

第四次上机作业

Anonymity 3220100000*

* College of Control Science and Engineering, Robotics Engineering

July 8, 2024

1 问题描述

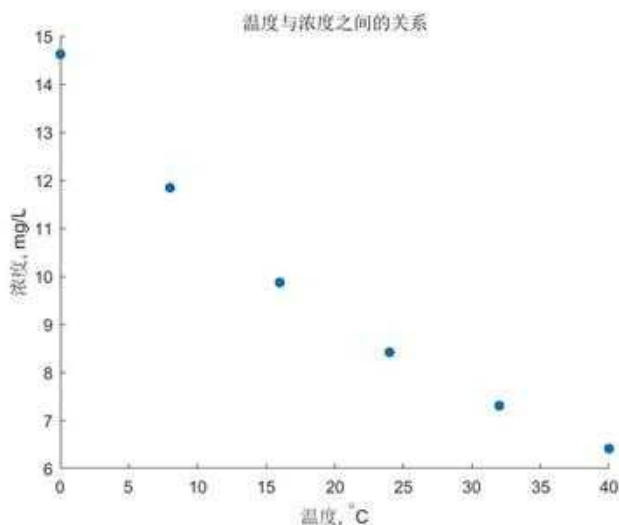
海平面下淡水中溶解氧的浓度是温度的函数, 如下表所示:

温度, °C	0	8	16	24	32	40
浓度, mg/L	14.621	11.843	9.870	8.418	7.305	6.413

利用上述数据, 确定 27°C 时的溶解氧浓度 (分别采用插值、拟合的方法求解并分析, 准确结果为 7.986mg/L)。

2 思路分析

首先绘制所给数据的散点图:



题目要求使用插值法和拟合法求解, 因此在求得插值或拟合函数后, 需将 27°C 条件代入求解, 并与真实值比较分析其准确性, 并对最终的结果进行比较。

2.1 拉格朗日插值法

对于给定的数据点 i , i 从 1 到 n , 其中 n 是数据点的数量计算拉格朗日基函数的分子, 即:

$$\text{numerator} = \prod_{\substack{j=1 \\ j \neq i}}^n (x - x_j)$$

其中, x_j 表示第 j 个数据点的温度值。

再对于每个数据点 i , 计算拉格朗日基函数的分母, 即:

$$\text{denominator} = \prod_{\substack{j=1 \\ j \neq i}}^n (x_i - x_j)$$

其中, x_i 表示第 i 个数据点的温度值。

将基函数的分子除以分母, 得到第 i 个拉格朗日基函数 $l_i(x)$, 将每个数据点的浓度值 y_i 乘以对应的基函数 $l_i(x)$, 并将它们相加, 得到拉格朗日插值多项式 $P(x)$ 。

由于所推数据点位于已知区域内, 属于内推, 因此拉格朗日插值法较为适合。

2.2 牛顿插值法

差商是用来构建插值多项式的重要步骤。给定数据点 (x_i, y_i) , 其中 x_i 表示温度, y_i 表示对应的溶解氧浓度。差商的计算使用递归公式:

$$f[x_i, x_{i-1}, \dots, x_{i-k}] = \frac{f[x_i, x_{i-1}, \dots, x_{i-k+1}] - f[x_{i-1}, x_{i-2}, \dots, x_{i-k}]}{x_i - x_{i-k}}$$

这里 $f[x_i, x_{i-1}, \dots, x_{i-k}]$ 表示 k 阶差商, 用于估计 $k-1$ 阶插值多项式的系数。初始时, 一阶差商就是数据点的斜率。

利用差商递归计算得到的系数构建插值多项式, 其形式为:

$$P(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \dots$$

其中, $P(x)$ 是插值多项式, $f[x_0]$ 是零阶差商, $f[x_0, x_1]$ 是一阶差商, 以此类推。

2.3 分段线性插值法

由于 *matlab* 中含有内置函数 *interp1*, 因此可直接调用。

假设有一组数据点 $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$, 分段线性插值将数据点按照 x 坐标的顺序排列, 并将相邻的数据点之间连线, 形成若干段线段。然后, 在每个相邻数据点之间的区间内, 使用线性插值的公式来计算介于这两个数据点之间的函数值。

例如, 在区间 $[x_1, x_2]$ 内, 可以使用线性插值公式来计算函数在该区间内的值:

$$y = y_1 + (x - x_1) \frac{(y_2 - y_1)}{(x_2 - x_1)}$$

其中, x_1 和 x_2 是相邻数据点的 x 坐标, y_1 和 y_2 分别是这两个数据点对应的 y 坐标, x 是我们估算的点的 x 坐标, y 是估算得到的点的 y 坐标。

2.4 二次样条插值

每个区间采用二次多项式 $f_i(x) = a_i x^2 + b_i x + c_i \quad x \in [x_{i-1}, x_i] \quad i = 1, 2, \dots, n$ 对于 $n+1$ 个数据点, 共有 n 个区间, $3n$ 个未知数。

相邻多项式在内部节点处函数值相等

$$a_i x_i^2 + b_i x_i + c_i = f(x_i) \quad a_{i+1} x_i^2 + b_{i+1} x_i + c_{i+1} = f(x_i) \quad i = 1, 2, \dots, n-1$$

第一和最后一个多项式通过端点

$$\begin{aligned} a_1 x_0^2 + b_1 x_0 + c_1 &= f(x_0) \\ a_n x_n^2 + b_n x_n + c_n &= f(x_n) \end{aligned}$$

内部节点的一阶导数相等

$$2a_i x_i + b_i = 2a_{i+1} x_i + b_{i+1} \quad i = 1, 2, \dots, n-1$$

在第一个节点处二阶导数为 0

$$a_1 = 0$$

连接前两点的为一条直线。

2.5 三次样条插值

三次样条插值的原理是基于曲线在每个节点处的曲率以及其一阶导数的连续性来确定曲线的形状。

具体来说, 三次样条插值将问题转化为求解一个线性方程组。在这个方程组中, 未知数是节点处的弯矩, 通过给定的边界条件 (通常是边界处的弯矩为零) 来确定。通过解这个方程组, 我们可以得到每个节点处的弯矩值。然后, 通过这些弯矩值以及节点处的函数值来构造出一个三次插值多项式, 从而得到了插值曲线。

2.6 逆插值法

逆插值法是指利用初始数据进行插值, 得到插值多项式后, 寻找对应的 x 值, 相当于方程求根。本题逆插值法采用线性插值法。

2.7 Hermite 插值

Hermite 插值多项式求解的基本思想是既然函数在插值点的值等于函数值与原函数的函数值相同, 这与 *Lagrange* 多项式的插值是相同的, 于是便运用了 *Lagrange* 插值方法得到的插值多项式函数, 为了保证插值多项式满足导数约束条件, 于是我们引入了新的项 $G(X)$, 具体的数学语言表达如下: $H_{2n+1}(x) = L_n(x) + G(x)$, 其中 $L_n(x)$ 为满足

$L_n(x_i) = f(x_i) \quad (i = 0, 1, \dots, n)$ 的 n 次 *Lagrange* 插值多项式, $G(x) = P_n(x) \cdot \omega_{n+1}(x)$, $P_n(x)$ 为待定的 n 次多项式。

然后根据导数的要求

$H'_{2n+1}(x_i) = L'_n(x_i) + P_n(x_i) \cdot \omega'_{n+1}(x_i) = f'(x_i), i = 0, 1, \dots, n$ 由此再求出 $P_n(x)$ 带入便可以得到插值多项式。

2.8 线性拟合

观察原数据点的散点图，发现线性程度较大，因此可采用线性拟合。代码实现可直接调用 *matlab* 内置函数。

2.9 二次拟合

原数据点单调，可以采用二次拟合进行。同样可直接调用 *matlab* 内置函数。

2.10 双曲线型拟合

原数据点单调，可以采用双曲线进行拟合：

$$\varphi(x) = 1/(ax + b)$$

同样可直接调用 *matlab* 内置函数。

2.11 反比例二次函数拟合

原数据点单调，可以采用反比例二次函数进行拟合：

$$\varphi(x) = 1/(a + bx^2)$$

同样可直接调用 *matlab* 内置函数。

2.12 指数函数拟合

原数据点单调，可以采用指数函数进行拟合：

$$\varphi(x) = ae^{-bx}$$

同样可直接调用 *matlab* 内置函数。

2.13 对数函数拟合

原数据点单调，可以采用指数函数进行拟合：

$$\varphi(x) = a \ln(x) + b$$

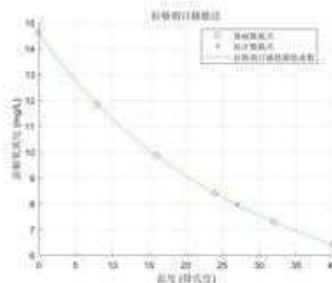
同样可直接调用 *matlab* 内置函数。

3 运行结果与分析

3.1 拉格朗日插值法

运行结果如下：

```
命令窗口
% 输出结果
disp(['在 ', num2str(x), ' 度时的溶解氧浓度为 ', num2str(y), ' mg/L'])
disp(['相对误差为 ', num2str(relative_error), '%'])
在 27 度时的溶解氧浓度为 7.968 mg/L
相对误差为 0.222402200524500%
fx >>
```

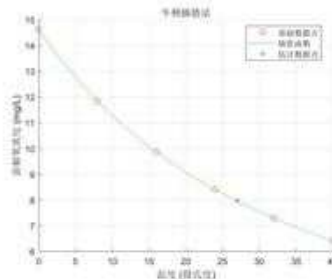


本方法选择合适的多项式次数，可以得到较好的结果。

3.2 牛顿插值法

运行结果如下：

```
命令窗口
ylabel('溶解氧浓度 (mg/L)');
title('牛顿插值法');
legend('Location', 'best');
grid on;
在 27 度时的溶解氧浓度为 7.968 mg/L
相对误差为 0.222402200524511%
fx >>
```

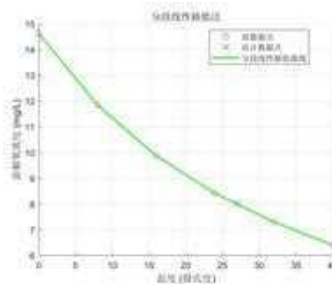


本方法通过选择合适的多项式次数，可以得到较好的结果，经过对于图像的分析可以发现没有龙格现象产生。

3.3 分段线性插值法

运行结果如下：

```
命令窗口
% 输出结果
disp(['在 ', num2str(x), ' 度时的溶解氧浓度为 ', num2str(y), ' mg/L'])
disp(['相对误差为 ', num2str(relative_error), '%'])
在 27 度时的溶解氧浓度为 8.001 mg/L
相对误差为 0.183132982719755%
fx >>
```

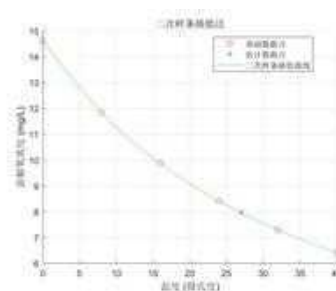


根据插线性值原理可知，在估计 27 度时的近似值时，只用到了 24 和 32 这两个点的数据，其它数据完全没用到，这样的近似结果显然不合理，可能产生了较大误差。

3.4 二次样条插值

运行结果如下：

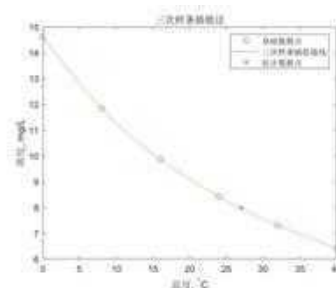
```
命令窗口
relative_error = abs((concentratio
disp(' 27℃时的溶解氧浓度为: ', nu
disp(' 相对误差为: ', num2str(rela
27℃时的溶解氧浓度为: 7.966 mg/L
相对误差为: 0.254083743628201%
fx >>
```



3.5 三次样条插值

运行结果如下：

```
命令窗口
disp(' 在 ', num2str(x_estimate), ' 度
relative_error = abs(f_estimate - 7.98
disp(' 相对误差为: ', num2str(relative_
在 27 度时的溶解氧浓度为 7.968 mg/L
相对误差为 0.226796633692300%
fx >>
```

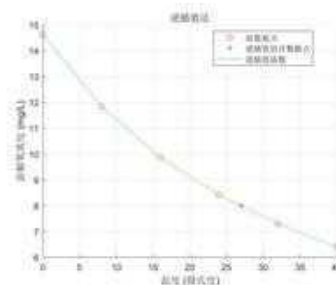


本方法在分段低次插值的基础上，同时保证了函数的导数和二阶导连续，因此整个函数图像是平滑连续的，有着较好的拟合效果。

3.6 逆插值法

运行结果如下：

```
命令窗口
% 输出结果
fprintf(' 27° C 时的溶解氧浓度为 %3f\n', rel
fprintf(' 相对误差为 %15f%%\n', rel
27° C 时的溶解氧浓度为 8.001 mg/L
相对误差为 0.183132982719755%
fx >>
```



程序中的逆插值法采用反向的线性插值法，因此与线性插值类似。

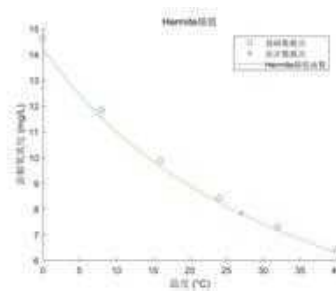
3.7 Hermite 插值

运行结果如下：

```

命令窗口
% 输出结果
disp(['在 ', num2str(temperature_at_27C)
disp([' 相对误差为 ', num2str(relative_er
在 27 ° C 时的溶解氧浓度为 7.822 mg/L
相对误差为 2.050834122439722%
fx >>

```



hermite 插值只需要一阶导数存在，相较于三次样条插值计算上更简单。

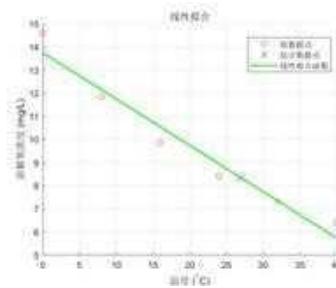
3.8 线性拟合

运行结果如下：

```

命令窗口
legend('原数据点', '估计数据点', '线性拟合
grid on;
hold off;
在 27 度时的溶解氧浓度为 8.342 mg/L
相对误差为 4.462183521687908%
fx >>

```



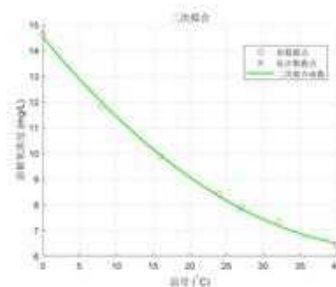
3.9 二次拟合

运行结果如下：

```

命令窗口
title('二次拟合');
legend('原数据点', '估计数据点', '二
grid on;
hold off;
在 27 度时的溶解氧浓度为 7.848 mg/L
相对误差为 1.728092521287290%
fx >>

```



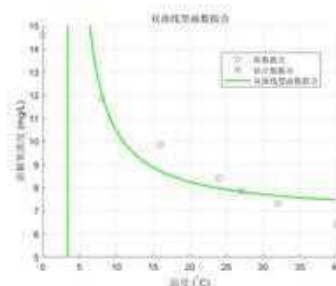
3.10 双曲线型拟合

运行结果如下：

```

命令窗口
lsqcurvefit stopped because the fina
its initial value is less than the v
<stopping criteria details>
在 27 度时的溶解氧浓度为 7.830 mg/L
相对误差为 1.956942262278576%
fx >> |

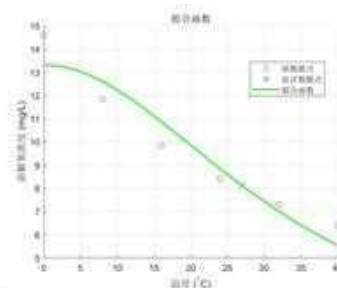
```



3.11 反比例二次函数拟合

运行结果如下：

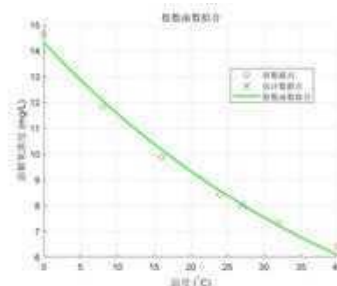
```
命令窗口
lsqcurvefit stopped because the final value of the cost function is less than the tolerance.
<stopping_criteria_details>
在 27 度时的溶解氧浓度为 8.122 mg/L.
相对误差为 1.709086396442746%
fx >> |
```



3.12 指数函数拟合

运行结果如下：

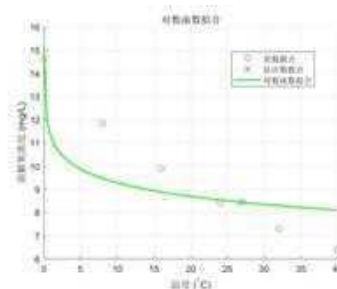
```
命令窗口
lsqcurvefit stopped because the final value of the cost function is less than the tolerance.
<stopping_criteria_details>
在 27 度时的溶解氧浓度为 8.034 mg/L.
相对误差为 0.601598334744551%
fx >> |
```



3.13 对数函数拟合

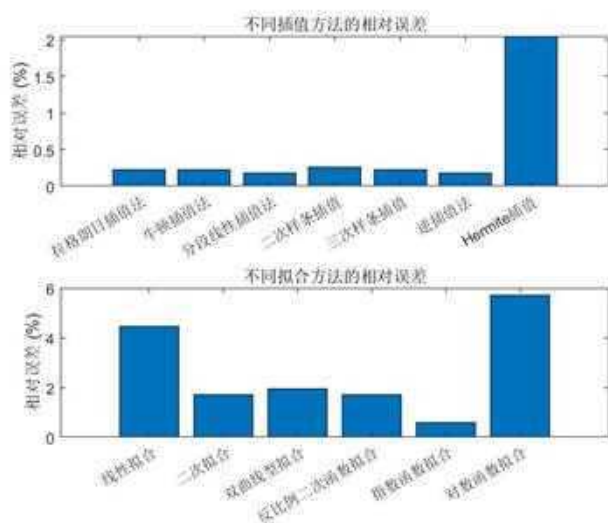
运行结果如下：

```
命令窗口
Optimization completed because the value of the optimality tolerance is less than the value of the optimality tolerance.
<stopping_criteria_details>
在 27 度时的溶解氧浓度为 8.443 mg/L.
相对误差为 5.725476180430597%
fx >> |
```



3.14 小结

对于以上 12 种插值和拟合方法，绘制相对误差的柱状图如下：



可以发现，总体来看，插值法的误差相对来说较拟合法较小，这是因为拟合不要求拟合函数必须过原点。

对于插值法，拉格朗日插值和牛顿插值法所得的插值函数相同，因此结果也相同；逆插值法采取的是反向的线性插值法，因此结果也完全一致。在日后应用时可以根据方便程度任选。

hermite 插值的求解过程只用到了两个点的数值及它们的导数值，而样条插值则用到了三个点的二阶导数值。样条插值要求二阶导数连续，而 *hermite* 插值只需要一阶导数存在。

如使用对数函数拟合，相对误差数量级在 10^{-2} ，误差较大，而使用指数函数拟合，相对误差数量级在 10^{-4} ，误差较小。因此在拟合时选择合适的函数类型非常重要，所以要根据原始点走向，多尝试可能的函数，从中选取最合适的函数类型。

4 实验心得

本次实验我熟悉了插值与拟合的多种方法：从函数角度看，插值法与最小二乘法都是得到估计值的一种根据。函数表求函数的近似表达式的问题，属于函数逼近问题。从几何上看，二者都是根据一系列数据点求曲线的近似曲线问题。

而插值法根据插值条件来选择近似函数；最小二乘法根据“偏差平方和最小”原则选择近似函数。

在拟合的方法时，选取合适的函数类型十分重要，应根据原始数据点选取合适的函数类型进行较多的尝试。

在完成程序过程中，我发现 *matlab* 对于插值和拟合有着较多的内置函数，合理的调用可以节省很多时间，更准确快速的得到结果。

5 源代码

5.1 绘制散点图

```
1 % 温度数据
2 temperature = [0, 8, 16, 24, 32, 40];
```

```

3 % 浓度数据
4 concentration = [14.621, 11.843, 9.870, 8.418, 7.305, 6.413];
5
6 % 绘制散点图
7 scatter(temperature, concentration, 'filled');
8 xlabel('温度, ^\circC');
9 ylabel('浓度, mg/L');
10 title('温度与浓度之间的关系');

```

5.2 拉格朗日插值法

```

1 % 给定的数据点
2 temperature = [0, 8, 16, 24, 32, 40]; % 温度, 摄氏度
3 concentration = [14.621, 11.843, 9.870, 8.418, 7.305, 6.413]; % 浓度,
   mg/L
4
5 % 要插值的温度
6 x = 27; % 摄氏度
7
8 % 计算拉格朗日插值多项式
9 n = length(temperature);
10 P = 0;
11
12 for i = 1:n
13     % 计算基函数的分子
14     numerator = 1;
15     for j = 1:n
16         if j ~= i
17             numerator = numerator * (x - temperature(j));
18         end
19     end
20
21     % 计算基函数的分母
22     denominator = 1;
23     for j = 1:n
24         if j ~= i
25             denominator = denominator * (temperature(i) - temperature(
                j));
26         end
27     end
28

```

```

29     % 计算基函数
30     li = numerator / denominator;
31
32     % 计算插值多项式
33     P = P + concentration(i) * li;
34 end
35
36 % 准确结果
37 accurate_result = 7.986; % mg/L
38
39 % 计算相对误差
40 relative_error = abs((P - accurate_result) / accurate_result) * 100;
41
42 % 绘制图像
43 figure;
44 hold on;
45
46 % 绘制原数据点
47 scatter(temperature, concentration, 'ro', 'DisplayName', '基础数据点')
48     ;
49
50 % 绘制估计数据点
51 plot(x, P, 'bx', 'DisplayName', '估计数据点');
52
53 % 绘制拟合后的函数图像
54 x_interp = linspace(min(temperature), max(temperature), 1000);
55 y_interp = zeros(size(x_interp));
56 for i = 1:length(x_interp)
57     % 计算拉格朗日插值多项式
58     P_interp = 0;
59     for j = 1:n
60         % 计算基函数的分子
61         numerator = 1;
62         for k = 1:n
63             if k ~= j
64                 numerator = numerator * (x_interp(i) - temperature(k))
65             ;
66         end
67     end
68     % 计算基函数的分母

```

```

68     denominator = 1;
69     for k = 1:n
70         if k ~= j
71             denominator = denominator * (temperature(j) -
              temperature(k));
72         end
73     end
74
75     % 计算基函数
76     li = numerator / denominator;
77
78     % 计算插值多项式
79     P_interp = P_interp + concentration(j) * li;
80 end
81 y_interp(i) = P_interp;
82 end
83 plot(x_interp, y_interp, 'g-', 'DisplayName', '拉格朗日插值插值函数');
84
85 hold off;
86 xlabel('温度 (摄氏度)');
87 ylabel('溶解氧浓度 (mg/L)');
88 title('拉格朗日插值法');
89 legend('Location', 'best');
90 grid on;
91
92 % 输出结果
93 disp(['在 ', num2str(x), ' 度时的溶解氧浓度为 ', num2str(P, '%.3f'), ' ',
      'mg/L']);
94 disp(['相对误差为 ', num2str(relative_error, '%.15f'), '%']);

```

5.3 牛顿插值法

```

1 % 给定数据
2 temperature = [0, 8, 16, 24, 32, 40]; % 温度
3 concentration = [14.621, 11.843, 9.870, 8.418, 7.305, 6.413]; % 浓度
4
5 % 牛顿插值计算
6 n = length(temperature);
7 coefficients = zeros(1, n);
8 coefficients(1) = concentration(1);
9

```

```

10 % 计算差商
11 for j = 2:n
12     for i = n:-1:j
13         concentration(i) = (concentration(i) - concentration(i-1)) / (
            temperature(i) - temperature(i-j+1));
14     end
15     coefficients(j) = concentration(j);
16 end
17
18 % 在 27 度时的溶解氧浓度
19 x = 27;
20 P = coefficients(1);
21 for k = 2:n
22     product = 1;
23     for j = 1:k-1
24         product = product * (x - temperature(j));
25     end
26     P = P + coefficients(k) * product;
27 end
28
29 % 准确结果
30 accurate_result = 7.986; % mg/L
31
32 % 计算相对误差
33 relative_error = abs((P - accurate_result) / accurate_result) * 100;
34
35 % 输出结果
36 disp(['在 ', num2str(x), ' 度时的溶解氧浓度为 ', num2str(P, '%.3f'), '
    mg/L']);
37 disp(['相对误差为 ', num2str(relative_error, '%.15f'), '%']);
38
39 % 绘制图像
40 figure;
41 hold on;
42
43 % 绘制原数据点
44 temperature = [0, 8, 16, 24, 32, 40]; % 温度
45 concentration = [14.621, 11.843, 9.870, 8.418, 7.305, 6.413]; % 浓度
46 scatter(temperature, concentration, 'ro', 'DisplayName', '基础数据点')
    ;
47

```

```

48 % 计算插值数据点
49 x_interp = linspace(min(temperature), max(temperature), 1000);
50 y_interp = zeros(size(x_interp));
51 for i = 1:length(x_interp)
52     P_interp = coefficients(1);
53     for k = 2:n
54         product = 1;
55         for j = 1:k-1
56             product = product * (x_interp(i) - temperature(j));
57         end
58         P_interp = P_interp + coefficients(k) * product;
59     end
60     y_interp(i) = P_interp;
61 end
62
63 % 绘制拟合后的函数图像
64 plot(x_interp, y_interp, 'g-', 'DisplayName', '插值函数');
65
66 % 绘制估计数据点
67 plot(x, P, 'bx', 'DisplayName', '估计数据点');
68
69 hold off;
70
71 xlabel('温度 (摄氏度)');
72 ylabel('溶解氧浓度 (mg/L)');
73 title('牛顿插值法');
74 legend('Location', 'best');
75 grid on;

```

5.4 分段线性插值法

```

1 % 给定数据
2 temperature = [0, 8, 16, 24, 32, 40]; % 温度数据
3 concentration = [14.621, 11.843, 9.870, 8.418, 7.305, 6.413]; % 浓度数据
4
5 % 要计算的温度
6 x = 27;
7
8 % 使用分段低次插值计算目标温度下的溶解氧浓度
9 P = interp1(temperature, concentration, x, 'linear');

```

```

10
11 % 准确结果
12 accurate_result = 7.986; % mg/L
13
14 % 计算相对误差
15 relative_error = abs((P - accurate_result) / accurate_result) * 100;
16
17 % 绘制图像
18 figure;
19 hold on;
20
21 % 绘制原数据点
22 scatter(temperature, concentration, 'ro', 'DisplayName', '原数据点');
23
24 % 绘制估计数据点
25 plot(x, P, 'bx', 'DisplayName', '估计数据点', 'MarkerSize', 10);
26
27 % 绘制拟合曲线
28 x_interp = linspace(min(temperature), max(temperature), 1000);
29 y_interp = interp1(temperature, concentration, x_interp, 'linear');
30 plot(x_interp, y_interp, 'g-', 'DisplayName', '分段线性插值曲线', '
    LineWidth', 1.5);
31
32 hold off;
33 xlabel('温度 (摄氏度)');
34 ylabel('溶解氧浓度 (mg/L)');
35 title('分段线性插值法');
36 legend('Location', 'best');
37 grid on;
38
39 % 输出结果
40 disp(['在 ', num2str(x), ' 度时的溶解氧浓度为 ', num2str(P, '%.3f'), '
    mg/L']);
41 disp(['相对误差为 ', num2str(relative_error, '%.15f'), '%']);

```

5.5 二次样条插值

```

1 % 给定的数据
2 temperature = [0, 8, 16, 24, 32, 40]; % 温度
3 concentration = [14.621, 11.843, 9.870, 8.418, 7.305, 6.413]; % 浓度
4

```

```

5 % 求解二次样条插值
6 n = length(temperature);
7 a = concentration; % 节点处的值
8 h = diff(temperature); % 节点之间的差值
9 alpha = zeros(1, n);
10 for i = 2:n-1
11     alpha(i) = 3 * (a(i+1) - a(i)) / h(i) - 3 * (a(i) - a(i-1)) / h(i-1);
12 end
13 l = zeros(1, n);
14 mu = zeros(1, n);
15 z = zeros(1, n);
16 l(1) = 1;
17 mu(1) = 0;
18 z(1) = 0;
19 for i = 2:n-1
20     l(i) = 2 * (temperature(i+1) - temperature(i-1)) - h(i-1) * mu(i-1);
21     mu(i) = h(i) / l(i);
22     z(i) = (alpha(i) - h(i-1) * z(i-1)) / l(i);
23 end
24 l(n) = 1;
25 z(n) = 0;
26 c = zeros(1, n);
27 b = zeros(1, n);
28 d = zeros(1, n);
29 for j = n-1:-1:1
30     c(j) = z(j) - mu(j) * c(j+1);
31     b(j) = (a(j+1) - a(j)) / h(j) - h(j) * (c(j+1) + 2 * c(j)) / 3;
32     d(j) = (c(j+1) - c(j)) / (3 * h(j));
33 end
34
35 % 计算估计数据点
36 x_interp = linspace(temperature(1), temperature(end), 1000);
37 y_interp = zeros(size(x_interp));
38 for i = 1:length(x_interp)
39     for j = 1:n-1
40         if temperature(j) <= x_interp(i) && x_interp(i) <= temperature(j+1)
41             y_interp(i) = a(j) + b(j) * (x_interp(i) - temperature(j))
                     + c(j) * (x_interp(i) - temperature(j))^2 + d(j) * (

```



```

        x_interp(i) - temperature(j))^3;
42         break;
43     end
44 end
45 end
46
47 % 绘制原数据点
48 scatter(temperature, concentration, 'ro', 'DisplayName', '基础数据点')
49 ;
50 hold on;
51 % 计算所需温度下的溶解氧浓度 (27°C)
52 desired_temperature = 27;
53 for i = 1:n-1
54     if temperature(i) <= desired_temperature && desired_temperature <=
        temperature(i+1)
55         concentration_27 = a(i) + b(i) * (desired_temperature -
            temperature(i)) + c(i) * (desired_temperature -
            temperature(i))^2 + d(i) * (desired_temperature -
            temperature(i))^3;
56         break;
57     end
58 end
59
60 % 输出结果
61 exact_concentration = 7.986; % 准确的溶解氧浓度
62 relative_error = abs((concentration_27 - exact_concentration) /
    exact_concentration); % 计算相对误差
63 disp(['27°C时的溶解氧浓度为: ', num2str(concentration_27, '%.3f'), ' mg
    /L']);
64 disp(['相对误差为: ', num2str(relative_error * 100, '%.15f'), '%']);
65
66 % 绘制估计数据点
67 plot(27, concentration_27, 'bx', 'DisplayName', '估计数据点');
68
69 % 绘制二次样条插值曲线
70 plot(x_interp, y_interp, 'g-', 'DisplayName', '二次样条插值曲线');
71
72 % 添加标签和标题
73 xlabel('温度 (摄氏度)');
74 ylabel('溶解氧浓度 (mg/L)');

```

```

75 title('二次样条插值法');
76
77 % 添加图例和网格
78 legend('Location', 'best');
79 grid on;

```

5.6 列主元消去法函数

```

1 function x=Column_principal(C)
2 n=size(C,1);
3 %消去过程
4 for k=1:n
5     a=C(k:n,k);
6     P=max(abs(a));
7     u=find(abs(a)==P);
8     if(u~=1)
9         C([k,(u+k-1)],:)=C([(u+k-1),k],:);
10    end
11    for i=k+1:n
12        factor=C(i,k)/C(k,k);
13        for j=1:(n+1)
14            C(i,j)=C(i,j)-C(k,j)*factor;
15        end
16    end
17 end
18
19 %回代过程
20 x(n)=C(n,(n+1))/C(n,n);
21 for i=(n-1):-1:1
22     sum=C(i,(n+1));
23     for j=i+1:n
24         sum=sum-C(i,j)*x(j);
25     end
26     x(i)=sum/C(i,i);
27 end

```

5.7 三次样条插值

```

1 x = [0 8 16 24 32 40];
2 y = [14.621 11.843 9.870 8.418 7.305 6.413];
3 n = size(x, 2); % 已知点的个数

```

```

4
5 M = zeros(1, n);
6 h = zeros(1, n - 1);
7 f = zeros(n, 3);
8
9 % 为 hi 赋值
10 for i = 2:n
11     h(i - 1) = x(i) - x(i - 1);
12 end
13
14 % 给 ui 赋值
15 for i = 1:(n - 2)
16     u(i) = h(i) / (h(i) + h(i + 1));
17 end
18
19 % 给 ri 赋值
20 for i = 1:(n - 2)
21     r(i) = 1 - u(i);
22 end
23
24 % 对差商表第一列赋值
25 for i = 1:n
26     f(i, 1) = y(i);
27 end
28
29 % 求差商表第2、3列
30 for i = 2:3
31     for k = i:n
32         if k >= i
33             f(k, i) = (f(k - 1, i - 1) - f(k, i - 1)) / (x(k + 1 - i)
34                 - x(k));
35         end
36     end
37 end
38 % 给 gi 赋值
39 for i = 1:n - 2
40     g(i) = 6 * f(i + 2, 3);
41 end
42
43 % 使用边界条件2

```

```

44 M(1) = 0;
45 M(n) = 0;
46 A = zeros(n - 2, n - 2);
47 % 为系数矩阵赋值
48 for i = 1:n - 2
49     A(i, i) = 2;
50 end
51 for i = 3:n - 1
52     A(i - 1, i - 2) = u(i - 1);
53     A(i - 2, i - 1) = r(i - 2);
54 end
55 g1 = transpose(g);
56 % 增广矩阵
57 B = [A, g1];
58 % 列主元消去法
59 Y = Column_principal(B);
60 % M
61 for i = 2:n - 1
62     M(i) = Y(i - 1);
63 end
64
65 x1 = 0:0.001:40;
66 m = length(x1);
67 f1 = zeros(1, m);
68 for i = 1:m
69     for k = 1:n - 1
70         if (x1(i) >= x(k)) && (x1(i) <= x(k + 1))
71             h = x(k + 1) - x(k);
72             f1(i) = M(k) * (x(k + 1) - x1(i))^3 / (6 * h)...
73                 + M(k + 1) * (x1(i) - x(k))^3 / (6 * h)...
74                 + (y(k) - M(k) * h^2 / 6) * ((x(k + 1) - x1(i)) / h)...
75                 + (y(k + 1) - M(k + 1) * h^2 / 6) * ((x1(i) - x(k)) /
76                     h);
77         end
78     end
79 end
80 % 估计数据点
81 x_estimate = 27;
82 f_estimate = interp1(x, y, x_estimate, 'spline');
83

```

```

84 % 绘图
85 plot(x, y, 'ro'); % 原数据点
86 hold on;
87 plot(x1, f1, 'g-'); % 拟合函数
88 plot(x_estimate, f_estimate, 'bx', 'DisplayName', '估计数据点'); % 估计数据点
89 legend('基础数据点', '三次样条插值曲线', '估计数据点');
90 xlabel('温度, ^\circC');
91 ylabel('浓度, mg/L');
92 title('三次样条插值法');
93
94 % 输出结果
95 disp(['在 ', num2str(x_estimate), ' 度时的溶解氧浓度为 ', num2str(f_estimate, '%.3f'), ' mg/L']);
96 relative_error = abs(f_estimate - 7.986) / 7.986 * 100; % 计算相对误差
97 disp(['相对误差为 ', num2str(relative_error, '%.15f'), '%']);

```

5.8 逆插值法

```

1 % 给定的数据
2 T_known = [0, 8, 16, 24, 32, 40]; % 已知的温度数据
3 C_known = [14.621, 11.843, 9.870, 8.418, 7.305, 6.413]; % 对应的溶解氧浓度数据
4
5 % 要求解的温度
6 T_req = 27;
7
8 % 逆插值
9 index = find(T_known <= T_req, 1, 'last'); % 找到小于等于要求温度的最后一个已知温度点的索引
10 T_lower = T_known(index); % 小于等于要求温度的最后一个已知温度点
11 C_lower = C_known(index); % 对应的溶解氧浓度
12 T_upper = T_known(index+1); % 大于要求温度的第一个已知温度点
13 C_upper = C_known(index+1); % 对应的溶解氧浓度
14
15 % 线性插值
16 C_req = C_lower + (T_req - T_lower) * (C_upper - C_lower) / (T_upper - T_lower);
17
18 % 真实准确结果
19 C_true = 7.986;

```

```

20
21 % 计算相对误差
22 relative_error = abs((C_req - C_true) / C_true);
23
24 % 绘制图像
25 figure;
26 hold on;
27
28 % 绘制原数据点
29 scatter(T_known, C_known, 'ro', 'DisplayName', '原数据点');
30
31 % 绘制逆插值的估计数据点
32 scatter(T_req, C_req, 'bx', 'DisplayName', '逆插值估计数据点');
33
34 % 绘制拟合后的函数图像
35 x_interp = linspace(min(T_known), max(T_known), 1000);
36 y_interp = interp1(T_known, C_known, x_interp, 'linear');
37 plot(x_interp, y_interp, 'g-', 'DisplayName', '逆插值函数');
38
39 hold off;
40 xlabel('温度 (摄氏度)');
41 ylabel('溶解氧浓度 (mg/L)');
42 title('逆插值法');
43 legend('Location', 'best');
44 grid on;
45
46 % 输出结果
47 fprintf('27°C 时的溶解氧浓度为 %.3f mg/L\n', C_req);
48 fprintf('相对误差为 %.15f%%\n', relative_error*100);

```

5.9 Hermite 插值函数

```

1 % 用Hermite插值函数计算溶解氧浓度
2 function result = hermite_interpolation(x, temperature, concentration,
    d_concentration)
3     result = 0;
4     n = length(temperature);
5     for i = 1:n
6         % 计算基础拉格朗日基函数的累积乘积
7         term = concentration(i);
8         for j = 1:n

```

```

9         if j ~= i
10             term = term * (x - temperature(j)) / (temperature(i) -
               temperature(j));
11         end
12     end
13     % 计算一阶导数乘积项
14     d_term = d_concentration(i);
15     for j = 1:n
16         if j ~= i
17             d_term = d_term * (x - temperature(j)) / (temperature(
               i) - temperature(j));
18         end
19     end
20     % 基础拉格朗日基函数乘以一阶导数乘积项
21     result = result + term + d_term;
22 end
23 end

```

5.10 Hermite 插值

```

1 % 给定数据点
2 temperature = [0, 8, 16, 24, 32, 40]; % 温度, 单位: °C
3 concentration = [14.621, 11.843, 9.870, 8.418, 7.305, 6.413]; % 浓度,
   单位: mg/L
4
5 % 计算一阶导数
6 d_concentration = gradient(concentration) ./ gradient(temperature);
7
8 % 计算拟合曲线
9 x_values = linspace(min(temperature), max(temperature), 1000);
10 y_values = arrayfun(@(x) hermite_interpolation(x, temperature,
   concentration, d_concentration), x_values);
11
12 % 计算27°C时的溶解氧浓度
13 temperature_at_27C = 27; % 27°C
14 concentration_at_27C = hermite_interpolation(temperature_at_27C,
   temperature, concentration, d_concentration);
15
16 % 准确结果
17 true_concentration = 7.986; % 准确溶解氧浓度, 单位: mg/L
18

```

```

19 % 计算相对误差
20 relative_error = abs(concentration_at_27C - true_concentration) /
    true_concentration;
21
22 % 绘制图像
23 figure;
24 % 绘制原数据点
25 scatter(temperature, concentration, 'ro', 'DisplayName', '基础数据点')
    ;
26 hold on;
27 % 绘制估计数据点
28 plot(temperature_at_27C, concentration_at_27C, 'bx', 'DisplayName', '
    估计数据点');
29 % 绘制Hermite插值函数
30 plot(x_values, y_values, 'g-', 'DisplayName', 'Hermite插值函数');
31 xlabel('温度 (°C)');
32 ylabel('溶解氧浓度 (mg/L)');
33 title('Hermite插值');
34 legend('基础数据点', '估计数据点', 'Hermite插值函数');
35
36 % 输出结果
37 disp(['在 ', num2str(temperature_at_27C), ' °C 时的溶解氧浓度为 ',
    num2str(concentration_at_27C, '%.3f'), ' mg/L']);
38 disp(['相对误差为 ', num2str(relative_error * 100, '%.15f'), '%']);

```

5.11 线性拟合

```

1 % 已知数据点
2 temperature = [0, 8, 16, 24, 32, 40]; % 温度数据
3 concentration = [14.621, 11.843, 9.870, 8.418, 7.305, 6.413]; % 浓度数
    据
4
5 % 进行直线拟合
6 p = polyfit(temperature, concentration, 1);
7
8 % 使用直线模型预测 27°C 时的溶解氧浓度
9 target_temperature = 27;
10 concentration_at_target = polyval(p, target_temperature);
11
12 % 真实的溶解氧浓度
13 true_concentration = 7.986;

```



```

14
15 % 计算相对误差
16 relative_error = abs(concentration_at_target - true_concentration) /
    true_concentration;
17
18 % 输出结果
19 disp(['在 ', num2str(target_temperature), ' 度时的溶解氧浓度为 ',
    num2str(concentration_at_target, '%.3f'), ' mg/L']);
20 disp(['相对误差为 ', num2str(relative_error * 100, '%.15f'), '%']);
21
22 % 绘制图像
23 figure;
24 scatter(temperature, concentration, 'ro', 'DisplayName', '原数据点');
    % 原数据点
25 hold on;
26 plot(target_temperature, concentration_at_target, 'bx', 'MarkerSize',
    10, 'DisplayName', '估计数据点'); % 估计数据点
27 x_values = linspace(0, 40, 100);
28 y_values = polyval(p, x_values);
29 plot(x_values, y_values, 'g-', 'LineWidth', 1.5, 'DisplayName', '拟合
    后的函数'); % 拟合后的函数图像
30 xlabel('温度 ( $\text{^\circ C}$ )');
31 ylabel('溶解氧浓度 (mg/L)');
32 title('线性拟合');
33 legend('原数据点', '估计数据点', '线性拟合函数', 'Location', '
    southwest');
34 grid on;
35 hold off;

```

5.12 二次拟合

```

1 % 已知数据点
2 temperature = [0, 8, 16, 24, 32, 40]; % 温度数据
3 concentration = [14.621, 11.843, 9.870, 8.418, 7.305, 6.413]; % 浓度数
    据
4
5 % 进行二次拟合
6 p = polyfit(temperature, concentration, 2);
7
8 % 使用二次拟合模型预测 27°C 时的溶解氧浓度
9 target_temperature = 27;

```

```

10 concentration_at_target = polyval(p, target_temperature);
11
12 % 真实的溶解氧浓度
13 true_concentration = 7.986;
14
15 % 计算相对误差
16 relative_error = abs(concentration_at_target - true_concentration) /
    true_concentration;
17
18 % 输出结果
19 disp(['在 ', num2str(target_temperature), ' 度时的溶解氧浓度为 ',
    num2str(concentration_at_target, '%.3f'), ' mg/L']);
20 disp(['相对误差为 ', num2str(relative_error * 100, '%.15f'), '%']);
21
22 % 绘制图像
23 figure;
24 scatter(temperature, concentration, 'ro', 'DisplayName', '原数据点');
    % 原数据点
25 hold on;
26 plot(target_temperature, concentration_at_target, 'bx', 'MarkerSize',
    10, 'DisplayName', '估计数据点'); % 估计数据点
27 x_values = linspace(0, 40, 100);
28 y_values = polyval(p, x_values);
29 plot(x_values, y_values, 'g-', 'LineWidth', 1.5, 'DisplayName', '拟合
    后的函数'); % 拟合后的函数图像
30 xlabel('温度 ( $^{\circ}\text{C}$ )');
31 ylabel('溶解氧浓度 (mg/L)');
32 title('二次拟合');
33 legend('原数据点', '估计数据点', '二次拟合函数', 'Location', '
    southwest');
34 grid on;
35 hold off;

```

5.13 双曲线型拟合

```

1 % 已知数据点
2 temperature = [0, 8, 16, 24, 32, 40]; % 温度数据
3 concentration = [14.621, 11.843, 9.870, 8.418, 7.305, 6.413]; % 浓度数
    据
4
5 % 定义双曲线型函数

```

```

6 fun = @(coeffs , x) x ./ (coeffs(1) * x + coeffs(2));
7
8 % 初始参数猜测
9 initial_guess = [1, 1];
10
11 % 进行双曲线型函数拟合
12 coeffs = lsqcurvefit(fun, initial_guess, temperature, concentration);
13
14 % 使用双曲线型函数模型预测 27°C 时的溶解氧浓度
15 target_temperature = 27;
16 concentration_at_target = target_temperature / (coeffs(1) *
    target_temperature + coeffs(2));
17
18 % 真实的溶解氧浓度
19 true_concentration = 7.986;
20
21 % 计算相对误差
22 relative_error = abs(concentration_at_target - true_concentration) /
    true_concentration;
23
24 % 输出结果
25 disp(['在 ', num2str(target_temperature), ' 度时的溶解氧浓度为 ',
    num2str(concentration_at_target, '%.3f'), ' mg/L']);
26 disp(['相对误差为 ', num2str(relative_error * 100, '%.15f'), '%']);
27
28 % 绘制图像
29 figure;
30 scatter(temperature, concentration, 'ro', 'DisplayName', '原数据点');
    % 原数据点
31 hold on;
32 plot(target_temperature, concentration_at_target, 'bx', 'MarkerSize',
    10, 'DisplayName', '估计数据点'); % 估计数据点
33 x_values = linspace(0, 40, 100);
34 y_values = fun(coeffs, x_values);
35 plot(x_values, y_values, 'g-', 'LineWidth', 1.5, 'DisplayName', '拟合
    后的函数'); % 拟合后的函数图像
36 ylim([5,15]);
37 xlabel('温度 (^{\circ}C)');
38 ylabel('溶解氧浓度 (mg/L)');
39 title('双曲线型函数拟合');
40 legend('原数据点', '估计数据点', '双曲线型函数拟合', 'Location', '

```

```

    southwest');
41 grid on;
42 hold off;

```

5.14 反比例二次函数拟合

```

1 % 已知数据点
2 temperature = [0, 8, 16, 24, 32, 40]; % 温度数据
3 concentration = [14.621, 11.843, 9.870, 8.418, 7.305, 6.413]; % 浓度数据
4
5 % 定义函数
6 fun = @(coeffs, x) 1 ./ (coeffs(1) + coeffs(2) * x.^2);
7
8 % 初始参数猜测
9 initial_guess = [1, 1];
10
11 % 进行拟合
12 coeffs = lsqcurvefit(fun, initial_guess, temperature, concentration);
13
14 % 使用模型预测 27°C 时的溶解氧浓度
15 target_temperature = 27;
16 concentration_at_target = 1 / (coeffs(1) + coeffs(2) *
    target_temperature^2);
17
18 % 真实的溶解氧浓度
19 true_concentration = 7.986;
20
21 % 计算相对误差
22 relative_error = abs(concentration_at_target - true_concentration) /
    true_concentration;
23
24 % 输出结果
25 disp(['在 ', num2str(target_temperature), ' 度时的溶解氧浓度为 ',
    num2str(concentration_at_target, '%.3f'), ' mg/L']);
26 disp(['相对误差为 ', num2str(relative_error * 100, '%.15f'), '%']);
27
28 % 绘制图像
29 figure;
30 scatter(temperature, concentration, 'ro', 'DisplayName', '原数据点');
    % 原数据点

```

```

31 hold on;
32 plot(target_temperature, concentration_at_target, 'bx', 'MarkerSize',
      10, 'DisplayName', '估计数据点'); % 估计数据点
33 x_values = linspace(0, 40, 100);
34 y_values = fun(coeffs, x_values);
35 plot(x_values, y_values, 'g-', 'LineWidth', 1.5, 'DisplayName', '拟合
      后的函数'); % 拟合后的函数图像
36 xlabel('温度 ( $^{\circ}\text{C}$ )');
37 ylabel('溶解氧浓度 (mg/L)');
38 title('拟合函数');
39 legend('原数据点', '估计数据点', '拟合函数', 'Location', 'southwest');
40 grid on;
41 hold off;

```

5.15 指数函数拟合

```

1 % 已知数据点
2 temperature = [0, 8, 16, 24, 32, 40]; % 温度数据
3 concentration = [14.621, 11.843, 9.870, 8.418, 7.305, 6.413]; % 浓度数
      据
4
5 % 定义指数函数
6 fun = @(coeffs, x) coeffs(1) * exp(-coeffs(2) * x);
7
8 % 初始参数猜测
9 initial_guess = [1, 1];
10
11 % 进行指数函数拟合
12 coeffs = lsqcurvefit(fun, initial_guess, temperature, concentration);
13
14 % 使用指数函数模型预测 27°C 时的溶解氧浓度
15 target_temperature = 27;
16 concentration_at_target = coeffs(1) * exp(-coeffs(2) *
      target_temperature);
17
18 % 真实的溶解氧浓度
19 true_concentration = 7.986;
20
21 % 计算相对误差
22 relative_error = abs(concentration_at_target - true_concentration) /
      true_concentration;

```

```

23
24 % 输出结果
25 disp(['在 ', num2str(target_temperature), ' 度时的溶解氧浓度为 ',
        num2str(concentration_at_target, '%.3f'), ' mg/L']);
26 disp(['相对误差为 ', num2str(relative_error * 100, '%.15f'), '%']);
27
28 % 绘制图像
29 figure;
30 scatter(temperature, concentration, 'ro', 'DisplayName', '原数据点');
    % 原数据点
31 hold on;
32 plot(target_temperature, concentration_at_target, 'bx', 'MarkerSize',
        10, 'DisplayName', '估计数据点'); % 估计数据点
33 x_values = linspace(0, 40, 100);
34 y_values = fun(coeffs, x_values);
35 plot(x_values, y_values, 'g-', 'LineWidth', 1.5, 'DisplayName', '拟合
    后的函数'); % 拟合后的函数图像
36 xlabel('温度 ( $^{\circ}\text{C}$ )');
37 ylabel('溶解氧浓度 (mg/L)');
38 title('指数函数拟合');
39 legend('原数据点', '估计数据点', '指数函数拟合', 'Location', '
    southwest');
40 grid on;
41 hold off;

```

5.16 对数函数拟合

```

1 % 已知数据点
2 temperature = [0.01, 8, 16, 24, 32, 40]; % 温度数据, 将初始点修改为0
    .01
3 concentration = [14.621, 11.843, 9.870, 8.418, 7.305, 6.413]; % 浓度数
    据
4
5 % 定义对数函数
6 fun = @(coeffs, x) coeffs(1) * log(x) + coeffs(2);
7
8 % 初始参数猜测
9 initial_guess = [1, 1];
10
11 % 进行对数函数拟合
12 coeffs = lsqcurvefit(fun, initial_guess, temperature, concentration);

```

```

13
14 % 使用对数函数模型预测 27°C 时的溶解氧浓度
15 target_temperature = 27;
16 concentration_at_target = coeffs(1) * log(target_temperature) + coeffs
    (2);
17
18 % 真实的溶解氧浓度
19 true_concentration = 7.986;
20
21 % 计算相对误差
22 relative_error = abs(concentration_at_target - true_concentration) /
    true_concentration;
23
24 % 输出结果
25 disp(['在 ', num2str(target_temperature), ' 度时的溶解氧浓度为 ',
    num2str(concentration_at_target, '%.3f'), ' mg/L']);
26 disp(['相对误差为 ', num2str(relative_error * 100, '%.15f'), '%']);
27
28 % 绘制图像
29 figure;
30 scatter(temperature, concentration, 'ro', 'DisplayName', '原数据点');
    % 原数据点
31 hold on;
32 plot(target_temperature, concentration_at_target, 'bx', 'MarkerSize',
    10, 'DisplayName', '估计数据点'); % 估计数据点
33 x_values = linspace(0.01, 40, 100); % 注意对数函数的定义域要求 x > 0
34 y_values = fun(coeffs, x_values);
35 plot(x_values, y_values, 'g-', 'LineWidth', 1.5, 'DisplayName', '拟合
    后的函数'); % 拟合后的函数图像
36 xlabel('温度 (^{\circ}C)');
37 ylabel('溶解氧浓度 (mg/L)');
38 title('对数函数拟合');
39 legend('原数据点', '估计数据点', '对数函数拟合', 'Location', '
    southwest');
40 grid on;
41 hold off;

```

5.17 柱状图绘制

```

1 % 插值方法
2 interpolation_methods = {'拉格朗日插值法', '牛顿插值法', '分段线性插值

```

```

    法', '二次样条插值', '三次样条插值', '逆插值法', 'Hermite插值'};
3 interpolation_errors = [0.222402200524500, 0.222402200524511,
    0.183132982719755, 0.254083743628201, 0.226796633692300,
    0.183132982719755, 2.050834122439722];
4
5 % 拟合方法
6 fitting_methods = {'线性拟合', '二次拟合', '双曲线型拟合', '反比例二次
    函数拟合', '指数函数拟合', '对数函数拟合'};
7 fitting_errors = [4.462183821687908, 1.728092521287290,
    1.956942262278576, 1.709086396442746, 0.601598334744551,
    5.725476180430597];
8
9 % 创建柱状图
10 figure;
11 subplot(2, 1, 1);
12 bar(interpolation_errors);
13 xticks(1:length(interpolation_methods));
14 xticklabels(interpolation_methods);
15 ylabel('相对误差 (%)');
16 title('不同插值方法的相对误差');
17
18 subplot(2, 1, 2);
19 bar(fitting_errors);
20 xticks(1:length(fitting_methods));
21 xticklabels(fitting_methods);
22 ylabel('相对误差 (%)');
23 title('不同拟合方法的相对误差');

```