计算方法Project1报告

PB19051035周佳豪

问题描述

根据每个1920~1970年的人口表，分别使用$Lagrange$插值与$Newton$插值估计其他年份的人口，并判断$Lagrangee$插值的准确性。

> Input:1920~1970的人口
>
> output:估计1910等年份的人口数

算法设计

- $Lagrange$插值

  根据公式$l_i(x)=\prod_{0\le j\le n,j\neq i}\frac{x-x_j}{x_i-x_j}, L_n(x)=\sum_{i=0}^{n}l_i(x)f(x_i)$进行估计

- $Lagrange$插值的准确性估计

  将x_0改为1910，$f(x_0)$改为91772，仿照上式构造$\widetilde{L_n(x)}$

  准确性公式为$\frac{x-x_0}{x_0-x_{n+1}}(L_n(x)-\widetilde{L_n(x)})$,其中$x_0=1920, x_{n+1}=1910$

- $Newton$插值
  - 先得到差商表，即$g_k,k=0,1,\dots,n$，g_k表示$f[x_0,x_1,\dots,x_k]$

    根据递推式$f[x_0,x_1,\dots,x_n]=\frac{f[x_1,x_2,\dots,x_n]-f[x_0,\dots,x_{n-1}]}{x_n-x_0}$即可求得$g_k$

    具体做法是根据递推式求得所有的$f[x_i,x_{i+1},\dots,x_j],i\le j$，再把相应的值赋给$g_k$
  - 根据公式$N(x)=f[x_0]+\sum_{k=1}^{n}f[x_0,x_1,\dots,x_k](x-x_0)(x-x_1)\dots(x-x_k)$估计不同年份的人口

实验结果

```
----test Lagrange interpolation:----
1910: 31872
1965: 193082
2002: 26138.7
---test accuracy:---
1965:-1091.93
2002:2.0764e+006
----test Newton interpolation:----
1965: 193082
2012: -136453
```

- $Lagrange$插值估计1910年人口数为31872千人，1965年人口数为193082千人，2002年人口数为26138.7千人。1965、2002年的人口数据准确性分别为-1091.93，2076400
- $Newton$插值估计1965年人口数为193082千人，2012年人口数为-136453千人

结果分析

- 发现在1920~1970年之间的估计较为准确，在此之外的估计有很大误差。原因是插值函数是多项式，只能很好的拟合插值点附近的区域，对离插值点较远的区域拟合结果很差。
- $Newton$插值和$Lagrange$插值对相同年份人数的估计值一样，根据算法得到的插值函数相同。

附录

```
#include <iostream>
#include <map>
#include <algorithm>
using namespace std;

class Solution
{
public:
    Solution()
    {
        f[1920] = 105711;
        f[1930] = 123203;
        f[1940] = 131669;
        f[1950] = 150697;
        f[1960] = 179323;
        f[1970] = 203212;
        n = 5;
        x = new double[n + 1];
        x[0] = 1920;
        x[1] = 1930;
        x[2] = 1940;
        x[3] = 1950;
        x[4] = 1960;
        x[5] = 1970;
        g = new double[n + 1];
        get_g();
    }
    double L(int year)
    {
        double result = 0;

        for (int i = 0; i <= n; i++)
        {
            double temp = 1;
            for (int j = 0; j <= i - 1; j++)
            {
                // cout<<"year-x[j]="<<year-x[j]<<endl;
                //  cout<<"x[i]-x[j]="<<x[i]-x[j]<<endl;
                temp *= (year - x[j]) / (x[i] - x[j]);
            }

            for (int j = i + 1; j <= n; j++)
            {
                // cout<<"year-x[j]="<<year-x[j]<<endl;
                // cout<<"x[i]-x[j]="<<x[i]-x[j]<<endl;
                temp *= (year - x[j]) / (x[i] - x[j]);
            }

            /*
```

```
            cout<<"temp = "<<temp<<endl;
            cout<<"f[x[i]] = "<<f[x[i]]<<endl;
            cout<<"temp*f[x[i]] = "<<temp*f[x[i]]<<endl;*/
            result += temp * f[x[i]];
        }
        return result;
    }
    double _L(int year)
    {
        x[0] = 1910;
        f[1910] = 91772;
        double result = L(year);
        x[0] = 1920;
        return result;
    }
    //得到准确性
    double get_error(int year)
    {
        return (year - 1920) / (1920 - 1910) * (L(year) - _L(year));
    }
    //得到函数g
    void get_g()
    {
        map<string, double> temp;
        for (int i = 0; i <= n; i++)
        {
            temp[to_string(i)] = f[x[i]];
            // cout<<"temp["<<i<<"] = "<<temp[to_string(i)]<<endl;
        }
        for (int i = 2; i <= n + 1; i++)
        {
            for (int j = 0; j <= n + 1 - i; j++)
            {
                string s, s1, s2;
                for (int k = j; k < j + i; k++)
                {
                    s += to_string(k);
                }
                for (int k = j + 1; k < j + i; k++)
                    s1 += to_string(k);
                for (int k = j; k < j + i - 1; k++)
                    s2 += to_string(k);
                temp[s] = (temp[s1] - temp[s2]) / (x[j + i - 1] - x[j]);

                // cout<<"temp["<<s<<"] = "<<temp[s]<<"=(temp["<<s1<<"]-temp["
<<s2<<"])/("<<x[j+i-1]<<"-"<<x[j]<<")"<<endl;
            }
        }
        string str;
        for (int i = 0; i <= n; i++)
        {
            str += to_string(i);
            g[i] = temp[str];
        }
    }
    double N(int year)
    {
        get_g();
```

```
        double t = 1, newton = g[0];
        for (int k = 1; k <= n; k++)
        {
            t *= (year - x[k - 1]);
            newton += t * g[k];
        }
        return newton;
    }

private:
    double *x;
    double *g;
    map<int, double> f;
    int n;
};

int main()
{
    Solution s;
    // Lagrange interpolation
    cout << "----test Lagrange interpolation:----" << endl;
    cout << "1910: " << s.L(1910) << endl;
    cout << "1965: " << s.L(1965) << endl;
    cout << "2002: " << s.L(2002) << endl;
    cout << "---test accuracy:---" << endl;
    cout << "1965:" << s.get_error(1965) << endl;
    cout << "2002:" << s.get_error(2002) << endl;
    // Newton interpolation
    cout << "----test Newton interpolation:----" << endl;
    cout << "1965: " << s.N(1965) << endl;
    cout << "2012: " << s.N(2012) << endl;

    return 0;
}
```