# 计算方法上机实习二 实习报告

2019级 大气科学学院 赵志宇

学号：191830227

## 一、分析报告

### 1.问题分析

本次的实习给出平面上的20个点，要求：

a) 用三次样条插值构造出插值曲线；

b) 用最小二乘法构造出拟合曲线，拟合方程为.

20个点的坐标如下：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x | 0.99 | 0.95 | 0.87 | 0.77 | 0.67 | 0.56 | 0.44 | 0.30 | 0.16 | 0.01 |
| y | 0.39 | 0.32 | 0.27 | 0.22 | 0.18 | 0.15 | 0.13 | 0.12 | 0.13 | 0.15 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x | 0.93 | 0.85 | 0.73 | 0.59 | 0.42 | 0.29 | 0.16 | 0.05 | -0.11 | -0.20 |
| y | 0.40 | 0.41 | 0.42 | 0.43 | 0.42 | 0.41 | 0.40 | 0.36 | 0.32 | 0.22 |

首先绘制出(x, y)的散点图，观察散点在平面上的分布情况.

从图中可以看出散点大致围出了一个闭合的区域，因此在进行三次样条插值时使用周期性边界条件.

### 2.算法细节

#### (1)三次样条插值的实现

使用三弯矩法计算各个区间上样条函数的系数.

给定函数在区间上的一组节点，将划分为n个子区间.

设样条函数为，每个区间上的样条函数为.

记.

计算得

不考虑边界条件时，有如下的方程组

再加上周期性边界条件.

由和，得

以上两个方程组联立可解出，进而求得的系数.

考虑到给出的散点围住了一块闭合区域，无法使用一个样条函数进行插值，所以考虑使用参数方程的形式进行插值.

设插值曲线的参数方程为

然后分别对和进行插值即可.

因为共有21个点（周期边界条件又加了一个点），所以u取值定在1到21之间，插值节点可以简单地设，方便计算.

记样条函数为，将它们联立即得到原问题的插值函数.

#### (2)最小二乘法的实现

将看作特征变量，看作目标变量，通过最小二乘法来计算中的参数.

由线性代数的相关知识可知，最小二乘解满足方程

解方程组即可求得.

#### (3)线性方程组的解法

使用高斯消元法解线性方程组.

对于线性方程组，构造增广矩阵，对矩阵进行初等变换，将化成行阶梯矩阵，通过回代求出x.

### 3.编程思路

程序主要分为5个模块，主程序jsff2，用来进行三次样条插值的子例程cubic\_spline，用来进行线性回归的子例程linear\_regression，用来解线性方程组的子例程gauss\_elimination，以及用来输出调试信息的子例程print\_matrix.

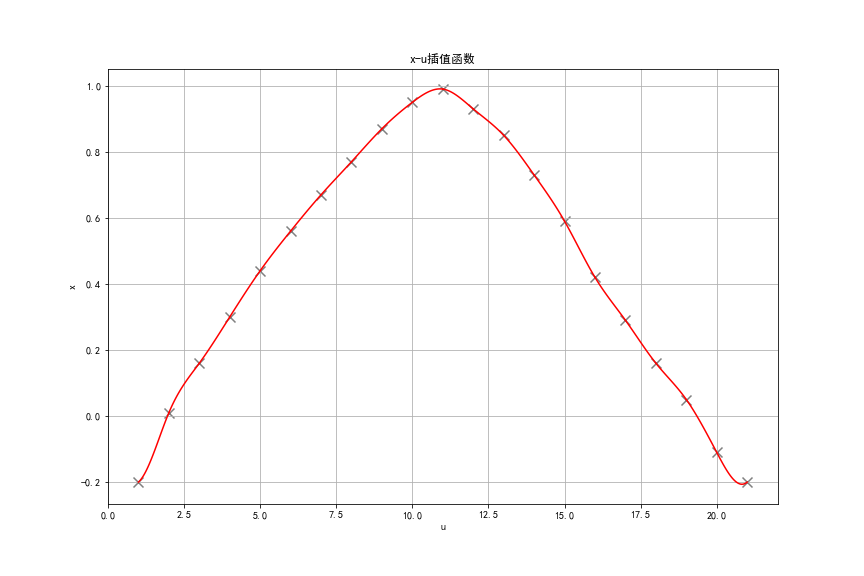
用变量n代表点的个数，注意一共有20个点，故n取20，但是周期边界条件要求第一个元素和最后一个元素相等，所以在数组末尾把第一个元素加进去，所以实际上一共有(n+1)个点，三次样条插值的增广矩阵的维数为(n+1)\*(n+2).

用变量m表示特征的个数，一共有四个特征(xi, yi, xiyi, yi2)，故m取4，但是为了让拟合出来的方程有截距，把1加入到了特征当中，所以实际上有(m+1)个特征，ATA的维数为(m+1)\*(m+1).

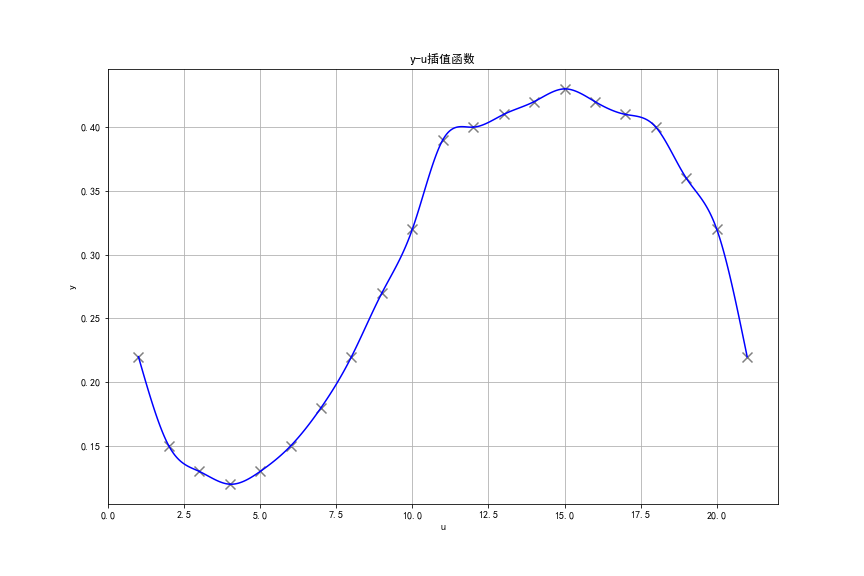
### 4.曲线图形

#### (1)图片

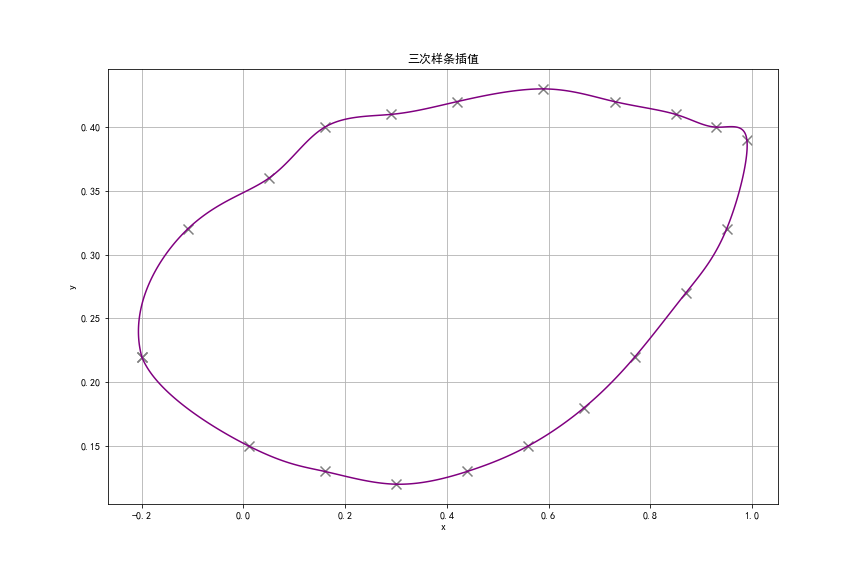
图像：



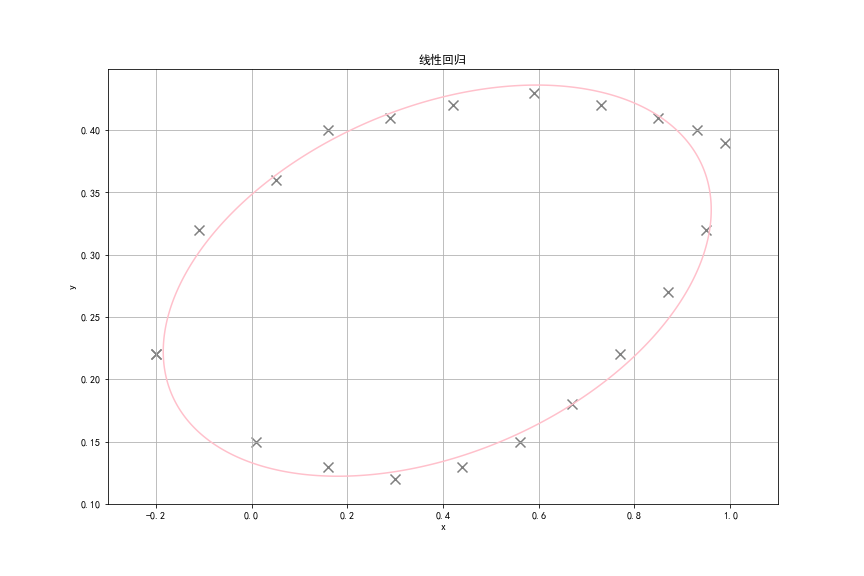
图像：



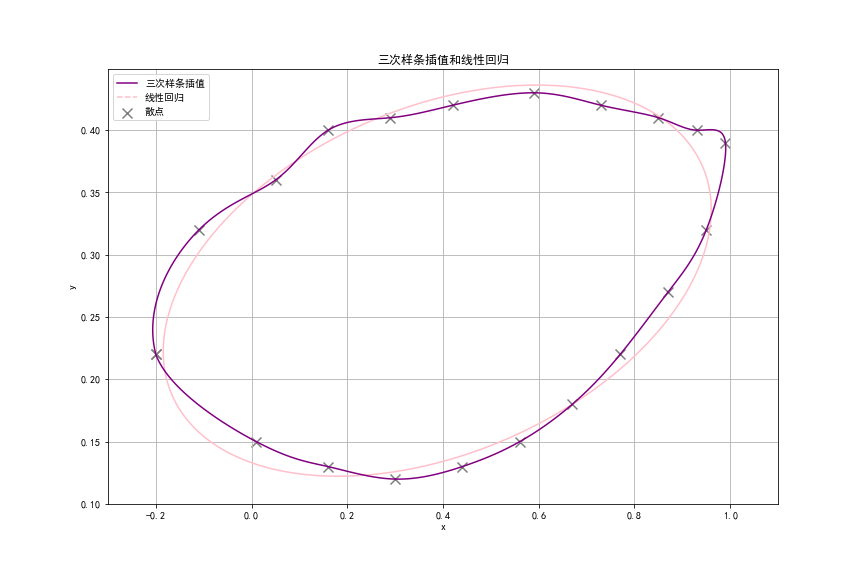
三次样条插值函数图像：



线性回归曲线图像：



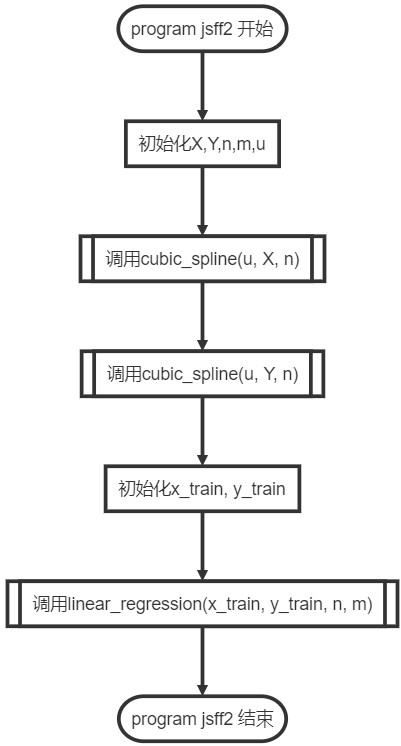
三次样条插值和线性回归：

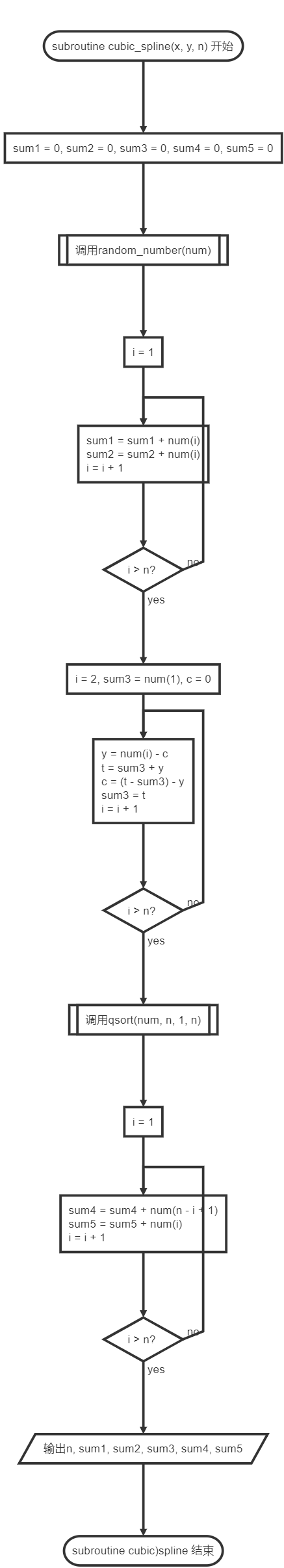


#### (2)分析

从上图可以看出，插值函数经过了每一个样本点，形状较为曲折，在训练集中的误差为0；拟合函数为一个椭圆，刻画了样本点的分布趋势，在训练集中的误差不为0.

## 二、编程流程图





## 三、源代码

### Fortran主程序

program jsff2
  
 ! homework2 of Numerical Methods
  
 ! arthor : zzy
  
   
 implicit none
  
 ! X : x coordinates, Y : y coordinates
  
 ! X(1) = X(21), Y(1) = Y(21) inorder to apply periodical boundary conditions
  
 real(8), dimension(21) :: X = [-0.20, 0.01, 0.16, 0.30, 0.44, 0.56, 0.67, 0.77, 0.87, 0.95,&
  
 0.99, 0.93, 0.85, 0.73, 0.59, 0.42, 0.29, 0.16, 0.05, -0.11, -0.20]
  
 real(8), dimension(21) :: Y = [0.22, 0.15, 0.13, 0.12, 0.13, 0.15, 0.18, 0.22, 0.27, 0.32,&
  
 0.39, 0.4, 0.41, 0.42, 0.43, 0.42, 0.41, 0.4, 0.36, 0.32, 0.22]
  
 ! u : the parameter of the curve X = X(u), Y = Y(u)
  
 real(8), dimension(21) :: u
  
 ! x\_train : character variables in linear regression
  
 real(8), dimension(20, 5) :: x\_train
  
 ! y\_train : target variables in linear regression
  
 real(8), dimension(20) :: y\_train
  
 ! i : loop variable, n : the number of samples, m : the number of chracters
  
 ! 20 + 1 points are used in cubic spline
  
 ! 20 points and 4 characters(x, y, x\*y, y^2) are used in linear regression
  
 integer(4) :: i, n = 20, m = 4
  
   
 ! initialize u
  
 do i = 1, n + 1
  
 u(i) = dble(i)
  
 end do
  
 print \*, 'Mx : '
  
 call cubic\_spline(u, X, n)
  
 print \*, 'My : '
  
 call cubic\_spline(u, Y, n)
  
   
 ! initialize x\_train and y\_train
  
 do i = 1, n
  
 x\_train(i, 1) = 1
  
 x\_train(i, 2) = X(i)
  
 x\_train(i, 3) = Y(i)
  
 x\_train(i, 4) = X(i) \* Y(i)
  
 x\_train(i, 5) = Y(i) \* Y(i)
  
 y\_train(i) = X(i) \* X(i)
  
 end do
  
   
 call linear\_regression(x\_train, y\_train, n, m)
  
   
end program jsff2
  
   
subroutine cubic\_spline(x, y, n)
  
 ! apply cubic spline interpolation algorithm
  
 ! parameters: x, y : coordinates of the points to be interpolated
  
 ! n : the number of the points to be interpolated
  
 ! author: zzy
  
   
 integer(4), intent(in) :: n
  
 integer(4) :: i
  
 real(8), intent(in), dimension(n + 1) :: x
  
 real(8), intent(in), dimension(n + 1) :: y
  
 ! B : augmented matrix, B's shape is (n + 1, n + 2) since there are (n + 1) points
  
 real(8), dimension(n + 1, n + 2) :: B
  
 ! M : second derivative of spline functions in each interval
  
 real(8), dimension(n + 1) :: M
  
 ! h : h(i) = x(i) - x(i - 1)
  
 real(8), dimension(n) :: h
  
   
 ! calculate h
  
 do i = 2, n + 1
  
 h(i) = x(i) - x(i - 1)
  
 end do
  
   
 ! calculate B according to three-moment method and periodical boundary condition
  
   
 ! M(1) == M(n + 1), periodical boundary condition
  
 B(1, 1) = 1
  
 B(1, n + 1) = -1
  
   
 do i = 2, n
  
 ! alpha(i) \* M(i - 1) + 2 \* M(i) + (1 - alpha(i)) \* M(i + 2) == beta(i)
  
 B(i, i - 1) = h(i) / (h(i) + h(i + 1))
  
 B(i, i) = 2.0
  
 B(i, i + 1) = h(i + 1) / (h(i) + h(i + 1))
  
 B(i, n + 2) = 6 / (h(i) + h(i + 1)) \* ((y(i + 1) - y(i)) / h(i + 1) - (y(i) - y(i - 1)) / h(i))
  
 end do
  
   
 ! periodical boundary condition
  
 B(n + 1, 2) = h(2) / (h(2) + h(n + 1))
  
 B(n + 1, n) = -h(n) / (h(2) + h(n + 1))
  
 B(n + 1, n + 1) = 2
  
 B(n + 1, n + 2) = 6 / (h(2) + h(n + 1)) \* ((y(2) - y(1)) / h(2) - (y(n + 1) - y(n)) / h(n + 1))
  
   
 call gauss\_elimination(B, M, n)
  
   
 print \*, M
  
   
end subroutine cubic\_spline
  
   
subroutine linear\_regression(A, y, n, m)
  
 ! apply linear regression algorithm
  
 ! parameters: A : matrix of character variables, shape is (n, m + 1)
  
 ! y : vector of target variables
  
 ! n : the number of (x, y)
  
 ! m : the number of characters
  
 ! author: zzy
  
   
 integer(4), intent(in) :: n, m
  
 real(8), dimension(n, m + 1) :: A
  
 ! ATA : result of transpose(A) \* A
  
 real(8), dimension(m + 1, m + 1) :: ATA
  
 ! B : agumented matrix
  
 real(8), dimension(m + 1, m + 2) :: B
  
 real(8), dimension(n) :: y
  
 ! ATy : result of transpose(A) \* y
  
 real(8), dimension(m + 1) :: ATy
  
 ! theta : solution of ATA\*b == ATy
  
 real(8), dimension(m + 1) :: theta
  
 integer(4) :: i, j
  
   
 ! call print\_matrix(A, n, m + 1)
  
 ! call print\_matrix(y, m + 1, 1)
  
   
 ATA = matmul(transpose(A), A)
  
 ATy = matmul(transpose(A), y)
  
   
 ! call print\_matrix(ATA, m + 1, m + 1)
  
 ! call print\_matrix(ATy, m + 1, 1)
  
   
 ! intialize agumented matrix
  
 do i = 1, m + 1
  
 do j = 1, m + 1
  
 B(i, j) = ATA(i, j)
  
 end do
  
 end do
  
   
 do i = 1, m + 1
  
 B(i, 6) = ATy(i)
  
 end do
  
   
 ! call print\_matrix(B, m + 1, m + 2)
  
   
 ! solve the equation ATA\*b == ATy
  
 call gauss\_elimination(B, theta, m)
  
   
 print \*, 'b : ', theta
  
   
end subroutine linear\_regression
  
   
subroutine gauss\_elimination(B, theta, n)
  
 ! apply gauss elimination algorithm
  
 ! parameters: B : agumented matrix
  
 ! theta : solution of linear equations
  
 ! n : the length of theta is (n + 1)
  
 ! author: zzy
  
   
 integer(4), intent(in) :: n
  
 real(8), intent(in out), dimension(n + 1, n + 2) :: B
  
 real(8), intent(in out), dimension(n + 1) :: theta
  
 integer(4) :: i, j, k
  
   
 ! use elementary transformation to transform B into upper triangular matrix
  
 do i = 1, n + 1 ! ii : rows
  
 do j = i + 1, n + 2 ! j : columns
  
 B(i, j) = B(i, j) / B(i, i)
  
 end do
  
 B(i, i) = 1
  
 do j = i + 1, n + 1 ! j : rows
  
 do k = i + 1, n + 2 ! k : columns
  
 B(j, k) = B(j, k) - B(j, i) \* B(i, k)
  
 end do
  
 B(j, i) = 0
  
 end do
  
 end do
  
   
 ! solve theta by transform B(1:n+1, 1:n+1) into diagonal matrix
  
 do i = n + 1, 1, -1
  
 do j = i + 1, n + 1
  
 B(i, n + 2) = B(i, n + 2) - theta(j) \* B(i, j)
  
 end do
  
 theta(i) = B(i, n + 2);
  
 end do
  
   
end subroutine gauss\_elimination
  
   
subroutine print\_matrix(A, m, n)
  
 ! debug function, print a matrix
  
 ! parameters: A : matrix to be printed
  
 ! (m, n) : shape of matrix
  
 ! author: zzy
  
   
 integer(4) :: m, n, i
  
 real(8), dimension(m, n) :: A
  
   
 do i = 1, m
  
 print \*, A(i, :)
  
 end do
  
   
end subroutine print\_matrix

### Python绘图程序

import matplotlib.pyplot as plt
  
import scipy.interpolate as spi
  
from sklearn.linear\_model import LinearRegression as LR
  
import numpy as np
  
import pandas as pd
  
plt.rcParams['font.sans-serif']=['SimHei'] # 用来正常显示中文标签
  
plt.rcParams['axes.unicode\_minus'] = False # 用来正常显示负号
  
   
# 初始化，x、y为坐标，Mx、My为Fortran计算出的二阶导数值，theta为Fortran计算出了线性拟合参数b
  
x = [-0.2, 0.01, 0.16, 0.3, 0.44, 0.56, 0.67, 0.77, 0.87, 0.95, 0.99, 0.93, 0.85, 0.73, 0.59, 0.42, 0.29, 0.16, 0.05, -0.11, -0.2]
  
y = [0.22, 0.15, 0.13, 0.12, 0.13, 0.15, 0.18, 0.22, 0.27, 0.32, 0.39, 0.4, 0.41, 0.42, 0.43, 0.42, 0.41, 0.4, 0.36, 0.32, 0.22]
  
Mx = [0.50709311714530891, -0.22746861616752340, 4.2781310867928156E-002, -3.6565142043757702E-003, -2.8155432864359401E-002, -3.7216398972686393E-003, -1.6957950326107054E-002, 1.1553140794287205E-002, -2.9254255223173112E-002, -1.4536363088545445E-002, -0.15260006354080916, 2.4936474200634589E-002, -6.7145718820811259E-002, 3.6462723365777105E-003, -6.7439613712450261E-002, 8.6112532988534593E-002, -3.7010568309589723E-002, 6.1929740249824290E-002, -9.0708395371916439E-002, 9.0381441575125630E-004, 0.50709311714530891]
  
My = [4.9409026797001335E-004, 7.8011944728198723E-002, -1.2541976469125507E-002, 3.2156038038295066E-002, 3.9177992819944257E-003, 1.2172841723718993E-002, 7.3907766026706191E-003, 1.8263994645139545E-002, -2.0446633589753450E-002, 6.3522360899939928E-002, -0.11364274563698991, 3.1048786156839283E-002, -1.0552577804301562E-002, 1.1161525060366965E-002, -3.4093343623231971E-002, 5.2116062456102332E-003, 1.3247097454725363E-002, -5.8199996064511685E-002, 3.9552879650763990E-002, -0.10001170135247860, 4.9409026797001335E-004]
  
theta = [-0.61675402721062844, 3.7372895747393553E-002, 6.4038200357358814, 2.6440756536123722, -13.302206388291150]
  
   
# 在一张图中绘制插值曲线和拟合曲线
  
fig, ax = plt.subplots(figsize=(12, 8))
  
   
for i in range(1, n + 1):
  
 u = np.linspace(i, i + 1, 100)
  
 xu = (Mx[i - 1] / 6 \* (i + 1 - u) \*\* 3 + Mx[i] / 6 \* (u - i) \*\* 3 + (x[i] - Mx[i] / 6) \* (u - i) + (x[i - 1] - Mx[i - 1] / 6) \* (i + 1 - u))
  
 yu = (My[i - 1] / 6 \* (i + 1 - u) \*\* 3 + My[i] / 6 \* (u - i) \*\* 3 + (y[i] - My[i] / 6) \* (u - i) + (y[i - 1] - My[i - 1] / 6) \* (i + 1 - u))
  
 if i == n:
  
 plt.plot(xu, yu, color='purple', label='三次样条插值')
  
 else:
  
 plt.plot(xu, yu, color='purple')
  
   
xx = np.arange(-0.3, 1.1, 0.001)
  
yy = np.arange(0.1, 0.45, 0.001)
  
xx, yy = np.meshgrid(xx, yy)
  
CS = ax.contour(xx, yy, zz, 0, colors='pink')
  
CS.collections[0].set\_label('线性回归')
  
   
ax.scatter(x, y, c='gray', marker='x', s=100, label='散点')
  
   
ax.grid()
  
ax.set\_xlabel('x')
  
ax.set\_ylabel('y')
  
plt.legend(loc='upper left')
  
plt.title('三次样条插值和线性回归')
  
   
plt.show()
  
plt.close()

## 四、运行结果

