实验　极大似然参数辨识方法

实验报告

学　　院：自动化学院

课　　程：过程建模与系统辨识

学　　号：20211002601

班　　级：231215

姓　　名：张所鑫

日期：2023 年 11 月 12 日

1．实验题目： 极大似然参数辨识方法估计参数值

2．实验目的

（1）掌握极大似然参数辨识方法，并能够加以应用。
（2）了解 Newton-Raphson 方法，理解原理。
（3）能够使用极大似然参数辨识方法和 Newton-Raphson 方法进行参数辨识
（4）能够应用 matlab 编写这两种辨识方法的程序。

3．实验主要原理

（1）极大似然估计法原理

极大似然估计的原理表述为：设 y 为一随机变量，在未知参数θ条件下，y 的概率分布密度函数 p(y|θ)的分布类型已知。为了得到θ的估计值，对随机变量 y 进行 N 次观测，得到一随机观测序列{y(k)}，其中 k = 1,2....N。如果把这 N 个观测值记作 $Y_N = [y(1), y(2), \ldots, y(N)]^T$则$Y_N$的联合密度(或概率分布)为 $p(Y_N|\theta)$，那么参数θ的极大似然估计就是使观测值$Y_N = [y(1), y(2), \ldots, y(N)]^T$出现最大的参数估计值$\hat{\theta}_{ML}$，$\hat{\theta}_{ML}$称为θ的极大似然估计。即

$$p(Y_N|\theta)|_{\hat{\theta}_{ML}} = \max$$

因此，极大似然参数估计的意义在于：对一组确定的随机观测值Y_N，设法找到极大似然估计值$\hat{\theta}_{ML}$，使随机变量 y 在$\hat{\theta}_{ML}$条件下的概率密度函数最大可能地逼近随机变量 y 在θ_0(θ的真值)条件下的概率密度函数，即

$$p(y|\theta)|_{\hat{\theta}_{ML}} \xrightarrow{\max} p(y|\theta_0)$$

现探讨对于 SISO 系统的递推极大估计对于一个线性定常系统，采用如下差分方程表示输出输入的关系

$$y(k) = -\sum_{i=1}^{n} a_i y(k-i) + \sum_{i=0}^{n} b_i u(k-i) + e(k)$$

其中 $e(k) = v(k) + d_1 v(k-1) + \cdots + d_n v(k-n)$
{v(k)}为零均值高斯白噪声，即

$$E\{v(k)\} = 0$$

$$E\{v(k)\}\{v(j)\} = \begin{cases} \sigma^2 & k = j \\ 0 & k \neq j \end{cases}$$

差分方程可以进一步写为

$$y(k) = -\sum_{i=1}^{n} a_i y(k-i) + \sum_{i=0}^{n} b_i u(k-i) + v(k) + \sum_{i=1}^{n} d_i v(k-i)$$

同样，递推极大似然估计解算的目标为：获得未知参数$\theta = [\alpha_0 \cdots \alpha_n \beta_0 \cdots \beta_n d_1 \cdots d_n]^T$的极大似然估计$\hat{\theta}_{ML}$。即在等式

$$v(k) = y(k) + \sum_{i=0}^{n} \alpha_i y(k-i) - \sum_{i=0}^{n} \beta_i u(k-i) - \sum_{i=1}^{n} d_i v(k-i)$$

为约束的条件下，使得指标函数 $J(\theta)=\sum_{k=n+1}^{n+N} V^2(k)$在$\hat{\theta}_{ML}$处取得极小值。
为了实现上述目标，用θ的二次型函数来逼近 J(θ)，从而导出近似的极大似然估计的递推公式。假定存在$\hat{\theta}_N$、P_N和余项β_N，其中P_N为实对称非奇异阵，采用θ的二次型函数来逼近 J(θ)的表达式为

$$J_N(\theta) = \sum_{k=n}^{n+N} V^2(k) = (\theta - \hat{\theta}_N)^T P_N^{-1} (\theta - \hat{\theta}_N) + \beta_N$$

因此

$$J_{N+1}(\theta)=\sum_{k=n}^{n+N+1}v^2(k)=\left(\theta-\hat{\theta}_N\right)^T P_N^{-1}\left(\theta-\hat{\theta}_N\right)+\beta_N+v^2(n+N+1)$$

将$v(n+N+1)$在$\hat{\theta}_N$处进行一阶泰勒展开之后，记

$$v_{N+1}=v(n+N+1)$$

$$\varphi_{N+1}^T=-\left[\frac{\partial v(n+N+1)}{\partial\theta}\right]\Bigg|_{\theta=\hat{\theta}_N}$$

$$\Delta=\theta-\hat{\theta}_N$$

可得到

$$J_{N+1}(\theta)=(\Delta-r_{N+1})^T P_{N+1}^{-1}(\Delta-r_{N+1})+\beta_{N+1}$$

其中

$$r_{N+1}=P_{N+1}\varphi_{N+1}v_{N+1}$$

$$P_{N+1}^{-1}=P_N^{-1}+\varphi_{N+1}\varphi_{N+1}^T$$

$$\beta_{N+1}=v_{N+1}^2+\beta_N-v_{N+1}\varphi_{N+1}^T P_{N+1}\varphi_{N+1}v_{N+1}$$

由于β_{N+1}中不包含θ，因此当$\Delta-r_{N+1}$=0 时，$J_{N+1}(\theta)$取得最小极小值，所以θ新的估计值$\hat{\theta}_{ML}$为

$$\hat{\theta}_{N+1}=\hat{\theta}_N+r_{N+1}$$

因此，只要求取r_{N+1}，就得到了极大似然估计的递推公式。

首先求取P_{N+1}，根据矩阵求逆引理，有

$$P_{N+1}=P_N[I-\varphi_{N+1}(1+\varphi_{N+1}^T P_N\varphi_{N+1})^{-1}\varphi_{N+1}^T P_N]$$

代入r_{N+1}的等式中，得

$$r_{N+1}=P_N\varphi_{N+1}\left(1+\varphi_{N+1}^T P_N\varphi_{N+1}\right)^{-1}v_{N+1}$$

所以新的估计值$\hat{\theta}_{N+1}$为

$$\hat{\theta}_{N+1}=\hat{\theta}_N+P_N\varphi_{N+1}\left(1+\varphi_{N+1}^T P_N\varphi_{N+1}\right)^{-1}v_{N+1}$$

上式中，$\hat{\theta}_N$和P_N为前一次递推结果，v_{N+1}根据系统差分方程可求得

$$v_{N+1}=y(n+N+1)-\psi^T(N)\hat{\theta}_N$$

其中：

$$\varphi(N)=[-y(N+n)\cdots-y(N)\ u(N+n)\cdots u(N)\ v(N+n)\cdots v(N)]^T$$

剩下的，只要能够求得$\varphi(N+1)$与$\varphi(N)$之间的递推关系，即可完成整个递推推导。根据$\varphi(N+1)$的定义，有:

$$\varphi_{N+1}=-\left[\frac{\partial v(n+N+1)}{\partial \theta}\right]\Big|_{\theta=\hat{\theta}_N}=\begin{bmatrix}\frac{\partial v(n+N+1)}{\partial a_1}\\ \vdots\\ \frac{\partial v(n+N+1)}{\partial a_n}\\ \frac{\partial v(n+N+1)}{\partial b_1}\\ \vdots\\ \frac{\partial v(n+N+1)}{\partial b_n}\\ \frac{\partial v(n+N+1)}{\partial d_1}\\ \vdots\\ \frac{\partial v(n+N+1)}{\partial d_n}\end{bmatrix}_{\theta=\hat{\theta}_N}$$

根据$v(k)=y(k)+\sum_{i=1}^{n}a_i y(k-i)-\sum_{i=0}^{n}b_i u(k-i)-\sum_{i=1}^{n}d_i v(k-i)$，经过 Z 变换后合并，最后可求得$\varphi_{N+1}$与$\varphi_N$之间的递推关系为

$$\varphi_{N+1}=\begin{bmatrix}-\hat{d}_1 & \cdots & \cdots & -\hat{d}_n & & & & & & & & \\ 1 & \cdots & \cdots & 0 & & 0 & & & & 0 & & \\ 0 & \ddots & \ddots & 0 & & & & & & & & \\ 0 & 0 & 1 & 0 & & & & & & & & \\ & & & & -\hat{d}_1 & \cdots & \cdots & -\hat{d}_n & 0 & & & & \\ & & & & 1 & \cdots & \cdots & 0 & 0 & & & \\ & 0 & & & 0 & \ddots & \ddots & 0 & 0 & & 0 & \\ & & & & 0 & 0 & 1 & 0 & 0 & & & \\ & & & & & & & & & -\hat{d}_1 & \cdots & \cdots & -\hat{d}_n \\ & 0 & & & & & 0 & & & 1 & \cdots & \cdots & 0 \\ & & & & & & & & & 0 & \ddots & \ddots & 0 \\ & & & & & & & & & 0 & 0 & 1 & 0 \end{bmatrix}\varphi_N+\begin{bmatrix}-y(n+N)\\ 0\\ \vdots\\ 0\\ u(n+N+1)\\ 0\\ \vdots\\ 0\\ v(n+N)\\ 0\\ \vdots\\ 0\end{bmatrix}$$

综合上列推导过程，极大似然参数估计的递推算法构成如下。

令K_{N+1}为增益矩阵，则有：

$$\begin{cases} \hat{\theta}_{N+1} = \hat{\theta}_N + K_{N+1} v_{N+1} \\ K_{N+1} = P_N \varphi_{N+1} \left(1 + \varphi_{N+1}^T P_N \varphi_{N+1}\right)^{-1} \\ P_{N+1} = P_N \left[I - K_{N+1} \varphi_{N+1}^T P_N\right] \\ v_{N+1} = y(n+N+1) - \psi^T(N) \hat{\theta}_N \end{cases}$$

其中

$$\psi(N) = [-y(N+n) \ \cdots \ -y(N) \ \ u(N+n) \cdots \ u(N) \ v(N+n) \cdots \ v(N)]^T$$

初始值一般$\hat{\theta}_0$取为充分小量，P_0取为单位矩阵，v_0设置为零向量，利用获得的系统输入、输出构造$\varphi(0)$，使之为不全为零的向量。

(2) Newton-Raphson 极大似然估计法原理

根据第 L 次迭代得到的参数估计$\hat{\theta}_L$、$\hat{\theta}_{L+1}$

$$J_{L+1}(\hat{\theta}_{L+1}) \leqslant J_L(\hat{\theta}_L)$$

其中$J_L(\theta) = \frac{1}{2}\sum_{K=(L-1)N+1}^{NL} \varepsilon^2(K)$

令$\hat{\theta}_{L+1} = \hat{\theta}_L + \delta\hat{\theta}$,对 $J_{L+1}(\hat{\theta}_{L+1})$进行泰勒展开如下：

$$J_{L+1}(\hat{\theta}(L+1)) = J_{L+1}(\hat{\theta}(L)) + \left[\frac{\partial J_{L+1}(\theta)}{\partial\theta}\bigg|_{\theta=\hat{\theta}(L)}\right]^T \delta\hat{\theta} + \frac{1}{2}\delta\hat{\theta}^T \left[\frac{\partial^2 J_{L+1}(\theta)}{(\partial\theta)^2}\right]\bigg|_{\theta=\hat{\theta}(L)} \delta\hat{\theta} + \cdots\cdots$$

$$\approx C + g^T \cdot \delta\hat{\theta} + \frac{1}{2}\delta\hat{\theta}^T Q \delta\hat{\theta}$$

其中，g 为梯度阵，Q 为 Hessian 阵。

$$g = \frac{\partial J_{L+1}(\theta)}{\partial\theta}\bigg|_{\theta=\hat{\theta}(L)} \qquad Q = \frac{\partial^2 J_{L+1}(\theta)}{(\partial\theta)^2}\bigg|_{\theta=\hat{\theta}(L)}$$

将 $J_{L+1}(\hat{\theta}_{L+1})$泰勒展开后的式子对$\delta\hat{\theta}$求偏导并令结果等于 0，即可求出$\delta\hat{\theta}$，具体见下式。

$$\frac{\partial J_{L+1}(\hat{\theta}(L+1))}{\partial\delta\hat{\theta}} = 0$$

则解得：

$$\delta\hat{\theta} = -Q^{-1}g$$

故可得到递推表达式如下：

$$\hat{\theta}(L+1) = \hat{\theta}(L) - Q^{-1}g$$

已知 $J_{L+1}(\theta) = \frac{1}{2}\sum_{k=NL+1}^{(L+1)N} \varepsilon^2(k)$

将上式对θ分别求一阶导和二阶导，具体如下：

$$\left.\frac{\partial J_{L+1}}{\partial\theta}\right|_{\theta=\hat{\theta}(L)}=\left[\sum_{k=NL+1}^{(L+1)N}\varepsilon(k)\cdot\left.\frac{\partial\varepsilon(k)}{\partial\theta}\right|_{\theta=\hat{\theta}(L)}\right]^{T}$$

$$\left.\frac{\partial^2 J_{L+1}}{\partial(\theta)^2}\right|_{\theta=\hat{\theta}(L)}=\sum_{k=NL+1}^{(L+1)N}\left[\frac{\partial\varepsilon(k)}{\partial\theta}\right]^{T}\left.\frac{\partial\varepsilon(k)}{\partial\theta}\right|_{\theta=\hat{\theta}(L)}+\sum_{k=NL+1}^{(L+1)N}\varepsilon(k)\left.\frac{\partial^2\varepsilon(k)}{\partial(\theta)^2}\right|_{\theta=\hat{\theta}(L)}$$

由系统模型可得：

$$\varepsilon(k)=y(k)+\sum_{i=1}^{n}a_i y(k-i)-\sum_{i=1}^{n}b_i u(k-i)-\sum_{i=1}^{n}d_i\varepsilon(k-i)$$

将噪声$\varepsilon(k)$对θ求偏导，具体如下：

$$\frac{\partial\varepsilon(k)}{\partial\theta}=\left[\frac{\partial\varepsilon(k)}{\partial a1}\cdots\frac{\partial\varepsilon(k)}{\partial an}\ \frac{\partial\varepsilon(k)}{\partial b1}\cdots\frac{\partial\varepsilon(k)}{\partial bn}\ \frac{\partial\varepsilon(k)}{\partial d1}\cdots\frac{\partial\varepsilon(k)}{\partial dn}\right]$$

$$\left.\frac{\partial^2 J_{L+1}}{\partial(\theta)^2}\right|_{\theta=\hat{\theta}(L)}\approx\sum_{k=NL+1}^{(L+1)N}\left[\frac{\partial\varepsilon(k)}{\partial\theta}\right]^{T}\left.\frac{\partial\varepsilon(k)}{\partial\theta}\right|_{\theta=\hat{\theta}(L)}$$

将西塔二阶导的式子中略去二阶导数项，可得：

$$\left.\frac{\partial^2 J_{L+1}}{\partial(\theta)^2}\right|_{\theta=\hat{\theta}(L)}\approx\sum_{k=NL+1}^{(L+1)N}\left[\frac{\partial\varepsilon(k)}{\partial\theta}\right]^{T}\left.\frac{\partial\varepsilon(k)}{\partial\theta}\right|_{\theta=\hat{\theta}(L)}$$

由上述推导可知，若要求得梯度阵和 Hessian 矩阵，则需计算出

$$\left.\frac{\partial\varepsilon(k)}{\partial\theta}\right|_{\theta=\hat{\theta}(L)}\qquad(k=NL+n+1,n+2,\cdots n+(L+1)N)$$

由系统模型可得：

$$\varepsilon(k)=y(k)+\sum_{i=1}^{n}a_i y(k-i)-\sum_{i=1}^{n}b_i u(k-i)-\sum_{i=1}^{n}d_i\varepsilon(k-i)$$

将上式分别对各待辨识参数求偏导，如下：

$$\frac{\partial\varepsilon(k)}{\partial a_j}=y(k-j)-\sum_{i=1}^{n}d_i\left.\frac{\partial\varepsilon(k-i)}{\partial a_j}\right|_{\theta=\hat{\theta}_L}$$

$$\frac{\partial\varepsilon(k)}{\partial b_j}=-u(k-j)-\sum_{i=1}^{n}d_i\left.\frac{\partial\varepsilon(k-i)}{\partial b_j}\right|_{\theta=\hat{\theta}(L)}$$

$$\frac{\partial\varepsilon(k)}{\partial d_j}=-\varepsilon(k-j)-\sum_{i=1}^{n}d_i\left.\frac{\partial\varepsilon(k-i)}{\partial d_j}\right|_{\theta=\hat{\theta}(L)}$$

将上式代入式（2），可求得梯度阵 g 和 Hessian 矩阵 Q。于是可得完整的递推公式，在$\hat{\theta}_L$已知的情况下求得$\hat{\theta}_{L+1}$。

因此，为使递推过程正常进行，我们需要进行一系列的初始化。

初始化的内容包括：

1.根据系统模型采集到的一系列模拟输入和输出。

2.根据输入输出数据通过最小二乘法辨识出的参数$\alpha_0 \cdots \alpha_n$， $\beta_0 \cdots \beta_n$。

3.噪声$\varepsilon(1), \varepsilon(2), \cdots \varepsilon(n)$及其偏导$\frac{\partial \varepsilon(1)}{\partial \theta}\Big|_{\theta=\hat{\theta}_L}, \frac{\partial \varepsilon(2)}{\partial \theta}\Big|_{\theta=\hat{\theta}_L}, \cdots, \frac{\partial \varepsilon(n)}{\partial \theta}\Big|_{\theta=\hat{\theta}_L}$均设置为 0。

4．实验对象或参数

动态系统模型为：

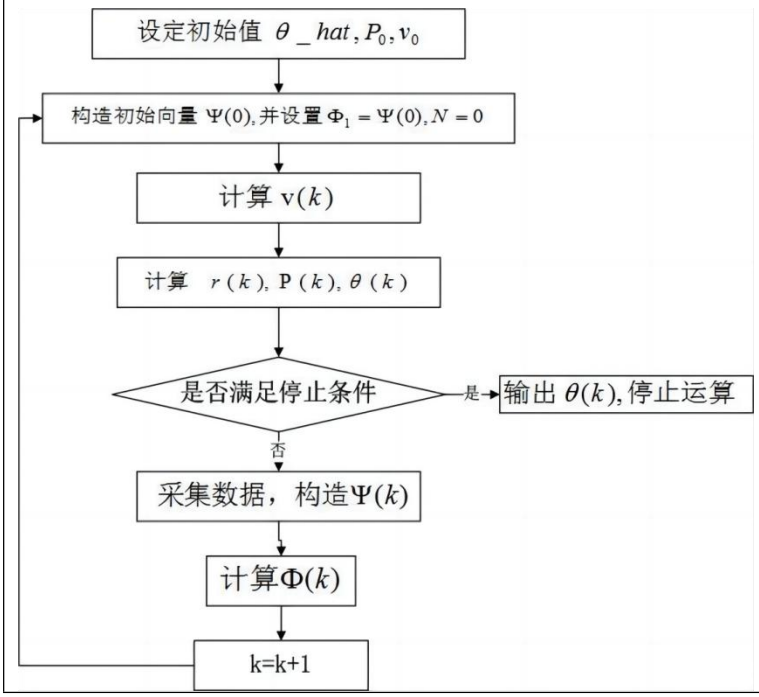
$$y(k) + a_1 * y(k-1) + a_2 * y(k-2) = b_1 * u(k-1) + b_2 * u(k-2) + E(k)$$

$$E(k) = \varepsilon(k) + d_1 * \varepsilon(k-1) + d_2 * \varepsilon(k-2)$$

其中模型参数a_1=-0.5；a_2=-0.2；b_1=1.5；b_2=-0.8；d_1=-0.8；d_2=0.3。噪声$\varepsilon(k)$是均值为 0，方差为 0.01 的高斯白噪声。U(k)是由 4 级移位寄存器产生的幅度为 1 的 M 序列。

5. 程序框图

递推极大似然估计法程序框图



6．程序代码

```
%% 极大似然参数估计
clear all
close all
% 产生仿真数据
n = 2;
total = 1000;
```

```
sigma = 0.1; % 噪声变量的均方根
% M 序列作为输入
z1 = 1; z2 = 1; z3 = 1; z4 = 0;
for i = 1:total
x1 = xor(z3, z4);
x2 = z1;
x3 = z2;
x4 = z3;
z(i) = z4;
if z(i) > 0.5
u(i) = -1;
else
u(i) = 1;
end
z1 = x1; z2 = x2; z3 = x3; z4 = x4;
end
figure(1);
stem(u, 'filled'), grid on;
title('Input Signal M Sequence');
% 系统输出
y(1) = 0; y(2) = 0;
v = sigma * randn(total, 1); % 噪声
y(1) = 1; y(2) = 0.01;
for k = 3:total
y(k) = 0.5 * y(k - 1) + 0.2 * y(k - 2) + u(k - 1) + 1.5 * u(k - 2) + v(k) - 0.8
* v(k - 1) + 0.3 * v(k - 2);
end
%初始化
theta0=0.001* ones(6,1); %参数
e1(1)=-0.5-theta0(1); e2(1)=-0.2-theta0(2); %误差初始化
e3(1)=1.0-theta0(3); e4(1)=1.5-theta0(4);
e5(1)=-0.8-theta0(5); e6(1)=0.3-theta0(6);
a_hat(1)=theta0(1); a_hat(2)=theta0(2); %参数分离
b_hat(1)=theta0(3); b_hat(2)=theta0(4);
c_hat(1)=theta0(5); c_hat(2)=theta0(6);
P0=eye(6,6); %矩阵 P 初始化
for i=1:n
yf(i)=0.1;
uf(i)=0.1;
vf(i)=0.1;
fai0(i,1)=-yf(i);
fai0(n+i,1)=uf(i);
fai0(2* n+i,1)=vf(i);
end
```

```
e(1)=1.0;
e(2)=1.0;
%递推算法
for i=n+1:total
pusai=[-y(i-1);-y(i-2);u(i-1);u(i-2);e(i-1);e(i-2)];
C=zeros(n* 3,n* 3);
Q=zeros(3* n,1);
Q(1)=-y(i-1);
Q(n+1)=u(i-1);
Q(2* n+1)=e(i-1);
for j=1:n
C(1,j)=-c_hat(j);
C(n+1,n+j)=-c_hat(j);
C(2* n+1,2* n+j)=-c_hat(j);
if j>1
C(j,j-1)=1.0;
C(n+j,n+j-1)=1.0;
C(2* n+j,2* n+j-1)=1.0
end
end
fai=C* fai0+Q;
K=P0* fai* inv(fai'* P0* fai+1);
P=[eye(6,6)-K* fai']* P0;
e(i)=y(i)-pusai'* theta0;
theta=theta0+K* e(i);
P0=P;
theta0=theta;
fai0=fai;
a_hat(1)=theta(1); a_hat(2)=theta(2);
b_hat(1)=theta(3); b_hat(2)=theta(4);
c_hat(1)=theta(5); c_hat(2)=theta(6);
e1(i)=-0.5-a_hat(1); e2(i)=-0.2-a_hat(2);
e3(i)=1.0-b_hat(1); e4(i)=1.5-b_hat(2);
e5(i)=-0.8-c_hat(1); e6(i)=0.3-c_hat(2);
end
% 绘制参数估计误差
figure(2);
plot(e1, '-r', 'LineWidth', 2); hold on;
plot(e2, '--g', 'LineWidth', 2); hold on;
plot(e3, ':b', 'LineWidth', 2); hold on;
plot(e4, '-.m', 'LineWidth', 2); hold on;
plot(e5, '-c', 'LineWidth', 2); hold on;
plot(e6, '--k', 'LineWidth', 2);
title('Parameter Estimation Error');
```

```
xlabel('times');
ylabel('error');
legend('e1', 'e2', 'e3', 'e4', 'e5', 'e6');
hold off;

% 绘制输出误差
figure(3);
plot(e, '-r', 'LineWidth', 2); % 修改为红色实线
title('Output Error');
xlabel('times');
ylabel('error');
%% Newton-Raphson 法
clear all
close all
clc
randn('seed',100)
sigma=0.1;
v=sigma*randn(1,16); %产生一组 16 个 N(0，1)的高斯分布的随机噪声
n=2;
N=200;
epsilon=0.001;%迭代终止条件
%M 序列产生程序
L=15; %M 序列的周期
y1=1;y2=1;y3=1;y4=0; %四个移位寄存器的输出初始值
for i=1:L
x1=xor(y3,y4);
x2=y1;
x3=y2;
x4=y3;
y(i)=y4;
if y(i)>0.5,u(i)=-1;
else u(i)=1;
end
y1=x1;y2=x2;y3=x3;y4=x4;
end
figure
stem(u),grid on
title('输入信号 M 序列')
%最小二乘辨识程序
z=zeros(1,16); %定义输出观测值的长度
for k=3:16
y(k)=0.5*y(k-1)+0.2*y(k-2)+1.0*u(k-1)+1.5*u(k-2)+1*v(k); %观测值
end
H=[-y(2) -y(1) u(2) u(1) ;-y(3) -y(2) u(3) u(2);-y(4) -y(3) u(4) u(3);
```

```
-y(5) -y(4) u(5) u(4);-y(6) -y(5) u(6) u(5);-y(7) -y(6) u(7) u(6);
-y(8) -y(7) u(8) u(7);-y(9) -y(8) u(9) u(8);-y(10) -y(9) u(10) u(9);
-y(11) -y(10) u(11) u(10);-y(12) -y(11) u(12) u(11);
-y(13) -y(12) u(13) u(12);-y(14) -y(13) u(14) u(13);
-y(15) -y(14) u(15) u(14)];
%给出样本观测矩阵
Z=[y(3);y(4);y(5);y(6);y(7);y(8);y(9);y(10);y(11);y(12);y(13);y(14);
y(15);y(16)]
%计算参数
c=inv(H'*H)*H'*Z;
%分离参数
a1=c(1),a2=c(2),b1=c(3),b2=c(4)
24
d1=0.1;d2=0.1;
theta0=[a1 a2 b1 b2 d1 d2]';
v(1)=0;v(2)=0;
vda1(1)=0;vda2(1)=0;vdb1(1)=0;vdb2(1)=0;vdd1(1)=0;vdd2(1)=0;
vda1(2)=0;vda2(2)=0;vdb1(2)=0;vdb2(2)=0;vdd1(2)=0;vdd2(2)=0;
j=1;
theta1=zeros(6,1);
vdtheda=zeros(6,1);
while j<501
for i=1:N+2
x1=xor(y3,y4);
x2=y1;
x3=y2;
x4=y3;
y(i)=y4;
if y(i)>0.5,u(i)=-1;
else u(i)=1;
end
y1=x1;y2=x2;y3=x3;y4=x4;
end
y(1)=0;y(2)=0;
v=randn(1,N+3);
y(1)=1;y(2)=0.01;
for k=3:N+2
y(k)=0.5*y(k-1)+0.2*y(k-2)+1.0*u(k-1)+1.5*u(k-2)+v(k)-0.8*v(k-1)+0.3*v(k-2)
;
end
Jd=0;Jdd=0;
a1=theta0(1);a2=theta0(2);b1=theta0(3);b2=theta0(4);d1=theta0(5);d2=theta0(
6);
for k=3:N+2
```

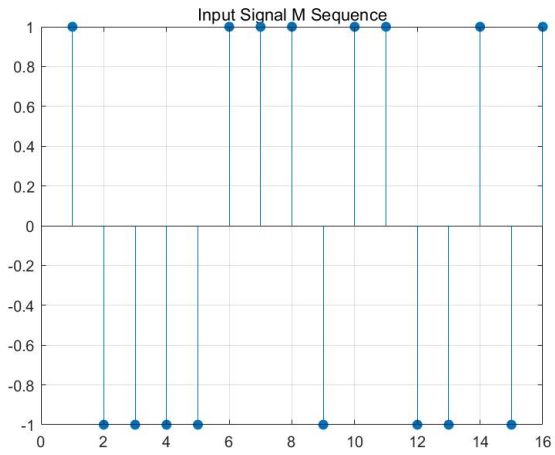
```
v(k)=y(k)+a1*y(k-1)+a2*y(k-2)-b1*u(k-1)-b2*u(k-2)-d1*v(k-1)-d2*v(k-2);%
vda1(k)=y(k-1)-d1*vda1(k-1)-d2*vda1(k-2);
vda2(k)=y(k-2)-d1*vda2(k-1)-d2*vda2(k-2);
vdb1(k)=-u(k-1)-d1*vdb1(k-1)-d2*vdb1(k-2);
vdb2(k)=-u(k-2)-d1*vdb2(k-1)-d2*vdb2(k-2);
vdd1(k)=-v(k-1)-d1*vdd1(k-1)-d2*vdd1(k-2);
vdd2(k)=-v(k-2)-d1*vdd2(k-1)-d2*vdd2(k-2);
vdtheda=[vda1(k),vda2(k),vdb1(k),vdb2(k),vdd1(k),vdd2(k)]';
Jd=Jd+v(k)*vdtheda;%梯度阵
Jdd=Jdd+vdtheda'*vdtheda;%H 矩阵
end
theta1=theta0;
theta0=theta0-inv(Jdd)*Jd;
a1=theta0(1);a2=theta0(2);b1=theta0(3);b2=theta0(4);d1=theta0(5);d2=theta0(
6);
e1(j)=a1; e2(j)=a2;
e3(j)=b1; e4(j)=b2;
e5(j)=d1; e6(j)=d2;
e7(j)=-0.5-a1; e8(j)=-0.2-a2;
e9(j)=1.0-b1; e10(j)=1.5-b2;
e11(j)=-0.8-d1; e12(j)=0.3-d2;
j=j+1;
v(1)=v(N+1);v(2)=v(N+2);
vda1(1)=vda1(N+1);vda2(1)=vda2(N+1);vdb1(1)=vdb1(N+1);
vdb2(1)=vdb2(N+1);vdd1(1)=vdd1(N+1);vdd2(1)=vdd2(N+1);
vda1(2)=vda1(N+2);vda2(2)=vda2(N+2);vdb1(2)=vdb1(N+2);
vdb2(2)=vdb2(N+2);vdd1(2)=vdd1(N+2);vdd2(2)=vdd2(N+2);
end
% 绘制 Newton-Raphson 法的结果
figure(4);
plot(e1, '-r', 'LineWidth', 2); hold on;
plot(e2, '--g', 'LineWidth', 2); hold on;
plot(e3, ':b', 'LineWidth', 2); hold on;
plot(e4, '-.m', 'LineWidth', 2); hold on;
plot(e5, '-c', 'LineWidth', 2); hold on;
plot(e6, '--k', 'LineWidth', 2);
title('Identification Results');
legend('e1', 'e2', 'e3', 'e4', 'e5', 'e6');
hold off;

figure(5);
plot(e7, '-r', 'LineWidth', 2); hold on;
plot(e8, '--g', 'LineWidth', 2); hold on;
plot(e9, ':b', 'LineWidth', 2); hold on;
```

```
plot(e10, '-.m', 'LineWidth', 2); hold on;
plot(e11, '-c', 'LineWidth', 2); hold on;
plot(e12, '--k', 'LineWidth', 2);
title('Identification Estimation Error Results');
xlabel('Iteration Number');
ylabel('Estimation Error');
legend('e7', 'e8', 'e9', 'e10', 'e11', 'e12');
hold off;
plot(e10, '-.md', 'LineWidth', 2); hold on;
plot(e11, '-c*', 'LineWidth', 2); hold on;
plot(e12, '--ko', 'LineWidth', 2);
title('Identification Estimation Error Results');
xlabel('Iteration Number');
ylabel('Estimation Error');
legend('e7', 'e8', 'e9', 'e10', 'e11', 'e12');
hold off;
```

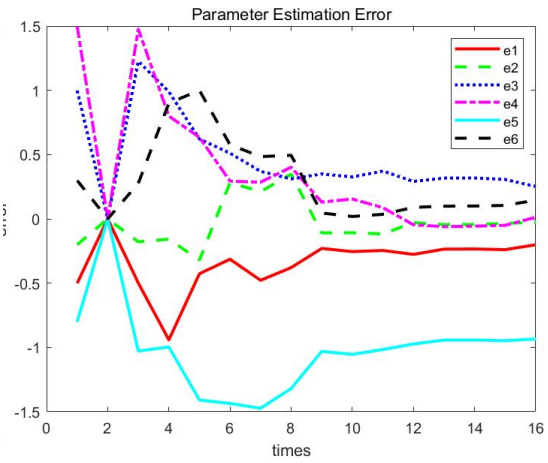
7. 实验结果及分析

(1) 极大似然估计法

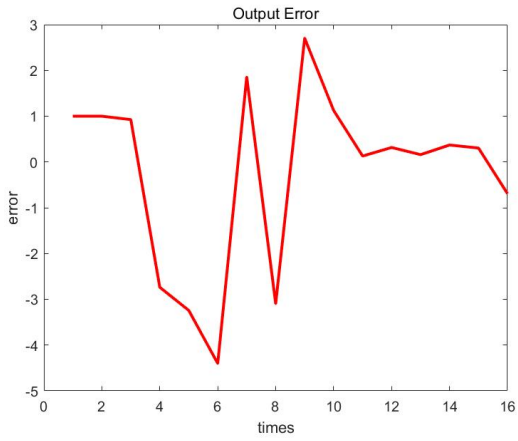
输入信号（长度 16 的 M 序列）



参数估计误差

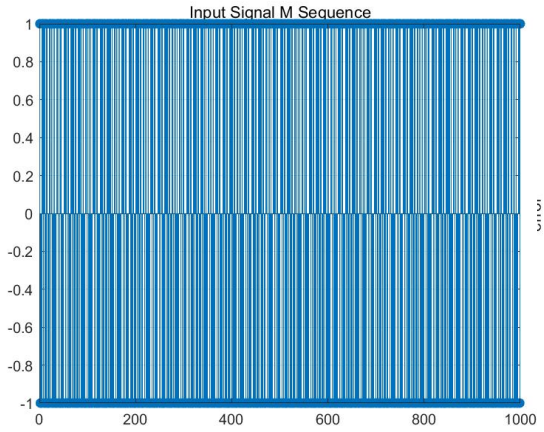


输出误差

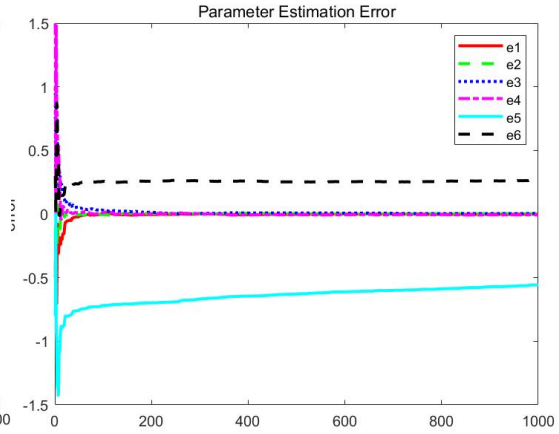


由于在输入为长度 16 的 M 序列时，输出无法稳定，推测是输入信号长度太小了，所以改成长度为 1000 的 M 序列重新测量得到结果如下（此时输出稳定）：

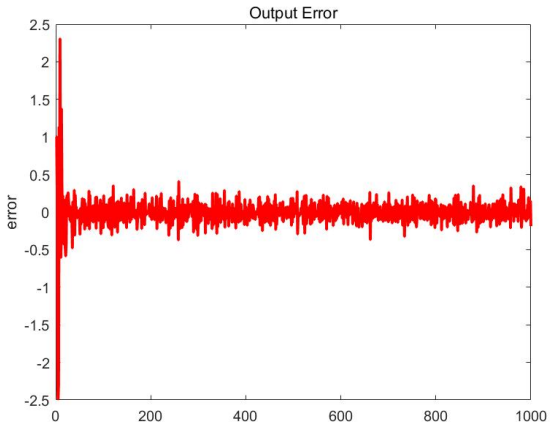
输入信号（长度 1000 的 M 序列）



参数估计误差



输出误差

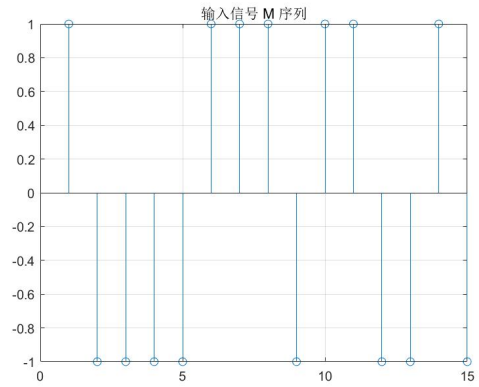


极大似然辨识参数：

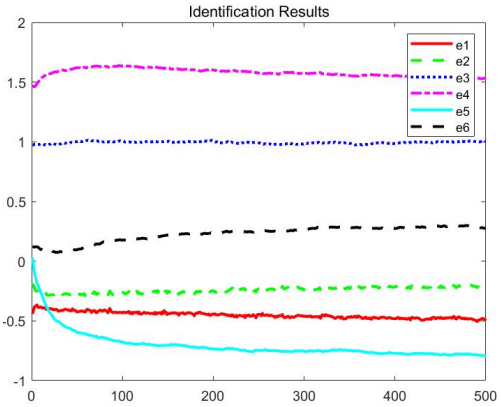
参数	a1	a2	b1	b2	d1	d2
真值	-0.5	-0.2	1.0	1.5	-0.8	0.3
估计值（1000）	-0.488	-0.198	0.989	1.503	-0.173	0.094

(2) Newton-Raphton 法

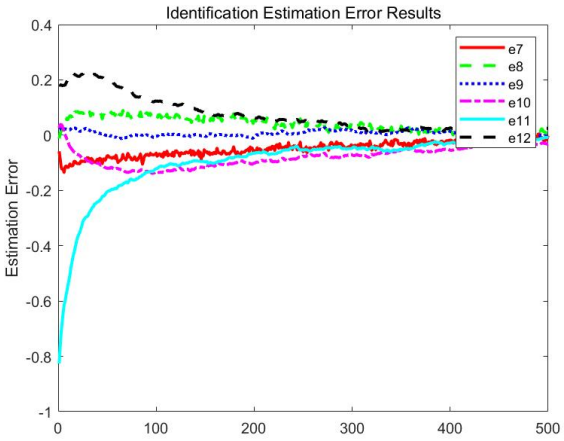
输入信号



辨识结果



辨识估计误差结果



Newton-Raphton 法辨识参数：

参数	a1	a2	b1	b2	d1	d2
真值	-0.5	-0.2	1.0	1.5	-0.8	0.3
估计值（16）	-0.4717	-0.2295	0.975	1.573	-0.7743	0.289

分析：

Newton-Raphton 法在短信号输入（例如一开始的长度为 16 的 M 序列输入信号）下就能提供稳定的估计结果，而极大似然估计发不能，说明 Newton-Raphton 法对初始值的敏感度较低。但极大似然辨识法在长信号输入下表现出色，参数估计的精度得到显著提高，这表明了在丰富的输入信息下，极大似然法能够更准确地捕获系统的动态行为。

8．结论

极大似然参数辨识法与 Newton-Raphton 法的比较分析：

（1）极大似然参数辨识法：

极大似然参数辨识法依赖于概率模型，它通过最大化似然函数来估计模型参数，该方法在统计特性已知的情况下能够提供有效的参数估计。

➢优点：

- 在噪声模型正确指定的前提下，极大似然估计能够得到一致且有效的参数估计值。
- 对于大样本数据，极大似然估计通常具有良好的渐近性质。

➢缺点：

- 需要对噪声的统计特性有准确的假设，若假设错误，估计结果可能会产生偏差。
- 在实际应用中可能需要较多的计算量，尤其是当模型复杂时。

（2）Newton-Raphton 法：

Newton-Raphton 是一种迭代算法，它通过对目标函数的一阶和二阶导数信息的利用来快速找到函数的零点，常用于求解极大似然估计中的最优化问题。

➢优点：

- 收敛速度快，尤其是在初始点接近真实值时，牛顿-拉弗森法通常具有二次收敛性。
- 灵活性高，能够应用于多种类型的优化问题。

➢缺点：

- 对初始值敏感，若初始估计值选取不当，可能导致迭代不收敛。
- 需要计算目标函数的一阶和二阶导数，对于复杂函数而言，这可能是一个挑战。

（3）总结：

极大似然参数辨识法和 Newton-Raphton 法各有优势和不足。极大似然法在噪声模型已知且准确的情况下提供了一种强大的参数估计工具，而 Newton-Raphton 则以其快速的收敛速度

在优化问题中得到了广泛应用。在实际选择时，应当根据具体问题的性质、所需的计算资源以及对估计精度的要求来决定使用哪种方法。