第六次作业

赖显松 2021214726

1 书面作业题目

P188-3

解：$L_1(x) = \frac{x-x_1}{x_0-x_1} f(x_0) + \frac{x-x_0}{x_1-x_0} f(x_1)$

$L_1(0.826) = \frac{0.826-0.83}{0.82-0.83} \times 2.270\text{5} + \frac{0.826-0.82}{0.83-0.82} 2.293319 = 2.284191\text{18}$

$L_2(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} f(x_0) + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} f(x_1) + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} f(x_2)$

$L_2(0.826) = \frac{(0.826-0.83)(0.826-0.84)}{(0.82-0.83)(0.82-0.84)} 2.270\text{5} + \frac{(0.826-0.82)(0.826-0.84)}{(0.83-0.82)(0.83-0.84)} 2.293319 + \frac{(0.826-0.82)(0.826-0.83)}{(0.84-0.82)(0.84-0.83)} 2.316367$

$= 2.284163659117$

实际结果 $e^{0.826} = 2.284163787$

- 一次 Lagrange 插值误差为：$-2.739175\text{6} \times 10^{-5}$
- 二次 lagrange 插值误差为：$-1.282981\text{68} \times 10^{-7}$

lagrange 误差界：

$M_2 \geq \max_{0.82 \leq x \leq 0.83} |f''(x)| = 2.293319$

$M_3 \geq \max_{0.82 \leq x \leq 0.84} |f'''(x)| = 2.316367$

- 一次 lagrange 插值误差界：

$\frac{M_2}{2!} |w_2(x)| = \frac{1}{2} \cdot 2.293319 \cdot |(0.84-0.82)(0.826-0.83)| = 2.752 \times 10^{-5}$

- 二次 lagrange 插值误差界：

$\frac{M_3}{3!} |w_3(x)| = \frac{1}{6} \cdot 2.316367 \cdot |(0.826-0.82)(0.826-0.83)(0.826-0.84)| = 1.29717 \times 10^{-7}$

P188 T7

解: 均差表

x_i	$f(x_i)$	一阶均差	二阶均差	三阶均差
0	3			
1	3	0		
$\frac{3}{2}$	$\frac{15}{4}$	$\frac{1}{2}$	$\frac{1}{3}$	
2	$\frac{5}{3}$	$-\frac{19}{6}$	$-\frac{11}{3}$	-2

- $N_3(x) = f(x_0) + f[x_0, x_1](x-x_0) + f[x_0, x_1, x_2](x-x_0)(x-x_1)$

  $+f[x_0, x_1, x_2, x_3](x-x_0)(x-x_1)(x-x_2)$

  $= 3 + \frac{1}{3}x(x-1) + (-2)x(x-1)(x-\frac{3}{2})$

- $R_3(x) = f[x, x_0, x_1, x_2, x_3](x-x_0)(x-x_1)(x-x_2)(x-x_3)$

  $= f[x, 0, 1, \frac{3}{2}, 2]\, x(x-1)(x-\frac{3}{2})(x-2)$

2 编程题目

2.1 P190 计算实习题 T1

求均差函数：

钱老师给出的范例，求均差函数（输入：x 序列，对应的函数值序列 f，求解所致最高阶次；输出：从一阶到制定阶次的 ndarray list）。代码如下：

```
def DivDiff(x,f,order):
#给出函数离散节点上的差商/均差(divided differences)
#从 1 阶最高算至 order 阶
    #数据点数
    n = x.size
    #初始化一个输出 list
    val = []
    #当前差商为 0 阶
    currDD = f.copy()
    #当前差商长度
    l = n
    #开始循环，直到覆盖 order 阶
    for i in range(1,order+1):
        #自变量差
        diffx = x[i:n]-x[0:n-i]
        #函数差，逐阶递推
        df = currDD[1:l]-currDD[0:l-1]
        #更新当前差商
        l = l-1
        currDD = np.zeros((l,))
        currDD = df/diffx
        val.append(currDD)
    return val
```

其中，三次样条插值用到的均差阶次为二阶，因此在调用这个函数的时候阶次参数输入 2。需要将均差输出的 list 中的二阶元素提取出来使用。

***加分项：三对角非线性方程组矩阵求解函数：**

三对角非线性方程组求解 LAE 函数，输入输出设置与 numpy 函数库一致：（输入：三对角非线性方程组矩阵 A，非线性方程组 b；输出：方程组的解 x）代码如下：

```
# 函数：A 为三对角矩阵求解 LAE,追赶法
def solveTridiagA(A, b):
    n = A.shape[0]
    [L, U] = LUDecomp(A, 'LU')
```

```
    # 求 y
    y = np.zeros((n,))
    y[0] = b[0]
    for i in range(1, n):
        y[i] = b[i]-L[i, i-1]*y[i-1]
    # 求 x
    x = np.zeros((n,))
    x[n-1] = y[n-1]/U[n-1, n-1]
    for i in range(n-2, -1, -1):
        x[i] = (y[i]-U[i, i+1]*x[i+1])/U[i, i]
    return x
```

三次样条插值函数：

三次样条插值函数选用了两种边界条件，整体代码较长，分部解释。（输入：待插值的点 x_intp，插值节点序列 x，插值节点序列对应的函数值 f，边界条件 1 或 2，可选参数传入边界条件）。

提取边界条件：

```
# 提取边界条件
    if method == 1:
        df0, dfn = arg[0], arg[1]
    elif method == 2:
        d2f0, d2fn = arg[0], arg[1]
    else:
        print('请输入正确的边界条件方法')
        return None
```

从可选参数 arg 中取出对应的左右端点一阶导或者是左右端点二阶导。

统一下标、计算二阶均差：

```
# 统一 n，注意，有 n+1 个插值节点，但是下表从 0 到 n，这里的 n 与下标统一
    n = len(x)-1

    # 计算所有的二阶均差
    mean_df = DivDiff(x, f, 2)
    mean_df2 = mean_df[1]
```

计算 h， μ 和 λ：

根据公式求解三个序列。

```
#计算 h，mu， lamda
h = x[1:n+1] - x[0: n]
mu = h.copy()
for j in range(1, n):
    mu[j] = h[j-1]/(h[j-1]+h[j])
```

```
lamda = -1*mu + 1
```

接下来这部分有两种不同的边界条件。

计算 d，构造方程组（边界条件 1）：

```
d = np.zeros(n+1,)
d[0] =  6 * ((f[1]-f[0])/(x[1]-x[0])**2 - df0/(x[1]-x[0]))
d[n] =  6 * (dfn/(x[n]-x[n-1]) - (f[n]-f[n-1])/(x[n]-x[n-1])**2)
for i in range(n-1):
    d[i+1] = 6 * mean_df2[i]
A = np.zeros((n+1, n+3))
A[0, 1], A[0, 2] = [2, 1]
A[n, n], A[n, n+1] = [1, 2]
for i in range(n-1):
    A[i+1, i+1] = mu[i+1]
    A[i+1, i+2] = 2
    A[i+1, i+3] = lamda[i+1]
A = A[:, 1:n+2]
# 求解 M(numpy 函数)
# M = np.linalg.solve(A, d)
# 自己写的三对角追赶法
M = lds.solveTridiagA(A, d)
```

计算 d，构造方程组（边界条件 2）：

```
d = np.zeros(n-1,)
for i in range(n-1):
    d[i] = 6 * mean_df2[i]
d[0] -= mu[1]*d2f0
d[n-2] -= lamda[n-1]*d2fn
A = np.zeros((n-1, n+1))
for i in range(n-1):
    A[i, i] = mu[i+1]
    A[i, i+1] = 2
    A[i, i+2] = lamda[i+1]
A = A[:, 1:n]
# 求解 M(numpy 函数)
# M1 = np.linalg.solve(A, d)
# 自己写的三对角追赶法
M1 = lds.solveTridiagA(A, d)
M = np.concatenate((np.array([d2f0]), M1, np.array([d2fn])))
```

求解 M（边界条件 1）：

```
# 求解 M(numpy 函数)
# M = np.linalg.solve(A, d)
# 自己写的三对角追赶法
M = lds.solveTridiagA(A, d)
```

求解 M（边界条件 2）：

```
# 求解 M(numpy 函数)
# M1 = np.linalg.solve(A, d)
# 自己写的三对角追赶法
M1 = lds.solveTridiagA(A, d)
M = np.concatenate((np.array([d2f0]), M1, np.array([d2fn])))
```

求解插值函数值：

找到插值点在节点序列中的位置并根据插值函数求出插值后的函数值。

```
j = np.searchsorted(x, x_intp)
if x_intp == x[j]:
        return f[j]
else:
        j -= 1
        s1 = M[j]/6/h[j]*(x[j+1]-x_intp)**3
        s2 = M[j+1]/6/h[j]*(x_intp-x[j])**3
        s3 = (f[j]-M[j]*h[j]**2/6) * (x[j+1]-x_intp)/h[j]
        s4 = (f[j+1]-M[j+1]*h[j]**2/6) * (x_intp-x[j])/h[j]
        return s1 + s2 + s3 + s4
```

函数准备结束，进入解题部分：

导入数据：

```
# 计算实习题第三题的数据
x = np.array([0.9,1.3,1.9,2.1,2.6,3.0,3.9,4.4,4.7,5,6,7,8,9.2,10.5,11.3,11.6,12,12.6,13,13.3])
f = np.array([1.3,1.5,1.85,2.1,2.6,2.7,2.4,2.15,2.05,2.1,2.25,2.3,2.25,1.95,1.4,0.9,0.7,0.6,0.5,0.4,0.25])
```

生成插值序列并用三次样条插值函数求解对应的函数值，将两种边界条件和 scipy 库中的函数所得到的结果通过 subplot 显示在同一张图片上。代码如下：

```
xx = np.arange(0.9, 13.3, 0.01)
yy = np.zeros(len(xx))
for i in range(len(xx)):
    yy[i] = itp.cubicSpline(xx[i], x, f, 1, 0, 0)
plt.subplot(3,1,1)
plt.plot(xx, yy)
plt.scatter(x, f)
plt.title('边界条件 1')
```

结果

三种方法得到的插值结果如图 2. 1 所示：

可以看出，两种边界条件得到的插值结果在两端会有稍微的不同。当导数值

都取 0 时，边界条件 2 的插值结果更接近 scipy 函数库中的插值函数，但是还是有一点不同，scipy 的结果在左边的端点处有轻微上凸，1 型下凹，2 型近似直线。

插值优化的目的是使曲线平滑。考虑端点最近邻的点，将一阶阶均差序列的首尾两侧作为的一阶导边界条件；将二阶均差序列的首尾两侧作为的二阶导边界条件。插值结果如图 2. 2 所示。

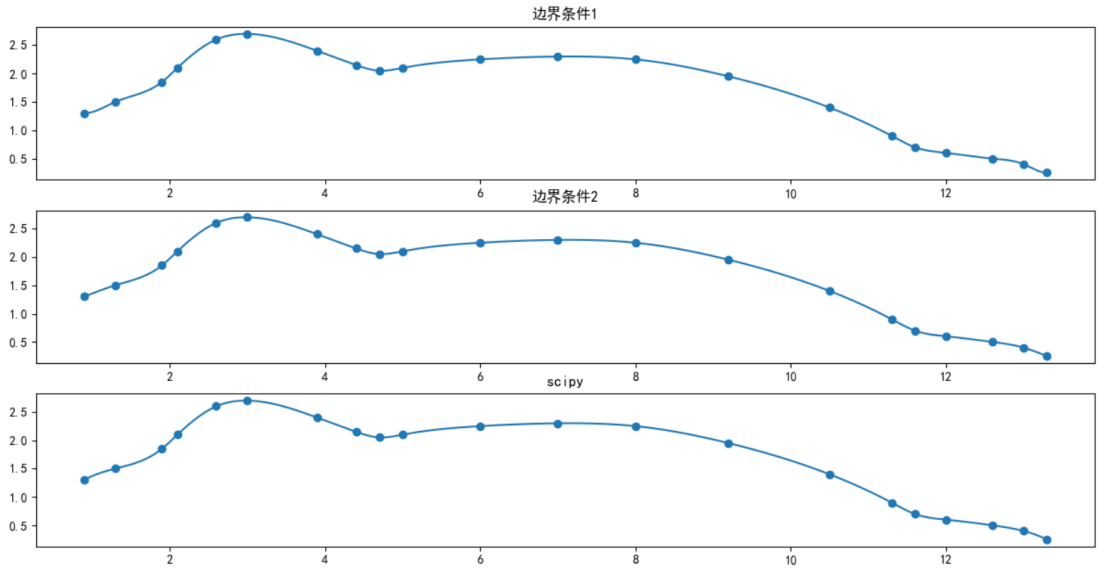

图 2. 1 3 种方法的插值结果

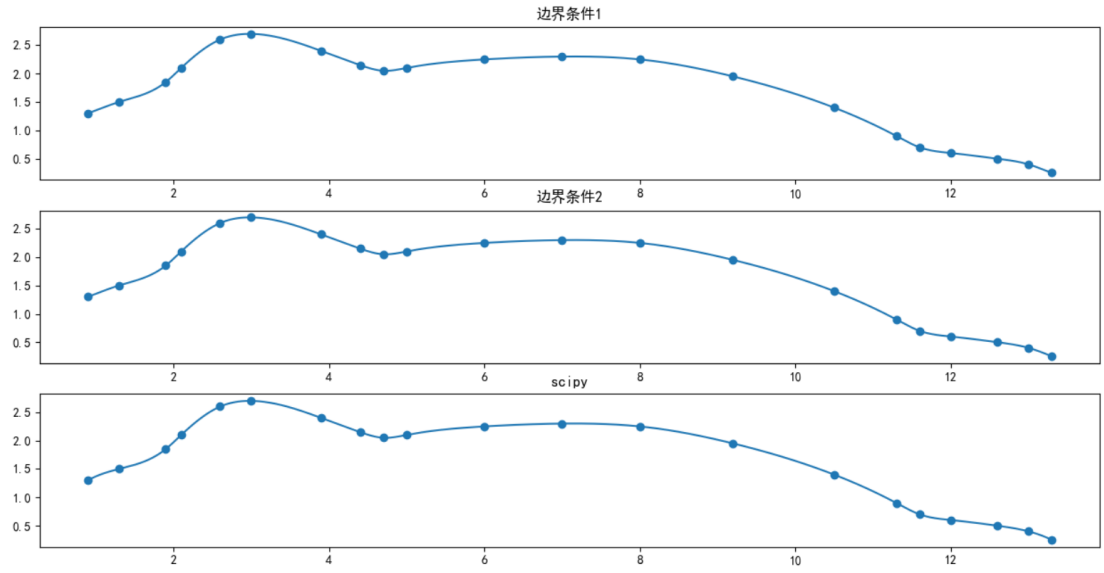

图 2. 2 3 种方法的插值结果

可以看到，经过边界条件调整的第一种边界条件的三次样条插值结果与第二种边界条件的端点处插值效果很接近了，但是这两种插值与 scipy 的插值结果还是有一点不同，光滑程度不够。

参考文献

[1] 关治．数值方法[M]．北京：清华大学出版社，2006：66-92．

附录

见附件 python 源代码。