计算物理作业 3

谢昀城 22307110070

2024 年 10 月 25 日

1 题目 1：牛顿法插值和三次样条插值

1.1 题目描述

Newton interpolation of:

1. 10 equal spacing points$cos(x), x\epsilon[0,\pi]$
2. 10 equal spacing points$\frac{1}{(1+25x^2)}, x\epsilon[-1,1]$

Compare the results with the cubic spline interpolation.

1.2 程序描述

本程序中，我们将首先使用递归的方法计算 $f[x_0,...x_n] = \frac{f[x_1,...x_n]-f[x_0,...x_{n-1}]}{x_n-x_0}$, 得到 Newton 插值系数，并还原出插值函数，在程序中由 f,Newton_coefficients,Newton_interpolatefunc 三个函数实现。

对于三次样条差值，我们通过求解以下 n-2 个线性方程组得到 $f''(x_i)$，并还原出插值函数，于函数 splines_funciton 中实现，其中求解线性方程组的部分，由 homework3 中的 guaussian_elimination 函数，通过高斯消元法实现。

$$\begin{aligned}&(x_i-x_{i-1})\,f''(x_{i-1})+2\,(x_{i+1}-x_{i-1})\,f''(x_i)+(x_{i+1}-x_i)\,f''(x_{i+1})\\&\qquad=\frac{6}{x_{i+1}-x_i}\left[f(x_{i+1})-f(x_i)\right]+\frac{6}{x_i-x_{i-1}}\left[f(x_i)-f(x_{i-1})\right],\quad i=2,\ldots,n-1\end{aligned}$$

$$f''(x_1)=f''(x_n)=0$$

本程序源文件为 Interpolation.py，在终端进入当前目录，使用命令 python -u Interpolation.py 运行本程序。运行时请保证 Python 第三方库 Numpy,Matplotlib 已安装，并且与 gaussian_elimination.py 置于同一文件下。程序开发环境为 Python3.12.3，可在 Python3.8 以上版本中运行。

1.3 伪代码

1.3.1 牛顿法插值伪代码：

Algorithm 1 f

function F(xl, yl)
  INPUT: xl (array of x values), yl (array of y values)
  OUTPUT: f[xl](float)

if length(xl) = 1 **then**

return $yl[0]$

else

return $(f(xl[1:], yl[1:]) - f(xl[0:-1], yl[0:-1]))\,/\,(xl[-1] - xl[0])$

end if

end function

Algorithm 2 NewtonCoefficients

function NewtonCoefficients(xl, yl)

INPUT: xl (array of x values), yl (array of y values)

OUTPUT: coefficients (array of Newton coefficients)

return $[f(xl[0:n], yl[0:n])\ \ \text{for}\ \ n \leftarrow 1$ **To** $n]$

end function

Algorithm 3 NewtonInterpolateFunc

function NewtonInterpolateFunc(x, xl, cl)

INPUT: x (array of interpolation points), xl (array of x values), cl (array of coefficients)

OUTPUT: y (array of interpolated values)

$y \leftarrow$ **zeros**(x.shape)

for $i \leftarrow 1$ **To** length(cl) **do**

$y \leftarrow y + cl[i] \times \prod([x - xi\ \text{for}\ xi \in xl[0:i]])$

end for

return y

end function

1.3.2 三次样条插值伪代码：

Algorithm 4 SplinesFunction

function SplinesFunction(x, x_l, y_l)

INPUT: x (interpolation point), x_l (array of x values), y_l (array of y values)

OUTPUT: result (float)

$n \leftarrow$ length(x_l)

$A \leftarrow$ **zeros**($n-2, n-2$)

$B \leftarrow$ **zeros**($n-2$)

for $i \leftarrow 1$ **To** $n-1$ **do**

$A[i-1, i-1] \leftarrow 2 \cdot (x_{i+1} - x_{i-1})$

if $i < n-2$ **then**

$A[i-1, i] \leftarrow x_{i+1} - x_i$

end if

if $i > 1$ **then**

$A[i-1, i-2] \leftarrow x_i - x_{i-1}$

end if

if $i > 1$ **and** $i < n-2$ **then**

$B[i-1] \leftarrow 6 \cdot \left(\frac{y_{i+1}-y_i}{x_{i+1}-x_i}\right) + 6 \cdot \left(\frac{y_{i-1}-y_i}{x_i-x_{i-1}}\right)$

end if

end for

$B[0] \leftarrow 6 \cdot \left(\frac{y_2-y_1}{x_2-x_1}\right) + 6 \cdot \left(\frac{y_0-y_1}{x_1-x_0}\right)$

$B[-1] \leftarrow 6 \cdot \left(\frac{y_{n-1}-y_{n-2}}{x_{n-1}-x_{n-2}}\right) + 6 \cdot \left(\frac{y_{n-3}-y_{n-2}}{x_{n-2}-x_{n-3}}\right)$

$M \leftarrow$ **hstack**$(A, B.\text{reshape}(-1,1))$

$cl \leftarrow$ **hstack**$(0,\text{GaussianElimination}(M).0)$

for $i \leftarrow 1$ **To** n **do**

if $x_{i-1} \le x \le x_i$ **then**

return $cl[i-1] \cdot \frac{(x_i-x)^3}{6\cdot(x_i-x_{i-1})} + cl[i] \cdot \frac{(x-x_{i-1})^3}{6\cdot(x_i-x_{i-1})} + \left(\frac{y_{i-1}}{x_i-x_{i-1}} - cl[i-1] \cdot \frac{x_i-x_{i-1}}{6}\right) \cdot (x_i-x) + \left(\frac{y_i}{x_i-x_{i-1}} - cl[i] \cdot \frac{x_i-x_{i-1}}{6}\right) \cdot (x-x_{i-1})$

end if

end for

end function

1.4 输入输出实例

对于本程序，运行后将生成 $cos(x)$ 和 $\frac{1}{1+25x^2}$ 的插值结果为 Interpolation1.png 和 Interpolation2.png，于当前目录下, 如图1和2所示, 并计算牛顿差值和三次样条差值的平均平方误差，程序运行截图如图3所示。可以看到，对于 $cos(x)$，两种插值方法效果均较好，牛顿法的误差比样条法更小。而对于 $\frac{1}{1+25x^2}$，牛顿法对于中心区域的插值效果比样条法好，但是在远离原点处产生了不存在的振荡，并且在超出插值范围后迅速增大，而样条法在远离原点处的效果较好。

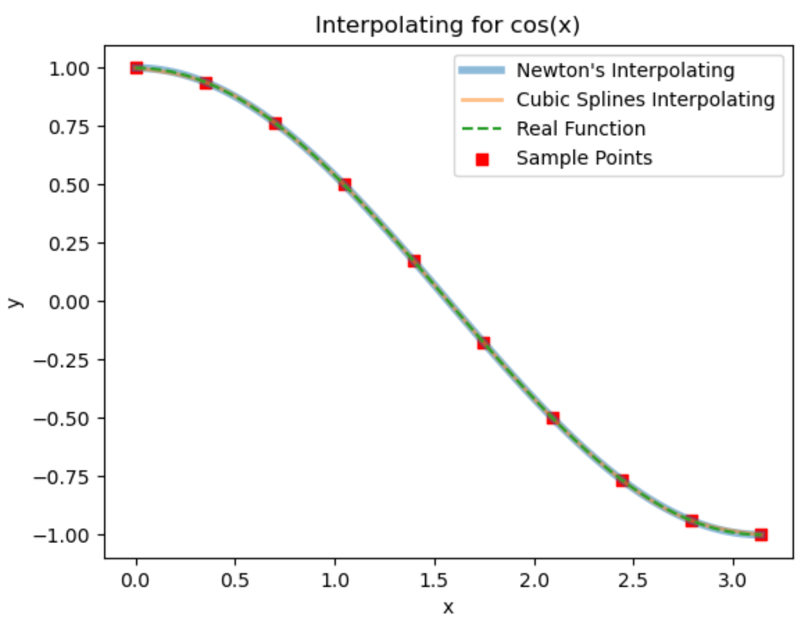


图 1: Cos(x) 插值

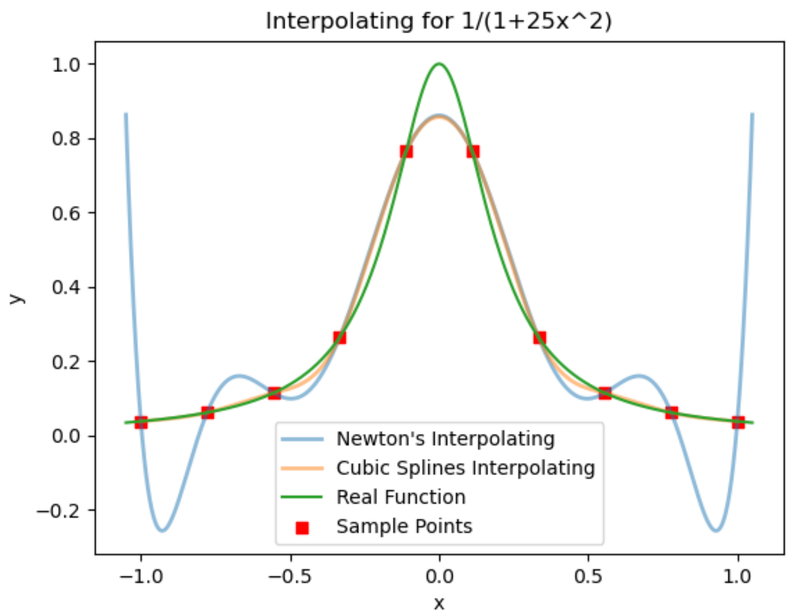
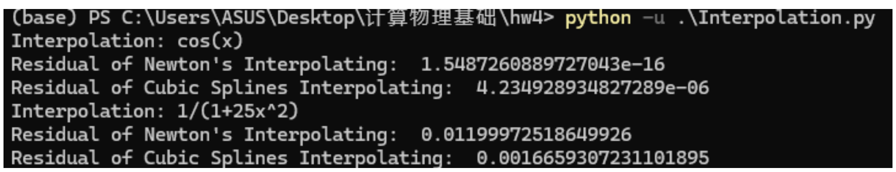


图 2: $1/(1+25x^2)$ 插值



```
(base) PS C:\Users\ASUS\Desktop\计算物理基础\hw4> python -u .\Interpolation.py
Interpolation: cos(x)
Residual of Newton's Interpolating:  1.5487260889727043e-16
Residual of Cubic Splines Interpolating:  4.234928934827289e-06
Interpolation: 1/(1+25x^2)
Residual of Newton's Interpolating:  0.011999725186499 26
Residual of Cubic Splines Interpolating:  0.0016659307231101895
```

图 3: 题目 1 程序运行截图

2 题目 3

2.1 题目描述

The table below gives the temperature along a metal rod whose ends are kept at fixed constant temperatures. The temperature is a function of the distance 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 $T(x) = a + bx + cx^2$

2.2 程序描述

在本程序中，我们将通过 SVD 分解对 T-x 做线性和二次拟合。即考虑:$A = (I, x)$ 和 $A = (I, x, x^2)$,$B = T$, 求解 $A\beta = B$。其中 SVD 分解使用 numpy 库的 np.linalg.svd 函数实现

本程序源文件为 least_square.py，在终端进入当前目录，使用命令 python -u least_square.py 运行本程序。运行时请保证 Python 第三方库 Numpy,Matplotlib 已安装。程序开发环境为 Python3.12.3，可在 Python3.8 以上版本中运行。

2.3 伪代码

2.3.1 Trapezoidal integrate 伪代码:

Algorithm 5 LeastSquarebySVD

$A \leftarrow (1, x)$ or $(1, x, x^2)$

$U, S, V \leftarrow SVD(A)$

$\beta \leftarrow VS^{-1}U^T B$

Return β

2.4 输入输出实例

对于本程序，运行后会生成线性和二次拟合曲线图4,"least_square.png" 于当前目录下, 并输出最小平方距离和 R^2, 程序运行截图为图5，得到:

Straight-line fit:$T = 8.262 + 6.052x, Leastsquares : 2811.047, R^2 = 0.6$

Quadratic-line fit:$T = 8.262 + 6.052x + 0.402x^2, Leastsquares : 331.145, R^2 = 0.95$

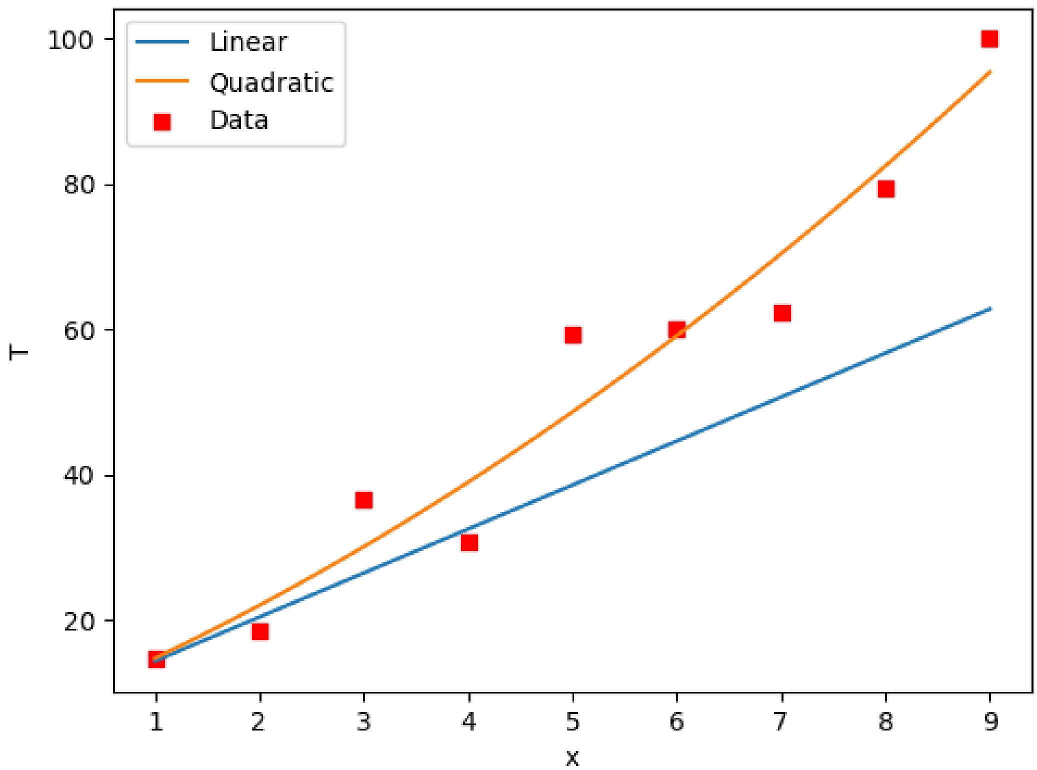


图 4: 拟合曲线

```
(base) PS C:\Users\ASUS\Desktop\计算物理基础\hw4> python -u .\least_square.py
straight-line fit:
T=8.262+6.052x
least square=2811.047
R^2=0.6
quadratic-line fit:
T=8.262+6.052x+0.402x^2
least square=331.145
R^2=0.95
```

图 5: 题目 2 程序运行截图