CompPhys Assignment 04

李尚坤　物理学系　20307130215

1　插值法

1.1　题目描述

Newton interpolation of:

(i) 10 equal spacing points of $\cos x$ within $[0, \pi]$,

(ii) 10 equal spacing points $\frac{1}{1+25x^2}$ within [-1,1].

Compare the results with the cubic spline interpolation.

1.2　解决方案描述

1.2.1　Newton Interpolation

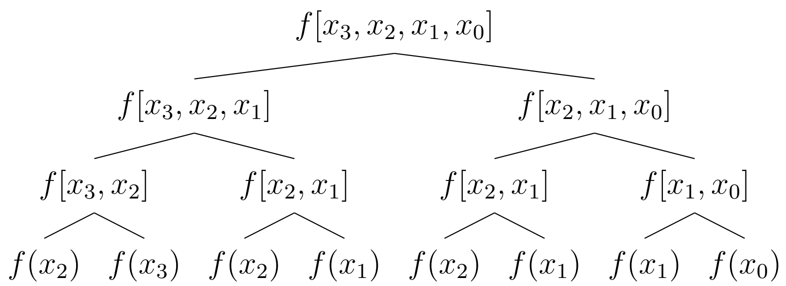
由牛顿插值法的一般公式：

$$\begin{aligned} f_n(x) =& f(x_0) + (x-x_0)f[x_1,x_0] + (x-x_0)(x-x_1)f[x_2,x_1,x_0] \\ &+ \cdots + (x-x_0)(x-x_1)\cdots(x-x_{n-1})f[x_n,x_{n-1},\cdots,x_0] \end{aligned} \tag{1}$$

其中 $(x_0, x_1, \cdots, x_{n-1})$ 与 $(f(x_0), f(x_1), \cdots, f(x_{n-1}))$ 为输入的样本值。$f[x_n, x_{n-1}, \cdots, x_0]$ 为定义的 Bracket Function，它可通过如下递推关系求得：

$$f[x_n, x_{n-1}, \cdots, x_0] = \frac{f[x_n, x_{n-1}, \cdots, x_1] - f[x_{n-1}, x_{n-2}, \cdots, x_0]}{x_n - x_0} \tag{2}$$

因此，只要计算出了 (1) 式中的 Bracket Function 即可。通过上面的递推公式，我们很容易想到利用递推的方法进行求解，我们以 4 个样本点为例来说明：



1. 如上树状图所示，$f[x_3, x_2, x_1, x_0]$ 可以由 $f[x_3, x_2, x_1]$ 与 $f[x_2, x_1, x_0]$ 计算得到；
2. 而要计算 $f[x_3, x_2, x_1]$，则可以再进行一次递归，它可以由 $f[x_3, x_2]$ 与 $f[x_2, x_1]$ 计算得到；
3. 对于 $f[x_3, x_2]$ 与 $f[x_2, x_1]$，它们可以直接由 Bracket Function 的定义计算得到
4. 当递归进行到了最底层之后，再依次返回每一层递归的结果，最终就可以计算得到 $f[x_3, x_2, x_1, x_0]$

当求得需要的 Bracket Function 之后，将其与对应的多项式相乘之后再求和，就是我们所求的函数。

1.2.2 Cubic Spline Interpolation

由三阶样条插值法的一般公式，对于样本值 $(x_1, x_2, \cdots, x_n)$ 与 $(f(x_1), f(x_2), \cdots, f(x_n))$，在区间 $[x_{i-1}, x_i]$ 中，其插值所得的函数表达式为：

$$\begin{aligned} f_{i-1}(x) =& \frac{f''(x_{i-1})}{6(x_i - x_{i-1})}(x_i - x)^3 + \frac{f''(x_i)}{6(x_i - x_{i-1})}(x - x_{i-1})^3 \\ &+ \left[\frac{f(x_{i-1})}{x_i - x_{i-1}} - \frac{f''(x_{i-1})(x_i - x_{i-1})}{6}(x_i - x)\right] \\ &+ \left[\frac{f(x_i)}{x_i - x_{i-1}} - \frac{f''(x_i)(x_i - x_{i-1})}{6}(x - x_{i-1})\right] \end{aligned} \tag{3}$$

因此，我们仅需求出这 n 个样本点处的二阶导数值，就可以得到每一区间的插值函数表达式。由一阶导函数连续可得：

$$\begin{aligned} &(x_i - x_{i-1})f''(x_{i-1}) + 2(x_{i+1} - x_{i-1})f''(x_i) + (x_{i+1} - x_i)f''(x_{i+1}) \\ &= \frac{6}{x_{i+1} - x_i}\left[f(x_{i+1} - f(x_i)\right] + \frac{6}{x_i - x_{i-1}}\left[f(x_{i-1} - f(x_i)\right] \end{aligned} \tag{4}$$

由上面的 n-2 个方程加上 $f''(x_1) = f''(x_n) = 0$ 这两个方程，就可以解出所有样本点处二阶导数值。这样一个 n 元 1 次方程组的系数矩阵是一个 n×n 的“三对角矩阵”，手工运算可以使用 Thomas Algorithm，此处直接调用库求解即可。

1.3 伪代码

Algorithm 1: Newton Interpolation

```
Input: n sample points (x_0, x_1, ..., x_{n-1}) and (f_0, f_1, ..., f_{n-1})
Output: The polynomial function f(x)
def Bracket_Function(x[n],f[n]):            \\Obtain f[x_n, x_{n-1}, ..., x_1, x_0]
    if LEN(x)>2 and LEN(y)>2 then
        return (Bracket_Function(x[1:n],f[1:n]) - Bracket_Function(x[0,n-1],y[0,n-1])) / (x[-1]-x[0])
    else
        return (f[1]-f[0]) / (x[1]-x[0])
    end
end
f(x) <- f_0 + sum_{i=0}^{n-1} (x - x_0)...(x - x_i) Bracket_Function(x[0 : i+1], f[0 : i+1])
PLOT(f(x))
return f(x)
```

Algorithm 2: Cubic Spline Interpolation

```
Input: n sample points (x_1, x_2, ..., x_n) and (f_1, f_2, ..., f_n)
Output: The interpolation function f(x)
for i <- 2 to n-1 do
    Matrix[i, i-1] <- x_i - x_{i-1}
    Matrix[i, i] <- 2(x_{i+1} - x_{i-1})
    Matrix[i, i+1] <- x_{i+1} - x_i
    Vector[i] <- 6/(x_{i+1} - x_i) [f_(x_{i+1}) - f(x_i)] + 6/(x_i - x_{i-1}) [f_(x_{i-1}) - f(x_i)]
end
Matrix[1,1], Matrix[n,n] <- 1
Vector[1], Vector[n] <- 0
f'' = Matrix^{-1} Vector
f(x) <- ∪_{i=2}^{n} {f_{i-1}(x)}   // the definition of f_{i-1}(x) is given in eq(3)
PLOT(f(x))
return f(x)
```

1.4 输入输出示例

分别以 $\cos x$ 和 $\frac{1}{1+25x^2}$ 的 10 个样本点为输入（在程序中生成），分别绘制出通过 Newton Interpolation（红线）和 Cubic Spline Interpolation（蓝线）计算得到的函数图像。

1. 对于函数 $\cos x$，函数图像如图 1 所示
2. 对于函数 $\frac{1}{1+25x^2}$，函数图像如图 2 所示

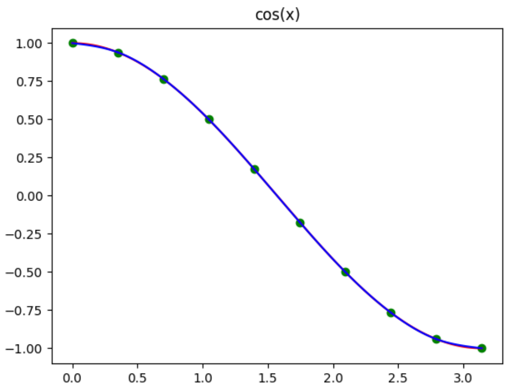


图 1: $\cos x$

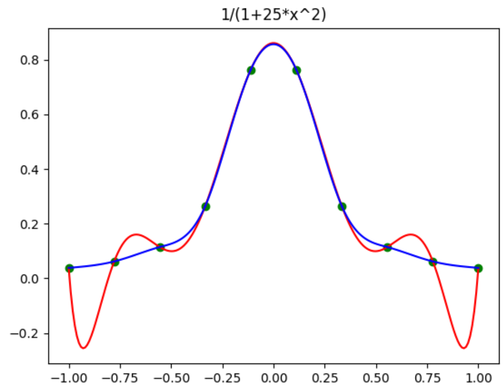


图 2: $\frac{1}{1+25x^2}$

从第一张图中可以看出，Newton Interpolation 以及 Cubic Spline Interpolation 均适用于函数 $\cos x$

从第二张图中可以看出，Newton Interpolation 不适用于函数 $\frac{1}{1+25x^2}$，而 Cubic Spline Interpolation 适用于函数 $\frac{1}{1+25x^2}$。

1.5 用户手册

1. 本程序的源程序为 Interpolation.py
2. 在执行源程序之前，应当先安装 numpy 以及 Matplotlib 库
3. 本程序分别利用 Newton Interpolation 以及 Cubic Spline Interpolation 对函数 $\cos x$ 与 $\frac{1}{1+25x^2}$ 中的 10 个样本点进行插值计算
4. 运行程序后，将首先展示函数 $\cos x$ 的插值计算结果，其中绿色点为样本点，红色线为 Newton Interpolation 计算结果，蓝色线为 Cubic Spline Interpolation 计算结果
5. 关闭这一窗口后，将展示函数 $\frac{1}{1+25x^2}$ 的插值计算结果，其中绿色点为样本点，红色线为 Newton Interpolation 计算结果，蓝色线为 Cubic Spline Interpolation 计算结果

2 最小二乘法

2.1 题目描述

The table below gives the temperature T along a metal rod whose ends are kept at fixed constant temperature. The temperature is a function of the distance x along the rod.

(1)Compute a least-squares, straight-line fit to these data using $T(x) = a + bx$

(2)Compute a least-squares, parabolic-line fit to these data using $T(x) = a + bx + cx^2$

x/cm	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0
T/°C	14.6	18.5	36.6	30.8	59.2	60.1	62.2	79.4	99.9

表 1: $x-T$ 数据表

2.2 解决方案描述

(1) 对于线性拟合而言:
题目中需要进行拟合的方程为:

$$T(x) = a + bx \tag{5}$$

总的误差函数:

$$\begin{aligned}\chi^2 &= \sum_{i=1}^{9}[y(x_i) - y_i]^2 \\ &= \sum_{i=1}^{4}[a + bx_i - y_i]^2\end{aligned} \tag{6}$$

若要总的误差最小，则应当有:

$$\frac{\partial\chi^2}{\partial a} = \frac{\partial\chi^2}{\partial b} = 0 \tag{7}$$

由此我们可以得到关于 a, b 的一个一元二次方程组:

$$\begin{cases} 9a + \left(\sum\limits_{i=1}^{9} x_i\right) b = \left(\sum\limits_{i=1}^{9} y_i\right) \\ \left(\sum\limits_{i=1}^{9} x_i\right) a + \left(\sum\limits_{i=1}^{9} x_i^2\right) b = \left(\sum\limits_{i=1}^{9} x_i y_i\right) \end{cases} \tag{8}$$

我们将它化为矩阵形式:

$$\begin{bmatrix} 9 & \sum\limits_{i=1}^{9} x_i \\ \sum\limits_{i=1}^{9} x_i & \sum\limits_{i=1}^{9} x_i^2 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum\limits_{i=1}^{9} y_i \\ \sum\limits_{i=1}^{9} x_i y_i \end{bmatrix} \tag{9}$$

随后求解这个方程，即可得到 a, b 的值。

(2) 对于二次拟合而言，步骤与上面类似，我们最终也可以得到一个一元三次方程组，将其写成矩阵形式如下:

$$\begin{bmatrix} 9 & \sum\limits_{i=1}^{9} x_i & \sum\limits_{i=1}^{9} x_i^2 \\ \sum\limits_{i=1}^{9} x_i & \sum\limits_{i=1}^{9} x_i^2 & \sum\limits_{i=1}^{9} x_i^3 \\ \sum\limits_{i=1}^{9} x_i^2 & \sum\limits_{i=1}^{9} x_i^3 & \sum\limits_{i=1}^{9} x_i^4 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \sum\limits_{i=1}^{9} y_i \\ \sum\limits_{i=1}^{9} x_i y_i \\ \sum\limits_{i=1}^{9} x_i^2 y_i \end{bmatrix} \tag{10}$$

同样求解这个方程即可得到 a, b, c 的值。

2.3 伪代码

Algorithm 3: Least-Squares Straight-line Fitting

Input: The initial data of $\mathbf{x}$ and $\mathbf{T}$

Output: The number a and b

$Matrix[1][1] \leftarrow 9$, $Matrix[1][2] \leftarrow \sum_{i=1}^{9} x_i$

$Matrix[2][1] \leftarrow \sum_{i=1}^{9} x_i$, $Matrix[2][2] \leftarrow \sum_{i=1}^{9} x_i^2$

$Vector[1] \leftarrow \sum_{i=1}^{9} T_i$, $Vector[2] \leftarrow \sum_{i=1}^{9} x_i T_i$

$Solution \leftarrow Matrix^{-1} Vector$

$a \leftarrow Solution[1]$

$b \leftarrow Solution[2]$

return a, b

Algorithm 4: Least-Squares Parabolic-line Fitting

Input: The initial data of $\mathbf{x}$ and $\mathbf{T}$

Output: The number a, b, c

$Matrix[1][1] \leftarrow 9$, $Matrix[1][2] \leftarrow \sum_{i=1}^{9} x_i$, $Matrix[1][3] \leftarrow \sum_{i=1}^{9} x_i^2$

$Matrix[2][1] \leftarrow \sum_{i=1}^{9} x_i$, $Matrix[2][2] \leftarrow \sum_{i=1}^{9} x_i^2$, $Matrix[2][3] \leftarrow \sum_{i=1}^{9} x_i^3$

$Matrix[3][1] \leftarrow \sum_{i=1}^{9} x_i^2$, $Matrix[3][2] \leftarrow \sum_{i=1}^{9} x_i^3$, $Matrix[3][3] \leftarrow \sum_{i=1}^{9} x_i^4$

$Vector[1] \leftarrow \sum_{i=1}^{9} x_i$, $Vector[2] \leftarrow \sum_{i=1}^{9} x_i T_i$, $Vector[3] \leftarrow \sum_{i=1}^{9} x_i^2 T_i$

$Solution \leftarrow Matrix^{-1} Vector$

$a \leftarrow Solution[1]$

$b \leftarrow Solution[2]$

$c \leftarrow Solution[3]$

return a, b, c

2.4 输入/输出示例

1. 当程序运行后，对于本题中的输入输出，最终结果如下表:

Straight-line Fit		Parabolic-line Fit		
a/°C	b/°C·cm^{-1}	a/°C	b/°C·cm^{-1}	c/°C·cm^{-2}
0.888889	10.073333	8.261905	6.051688	0.402165

表 2: 输出数据

2. 程序在终端运行时的输出结果如下：

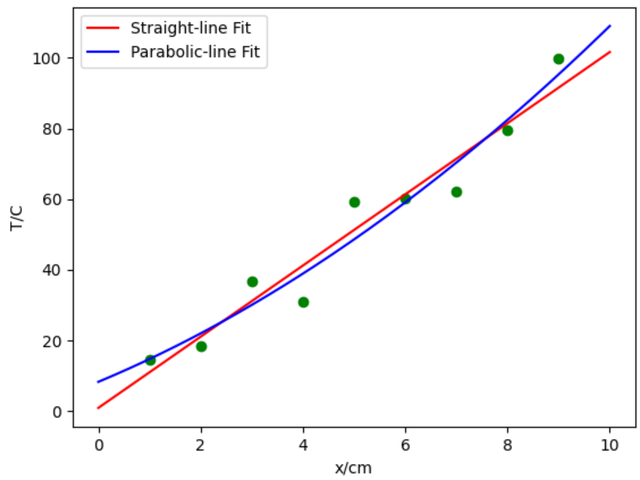


图 3: 拟合结果图

```
Straight-line Fit: T=a+bx    a= 0.8888888888888637 b= 10.073333333333334
Parabolic-line Fit: T=a+bx+cx^2    a= 8.26190476190525 b= 6.051688311688366 c= 0.4021645021644897
```

图 4: a, b 拟合值

2.5 用户手册

1. 本程序的源程序为 Least_square.py
2. 在执行源程序之前，应当先安装 numpy 以及 Matplotlib 库
3. 本程序分别利用最小二乘法对给出的数据进行函数拟合，通过求解一个线性方程组得到函数中的参数值
4. 运行程序后，在终端将输出函数中的参数值
5. 同时将展示拟合所得的函数图像