摘　　要

在本次数值逼近课程设计中，围绕数值逼近中的若干典型问题，研究了 Runge 现象、D2 样条、最小二乘法以及二分法四个方面的问题，并对相关问题进行了求解与结果分析。这些理论与方法在数值计算与工程应用中都具有重要意义，为实际问题中的数值逼近、参数估计及方程求解提供了有效手段。

Runge 现象是指在利用多项式插值逼近函数时，随着插值多项式函数次数的增加，插值多项式在区间端点附近出现明显振荡，反而降低了逼近精度。本文选取函数 $f(x)=\frac{1}{1+25x^2}$，构造不同次数的 Lagrange 插值多项式，并通过绘图对插值效果进行对比与分析。

为克服 Runge 现象，本文采用三次样条插值方法构造整体光滑且次数不超过三次的分段多项式。在满足插值条件的同时，引入区间端点二阶导数为零的自然边界条件，利用三弯矩方程求解样条函数，从而获得具有良好光滑性的插值结果。

最小二乘法通过最小化误差平方和寻找数据的最优拟合函数，广泛用于曲线拟合和参数估计问题。本文针对化学反应生成物浓度随时间变化的数据，采用 $\mathrm{y}=a+bt+ct^2$ 的非线性最小二乘法来得到最佳近似的二次多项式。

科学与工程计算中很多问题的求解最终可归结为方程的求解，然而待求解方程常常为超越方程，难以求出其精确解。本文通过不断把函数 $f(x)=e^{-x}-\sin\frac{\pi x}{2}$ 的有根区间一分为二，使区间的两个端点逐步逼近零点，进而得到零点的近似值。

关键词： Runge 现象，D2 样条，最小二乘法，二分法

ABSTRACT

In this numerical approximation course project, several classical problems in numerical approximation were investigated, including the Runge phenomenon, D2 spline interpolation, the least squares method, and the bisection method. Corresponding problems were solved and the results were analyzed. These theories and methods play an important role in numerical computation and engineering applications, providing effective tools for numerical approximation, parameter estimation, and equation solving in practical problems.

The Runge phenomenon refers to the situation that when polynomial interpolation is used to approximate a function, increasing the degree of the interpolating polynomial may lead to significant oscillations of the interpolating polynomial near the endpoints of the interval, thereby reducing the approximation accuracy. In this study, the function $f(x)=\frac{1}{1+25x^2}$ was selected, and Lagrange interpolation polynomials of different degrees were constructed. The interpolation results were compared and analyzed through graphical visualization.

To overcome the Runge phenomenon, cubic spline interpolation was employed to construct a piecewise polynomial that is globally smooth and of degree no higher than three. While satisfying the interpolation conditions, natural boundary conditions with zero second derivatives at the endpoints were imposed. The spline function was then obtained by solving the tridiagonal system derived from the moment equations, resulting in an interpolation with good smoothness.

The least squares method determines the optimal fitting function by minimizing the sum of squared errors and is widely used in curve fitting and parameter estimation. In this work, experimental data describing the variation of product concentration with time in a chemical reaction were fitted using a nonlinear least squares approach, yielding the best approximate quadratic polynomial of the form $\mathrm{y}=a+bt+ct^2$.

In scientific and engineering computations, many problems can ultimately be reduced to solving equations; however, the equations involved are often transcendental and lack closed-form solutions. In this study, the bisection method was applied to repeatedly bisect the root-containing interval of the function $f(x)=e^{-x}-\sin\frac{\pi x}{2}$, so that the endpoints of the interval gradually converge to the root, and an approximate numerical solution of the root was obtained.

KEY WORDS: Runge phenomenon, D2 spline, Least squares method, Bisection method

目录

第一章 Runge 现象 1
1.1 问题描述 1
1.2 求解方法 1
1.3 结果分析 1
1.4 代码 2
第二章 D2 样条 4
2.1 问题描述 4
2.2 求解方法 4
2.3 结果分析 6
2.4 代码 7
第三章 最小二乘法 9
3.1 问题描述 9
3.2 求解方法 9
3.3 结果分析 10
3.4 代码 11
第四章 二分法 13
4.1 问题描述 13
4.2 求解方法 13
4.3 结果分析 14
4.4 代码 15
参考文献 17

第一章 Runge 现象

1.1 问题描述

Runge 现象是数值逼近中多项式插值的一种典型问题，指在利用高次多项式对函数进行插值逼近时，随着插值多项式次数的增加，插值多项式在区间端点附近会出现明显的振荡，反而降低了对原函数的逼近精度。

函数 $f(x)=\frac{1}{1+25x^2}$，$x\in[-1,1]$ 连续，各阶导数均存在，在 $[-1,1]$ 上 $x_i=-1+\frac{2}{n}i$，$i=0,1,2,\cdots,n$，$L_n(x)=\sum_{i=0}^{n}\frac{1}{1+25x_i^2}\frac{\omega(x)}{(x-x_i)\omega'(x_i)}$，考虑 $n=4$，$n=8$ 和 $n=12$ 三种情况。

1.2 求解方法

对于确定的 $n=4,8,12$，可得等距的 $n+1$ 个插值节点 $x_i=-1+\frac{2}{n}i$，$i=0,1,2,\cdots,n$，在区间 $[-1,1]$ 上构造 Lagrange 插值函数 $L_n(x)$，满足以下条件：

$$L_n(x_i)=f(x_i),\ i=0,1,2,\cdots,n, \tag{1.1}$$

$$L_n(x)=\sum_{i=0}^{n}\frac{1}{1+25x_i^2}\frac{\omega(x)}{(x-x_i)\omega'(x_i)}, \tag{1.2}$$

其中 $\omega(x)$ 满足 $\omega(x)=(x-x_0)(x-x_1)\cdots(x-x_n)$.

对 $f(x),L_4(x),L_8(x),L_{12}(x)$ 在同一图中进行可视化，对比和分析结果。

1.3 结果分析

对比分析图 1-1 我们可以发现：

（1）函数振荡：在插值多项式的次数较高时，插值函数在插值节点附近，特别是区间端点附近出现较为明显的振荡，这种振荡会导致插值函数在插值点附近剧烈波动，对原函数的近似程度降低。

（2）过度拟合：高次插值多项式可能会产生过度拟合，在插值点处函数拟合效果非常好，但在其他位置却大幅度偏离原函数。高次插值多项式无法体现出原函数的整体趋势和特征。

（3）边界效应：Runge 现象在区间端点处最明显。由于在区间端点附近的插值点较少，高次多项式难以准确地体现原函数在区间端点的变化趋势，从而产生较大的误差。

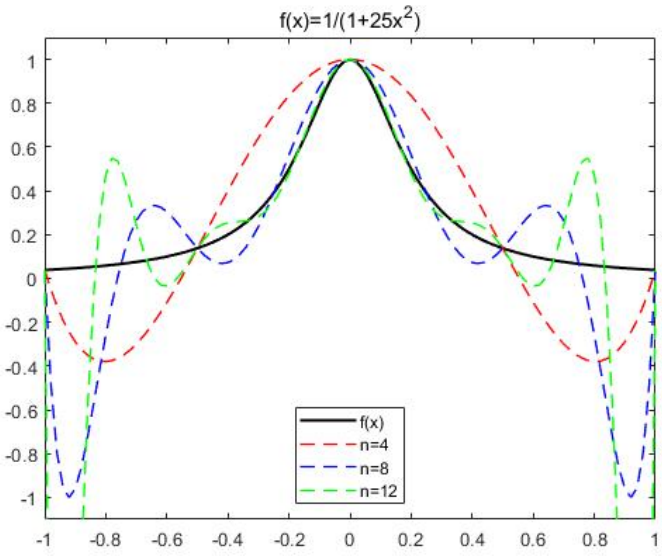

图 1-1 多项式插值 Runge 现象

为克服 Runge 现象，可以采用不等距的插值节点或分段低次多项式插值（例如：三次 Hermite 插值，三次样条插值）的方法来减少因为次数过高导致的振荡影响。

1.4 代码

采用 MATLAB R2023b 软件进行编程，源代码如下：

```
clear;clc;
f=@(x)(1./(1+25*x.^2)); % 原函数
N=[4 8 12]; % 插值多项式次数
X=linspace(-1,1,100+1);
plot(X,f(X),'k-','LineWidth',1.5); % 画原函数
hold on;
for i=1:3
    x=linspace(-1,1,N(i)+1);
    Ln=interp_L(x,f,X); % Lagrange 插值
    if i==1
        plot(X,Ln,'r--','LineWidth',1); % 画插值函数
    elseif i==2
        plot(X,Ln,'b--','LineWidth',1); % 画插值函数
    else
        plot(X,Ln,'g--','LineWidth',1); % 画插值函数
    end
    hold on;
end
xlim([-1,1]);
xticks(-1:0.2:1);
ylim([-1.1,1.1]);
legend('f(x)','n=4','n=8','n=12','location','south')
title('f(x)=1/(1+25x^2)');
```

```
24.
25.
26.
27.function Ln=interp_L(x,f,X) % 计算 Lagrange 插值
28.     n=length(x);
29.     nn=length(X);
30.     for k=1:nn
31.         L=0;
32.         for i=1:n
33.             l=1;
34.             for j=1:n
35.                 if i~=j
36.                     l=l*(X(k)-x(j))/(x(i)-x(j)); %l_k(x)
37.                 end
38.             end
39.             L=L+f(x(i))*l;
40.         end
41.         Ln(k)=L;
42.     end
43.end
```

第二章 D2 样条

2.1 问题描述

三次 Spline 插值是一种常用的分段多项式插值方法。它将插值区间划分为若干子区间，在每个子区间上构造次数不超过三次的多项式，并在各插值节点处保证函数值及一阶、二阶导数的连续性，从而获得整体光滑的插值函数。与高次多项式插值相比，三次 Spline 插值能够有效避免 Runge 现象，同时具有较高的插值精度和良好的数值稳定性。根据端点条件的不同，三次样条插值可分为 D1 样条、D2 样条以及周期样条。其中自然样条是 D2 样条的特殊情况，通过设定区间端点处二阶导数为零，使插值曲线在端点处更加平滑。

求满足如下条件的三次 Spline 插值函数，其中边界条件为自然边界条件。

表 2-1 插值节点及其函数值

x	1	2	4	5
$f(x)$	1	3	4	2

2.2 求解方法

利用节点处的二阶导数值来求解 D2 样条 $S(x)$：

假设 $S(x)$ 的二阶导数值 $S''(x_j)=M_j$，$j=0,1,2,\cdots,n$ 已知，由于 $S''(x)$ 在 $\left[x_j,x_{j+1}\right]$ 上为一个线性函数，又知 $S''(x_j)=M_j$，$S''(x_{j+1})=M_{j+1}$，因此可以由线性插值得到

$$S''(x)=\frac{x_{j+1}-x}{x_{j+1}-x_j}M_j+\frac{x-x_j}{x_{j+1}-x_j}M_{j+1}, \tag{2.1}$$

对(2.1)式连续积分两次，则得到含有两个积分常数的 Spline 函数 $S_j(x)$，再由插值条件 $S(x_j)=y_j$，$S(x_{j+1})=y_{j+1}$，可以确定两个积分常数，从而得到区间 $\left[x_j,x_{j+1}\right]$ 上的用二阶导数值 M_j 表示的 Spline 函数 $S_j(x)$，即

$$S(x)=\frac{(x_{j+1}-x)^3}{6(x_{j+1}-x_j)}M_j+\frac{(x-x_j)^3}{6(x_{j+1}-x_j)}M_{j+1}$$
$$+(y_j-\frac{(x_{j+1}-x_j)^2M_j}{6})\frac{x_{j+1}-x}{x_{j+1}-x_j}+(y_{j+1}-\frac{(x_{j+1}-x_j)^2M_{j+1}}{6})\frac{x-x_j}{x_{j+1}-x_j}, \tag{2.2}$$
$$x\in\left[x_j,x_{j+1}\right],j=0,1,\cdots n-1.$$

由此可见只需要得到$S(x)$的二阶导数值$S''(x_j)=M_j$，即可由(2.2)式求得$S(x)$，因此问题转化为求M_j.

假设(2.2)式已知，对其求一阶导得

$$S'(x)=-\frac{(x_{j+1}-x)^2}{2(x_{j+1}-x_j)}M_j+\frac{(x-x_j)^2}{2(x_{j+1}-x_j)}M_{j+1}+\frac{y_{j+1}-y_j}{x_{j+1}-x_j}-\frac{M_{j+1}-M_j}{6}(x_{j+1}-x_j). \tag{2.3}$$

再由$S'(x)$的连续性可知在x_j点处有

$$S'(x_j+0)=S'(x_j-0),j=1,2,\cdots,n-1, \tag{2.4}$$

其中

$$S'(x_j+0)=S'_j(x_j)=-\frac{x_{j+1}-x_j}{3}M_j-\frac{x_{j+1}-x_j}{6}M_{j+1}+\frac{y_{j+1}-y_j}{x_{j+1}-x_j}, \tag{2.5}$$

$$S'(x_j-0)=S'_{j-1}(x_j)=\frac{x_j-x_{j-1}}{6}M_{j-1}+\frac{x_j-x_{j-1}}{3}M_j+\frac{y_j-y_{j-1}}{x_j-x_{j-1}}, \tag{2.6}$$

将(2.5)式和(2.6)式代入(2.4)式，化简得到

$$\mu_jM_{j-1}+2M_j+\lambda_jM_{j+1}=d_j, \tag{2.7}$$

其中$\mu_j=\frac{x_j-x_{j-1}}{x_{j+1}-x_{j-1}}$，$\lambda_j=\frac{x_{j+1}-x_j}{x_{j+1}-x_{j-1}}$，$d_j=6f\left[x_{j-1},x_j,x_{j+1}\right]$，$j=1,2,\cdots,n-1$.

已知区间端点的二阶导数值$S''_0(x_0)=f''(x_0)=M_0$，$S''_{n-1}(x_n)=f''(x_n)=M_n$，代入(2.7)式，化简得到$n-1$阶的三弯矩方程

$$\begin{pmatrix}2&\lambda_1&0&&&0\\\mu_2&2&\lambda_2&&&\\&\mu_3&2&\ddots&&\\&&\ddots&\ddots&&\\&&&\mu_{n-2}&2&\lambda_{n-2}\\0&&&0&\mu_{n-1}&2\end{pmatrix}\begin{pmatrix}M_1\\M_2\\\vdots\\\vdots\\M_{n-2}\\M_{n-1}\end{pmatrix}=\begin{pmatrix}6f\left[x_0,x_1,x_2\right]-\mu_1M_0\\6f\left[x_1,x_2,x_3\right]\\\vdots\\\vdots\\6f\left[x_{n-3},x_{n-2},x_{n-1}\right]\\6f\left[x_{n-2},x_{n-1},x_n\right]-\lambda_{n-1}M_n\end{pmatrix}, \tag{2.8}$$

求解三弯矩方程，得$S(x)$的二阶导数值$S''(x_j)=M_j$，代入(2.2)式即得$S(x)$表达式。

2.3 结果分析

首先根据上一节的计算公式计算得到 M 值如下表：

表 2-2 D2 样条 S(x)的二阶导数值 M

M_0	M_1	M_2	M_3
0	$-\frac{3}{4}$	$-\frac{9}{4}$	0

后由(2.2)式计算得到 $S(x)$ 的分段解析式如下：

$$S(x)=\begin{cases}-\frac{1}{8}x^3+\frac{3}{8}x^2+\frac{7}{4}x-1, x\in[1,2]\\ -\frac{1}{8}x^3+\frac{3}{8}x^2+\frac{7}{4}x-1, x\in[2,4]\\ \frac{3}{8}x^3-\frac{45}{8}x^2+\frac{103}{4}x-33, x\in[4,5]\end{cases}. \tag{2.9}$$

再对符合自然边界条件的三次样条函数 $S(x)$ 进行可视化，结果如下图：

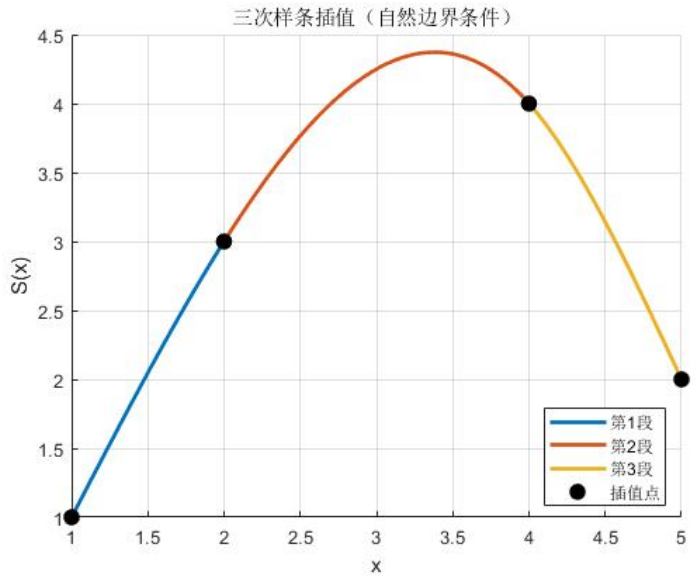

图 2-1 分段的三次样条函数 S(x)

根据图 2-1 可以看出样条函数 $S(x)$ 经过插值点，具有良好的光滑性，没有出现 Runge 现象。此外可以对不均匀插值点进行插值计算也是三次样条插值的另一优点。

但从(2.8)式可以看出，当插值规模较大时，求解线性方程组的计算量较大；并且插值结果与边界条件密切相关，需要对边界条件慎重选择。

2.4 代码

采用 MATLAB R2023b 软件进行编程，源代码如下：

```
clear;clc;
X=[1 2 4 5]'; % 原始数据 X
Y=[1 3 4 2]'; % 原始数据 Y
[D,h,A,g,M]=threespline(X,Y); % 求解三弯矩方程
syms x;
figure;hold on;grid on;
colors=lines(length(X)-1);% 不同颜色
for i=1:length(X)-1
    p1=(X(i+1)-x)^3*M(i)/(6*h(i));
    p2=(x-X(i))^3*M(i+1)/(6*h(i));
    p3=(Y(i)-M(i)*h(i)^2/6)*(X(i+1)-x)/h(i);
    p4=(Y(i+1)-M(i+1)*h(i)^2/6)*(x-X(i))/h(i);
    S=simplifyFraction(p1+p2+p3+p4) % 求出 S(x)表达式
    f=matlabFunction(S);
    xx=linspace(X(i), X(i+1), 200);
    yy=f(xx);
    plot(xx,yy,'LineWidth',2,'Color',colors(i,:)); % 绘制插值函数
end
plot(X,Y,'ko','MarkerFaceColor','k','MarkerSize',8); % 绘制插值点
xlabel('x');
ylabel('S(x)');
title('三次样条插值（自然边界条件）');
legend({'第 1 段','第 2 段','第 3 段','插值点'},'Location','best');
hold off;


function [D,h,A,g,M]=threespline(X,Y) % 求解三弯矩方程
    n=length(X);
    A=zeros(n,n);
    A(:,1)=Y;
    D=zeros(n-2);
    g=zeros(n-2,1);
    h=zeros(n-2,0);
    for j=2:n
        for i=j:n
            A(i,j)=(A(i,j-1)-A(i-1,j-1))/(X(i)-X(i-j+1)); % 差商表 A
        end
    end
    for i=1:n-2
        g(i,1)=6*A(i+2,3); % 右端向量 g
    end
    for i=1:n-1
```

```
43.         h(i)=X(i+1)-X(i); % 间隔 h
44.     end
45.     D=D+eye(n-2)*2; % 三弯矩矩阵 D
46.     for i=2:n-2
47.         D(i-1,i)=h(i)/(h(i-1)+h(i)); % lamuda_i
48.         D(i,i-1)=h(i)/(h(i)+h(i+1)); % miu_i
49.     end
50.     M=D\g; % 二阶导数值 M
51.     M=[0;M;0]; % 加入自然边界条件
52.end
```

第三章 最小二乘法

3.1 问题描述

最小二乘法是一种常用的数据拟合与参数估计方法，其基本思想是通过最小化观测数据与模型预测值之间误差的平方和，来确定最优的近似函数或参数。当实验数据中存在测量误差或无法用精确函数描述时，最小二乘法能够给出意义下的“最佳”拟合结果。

在某化学反应中，由实验得到的生成物浓度 y 与时间 t 如下表，欲求浓度 y 与时间 t 的函数关系 $y=a+bt+ct^2$.

表格 3-1 生成物浓度与时间的数据

t	1	2	3	4	6	8	10	12	14	16
y	4.00	6.41	8.01	8.79	9.53	9.86	10.33	10.42	10.53	10.61

3.2 求解方法

给定一组数据 $(x_i,y_i),i=0,1,\cdots,m$，设 $\Phi=span\{1,x,x^2,\cdots,x^n\}$，则对任意函数 $\varphi(x)\in\Phi,\varphi(x)=\sum_{k=0}^{n}a_kx^k$，需要 $S(a_0,a_1,\cdots,a_n)=\sum_{i=0}^{m}\left[\sum_{k=0}^{n}a_kx^k-y_i\right]^2$ 最小。由拉格朗日乘数法，求解 $S(a_0,a_1,\cdots,a_n)=\sum_{i=0}^{m}\left[y_i-\left(a_0+a_1x_i+\cdots+a_nx_i^n\right)\right]^2$ 的极小值问题等价于求解方程组

$$\frac{\partial S}{\partial a_i}=0,i=0,1,\cdots,n, \tag{3.1}$$

即

$$\begin{cases}\dfrac{\partial S}{\partial a_0}=2\sum_{i=0}^{m}\left[y_i-\left(a_0+a_1x_i+\cdots+a_nx_i^n\right)\right](-1)=0,\\ \dfrac{\partial S}{\partial a_1}=2\sum_{i=0}^{m}\left[y_i-\left(a_0+a_1x_i+\cdots+a_nx_i^n\right)\right](-x_i)=0,\\ \cdots\\ \dfrac{\partial S}{\partial a_n}=2\sum_{i=0}^{m}\left[y_i-\left(a_0+a_1x_i+\cdots+a_nx_i^n\right)\right](-x_i^n)=0,\end{cases} \tag{3.2}$$

化简可得

$$\begin{pmatrix} m+1 & \sum_{i=0}^{m} x_i & \sum_{i=0}^{m} x_i^2 & \cdots & \sum_{i=0}^{m} x_i^n \\ \sum_{i=0}^{m} x_i & \sum_{i=0}^{m} x_i^2 & \sum_{i=0}^{m} x_i^3 & \cdots & \sum_{i=0}^{m} x_i^{n+1} \\ \vdots & \vdots & \vdots & & \vdots \\ \sum_{i=0}^{m} x_i^n & \sum_{i=0}^{m} x_i^{n+1} & \sum_{i=0}^{m} x_i^{n+2} & \cdots & \sum_{i=0}^{m} x_i^{2n} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} \sum_{i=0}^{m} y_i \\ \sum_{i=0}^{m} x_i y_i \\ \vdots \\ \sum_{i=0}^{m} x_i^n y_i \end{pmatrix}. \tag{3.3}$$

求解方程组(3.3)即可求得函数 $\varphi(x)$.

3.3 结果分析

设 $y = a + bt + cy^2$，利用(3.3)式建立法方程组，解得最优拟合参数：

表 3-1 最优拟合参数

a	b	c
4.1490	1.1436	-0.0483

即得最优拟合二次多项式为 $y = 4.1490 + 1.1436t - 0.0483t^2$.

对原始数据点和求得的二次多项式进行可视化，如下图：

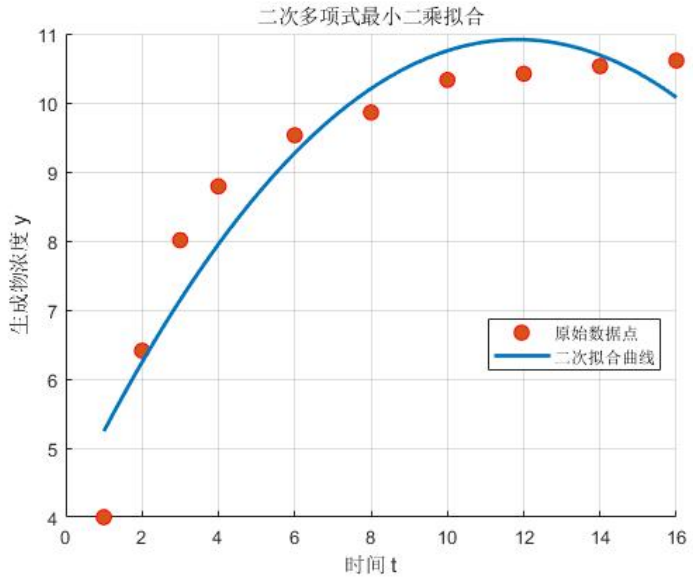

图 3-1 二次多项式最小二乘拟合

根据图 3-1 可以看出原始数据点近似于二次函数型，用二次多项式来拟合效果较好。求得的二次多项式很好地近似拟合了原始数据点，方便研究其变化趋势。

下面绘制原始数据与拟合函数的残差图来进行残差分析：

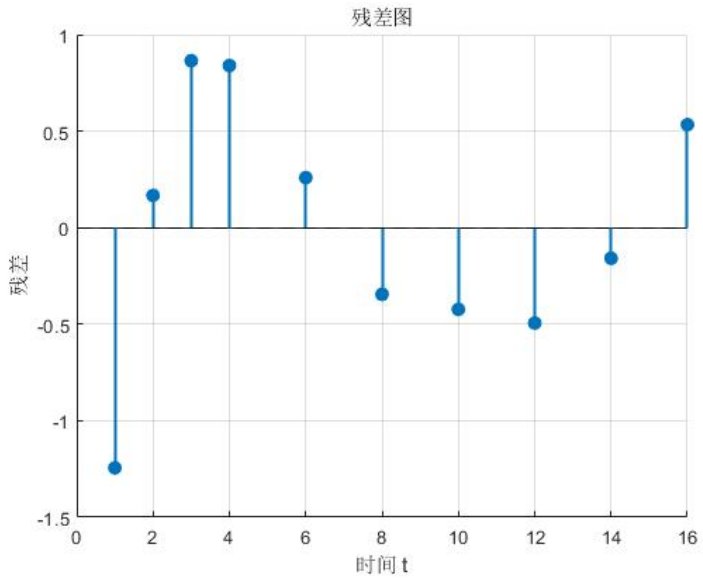

图 3-2 拟合函数残差图

由图 3-2 可以分析出用二次多项式来拟合效果不错，具有较小的残差，并且在中央数据点的拟合效果更佳。若要进一步提高拟合精度，可以选取其他多项式函数或引入非线性函数模型来进行拟合。

3.4 代码

采用 MATLAB R2023b 软件进行编程，源代码如下：

```
clear;clc;
t=[1 2 3 4 6 8 10 12 14 16]'; % 原始数据点
y=[4.00 6.41 8.01 8.79 9.53 9.86 10.33 10.42 10.53 10.61]'; % 原始数据点
A0=0;A1=0;A2=0;A3=0;A4=0;
b1=0;b2=0;b3=0;
n=length(t);
for i=1:n
    A0=A0+t(i)^0;
    A1=A1+t(i)^1;
    A2=A2+t(i)^2;
    A3=A3+t(i)^3;
    A4=A4+t(i)^4;
    b1=b1+t(i)^0*y(i);
    b2=b2+t(i)^1*y(i);
    b3=b3+t(i)^2*y(i);
end
A=[A0 A1 A2;
   A1 A2 A3;
   A2 A3 A4]; % 法方程组
b=[b1 b2 b3]'; % 右端向量
```

```
M=A\b; %最优参数
p=flip(M) % 反转
P=poly2sym(p) % 二次函数


figure; hold on; grid on;
plot(t,y,'ro','MarkerSize',8,'MarkerFaceColor',[0.85 0.33 0.10]); % 原始数据点
tt=linspace(min(t), max(t), 300);
yy=polyval(p,tt); % 拟合曲线
plot(tt,yy,'b-','LineWidth',2,'Color',[0.00 0.45 0.74]);

xlabel('时间 t');
ylabel('生成物浓度 y');
title('二次多项式最小二乘拟合');
legend('原始数据点','二次拟合曲线','Location','best');
hold off;


figure;hold on; grid on;
y_fit=polyval(p,t); % 在数据点处的拟合值
res=y-y_fit; % 残差

stem(t,res,'filled','LineWidth',1.5,'Color',[0.00 0.45 0.74]);
yline(0, 'k--');
xlabel('时间 t');
ylabel('残差');
title('残差图');
hold off;
```

第四章 二分法

4.1 问题描述

科学与工程计算中很多问题的求解最终可归结为方程的求解，然而待求解方程常常为超越方程，难以求出其精确解。二分法是一种求解非线性方程数值解的基本方法，其基本思想是利用连续函数在有根区间内符号变化的性质，将包含零点的区间不断二分使区间长度逐渐缩小，从而得到方程零点的近似值。二分法具有算法简单、收敛性可靠的优点，适用于求解连续函数的单根问题，虽然收敛速度相对较慢，但在科学与工程计算中具有重要的实用价值。

用二分法求 $e^{-x}-\sin\frac{\pi x}{2}=0$ 在区间 $[0,1]$ 内的一个根，要求误差不超过 $\frac{1}{2^5}$.

4.2 求解方法

由零点存在性定理，若 $f(x)$ 连续，且在 $[a,b]$ 上有 $f(a)f(b)<0$，则 $f(x)$ 在 $[a,b]$ 上必有零点 x^* 存在，令 $x_0=\frac{a+b}{2}$，则

（1）若 $f(x_0)=0$，则 x_0 即为所求根；

（2）若 $f(a)f(x_0)>0$，则一定有 $f(x_0)f(b)<0$，从而 $[x_0,b]$ 为有根区间；

（3）若 $f(a)f(x_0)<0$，则 $[a,x_0]$ 为有根区间。

反复进行下去，则有 $[a,b]\supset[a_1,b_1]\supset[a_2,b_2]\supset\cdots\supset[a_n,b_n]\supset\cdots$，其中每个区间长度均为前一个区间长度的一半，记 $x_n=\frac{a_n+b_n}{2}$，则有

$$\left|x_n-x^*\right|=\left|\frac{a_n+b_n}{2}-x^*\right|\le\frac{1}{2}\left(\left|a_n-x^*\right|+\left|b_n-x^*\right|\right)=\frac{(b_n-a_n)}{2}=\frac{(b-a)}{2^{n+1}},\tag{4.1}$$

当 $n\to\infty$ 时，由(4.1)式得 $\left|x_n-x^*\right|\to 0$，即 $\lim_{n\to\infty}x_n=x^*$.

但在实际计算时需要给出迭代的终止条件，通常有以下三种常用终止条件：

（1）给定精度 $\delta>0$，若对 $x_k=\frac{a_k+b_k}{2}$ 有 $\left|f(x_k)\right|\le\delta$，则停止，取 $x^*\approx x_k=\frac{a_k+b_k}{2}$；

（2）给定精度 $\delta>0$，若有某个有根区间 $[a_k,b_k]$ 的长度满足 $b_k-a_k\le\delta$，则停止，取 $x^*\approx x_k=\frac{a_k+b_k}{2}$；

（3）对于给定的精度$\varepsilon>0$，若要求$\left|x^*-x_n\right|<\varepsilon$，则可以计算出所需二分的次数$n$.事实上，可以进行迭代次数估计，欲使，$\left|x^*-x_n\right|<\varepsilon$，只要$\frac{b-a}{2^{n+1}}<\varepsilon$，由此可得$n>\log_2(\frac{b-a}{\varepsilon})-1$.

本文采用第三种终止条件来对问题进行求解。

4.3 结果分析

根据上一节的求解方法，先计算得到终止条件$n=4$，后进行如下迭代过程：

表 4-1 二分法迭代过程

n	a_n	b_n	x_n	$f(x_n)$
0	0	1	0.5	-0.10058
1	0	0.5	0.25	0.39612
2	0.25	0.5	0.375	0.13172
3	0.375	0.5	0.4375	0.011255
4	0.4375	0.5	0.46875	-0.045775

取$x_4=\frac{a_4+b_4}{2}=0.4688$为$x^*$的近似值。

将函数可视化并标注近似值点和真实值所在区间：

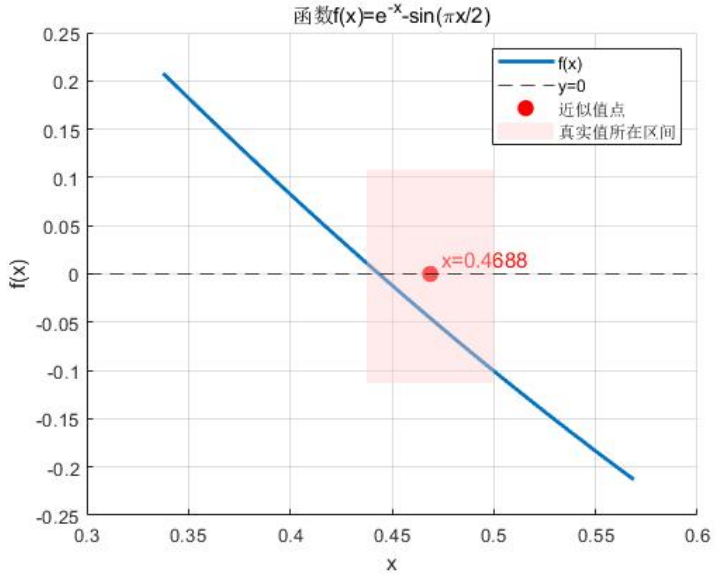

图 4-1 原始函数图像

通过对图像进行分析，我们可以看到函数零点落在以近似值点x_4为中心，以ε为半径的区间中，采用二分法可以正确求出数值解。

但可以看到二分法的收敛速度较慢，并且仍与真实值存在一定的差距。若想进一步得到更加精确的结果，可以通过减小允许的误差ε，或者采用其他收敛速度更快的数值解解法来实现。

4.4 代码

采用 MATLAB R2023b 软件进行编程，源代码如下：

```
clear; clc;
f=@(x) exp(-x)-sin(pi*x/2);
a=0;b=1;
tol=1/2^5;
n=log2((b-a)/tol)-1
n=ceil(n);
Record=zeros(n+1,5); % 每一行：[k,a_k,b_k,c_k,f(c_k)]
for k=0:n
    c=(a+b)/2; % 二分
    Record(k+1,:)=[k,a,b,c,f(c)]; % 记录迭代过程
    if f(a)*f(c)<0 % 根在左半区间
        b=c;
    elseif f(c)*f(b)<0 % 根在右半区间
        a=c;
    else % 找到真实值
        break;
    end
end
disp('二分法迭代过程：');
disp(array2table(Record,'VariableNames',{'n','a','b','c','f(c)'})); % 输出迭代过程
disp(['根的近似值为：', num2str(c)]); % 输出根近似值


figure; hold on; grid on; % 绘图
x=linspace(a-0.1,b+0.1,500);
y=f(x);
plot(x,y,'LineWidth',2); % 原函数

yline(0,'k--'); % y=0线

plot(c,0,'ro','MarkerSize',8,'MarkerFaceColor','r');% 零点标注
text(c,0,sprintf('  x=%.4f',c),'FontSize',11,'Color','r','VerticalAlignment','bottom');

x_patch=[c-tol,c+tol,c+tol,c-tol];
```

```
y_patch=[min(f(x))+0.1,min(f(x))+0.1,max(f(x))-0.1,max(f(x))-0.1];
patch(x_patch, y_patch,[1 0.8 0.8],'FaceAlpha',0.4,'EdgeColor','none'); % 绘制真实值所在区间

xlabel('x');
ylabel('f(x)');
title('函数 f(x)=e^{-x}-sin(\pix/2)');
legend('f(x)','y=0','近似值点','真实值所在区间','Location','northeast');
hold off;
```

参考文献

[1] 蒋尔雄, 赵风光, 苏仰锋. 数值逼近(第 2 版)(附光盘)[M]. 复旦大学出版社, 2008.
[2] 王晓峰, 王军涛. 数值逼近[M]. 河南大学出版社, 2018.