

**过程建模与系统辨识课程报告**

**基于子空间辨识方法、递推最小二乘辨识和递推极大似然估计法的电吹风系统参数辨识与MATLAB仿真分析**

**学 院： 未来技术学院**

**课 程： 过程建模与系统辨识**

**指导老师： 万雄波**

**学 号： 20211003337**

**姓 名： 曾康慧**

**2023 年 12 月 17 日**

# 摘要

本文主要研究了子空间辨识方法（SMI）、递推最小二乘辨识（RLS）和递推极大似然估计法(RML)在系统参数辨识中的应用。我们选择了Dryer数据集作为研究对象，该数据集的物理模型类似于电吹风，系统输入是加热器的电压，系统输出是出口空气的温度。通过MATLAB仿真，我们对这三种方法进行了详细的比较和分析，结果表明，这三种方法在系统参数辨识中都有良好的性能，为进一步的研究提供了有价值的参考。

**关键词**：系统辨识，子空间辨识方法，递推最小二乘辨识，递推极大似然估计法，Dryer数据集，MATLAB仿真

目录

[摘要 2](#_Toc153645998)

[一、 Dryer实例研究 5](#_Toc153645999)

[1. 问题描述 5](#_Toc153646000)

[2. 研究意义 5](#_Toc153646001)

[二、 子空间辨识方法 6](#_Toc153646002)

[1. 研究背景 6](#_Toc153646003)

[2. 基本思想 6](#_Toc153646004)

[3. 问题描述 7](#_Toc153646005)

[4. 辨识步骤 7](#_Toc153646006)

[4.1 确定或者 8](#_Toc153646007)

[4.2 计算系统矩阵 8](#_Toc153646008)

[5. SMI 基本方法介绍 9](#_Toc153646009)

[6. 子空间辨识方法实例研究 10](#_Toc153646010)

[三、 最小二乘辨识算法 15](#_Toc153646011)

[1. 模型选择 15](#_Toc153646012)

[2. 阶次辨识 15](#_Toc153646013)

[3. 白噪声检验 16](#_Toc153646014)

[4. 参数辨识 19](#_Toc153646015)

[四、 递推极大似然估计法 22](#_Toc153646016)

[1. 模型选择 22](#_Toc153646017)

[2. 白噪声检验 22](#_Toc153646018)

[3. 参数辨识 24](#_Toc153646019)

[五、 三种辨识方法比较 27](#_Toc153646020)

[参考文献 28](#_Toc153646021)

[附录 29](#_Toc153646022)

[ARX\_main.m 29](#_Toc153646023)

[RLS.m 31](#_Toc153646024)

[estimate\_z.m 32](#_Toc153646025)

[AIC\_main.m 33](#_Toc153646026)

[plot\_AIC.m 34](#_Toc153646027)

[ARX\_RML\_main.m 34](#_Toc153646028)

[RML.m 37](#_Toc153646029)

# 一、 Dryer实例研究

## 1. 问题描述

在 DaISy 提供的实际数据集上进行辨识，选取其中 Mechanical Systems 分类下的dryer 数据集，使用子空间辨识方法、常用的最小二乘辨识和递推极大似然估计法方法分别进行仿真分析。

dryer 数据集的物理模型类似于电吹风系统，如图 1 所示，系统输入是加热器的电压，系统输出是出口空气的温度。

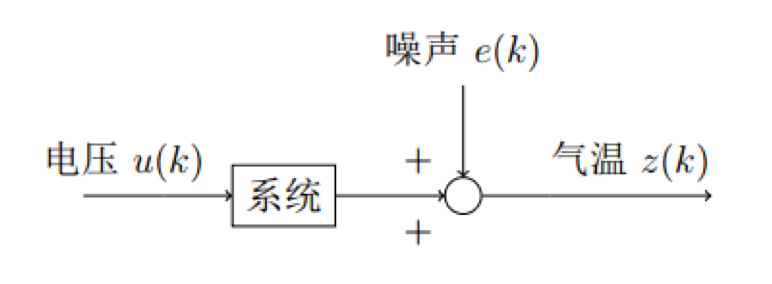


图 1 dryer 模型

## 2. 研究意义

通过子空间辨识方法（SMI）、递推最小二乘辨识（RLS）和递推极大似然估计法(RML)这三种不同的系统辨识方法，对电吹风系统进行了系统参数辨识与仿真分析不仅可以提高我们对这些系统辨识方法的理解，而且可以为我们提供一个实际的应用场景，以便我们更好地理解这些方法在实际问题中的应用。同时，电吹风系统的仿真分析对于电吹风系统的设计和优化具有重要的参考价值，借此我们可以更深入地理解这些系统辨识方法在处理实际问题时的性能和效率。

# 二、 子空间辨识方法

## 1. 研究背景

用于系统辨识的传统方法，如预测误差方法（PEM）和辅助变量法（IVM）都以优化思想为基础，其中系统参数都是通过最小化某个恰当的目标函数得到，这些方法存在固有缺陷：（1）由于目标函数与系统参数之间一般呈非线性关系，因此需要迭代优化；（2）由于存在局部极值和非凸性，辨识结果一般对优化算法的初始条件敏感；（3）多变量系统辨识参数化比较困难。子空间模型辨识方法（SMI）可以避免这些问题。

SMI(Subspace Identification)方法综合了系统理论，线性代数和统计学三方面的思想[1]，特点是直接由输入和输出数据辨识线性时不变状态空间模型，相比于传统的 LTI 辨识方法，诸如 PEM 和 IVM，其优点如下：（1）不需要参数化；（2）不需要迭代优化；（3）算法实现仅依赖于一些简单可靠的线性代数工具，如 QR 分解、SVD 分解等；（4）直接估计状态空间模型，适宜于多变量系统辨识[2]。

## 2. **基本思想**

SMI 的基本思想可以追溯到 20 世纪 60 年代提出的状态空间实现理论。基于该理论，系统的状态空间表达式可以由脉冲响应系数组成的 Hankel 矩阵估计得到[3]，但是，因为获取可靠的脉冲响应估计比较困难，人们开始研究直接由系统的输入输出数据辨识状态空间模型的方法．其算法的基本思路就是由输入输出Hankel 投影的行子空间和列子空间来获取模型参数，“子空间辨识”因此得名。

## 3. 问题描述

SMI 算法直接由给定的输入输出数据

式中，和分别是过程在 时刻的状态向量，输入观测向量和输出观测向量；和分别是系统的输出测量噪声和过程噪声；各矩阵具有相应的维数，此外，为保证辨识结果具有好的统计特性，做出如下假设：

（1）系统是渐进稳定的，即Ａ的特征值严格在单位圆内；

（2） 可观测；可控

（3） 输入确定性的拟平稳序列，且和过程噪声和测量噪声无关；

（4） 过程噪声和测量噪声都是平稳零均值白噪声序列，且

式中，E 表示期望算子，，如果 ；，如果 。

基于此模型，子空间辨识算法的目的可以归结为：通过给定的输入输出观测序列（N 表示样本个数），确定系统矩阵 及协方差矩阵

## 4. 辨识步骤

子空间辨识方法一般由两步组成：（1）确定扩展可观性矩阵或者估计出系统的状态序列；（2）计算系统矩阵

### 4.1 确定或者

实现这一步有两种策略。

策略一：依据子空间等价原理。首先基于系统的输入输出矩阵等式

通过使用（斜）投影或者辅助变量消除噪声以及未来输入，得到

式中，是估计的状态序列。由线性代数知识可知，在满秩时，的列空间与的列空间重合，且其维数等于系统的阶次 n，这时可以由𝑂J的列空间计算得到。具体说，就是对𝑂J进行 SVD 分解

式中，𝑆 ∈ 𝑅1×1，则，进而得到状态序列，.

策略二：依据基于 CCA 的随机实现理论，通过分解某个条件协方差矩阵可以得到扩展可观性矩阵和廓镇可控性矩阵，同时也可以估计得到系统的状态序列 ，它被表达为 Hankel 矩阵的线性组合。

### 4.2 **计算系统矩阵**

系统矩阵的计算有两种方法[4]，实现法和回归法。实现法是在没有估计状态序列的情况下，分步计算系统矩阵。先由直接计算矩阵 A 和 C，然后在根据Toeplitz 矩阵构造 LS 问题计算系统矩阵 B 和 D。回归法基于估计的状态序列，

对如下线性方程组

进行最小二乘求解，即

另外，噪声的协方差矩阵 Q、 R、 S 可以由该 LS 问题的残差估计得到，即

## 5. SMI 基本方法介绍

SMI 的基本算法有 3 中，即 MOESP、N4SID 和 CVA。基本算法都是基于离散时间线性状态空间模型提出的，而且要求系统开环。前两者源于状态空间的实现理论，后者源于 Akaike 对 CCA 所做的工作。相关的算法都始于时间序列辨识，之后逐步发展到带有外部输入信号的系统。

在算法的策略上，请两者都是用策略一确定扩展可观性矩阵，CVA 使用策略二。在计算系统矩阵方面，MOESP 使用实现法，而 CVA 和 N4SID 使用回归法。

在算法实现方面，三者都可以分作 4 步：第一步是构建综合“过去”和“未来”的输入输出 Hankel 矩阵；第二步是对该 Hankel 矩阵进行 QR 分解；第三步是通过对某个特定矩阵进行 SVD 分解得到扩展可观性矩阵；第四步，计算系统矩阵。

在算法的统计特性方面，算法的统计特性依赖于对输入信号，噪声性质以及真实系统所作的假设。其中，CVA 估计是一致的，CVA 估计和 PEM 估计一样渐进有效；3 种方法在子空间估计方面的一致性研究表明，当系统仅有测量噪声时，输入信号只要持续激励就可以保证估计的一致性；当系统还存在过程噪声时，PE条件就不充分了，但是此时输入信号如果是白噪声或者低阶 ARMA 过程，则估计具有一致性。

在算法性能方面，早期多使用波特图显示偏差误差，使用极点分布图显示方差误差。也可以使用 Kullback 信息矩阵作为模型逼近误差的测度，优势在于它可以避免计算参数估计误差的 Fisher 信息矩阵或其协方差矩阵，还可以利用浮点计算次数和测试数据集上的仿真误差与预测误差比较算法的计算复杂性和泛化性能。

另外，辨识模型的稳定性也是 一个重要方面。就子空间辨识而言，通过将扩展可观性矩阵的最后1行置0来保证模型稳定性。对一个最小二乘问题进行正则化也可以解决模型稳定性问题，其中正则项包含了矩阵Ａ和一个权矩阵。通过对一个最小二乘问题施加一个稳定性约束来解决稳定性问题。但是保稳定性的辨识算法需要小心使用，否则不稳定系统也会被辨识为一个稳定系统。

## 6. 子空间辨识方法实例研究

本小节将在实例研究的基础上对上述的基本子空间辨识算法与传统辨识算法 PEM 作个比较。前面说过，SMI 较 PEM 有很多优点，譬如不需要参数化，不需要迭代优化，易于多变量系统辨识等。

使用 PEM、CVA、MOESP 和 N4SID 算法，对系统进行辨识。使用 matlab 工具箱**system identification toolbox**中自带的算法。四种算法对模型的阶次辨识结果相同，均为 3，如图二所示。

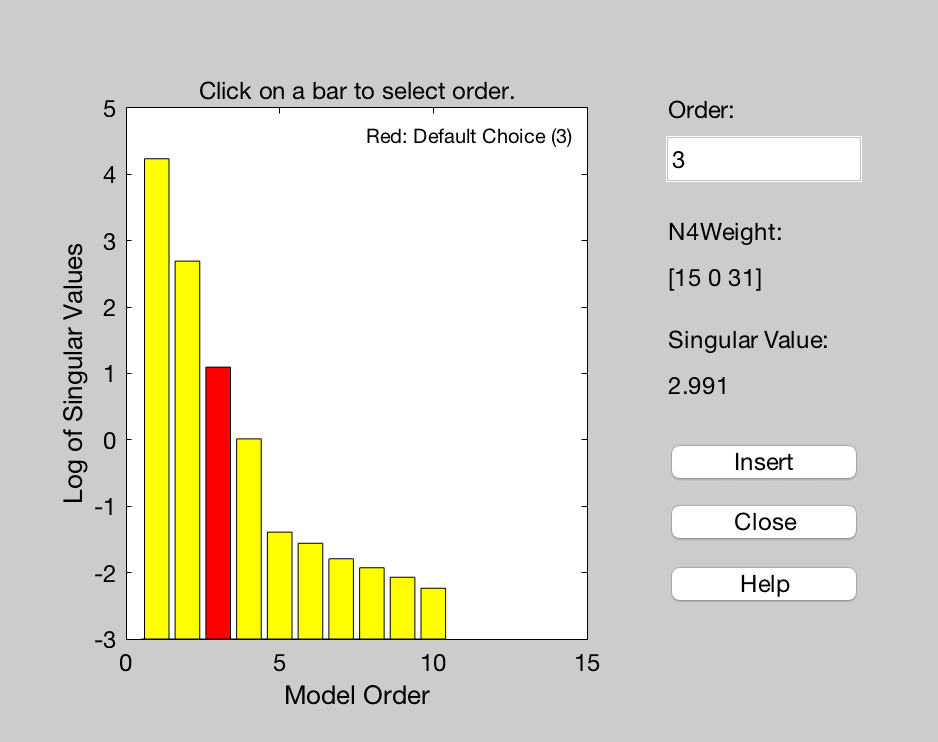


图 2 模型阶次辨识结果

由上图，3 阶之后落差明显，则 3 阶是最好的辨识阶次，说明四种算法在此数据集上的模型阶次辨识效果相同。

在参数调整较好的情况下，四种算法对应的运算时间如表 1 所示。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 算法 | PEM | CVA | MOESP | N4SID |
| 运行时间(s) | **2.3** | **1.5** | **1.0** | **0.4** |

表 1 四种算法运行时间比较

由上表可以看出，SMI 算法叫 PEM 算法具有更快的运算速度，这是由于SMI 自身算法结构比 PEM 算法简单，仅依靠简单的线性代数工具，而在 SMI 算法内部，CVA 算法速度最慢，MOESP 和 N4SID 其次。

用四种算法辨识的模型计算输出，与原始输出数据比较，其与原始数据的符合度如图 3~图 6 所示。

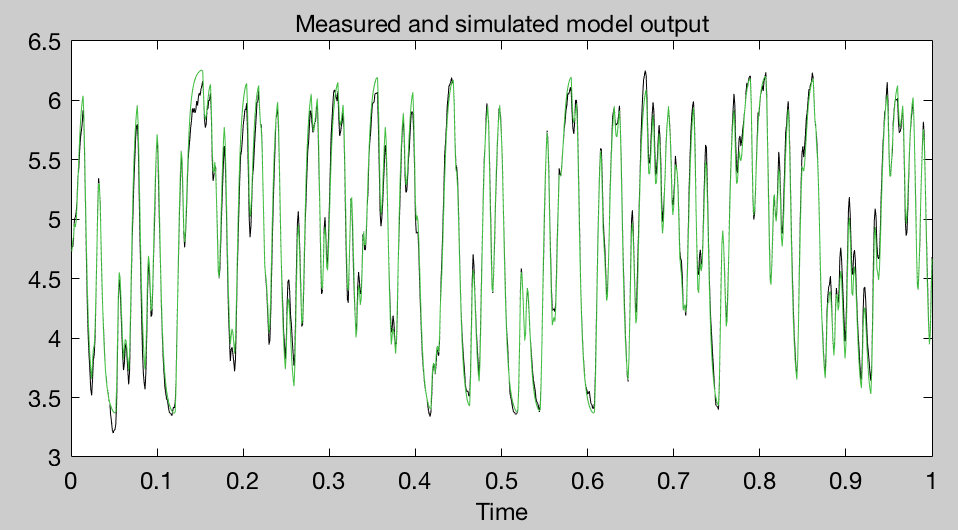


图 3 PEM 与原始输出数据比较

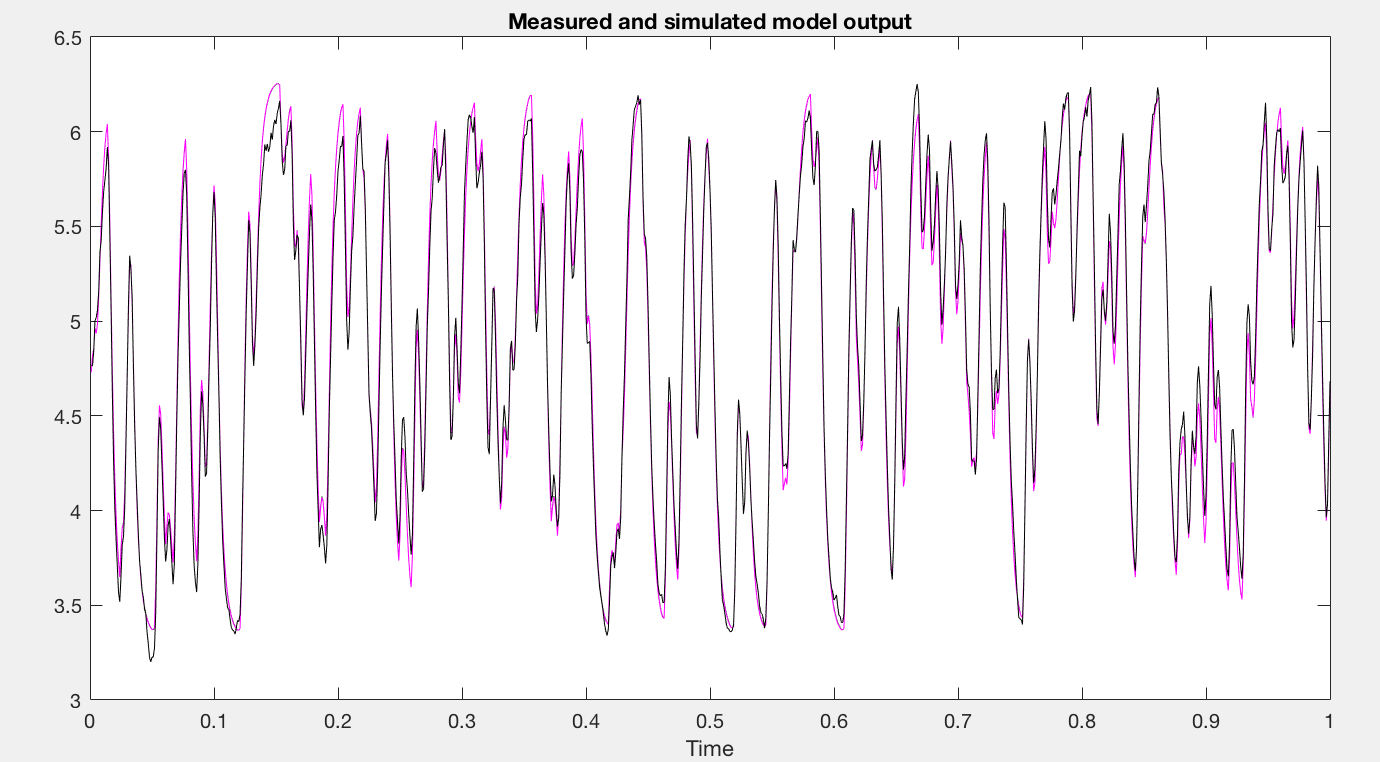


图 4 CVA 与原始输出数据比较

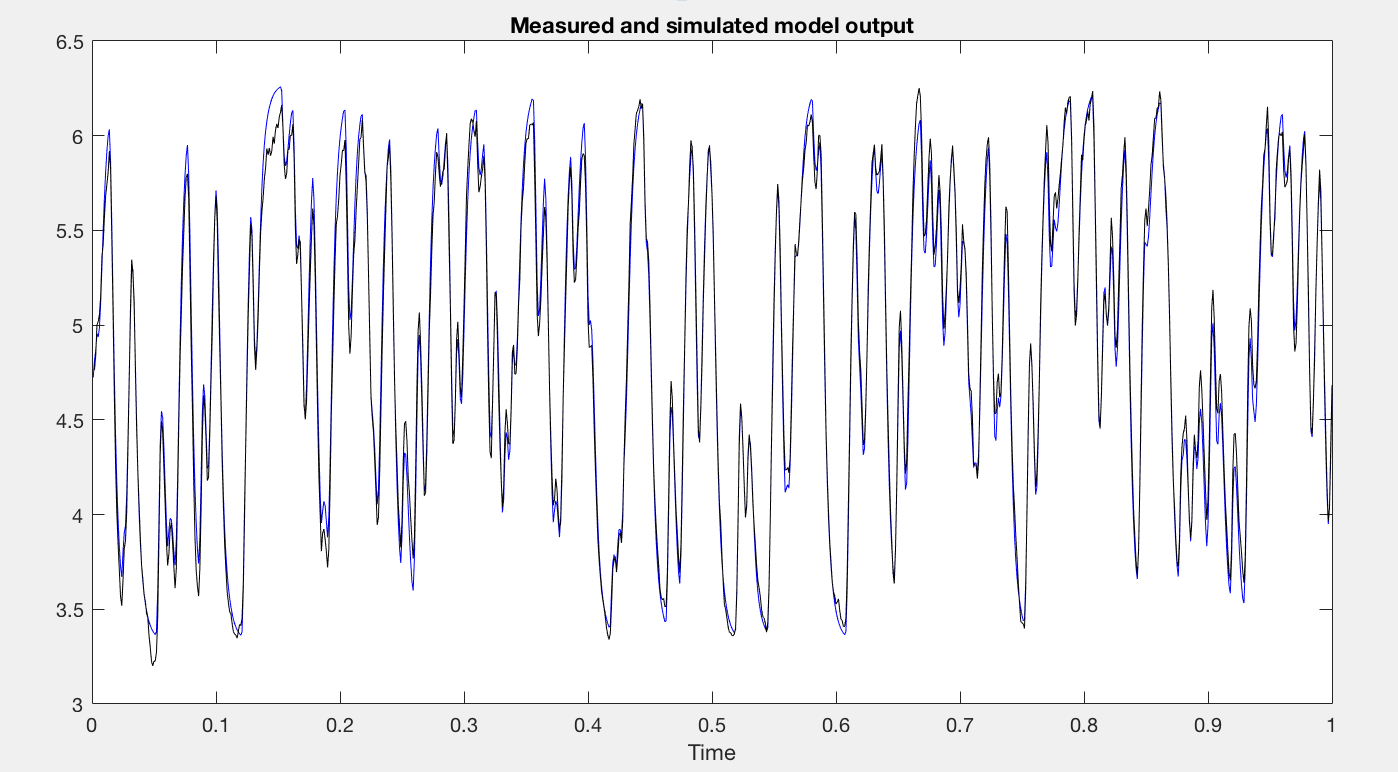


图 5 MOESP 与原始输出数据index-7_3.png

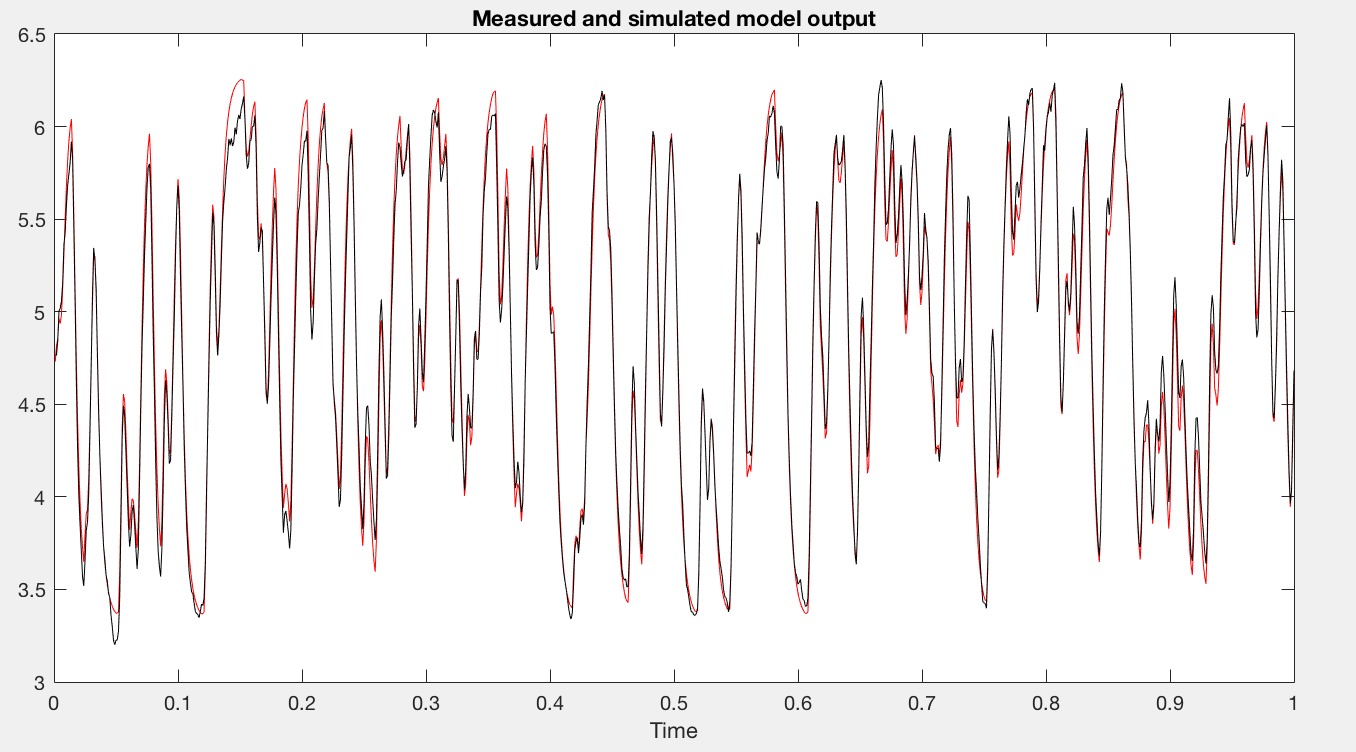


图 6 N4SID 与原始输出数据

上图中，黑色线条代表原始输出，彩色线条代表辨识输出，可见，辨识结果与原始输出有较好的符合性，说明了辨识算法的有效性。

对四种算法的结果与原始结果进行结果契合度比较，比较结果如表 2 所示。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 算法 | PEM | CVA | MOESP | N4SID |
| 契合度(%) | **88.40** | **88.38** | **88.45** | **88.45** |

表 2 四种算法契合度比较

由上表可以看出，在对相同的对象进行辨识时，MOESP 和 N4SID 的辨识契合度最高，效果最好；PEM 和 CVA 其次。说明对此数据集 SMI 辨识方法有效，但同时也存在不足，可能是由于模型本身并不是十分简单的模型，需要进行更加附加的辨识过程才能得到更好而效果。但是相对于 PEM 方法而言，SMI 自身算法结构比 PEM 算法简单，仅依靠简单的线性代数工具，运行读快于 SMI 算法，因此在对辨识契合度要求不高的情况下，SMI 算法很适用。

# 三、 最小二乘辨识算法

## 1. **模型选择**

选取工程实际中常用的 ARX 模型，设其模型结构为：

其中，𝑣(𝑘)是均值为 0，方差为 1 的白噪声序列，

将其写为最小二成形式则为：

待辨识参数为：

index-9_3.png

## **2.** **阶次辨识**

利用 AIC 定阶法对模型进行阶次辨识，对 ARX 模型，其准则如下：

式中，L 为数据长度，为噪声方差估计值，𝐽(L)为递推至 L 步的损失函数值，为模型阶次。选择使变化变小的作为模型阶次。

取 dryer 数据集的所有输入和输出，= 1,2, ⋯ ,10， = 1,2, ⋯ ,10，使用递归最小二乘法求得噪声方差估计值，带入 AIC 准则求得不同下的 AIC 变化曲线，如图 7 所示。

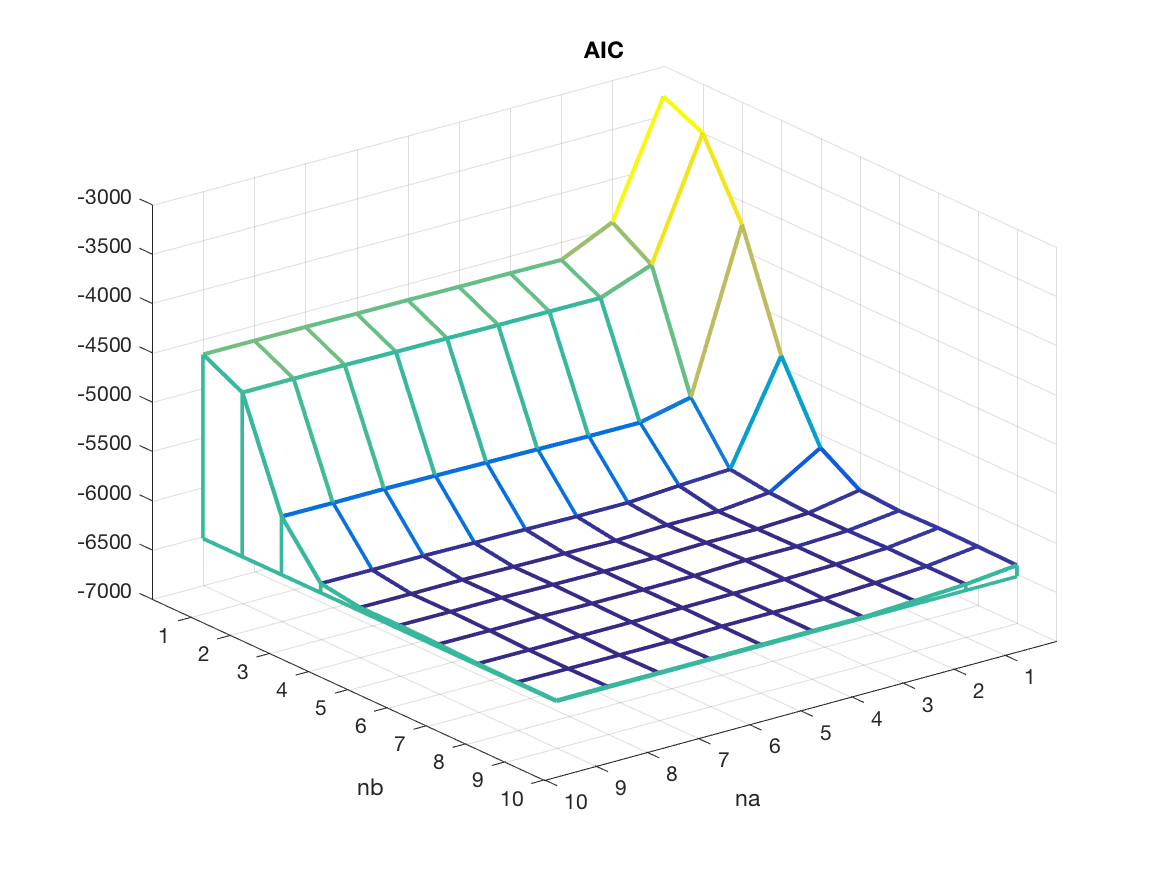
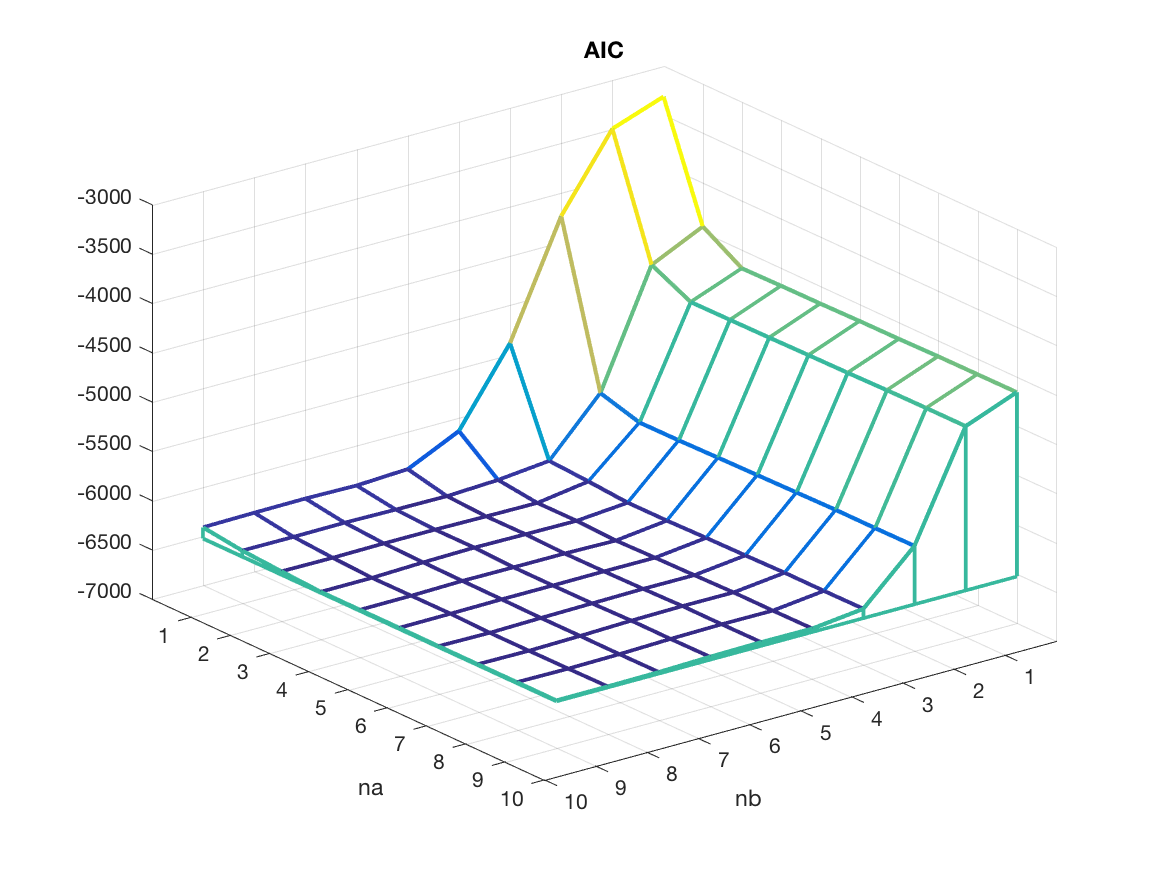


图 7 AIC 在不同阶次的变化曲线

由图 7 可以看出，当大于 2，大于 4 之后，AIC 变化非常小，故得到模型辨识阶次为：

## 3. **白噪声检验**

利用自相关系数检验法对模型残差序列进行白色性检验，如果能判断是零均值白噪声序列，则认为模型是“好”的。

根据 AIC 定阶法的结果，取计算残差序列的自相关系数为：

其中，是 的自相关函数，绘出自相关系数如图 8 所示。

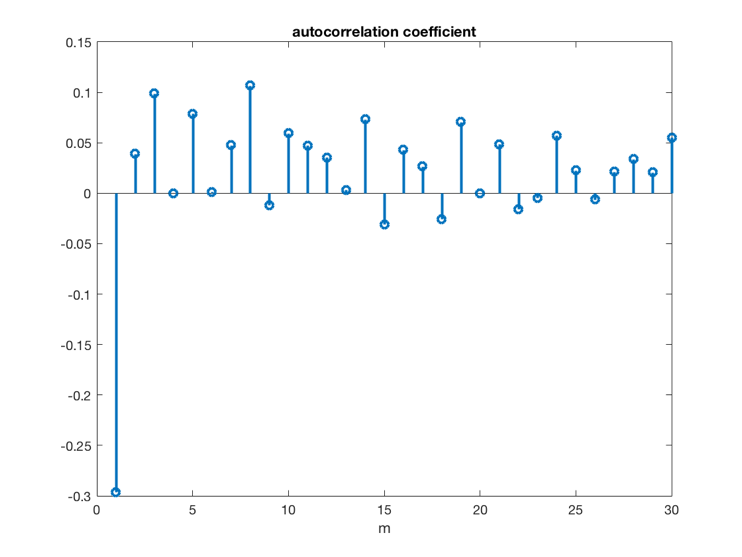


图 8 残差序列ε 𝑘 的自相关系数

取 m=25，利用下面两个不等式来判断残差序列的白色性：

1. （白色性阈值）
2. ，置信度取 0.05

经过实验，有

（不满足条件）

（满足条件）

根据残差可知，

如果 ，当有一个不等式或两个不等式都成立时，则是白噪声序列，可信度为 95%。因此本实验中的残差序列满足以上条件，说明是白噪声序列，也进一步说明选取 ARX 模型可以很好地表示原系统。

## 4. 参数辨识



程序框图

采用递推最小二乘法对系统进行辨识，取根据 AIC 定阶法的结果，取，得到的辨识参数如下：

参数𝜃随迭代次数的变化曲线如图 9 所示。其中，实线代表迭代过程中参数实际值，虚线代表最终参数估计值，从图中可以看出，参数均很快收敛到一个稳定值，说明系统对最小二乘辨识是可收敛的，有效的。

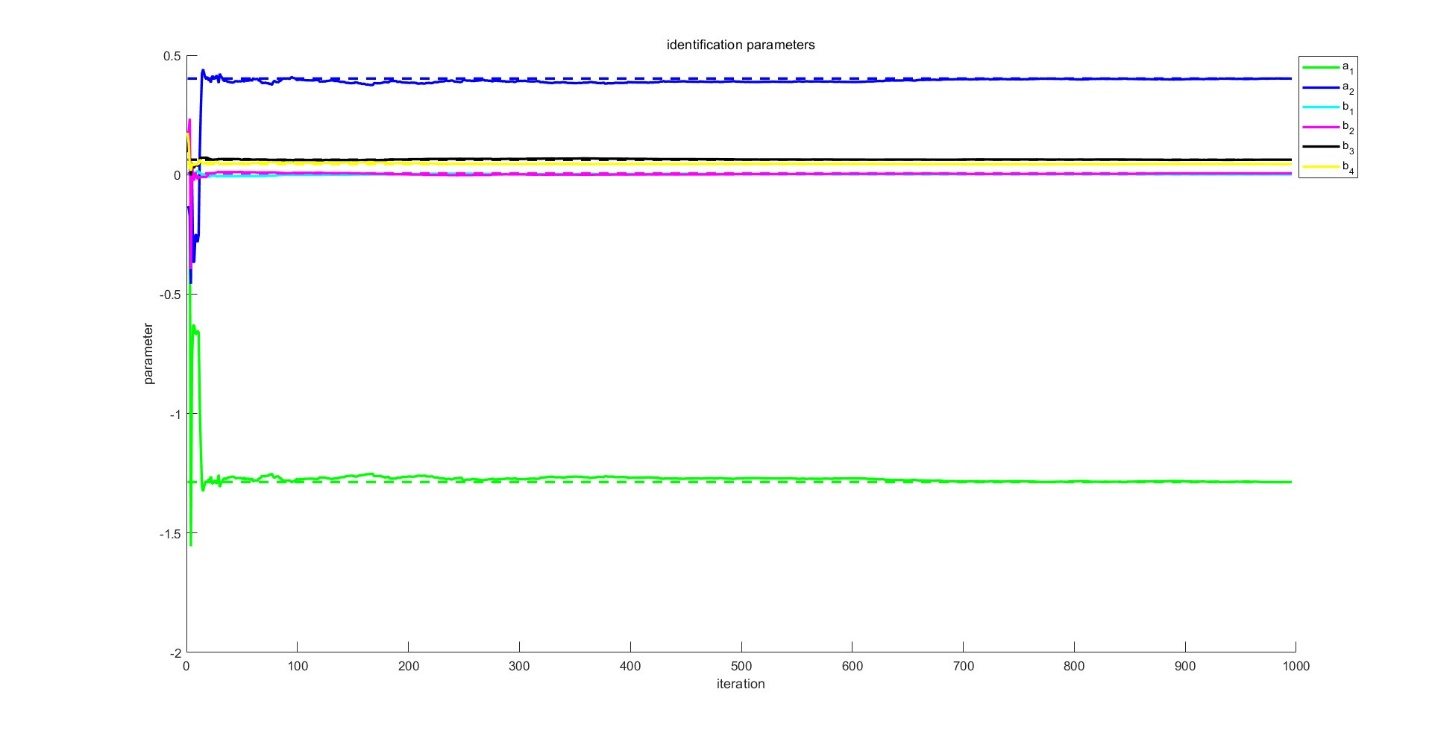


图 9 参数𝜃随迭代次数变化曲线

为进一步检验模型的有效性，绘制出输出真实值与估计值的比较图以及残差曲线，如图 10~图 11 所示。

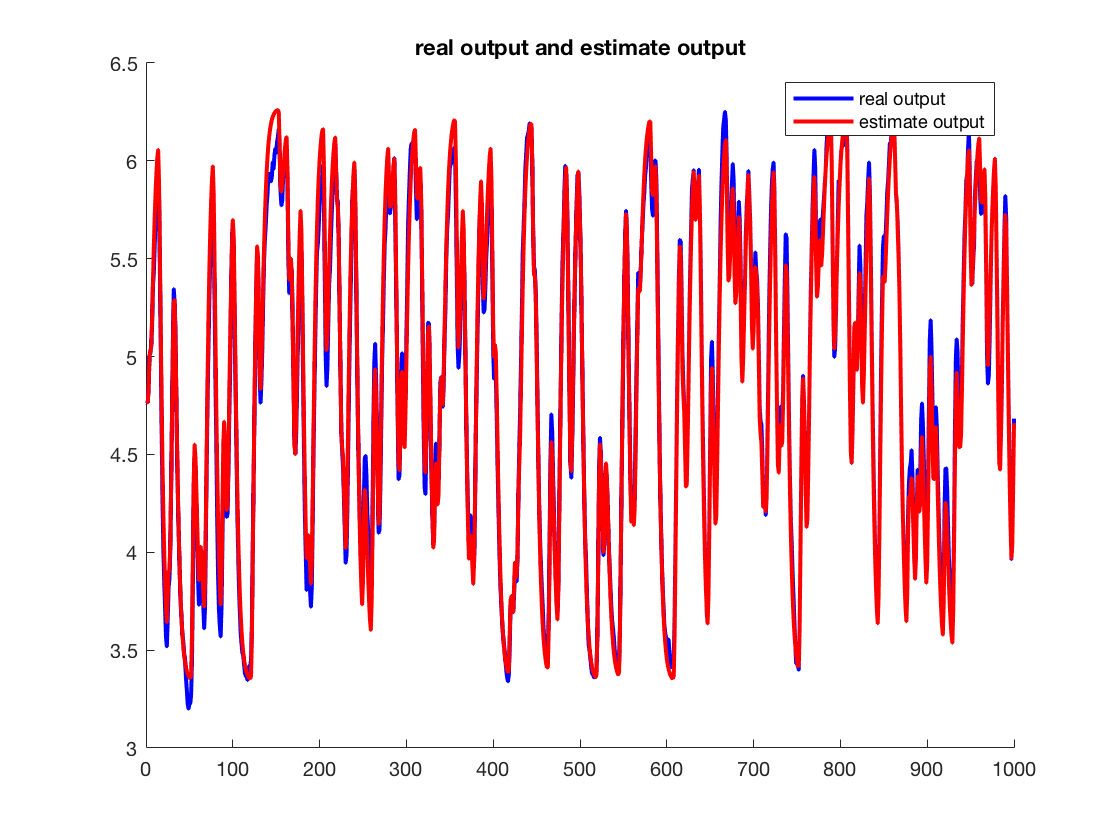


图 10 输出真实值与估计值比较图

由图 10 可以看出，其中红色曲线表示输出估计值，蓝色曲线表示输出实际值，估计值和实际值的相关性为 99.34%，说明模型能很好的估计系统的输出。

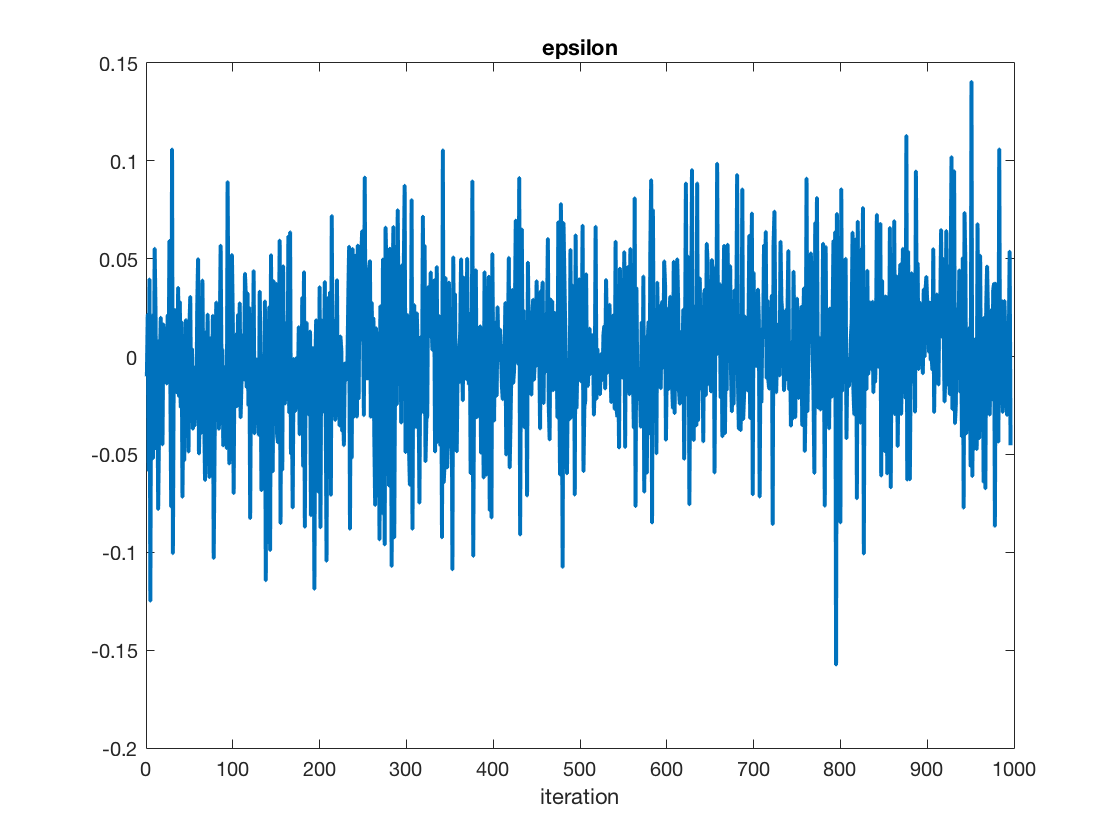


图 11 模型残差曲线

由图 11 可以看到，模型残差都是在 0 附近波动，也说明估计值与真实值十分接近，因此对 dryer 数据集使用 ARX 模型进行辨识得出的参数能很好地模型真实系统。

# 四、 递推极大似然估计法

## 1. **模型选择**

选取工程实际中常用的 ARXAX 模型，设其模型结构为：

其中，𝑣(𝑘)是均值为 0，方差为 1 的白噪声序列，

待辨识参数为：

## 2. **白噪声检验**

利用自相关系数检验法对模型残差序列进行白色性检验，如果能判断是零均值白噪声序列，则认为模型是“好”的。

根据 AIC 定阶法的结果，取计算残差序列的自相关系数为：

其中，是 的自相关函数，绘出自相关系数如图 8 所示。

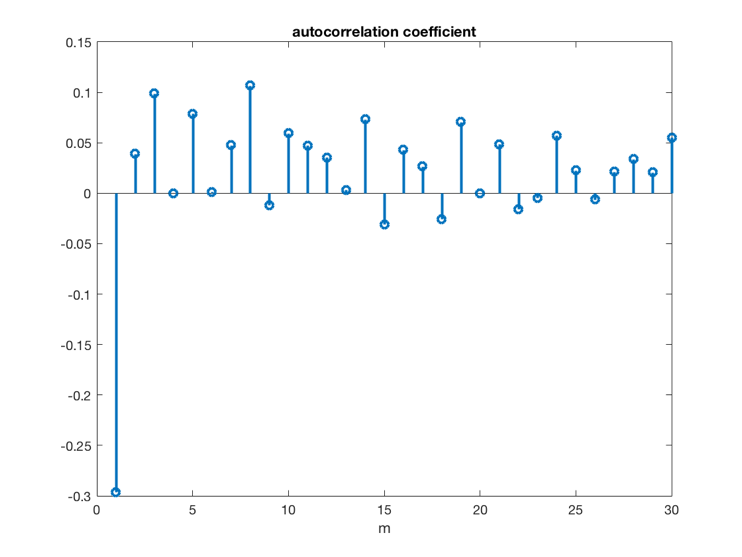


图 12 残差序列ε 𝑘 的自相关系数

取 m=25，利用下面两个不等式来判断残差序列的白色性：

1. （白色性阈值）
2. ，置信度取 0.05

经过实验，有

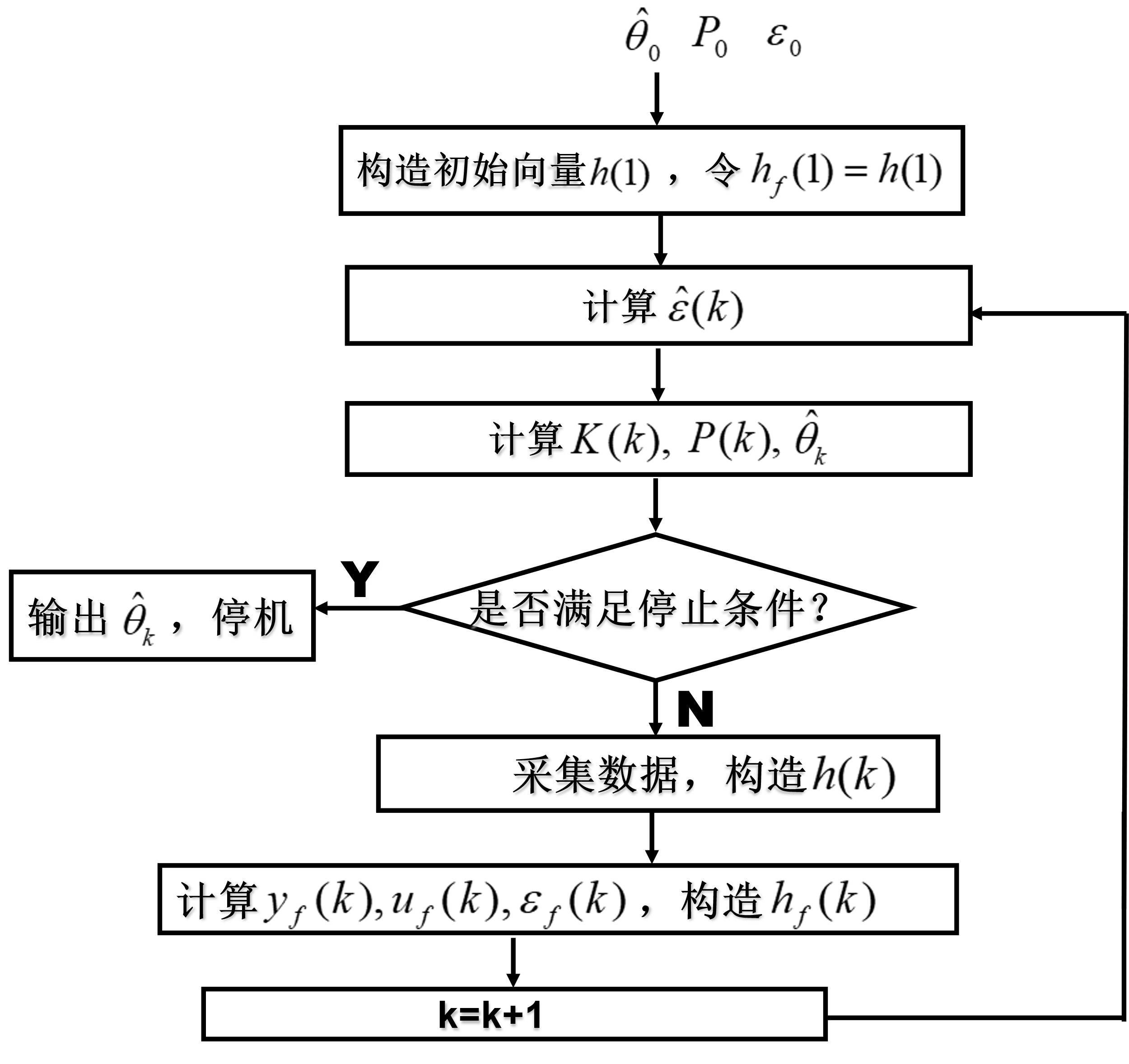
（满足条件）

（满足条件）

根据残差可知，

如果 ，当有一个不等式或两个不等式都成立时，则是白噪声序列，可信度为 95%。因此本实验中的残差序列满足以上条件，说明是白噪声序列，也进一步说明选取 ARX 模型可以很好地表示原系统。

## 3. 参数辨识



程序框图

采用递推最小二乘法对系统进行辨识，取根据 AIC 定阶法的结果，取，得到的辨识参数如下：

参数𝜃随迭代次数的变化曲线如图 9 所示。其中，实线代表迭代过程中参数实际值，虚线代表最终参数估计值，从图中可以看出，参数均很快收敛到一个稳定值，说明系统对最小二乘辨识是可收敛的，有效的。

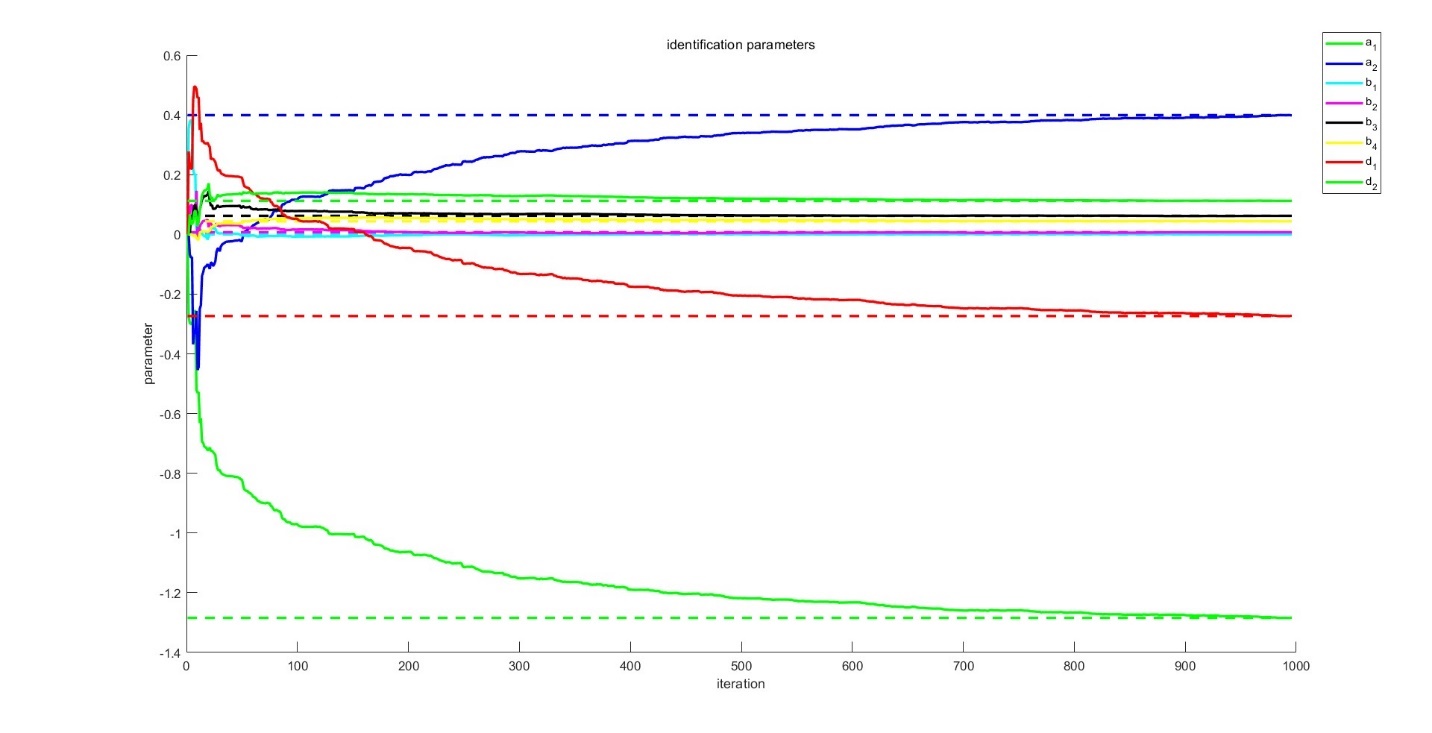


图 13 参数𝜃随迭代次数变化曲线

为进一步检验模型的有效性，绘制出输出真实值与估计值的比较图以及残差曲线，如图 10~图 11 所示。

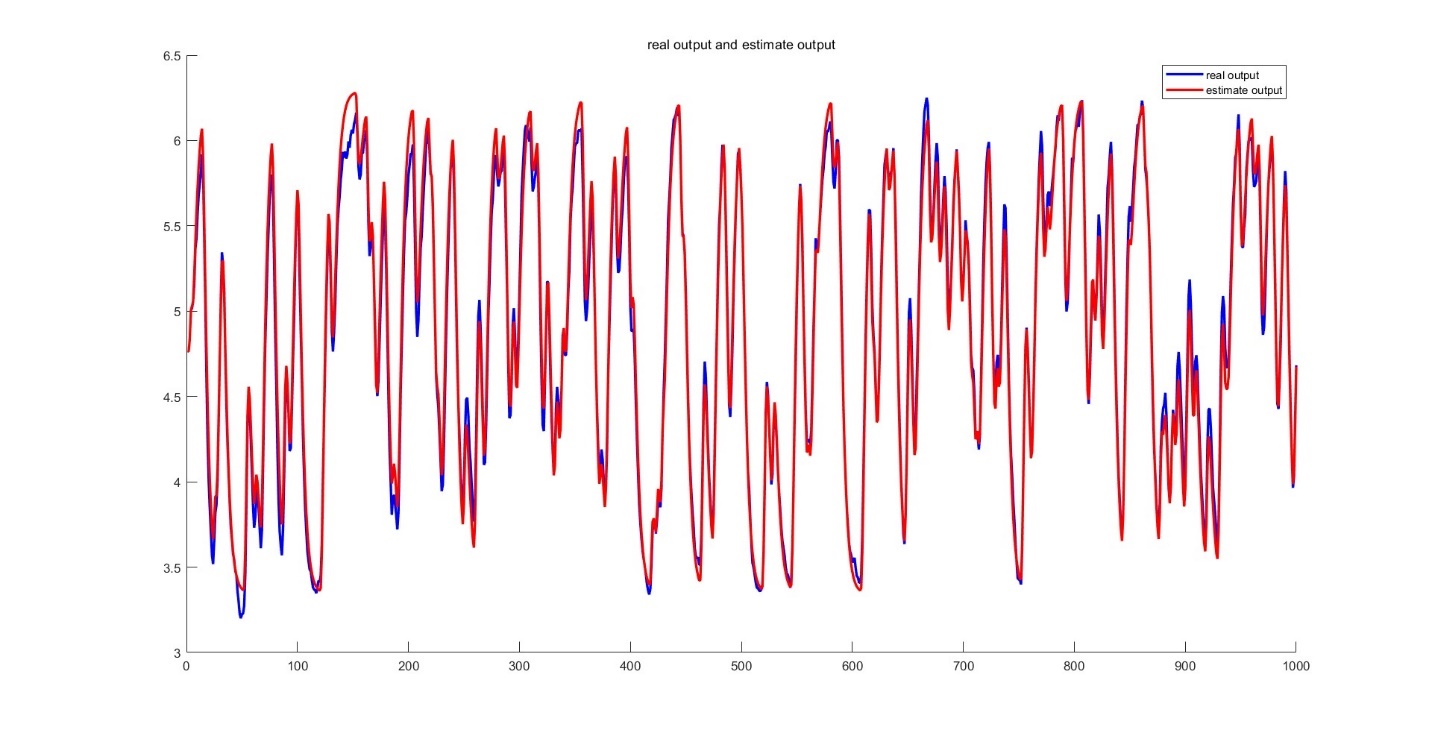


图 14 输出真实值与估计值比较图

由图 10 可以看出，其中红色曲线表示输出估计值，蓝色曲线表示输出实际值，估计值和实际值的相关性为 99.97%，说明模型能很好的估计系统的输出。

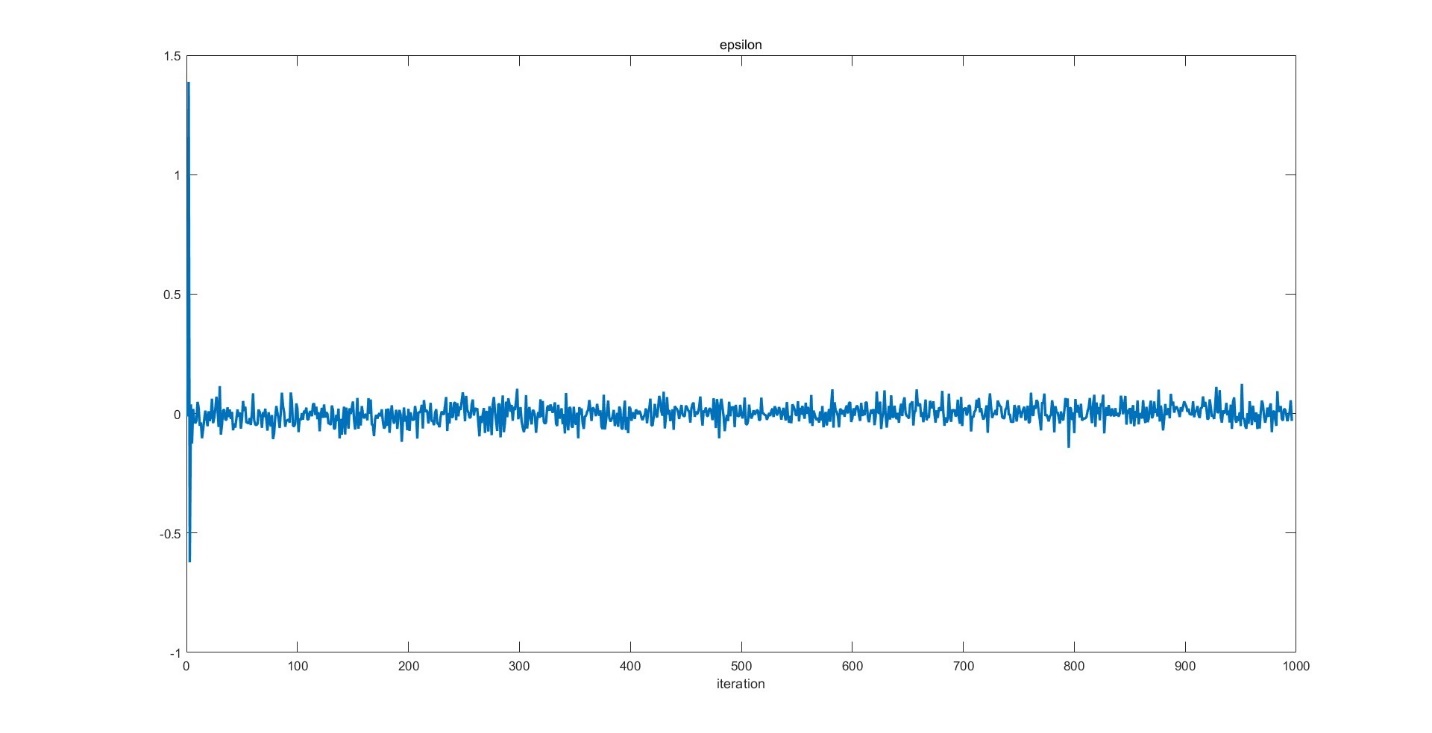


图 15 模型残差曲线

由图 11 可以看到，模型残差都是在 0 附近波动，也说明估计值与真实值十分接近，因此对 dryer 数据集使用 ARX 模型进行辨识得出的参数能很好地模型真实系统。

# 五、 三种辨识方法比较

（1） 从算法原理上：子空间辨识方法是基于离散时间线性状态空间模型提出的，直接估计系统状态空间模型；递推最小二乘法直接辨识系统参数，以优化损失函数为目标。

（2） 从运算时间上：子空间辨识方法由于不需要参数化；不需要迭代优化；算法实现仅依赖于一些简单可靠的线性代数工具，如 QR 分解、SVD 分解等，而递推最小二乘法需要进行迭代优化，因此子空间辨识法的运行速度快于递推最小二乘法。

（3） 从辨识结果来看： 表 3 两种方法辨识结果比较

|  |  |  |
| --- | --- | --- |
|  | 模型阶次 | 输出契合度 |
| 子空间辨识 | **3** | **88.45%** |
| 递推最小二乘法 | **6** | **99.34%** |
| 递推极大似然法 | **8** | **99.97%** |

由上表可以看出，三种辨识方法的辨识出的模型阶次有所差异，这可能是由于辨识方法本身的精度和原理不同造成的，但是相差不大；而在输出契合度上，子空间辨识<递推最小二乘法<递推极大似然法，子空间辨识更适用于多变量系统辨识，因此在此系统上的表现不如最小二乘法和递推极大似然法。而递推极大似然法的输出契合度要略好于递推最小二乘法，因为最小二乘法假设误差服从正态分布，因此在误差满足正态分布的情况下，最小二乘法通常能够提供较为准确的参数估计，在满足假设的情况下能够提供较好的估计结果，但它对异常值比较敏感。递推极大似然法则更适用于非正态分布的情况，它通过最大化观测数据的似然函数来估计参数，因此更加灵活，能够适应不同类型的数据分布和模型结构，在动态系统参数估计中也有较好的应用。 所以递推极大似然法的输出契合度要略好于递推最小二乘法。

# 参考文献

[1] Wouter Favoreel, Bart De Moor, Peter Van Overschee, Subspace state space system identification for industrial processes, Journal of Process Control, Volume 10, Issues 2–3, April 2000, Pages 149-155, ISSN 0959-152

[2] 李幼凤,苏宏业,褚健. 子空间模型辨识方法综述[J]. 化工学报,2006,(03):473-479.

[3] Mats Viberg, Subspace-based methods for the identification of linear time-invariant systems, Automatica, Volume 31, Issue 12, December 1995, Pages 1835-1851, ISSN 0005-1098

[4] Tohru Katayama, Giorgio Picci, Realization of stochastic systems with exogenous inputs and subspace identification methods, Automatica, Volume 35, Issue 10, October 1999, Pages 1635-1652, ISSN 0005-1098

[5] 萧德云. 系统辨识理论及应用[M]. 清华大学出版社, 2014.

[6] “Daisy, database for the identification of systems.” http://homes.esat.kuleuven.be/~smc/daisy/

# 附录

## ARX\_main.m

%% DaISy dataset identification

%递推最小二乘法参数辨识

clear;

close all;

%% initial

load('data.dat');

u=data(:,1);

z=data(:,2);

% %plot(z);

L=length(u);

na=2;

nb=4;

L=L-max(na,nb);

%% parameter identification

[theta, Inn, J, lambda,epsilon] = RLS(na, nb, z, u, L);

theta\_final=mean(theta(:,end-5:end),2);

disp('theta :');

disp(theta\_final);

%% estimate z

z\_est=estimate\_z(theta\_final,na,nb,u,z,L);

%% epsilon

epsilon=epsilon(1:L);

R=xcorr(epsilon, 'unbiased');

m=30;

rho=R(L+1:L+m)/R(L);

t1=L\*sum(rho.^2);

t2=m+1.65\*sqrt(2\*m);

t3=max(abs(rho));

t4=1.98/sqrt(L);

disp(['mean(epsilon)=',num2str(mean(epsilon))]);

disp(['L\*sum(rho.^2)=',num2str(t1),'<=m+1.65\*sqrt(2\*m)=',num2str(t2)]);

disp(['max(abs(rho))=',num2str(t3),'<=1.98\*sqrt(L)=',num2str(t4)]);

%% plot theta and rho

%plot theta

c = ['r', 'g', 'b', 'c', 'm', 'k', 'y'];

para\_num = size(theta, 1);

h = zeros(para\_num, 1);

str = cell(1, para\_num);

for i = 1:na

str{i} = ['a\_', num2str(i)];

end

for i = 1:nb

str{i + na} = ['b\_', num2str(i)];

end

figure;

hold on;

for i = 1:para\_num

h(i) = plot(1:L, theta(i, :), c(mod(i, length(c)) + 1));

plot([1, L], [theta\_final(i) theta\_final(i)], [c(mod(i, length(c)) + 1), '--']);

end

legend(h, str);

xlabel('iteration');ylabel('parameter');,title('identification parameters');

set(findobj(get(gca,'Children'),'LineWidth',0.5),'LineWidth',2);

%plot epsilon

figure;

plot(epsilon);

xlabel('iteration');title('epsilon');

set(findobj(get(gca,'Children'),'LineWidth',0.5),'LineWidth',2);

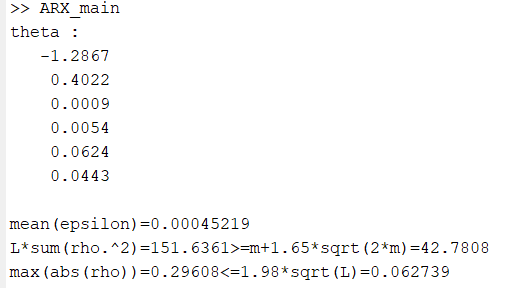
%plot Autocorrelation coefficient

figure;

stem(rho);

xlabel('m');title('autocorrelation coefficient');

set(findobj(get(gca,'Children'),'LineWidth',0.5),'LineWidth',2);



命令行输出结果

### RLS.m

%% RLS递推最小二乘法函数

function [theta, Inn, J, lambda,epsilon] = RLS(n\_a, n\_b, z, u, L)

nMax = max(n\_a, n\_b);

h = zeros(n\_a + n\_b, L + nMax);

s = zeros(L + nMax, 1);

Inn = zeros(L + nMax, 1); % new information

K = zeros(n\_a + n\_b, L + nMax); % gain

P = ones(n\_a + n\_b, n\_a + n\_b, L + nMax); % Covariance matrix

P(:, :, nMax) = eye(n\_a + n\_b) \* 1000;

theta = ones(n\_a + n\_b, L + nMax) \* 0.0001; % model parameter

J = zeros(L + nMax, 1); %loss function

for k = nMax + 1 : L + nMax

for i = 1:n\_a

h(i, k) = -z(k - i);

end

for i = 1:n\_b

h(n\_a + i, k) = u(k - i);

end

s(k) = h(:, k)' \* P(:, :, k-1) \* h(:, k) + 1.0;

Inn(k) = z(k) - h(:, k)' \* theta(:, k-1);

K(:, k) = P(:, :, k-1) \* h(:, k) / s(k);

P(:, :, k) = P(:, :, k-1) - K(:, k) \* K(:, k)' \* s(k);

theta(:, k) = theta(:, k-1) + K(:, k) \* Inn(k);

% count loss

J(k) = J(k-1) + Inn(k)^2 / s(k);

end

theta = theta(:, nMax+1 : end);

Inn = Inn(nMax+1 : end);

lambda = sqrt(J(L + nMax) / L);

epsilon = z(nMax+1:nMax+L) - h(:, nMax + 1 : nMax + L)' \* theta(:, end);

end

### estimate\_z.m

function z\_est=estimate\_z(theta,na,nb,u,z,L)

L=L+max(na,nb);

nMax = max(na, nb);

h = zeros(na + nb, 1);

z\_est=z;

for k = nMax+1 : L

for i = 1:na

h(i) = -z\_est(k - i);

end

for i = 1:nb

h(na + i) = u(k - i);

end

z\_est(k) = h' \* theta;

end

figure;

hold on;

plot(1:L,z,'b');

plot(1:L,z\_est,'r');

legend('real output','estimate output');

title('real output and estimate output');

set(findobj(get(gca,'Children'),'LineWidth',0.5),'LineWidth',2);

hold off;

end

## AIC\_main.m

%% DaISy dataset identification 绘制AIC热力图

% AIC to determine na and nb

clear;

close all;

%% initial

load('data.dat');

u=data(:,1);

z=data(:,2);

L=length(u);

%% AIC

K1=10;

K2=10;

K3=5;

%AIC=zeros(K1,K2,K3);

AIC=zeros(K1,K2);

for na=1:K1

for nb=1:K2

iter=L-max(na,nb);

[~,~,J,~,~]=RLS(na,nb,z,u,iter);

sigma\_v2=J(iter)/iter;

AIC(na,nb)=iter\*log(sigma\_v2)+2\*(na+nb);

end

end

plot\_AIC(AIC);

### plot\_AIC.m

function [f1,f2]=plot\_AIC(AIC)

f1=figure;

meshz(AIC);

title('AIC');

xlabel('nb');

ylabel('na');

set(gca,'xdir', 'reverse');

set(gca,'ydir', 'reverse');

set(gca,'xTick',1:10);

set(gca,'yTick',1:10);

set(findobj(get(gca,'Children'),'LineWidth',0.5),'LineWidth',2);

print(f1,'-dpng','AIC\_nb');

f2=figure;

meshz(AIC');

title('AIC');

xlabel('na');

ylabel('nb');

set(gca,'xdir', 'reverse');

set(gca,'ydir', 'reverse');

set(gca,'xTick',1:10);

set(gca,'yTick',1:10);

set(findobj(get(gca,'Children'),'LineWidth',0.5),'LineWidth',2);

print(f2,'-dpng','AIC\_na');

end

## ARX\_RML\_main.m

%% DaISy dataset identification

%递推极大似然估计

clear;

close all;

%% initial

load('data.dat');

u=data(:,1);

z=data(:,2);

% %plot(z);

L=length(u);

na=2;

nb=4;

L=L-max(na,nb);

%% parameter identification

[theta, Inn, J, lambda,epsilon] = RML(na, nb, z, u, L);

theta\_final=mean(theta(:,end-5:end),2);

disp('theta :');

disp(theta\_final);

%% estimate z

z\_est=estimate\_z(theta\_final,na,nb,u,z,L);

%% epsilon

epsilon=epsilon(1:L);

R=xcorr(epsilon, 'unbiased');

m=30;

rho=R(L+1:L+m)/R(L);

t1=L\*sum(rho.^2);

t2=m+1.65\*sqrt(2\*m);

t3=max(abs(rho));

t4=1.98/sqrt(L);

disp(['mean(epsilon)=',num2str(mean(epsilon))]);

disp(['L\*sum(rho.^2)=',num2str(t1),'<=m+1.65\*sqrt(2\*m)=',num2str(t2)]);

disp(['max(abs(rho))=',num2str(t3),'<=1.98\*sqrt(L)=',num2str(t4)]);

%% plot theta and rho

%plot theta

c = ['r', 'g', 'b', 'c', 'm', 'k', 'y',];

para\_num = size(theta, 1);

h = zeros(para\_num, 1);

str = cell(1, para\_num);

for i = 1:na

str{i} = ['a\_', num2str(i)];

end

for i = 1:nb

str{i + na} = ['b\_', num2str(i)];

end

for i = 1:2

str{na + nb + i} = ['d\_', num2str(i)];

end

figure;

hold on;

for i = 1:para\_num

h(i) = plot(1:L, theta(i, :), c(mod(i, length(c)) + 1));

plot([1, L], [theta\_final(i) theta\_final(i)], [c(mod(i, length(c)) + 1), '--']);

end

legend(h, str);

xlabel('iteration');ylabel('parameter');,title('identification parameters');

set(findobj(get(gca,'Children'),'LineWidth',0.5),'LineWidth',2);

%plot epsilon

figure;

plot(epsilon);

xlabel('iteration');title('epsilon');

set(findobj(get(gca,'Children'),'LineWidth',0.5),'LineWidth',2);

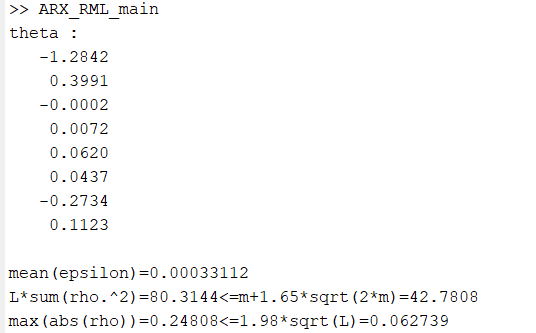
%plot Autocorrelation coefficient

figure;

stem(rho);

xlabel('m');title('autocorrelation coefficient');

set(findobj(get(gca,'Children'),'LineWidth',0.5),'LineWidth',2);



命令行输出结果

### RML.m

%% RML递推极大似然估计函数

function [theta, Inn, J, lambda,epsilon] = RML(n\_a, n\_b, z, u, L)

nd=2;

nMax = max(n\_a, n\_b);

h = zeros(n\_a + n\_b, L + nMax);

s = zeros(L + nMax, 1);

Inn = zeros(L + nMax, 1); % new information

K = zeros(n\_a + n\_b+nd, L + nMax); % gain

P = ones(n\_a + n\_b+ nd, n\_a + n\_b+ nd, L + nMax); % Covariance matrix

P(:, :, nMax) = eye(n\_a + n\_b+nd) \* 1000;

theta = ones(n\_a + n\_b+nd, L + nMax) \* 0.0001; % model parameter

J = zeros(L + nMax, 1); %loss function

uf = zeros(nMax, 1); % yf(k-i)输入

zf = zeros(nMax, 1); % zf(k-i)输出

v1 = zeros(nMax, 1);

v1f = zeros(nMax, 1);

for k = nMax + 1 : L + nMax

% 构造数据向量和滤波数据向量

for i = 1:n\_a

h(i, k) = -z(k - i);

hf(i, k) = -zf(k - i);

end

for i = 1:n\_b

h(n\_a + i, k) = u(k - i);

hf(n\_a + i, k) = uf(k - i);

end

for i = 1:nd

h(n\_a + n\_b + i, k) = v1(k - i);

hf(n\_a + n\_b + i, k) = v1f(k - i);

end

% 辨识算法

s(k) = hf(:, k)' \* P(:, :, k-1) \* hf(:, k) + 1.0;

Inn(k) = z(k) - h(:, k)' \* theta(:, k-1);

K(:, k) = P(:, :, k-1) \* hf(:, k) / s(k);

P(:, :, k) = P(:, :, k-1) - K(:, k) \* K(:, k)' \* s(k);

theta(:, k) = theta(:, k-1) + K(:, k) \* Inn(k);

% 损失函数

J(k) = J(k-1) + Inn(k)^2 / s(k);

% 噪声估计

v1(k) = z(k) - h(:, k)' \* theta(:, k);

% 输入、输出和噪声估计滤波值

zf(k) = z(k);

uf(k) = u(k);

v1f(k) = v1(k);

for i = 1:nd

zf(k) = zf(k) - theta(n\_a + n\_b + i, k) \* zf(k - i);

uf(k) = uf(k) - theta(n\_a + n\_b + i, k) \* uf(k - i);

v1f(k) = v1f(k) - theta(n\_a + n\_b + i, k) \* v1f(k - i);

end

end

theta = theta(:, nMax+1 : end);

Inn = Inn(nMax+1 : end);

lambda = sqrt(J(L + nMax) / L);

epsilon = z(nMax+1:nMax+L) - h(:, nMax + 1 : nMax + L)' \* theta(:, end);

end