Lab2 Hermite Interpolation

Sheng Zhang
2nd class of Information and computing science
2016310200210

1 Lab Purpose

- Master theory of Hermite interpolation.
- Implementation of Hermite interpolation using C language.

2 Lab Requirements

Given n+1 t_i, s_i and v_i(i=0,1,...,n), using Hermite interpolation to construct polynomial which meets the conditions that $H_{2n+1}(t_i) = s_i$ and $H'_{2n+1}(t_i) = v_i$. Try to calculate the distance and speed for given m time points.

2.1 Definition of Function interface

```
void Hermite_Interpolation(int N, double t[], double s[], double v[],
    int m, double ht[], double hs[], double hv[]);
```

The original code is as follow:

```
#include<stdio.h>
#define MAXN 5
#define MAXM 10
void Hermite_Interpolation( int N, double t[], double s[], double v[],
    int m, double ht[], double hs[], double hv[] );
int main()
{
  int N, m;
  double t[MAXN], s[MAXN], v[MAXN];
  double ht[MAXM], hs[MAXM], hv[MAXM];
  int i;
  while ( scanf("%d", &N) != EOF ) {
    for ( i=0; i<N; i++ )
      scanf("%lf", &t[i]);
    for ( i=0; i<N; i++ )
      scanf("%lf", &s[i]);
    for ( i=0; i<N; i++ )
      scanf("%lf", &v[i]);
    scanf("%d", &m);
    for ( i=0; i<m; i++ )
      scanf("%lf", &ht[i]);
    Hermite_Interpolation( N, t, s, v, m, ht, hs, hv );
    for ( i=0; i<m; i++ )
      printf("%.4lf ", hs[i]);
    printf("\n");
    for ( i=0; i<m; i++ )
      printf("%.4lf ", hv[i]);
    printf("\n\n");
  }
  return 0;
}
```

2.2 Data requirements

input data:
2
0.0 1.0
0.0 1.0
0.0 0.0
5
0.0 0.2 0.5 0.8 1.0

3
0.0 0.5 1.0
100.0 170.0 200.0
30.0 150.0 0.0
5
0.0 0.25 0.5 0.75 1.0

5
0.0 1.0 2.0 3.0 4.0
0.0 60.0 160.0 260.0 300.0
5.0 70.0 100.0 120.0 20.0
10
0.5 1.0 1.5 2.0 2.5 3.0 3.5 3.8 3.95 4.0

output data:
0.0000 0.1040 0.5000 0.8960 1.0000
0.0000 0.9600 1.5000 0.9600 0.0000

100.0000 127.9297 170.0000 195.9766 200.0000
30.0000 165.4688 150.0000 52.9688 0.0000

30.2222 60.0000 105.9303 160.0000 206.3438 260.0000 307.9764 305.7687 299.9796 300.0000
62.6024 70.0000 109.0488 100.0000 92.9745 120.0000 41.2374 -44.8421 -16.2783 20.0000

3 Method

There is the detail of Hermite interpolation, and some known expressions are given.

$$H_{2n+1}(x) = \sum_{i=0}^{n} y_i * h_i(x) + \sum_{i=0}^{n} y_i^{'} * \hat{h}_i(x) \tag{1}$$

$$h_i(x) = [1 - 2\hat{l}_i(x_i)(x - x_i)]l_i^2 \tag{2}$$

$$l_i(x) = \prod_{\substack{j \neq i \\ j=0}}^{n} \frac{x - x_j}{x_i - x_j} \tag{3}$$

$$\hat{h}_i(x) = (x - x_i)l_i^2(x) \tag{4}$$

$$l_i^{'}(x) = [\sum_{\substack{j \neq i \\ j=0}}^{n} \frac{1}{x - x_j}]l_i(x) \tag{5}$$

$$l_i^{'}(x_i) = \sum_{\substack{j \neq i \\ j=0}}^{n} \frac{1}{x_i - x_j} \tag{6}$$

Here, x_i are interpolated points, y_i are the value of interpolated of points and they're already known. So utilizing these conditions, we can get final expression of H_{2n+1}.

As for $H^{'}_{2n+1}(x)$, we can easily get the expression after some inferences.

$$\begin{aligned} H^{'}_{2n+1}(x) &= \sum_{i=0}^{n} y_i * h^{'}_i(x) + \sum_{i=0}^{n} y^{'}_i * \hat{h}^{'}_i(x) \\ &= 2y_i \sum_{i=0}^{n} \{[\sum_{\substack{j\neq i \\ j=0}}^{n} \frac{1}{x - x_j}] l_i^2(x)[1 - 2(\sum_{\substack{j\neq i \\ j=0}}^{n} \frac{1}{x_i - x_j})] - \sum_{\substack{j\neq i \\ j=0}}^{n} \frac{1}{x_i - x_j} l_i^2(x)\} + y^{'}_i \sum_{i=0}^{n} \hat{h}^{'}_i(x) \end{aligned} \tag{7}$$

For convenience, I define $\sum_{\substack{j\neq i \\ j=0}}^{n} \frac{1}{x-x_j}$ and $\sum_{\substack{j\neq i \\ j=0}}^{n} \frac{1}{x_i-x_j}$ as (*1) and (*2) respectively. So the $H_{2n+1}(x)$ and $H^{'}_{2n+1}(x)$ can easily represented by (*1) and (*2). So once we understand how to implement these two expressions, the rest of parts is very straightforward.

```
double get_star1(double x, int i, double t[], int N)
{
    int j;
    double result = 0;
    for(j = 0; j<N; j++)
    {
        double xj = t[j];
        if(j == i || xj == x)
            continue;
        result += 1/(x-xj);
    }
    return result;
}

double get_star2(int i, double t[], int N)
{
    double result = 0;
    double xi = t[i];
    int j;
    for(j = 0; j < N; j++)
    {
        double xj = t[j];
        if(j == i || xi == xj)
            continue;
        result += 1/(xi-xj);
    }
    return result;
}
```

Also, I will show my all code in the appendix, so teacher you can check the whole code.

4 Thinking

C language is a fundamental programming language, indeed, it trains our algorithm mind. Not only when in this experiment, but also when I am trying to do some programs, I always can feel the importance of algorithm implement by ourselves. In my opinion, math is important, programming is also important. It doesn't matter that which is more important, because if you lose math mind, you will not be creative in algorithm, and if you aren't good at programming, you cannot make your good idea come true.
Okay, back to our theme. In this lab experiment, I compute out $H(x)$ and $H^{'}(x)$, then I split the expressions into several parts. And then I implement these parts respectively. Naturally, if there is no mistakes in programming, the results of the code will be right.

5 Appendix

```
#include<stdio.h>
#include<math.h>
#define MAXN 5
#define MAXM 10

double get_li(double x, int i, double t[], int N)
{
    double xi = t[i];
    double result = 1;
    int j;
    for(j = 0; j < N; j++)
    {
        if(j == i)
            continue;
        double xj = t[j];
        result *= (x-xj)/(xi-xj);
    }
    return result;
}

double get_star1(double x, int i, double t[], int N)
{
    int j;
    double result = 0;
    for(j = 0; j<N; j++)
    {
        double xj = t[j];
        if(j == i || xj == x)
            continue;
        result += 1/(x-xj);
    }
    return result;
}

double get_star2(int i, double t[], int N)
{
    double result = 0;
    double xi = t[i];
    int j;
    for(j = 0; j < N; j++)
    {
        double xj = t[j];
        if(j == i || xi == xj)
            continue;
        result += 1/(xi-xj);
    }
    return result;
}


double get_hi_x(double x, int i, double t[], int N)
{
    double xi = t[i];
    double li_differential = get_star2(i, t, N);
    double li_square = pow(get_li(x, i, t, N),2);
    double hi_x = (1-2*li_differential*(x-xi)) * li_square;
    return hi_x;
}

double get_hi_x_hat(double x, int i, double t[], int N)
{
```

```
    double li_square = pow(get_li(x, i, t, N),2);
    double xi = t[i];
    double hi_x_hat = (x-xi)*li_square;
    return hi_x_hat;
}

double get_Hx(double x, double t[], double s[], double v[], int N)
{
    int i;
    double first = 0.0;
    double second = 0.0;
    for(i = 0; i < N; i++)
    {
        double yi = s[i];
        double yi_differential = v[i];
        first += yi * get_hi_x(x, i, t, N);
        second += yi_differential * get_hi_x_hat(x, i, t, N);
    }
    return first + second;
}


double get_Hx_differential(double x, double t[], double s[], double v
    [], int N)
{
    int i;
    double first = 0;
    double second = 0;
    double result = 0;
    for(i =0; i < N; i++)
    {
        double xi = t[i];
        double yi = s[i];
        double yi_differential = v[i];
        double li_square = pow(get_li(x, i, t, N),2);
        first = 2 * yi * li_square * (get_star1(x, i, t, N) * (1-2*
            get_star2(i, t, N)*(x-xi)) - get_star2(i, t, N));
        second = yi_differential * li_square * (1+2*get_star1(x, i, t,
            N)*(x-xi));
        result += first + second;
    }
    return result;
}
void Hermite_Interpolation( int N, double t[], double s[], double v[],
    int m, double ht[], double hs[], double hv[] )
{
    int i;
    for(i = 0; i < m; i++)
    {
        double x = ht[i];
        hs[i] = get_Hx(x, t, s, v, N);
        hv[i] = get_Hx_differential(x, t, s, v, N);
    }
}

int main()
{
  int N, m;
  double t[MAXN], s[MAXN], v[MAXN];
  double ht[MAXM], hs[MAXM], hv[MAXM];
  int i;
  while ( scanf("%d", &N) != EOF ) {
    for ( i=0; i<N; i++ )
```

```
            scanf("%lf", &t[i]);
        for ( i=0; i<N; i++ )
            scanf("%lf", &s[i]);
        for ( i=0; i<N; i++ )
            scanf("%lf", &v[i]);
        scanf("%d", &m);
        for ( i=0; i<m; i++ )
          scanf("%lf", &ht[i]);
        Hermite_Interpolation( N, t, s, v, m, ht, hs, hv );
        for ( i=0; i<m; i++ )
          printf("%.4lf ", hs[i]);
        printf("\n");
        for ( i=0; i<m; i++ )
          printf("%4lf ", hv[i]);
        printf("\n\n");
    }
    return 0;
}
```