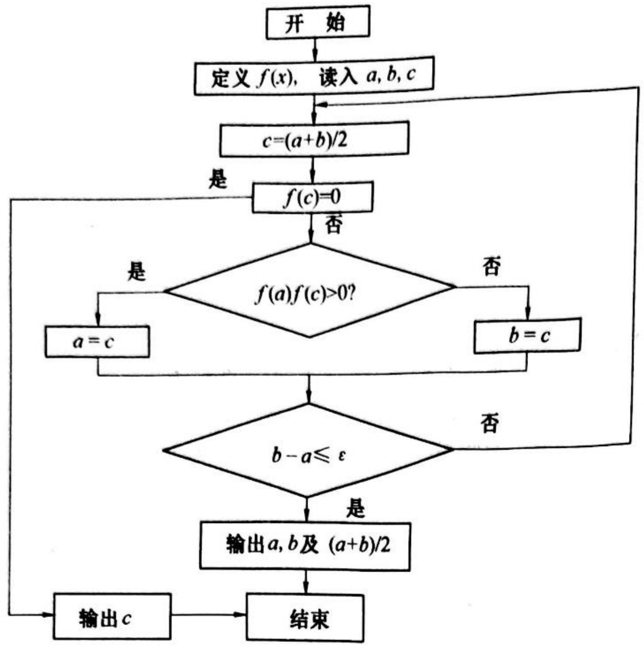
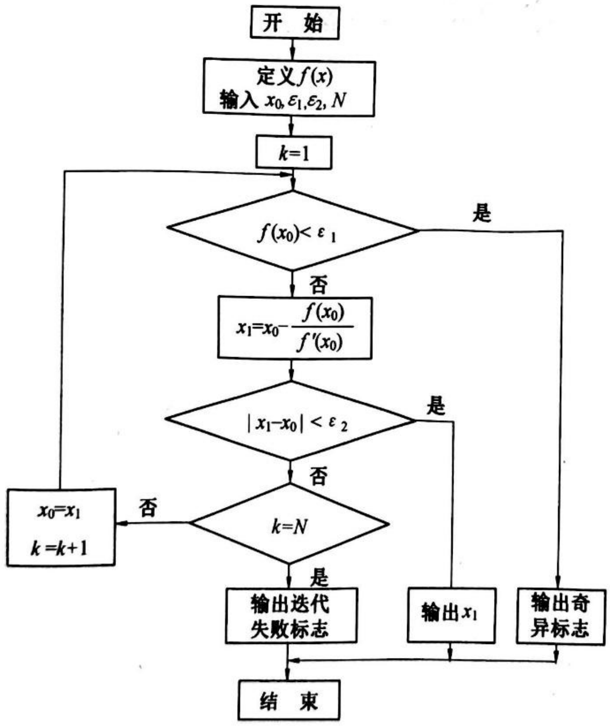
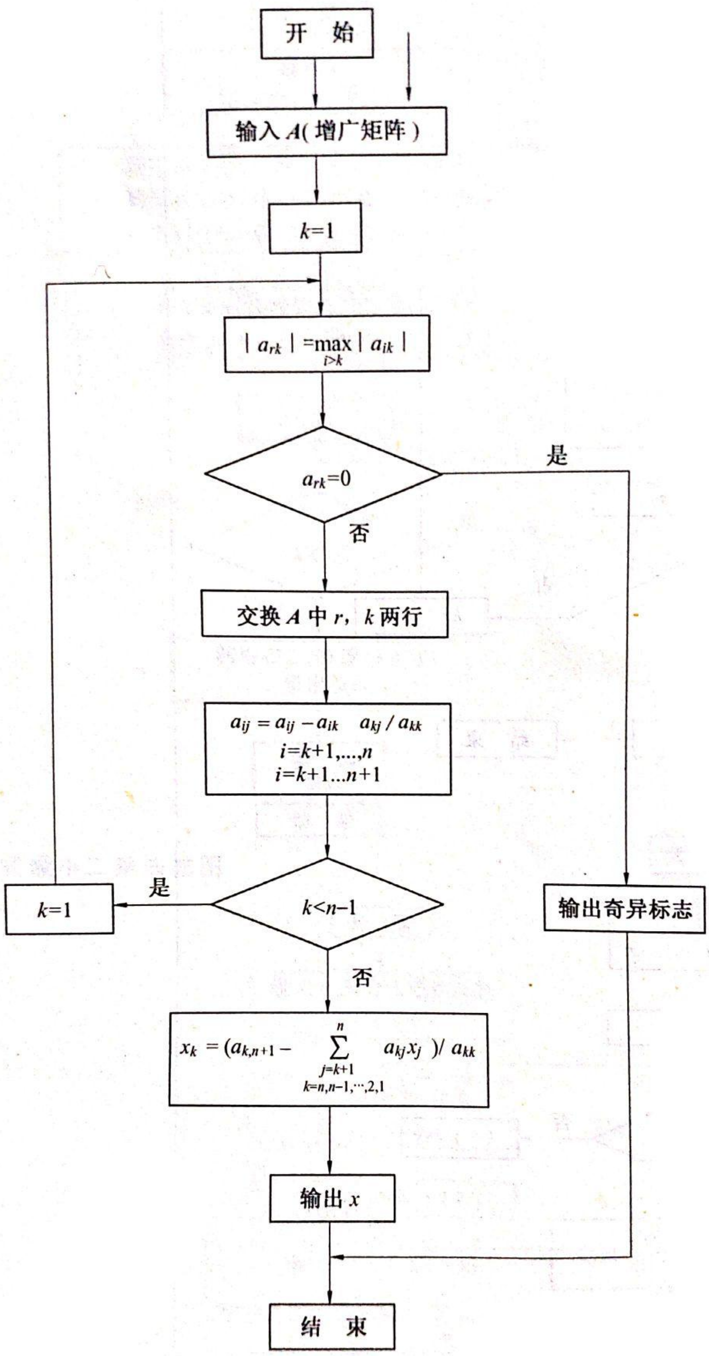
二分法框图


开 始
定义 $f(x)$， 读入 a, b, c
$c=(a+b)/2$
是
$f(c)=0$
否
是
$f(a)f(c)>0?$
否
$a=c$
$b=c$
否
$b-a\leqslant\varepsilon$
是
输出 a, b 及 $(a+b)/2$
输出 c
结束

牛顿法框图


开 始
定义 $f(x)$
输入 $x_0, \varepsilon_1, \varepsilon_2, N$
$k=1$
$f(x_0)<\varepsilon_1$
是
否
$x_1=x_0-\dfrac{f(x_0)}{f'(x_0)}$
$|x_1-x_0|<\varepsilon_2$
是
否
$x_0=x_1$
$k=k+1$
否
$k=N$
是
输出迭代
失败标志
输出 x_1
输出奇
异标志
结 束


开 始
输入 A(增广矩阵)
k=1
$|a_{rk}|=\max_{i>k}|a_{ik}|$
a_{rk}=0
是
否
交换 A 中 r，k 两行
$a_{ij}=a_{ij}-a_{ik}\quad a_{kj}/a_{kk}$
$i=k+1,\ldots,n$
$i=k+1\ldots n+1$
输出奇异标志
k=1
是
$k<n-1$
否
$x_k=(a_{k,n+1}-\sum_{j=k+1}^{n}\quad a_{kj}x_j\ )/\ a_{kk}$
$k=n,n-1,\cdots,2,1$
输出 x
结 束

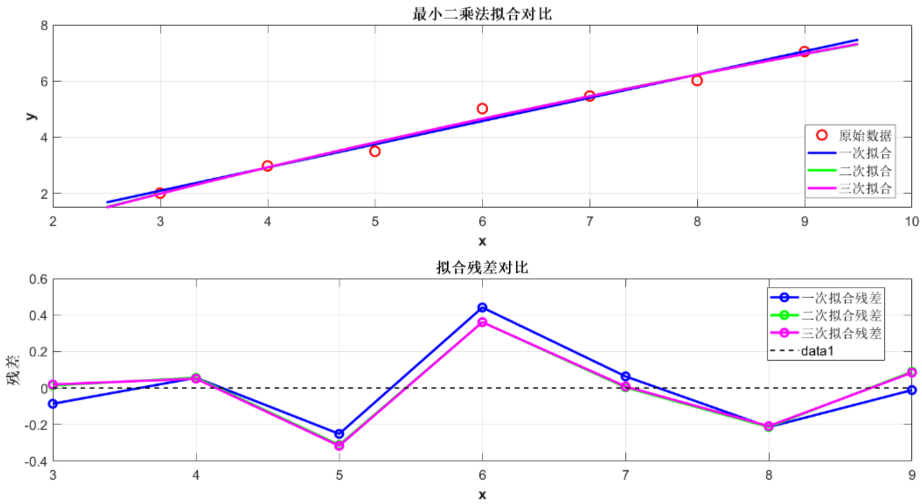


表 1: 不同阶次多项式拟合结果对比

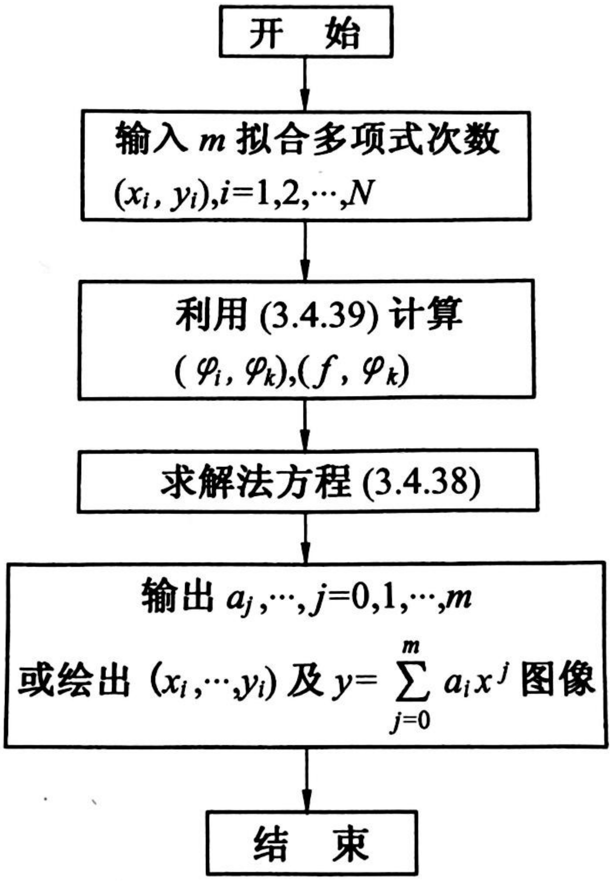
拟合类型	拟合方程	SSE	R^2
一次多项式	$y = 0.8275x - 0.3864$	0.318311	0.983669
二次多项式	$y = -0.0201x^2 + 1.0689x - 1.0302$	0.284310	0.985414
三次多项式	$y = 0.0008x^3 - 0.0351x^2 + 1.1531x - 1.1752$	0.284160	0.985421

结论与讨论

从 SSE 和 R^2 指标来看，二次和三次多项式的拟合效果略优于一次多项式，但二次和三次之间差异非常小。考虑到三次项系数很小（0.0008），实际上二次模型已经足够。

在残差图中，一次拟合的残差呈现明显的趋势（先正后负再正），而二次和三次拟合的残差趋势不明显，且残差绝对值较小，说明二次和三次拟合更好地捕捉了数据的变化趋势。在实际应用中，应选择尽可能简单的模型（如二次）以避免过拟合，同时保证拟合精度。

通过本实验，我们掌握了最小二乘法的原理和实现，并学会了如何通过 SSE、R^2 和残差分析等方法来评估拟合效果，为后续的数据分析工作奠定了基础。


开 始
输入 m 拟合多项式次数
$(x_i, y_i), i=1,2,\cdots,N$
利用 (3.4.39) 计算
$(\varphi_i, \varphi_k),(f, \varphi_k)$
求解法方程 (3.4.38)
输出 $a_j,\cdots, j=0,1,\cdots,m$
或绘出 $(x_i,\cdots y_i)$ 及 $y=\sum_{j=0}^{m} a_i x^j$ 图像
结 束

```matlab
function T = romberg(f, a, b, n)
    % 龙贝格积分法（省略打印表格文本）
    % 输入:
    %   f - 被积函数句柄
    %   a - 积分下限
    %   b - 积分上限
    %   n - 迭代次数
    % 输出:
    %   T - 龙贝格 T 数表

    % 初始化 T 表
    T = zeros(n+1, n+1);

    % 计算 T(0,0) - 梯形公式
    h = b - a;
    T(1,1) = h/2 * (f(a) + f(b));
    % 龙贝格迭代
    for k = 1:n
        % 计算复合梯形公式 T(k,0)
        sum_val = 0;
        m = 2^(k-1);
        for i = 1:m
            x = a + (2*i-1) * h/2;
            sum_val = sum_val + f(x);
        end
        T(k+1,1) = 0.5 * T(k,1) + h/2 * sum_val;

        % Richardson 外推
        for j = 1:k
            T(k+1,j+1) = (4^j * T(k+1,j) - T(k,j)) / (4^j - 1);
        end
        h = h / 2;
    end
end
```