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

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；

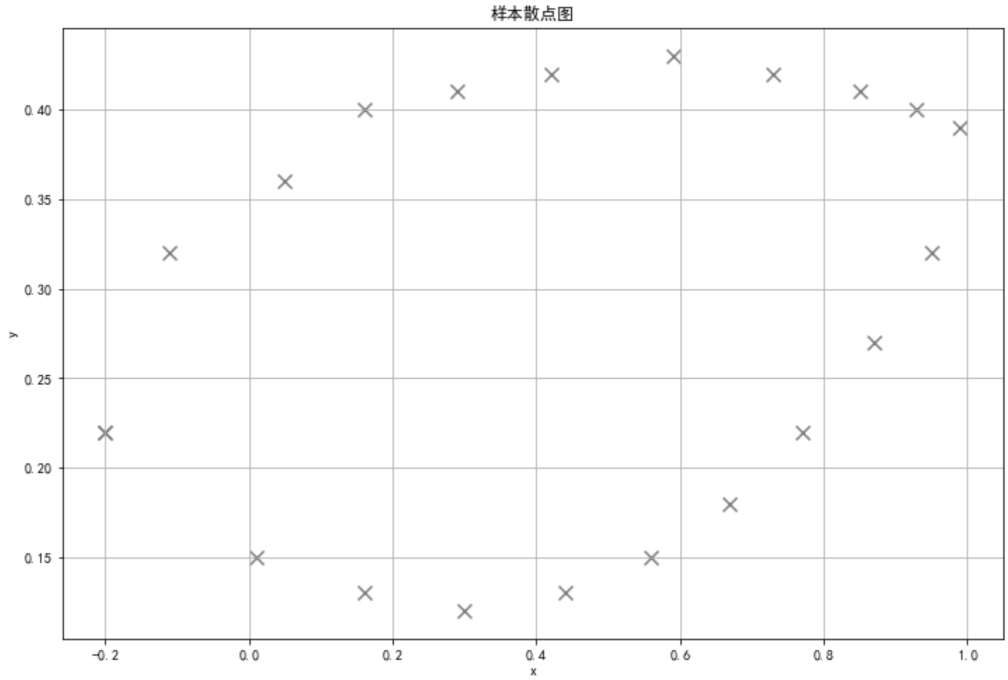
b) 用最小二乘法构造出拟合曲线，拟合方程为$b_0 + b_1x + b_2y + b_3xy + b_4y^2 = x^2$.

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$y=f(x)$在区间$[a,b]$上的一组节点$(x_k,y_k)\ (k=0,1,2,\cdots,n)$，将$[a,b]$划分为n个子区间$[x_{i-1},x_i]\ (i=1,2,\cdots,n)$.

设样条函数为$S(x)$，每个区间上的样条函数为$S_i(x)=a_ix^3+b_ix^2+c_ix+d_i\ (i=1,2,\cdots,n)$.

记$M_i=S''(x_i),h_i=x_i-x_{i-1}\ (i=1,2,\cdots,n)$.

计算得

$$S(x)=\frac{M_{i-1}}{6h_i}(x_i-x)^3+\frac{M_i}{6h_i}(x-x_{i-1})^3+(\frac{y_i}{h_i}-\frac{h_iM_i}{6})(x-x_{i-1})+(\frac{y_{i-1}}{h_i}-\frac{h_iM_{i-1}}{6})(x_i-x)$$
$$x_{i-1}\le x\le x_i, i=1,2,\cdots,n-1$$

不考虑边界条件时，有如下的方程组

$$\begin{cases}\alpha_1M_0+2M_1+(1-\alpha_1)M_2=\beta_1\\ \alpha_1M_1+2M_2+(1-\alpha_2)M_3=\beta_2\\ \qquad\qquad\qquad\vdots\\ \alpha_{n-1}M_{n-2}+2M_1+(1-\alpha_{n-1})M_n=\beta_{n-1}\end{cases}$$

$$\alpha_i=\frac{h_i}{h_i+h_{i+1}},\beta_i=\frac{6}{h_i+h_{i+1}}(\frac{y_{i+1}-y_i}{h_{i+1}}-\frac{y_i-y_{i-1}}{h_i})$$

再加上周期性边界条件$y_0=y_n,y_0'=y_n',y_0''=y_n''$.

由$M_0=M_n$和$y_0'=y_n'$，得

$$\begin{cases}M_0=M_n\\ \frac{h_1}{h_1+h_n}M_1-\frac{h_n}{h_1+h_n}M_{n-1}+2M_n=\frac{6}{h_1+h_n}(\frac{y_1-y_0}{h_1}-\frac{y_n-y_{n-1}}{h_n})\end{cases}$$

以上两个方程组联立可解出M_i，进而求得$S(x)$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$S(x)$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

$$\begin{cases} x = \phi(u) \\ y = \psi(u) \end{cases} \quad (1 \le u \le 21)$$

然后分别对(u_i, ϕ_i)和(u_i, ψ_i)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），所以使u的取值在1到21之间，插值节点可以简单地设$u_i = i$，方便计算.

记样条插值函数为$x = \Phi(u), y = \Psi(u)$，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$(x_i, y_i, x_i y_i, y_i^2)$看作特征变量，$x_i^2$看作目标变量，通过最小二乘法来计算$x^2 = b_0 + b_1 x + b_2 y + b_3 xy + b_4 y^2$中的参数$b_i$.

由线性代数的相关知识可知，最小二乘解$b = (b_0, b_1, b_2, b_3, b_4)^T$满足方程

$$A^T A b = A^T y$$

$$A = \begin{bmatrix} 1 & x_1 & y_1 & x_1 y_1 & y_1^2 \\ 1 & x_2 & y_2 & x_2 y_2 & y_2^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_m & y_m & x_m y_m & y_m^2 \end{bmatrix}, y = \begin{bmatrix} x_1^2 \\ x_2^2 \\ \vdots \\ x_m^2 \end{bmatrix}$$

解方程组即可求得b_i.

(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$Ax = b$，构造增广矩阵$B = [A \ \ b]$，对矩阵B进行初等变换，将B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, yi^2)，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，A^{T}A的维数为(m+1)*(m+1).

4.运行结果分析

(1)插值与拟合结果

所有分段插值函数如下，$\Phi_i(u), \Psi_i(u)$代表区间 $u \in [i, i+1]$上的插值函数：

$$\Phi_1(u) = -0.122426955552139u^3 + 0.620827425229071u^2 - 0.795493586822241u + 0.0970931171453089$$
$$\Phi_2(u) = 0.0450416545059086u^3 - 0.383984235119213u^2 + 1.2141297399838u - 1.24265577553802$$
$$\Phi_3(u) = -0.00773963751205065u^3 + 0.09104739304242u^2 - 0.210965163351066u + 0.182439165496785$$
$$\Phi_4(u) = -0.00408315310999727u^3 + 0.0471695802177794u^2 - 0.0354538822501808u - 0.0515759554439212$$
$$\Phi_5(u) = 0.00407229882784846u^3 - 0.0751621988499066u^2 + 0.576204994014763u - 1.07100735230721$$
$$\Phi_6(u) = -0.00220605173813974u^3 + 0.0378481113378809u^2 - 0.101856876648706u + 0.285116427166702$$
$$\Phi_7(u) = 0.00475184852006571u^3 - 0.108267794084433u^2 + 0.920954511375397u - 2.10144371187308$$
$$\Phi_8(u) = -0.00680123266957672u^3 + 0.169006154466985u^2 - 1.29723713664059u + 3.81373433406101$$
$$\Phi_9(u) = 0.00245298202243794u^3 - 0.0808576422174111u^2 + 0.951537074050127u - 2.93258854119811$$
$$\Phi_{10}(u) = -0.023010616742044u^3 + 0.683050320717046u^2 - 6.68754259344141u + 22.5310106047535$$
$$\Phi_{11}(u) = 0.0295894229569073u^3 - 1.05275098934835u^2 + 12.4062718411197u - 47.479642496811$$
$$\Phi_{12}(u) = -0.015347032170241u^3 + 0.564961395228992u^2 - 7.00627679288179u + 30.170552191783$$
$$\Phi_{13}(u) = 0.0117986651928982u^3 - 0.493720801933434u^2 + 6.75659179168742u - 29.4685451939834$$
$$\Phi_{14}(u) = -0.011847647674838u^3 + 0.499424338511485u^2 - 7.14744013401028u + 35.4169367476484$$
$$\Phi_{15}(u) = 0.0255920244501641u^3 - 1.18536090711361u^2 + 18.1243384919536u - 90.9419557980455$$
$$\Phi_{16}(u) = -0.0205205168830207u^3 + 1.02804107687926u^2 - 17.2900932435877u + 97.9350133691652$$
$$\Phi_{17}(u) = 0.016490051426569u^3 - 0.859497906909814u^2 + 14.7980694808266u - 83.8979087358491$$
$$\Phi_{18}(u) = -0.0254396892702901u^3 + 1.40470809072058u^2 - 25.9576384760735u + 160.636339000187$$
$$\Phi_{19}(u) = 0.0152687016312779u^3 - 0.915670190668801u^2 + 18.1295488747951u - 118.582514278605$$
$$\Phi_{20}(u) = 0.0843648837882596u^3 - 5.0614411200877u^2 + 101.0449674666u - 671.351971603004$$

$$\Psi_1(u) = 0.0129196424100381u^3 - 0.0385118820961294u^2 - 0.0449018505818788u + 0.29049409026797$$
$$\Psi_2(u) = -0.015092320199554u^3 + 0.129559893561424u^2 - 0.381045384015591u + 0.51458975538192$$
$$\Psi_3(u) = 0.00744966908457009u^3 - 0.0733180099956936u^2 + 0.227588313840762u - 0.0940439168444352$$
$$\Psi_4(u) = -0.00470637312605011u^3 + 0.0725544965317488u^2 - 0.355901708096683u + 0.683942767945957$$
$$\Psi_5(u) = 0.00137584040695409u^3 - 0.0186787064633142u^2 + 0.100264294063634u - 0.0763338596045747$$
$$\Psi_6(u) = -0.000797010853508062u^3 + 0.0204326162250046u^2 - 0.134403632529536u + 0.393001955434792$$
$$\Psi_7(u) = 0.00181220300707815u^3 - 0.0343608748473059u^2 + 0.249150814513381u - 0.501958465503483$$
$$\Psi_8(u) = -0.00645177137248217u^3 + 0.163974510262142u^2 - 1.33753228662778u + 3.72919657895604$$
$$\Psi_9(u) = 0.0139948324149489u^3 - 0.388083791998497u^2 + 3.63099246352029u - 11.1763778503021$$
$$\Psi_{10}(u) = -0.029527517756155u^3 + 0.917586713134619u^2 - 9.42571259853971u + 32.3459724280901$$
$$\Psi_{11}(u) = 0.0241152552989715u^3 - 0.852624797684555u^2 + 10.0466139930531u - 39.0525582066838$$
$$\Psi_{12}(u) = -0.00693356066019014u^3 + 0.265132576845265u^2 - 3.36647447150244u + 14.5997954131197$$
$$\Psi_{13}(u) = 0.00361901714411142u^3 - 0.146417957522496u^2 + 1.98368247527845u - 8.58421802293079$$
$$\Psi_{14}(u) = -0.00754247811393316u^3 + 0.322364843315376u^2 - 4.57927676625408u + 22.042925382376$$
$$\Psi_{15}(u) = 0.00655082497814037u^3 - 0.311833795827933u^2 + 4.9337028614267u - 25.5219731613395$$
$$\Psi_{16}(u) = 0.00133924853485252u^3 - 0.0616781265501159u^2 + 0.931212123179315u - 4.1753555727953$$
$$\Psi_{17}(u) = -0.0119078489198728u^3 + 0.613923843640878u^2 - 10.5540213700676u + 60.9076342222704$$
$$\Psi_{18}(u) = 0.0162921459525459u^3 - 0.908875879469737u^2 + 16.8563736471156u - 103.554735895134$$
$$\Psi_{19}(u) = -0.0232607635005404u^3 + 1.34563995935619u^2 - 25.9794272607746u + 167.738669477342$$
$$\Psi_{20}(u) = 0.0167509652700748u^3 - 1.05506376688073u^2 + 22.0346472365455u - 152.355160139217$$

拟合出的系数 b 和回归方程如下：

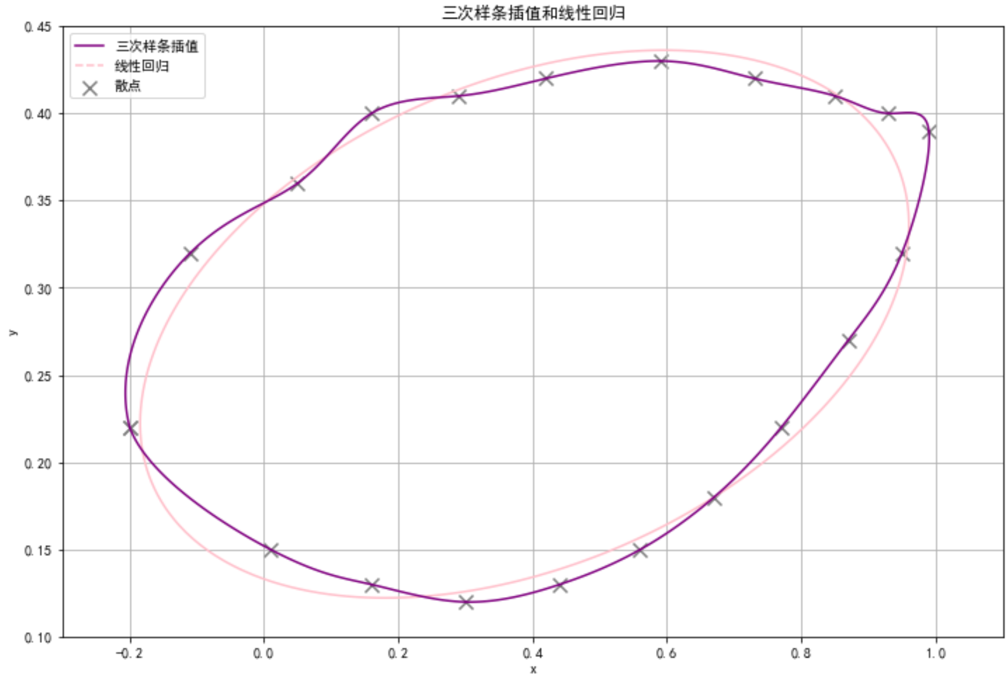
$$b_0 = -0.61675402721062844$$
$$b_1 = 0.037372895747393553$$
$$b_2 = 6.4038200357358814$$
$$b_3 = 2.6440756536123722$$
$$b_4 = -13.302206388291150$$
$$x^2 = -0.616754 + 0.0373729x + 6.40382y + 2.64408xy - 13.3022y^2$$

拟合函数的误差平方和Q，复相关系数R如下：

$$Q = 0.018993061877819149$$
$$R = 0.99573520194623211$$

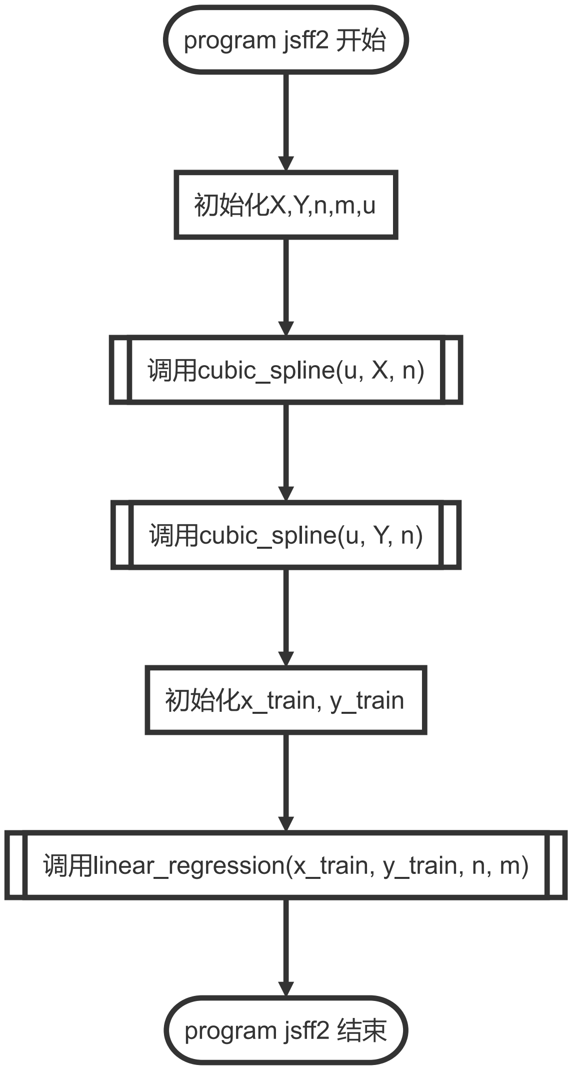
可以看到Q较小，R很接近1，说明拟合效果较好.

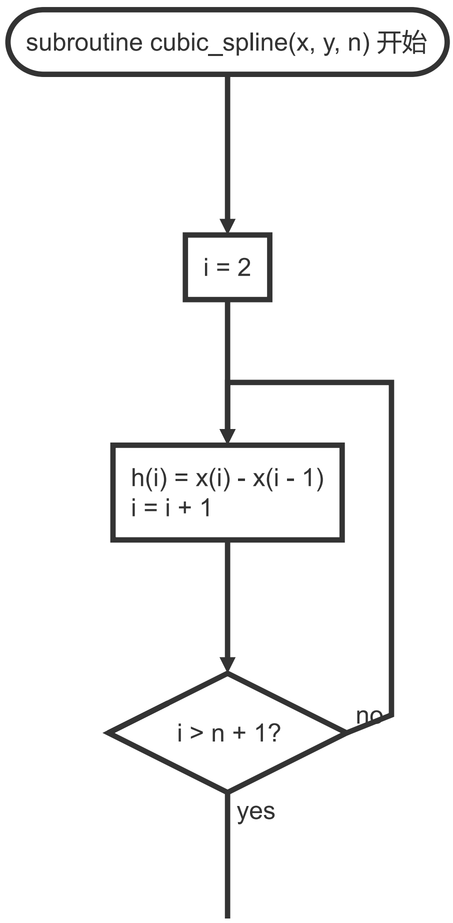
(2)曲线图

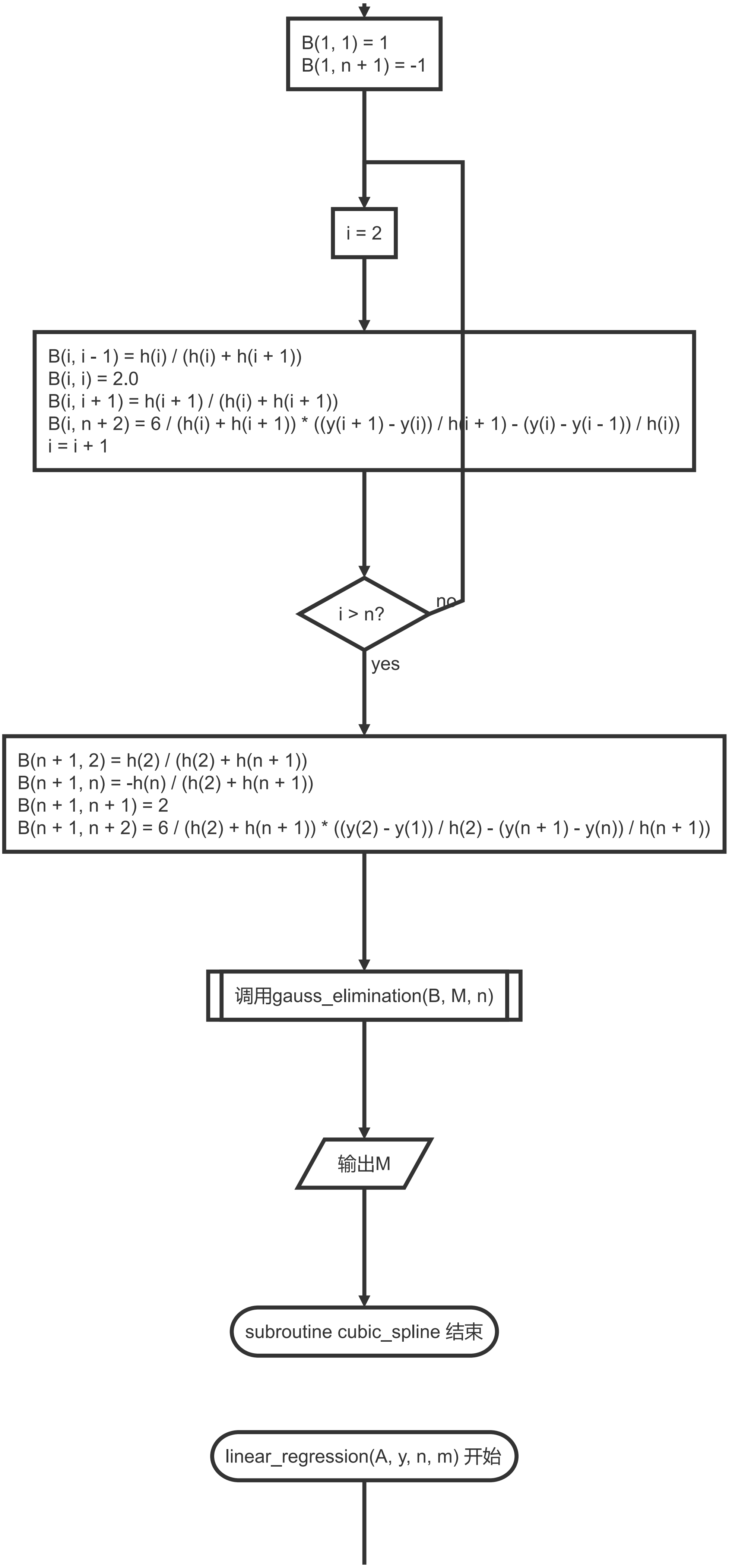


从上图可以看出，插值函数经过了每一个样本点，形状较为曲折，在训练集中的误差平方和为0；拟合函数为一个椭圆，刻画了样本点的分布趋势，在训练集中的误差平方和不为0.

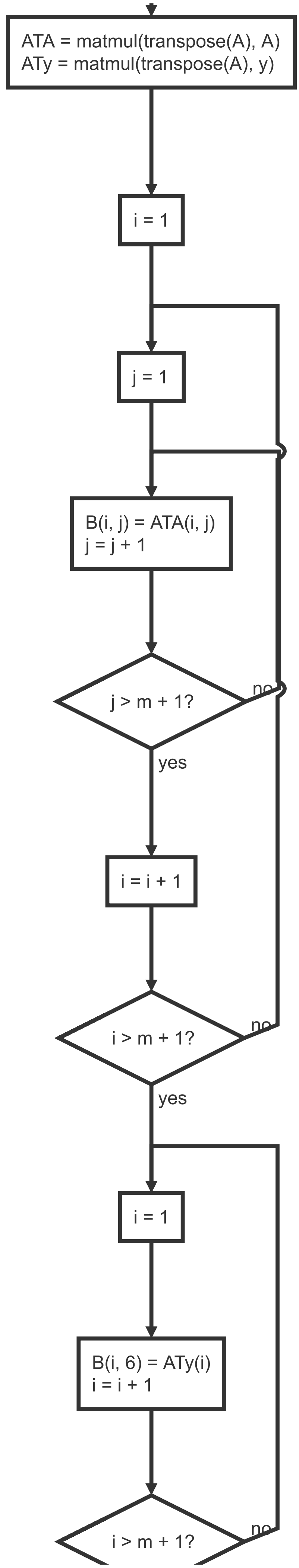
二、编程流程图


program jsff2 开始
初始化X,Y,n,m,u
调用cubic_spline(u, X, n)
调用cubic_spline(u, Y, n)
初始化x_train, y_train
调用linear_regression(x_train, y_train, n, m)
program jsff2 结束

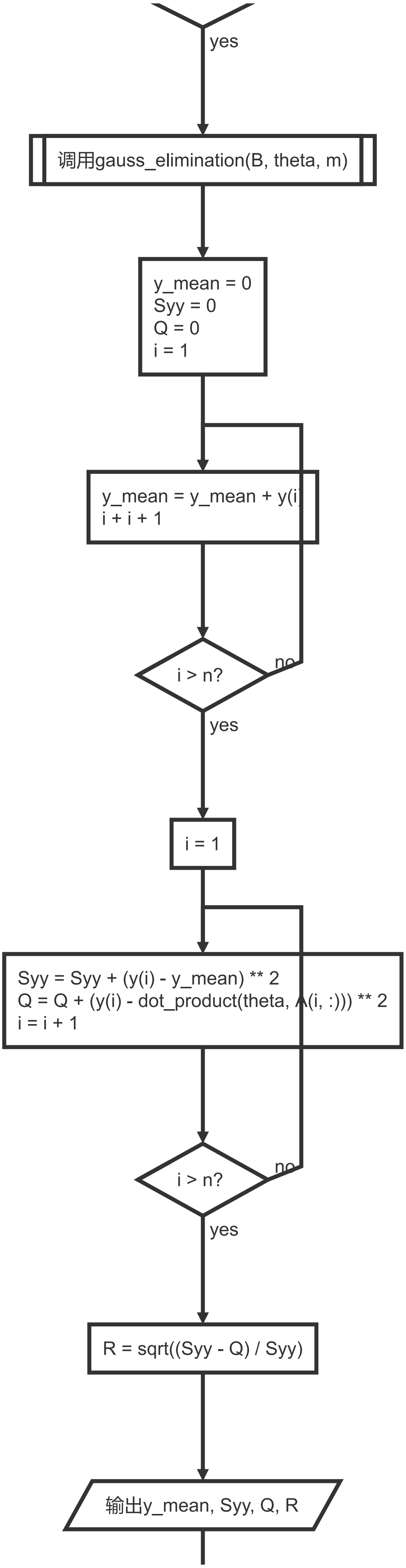

subroutine cubic_spline(x, y, n) 开始
i = 2
h(i) = x(i) - x(i - 1)
i = i + 1
i > n + 1?
no
yes

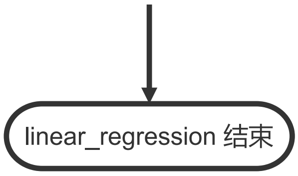


B(1, 1) = 1
B(1, n + 1) = -1
i = 2
B(i, i - 1) = h(i) / (h(i) + h(i + 1))
B(i, i) = 2.0
B(i, i + 1) = h(i + 1) / (h(i) + h(i + 1))
B(i, n + 2) = 6 / (h(i) + h(i + 1)) * ((y(i + 1) - y(i)) / h(i + 1) - (y(i) - y(i - 1)) / h(i))
i = i + 1
i > n?
no
yes
B(n + 1, 2) = h(2) / (h(2) + h(n + 1))
B(n + 1, n) = -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))
调用gauss_elimination(B, M, n)
输出M
subroutine cubic_spline 结束
linear_regression(A, y, n, m) 开始



ATA = matmul(transpose(A), A)
ATy = matmul(transpose(A), y)
i = 1
j = 1
B(i, j) = ATA(i, j)
j = j + 1
j > m + 1?
no
yes
i = i + 1
i > m + 1?
no
yes
i = 1
B(i, 6) = ATy(i)
i = i + 1
i > m + 1?
no


yes
调用gauss_elimination(B, theta, m)
y_mean = 0
Syy = 0
Q = 0
i = 1
y_mean = y_mean + y(i)
i + i + 1
i > n?
no
yes
i = 1
Syy = Syy + (y(i) - y_mean) ** 2
Q = Q + (y(i) - dot_product(theta, A(i, :))) ** 2
i = i + 1
i > n?
no
yes
R = sqrt((Syy - Q) / Syy)
输出y_mean, Syy, Q, R



三、源代码

Fortran主程序

```fortran
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

    implicit none
    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

    implicit none
    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
    ! y_mean : mean value of y, Syy : variance of y, Q : sum of squared error (SSE), R : multiple correlation coefficient
    real(8) :: y_mean, Syy, Q, R
    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

    ! calculate y_mean, Syy, Q, R
    y_mean = 0
    Syy = 0
    Q = 0

    do i = 1, n
        y_mean = y_mean + y(i)
    end do
    y_mean = y_mean / dble(n)

    do i = 1, n
        Syy = Syy + (y(i) - y_mean) ** 2
        Q = Q + (y(i) - dot_product(theta, A(i, :))) ** 2
    end do

    R = sqrt((Syy - Q) / Syy)

    print *, 'y_mean :', y_mean
    print *, 'Syy :', Syy
    print *, 'Q :', Q
    print *, 'R :', R

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)
```

```fortran
    ! author: zzy

    implicit none
    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

    implicit none
    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

```python
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
import sympy as sy
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计算出的线性拟合参数b
n = 20
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]
```

```python
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]

# 通过Mx,My计算各个区间上的分段插值函数
U = sy.symbols('U')
xu, yu= [], [] # 储存分段擦绘制函数
xu_coeffs, yu_coeffs = [], [] # 储存分段插值函数的系数
for i in range(1, n + 1):
    phi = sy.simplify(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))
    psi = sy.simplify(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))
    phi = sy.Poly(phi)
    psi = sy.Poly(psi)
    xu.append(np.poly1d(phi.coeffs()))
    yu.append(np.poly1d(psi.coeffs()))
    xu_coeffs.append(phi.coeffs())
    yu_coeffs.append(psi.coeffs())

# 开始画图，在一张图中绘制插值曲线和拟合曲线
fig, ax = plt.subplots(figsize=(12, 8))

# 分段绘制插值曲线
for i in range(n):
    u = np.linspace(i + 1, i + 2, 100)
    if i == n - 1:
        plt.plot(xu[i](u), yu[i](u), color='purple', label='三次样条插值')
    else:
        plt.plot(xu[i](u), yu[i](u), color='purple')

# 利用等高线绘制拟合曲线
xx = np.linspace(-0.3, 1.1, 100)
yy = np.linspace(0.1, 0.45, 100)
xx, yy = np.meshgrid(xx, yy)
zz = theta[0] + theta[1] * xx + theta[2] * yy + theta[3] * np.multiply(xx, yy) + theta[4] * np.multiply(yy, yy) - np.multiply(xx, xx)
CS = ax.contour(xx, yy, zz, 0, colors='pink')
CS.collections[0].set_label('线性回归')

# 绘制散点图
ax.scatter(x, y, c='gray', marker='x', s=100, label='散点')

# 图片细节调整
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

```
shenye@shenye-virtual-machine:~/FortranPrograms$ gfortran jsff2.f90 -o jsff2 && ./jsff2
 Mx :
  0.50709311714530891      -0.22746861616752340        4.2781310867928156E-002  -3.6565142043757694E-003  -2.8155432864359405E-002
  -3.7216398972686384E-003  -1.6957950326107057E-002   1.1553140794287204E-002  -2.9254255223173108E-002  -1.4536363088545449E-002
 -0.15260006354080916        2.4936474200634585E-002  -6.7145718820811245E-002   3.6462723365777061E-003  -6.7439613712450261E-002
   8.6112532988534593E-002  -3.7010568309589723E-002   6.1929740249824290E-002  -9.0708395371916425E-002   9.0381441575124177E-004
  0.50709311714530891
 My :
  0.54284566866118333       -7.2469376492327445E-002   2.9155454261828601E-002   2.0614643203788051E-002   7.1090710492157285E-003
   1.1291279168918538E-002   7.6340818456360699E-003   1.8196899987278390E-002  -2.0428142040654965E-002   6.3517252959952766E-002
 -0.11364127798094577        3.1048150093328578E-002  -1.0551535134211474E-002   1.1157999534473027E-002  -3.4080286625660552E-002
   5.1629044339197580E-003   1.3428847529025081E-002  -5.8878294503158242E-002   4.2084323318493912E-002 -0.10945917758138719
   3.5752551515874427E-002
 b : -0.61675402721062844        3.7372895747393553E-002   6.4038200357358814        2.6440756536123722       -13.302206388291150

 y_mean :   0.35566500194840145
 Syy :    2.2314830588593790
 Q :    1.8993061877819149E-002
 R :   0.99573520194623211
```

结果解释：

Mx为 x 对 u 插值所得到的各个区间上的分段插值函数的二阶导数值.

My为 y 对 u 插值所得到的各个区间上的分段插值函数的二阶导数值.

根据 Mx 可以构造出 x-u 插值函数$x = \Phi(u)$，根据 M 可以构造出 y-u 插值函数$y = \Psi(u)$.

b 为向量$(b_0, b_1, b_2, b_3, b_4)^T$，是线性回归的拟合方程$b_0 + b_1 x + b_2 y + b_3 xy + b_4 y^2 = x^2$的系数.

y_mean为目标变量(x^2)的平均值，Syy为目标变量的总平方和（SST），Q为目标变量的误差平方和（SSE），R为复相关系数.