0$ hay $P_n(x) \equiv Q_n(x)$.
Vậy đa thức nội suy là duy nhất.

3 Lý thuyết chính về nội suy Lagrange

3.1 Ý tưởng

Trên thực tế, ta thường gặp những hàm số không biết biểu thức cụ thể của chúng hoặc biểu thức hàm số quá phức tạp. Bằng đo đạc, thực nghiệm ta thu được một bảng số các giá tri y_i tại các điểm x_i thuộc đoạn $[a,b]$. Ta muốn biết giá trị của y tại điểm $x = x_i$.

Ta tìm cách thay hàm f(x) bởi hàm P(x) đơn giản hơn để sai lệch của P(x) và f(x) không đáng kể.

Thường P(x) được chọn là đa thức (giá trị tính và các phép đạo hàm, tích phân dễ dàng thực hiện).

3.2 Công thức

$$L_i(x) = \frac{(x-x_0)\dots(x-x_{i-1})(x-x_{i+1})\dots(x-x_n)}{(x_i-x_0)\dots(x_i-x_{i-1})(x_i-x_{i+1})\dots(x_i-x_n)}$$

$$P_n(x) = \sum_{i=0}^{n} y_i L_i(x)$$

$P_n(x)$ là đa thức nội suy Lagrange.
Chứng minh: $P_n(x_i) = y_i, i = \overline{0,n}$.

Dễ thấy, $L_i(x_j) = \begin{cases} 0, i \neq j. \\ 1, i = j. \end{cases}$

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

3.3 Sai số

$M = \sup\limits_{x\in[a,b]} |f^{(n+1)}(x)|$

$|R_n(\overline{x})| = |f(\overline{x}) - P_n(\overline{x})| \leq \frac{M}{(n+1)!}|(\overline{x}-x_0)(\overline{x}-x_1)\dots(\overline{x}-x_n)|$

4 Thuật toán

4.1 Biến đổi biểu thức

Ta có công thức :

$$L_i(x) = \frac{(x-x_0)\dots(x-x_{i-1})(x-x_{i+1})\dots(x-x_n)}{(x_i-x_0)\dots(x_i-x_{i-1})(x_i-x_{i+1})\dots(x_i-x_n)}$$

$$P_n(x) = \sum_{i=0}^{n} y_i L_i(x)$$

Biến đổi : Đặt $w_{n+1}(x) = \prod_{i=0}^{n}(x-x_i)$

Khi đó, $P_n(x) = \sum_{i=0}^{n} \frac{y_i}{w'_{n+1}(x_i)} . \frac{w_{n+1}(x)}{x-x_i}$

Ta cần tính :

- $\frac{y_i}{w'_{n+1}(x_i)}$
- $\frac{w_{n+1}(x)}{x-x_i}$

4.2 Ký hiệu chung trong code

```
ldb = long double
FOR(i, a, b) = for (int i = a; i <= b; i++)
FORD(i, b, a) = for (int i = b; i >= a; i--)
```

4.3 Gói Print(v : vector<ldb>) $\longrightarrow$ void

Đây là gói in các giá trị của vector v (bắt đầu từ v_0).

- **Đầu vào:** Vector v.

- **Đầu ra:** In giá trị các phần tử trong vector(bắt đầu từ v_0).

Code:

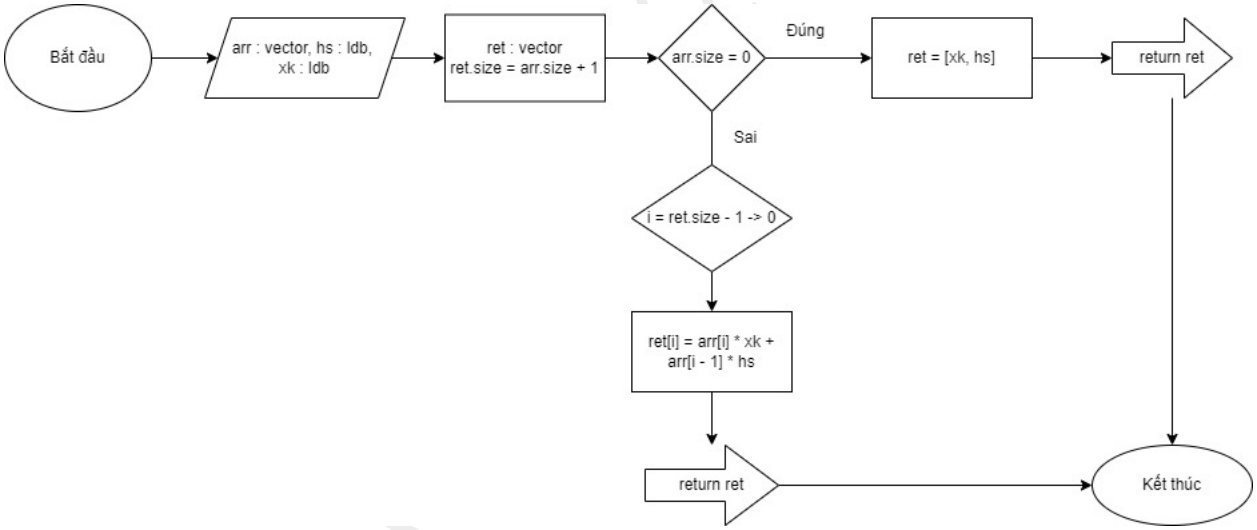
```
void Print(vector<ldb> &v) {
        cout << "In cac he so tu x^0 den x^" << v.size() - 1 << "\n";
        for (auto x : v) cout << x << " " ; cout << "\n\n";
}
```

4.4 Gói hoocnerNhan(arr : vector<ldb>, hs : ldb, xk : ldb) $\longrightarrow$ vector<ldb>

Đây là gói tính các hệ số khi thực hiện phép nhân đa thức với $hs * x + xk$.

- **Đầu vào:** Vector `arr` là các hệ số của đa thức nhân (hệ số đa thức từ hệ số tự do), số thực `hs`, `xk`
- **Đầu ra:** Vector chứa các hệ số khi nhân đa thức hệ số `arr` với `(hs*x-xk)` (hệ số đa thức từ hệ số tự do)

Sơ đồ thuật toán:



Code:

```cpp
vector<ldb> hoocnerNhan(vector<ldb> &arr, ldb hs, ldb xk) {
    vector<ldb> ret; ret.resize(arr.size() + 1);

    if (arr.size() == 0) {
            ret.resize(2);
            ret[0] = xk, ret[1] = hs;
            return ret;
    }

    FORD(i, ret.size() - 1, 0) {
            ldb tmp = 0;                                    // hệ số x^i
            if (i > 0) tmp += arr[i - 1] * hs;
            if (i < arr.size()) tmp += arr[i] * xk;
            ret[i] = tmp;
    }

    return ret;
}
```

Ví dụ:

```
hoocnerNhan([6, -5, 1], 1, -2)
>> [-12, 16, -7, 1]
```

Do [6, -5, 1] là hệ số của $(6 - 5x + x^2)$ nên ta xét phép nhân đa thức:

$$(6 - 5x + x^2)(x - 2) = -12 + 16x - 7x^2 + x^3$$

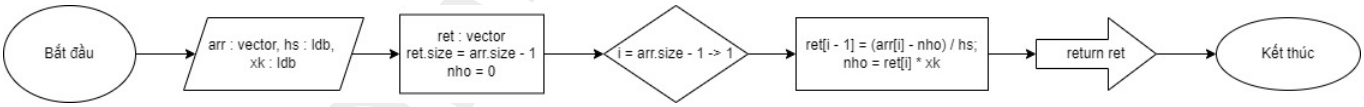
$\Longrightarrow$ Đầu ra sẽ là [-12, 16, -7, 1]

4.5 Gói hoocnerChia(arr : vector<ldb>, hs : ldb, xk : ldb) $\longrightarrow$ vector<ldb>

Đây là gói tính các hệ số khi thực hiện phép chia đa thức cho $hs * x + xk$.

- **Đầu vào:** Vector arr là các hệ số của đa thức bị chia (hệ số đa thức từ hệ số tự do), số thực hs, xk
- **Đầu ra:** Vector chứa các hệ số khi chia đa thức hệ số arr với (hs*x-xk) (hệ số đa thức từ hệ số tự do)

Sơ đồ thuật toán:



Code:

```
vector<ldb> hoocnerChia(vector<ldb> &arr, ldb hs, ldb xk)        {
        vector<ldb> ret; ret.resize(arr.size() - 1);

        ldb nho = 0;
        FORD(i, arr.size() - 1, 1) {
                ldb val = (arr[i] - nho) / hs;                          // hệ số x^(i-1)
                nho = val * xk;
                ret[i - 1] = val;
        }

        return ret;
}
```

Ví dụ:

```
hoocnerChia([6, -5, 1], 1, -2)
>> [-3, 1]
```

Do [6, -5, 1] là hệ số của $(6 - 5x + x^2)$ nên ta xét phép chia đa thức:

$$(6 - 5x + x^2)/(x - 2) = -3 + x$$

$\Longrightarrow$ Đầu ra sẽ là [-3, 1]

4.6 Gói Get-Lagrange(arr-wi : vector<ldb>, pos : int) $\longrightarrow$ void

Đây là gói tính hệ số của $y_{pos}.L_{pos}(x)$ và cập nhật lên cho hệ số của đa thức nội suy Lagrange cần tìm.

- **Đầu vào:** Vector arr-wi là các hệ số của đa thức $(x - x_1)(x - x_2)...(x - x_n)$(hệ số đa thức từ hệ số tự do), số nguyên pos.
- **Đầu ra:** Không trả về giá trị hay kết quả gì (chỉ tính toán trực tiếp).

Code:

```
void Get_Lagrange(vector<ldb> &arr_wi, int pos) {
        vector<ldb> ret = hoocnerChia(arr_wi, 1, -a[pos].x);

        ldb val = a[pos].y / dw[pos];
        FOR(i, 0, arr_Lagrange.size() - 1) arr_Lagrange[i] += ret[i] * val;
        //val là giá trị y_pos / w'(x_pos)
        // ret[i] là hệ số x^i của w_{n+1}(x) / (x - x_i)
        //arr_Lagrange[i] là hệ số x^i của đa thức nội suy Lagrange

        return ;
}
```

4.7 Gói Init() $\longrightarrow$ void

Đây là gói để khởi tạo $w'_{n+1}(x_i)$ và tính hệ số của đa thức nội suy Lagrange.

- **Đầu vào:** Không có đầu vào.
- **Đầu ra:** Không trả về giá trị hay kết quả gì (chỉ tính toán trực tiếp).

Code:

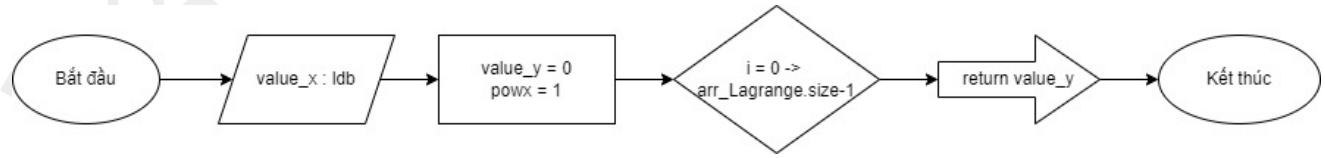
```cpp
void Init() {

    sort (a, a + n + 1);

    cout << "Sap xep theo cac moc noi suy tang dan\n";
    FOR(i, 0, n) cout << a[i].x << " " << a[i].y << "\n"; cout << "\n";


    // Tính dw[i] = w'(x_i)
    FOR(i, 0, n) {
            dw[i] = 1;
            FOR(j, 0, n) if (i != j) dw[i] *= (a[i].x - a[j].x);
    }

    // arr_wi là hệ số của (x-x_1)(x-x_2)...(x-x_n)
    FOR(i, 0, n) arr_wi = hoocnerNhan(arr_wi, 1, -a[i].x);


    // arr_Lagrange chứa hệ số của đa thức nội suy Lagrange
    arr_Lagrange.resize(arr_wi.size() - 1);
    FOR(i, 0, n) {
            Get_Lagrange(arr_wi, i);
            // Cập nhật y_i.L_i(x) cho arr_Lagrange
    }

    Print(arr_Lagrange);
}
```

4.8 Gói Get-Value-y(value-x : ldb) $\longrightarrow$ ldb

Đây là gói tính giá trị nội suy tại x = value-x.

- **Đầu vào:** Mốc nội suy cần tính value-x.
- **Đầu ra:** Giá trị nội suy tại x = value-x.

Sơ đồ thuật toán:



Code:

```
ldb Get_Value_y(ldb value_x) {
        ldb value_y = 0;

           ldb powx = 1; // lũy thừa của value_x
           FOR(i, 0, arr_Lagrange.size() - 1) {
                   value_y += arr_Lagrange[i] * powx;
                   powx *= value_x;
           }

           return value_y;
}
```

5 Hướng dẫn chạy code

Code nhập và in ra file nên ta sẽ tạo trước file nsLagrange.inp và nsLagrange.out.

5.1 Nhập input

- Nhập số nguyên n - số lượng mốc nội suy đã cho.
- Nhập lần lượt n bộ điểm $x_i y_i, (x_i \neq x_j \forall i \neq j)$.
- Nhập số nguyên m - số lượng mốc nội suy cần tính giá trị nội suy.
- Nhập m mốc nội suy cần tính.

5.2 Chạy code

Từng phần code đã được nêu như trên.

5.3 Đọc output

- Hiển thị lại bộ điểm theo sắp xếp theo mốc nội suy tăng dần.
- Hiển thị hệ số của đa thức nội suy Lagrange đã tính được (bắt đầu từ hệ số tự do).
- **"Gia tri da thuc ns Lagrange tai $x = x_0$ bang y_0"** là hiển thị giá trị đa thức nội suy Lagrange y_0 tại mốc nội suy x_0.

6 Hệ thống ví dụ

Ví dụ 1

Ta tạo input từ hàm $f(x) = x^{\frac{2}{3}} + \frac{1}{2x}$

1.000	1.5000000000000000
1.125	1.5261316221750008
1.250	1.5603972084031947
1.375	1.6001582244485388
1.500	1.6437040304377815
1.625	1.6898860111120257
1.750	1.7379107191052117
1.875	1.7872171655600166

Cần tính giá trị tại:1.6

Kết quả mong đợi: Giá trị chính xác tại $x = 1.6 = 1.6804807573413576$.

Chạy thuật toán nội suy Lagrange :

```
Sap xep theo cac moc noi suy tang dan
1.0000000000 1.5000000000
1.1250000000 1.5261316222
1.2500000000 1.5603972084
1.3750000000 1.6001582244
1.5000000000 1.6437040304
1.6250000000 1.6898860111
1.7500000000 1.7379107191
1.8750000000 1.7872171656

In cac he so tu x^0 den x^7
3.0261732179 -6.0605071673 9.8110550860 -8.8758540727 5.0583900302
-1.7858356610 0.3577302991 -0.0311517322

Gia tri da thuc ns Lagrange tai x = 1.0000000000 bang 1.5000000000
Gia tri da thuc ns Lagrange tai x = 1.1250000000 bang 1.5261316222
Gia tri da thuc ns Lagrange tai x = 1.2500000000 bang 1.5603972084
Gia tri da thuc ns Lagrange tai x = 1.3750000000 bang 1.6001582244
Gia tri da thuc ns Lagrange tai x = 1.5000000000 bang 1.6437040304
Gia tri da thuc ns Lagrange tai x = 1.6250000000 bang 1.6898860111
Gia tri da thuc ns Lagrange tai x = 1.7500000000 bang 1.7379107191
Gia tri da thuc ns Lagrange tai x = 1.8750000000 bang 1.7872171656
Gia tri da thuc ns Lagrange tai x = 1.6000000000 bang 1.6804808027
```

Kiểm tra sai số: $|1.6804807573413576 - 1.6804808027| = 4.53586424e\text{-}8$

Nhận xét:

- Sai số rất nhỏ nên ta có thể bỏ qua được.
- Đây là sai số do công kiểu dữ liệu và do công thức tính toán của phương pháp nội suy Lagrange.

Ví dụ 2

Ta có bảng nhiệt độ trái đất các năm

1880	-0.166
1885	-0.328
1890	-0.348
1895	-0.227
1900	-0.08
1905	-0.263
1910	-0.436
1915	-0.145
1920	-0.276
1925	-0.223
1930	-0.158
1935	-0.199
1940	0.124
1945	0.09
1950	-0.174
1955	-0.141
1960	-0.025
1965	-0.107
1970	0.025
1975	-0.013
1980	0.259
1985	0.119
1990	0.45
1995	0.446
2000	0.393
2005	0.675
2010	0.723
2015	0.894
2020	1.021

Cần tính giá trị tại: 2012

Kết quả mong đợi: Giá trị trong xấp xỉ trong khoảng $(0.723, 0.894)$

Chạy thuật toán nội suy Lagrange :

```
Gia tri da thuc ns Lagrange tai x = 2012.0000000000 bang
2484976769701612729460894426122655105024.0000000000
```

Nhận xét Phép tính hệ số đa thức nội suy Lagrange ra kết quả lớn dẫn đến hiện tượng tràn số kiểu dữ liệu **long double** trong **C++** dẫn đến code không chạy được như mong muốn.

7 Khối lượng tính toán của nội suy Lagrange

Khối lượng tính toán của nội suy Lagrange : $3.n^2$

8 Đánh giá

8.1 Ưu điểm:

- Phương pháp nội suy Lagrange không quá phức tạp về tính toán, chỉ sử dụng phép tính cộng trừ nhân chia.

8.2 Nhược điểm:

- Sai số của đa thức nội suy phụ thuộc vào các mốc nội suy. Do đó ta không thể khống chế được sai số như mong muốn. Thậm chí, sai số có thể tiến ra vô cùng.
- Khi bổ sung thêm mốc nội suy thì ta phải tính toán lại từ đầu chứ không dùng được kết quả đã tính toán.

9 Tài liệu tham khảo

[1] *Bài giảng Giải tích số*, TS Hà Thị Ngọc Yến, Viện Toán ứng dụng và Tin học, Đại học Bách Khoa Hà Nội, 2018

[2] *Giáo trình Giải tích số*, Lê Trọng Vinh, Nhà xuất bản Khoa học và Kỹ thuật, 2007