文章编号:1674-7070(2011)06-0481-30

系统辨识(5):迭代搜索原理与辨识方法

丁锋[1,2,3]

摘要

递推辨识与迭代辨识构成了两类重要的参数估计方法. 递推辨识的递推变量与时间有关,因而可以用于在线估计系统参数;迭代辨识的迭代变量是自然数,与客观世界的时间无关,通常用于离线估计系统参数. 基于辅助模型辨识思想、多新息辨识理论、递阶辨识原理、耦合辨识概念等辨识方法都可以用递推算法和迭代算法实现. 迭代方法渊源很早,如求解矩阵方程 $\boldsymbol{Ax}=\boldsymbol{b}$ 的雅可比迭代、高斯-赛德尔迭代等. 迭代辨识方法主要使用梯度搜索、最小二乘搜索、牛顿搜索原理来实现. 为此主要研究了 CARMA 系统和 Box-Jenkins 系统的最小二乘迭代辨识方法与梯度迭代辨识方法. 这些方法也可推广到其他所有方程误差类系统和输出误差类系统,以及非线性系统. 迭代辨识方法通常用于有限量测数据的系统辨识,其收敛性证明是辨识领域极具挑战性的研究课题.

关键词

迭代辨识;递推辨识;参数估计;FIR 模型;CAR 模型;CARMA 模型;CARAR 模型;CARARMA 模型;输出误差模型;OEMA 模型;OEAR 模型;辅助模型辨识;多新息辨识;递阶辨识;耦合辨识

中图分类号 TP273

文献标志码 A

收稿日期 2011-07-18

资助项目 国家自然科学基金(60973043)

作者简介

丁锋,男,博士,教授,博士生导师,主要从事系统辨识、过程建模、自适应控制方面的研究. fding@ jiangnan. edu. cn

1 江南大学 物联网工程学院,无锡,214122

2 江南大学 控制科学与工程研究中心,无锡,214122

3 江南大学 教育部轻工过程先进控制重点实验室,无锡,214122

0 引言

"Apple 的创始人史蒂夫·乔布斯(Steve Jobs)的才华、激情和精力是无尽创新的来源,丰富和改善了我们的生活,世界因他而无限美好." 这是对乔布斯一生成就的高度总结,也是对当今高度发展的自动化电子产品和应用软件的创造者和开发者的高度歌颂[1-5]. 创新是这个时代的标识. 创新需要才华,创新需要激情,创新需要敏锐的洞察力,创新需要坚强的毅力. 创新是技术的进步,创新是科学的源泉,创新是科学发展的动力. 新思想、新理论、新原理、新概念的诞生都是科学史上的重要里程碑. 就研究建立系统数学模型的理论与方法的系统辨识而言,辅助模型辨识思想、多新息辨识理论、递阶辨识原理、耦合辨识概念的诞生,对推动系统辨识学科的研究进程有重要作用.

科学就是解释自然现象,探索未知世界,揭示事物的运动规律. 客观事物具有多样性,具有简单性和复杂性两种特征. 事物的一些表层特征可以通过观察得到,这是事物简单性的一个方面. 事物的本质特征是事物复杂性的一个方面,需要通过对观测信息(数据)的处理,加工、抽象、推断获得.

系统的外部信息(即系统的输入输出数据)通常是可以量测得到的,而系统本质特征(即数学模型)是通过输入输出数据反映出来的. 对于一些简单问题,可以通过求解代数方程获得系统的数学模型,然而,对于一些复杂系统,甚至模型方程中还包含了系统的不可测干扰噪声,即使是线性系统,也难以用拉普拉斯变换(Laplace transform)方法,即通过输出拉普拉斯变换与输入信号拉普拉斯变换之比求得系统的传递函数(数学模型).

很多辨识方法可以利用系统的输入输出数据来建立系统的数学模型,如递推最小二乘参数辨识方法. 最小二乘方法适合线性参数估计问题,然而,客观世界更多的是非线性问题,量测数据包含噪声干扰,给建立描述事物运动规律的数学模型带来特别的困难. 在这种情况下,如何利用系统的观测信息,利用梯度搜索及牛顿搜索原理,通过迭代技术来建立系统的数学模型是摆在控制科学家面前的重要任务. 这也是迭代辨识思想产生的根源.

与递推辨识类似,迭代辨识也是一大类辨识方法. 迭代方法渊源很早,本文作者首次将它系统地用于系统辨识的研究. 迭代辨识方法一般用于辨识模型信息向量中含有未知项的系统辨识. 因此,迭代辨

识思想适合于所有方程误差类模型和输出误差类模型,即线性回归模型、伪线性回归模型,以及非线性系统辨识方法的研究.

迭代辨识的基本思想是采用交互估计理论和递阶辨识原理,利用批数据来刷新参数估计,信息向量中的未知项用前一步迭代参数估计进行估算,然后用估计的未知项代替信息向量中的真实未知项,参数估计利用代替后的信息向量进行刷新. 二者执行了一个递阶计算过程.

一般来说,递推算法(recursive algorithm)用于在线辨识(on-line identification),而迭代算法(iterative algorithm)用于离线辨识(off-line identification). 为了区别在线辨识和离线辨识,本文用带下标 k 的参数向量 $\hat{\boldsymbol{\theta}}_k$ 表示迭代算法给出的参数估计,用不带下标的参数向量 $\hat{\boldsymbol{\theta}}(t)$ 表示递推算法给出的参数估计[6-7]. 这里 k 是一个与时间无关的迭代变量,t 是一个时间变量. 注意,利用动态有限数据窗内的数据的迭代算法,给出的参数估计 $\hat{\boldsymbol{\theta}}_k(t)$ 也可用于在线辨识[8]. 重要的是,递推算法参数估计中的 t 代表时间或与时间有关的量,而迭代算法参数估计中的迭代变量 k 与时间无关.

本文作者提出的多变量系统递阶梯度迭代算法和递阶随机梯度算法、递阶最小二乘迭代算法和递阶最小二乘算法论文分别发表在《Automatica》2005 年第 2 期[9]和《IEEE Transactions on Automatic Control》2005 年第 3 期[10],随后 Hammerstein 非线性 ARMAX 系统的最小二乘迭代辨识与递推增广最小二乘辨识 Regular Paper 论文发表在《Automatica》2005 年第 9 期[6]上,继而又提出了相应的梯度迭代算法与增广随机梯度算法[7]. 最近,本文作者等详细研究了输出误差模型(OE,Output Error model)和输出误差滑动平均模型(OEMA,Output Error Moving Average model)的梯度迭代算法和最小二乘迭代算法[11],提出了 Hammerstein 非线性系统的投影算法、随机梯度算法、牛顿递推算法和牛顿迭代算法[12]. 王冬青等[13]研究了 Wiener 非线性 ARMAX 系统的最小二乘迭代算法和梯度迭代算法.

本文首先介绍用于线性问题的最小二乘原理、用于非线性问题的梯度搜索原理和牛顿迭代方法;然后研究受控自回归滑动平均(CARMA)系统和 Box-Jenkins 系统的梯度迭代辨识方法和最小二乘迭代辨识方法;最后介绍一类维纳(Wiener)系统的迭代辨识方法.

本文较长,为便于阅读,特将本文框架结构列示如下.

1 最小二乘原理与迭代搜索原理

先引入符号:"$A =: X$" 或"$X := A$" 表示"A 记作(定义为) X" 之意(因为符号 $\triangleq$ 没有左右之分,含义模糊).

最小二乘原理和迭代搜索原理是研究系统参数估计算法的基本工具.

1.1 最小二乘原理

1.1.1 一维例子

假设测量某个物体长度,n 个人测得的长度分别为 $x_1, x_2, \cdots, x_n$,那么该物体长度 x 最可能是多少?按照统计学原理,定义误差平方和:

$$f(x) = (x_1 - x)^2 + (x_2 - x)^2 + \cdots + (x_n - x)^2.$$

令 $f(x)$ 关于 x 的导数为零,得到

$$f'(x) = -2(x_1 - x) - 2(x_2 - x) - \cdots - 2(x_n - x) = 0.$$

求解得到

$$x = \frac{x_1 + x_2 + \cdots + x_n}{n}.$$

且 $f''(x) = 2n > 0$,可知 $f\left(\frac{x_1 + x_2 + \cdots + x_n}{n}\right)$ 是误差函数 $f(x)$ 的最小值. 这就是最小二乘(least squares)的基本原理. "二乘"可理解为"平方"的意思,即误差平方和.

当然,也可以定义误差的绝对值和:

$$f_1(x) = |x_1 - x| + |x_2 - x| + \cdots + |x_n - x|,$$

或误差的 4 次方(6 次方,8 次方,…)和:

$$f_2(x) = (x_1 - x)^4 + (x_2 - x)^4 + \cdots + (x_n - x)^4,$$

求它们的极值,但这些函数都比求二次函数 $f(x)$ 的极值更复杂,这就是最小二乘得到广泛应用的原因. 最小二乘在系统辨识、线性拟合、非线性拟合等各个领域都有广泛应用.

1.1.2 二维例子

图1中有很多"点":$(x_1,y_1),(x_2,y_2),\cdots,(x_n,y_n)$,它们都很接近某条直线. 假设这条待定的直线为

$$y = kx + b. \tag{1}$$

点 $(x_1,y_1),(x_2,y_2),\cdots,(x_n,y_n)$,不可能都使上述方程成立. 因此,目标是确定参数 k 和 b 使得每个点到该直线"距离"(不是垂直距离,当然可以考虑垂直距离. 为使求解简单,这里采用的是纵坐标差)

$$\varepsilon_i := y_i - kx_i - b \tag{2}$$

的平方和

$$J_1(k,b) = \sum_{j=1}^{n} \varepsilon_j^2 = (y_1 - kx_1 - b)^2 + (y_2 - kx_2 - b)^2 + \cdots + (y_n - kx_n - b)^2 \tag{3}$$

最小. 这就是最小二乘拟合(least squares fitting)问题.

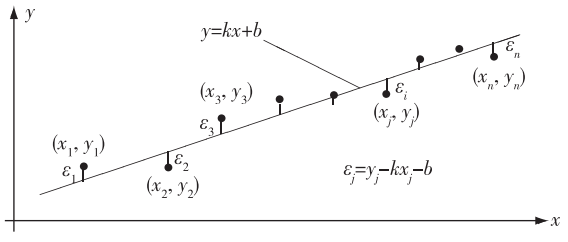

图1 离散点线性拟合示意

Fig. 1 The linear fitting of discrete dots

分别对 k 和 b 求偏导,并令其为零得到

$$\frac{\partial J_1(k,b)}{\partial k} = -2x_1(y_1 - kx_1 - b) - 2x_2(y_2 - kx_2 - b) - \cdots - 2x_n(y_n - kx_n - b) = 0,$$

$$\frac{\partial J_1(k,b)}{\partial b} = -2(y_1 - kx_1 - b) - 2(y_2 - kx_2 - b) - \cdots - 2(y_n - kx_n - b) = 0.$$

或

$$\left(\sum_{j=1}^{n} x_j^2\right)k + \left(\sum_{j=1}^{n} x_j\right)b = \sum_{j=1}^{n} x_j y_j,$$

$$\left(\sum_{j=1}^{n} x_j\right)k + nb = \sum_{j=1}^{n} y_j.$$

因此,求得的参数拟合值为

$$\begin{bmatrix} k \\ b \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{n} x_j^2 & \sum_{j=1}^{n} x_j \\ \sum_{j=1}^{n} x_j & n \end{bmatrix}^{-1} \begin{bmatrix} \sum_{j=1}^{n} x_j y_j \\ \sum_{j=1}^{n} y_j \end{bmatrix}.$$

式(2)中 ε_i 可理解为点 (x_j,y_j) 偏离直线 $y = kx + b$ 的"误差". 如果所有误差 $\varepsilon_j(j=1,2,\cdots,n)$ 都为零,那么这些点就都在一条直线上. 因此,极小化式(3)中误差平方和准则函数 J_1,就是得到一条离所有点最接近的直线.

为了处理高维(大于2个未知参数)线性拟合情形,定义由观测(数据)构成的信息向量 $\boldsymbol{\varphi}_j$ 和拟合模型(1)的参数向量 $\boldsymbol{\theta}$ 如下:

$$\boldsymbol{\varphi}_j = \begin{bmatrix} x_j \\ 1 \end{bmatrix}, \quad \boldsymbol{\theta} = \begin{bmatrix} k \\ b \end{bmatrix}. \tag{5}$$

那么由式(2)可得

$$y_j = kx_j + b + \varepsilon_j = [x_j, 1]\begin{bmatrix} k \\ b \end{bmatrix} + \varepsilon_j = \boldsymbol{\varphi}_j^{\mathrm{T}}\boldsymbol{\theta} + \varepsilon_j. \tag{6}$$

式中上标T表示向量或矩阵转置. 式(6)称为线性回归模型(在系统辨识中称为辨识模型或辨识表达式),因为观测 y_j 是参数向量 $\boldsymbol{\theta}$ 的线性函数. 偏差 ε_j 简称为噪声或随机干扰(对存在随机测量误差的随机系统而言),目标是使误差准则函数值 J_1 最小,也就是直线上下点的距离和相等,即上面的正偏差和与下面的负偏差和相等,所以有

$$\frac{\varepsilon_1 + \varepsilon_2 + \cdots + \varepsilon_n}{n} = 0.$$

这可理解为数学上的平均值或概率论中的"样本均值"为零,而准则函数除以 n,即 $\frac{J_1}{n}$ 可理解为噪声 ε_j 的样本方差. J_1 达到最小,即噪声方差最小. 因此,从统计意义上讲,拟合得到的最优参数向量 $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$ 称为 $\boldsymbol{\theta}$ 的估计,此时的残差

$$\varepsilon_j(\hat{\boldsymbol{\theta}}) = y_j - \boldsymbol{\varphi}_j^{\mathrm{T}}\hat{\boldsymbol{\theta}}$$

就是不相关零均值白噪声,且方差最小(对于足够大的 n).

下面求解向量形式的参数估计. 参照式(6),准则函数(3)可简写为

$$J_2(\boldsymbol{\theta}) = \sum_{j=1}^{n} \varepsilon_j^2 = \sum_{j=1}^{n} (y_j - \boldsymbol{\varphi}_j^{\mathrm{T}}\boldsymbol{\theta})^2. \tag{7}$$

对 $\boldsymbol{\theta}$ 求偏导(本文定义标量对列向量的偏导为列向量),并令其为零可得

$$\frac{\partial J_2(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -2\sum_{j=1}^{n} \boldsymbol{\varphi}_j(y_j - \boldsymbol{\varphi}_j^{\mathrm{T}}\boldsymbol{\theta}) =$$

$$-2\Big[\sum_{j=1}^{n}\boldsymbol{\varphi}_j y_j-\Big(\sum_{j=1}^{n}\boldsymbol{\varphi}_j\boldsymbol{\varphi}_j^{\mathrm{T}}\Big)\boldsymbol{\theta}\Big]=0.$$

或

$$\sum_{j=1}^{n}\boldsymbol{\varphi}_j y_j-\Big(\sum_{j=1}^{n}\boldsymbol{\varphi}_j\boldsymbol{\varphi}_j^{\mathrm{T}}\Big)\boldsymbol{\theta}=0.$$

故 $\boldsymbol{\theta}$ 的最小二乘估计为

$$\hat{\boldsymbol{\theta}}=\Big(\sum_{j=1}^{n}\boldsymbol{\varphi}_j\boldsymbol{\varphi}_j^{\mathrm{T}}\Big)^{-1}\sum_{j=1}^{n}\boldsymbol{\varphi}_j y_j. \tag{8}$$

将 $\boldsymbol{\varphi}_j$ 的定义式(5)代入式(8),即得式(4).

定义

$$\boldsymbol{Y}_n:=\begin{bmatrix}y_1\\y_2\\\vdots\\y_n\end{bmatrix}\in\mathbf{R}^n,\quad \boldsymbol{H}_n:=\begin{bmatrix}\boldsymbol{\varphi}_1^{\mathrm{T}}\\\boldsymbol{\varphi}_2^{\mathrm{T}}\\\vdots\\\boldsymbol{\varphi}_n^{\mathrm{T}}\end{bmatrix}\in\mathbf{R}^{n\times 2},$$

$$\boldsymbol{\varepsilon}_n:=\begin{bmatrix}\varepsilon_1\\\varepsilon_2\\\vdots\\\varepsilon_n\end{bmatrix}\in\mathbf{R}^n.$$

因为

$$\boldsymbol{H}_n^{\mathrm{T}}\boldsymbol{H}_n=[\boldsymbol{\varphi}_1,\boldsymbol{\varphi}_2,\cdots,\boldsymbol{\varphi}_n]\begin{bmatrix}\boldsymbol{\varphi}_1^{\mathrm{T}}\\\boldsymbol{\varphi}_2^{\mathrm{T}}\\\vdots\\\boldsymbol{\varphi}_n^{\mathrm{T}}\end{bmatrix}=\sum_{j=1}^{n}\boldsymbol{\varphi}_j\boldsymbol{\varphi}_j^{\mathrm{T}},$$

$$\boldsymbol{H}_n^{\mathrm{T}}\boldsymbol{Y}_n=[\boldsymbol{\varphi}_1,\boldsymbol{\varphi}_2,\cdots,\boldsymbol{\varphi}_n]\begin{bmatrix}y_1\\y_2\\\vdots\\y_n\end{bmatrix}=\sum_{j=1}^{n}\boldsymbol{\varphi}_j y_j.$$

那么式(8)的最小二乘估计可以表示为

$$\hat{\boldsymbol{\theta}}=(\boldsymbol{H}_n^{\mathrm{T}}\boldsymbol{H}_n)^{-1}\boldsymbol{H}_n^{\mathrm{T}}\boldsymbol{Y}_n. \tag{9}$$

由式(6)可知,

$$\varepsilon_j=y_j-\boldsymbol{\varphi}_j^{\mathrm{T}}\boldsymbol{\theta}.$$

当 $j=1,2,\cdots,n$ 时,共得 n 个方程,

$$\begin{cases}\varepsilon_1=y_1-\boldsymbol{\varphi}_1^{\mathrm{T}}\boldsymbol{\theta},\\\varepsilon_2=y_2-\boldsymbol{\varphi}_2^{\mathrm{T}}\boldsymbol{\theta},\\\quad\vdots\\\varepsilon_n=y_n-\boldsymbol{\varphi}_n^{\mathrm{T}}\boldsymbol{\theta}.\end{cases}$$

把它们写成向量形式为

$$\begin{bmatrix}\varepsilon_1\\\varepsilon_2\\\vdots\\\varepsilon_n\end{bmatrix}=\begin{bmatrix}y_1\\y_2\\\vdots\\y_n\end{bmatrix}-\begin{bmatrix}\boldsymbol{\varphi}_1^{\mathrm{T}}\\\boldsymbol{\varphi}_2^{\mathrm{T}}\\\vdots\\\boldsymbol{\varphi}_n^{\mathrm{T}}\end{bmatrix}\boldsymbol{\theta}.$$

即得误差向量方程

$$\boldsymbol{\varepsilon}_n=\boldsymbol{Y}_n-\boldsymbol{H}_n\boldsymbol{\theta}. \tag{10}$$

令 $\|\boldsymbol{X}\|^2:=\mathrm{tr}[\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}]$. 准则函数 J_2 可以写成向量乘积形式:

$$J_2(\boldsymbol{\theta})=\sum_{j=1}^{n}\varepsilon_j^2=\boldsymbol{\varepsilon}_n^{\mathrm{T}}\boldsymbol{\varepsilon}_n=(\boldsymbol{Y}_n-\boldsymbol{H}_n\boldsymbol{\theta})^{\mathrm{T}}(\boldsymbol{Y}_n-\boldsymbol{H}_n\boldsymbol{\theta})=\|\boldsymbol{Y}_n-\boldsymbol{H}_n\boldsymbol{\theta}\|^2.$$

将式(9)参数估计代入上式,可得数据长度为 n 时的准则函数值:

$$J_2(\hat{\boldsymbol{\theta}})=(\boldsymbol{Y}_n-\boldsymbol{H}_n\hat{\boldsymbol{\theta}})^{\mathrm{T}}(\boldsymbol{Y}_n-\boldsymbol{H}_n\hat{\boldsymbol{\theta}})=$$
$$[\boldsymbol{Y}_n-\boldsymbol{H}_n(\boldsymbol{H}_n^{\mathrm{T}}\boldsymbol{H}_n)^{-1}\boldsymbol{H}_n^{\mathrm{T}}\boldsymbol{Y}_n]^{\mathrm{T}}[\boldsymbol{Y}_n-\boldsymbol{H}_n(\boldsymbol{H}_n^{\mathrm{T}}\boldsymbol{H}_n)^{-1}\boldsymbol{H}_n^{\mathrm{T}}\boldsymbol{Y}_n]=$$
$$\boldsymbol{Y}_n^{\mathrm{T}}[\boldsymbol{I}_n-\boldsymbol{H}_n(\boldsymbol{H}_n^{\mathrm{T}}\boldsymbol{H}_n)^{-1}\boldsymbol{H}_n^{\mathrm{T}}]^{\mathrm{T}}[\boldsymbol{I}_n-\boldsymbol{H}_n(\boldsymbol{H}_n^{\mathrm{T}}\boldsymbol{H}_n)^{-1}\boldsymbol{H}_n^{\mathrm{T}}]\boldsymbol{Y}_n=$$
$$\boldsymbol{Y}_n^{\mathrm{T}}[\boldsymbol{I}_n-\boldsymbol{H}_n(\boldsymbol{H}_n^{\mathrm{T}}\boldsymbol{H}_n)^{-1}\boldsymbol{H}_n^{\mathrm{T}}]^2\boldsymbol{Y}_n=$$
$$\boldsymbol{Y}_n^{\mathrm{T}}[\boldsymbol{I}_n-\boldsymbol{H}_n(\boldsymbol{H}_n^{\mathrm{T}}\boldsymbol{H}_n)^{-1}\boldsymbol{H}_n^{\mathrm{T}}]\boldsymbol{Y}_n.$$

1.2 梯度搜索原理

梯度迭代或最速下降搜索方法,或梯度法是十分基本而又最为古老的一种搜索方法,它的迭代过程简单,使用方便,而且又是理解其他一些迭代方法的基础[18]. 下面先介绍一个简单迭代算法.

1.2.1 一个简单迭代算法

对于方程

$$5x=20,$$

其解为 $x=\dfrac{20}{5}=4$. 但也可以进行迭代求解,把方程变换为

$$(5-a)x=20-ax,\quad a>0.$$

两边除以(5 − a)得到

$$x=\frac{20-ax}{5-a}.$$

采用迭代方法求解:把方程左边的 x 换为 x_k,右边的 x 换为 x_{k-1},可得递归方程

$$x_k=\frac{20-ax_{k-1}}{5-a},\quad k=1,2,3,\cdots. \tag{11}$$

取初值 $x_0=1$,当 $a=0.1$ 和 $a=0.5$ 时,x 的迭代解 x_k 如表1所示;当 $a=1.0$ 和 $a=2.2$ 时,x 的迭代解 x_k 如表2所示;迭代误差 $\delta:=\dfrac{|x_k-x|}{|x|}\times 100\%$ 随 k 变化情况如图2所示.

从表1、2可以看出:随着 a 值增大,迭代解精度变低,迭代解误差 δ 变大;随着 a 值减小,迭代解收敛于真解 $x=4$ 的速度变快. 当 $a\geqslant 2.5$ 时,迭代解 x_k 不收敛,原因很简单,因为迭代解(11)对应的离散系统特征值在单位圆上或单位圆外,系统不稳定,迭代解发散,不收敛于真解.

表 1　x 的迭代解 x_k($a=0.1$ 和 $a=0.5$)

Table 1　Iterative solutions($a=0.1$ and $a=0.5$)

$a=0.1$			$a=0.5$		
迭代次数 k	迭代解 x_k	误差 δ/%	迭代次数 k	迭代解 x_k	误差 δ/%
0	1.000 000 00	75.000 000 00	0	1.000 000 00	75.000 000 00
1	4.061 224 49	1.530 612 24	1	4.333 333 33	8.333 333 33
2	3.998 750 52	0.031 236 98	2	3.962 962 96	0.925 925 93
3	4.000 025 50	0.000 637 49	3	4.004 115 23	0.102 880 66
4	3.999 999 48	0.000 013 01	4	3.999 542 75	0.011 431 18
5	4.000 000 01	0.000 000 27	5	4.000 050 81	0.001 270 13
6	4.000 000 00	0.000 000 01	6	3.999 994 35	0.000 141 13
7	4.000 000 00	0.000 000 00	7	4.000 000 63	0.000 015 68
8	4.000 000 00	0.000 000 00	8	3.999 999 93	0.000 001 74
9	4.000 000 00	0.000 000 00	9	4.000 000 01	0.000 000 19
10	4.000 000 00	0.000 000 00	10	4.000 000 00	0.000 000 02
真解	4.000 000 00		真解	4.000 000 00	

表 2　x 的迭代解 x_k($a=1.0$ 和 $a=2.2$)

Table 2　Iterative solutions($a=1.0$ and $a=2.2$)

$a=1.0$			$a=2.2$		
迭代次数 k	迭代解 x_k	误差 δ/%	迭代次数 k	迭代解 x_k	误差 δ/%
0	1.000 000 00	75.000 000 00	0	1.000 000 00	75.000 000 00
1	4.750 000 00	18.750 000 00	1	6.357 142 86	58.928 571 43
2	3.812 500 00	4.687 500 00	2	2.147 959 18	46.301 020 41
3	4.046 875 00	1.171 875 00	3	5.455 174 93	36.379 373 18
4	3.988 281 25	0.292 968 75	4	2.856 648 27	28.583 793 21
5	4.002 929 69	0.073 242 19	5	4.898 347 79	22.458 694 67
6	3.999 267 58	0.018 310 55	6	3.294 155 31	17.646 117 24
7	4.000 183 11	0.004 577 64	7	4.554 592 26	13.864 806 40
8	3.999 954 22	0.001 144 41	8	3.564 248 94	10.893 776 46
9	4.000 011 44	0.000 286 10	9	4.342 375 83	8.559 395 79
10	3.999 997 14	0.000 071 53	10	3.730 990 42	6.725 239 55
真解	4.000 000 00		真解	4.000 000 00	

Matlab 程序如下.

把下列程序写到 Iterative01.m 文件中,依次运行 $a=0.1,0.5,1.0,2.2$,可得到表 1、2 的迭代解和误差.

```
1  %------------------------------------------------------------*
2  % Filename:Iterative01.m for the linear equation: *
3  % 5x=20, *
4  % The iterative algorithm: *
5  % x(k)=[20-a*x(k-1)]/(5-a),x(0)=1 *
6  % a=0.1,0.5,1.0 and 2.2 *
7  %------------------------------------------------------------*
8  clear;format short g
9  M='The iterative algorithm for 5x=20'
10 % 5x=20-->(5-a)x=20-ax-->x=(20-ax)/(5-a)
11 x0=20/5;% The true solution
12 a=2.2;% a=0.1,0.5,1.0 and 2.2
13 x=1;% The inital value x=x(0)=1;1,8
14 delta=abs(x-x0)/x0*100;% The error
15 ls=[0,x,delta];
16 for k=1:20
17     x=(20-a*x)/(5-a);% x_k
18     delta=abs(x-x0)/x0*100;% The error
19     ls=[ls;k,x,delta];
20 end
21 fprintf('a=%5.2f\n %s\n','a,'k x(k) \delta\(\%)\\\
```

```
    \\hline')
22  fprintf('%5d %12.8f %12.8f\\\\\n',ls');
23  fprintf('True solustion %12.8f %s\n',x0,'\\\hline');
24  if a==1
25      data1=[ls(:,1),ls(:,3)/100]
26      save data1 data1
27  else % a==2.2
28      load data1
29      z0=[data1,ls(:,3)/100];
30      figure(1);jk=z0(:,1);
31      plot(jk,z0(:,2),'k',jk,z0(:,2),'k.',...
32          jk,z0(:,3),'b',jk,z0(:,3),'k.');% The error
            curve
33      text(6,0.215,'\ita}=2.2')
34      text(2,0.1,'\ita}=1.0')
35  end
36  xlabel('\it k');ylabel('\it \delta');
```

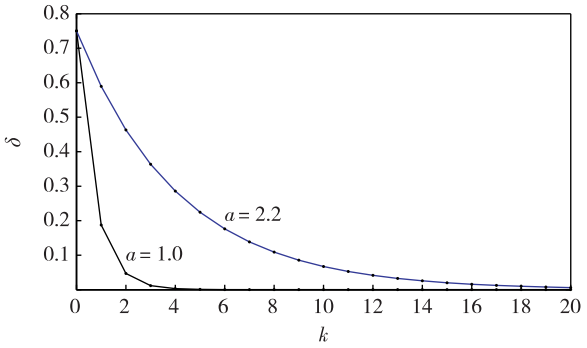

图 2　迭代误差 δ 随差 k 变化曲线($a=1.0$ 和 $a=2.2$)

Fig. 2　The iterative errors δ versus k ($a=1.0$ and $a=2.2$)

从这个例子可知,迭代求解算法是非常有用的. 求解代数方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 的雅可比(Jacobi)迭代和高斯-赛德尔(Gauss-Seidel)迭代是非常著名的迭代方法. 最近,本文作者等提出了求解(线性或非线性)代数方程的递阶梯度迭代算法和递阶最小二乘迭代算法,可用于求解 $\boldsymbol{Ax}=\boldsymbol{b}$,$\boldsymbol{AXB}=\boldsymbol{F}$,西尔维斯特矩阵方程 $\boldsymbol{AX}+\boldsymbol{XB}=\boldsymbol{F}$,以及一般矩阵方程[14-15]

$$\boldsymbol{A}_1\boldsymbol{XB}_1+\boldsymbol{A}_2\boldsymbol{XB}_2+\cdots+\boldsymbol{A}_p\boldsymbol{XB}_p=\boldsymbol{F},$$

和一般耦合矩阵方程[16-17]

$$\begin{cases}\boldsymbol{A}_{11}\boldsymbol{X}_1\boldsymbol{B}_{11}+\boldsymbol{A}_{12}\boldsymbol{X}_2\boldsymbol{B}_{12}+\cdots+\boldsymbol{A}_{1p}\boldsymbol{X}_p\boldsymbol{B}_{1p}=\boldsymbol{F}_1,\\ \boldsymbol{A}_{21}\boldsymbol{X}_1\boldsymbol{B}_{21}+\boldsymbol{A}_{22}\boldsymbol{X}_2\boldsymbol{B}_{22}+\cdots+\boldsymbol{A}_{2p}\boldsymbol{X}_p\boldsymbol{B}_{2p}=\boldsymbol{F}_2,\\ \quad\vdots\\ \boldsymbol{A}_{p1}\boldsymbol{X}_1\boldsymbol{B}_{p1}+\boldsymbol{A}_{p2}\boldsymbol{X}_2\boldsymbol{B}_{p2}+\cdots+\boldsymbol{A}_{pp}\boldsymbol{X}_p\boldsymbol{B}_{pp}=\boldsymbol{F}_p.\end{cases}$$

这里 $\boldsymbol{X}$ 和 $\boldsymbol{X}_j$ 是未知矩阵,$\boldsymbol{A}$,$\boldsymbol{B}$,$\boldsymbol{F}$,$\boldsymbol{F}_i$,$\boldsymbol{A}_i$,$\boldsymbol{B}_i$,$\boldsymbol{A}_{ij}$和 $\boldsymbol{B}_{ij}$ 为适当维数常数矩阵. 详细内容可参见文献[8]“递阶辨识方法”.

1.2.2　梯度搜索原理

假定无约束极值问题,

$$\min f(\boldsymbol{x}),\quad \boldsymbol{x}\in\mathbf{R}^n.$$

其中 $\mathbf{R}^n$ 为 n 维欧氏空间(Euclidean space),目标函数 $f(\boldsymbol{x})$ 有一阶连续偏导数,具有极小点 $\boldsymbol{x}^*$. 以 $\boldsymbol{x}^{(k)}$ 表示极小点的第 k 次近似. 为了求得第 $k+1$ 次近似点 $\boldsymbol{x}^{(k+1)}$,在点 $\boldsymbol{x}^{(k)}$ 沿方向 $\boldsymbol{p}^{(k)}$ 作射线

$$\boldsymbol{x}=\boldsymbol{x}^{(k)}+\lambda\boldsymbol{p}^{(k)},\quad \lambda\geqslant 0.$$

将 $f(\boldsymbol{x})$ 在 $\boldsymbol{x}^{(k)}$ 点处展成泰勒级数

$$f(\boldsymbol{x})=f(\boldsymbol{x}^{(k)}+\lambda\boldsymbol{p}^{(k)})=f(\boldsymbol{x}^{(k)})+\lambda[\nabla f(\boldsymbol{x}^{(k)})]^{\mathrm{T}}\boldsymbol{p}^{(k)}+o(\lambda),$$

其中 $\nabla f(\boldsymbol{x}^{(k)})$ 或 $\mathrm{grad}[f(x^{(k)})]$ 为 $f(\boldsymbol{x})$ 在点 $\boldsymbol{x}^{(k)}$ 的梯度,且

$$\lim_{\lambda\to 0}\frac{o(\lambda)}{\lambda}=0.$$

对于充分小的 λ,只要

$$\nabla f(\boldsymbol{x}^{(k)})^{\mathrm{T}}\boldsymbol{p}^{(k)}<0,\tag{12}$$

就可保证 $f(\boldsymbol{x}^{(k)}+\lambda\boldsymbol{p}^{(k)})<f(\boldsymbol{x}^{(k)})$. 这时若取

$$\boldsymbol{x}^{(k+1)}=\boldsymbol{x}^{(k)}+\lambda\boldsymbol{p}^{(k)},$$

就能通过迭代,使目标函数值得到改善.

下面再考察不同的方向 $\boldsymbol{p}^{(k)}$. 假定 $\boldsymbol{p}^{(k)}$ 的模一定,且不为 0,并设 $\nabla f(\boldsymbol{x}^{(k)})\neq 0$(否则,$\boldsymbol{x}^{(k)}$ 是一驻点),使式(12)成立的 $\boldsymbol{p}^{(k)}$ 有无限多个. 为了使目标函数能得到尽量大的改善,必须寻求使 $\nabla f(\boldsymbol{x}^{(k)})^{\mathrm{T}}\boldsymbol{p}^{(k)}$ 取最小值的 $\boldsymbol{p}^{(k)}$. 由线性代数知道

$$\nabla f(\boldsymbol{X}^{(k)})^{\mathrm{T}}\boldsymbol{p}^{(k)}=\|\nabla f(\boldsymbol{X}^{(k)})\|\cdot\|\boldsymbol{p}^{(k)}\|\cos\gamma,$$

其中 γ 为向量 $\nabla f(\boldsymbol{x}^{(k)})$ 与 $\boldsymbol{p}^{(k)}$ 的夹角. 当 $\boldsymbol{p}^{(k)}$ 与 $\nabla f(\boldsymbol{x}^{(k)})$ 同向时,$\gamma=0°$,$\cos\gamma=1$,$\nabla f(\boldsymbol{x}^{(k)})^{\mathrm{T}}\boldsymbol{p}^{(k)}$ 取最大值;当 $\boldsymbol{p}(k)$ 与 $\nabla f(\boldsymbol{x}^{(k)})$ 反向时 $\gamma=180°$,$\cos\gamma=-1$,这时式(12)成立,而且其左端取最小值,我们称方向

$$\boldsymbol{p}^{(k)}=-\nabla f(\boldsymbol{x}^{(k)})$$

为负梯度方向,它是使函数值下降最快的方向(在 $\boldsymbol{x}^{(k)}$ 的某一小范围内). 在最小化问题中,寻求的正是这一方向. 为了得到下一个近似极小,在选定了搜索方向后,还要确定步长 λ,有很多方法可用于选择步长 λ.

一种方法就是取 λ 为某一常数进行试算,检验是否满足不等式

$$f(\boldsymbol{x}^{(k)}-\lambda\nabla f(\boldsymbol{x}^{(k)}))<f(\boldsymbol{x}^{(k)}).\tag{13}$$

若上述不等式成立,就可以迭代下去,否则缩小 λ 使满足式(13). 由于采用负梯度方向,满足不等式(13)的 λ 是存在的.

另一种方法是解

$$\min_{\lambda\geqslant 0} f(\boldsymbol{x}^{(k)} - \lambda \nabla f(\boldsymbol{x}^{(k)})). \tag{14}$$

即通过负梯度方向的一维搜索来确定使 $f(\boldsymbol{x})$ 最小的 $\lambda=\lambda_k$. 这种梯度法就是所谓的最速下降法. 最速下降法(steepest descent method)是一种最基本的算法,由法国数学家 Cauchy 于 1847 年首先提出. 在每次迭代中,沿最速下降方向(负梯度方向)进行搜索,每步沿负梯度方向取最优步长,因此这种方法也称为最优梯度法. 其特点是方法简单,只以一阶梯度的信息确定下一步的搜索方向,工作量小,存储变量较少,初始点要求不高,但收敛速度慢,越是接近极值点,收敛越慢. 最速下降法适用于寻优过程的前期迭代或作为间插步骤,当接近极值点时,宜选用其他收敛快的算法.

1.2.3 梯度迭代算法实现

现将最速下降法算法步骤总结如下.

1) 给定初始近似点 $\boldsymbol{x}_0\in\mathbf{R}^n$ 及精度 $\varepsilon>0$.

2) 已给定 $\boldsymbol{x}^{(k)}$. 如 $\|\nabla f(\boldsymbol{x}^{(k)})\|\leqslant\varepsilon$,则 $\boldsymbol{x}^{(k)}$ 为所求得近似解;如 $\|\nabla f(\boldsymbol{x}^{(k)})\|>\varepsilon$,则用试探法选择一个小步长 λ_k,确定下一个近似点

$$\boldsymbol{x}^{(k+1)}=\boldsymbol{x}^{(k)}-\lambda_k\nabla f(\boldsymbol{x}^{(k)}),$$

或用一维搜索法求步长 λ_k,用式(14)求最佳的步长 λ_k,即函数

$$g(\lambda)=f(\boldsymbol{x}^{(k)}-\lambda\nabla f(\boldsymbol{x}^{(k)})) \tag{15}$$

的最小值点 $\lambda=\lambda_k$,并确定下一个近似点

$$\boldsymbol{x}^{(k+1)}=\boldsymbol{x}^{(k)}-\lambda_k\nabla f(\boldsymbol{x}^{(k)}),$$

或

$$\boldsymbol{x}^{(k+1)}=\boldsymbol{x}^{(k)}-\lambda_k\mathrm{grad}[f(\boldsymbol{x}^{(k)})], \tag{16}$$

直到达到要求的精度为止.

例 1 求 $\min f(\boldsymbol{x})=\min(x_1^2+4x_2^2)$,$\varepsilon=0.10$(保留 2 位小数). 结合计算步骤,取 $\boldsymbol{x}^{(0)}=(2,2)^{\mathrm{T}}$. 计算梯度:

$$\nabla f(\boldsymbol{x}^{(k)})=(2x_1,8x_2)^{\mathrm{T}},\ \nabla f(\boldsymbol{x}^{(0)})=(4,16)^{\mathrm{T}},$$

$$\|\nabla f(\boldsymbol{x}^{(0)})\|=\sqrt{4^2+16^2}>16.49>\varepsilon,$$

故需进一步迭代求 $f(\boldsymbol{x}^{(1)})$. 对于

$$\min_{\lambda\geqslant 0} f(\boldsymbol{x}^{(0)}-\lambda\nabla f(\boldsymbol{x}^{(0)})),$$

记

$$\varphi(\lambda)=f(\boldsymbol{x}^{(0)}-\lambda\nabla f(\boldsymbol{x}^{(0)}))=f(2-4\lambda,2-16\lambda)=(2-4\lambda)^2+4(2-16\lambda)^2.$$

显然 $\varphi''(\lambda)>0$,所以使 $\varphi'(\lambda)=0$ 的即为所求得一维搜索极小点.

令 $\varphi'(\lambda)=0$,可得 $\lambda_0\approx 0.13$,所以

$$\boldsymbol{x}^{(1)}=\boldsymbol{x}^{(0)}-\lambda_0\nabla f(\boldsymbol{x}^{(0)})=(1.48,-0.08)^{\mathrm{T}},$$

$$\nabla f(\boldsymbol{x}^{(1)})=(2.96,-0.64)^{\mathrm{T}},$$

$$\|\nabla f(\boldsymbol{x}^{(1)})\|=3.03>\varepsilon.$$

如还需要继续迭代,则用同样的方法迭代. 各步情况见表 3,其中 $\boldsymbol{x}^{(k)}=(x_1^{(k)},x_2^{(k)})$.

表 3 迭代计算结果

Table 3 The results of iteration

k	$(x_1^{(k)},x_2^{(k)})$	$\nabla f(\boldsymbol{x}^{(k)})$	$\|\nabla f(\boldsymbol{x}^{(k)})\|$	λ_k
0	(2,2)	(4,16)	16.49	0.13
1	(1.48,-0.08)	(2.96,-0.64)	3.03	0.44
2	(0.18,0.20)	(0.36,1.60)	1.64	0.13
3	(0.13,0.00)	(0.26,0.00)	0.26	0.50
4	(0.00,0.00)	(0.00,0.00)	$0.00<\varepsilon$	

由表 3 可知近似最优解为 $\boldsymbol{x}^{(4)}=(0,0)^{\mathrm{T}}$,事实上它也是真实最优解.

1.3 牛顿迭代方法

牛顿迭代法是求解非线性方程 $f(x)=0$ 的一种重要和常用的迭代方法,其基本思想是将非线性函数逐步线性化,从而将非线性方程 $f(x)=0$ 近似地转化为线性方程来求解[18].

1.3.1 求方程的根或函数的零点

计算实系数 x 的二次三项式

$$f(x)=ax^2+bx+c,\quad a\neq 0$$

的零点,即方程 $ax^2+bx+c=0$ 的根,可以利用求根公式

$$x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}.$$

求 3 次和 4 次实系数多项式的零点,也有相应的代数求根公式. 但是,Abel 定理告诉我们,一般 5 次及 5 次以上实系数方程

$$a_0x^n+a_1x^{n-1}+a_2x^{n-2}+\cdots+a_{n-1}x+a_n=0,a_0\neq 0,n\geqslant 5$$

的代数求根公式不存在. 为求解高于 5 次方程的根,牛顿(Newton)迭代方法就显示出独到优点. 当然,牛顿方法也适合求非多项式方程的根.

牛顿方法可以找已知函数的近似根. 假设函数 $f(x)$ 在闭区间 $[a,b]$ 上可微,且 $f(a)f(b)<0$. 目的是在 $[a,b]$ 内找一点 r 是 $f(x)=0$ 的根. 设函数 $f(x)$ 如图 3 所示,$f(x)$ 有根 $x=r$.

设点 x_0 是 r 的一个初始近似根,在点 x_0 作横轴垂线交函数曲线于点 $(x_0,f(x_0))$,在点 $(x_0,f(x_0))$ 作切线交横轴于一个新点 x_1,参见图 4. 那么 x_1 是 r 的一个比 x_0 更好的一个近似. 现在重复这个过程,得到一系列点 $x_2,x_3,\cdots$,直到足够接近 r. 图 5 说明了确定 x_2 的过程.

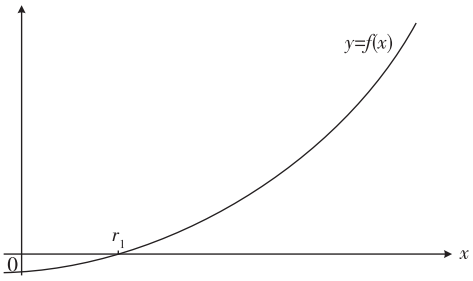

图 3　函数 $y=f(x)$ 曲线

Fig. 3　The diagram of function $y=f(x)$

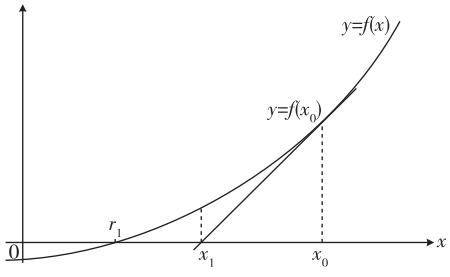

图 4　经过点 $(x_0,f(x_0))$ 的切线

Fig. 4　The diagram of the tangent passing $(x_0,f(x_0))$

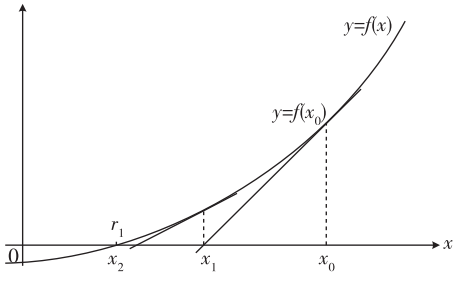

图 5　经过点 $(x_1,f(x_1))$ 的切线

Fig. 5　The diagram of the tangent passing $(x_1,f(x_1))$

下面用数学表达式描述这个过程. 由图 4 知,通过点 $(x_0,f(x_0))$ 切线的斜率等于过点 $(x_1,0)$ 和点 $(x_0,f(x_0))$ 直线的斜率,即

$$f'(x_0)=\frac{0-f(x_0)}{x_1-x_0},$$

或

$$x_1=x_0-\frac{f(x_0)}{f'(x_0)}.$$

重复此过程,得到求解函数 $f(x)$ 零点或方程 $f(x)=0$ 的根的迭代公式:

$$x_k=x_{k-1}-\frac{f(x_{k-1})}{f'(x_{k-1})},\quad k=1,2,3,\cdots. \tag{17}$$

这个迭代技术就是牛顿方法的核心,它很容易用计算机程序实现.

例 2　考虑函数 $f(x)=x^7+9x^5-13x-17$. 区间 $[1,2]$ 上有一个根(因为 $f(1)<0,f(2)>0$). 设初始点 $x_0=1$,使用牛顿方法计算的结果如表 4 所示.

表 4　迭代计算结果

Table 4　The results of iteration

k	x_{k-1}	$f(x_{k-1})$	$f'(x_{k-1})$	x_k
1	1.000 000 000	−20.000 000 00	39.000 000 0	1.512 820 513
2	1.512 820 513	52.782 871 88	306.613 073 9	1.340 672 368
3	1.340 672 368	12.337 512 68	173.027 006 2	1.269 368 397
4	1.269 368 397	1.469 113 53	133.115 961 8	1.258 332 053
5	1.258 332 053	0.030 535 47	127.610 724 3	1.258 092 767
6	1.258 092 767	0.000 014 07	127.493 240 3	1.258 092 657

在迭代计算 6 次后,根 r 的一个近似为 $r=1.258\,092\,657$,精度达到 10^{-6}.

1.3.2　求函数极值

牛顿方法(Newton method)是一个著名的找一个或多个自变量方程根的算法. 它可以用来找标量函数 $f(x)$ 的局部最大和最小点 x^*. 如果 x^* 是 $f(x)$ 的平衡点,那么 x^* 是 $f(x)$ 导数 $f'(x)$ 的根. 因此,牛顿方法可应用于导函数 $f'(x)$. 假设 $f(x)$ 是一个二次可微函数,选择初值 x_0 足够接近 x^*,那么由下式定义的序列 $\{x_k,k=1,2,3,\cdots\}$ 收敛于 x^*,

$$x_k=x_{k-1}-\frac{f'(x_{k-1})}{f''(x_{k-1})}. \tag{18}$$

式(18)就是求函数 $f(x)$ 极值点的牛顿迭代算法.

此迭代算法可以推广到多个自变量(高维)情形,即求多元函数 $f(\boldsymbol{x})=f(x_1,x_2,\cdots,x_n)$ 的极值. 对于多个自变量 $\boldsymbol{x}=(x_1,x_2,\cdots,x_n)$,只需把导数用梯度

$$\text{grad}[f(\boldsymbol{x})]:=\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}}=\begin{bmatrix}\dfrac{\partial f(\boldsymbol{x})}{\partial x_1}\\ \dfrac{\partial f(\boldsymbol{x})}{\partial x_2}\\ \vdots\\ \dfrac{\partial f(\boldsymbol{x})}{\partial x_n}\end{bmatrix}\in\mathbf{R}^n$$

代替(梯度有时也用 $\nabla f(\boldsymbol{x})$ 表示),二阶导数换为 Hessian 矩阵 $\boldsymbol{H}(\boldsymbol{x})$,就可得到求多元函数 $f(\boldsymbol{x})$ 极值点的牛顿迭代算法:

$$\boldsymbol{x}_k=\boldsymbol{x}_{k-1}-[\boldsymbol{H}(\boldsymbol{x}_{k-1})]^{-1}\text{grad}[f(\boldsymbol{x}_{k-1})],$$

$$\boldsymbol{H}(\boldsymbol{x})=\frac{\partial^2 f(\boldsymbol{x})}{\partial \boldsymbol{x}\partial \boldsymbol{x}^{\mathrm{T}}}. \tag{19}$$

通常在牛顿方法中引入一个收敛因子或小步长(step-size)μ 或变步长因子 μ_k,得到

$$\boldsymbol{x}_k=\boldsymbol{x}_{k-1}-\mu_k[\boldsymbol{H}(\boldsymbol{x}_{k-1})]^{-1}\mathrm{grad}[f(\boldsymbol{x}_{k-1})],\quad k=1,2,3,\cdots,n. \tag{20}$$

梯度是函数 $f(\boldsymbol{x})=f(x_1,x_2,\cdots,x_n)$ 对列向量 $\boldsymbol{x}\in\mathbf{R}^n$ 的一阶偏导数,本文把梯度定义为一个列向量. 海赛矩阵是函数 $f(\boldsymbol{x})=f(x_1,x_2,\cdots,x_n)$ 的二阶偏导数,即梯度对 $\boldsymbol{x}^{\mathrm{T}}$ 的导数. 海赛矩阵(Hessian matrix)定义为

$$\boldsymbol{H}(\boldsymbol{x}):=\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}\partial \boldsymbol{x}^{\mathrm{T}}}=\frac{\partial \mathrm{grad}[f(\boldsymbol{x})]}{\partial \boldsymbol{x}^{\mathrm{T}}}=\begin{bmatrix}\frac{\partial^2 f(\boldsymbol{x})}{\partial x_1^2} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1\partial x_2} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1\partial x_n}\\ \frac{\partial^2 f(\boldsymbol{x})}{\partial x_2\partial x_1} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_2^2} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_2\partial x_n}\\ \vdots & \vdots & & \vdots\\ \frac{\partial^2 f(\boldsymbol{x})}{\partial x_n\partial x_1} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_n\partial x_2} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_n^2}\end{bmatrix}\in\mathbf{R}^{n\times n}.$$

牛顿方法的几何解释如下.

在每一次迭代中,$f(\boldsymbol{x})$ 用在 $\boldsymbol{x}=\boldsymbol{x}_{k-1}$ 的二次函数近似,即用在点 $\boldsymbol{x}=\boldsymbol{x}_{k-1}$ 展开为 Taylor 级数的二次项来近似:

$$f(\boldsymbol{x})=f(\boldsymbol{x}_{k-1})+f'(\boldsymbol{x}_{k-1})(\boldsymbol{x}-\boldsymbol{x}_{k-1})+\frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}_{k-1})^{\mathrm{T}}\boldsymbol{H}(\boldsymbol{x}_{k-1})(\boldsymbol{x}-\boldsymbol{x}_{k-1})+o(\|\boldsymbol{x}-\boldsymbol{x}_{k-1}\|^2).$$

然后通过极小或极大这个二次函数来获得下一个点 $\boldsymbol{x}=\boldsymbol{x}_k$:令 $f'(\boldsymbol{x})=\boldsymbol{0}$ 得到迭代解(20).

牛顿方法比找极小或极大的梯度下降方法收敛速度快,然而,使用牛顿方法需要知道 $f(\boldsymbol{x})$ 的海赛矩阵,这可能使计算困难. 有多种准牛顿方法(quasi-Newton methods)使用近似的海赛矩阵. 牛顿方法的不足之处是需要支付大量的计算,它在每一步都要计算海赛矩阵及其逆.

如果牛顿迭代算法中的迭代变量 k 换为与时间有关的递推变量 t,在辨识中就得到牛顿递推算法.

如前所述,迭代方法能够用于线性回归模型、伪线性回归模型以及非线性系统辨识算法的研究,包括方程误差类模型(CAR,CARMA,CARAR,CARAR-MA)和输出误差类模型(OE,OEMA,OEAR,Box-Jenkins).

本文应用迭代技术,仅讨论线性系统受控自回归滑动平均模型(CARMA)和 Box-Jenkins 模型的迭代辨识问题. 非线性系统的迭代辨识可参考文献[6-7,12]. 迭代辨识的相关研究成果和值得研究的课题如下:

1) CARMA 模型的最小二乘迭代辨识方法[19];

2) 动态调节模型的最小二乘迭代辨识方法[20];

3) 有限数据长度下多变量 CARMA 系统的最小二乘迭代参数估计方法[21];

4) 多变量 CARARMA 系统的最小二乘迭代辨识方法和梯度迭代辨识方法[22];

5) 使用数据滤波的输出误差滑动平均系统(OEMA)的最小二乘递推辨识方法与最小二乘迭代辨识方法[23];

6) 有限量测数据下 Box-Jenkins 模型的最小二乘迭代方法[24];

7) 有限量测数据下 Box-Jenkins 模型的梯度迭代参数估计方法[25];

8) 有限量测数据下输出误差系统和输出误差滑动平均系统梯度迭代辨识方法与最小二乘迭代辨识方法[11];

9) 基于过参数化方法的 Hammerstein 非线性 ARMAX 系统的最小二乘迭代方法和递推增广最小二乘辨识方法[6];

10) 基于过参数化方法的 Hammerstein 非线性 ARMAX 系统的梯度迭代方法和增广随机梯度辨识方法[7];

11) 多变量系统(multivariable ARX-like system)的递阶梯度迭代辨识方法和递阶最小二乘迭代辨识方法[9-10,26];

12) 一类非均匀采样输出误差系统的辅助模型最小二乘迭代辨识方法[27];

13) 有色噪声系统的迭代辨识与递推辨识方法仿真比较研究[28];

14) 双输入多率输出误差系统的辅助模型最小二乘迭代辨识方法[29];

15) 多率多输入系统的最小二乘迭代辨识方法与梯度迭代辨识方法[30];

16) 有色噪声干扰多变量系统的最小二乘迭代辨识和梯度迭代辨识;

17) 有色噪声干扰 Hammerstein 非线性方程误差类系统的最小二乘迭代辨识和梯度迭代辨识;

18) Hammerstein 非线性输出误差类系统的最小二乘迭代辨识和梯度迭代辨识[12];

19) Wiener 非线性系统的最小二乘迭代辨识和

梯度迭代辨识[13];

20) 非均匀采样数据系统的最小二乘迭代辨识和梯度迭代辨识[13];

21) 反馈回路非线性系统的最小二乘迭代辨识和梯度迭代辨识;

22) 多变量受控自回归滑动平均系统(multivariable CARMA-like system)递阶增广最小二乘辨识算法与递阶最小二乘迭代算法[32];

23) 多变量输出误差滑动平均系统(multivariable OEMA-like system)的递阶梯度迭代参数估计算法[33].

2 受控自回归滑动平均模型(CARMA)

受控自回归滑动平均模型(CARMA, Controlled AutoRegressive Moving Average model),或称为带外加输入自回归滑动平均模型(ARMAX, AutoRegressive Moving Average model with eXogenous input)是一种应用非常广泛的系统模型. 现基于迭代技术、最小二乘原理和梯度搜索原理,研究辨识 CARMA 模型的最小二乘迭代辨识方法和梯度迭代辨识方法. 与 CARMA 模型的递推增广最小二乘算法和增广随机梯度算法相比,两个迭代算法在每一步迭代计算中,同时利用了系统所有量测数据信息,因而分别比同类的递推增广最小二乘算法和增广随机梯度递推算法具有更高的参数精度和更快的收敛速度.

考虑 CARMA/ARMAX 模型描述的系统:

$$A(z)y(t) = B(z)u(t) + D(z)v(t), \quad (21)$$

其中$\{u(t)\}$和$\{y(t)\}$分别是系统的输入和输出序列,$\{v(t)\}$是零均值方差为 σ^2 的随机白噪声序列,z^{-1}为单位后移算子($z^{-1}y(t) = y(t-1)$, $zy(t) = y(t+1)$),$A(z)$, $B(z)$和 $D(z)$是单位后移算子 z^{-1} 的多项式:

$$A(z) := 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_{n_a} z^{-n_a},$$

$$B(z) := b_1 z^{-1} + b_2 z^{-2} + \cdots + b_{n_b} z^{-n_b},$$

$$D(z) := 1 + d_1 z^{-1} + d_2 z^{-2} + \cdots + d_{n_d} z^{-n_d}.$$

设阶次 n_a, n_b 和 n_d 已知,记 $n := n_a + n_b + n_d$,且 $t \leqslant 0$ 时,$y(t) = 0$, $u(t) = 0$, $v(t) = 0$.

2.1 递推增广最小二乘算法

先介绍 CARMA 模型的递推增广最小二乘算法,并分析其问题,进而引出最小二乘迭代辨识方法. 定义增广参数向量 $\boldsymbol{\theta}$ 和包含噪声项的增广信息向量 $\boldsymbol{\varphi}(t)$如下:

$$\boldsymbol{\theta} := [a_1, a_2, \cdots, a_{n_a}, b_1, b_2, \cdots, b_{n_b}, d_1, d_2, \cdots, d_{n_d}]^{\mathrm{T}} \in \mathbf{R}^n,$$

$$\boldsymbol{\varphi}(t) := [-y(t-1), -y(t-2), \cdots, -y(t-n_a), u(t-1), u(t-2), \cdots, u(t-n_b), v(t-1), v(t-2), \cdots, v(t-n_d)]^{\mathrm{T}} \in \mathbf{R}^n. \quad (22)$$

由式(21)可得 CARMA 模型描述系统的辨识模型:

$$y(t) = [1 - A(z)]y(t) + B(z)u(t) + D(z)v(t) = \boldsymbol{\varphi}^{\mathrm{T}}(t)\boldsymbol{\theta} + v(t). \quad (23)$$

下列递推增广最小二乘算法(RELS)可以获得模型(23)参数向量 $\boldsymbol{\theta}$ 的估计 $\hat{\boldsymbol{\theta}}(t)$:

$$\hat{\boldsymbol{\theta}}(t) = \hat{\boldsymbol{\theta}}(t-1) - \boldsymbol{P}(t-1)\hat{\boldsymbol{\phi}}(t)[y(t) - \hat{\boldsymbol{\phi}}^{\mathrm{T}}(t)\hat{\boldsymbol{\theta}}(t-1)], \quad (24)$$

$$\boldsymbol{P}(t) = \boldsymbol{P}(t-1) - \frac{\boldsymbol{P}(t-1)\hat{\boldsymbol{\phi}}(t)\hat{\boldsymbol{\phi}}^{\mathrm{T}}(t)\boldsymbol{P}(t-1)}{1 + \hat{\boldsymbol{\phi}}^{\mathrm{T}}(t)\boldsymbol{P}(t-1)\hat{\boldsymbol{\phi}}(t)},$$

$$\boldsymbol{P}(0) = p_0 \boldsymbol{I}, \quad (25)$$

$$\hat{\boldsymbol{\phi}}(t) = [-y(t-1), -y(t-2), \cdots, -y(t-n_a), u(t-1), u(t-2), \cdots, u(t-n_b), \hat{v}(t-1), \hat{v}(t-2), \cdots, \hat{v}(t-n_d)]^{\mathrm{T}}, \quad (26)$$

$$\hat{v}(t) = y(t) - \hat{\boldsymbol{\phi}}^{\mathrm{T}}(t)\hat{\boldsymbol{\theta}}(t), \quad (27)$$

$$\hat{\boldsymbol{\theta}}(t) = [\hat{a}_1(t), \hat{a}_2(t), \cdots, \hat{a}_{n_a}(t), \hat{b}_1(t), \hat{b}_2(t), \cdots, \hat{b}_{n_b}(t), \hat{d}_1(t), \hat{d}_2(t), \cdots, \hat{d}_{n_d}(t)]^{\mathrm{T}}. \quad (28)$$

RELS 算法计算步骤如下:

1) 令 $t = 1$,给定数据长度 L,置初值 $\hat{\boldsymbol{\theta}}(0) = \mathbf{1}_n / p_0$, $\boldsymbol{P}(0) = p_0 \boldsymbol{I}$, $p_0 = 10^6$, $\hat{v}(i) = 1/p_0$ for $i \leqslant 0$;

2) 采集输入输出数据 $u(t)$ 和 $y(t)$,由式(26)构成信息向量 $\hat{\boldsymbol{\phi}}(t)$;

3) 由式(25)计算协方差阵 $\boldsymbol{P}(t)$;

4) 通过式(24)刷新参数估计 $\hat{\boldsymbol{\theta}}(t)$;

5) 由式(27)计算估计残差 $\hat{v}(t)$;

6) 如果 $t = L$,则可终止递推计算过程,得到参数估计 $\hat{\boldsymbol{\theta}}(L)$,否则 t 增 1 转到第 2 步,进行递推计算.

RELS 算法计算参数估计 $\hat{\boldsymbol{\theta}}(t)$的流程如图 6 所示.

从上述递推计算过程看,RELS 算法有一个缺点:实际中人们只能采集到有限的数据(很有可能是小样本数据)$\{u(i), y(i): i = 1, 2, \cdots, L\}$(比如说数据长度 $L = 1\,000$),在每步递推计算参数估计中,例如在第 t 步(如 $t = 10$)递推计算参数估计时,RELS 算法只利用了 t 时刻及 t 时刻之前的数据$\{u(i), y(i): i = 1, 2, \cdots, t = 10\}$,而未利用系统 $t(=10)$时刻以后的数据$\{u(i), y(i): i = t+1 = 11, t+2, t+3, \cdots, L\}$. 也就是说,RELS 算法没有充分利用系统数据(尽管在整个递推计算获得 $\hat{\boldsymbol{\theta}}(L)$的过程中,RELS 算法利用了系统所有数据). 一个自然的问题是:是否存在一个迭代算法,在每步迭代计算过程中,同时使用系统可得到的所有数据$\{u(i), y(i): i = 1, 2, 3,$

…,L},以从数据中提取尽可能多的知识来提高参数估计精度? 答案是肯定的. 这就是下面要介绍的梯度迭代算法和最小二乘迭代辨识方法. 注意:从这一段中我们可以明白充分利用系统数据与利用了系统所有数据是两个不同的概念.

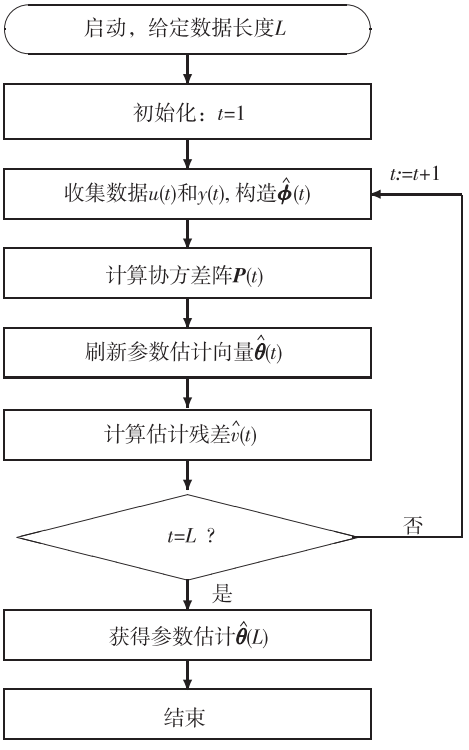

图6 计算 RELS 参数估计 $\hat{\boldsymbol{\theta}}(L)$ 的流程

Fig. 6 The flowchart for computing the RELS parameter estimate $\hat{\boldsymbol{\theta}}(L)$

2.2 最小二乘迭代辨识方法

考虑从 $i=t-p+1$ 到 $i=t$ 最新的 p 组数据,定义堆积输出向量 $\boldsymbol{Y}(t)$、堆积信息矩阵 $\boldsymbol{\Phi}(t)$、堆积白噪声向量 $\boldsymbol{V}(t)$ 如下:

$$\boldsymbol{Y}(t):=\begin{bmatrix} y(t)\\ y(t-1)\\ \vdots\\ y(t-p+1)\end{bmatrix}\in\mathbf{R}^{p};$$

$$\boldsymbol{\Phi}(t):=\begin{bmatrix} \boldsymbol{\varphi}^{\mathrm{T}}(t)\\ \boldsymbol{\varphi}^{\mathrm{T}}(t-1)\\ \vdots\\ \boldsymbol{\varphi}^{\mathrm{T}}(t-p+1)\end{bmatrix}\in\mathbf{R}^{p\times n};\tag{29}$$

$$\boldsymbol{V}(t):=\begin{bmatrix} v(t)\\ v(t-1)\\ \vdots\\ v(t-p+1)\end{bmatrix}\in\mathbf{R}^{p}.\tag{30}$$

注意:如果取 $p=L,t=L$ (L 为数据长度),那么 $\boldsymbol{Y}(t)$ 和 $\boldsymbol{\Phi}(t)$ 就包含了所有量测输入输出数据{$u(t)$, $y(t):t=0,1,2,\cdots,L$}(参见后面的有限量测数据 CARMA 模型的最小二乘的迭代算法). 由式(23)可得

$$\boldsymbol{Y}(t)=\boldsymbol{\Phi}(t)\boldsymbol{\theta}+\boldsymbol{V}(t).\tag{31}$$

定义准则函数:

$$J_3(\boldsymbol{\theta}):=\|\boldsymbol{Y}(t)-\boldsymbol{\Phi}(t)\boldsymbol{\theta}\|^2.$$

极小化准则函数 $J_3(\boldsymbol{\theta})$,令其对 $\boldsymbol{\theta}$ 的导数为零得到

$$\frac{\partial J_3(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}=-2\boldsymbol{\Phi}^{\mathrm{T}}(t)[\boldsymbol{Y}(t)-\boldsymbol{\Phi}(t)\boldsymbol{Y}(t)]=0.$$

假设信息向量 $\boldsymbol{\varphi}(t)$ 是持续激励的($p\geqslant n_a+n_b+n_d$),即[$\boldsymbol{\Phi}^{\mathrm{T}}(t)\boldsymbol{\Phi}(t)$]是可逆矩阵. 上式给出参数向量 $\boldsymbol{\theta}$ 的最小二乘估计(LSE, Least Squares Estimate)(LS 估计):

$$\hat{\boldsymbol{\theta}}(t)=[\boldsymbol{\Phi}^{\mathrm{T}}(t)\boldsymbol{\Phi}(t)]^{-1}\boldsymbol{\Phi}^{\mathrm{T}}(t)\boldsymbol{Y}(t).\tag{32}$$

式(32)不可能计算最小二乘估计 $\hat{\boldsymbol{\theta}}(t)$,因为 $\boldsymbol{\Phi}(t)$(也就是 $\boldsymbol{\varphi}(t)$)中包含了不可测噪声项 $v(t-i)$(参见式(22)). 这里采用递阶辨识原理:令 $k=1,2,3,\cdots$是一个迭代变量,$\hat{\boldsymbol{\theta}}_k(t)$ 为 $\boldsymbol{\theta}$ 的迭代估计,信息向量 $\boldsymbol{\varphi}(t)$ 中未知项 $v(t-i)$ 用其第 $k-1$ 次迭代估计值 $\hat{v}_{k-1}(t-i)$ 代替,代替后的 $\boldsymbol{\varphi}(t)$ 记作

$$\hat{\boldsymbol{\varphi}}_k(t):=[-y(t-1),-y(t-2),\cdots,-y(t-n_a),u(t-1),u(t-2),\cdots,u(t-n_b),\hat{v}_{k-1}(t-1),\hat{v}_{k-1}(t-2),\cdots,\hat{v}_{k-1}(t-n_d)]^{\mathrm{T}}\in\mathbf{R}^{n_a+n_b+n_d}.\tag{33}$$

由式(33)可得

$$v(t-i)=y(t-i)-\boldsymbol{\varphi}^{\mathrm{T}}(t-i)\boldsymbol{\theta}.$$

用 $\hat{\boldsymbol{\varphi}}_k(t-i)$ 和 $\hat{\boldsymbol{\theta}}_k(t)$ 代替上式中 $\boldsymbol{\varphi}(t-i)$ 和 $\boldsymbol{\theta}$,那么 $v(t-i)$ 的第 k 次迭代估计 $\hat{v}_k(t-i)$ 可由下式计算:

$$\hat{v}_k(t-i)=y(t-i)-\hat{\boldsymbol{\varphi}}_k^{\mathrm{T}}(t-i)\hat{\boldsymbol{\theta}}_k(t).\tag{34}$$

用 $\hat{\boldsymbol{\varphi}}_k(t-i)$ 代替 $\boldsymbol{\Phi}(t)$ 中未知 $\boldsymbol{\varphi}(t-i)$,代替后的 $\boldsymbol{\Phi}(t)$ 记作

$$\hat{\boldsymbol{\Phi}}_k(t):=\begin{bmatrix}\hat{\boldsymbol{\varphi}}_k^{\mathrm{T}}(t)\\ \hat{\boldsymbol{\varphi}}_k^{\mathrm{T}}(t-1)\\ \vdots\\ \hat{\boldsymbol{\varphi}}_k^{\mathrm{T}}(t-p+1)\end{bmatrix}\in\mathbf{R}^{p\times n}.\tag{35}$$

用 $\hat{\boldsymbol{\Phi}}_k(t)$ 代替式(32)中 $\boldsymbol{\Phi}(t)$,可得 CARMA 模型的最小二乘迭代辨识算法(CARMA-LSI, Least Squares based Iterative identification algorithm for CARMA models)(CARMA-LSI 算法):

$$\hat{\boldsymbol{\theta}}_k(t)=[\hat{\boldsymbol{\Phi}}_k^{\mathrm{T}}(t)\hat{\boldsymbol{\Phi}}_k(t)]^{-1}\hat{\boldsymbol{\Phi}}_k^{\mathrm{T}}(t)\boldsymbol{Y}(t),\quad k=1,2,3,\cdots,\tag{36}$$

$$\hat{\boldsymbol{\Phi}}_k(t)=[\hat{\boldsymbol{\varphi}}_k(t),\hat{\boldsymbol{\varphi}}_k(t-1),\cdots,\hat{\boldsymbol{\varphi}}_k(t-p+1)]^{\mathrm{T}},\tag{37}$$

$$\boldsymbol{Y}(t)=[y(t),y(t-1),\cdots,y(t-p+1)]^{\mathrm{T}},\tag{38}$$

$$\hat{\boldsymbol{\varphi}}_k(t)=[\,-y(t-1),-y(t-2),\cdots,-y(t-n_a),u(t-1),u(t-2),\cdots,u(t-n_b),\hat{v}_{k-1}(t-1),\hat{v}_{k-1}(t-2),\cdots,\hat{v}_{k-1}(t-n_d)\,]^{\mathrm{T}},\tag{39}$$

$$\hat{v}_k(t-i)=y(t-i)-\hat{\boldsymbol{\varphi}}_k^{\mathrm{T}}(t-i)\hat{\boldsymbol{\theta}}_k(t),\quad i=1,2,\cdots,n_d.\tag{40}$$

迭代辨识采用交互估计理论和递阶辨识原理,在每步迭代计算中,参数估计 $\hat{\boldsymbol{\theta}}_k(t)$ 依赖于噪声估计 $\hat{v}_{k-1}(t-i)$,参见式(36),(37)和(39);反过来,噪声估计 $\hat{v}_k(t-i)$ 又通过前一步迭代的参数估计 $\hat{\boldsymbol{\theta}}_k(t)$ 计算,参见式(40),二者执行了一个递阶计算过程.

CARMA-LSI 算法的计算步骤如下.

1) 确定 p,令 $t=p$. 收集输入输出数据 $\{u(i),y(i):i=0,1,\cdots,p-1\}$,给定参数估计精度 ε.

2) 收集输入输出数据 $u(t)$ 和 $y(t)$,用式(38)构造 $\boldsymbol{Y}(t)$.

3) 令 $k=1$,置初值($i=1,2,\cdots,n_d$)(这样的初值是保证式(36)中矩阵 $\hat{\boldsymbol{\Phi}}_k^{\mathrm{T}}(t)\hat{\boldsymbol{\Phi}}_k(t)$ 可逆的).

4) 用式(39)构造 $\hat{\boldsymbol{\varphi}}_k(t)$,用式(37)构造 $\hat{\boldsymbol{\Phi}}_k(t)$.

5) 用式(36)刷新参数估计 $\boldsymbol{\theta}_k(t)$.

6) 用式(40)计算 $\hat{v}_k(t-i)$.

7) 比较 $\hat{\boldsymbol{\theta}}_k(t)$ 与 $\hat{\boldsymbol{\theta}}_{k-1}(t)$:如果 $\|\hat{\boldsymbol{\theta}}_k(t)-\hat{\boldsymbol{\theta}}_{k-1}(t)\|>\varepsilon$,$k$ 增 1,转到步骤 4;否则,获得迭代次数 k 和参数估计向量 $\hat{\boldsymbol{\theta}}_k(t)$,$t$ 增 1,转到步骤 2.

CARMA-LSI 算法计算参数估计 $\hat{\boldsymbol{\theta}}_k(t)$ 的流程如图 7 所示.

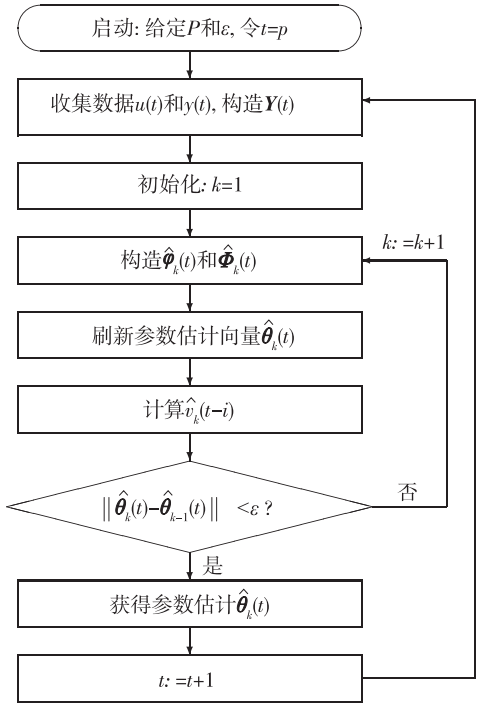

图 7 计算 CARMA-LSI 参数估计 $\hat{\boldsymbol{\theta}}_k(t)$ 的流程

Fig. 7 The flowchart for computing the CARMA-LSI parameter estimate $\hat{\boldsymbol{\theta}}_k(t)$

CARMA-LSI 算法是利用数据窗长度为 p 的有限数据窗内的数据(动态数据窗,因为窗是随 t 移动的)极小化准则函数得到的,因此具有跟踪时变参数的能力,也可用于在线辨识. 后面要讨论的与时间 t 有关的迭代算法,如 CARMA-GI, CARAR-LSI, CAR-AR-GI 算法等,都具有这些性质.

2.2.1 有限量测数据 CARMA-LSI 算法

在式(29)—(30)中取 $p=L,t=L$ (L 为数据长度),有

$$\boldsymbol{Y}(L):=\begin{bmatrix}y(L)\\y(L-1)\\\vdots\\y(1)\end{bmatrix}\in\mathbf{R}^L,$$

$$\boldsymbol{\Phi}(L):=\begin{bmatrix}\boldsymbol{\varphi}^{\mathrm{T}}(L)\\\boldsymbol{\varphi}^{\mathrm{T}}(L-1)\\\vdots\\\boldsymbol{\varphi}^{\mathrm{T}}(1)\end{bmatrix}\in\mathbf{R}^{L\times n},\tag{41}$$

$$\boldsymbol{V}(L):=\begin{bmatrix}v(L)\\v(L-1)\\\vdots\\v(1)\end{bmatrix}\in\mathbf{R}^L.\tag{42}$$

$\boldsymbol{Y}(L)$ 和 $\boldsymbol{\Phi}(L)$ 就包含了所有量测输入输出数据 $\{u(t),y(t):t=0,1,2,\cdots,L\}$. 由式(23)可得

$$\boldsymbol{Y}(L)=\boldsymbol{\Phi}(L)\boldsymbol{\theta}+\boldsymbol{V}(L).\tag{43}$$

定义准则函数:

$$J_4(\boldsymbol{\theta}):=\|\boldsymbol{Y}(L)-\boldsymbol{\Phi}(L)\boldsymbol{\theta}\|^2.\tag{44}$$

按照 CARMA-LSI 算法的推导思路,可以得到有限量测数据 CARMA 模型的最小二乘迭代辨识算法(CARMA-LSI, Least Squares based Iterative algorithm for CARMA models with finite measurement data):

$$\hat{\boldsymbol{\theta}}_k=[\hat{\boldsymbol{\Phi}}_k^{\mathrm{T}}(L)\hat{\boldsymbol{\Phi}}_k(L)]^{-1}\hat{\boldsymbol{\Phi}}_k^{\mathrm{T}}(L)\boldsymbol{Y}(L),\quad k=1,2,3,\cdots\tag{45}$$

$$\hat{\boldsymbol{\Phi}}_k(L)=[\hat{\boldsymbol{\varphi}}_k(L),\hat{\boldsymbol{\varphi}}_k(L-1),\cdots,\hat{\boldsymbol{\varphi}}_k(1)]^{\mathrm{T}},\tag{46}$$

$$\boldsymbol{Y}(L)=[y(L),y(L-1),\cdots,y(1)]^{\mathrm{T}},\tag{47}$$

$$\hat{\boldsymbol{\varphi}}_k(t)=[\,-y(t-1),-y(t-2),\cdots,-y(t-n_a),u(t-1),u(t-2),\cdots,u(t-n_b),\hat{v}_{k-1}(t-1),\hat{v}_{k-1}(t-2),\cdots,\hat{v}_{k-1}(t-n_d)\,]^{\mathrm{T}},\tag{48}$$

$$\hat{v}_k(t)=y(t)-\hat{\boldsymbol{\varphi}}_k^{\mathrm{T}}(t)\hat{\boldsymbol{\theta}}_k,\quad t=1,2,\cdots,L.\tag{49}$$

有限量测数据 CARMA-LSI 算法的计算步骤如下.

1) 收集输入输出数据$\{u(t),y(t):t=1,2,\cdots,L\}$,用式(47)构造 $\boldsymbol{Y}(L)$,给定参数估计精度 ε.

2) 令 $k=1$,置初值 $\hat{v}_0(t)=$随机数.

3) 用式(48)构造 $\hat{\boldsymbol{\varphi}}_k(t)$,用式(46)构造 $\hat{\boldsymbol{\Phi}}_k(L)$.

4) 用式(45)刷新参数估计 $\hat{\boldsymbol{\theta}}_k$.

5) 用式(49)计算 $\hat{v}_k(t)$.

6) 比较 $\hat{\boldsymbol{\theta}}_k$ 与 $\hat{\boldsymbol{\theta}}_{k-1}$:如果 $\|\hat{\boldsymbol{\theta}}_k-\hat{\boldsymbol{\theta}}_{k-1}\|\leqslant\varepsilon$,中断循环过程,获得迭代次数 k 和参数估计向量 $\hat{\boldsymbol{\theta}}_k$;否则,$k$ 增 1,转到步骤 3.

有限量测数据 CARMA-LSI 算法计算参数估计 $\hat{\boldsymbol{\theta}}_k$ 的流程如图 8 所示.

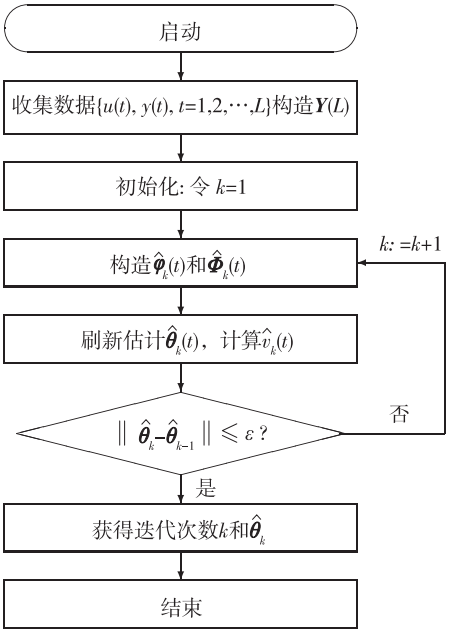

图 8 计算有限量测数据 CARMA-LSI 参数估计 $\hat{\boldsymbol{\theta}}_k$ 的流程

Fig. 8 The flowchart for computing the CARMA-LSI parameter estimate $\hat{\boldsymbol{\theta}}_k$

迭代算法充分使用了系统数据. 从上述计算步骤看,有限量测数据 CARMA-LSI 算法在每一步迭代计算参数估计的过程中,都使用了系统直到 $t=L$ 时刻所有输入输出数据$\{u(t),y(t):t=1,2,\cdots,L\}$,是离线的迭代算法. 对于 $n_d=0$ 时的方程误差模型(即 ARX 模型),就不需要迭代计算,是离线的一次完成最小二乘算法. 迭代算法一般用于信息向量含有未知项的辨识问题,如 CARAR 模型、OEMA 模型和 Box-Jenkins 模型等.

多新息辨识方法是通过扩展新息(innovation)长度,充分使用系统数据信息来提高参数估计精度的,这里的最小二乘迭代算法是通过迭代计算反复(重复)利用系统数据,提取信息来改善参数估计精度的(参见文献[8]"多新息辨识方法").

2.2.2 迭代算法的计算量问题

从 RELS 算法计算参数估计 $\hat{\boldsymbol{\theta}}(L)$ 流程图 6 与有限量测数据 CARMA-LSI 算法计算 $\hat{\boldsymbol{\theta}}_k$ 流程图 8 可以看出二者的差别. RELS 算法得到估计 $\hat{\boldsymbol{\theta}}(L)$ 的计算量等于每步递推的计算量乘以 L,CARMA-LSI 算法得到估计 $\hat{\boldsymbol{\theta}}_k$ 的计算量等于每步迭代的计算量乘以 k,k 为迭代次数. 仿真研究表明:CARMA-LSI 算法具有很高的收敛速度,通常只需迭代几次(几步),就可获得高精度的参数估计(最小二乘迭代算法都具有这一性质),精度高于 RELS 算法. 尽管迭代算法计算量有所增加,然而在某些要求高精度参数估计场合,用牺牲计算量来提高参数估计精度也是可取的,况且所增加的计算量是现代计算机完全可以胜任的.

有限量测数据 CARMA-LSI 算法是一种离线算法,不适用于在线辨识系统参数. 另外,在用 CARMA-LSI 算法计算系统的参数估计时,每一步迭代计算中都要进行数据乘积矩矩阵$[\hat{\boldsymbol{\Phi}}_k^{\mathrm{T}}(t)\hat{\boldsymbol{\Phi}}_k(t)]$的求逆运算,增加了每步计算量(总体计算量未必增加太大,因为迭代算法收敛速度快). 由于矩阵 $\hat{\boldsymbol{\Phi}}_k^{\mathrm{T}}(t)\cdot\hat{\boldsymbol{\Phi}}_k(t)\in\mathbf{R}^{(n_a+n_b+n_d)\times(n_a+n_b+n_d)}$ 和 $\hat{\boldsymbol{\Phi}}_k^{\mathrm{T}}(L)\hat{\boldsymbol{\Phi}}_k(L)\in\mathbf{R}^{(n_a+n_b+n_d)\times(n_a+n_b+n_d)}$ 的子矩阵$(1:n_a+n_b,1:n_a+n_b)$是不随 t 或 k 变化的(这里使用了 Matlab 中的冒号":"运算符),故可采用块矩阵求逆公式来减小计算量,这里从略.

2.2.3 迭代算法的收敛性

迭代辨识算法的收敛性研究是很困难的,也是辨识领域的研究难题,没有现成方法可以套用. 本文的所有最小二乘迭代算法和梯度迭代辨识算法的收敛性都是有待研究的新课题. 科学研究中难以用理论证明的问题,用仿真比较进行研究也不失为一种好办法. 仿真例子可以说明这些迭代算法比同类的递推算法精度高.

2.2.4 仿真试验

例 3 考虑 CARMA 模型仿真对象:

$$\begin{cases}A(z)y(t)=B(z)u(t)+D(z)v(t),\\ A(z)=1+a_1z^{-1}+a_2z^{-2}=1-1.60z^{-1}+0.80z^{-2},\\ B(z)=b_1z^{-1}+b_2z^{-2}=0.40z^{-1}+0.30z^{-2},\\ D(z)=1+d_1z^{-1}=1-0.64z^{-1},\\ \boldsymbol{\theta}=[a_1,a_2,b_1,b_2,d_1]^{\mathrm{T}}=\\ \quad[-1.60,0.80,0.40,0.30,-0.64]^{\mathrm{T}}.\end{cases}$$

仿真时,输入$\{u(t)\}$采用零均值单位方差不相关可测随机信号序列,$\{v(t)\}$采用零均值方差为σ^2白噪声序列. 考虑3种不同噪声水平,噪声方差分别为$\sigma^2=0.10^2$,$\sigma^2=0.50^2$和$\sigma^2=1.00^2$时,对应的输出噪信比分别为$\delta_{ns}=7.66\%$,$\delta_{ns}=38.30\%$和$\delta_{ns}=76.59\%$. 分别用RELS算法(24)—(28)和CARMA-LSI算法(45)—(49)估计这个系统的参数. 不同噪声方差和数据长度$t=L$下,RELS参数估计及其误差$\delta:=\|\hat{\boldsymbol{\theta}}(t)-\boldsymbol{\theta}\|/\|\boldsymbol{\theta}\|$如表5所示;当数据长度分别为$L=1\,000$,$L=2\,000$和$L=3\,000$时,不同噪声方差下和迭代次数下,CARMA-LSI参数估计及其误差如表7—9所示;CARMA-LSI参数估计及其误差$\delta:=\|\hat{\boldsymbol{\theta}}_k(L)-\boldsymbol{\theta}\|/\|\boldsymbol{\theta}\|$随迭代次数$k$变化曲线如图9—11所示. 表6是从表7—9中提取的迭代次数$k=10$时的CARMA-LSI参数估计及其误差.

由表5—9和图9—11,可以得到如下结论:

1) 不同噪声方差下,随着数据长度的增加,两个算法的参数估计误差都随之减小,参见表5和6;

2) 加入到系统中的噪声水平越高,相同数据长度下,参数估计收敛于真值的速度越慢,参见表5和6;

3) 在相同数据长度和噪声方差下,CARMA-LSI算法的收敛速度明显好于RELS算法,参见表5和6;

4) 对于迭代算法,由于数据长度是有限的,故其参数估计误差不会随迭代次数k的增加而趋近于零,参见图9—11;

5) 最小二乘迭代算法CARMA-LSI具有很快的收敛速度,一般只需迭代几次,参数估计会很快收敛于真参数附近,参数估计误差收敛到一定误差范围内,参见图9—11.

表5 例3的RELS估计$\hat{\boldsymbol{\theta}}(t)$

Table 5 The RELS estimates and errors

σ^2	$t=L$	a_1	a_2	b_1	b_2	d_1	δ/%
	1 000	−1.595 98	0.798 14	0.398 29	0.313 98	−0.552 52	4.515 68
0.10^2	2 000	−1.597 86	0.798 65	0.399 63	0.305 65	−0.583 51	2.892 79
	3 000	−1.597 83	0.798 37	0.400 64	0.303 24	−0.598 93	2.102 00
	1 000	−1.581 08	0.790 49	0.388 50	0.368 54	−0.513 77	7.413 66
0.50^2	2 000	−1.587 69	0.792 02	0.396 65	0.329 15	−0.560 56	4.374 61
	3 000	−1.588 04	0.791 15	0.402 26	0.317 22	−0.580 49	3.244 93
	1 000	−1.559 48	0.773 17	0.377 70	0.433 85	−0.485 17	10.767 44
1.00^2	2 000	−1.572 21	0.779 96	0.393 46	0.357 09	−0.542 03	6.038 80
	3 000	−1.575 40	0.781 30	0.404 76	0.333 17	−0.565 93	4.426 90
真值(true values)		−1.600 00	0.800 00	0.400 00	0.300 00	−0.640 00	

表6 例3的CARMA-LSI迭代估计$\hat{\boldsymbol{\theta}}(L)$($k=10$)

Table 6 The CARMA-LSI iterative estimates and errors $\hat{\boldsymbol{\theta}}(L)$ and errors ($k=10$)

σ^2	L	a_1	a_2	b_1	b_2	d_1	δ/%
	1 000	−1.597 46	0.798 77	0.398 95	0.309 81	−0.597 85	2.208 21
0.10^2	2 000	−1.600 34	0.800 72	0.399 85	0.302 10	−0.613 64	1.346 86
	3 000	−1.600 41	0.800 51	0.400 74	0.299 90	−0.620 64	0.986 59
	1 000	−1.588 59	0.792 12	0.394 74	0.348 62	−0.587 97	3.702 76
0.50^2	2 000	−1.599 51	0.797 52	0.399 30	0.311 49	−0.612 63	1.516 70
	3 000	−1.600 04	0.795 77	0.403 84	0.300 51	−0.620 25	1.046 72
	1 000	−1.581 36	0.785 63	0.389 61	0.395 58	−0.579 72	5.899 17
1.00^2	2 000	−1.596 72	0.791 03	0.398 68	0.323 81	−0.609 77	2.019 38
	3 000	−1.598 30	0.786 90	0.407 99	0.302 00	−0.618 45	1.353 11
真值(true values)		−1.600 00	0.800 00	0.400 00	0.300 00	−0.640 00	

表 7　不同方差下 CARMA-LSI 估计及其误差($L=1\ 000$)

Table 7　The CARMA-LSI estimates and errors versus iteration k ($L=1\ 000$)

σ^2	k	a_1	a_2	b_1	b_2	d_1	δ/%
0.10^2	1	-1.591 91	0.793 99	0.396 29	0.311 86	-0.001 57	32.507 14
	2	-1.597 58	0.799 08	0.398 35	0.309 79	-0.453 72	9.496 28
	5	-1.598 03	0.799 51	0.398 95	0.309 60	-0.598 04	2.194 38
	10	-1.597 46	0.798 77	0.398 95	0.309 81	-0.597 85	2.208 21
0.50^2	1	-1.49105	0.708 17	0.384 63	0.386 35	-0.002 58	33.544 74
	2	-1.590 55	0.796 18	0.392 32	0.348 03	-0.466 82	9.170 86
	5	-1.597 55	0.802 76	0.394 83	0.345 51	-0.594 94	3.275 66
	10	-1.588 59	0.792 12	0.394 74	0.348 62	-0.587 97	3.702 76
1.00^2	1	-1.364 41	0.602 20	0.374 87	0.479 47	0.001 31	37.361 28
	2	-1.584 21	0.791 71	0.385 26	0.394 83	-0.486 66	9.252 47
	5	-1.599 17	0.804 86	0.389 72	0.389 56	-0.595 07	5.133 04
	10	-1.581 36	0.785 63	0.389 61	0.395 58	-0.579 72	5.899 17
真值(true values)		-1.600 00	0.800 00	0.400 00	0.300 00	-0.640 00	

表 8　不同方差下 CARMA-LSI 估计及其误差($L=2\ 000$)

Table 8　The CARMA-LSI estimates and errors versus iteration k ($L=2\ 000$)

σ^2	k	a_1	a_2	b_1	b_2	d_1	δ/%
0.10^2	1	-1.594 52	0.795 74	0.399 00	0.304 32	-0.000 55	32.551 45
	2	-1.600 34	0.800 97	0.399 86	0.302 12	-0.449 79	9.682 57
	5	-1.600 98	0.801 59	0.399 86	0.301 84	-0.614 28	1.316 20
	10	-1.600 34	0.800 72	0.399 85	0.302 10	-0.613 64	1.346 86
0.50^2	1	-1.493 51	0.707 86	0.395 17	0.352 51	0.000 78	33.502 73
	2	-1.600 57	0.802 70	0.399 11	0.311 05	-0.471 11	8.616 39
	5	-1.611 72	0.812 78	0.399 51	0.306 82	-0.624 29	1.240 74
	10	-1.599 51	0.797 52	0.399 30	0.311 49	-0.612 63	1.516 70
1.00^2	1	-1.354 93	0.590 77	0.391 78	0.417 98	0.004 75	37.180 03
	2	-1.597 70	0.800 23	0.397 77	0.323 38	-0.500 41	7.206 19
	5	-1.624 17	0.822 59	0.399 11	0.313 49	-0.636 44	1.828 16
	10	-1.596 72	0.791 03	0.398 68	0.323 81	-0.609 77	2.019 38
真值(true values)		-1.600 00	0.800 00	0.400 00	0.300 00	-0.640 00	

表 9　不同方差下 CARMA-LSI 估计及其误差($L=3\ 000$)

Table 9　The CARMA-LSI estimates and errors versus iteration k ($L=3\ 000$)

σ^2	k	a_1	a_2	b_1	b_2	d_1	δ/%
0.10^2	1	-1.594 56	0.795 56	0.400 10	0.302 25	-0.003 10	32.421 33
	2	-1.600 43	0.800 81	0.401 04	0.299 92	-0.451 33	9.603 87
	5	-1.601 18	0.801 53	0.400 75	0.299 57	-0.621 35	0.955 52
	10	-1.600 41	0.800 51	0.400 74	0.299 90	-0.620 64	0.986 59
0.50^2	1	-1.491 15	0.704 87	0.401 15	0.344 60	-0.012 66	32.847 96
	2	-1.602 61	0.803 34	0.405 29	0.299 44	-0.479 22	8.191 45
	5	-1.614 97	0.814 84	0.403 99	0.294 43	-0.634 13	1.167 19
	10	-1.600 04	0.795 77	0.403 84	0.300 51	-0.620 25	1.046 72
1.00^2	1	-1.348 65	0.584 19	0.404 57	0.405 22	-0.020 84	36.143 67
	2	-1.607 47	0.806 01	0.410 48	0.297 94	-0.519 90	6.156 73
	5	-1.636 33	0.831 92	0.408 33	0.286 44	-0.654 89	2.699 77
	10	-1.598 30	0.786 90	0.407 99	0.302 00	-0.618 45	1.353 11
真值(true values)		-1.600 00	0.800 00	0.400 00	0.300 00	-0.640 00	

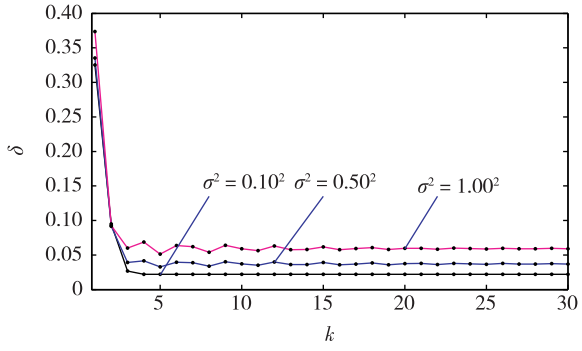

图 9　不同噪声方差下 CARMA-LSI 估计误差 δ 随 k 变化曲线($L=1\ 000$)

Fig. 9　The estimation errors δ versus k with different σ^2 ($L=1\ 000$)

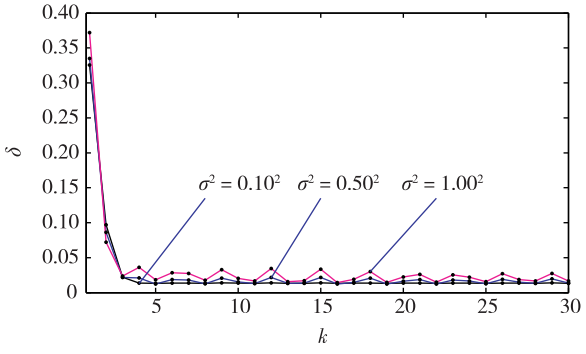

图 10　不同噪声方差下 CARMA-LSI 估计误差 δ 随 k 变化曲线($L=2\ 000$)

Fig. 10　The estimation errors δ versus k with different σ^2 ($L=2\ 000$)

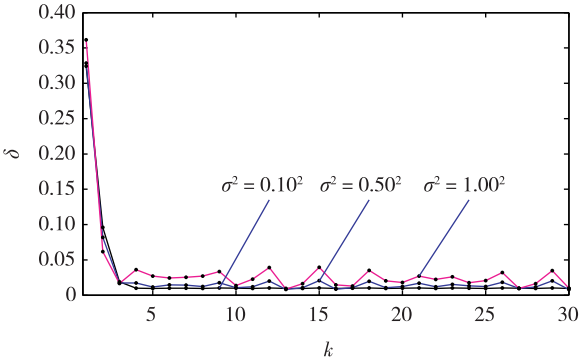

图 11　不同噪声方差下 CARMA-LSI 估计误差 δ 随 k 变化曲线($L=3\ 000$)

Fig. 11　The estimation errors δ versus k with different σ^2 ($L=3\ 000$)

2.2.5　Matlab 程序

把下列程序写到 CARMA_LSI. m 文件中,数据长度分别为 $L=1\ 000$, $L=2\ 000$ 和 $L=3\ 000$ 时,依次取噪声方差 $\sigma^2=0.10^2$, 0.50^2 和 1.00^2,运行该程序,可得到上述例子的仿真结果(参数估计表和误差曲线图).

```
1  %————————————————————————————————*
2  % Filename:CARMA_LSI.m for the RELS and CARMA-LSI algorithms *
3  % for CARMA models *
4  % A(z)y(t) = B(z)u(t) + D(z)v(t) *
5  % The data length L = 1000,2000 and 3000 *
6  % The noise variance sigma^2 = 0.10^2,0.50^2 and 1.00^2 *
7  %————————————————————————————————*
8  clear;format short g
9  Method = 'The RELS and CARMA-LSI algorithms for CARMA models'
10 PlotLength = 30;
11 L = 1000;% The data legth L = 1000,2000,3000 for CARMA-LSI
12 length1 = 3100;
13 sigma = .1;% The noise variance sigma = 0.10,0.50 and 1.00
14
15 na = 2;nb = 2;nd = 1;n = na + nb + nd;
16 a = [1, -1.6,0.8];b = [0,0.4,0.3];d = [1, -0.64];
17 par0 = [a(2:na + 1),b(2:nb + 1),d(2:nd + 1)]';
18 p0 = 1e6;P = eye(n) * p0;par1 = ones(n,1);
19 %——Compute the noise-to-signal ratio
20 sy = f_integral(a,b);sv = f_integral(a,d);
21 [sy sv];
22 delta_ns = sqrt(sv/sy) * 100 * sigma;
23 fprintf('$\\sigma^2 = %5.2f^2$, $\\delta_{\\ns} = %6.2f%s\n',sigma,delta_ns,'\%');
24
25 %——Generate the input-output data
26 rand('state',10);u = (rand(length1,1) - 0.5) * sqrt(12);% The input
27 randn('state',8);v = randn(length1,1) * sigma;% The noise
28 y = ones(length1,1)/p0;
29 for t = n:length1
30     y(t) = par0' * [-y(t-1:-1:t-na);u(t-1:-1:t-nb);v(t-1:-1:t-nd)] + v(t);
31 end
32 % Gz = tf(b,a,1);Gn = tf(d,a,1);% y = lsim(Gz,u) + lsim(Gn,v);
33 %——Set the initial values
34 randn('state',0);v0 = randn(length1,1);v1 = zeros
```

```
(length1,1);
% ——The RELS algorithm for CARMA systems
t0=30;jj=0;j1=0;
for t=t0:length1
    jj=jj+1;
    varphi=[-y(t-1:-1:t-na);u(t-1:-1:t-nb);v1(t-1:-1:t-nd)];
    P=P-P*varphi*varphi'*P/(1+varphi'*P*varphi);
    par1=par1+P*varphi*(y(t)-varphi'*par1);
    delta=norm(par1-par0)/norm(par0);
    v1(t)=y(t)-varphi'*par1;
    ls(jj,:)=[jj,par1',delta];
    if (jj==100)|(jj==200)|(jj==500)|mod(jj,1000)==0
        j1=j1+1;
        ls100(j1,:)=[jj,par1',delta*100];
    end
    if jj==3000
        break
    end
end
ls100(j1+1,:)=[0,par0',0];
fprintf('The RELS estimates')
fprintf('\n t a_1 a_2 b_1 b_2')
fprintf('%s\n','d_1 \delta\(\%)\\\\');
fprintf('%5d %10.5f %10.5f %10.5f %10.5f %10.5f %10.5f\\\\\n',ls100');

figure(1);k=(t0:3000)';
plot(ls(k,1),ls(k,n+2));
xlabel('\it t');ylabel('{\it\delta}');
if sigma==0.1
    dat1=[ls(:,1),ls(:,n+2)];
    save dat1 dat1
elseif sigma==0.5
    load dat1
    dat2=[dat1,ls(:,n+2)];
    save dat2 dat2
else
    load dat2
    z0=[dat2,ls(:,n+2)];
    k=(t0:5:3000)';jk=z0(k);
    figure(2);
    plot(jk,z0(k,2),'k',jk,z0(k,3),'b',jk,z0(k,4),'m')
    xlabel('\it t');ylabel('{\it\delta}');
end
% ——The CARMA-LSI algorithm
jj=0;Y=y(t0:t0+L-1);
for k=1:PlotLength
    jj=jj+1;j1=0;
    for t=n+1:length1
        varphi2=[-y(t-1:-1:t-na);u(t-1:-1:t-nb);v0(t-1:-1:t-nd)];
        if t>=t0 t<=t0+L-1
            j1=j1+1;
            Phi(j1,:)=varphi2';
        end
    end
    par2=Phi\Y;
    for t=n+1:length1
        varphi2=[-y(t-1:-1:t-na);u(t-1:-1:t-nb);v0(t-1:-1:t-nd)];
        v0(t)=y(t)-varphi2'*par2;
    end
    delta=norm(par2-par0)/norm(par0);
    ls2(jj,:)=[jj,par2',delta];
    ls200(jj,:)=[jj,par2',delta*100];
    end
ls200(jj+1,:)=[0,par0',0];
fprintf('The CARMA-LSI estimates with the data length L=%d\n',L)
fprintf('\n %s','k a_1 a_2 b_1')
fprintf('%s\n','b_2 d_1 \delta\(\%)\\\\\hline');
fprintf('%4d %10.5f %10.5f %10.5f %10.5f %10.5f %10.5f\\\\\n',ls200');

figure(3);plot(ls2(:,1),ls2(:,n+2));
xlabel('\it t');ylabel('{\it\delta}');
if sigma==0.1
    data1=[ls2(:,1),ls2(:,n+2)];
    save data1 data1
elseif sigma==0.5
    load data1
    data2=[data1,ls2(:,n+2)];
    save data2 data2
else
    load data2
    z0=[data2,ls2(:,n+2)];
    figure(4);k=(1:PlotLength);jk=z0(k,1);
    plot(jk,z0(k,2),'k',jk,z0(k,3),'b',jk,z0(k,4),'m',
        jk,z0(k,2),'k.',jk,z0(k,3),'k.',jk,z0(k,4),'k.')
    axis([0.8,PlotLength,0,0.4])
    xlabel('\it k');ylabel('{\it\delta}');
    if L==1000
```

```
121        text(8,0.15,'{\it\sigma}^2=0.10^2')
122        text(15,0.15,'{\it\sigma}^2=0.50^2')
123        text(22,0.15,'{\it\sigma}^2=1.00^2')
124        line([5,8],[0.021,0.135]);
125        line([12,15],[0.039,0.135]);
126        line([20,22],[0.059,0.135]);
127     elseif L==2000
128        text(8,0.15,'{\it\sigma}^2=0.10^2')
129        text(15,0.15,'{\it\sigma}^2=0.50^2')
130        text(22,0.15,'{\it\sigma}^2=1.00^2')
131        line([4,8],[0.013,0.135]);
132        line([12,15],[0.021,0.135]);
133        line([18,22],[0.030,0.135]);
134     elseif L==3000
135        text(12,0.15,'{\it\sigma}^2=0.10^2')
136        text(18,0.15,'{\it\sigma}^2=0.50^2')
137        text(24,0.15,'{\it\sigma}^2=1.00^2')
138        line([9,12],[0.01,0.135]);
139        line([15,18],[0.020,0.135]);
140        line([21,24],[0.027,0.135]);
141     end
142  end
```

2.3 梯度迭代辨识方法

令 $k=1,2,3,\cdots$ 是一个迭代变量,$\hat{\boldsymbol{\theta}}_k(t)$ 为 $\boldsymbol{\theta}$ 的迭代估计,$\lambda_{\max}[\boldsymbol{X}]$ 为对称矩阵 $\boldsymbol{X}$ 的最大特征值.

对于优化问题 J_4,使用负梯度搜索算法(16)可得迭代算法:

$$\hat{\boldsymbol{\theta}}_k(t)=\boldsymbol{\theta}_{k-1}(t)-\frac{\mu_k(t)}{2}\text{grad}[J_4(\hat{\boldsymbol{\theta}}_{k-1}(t))]=$$
$$\hat{\boldsymbol{\theta}}_{k-1}(t)+\mu_k(t)\boldsymbol{\Phi}^{\mathrm{T}}(t)[\boldsymbol{Y}(t)-\boldsymbol{\Phi}(t)\hat{\boldsymbol{\theta}}_{k-1}(t)],\quad(50)$$

式(50)中 $\mu_k(t)$ 为迭代步长(iterative step-size)或收敛因子(convergence factor). 辨识的困难是 $\boldsymbol{\Phi}(t)$(也就是 $\boldsymbol{\varphi}(t)$)中包含了未知变量 $v(t-i)$,故这个梯度算法无法计算估计 $\hat{\boldsymbol{\theta}}_k(t)$. 解决方案是采用递阶辨识原理:这些未知变量用其第 k 次迭代估计值代替,得到

$$\hat{\boldsymbol{\theta}}_k(t)=\hat{\boldsymbol{\theta}}_{k-1}(t)+\mu_k(t)\hat{\boldsymbol{\Phi}}_k^{\mathrm{T}}(t)[\boldsymbol{Y}(t)-\hat{\boldsymbol{\Phi}}_k(t)\hat{\boldsymbol{\theta}}_{k-1}(t)],$$

或

$$\hat{\boldsymbol{\theta}}_k(t)=[\boldsymbol{I}-\mu_k(t)\hat{\boldsymbol{\Phi}}_k^{\mathrm{T}}(t)\hat{\boldsymbol{\Phi}}_k(t)]\hat{\boldsymbol{\theta}}_{k-1}(t)+$$
$$\mu_k(t)\hat{\boldsymbol{\Phi}}_k^{\mathrm{T}}(t)\boldsymbol{Y}(t).$$

上式可以看作一个离散时间系统(即时标 k 的差分方程,t 看作常量). 为保证 $\hat{\boldsymbol{\theta}}_k(t)$ 的收敛性,矩阵 $[\boldsymbol{I}-\mu_k(t)\hat{\boldsymbol{\Phi}}_k^{\mathrm{T}}(t)\hat{\boldsymbol{\Phi}}_k(t)]$ 的所有特征值必须在单位圆内,因此 $\mu_k(t)$ 的一个保守选择是满足

$$0<\mu_k(t)\leqslant\frac{2}{\lambda_{\max}[\hat{\boldsymbol{\Phi}}_k^{\mathrm{T}}(t)\hat{\boldsymbol{\Phi}}_k(t)]}.\quad(51)$$

由此可得 CARMA 模型的梯度迭代辨识算法(CARMA-GI, Gradient based Iterative identification algorithm for CARMA models)(CARMA-GI 算法):

$$\hat{\boldsymbol{\theta}}_k(t)=\hat{\boldsymbol{\theta}}_{k-1}(t)+\mu_k(t)\hat{\boldsymbol{\Phi}}_k^{\mathrm{T}}(t)[\boldsymbol{Y}(t)-\hat{\boldsymbol{\Phi}}_k(t)\hat{\boldsymbol{\theta}}_{k-1}(t)],\quad k=1,2,3,\cdots\quad(52)$$

$$\hat{\boldsymbol{\Phi}}_k(t)=[\hat{\boldsymbol{\varphi}}_k(t),\hat{\boldsymbol{\varphi}}_k(t-1),\cdots,\hat{\boldsymbol{\varphi}}_k(t-p+1)]^{\mathrm{T}},\quad(53)$$

$$\boldsymbol{Y}(t)=[y(t),y(t-1),\cdots,y(t-p+1)]^{\mathrm{T}},\quad(54)$$

$$\hat{\boldsymbol{\varphi}}_k(t)=[-y(t-1),-y(t-2),\cdots,$$
$$-y(t-n_a),u(t-1),u(t-2),\cdots,u(t-n_b),\hat{v}_{k-1}(t-1),$$
$$\hat{v}_{k-1}(t-2),\cdots,\hat{v}_{k-1}(t-n_d)]^{\mathrm{T}},\quad(55)$$

$$\hat{v}_k(t-i)=y(t-i)-\hat{\boldsymbol{\varphi}}_k^{\mathrm{T}}(t-i)\hat{\boldsymbol{\theta}}_k(t),$$
$$i=1,2,\cdots,n_d,\quad(56)$$

$$0<\mu_k(t)\leqslant\frac{2}{\lambda_{\max}[\hat{\boldsymbol{\Phi}}_k^{\mathrm{T}}(t)\hat{\boldsymbol{\Phi}}_k(t)]}.\quad(57)$$

多新息随机梯度辨识方法通过扩展新息(innovation)长度、充分使用系统数据信息(系统的新息)来提高参数估计精度,这里的梯度迭代算法也是反复利用系统数据来改善参数估计精度的. 这个梯度迭代算法中的 p 也可看作新息长度(参见文献[8]"多新息辨识方法").

CARMA-GI 算法的计算步骤如下.

1) 确定 p,给定参数估计精度 ε,置 $t=1$,$\hat{\boldsymbol{\theta}}_0(t)=\boldsymbol{1}_n/p_0$,$p_0=10^6$.

2) 收集输入输出数据 $u(t)$ 和 $y(t)$,用式(54)构造 $\boldsymbol{Y}(t)$.

3) 令 $k=1$,置初值 $\hat{v}_0(t-i)=1/p_0$.

4) 用式(55)构造 $\hat{\boldsymbol{\varphi}}_k(t)$,用式(53)构造 $\hat{\boldsymbol{\Phi}}_k(t)$.

5) 根据式(57)选择一个大 $\mu_k(t)$,用式(52)刷新参数估计 $\hat{\boldsymbol{\theta}}_k(t)$.

6) 用式(56)计算 $\hat{v}_k(t-i)$.

7) 比较 $\hat{\boldsymbol{\theta}}_k(t)$ 与 $\hat{\boldsymbol{\theta}}_{k-1}(t)$:如果 $\|\hat{\boldsymbol{\theta}}_k(t)-\hat{\boldsymbol{\theta}}_{k-1}(t)\|>\varepsilon$,$k$ 增 1,转到步骤 4;否则,获得迭代次数 k 和参数估计向量 $\hat{\boldsymbol{\theta}}_k(t)$,令 $\hat{\boldsymbol{\theta}}_0(t+1):=\hat{\boldsymbol{\theta}}_k(t)$,$t$ 增 1,转到步骤 2.

2.3.1 有限量测数据 CARMA-GI 算法

这里不加推导地给出有限量测数据 CARMA 模型的梯度迭代辨识算法(CARMA-GI, Gradient based Iterative identification algorithm for CARMA models with finite measurement data):

$$\hat{\boldsymbol{\theta}}_k=\hat{\boldsymbol{\theta}}_{k-1}+\mu_k\hat{\boldsymbol{\Phi}}_k^{\mathrm{T}}(L)[\boldsymbol{Y}(L)-\hat{\boldsymbol{\Phi}}_k(L)\hat{\boldsymbol{\theta}}_{k-1}],$$
$$k=1,2,3,\cdots\quad(58)$$

$$\hat{\boldsymbol{\Phi}}_k(L)=[\hat{\boldsymbol{\varphi}}_k(L),\hat{\boldsymbol{\varphi}}_k(L-1),\cdots,\hat{\boldsymbol{\varphi}}_k(1)]^{\mathrm{T}},\quad(59)$$

$$\boldsymbol{Y}(L)=[y(L),y(L-1),\cdots,y(1)]^{\mathrm{T}},\quad(60)$$

$$\hat{\boldsymbol{\varphi}}_k(t)=[-y(t-1),-y(t-2),\cdots,-y(t-n_a),u(t-1),u(t-2),\cdots,u(t-n_b),\hat{v}_{k-1}(t-1),\hat{v}_{k-1}(t-2),\cdots,\hat{v}_{k-1}(t-n_d)]^{\mathrm{T}},\quad(61)$$

$$\hat{v}_k(t)=y(t)-\hat{\boldsymbol{\varphi}}_k^{\mathrm{T}}(t)\hat{\boldsymbol{\theta}}_k,\quad t=1,2,\cdots,L,\quad(62)$$

$$0<\mu_k\leqslant\frac{2}{\lambda_{\max}[\hat{\boldsymbol{\Phi}}_k^{\mathrm{T}}(L)\hat{\boldsymbol{\Phi}}_k(L)]}.\quad(63)$$

为便于说明 CARMA-GI 算法的参数精度高,下面给出可比较的估计模型(23)参数向量 $\boldsymbol{\theta}$ 的同类增广随机梯度算法(ESG):

$$\hat{\boldsymbol{\theta}}(t)=\hat{\boldsymbol{\theta}}(t-1)+\frac{\hat{\boldsymbol{\phi}}(t)}{r(t)}[y(t)-\hat{\boldsymbol{\phi}}^{\mathrm{T}}(t)\hat{\boldsymbol{\theta}}(t-1)],\quad(64)$$

$$r(t)=r(t-1)+\|\hat{\boldsymbol{\phi}}(t)\|^2,r(0)=1,\quad(65)$$

$$\hat{\boldsymbol{\phi}}(t)=[-y(t-1),-y(t-2),\cdots,-y(t-n_a),u(t-1),u(t-2),\cdots,u(t-n_b),\hat{v}(t-1),\hat{v}(t-2),\cdots,\hat{v}(t-n_d)]^{\mathrm{T}},\quad(66)$$

$$\hat{v}(t)=y(t)-\hat{\boldsymbol{\phi}}^{\mathrm{T}}(t)\hat{\boldsymbol{\theta}}(t),\quad(67)$$

$$\hat{\boldsymbol{\theta}}(t)=[\hat{a}_1(t),\hat{a}_2(t),\cdots,\hat{a}_{n_a}(t),\hat{b}_1(t),\hat{b}_2(t),\cdots,\hat{b}_{n_b}(t),\hat{d}_1(t),\hat{d}_2(t),\cdots,\hat{d}_{n_d}(t)]^{\mathrm{T}}.\quad(68)$$

2.3.2 仿真试验

例 4 考虑 CARMA 模型仿真对象:

$$\begin{cases}A(z)y(t)=B(z)u(t)+D(z)v(t),\\ A(z)=1+a_1z^{-1}+a_2z^{-2}=1-1.60z^{-1}+0.80z^{-2},\\ B(z)=b_1z^{-1}+b_2z^{-2}=0.40z^{-1}+0.30z^{-2},\\ D(z)=1+d_1z^{-1}=1-0.64z^{-1},\\ \boldsymbol{\theta}=[a_1,a_2,b_1,b_2,d_1]^{\mathrm{T}}=\\ \quad[-1.60,0.80,0.40,0.30,-0.64]^{\mathrm{T}}.\end{cases}$$

仿真时,输入$\{u(t)\}$采用零均值单位方差不相关可测随机信号序列,$\{v(t)\}$采用零均值方差为 σ^2 白噪声序列. 考虑 3 种不同噪声水平,噪声方差分别为 $\sigma^2=0.10^2$,$\sigma^2=0.50^2$ 和 $\sigma^2=1.00^2$ 时,对应的输出噪信比分别为 $\delta_{\mathrm{ns}}=7.66\%$,$\delta_{\mathrm{ns}}=38.30\%$ 和 $\delta_{\mathrm{ns}}=76.59\%$. 分别用 ESG 算法(64)—(68)和 CARMA-GI 算法(58)—(63)估计这个系统的参数. 不同噪声方差和数据长度 $t=L$ 下,ESG 参数估计及其误差 $\delta:=\|\hat{\boldsymbol{\theta}}(t)-\boldsymbol{\theta}\|/\|\boldsymbol{\theta}\|$ 如表 10 所示;当数据长度分别为 $L=1\,000$,$L=2\,000$ 和 $L=3\,000$ 时,不同噪声方差下和迭代次数下,CARMA-GI 参数估计及其误差如表 12—14 所示;CARMA-GI 参数估计及其误差 $\delta:=\|\hat{\boldsymbol{\theta}}_k(L)-\boldsymbol{\theta}\|/\|\boldsymbol{\theta}\|$ 随迭代次数 k 变化曲线如图 12—14 所示. 其中取

$$\mu_k=\frac{1}{\lambda_{\max}[\hat{\boldsymbol{\Phi}}_k^{\mathrm{T}}(L)\hat{\boldsymbol{\Phi}}_k(L)]}.$$

表 11 是从表 12—14 中提取的迭代次数 $k=500$ 时的 CARMA-GI 参数估计及其误差.

由表 10—14 和图 12—14,可知 CARMA-GI 算法估计精度高于 ESG 算法,但收敛速度比 CARMA-LSI 慢,CARMA-GI 算法需要迭代大约 500 次,CARMA-LSI 算法大约需要几次. 对有色噪声模型,噪声方差太小,不利于噪声模型参数的辨识(参见表 11 中$\sigma^2=0.10^2$ 时的参数估计及其误差);若噪声方差为零,则噪声模型参数不可辨识.

2.3.3 Matlab 程序

把下列程序写到 CARMA-GI.m 文件中,数据长度分别为 $L=1\,000$,$L=2\,000$ 和 $L=3\,000$ 时,依次取噪声方差 $\sigma^2=0.10^2,0.50^2$ 和 1.00^2,运行该程序,可得到例 4 的仿真结果(参数估计表和误差曲线图).

表 10 例 4 的 ESG 估计 $\hat{\boldsymbol{\theta}}(t)$

Table 10 The ESG estimates and errors

σ^2	$t=L$	a_1	a_2	b_1	b_2	d_1	δ/%
0.10^2	1 000	−0.690 05	−0.062 17	0.035 71	0.364 45	0.328 71	82.808 19
	2 000	−0.707 23	−0.044 48	0.051 46	0.371 63	0.347 97	82.278 82
	3 000	−0.718 74	−0.034 26	0.061 16	0.376 47	0.359 83	81.964 33
0.50^2	1 000	−0.541 75	−0.163 98	0.046 96	0.244 54	0.489 06	94.567 53
	2 000	−0.559 42	−0.147 97	0.060 10	0.254 97	0.509 74	94.145 76
	3 000	−0.571 41	−0.138 61	0.068 71	0.261 90	0.522 58	93.882 18
1.00^2	1 000	−0.493 99	−0.206 85	0.082 34	−0.022 02	0.441 66	96.734 01
	2 000	−0.508 20	−0.187 03	0.090 77	−0.006 54	0.467 21	96.333 29
	3 000	−0.518 79	−0.175 61	0.096 95	0.003 41	0.483 26	96.072 08
真值(true values)		−1.600 00	0.800 00	0.400 00	0.300 00	−0.640 00	

表 11　例 4 的 CARMA-GI 迭代估计 $\hat{\boldsymbol{\theta}}(L)$ ($k=500$)

Table 11　The CARMA-GI iterative estimates and errors $\hat{\boldsymbol{\theta}}(L)$ and errors ($k=500$)

σ^2	L	a_1	a_2	b_1	b_2	d_1	$\delta(\%)$
	1 000	−1.589 86	0.792 13	0.395 72	0.312 69	0.130 19	39.214 80
0.10^2	2 000	−1.592 46	0.793 87	0.398 76	0.305 16	0.130 08	39.202 36
	3 000	−1.592 30	0.793 48	0.399 79	0.303 21	0.138 36	39.623 07
	1 000	−1.588 05	0.793 96	0.394 69	0.348 80	−0.586 31	3.765 11
0.50^2	2 000	−1.599 63	0.801 83	0.399 24	0.311 29	−0.611 39	1.568 96
	3 000	−1.600 31	0.801 59	0.403 70	0.300 16	−0.618 94	1.091 68
	1 000	−1.580 64	0.788 51	0.389 65	0.395 95	−0.578 92	5.925 40
1.00^2	2 000	−1.596 74	0.799 09	0.398 62	0.323 63	−0.609 39	1.977 01
	3 000	−1.598 28	0.799 41	0.407 47	0.301 24	−0.618 08	1.184 08
真值(true values)		−1.600 00	0.800 00	0.400 00	0.300 00	−0.640 00	

表 12　不同方差下 CARMA-GI 估计及其误差($L=1\,000$)

Table 12　The CARMA-GI estimates and errors versus iteration k ($L=1\,000$)

σ^2	k	a_1	a_2	b_1	b_2	d_1	$\delta/\%$
	1	−0.473 15	−0.330 04	0.030 22	0.077 79	0.003 96	90.306 12
	10	−0.808 70	0.070 20	0.234 44	0.360 39	0.549 35	82.145 05
0.10^2	100	−1.543 45	0.749 39	0.396 31	0.339 97	0.557 53	61.112 42
	200	−1.583 86	0.786 67	0.395 04	0.315 32	0.405 09	53.213 43
	500	−1.589 86	0.792 13	0.395 72	0.312 69	0.130 19	39.214 80
	1	−0.468 88	−0.319 10	0.025 26	0.067 20	0.004 84	90.229 98
	10	−0.796 90	0.077 47	0.203 59	0.334 32	0.541 52	82.119 64
0.50^2	100	−1.476 60	0.691 63	0.387 53	0.406 17	−0.096 03	29.430 54
	200	−1.565 56	0.773 17	0.392 20	0.360 16	−0.448 75	10.451 76
	500	−1.588 05	0.793 96	0.394 69	0.348 80	−0.586 31	3.765 11
	1	−0.462 40	−0.304 31	0.017 70	0.048 97	0.004 72	90.175 28
	10	−0.768 73	0.075 55	0.151 85	0.271 40	0.510 63	82.110 24
1.00^2	100	−1.449 57	0.669 27	0.381 15	0.474 02	−0.292 27	22.261 59
	200	−1.569 56	0.778 13	0.388 90	0.404 84	−0.553 34	7.203 90
	500	−1.580 64	0.788 51	0.389 65	0.395 95	−0.578 92	5.925 40
真值(true values)		−1.600 00	0.800 00	0.400 00	0.300 00	−0.640 00	

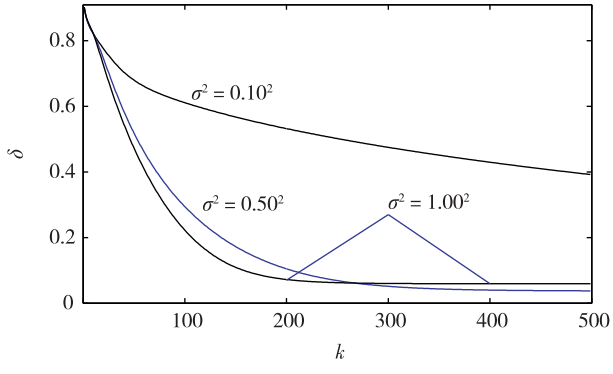

图 12　不同噪声方差下 CARMA-GI 估计误差 δ 随 k 变化曲线($L=1\,000$)

Fig. 12　The estimation errors δ versus k with different σ^2 ($L=1\,000$)

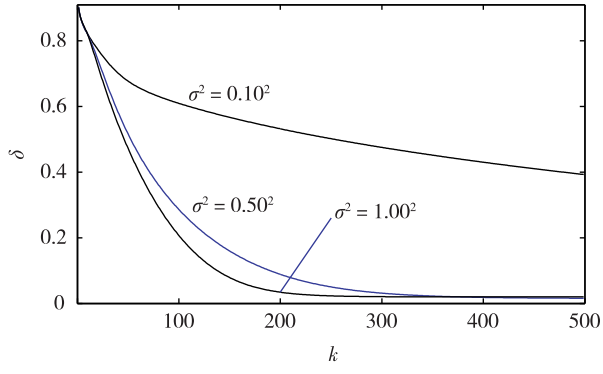

图 13　不同噪声方差下 CARMA-GI 估计误差 δ 随 k 变化曲线($L=2\,000$)

Fig. 13　The estimation errors δ versus t with different σ^2 ($L=2\,000$)

表 13　不同方差下 CARMA-GI 估计及其误差($L=2\ 000$)

Table 13　The CARMA-GI estimates and errors versus iteration k ($L=2\ 000$)

σ^2	k	a_1	a_2	b_1	b_2	d_1	δ/%
	1	-0.472 54	-0.332 01	0.031 33	0.079 21	0.001 69	90.327 29
	10	-0.804 95	0.064 99	0.232 96	0.36239	0.54231	82.107 45
0.10^2	100	-1.544 57	0.749 84	0.398 66	0.33234	0.55447	60.941 12
	200	-1.586 33	0.788 31	0.398 44	0.30787	0.40451	53.176 16
	500	-1.592 46	0.793 87	0.398 76	0.305 16	0.130 08	39.202 36
	1	-0.468 87	-0.321 98	0.026 52	0.068 40	0.002 13	90.251 20
	10	-0.792 21	0.068 98	0.209 35	0.331 14	0.531 48	82.022 21
0.50^2	100	-1.479 92	0.691 78	0.395 73	0.371 89	-0.106 84	28.594 40
	200	-1.576 08	0.779 99	0.398 52	0.322 98	-0.470 76	8.837 67
	500	-1.599 63	0.801 83	0.399 24	0.311 29	-0.611 39	1.568 96
	1	-0.462 14	-0.305 98	0.018 77	0.049 64	0.002 15	90.172 68
	10	-0.762 40	0.066 81	0.162 97	0.264 63	0.499 89	82.011 54
1.00^2	100	-1.450 28	0.665 82	0.393 66	0.409 47	-0.304 30	20.684 99
	200	-1.585 45	0.788 44	0.398 48	0.332 40	-0.584 57	3.403 25
	500	-1.596 74	0.799 09	0.398 62	0.323 63	-0.609 39	1.977 01
真值(true values)		-1.600 00	0.800 00	0.400 00	0.300 00	-0.640 00	

表 13　不同方差下 CARMA-GI 估计及其误差($L=3\ 000$)

Table 13　CARMA-GI estimates and errors versus iteration k ($L=3\ 000$)

σ^2	k	a_1	a_2	b_1	b_2	d_1	δ/%
	1	-0.473 46	-0.336 24	0.034 41	0.080 42	-0.001 05	90.344 62
	10	-0.798 96	0.056 01	0.234 00	0.362 18	0.536 21	82.234 87
0.10^2	100	-1.539 93	0.745 43	0.399 13	0.333 91	0.557 87	61.137 26
	200	-1.585 79	0.787 56	0.399 24	0.306 25	0.410 85	53.499 33
	500	-1.592 30	0.793 48	0.399 79	0.303 21	0.138 36	39.623 07
	1	-0.469 84	-0.326 32	0.029 95	0.070 22	0.000 27	90.276 81
	10	-0.786 59	0.059 51	0.214 94	0.331 06	0.525 71	82.139 84
0.50^2	100	-1.471 46	0.683 25	0.401 46	0.368 21	-0.094 35	29.353 11
	200	-1.574 07	0.777 27	0.403 98	0.313 74	-0.466 46	9.035 50
	500	-1.600 31	0.801 59	0.403 70	0.300 16	-0.618 94	1.091 68
	1	-0.462 87	-0.309 35	0.02192	0.051 60	0.001 26	90.191 30
	10	-0.758 14	0.059 10	0.172 20	0.264 52	0.496 30	82.107 29
1.00^2	100	-1.438 32	0.654 31	0.405 80	0.399 33	-0.292 15	21.491 39
	200	-1.584 53	0.786 47	0.408 19	0.312 08	-0.588 37	2.924 51
	500	-1.598 28	0.799 41	0.407 47	0.301 24	-0.618 08	1.184 08
真值(true values)		-1.600 00	0.800 00	0.400 00	0.300 00	-0.640 00	

```
1  %--------------------------------------------------*
2  % Filename:CARMA_GI.m for the ESG and CARMA-GI algorithms for *
3  % the CARMA models *
4  % A(z)y(t) = B(z)u(t) + D(z)v(t) *
5  % The data length L = 1000,2000 and 3000 *
6  % The noise variance sigma^2 = 0.10^2,0.50^2 and 1.00^2 *
7  %--------------------------------------------------*
8  clear;format short g
9  Method = 'The ESG and CARMA-GI algorithms for CARMA models'
10 PlotLength = 500;
11 L = 1000;% L = 1000,2000 and 3000
12 sigma = .1;% The noise variance sigma = 0.10,0.50 and
```

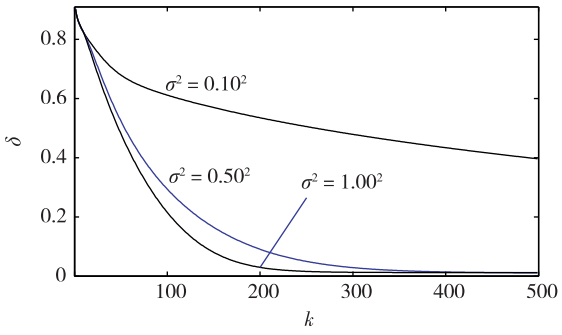

图 14　不同噪声方差下 CARMA-GI 估计误差 δ 随 k 变化曲线($L=3\ 000$)

Fig. 14　The estimation errors δ versus k with different σ^2 ($L=3\ 000$)

```
    1.00
13  length1 =3100;
14
15  na =2;nb =2;nd =1;n = na + nb + nd;
16  a =[1, -1.6,0.8];b =[0,0.4,0.3];d =[1, -0.64];
17  par0 =[a(2:na +1),b(2:nb +1),d(2:nd +1)]';
18  p0 =1e6;r =1;
19  par1 =ones(n,1)/p0;par2 =ones(n,1)/p0;
20  %——Compute the noise - to - signal ratio
21  sy =f_integral(a,b);sv =f_integral(a,d);
22  [sy sv];
23  delta_ns =sqrt(sv/sy) * 100 * sigma;
24  fprintf('\\sigma^2 =%5.2f^2,\\delta_{\\ns} =%6.2f%
    s\n',…
25      sigma,delta_ns,'\%');
26  %——Generate the input - output data
27  rand('state',10);u =(rand(length1,1) -0.5) * sqrt
    (12);% The input
28  randn('state',8);v =randn(length1,1) * sigma;% The
    noise
29  y =ones(length1,1)/p0;
30  randn('state',0);v0 =randn(length1,1);v1 =zeros
    (length1,1);
31  for t =n:length1
32      y(t) =par0' * [ -y(t-1: -1:t-na);u(t-1: -1:t
    -nb);v(t-1: -1:t-nd)] +v(t);
33  end
34  %Gz =tf(b,a,1);Gn =tf(d,a,1);% y =lsim(Gz,u) +
    lsim(Gn,v);
35  %——The ESG algorithm for CARMA systems
36  t0 =30;jj =0;j1 =0;
37  for t =t0:length1
38      jj =jj +1;
39      varphi =[ -y(t-1: -1:t-na);u(t-1: -1:t-
    nb);v1(t-1: -1:t-nd)];
40      r =r + varphi' * varphi;
41      par1 =par1 + varphi * (y(t) - varphi' * par1)/r;
42      delta =norm(par1 - par0)/norm(par0);
43      v1(t) =y(t) - varphi' * par1;
44      ls(jj,:) =[jj,par1',delta];
45      if (jj ==100) | (jj ==200) | (jj ==500) | mod(jj,
    1000) ==0
46          j1 =j1 +1;
47          ls100(j1,:) =[jj,par1',delta * 100];
48      end
49      if jj ==3000
50          break
51      end
52  end
53  ls100(j1 +1,:) =[0,par0',0];
54  fprintf('The ESG estimates')
55  fprintf('\n t a_1 a_2 b_1 b_2')
56  fprintf('%s\n','d_1 δ(%) \\');
57  fprintf('%5d %10.5f %10.5f %10.5f %10.5f %10.5f
    %10.5f\\\\\n',ls100');
58
59  figure(1);k =(t0:3000)';
60  plot(ls(k,1),ls(k,n +2));
61  xlabel('\it t');ylabel('{\it\delta}');
62  if sigma ==0.1
63      dat1 =[ls(:,1),ls(:,n +2)];
64      save dat1 dat1
65  elseif sigma ==0.5
66      load dat1
67      dat2 =[dat1,ls(:,n +2)];
68      save dat2 dat2
69  else
70      load dat2
71      z0 =[dat2,ls(:,n +2)];
72      k =(t0:5:3000)';jk =z0(k);
73      figure(2);
74      plot(jk,z0(k,2),'k',jk,z0(k,3),'b',jk,z0(k,4),
    'm')
75      xlabel('\it t');ylabel('{\it\delta}');
76  end
77  %——The CARMA - GI algorithm
78  jj =0;j2 =0;
79  Y =y(t0:t0 + L -1);
80  for k =1:PlotLength
81      jj =jj +1;j1 =0;
82      for t =n +1:length1
```

```
        varphi2 = [ -y(t-1:-1:t-na);u(t-1:-1:t-nb);v0(t-1:-1:t-nd)];
        if t >= t0 t <= t0 + L - 1
            j1 = j1 + 1;
            Phi(j1,:) = varphi2';
        end
        end
        par2 = par2 + Phi' * (Y - Phi * par2) * 1.0/max(eig(Phi' * Phi));
        for t = n + 1:length1
            varphi2 = [ -y(t-1:-1:t-na);u(t-1:-1:t-nb);v0(t-1:-1:t-nd)];
            v0(t) = y(t) - varphi2' * par2;
        end
        delta = norm(par2 - par0)/norm(par0);
        ls2(jj,:) = [jj,par2',delta];
        if jj == 1|jj == 2|jj == 5|jj == 10|jj == 20|jj == 50|...
            jj == 100|jj == 200|jj == 500
            j2 = j2 + 1;
            ls200(j2,:) = [jj,par2',delta * 100];
        end
    end
    ls200(j2 + 1,:) = [0,par0',0];

    fprintf('The CARMA - GI algorithm with L = %d',L);
    fprintf('\n %s','k a_1 a_2 b_1 ')
    fprintf(' %s\n','b_2 d_1 \delta\ (\%)\ \ \\\hline');
    fprintf('%5d %10.5f %10.5f %10.5f %10.5f %10.5f %10.5f\\\\\n',ls200');

    figure(3);
    plot(ls2(:,1),ls2(:,n + 2));
    xlabel('\it k');ylabel('{\it\delta}');

    if sigma == 0.1
        data1 = [ls2(:,1),ls2(:,n + 2)];
        save data1 data1
    elseif sigma == 0.5
        load data1
        data2 = [data1,ls2(:,n + 2)];
        save data2 data2
    else % sigma = 1
        load data2
        z0 = [data2,ls2(:,n + 2)];
        figure(4);k = (1:PlotLength - 1);jk = z0(k,1);
        plot(jk,z0(k,2),'k',jk,z0(k,3),'b',jk,z0(k,4),'k')
        axis([0,PlotLength,0,0.91])
        xlabel('\it k');ylabel('{\it\delta}');
        if L == 1000
            text(110,0.63,'{\it\sigma}^2 = 0.10^2')
            text(110,0.3,'{\it\sigma}^2 = 0.50^2')
            text(300,0.3,'{\it\sigma}^2 = 1.00^2')
            line([200,300],[0.07,0.27]);
            line([400,300],[0.059,0.27]);
        elseif L == 2000
            text(110,0.63,'{\it\sigma}^2 = 0.10^2')
            text(110,0.29,'{\it\sigma}^2 = 0.50^2')
            text(250,0.29,'{\it\sigma}^2 = 1.00^2')
            line([200,250],[0.034,0.26]);
        elseif L == 3000
            text(110,0.635,'{\it\sigma}^2 = 0.10^2')
            text(110,0.295,'{\it\sigma}^2 = 0.50^2')
            text(250,0.295,'{\it\sigma}^2 = 1.00^2')
            line([200,250],[0.029,0.265]);
        end
    end
```

3 Box-Jenkins 模型(BJ)

考虑下列 Box-Jenkins 模型(BJ)描述的有色噪声系统:

$$y(t) = \frac{B(z)}{A(z)}u(t) + \frac{D(z)}{C(z)}v(t). \tag{69}$$

其中$\{u(t)\}$和$\{y(t)\}$分别为系统的输入和输出序列,$\{v(t)\}$为零均值方差为σ^2的随机白噪声序列,$A(z),B(z),C(z)$和$D(z)$均为单位后移算子z^{-1}的多项式:

$$A(z) := 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_{n_a} z^{-n_a},$$

$$B(z) := b_1 z^{-1} + b_2 z^{-2} + \cdots + b_{n_b} z^{-n_b},$$

$$C(z) := 1 + c_1 z^{-1} + c_2 z^{-2} + \cdots + c_{n_c} z^{-n_c},$$

$$D(z) := 1 + d_1 z^{-1} + d_2 z^{-2} + \cdots + d_{n_d} z^{-n_d}.$$

设阶次n_a, n_b, n_c和n_d已知,记$n := n_a + n_b + n_c + n_d$. 系统量测噪声$w(t) := \frac{D(z)}{C(z)}v(t)$为自回归滑动平均模型(ARMA 模型).

定义系统模型输出和噪声模型输出分别为

$$x(t) := \frac{B(z)}{A(z)}u(t), \tag{70}$$

$$w(t) := \frac{D(z)}{C(z)}v(t). \tag{71}$$

则式(69)可写为

$$y(t) = x(t) + w(t). \tag{72}$$

置系统参数向量 $\boldsymbol{\theta}$,系统模型参数向量 $\boldsymbol{\theta}_{\rm s}$ 和噪声模型参数向量 $\boldsymbol{\theta}_{\rm n}$ 分别为

$$\boldsymbol{\theta}:=\begin{bmatrix}\boldsymbol{\theta}_{\rm s}\\ \boldsymbol{\theta}_{\rm n}\end{bmatrix}\in\mathbf{R}^{n_a+n_b+n_c+n_d},$$

$$\boldsymbol{\theta}_{\rm s}:=[a_1,a_2,\cdots,a_{n_a},b_1,b_2,\cdots,b_{n_b}]^{\rm T}\in\mathbf{R}^{n_a+n_b},$$

$$\boldsymbol{\theta}_{\rm n}:=[c_1,c_2,\cdots,c_{n_c},d_1,d_2,\cdots,d_{n_d}]^{\rm T}\in\mathbf{R}^{n_c+n_d},$$

信息向量 $\boldsymbol{\varphi}(t)$,系统模型信息向量 $\boldsymbol{\varphi}_{\rm s}(t)$ 和噪声模型信息向量 $\boldsymbol{\varphi}_{\rm n}(t)$ 分别为

$$\boldsymbol{\varphi}(t):=\begin{bmatrix}\boldsymbol{\varphi}_{\rm s}(t)\\ \boldsymbol{\varphi}_{\rm n}(t)\end{bmatrix}\in\mathbf{R}^{n_a+n_b+n_c+n_d},$$

$$\boldsymbol{\varphi}_{\rm s}(t):=[-x(t-1),-x(t-2),\cdots,-x(t-n_a),u(t-1),u(t-2),\cdots,u(t-n_b)]^{\rm T}\in\mathbf{R}^{n_a+n_b},$$

$$\boldsymbol{\varphi}_{\rm n}(t):=[-w(t-1),-w(t-2),\cdots,-w(t-n_c),v(t-1),v(t-2),\cdots,v(t-n_d)]^{\rm T}\in\mathbf{R}^{n_c+n_d}.$$

这里的下标 s 和 n 分别表示系统模型和噪声模型之意,取自于英文“system”和“noise”的首字母. 那么式(70)—(72)写成向量形式分别为

$$x(t)=\boldsymbol{\varphi}_{\rm s}^{\rm T}(t)\boldsymbol{\theta}_{\rm s},\tag{73}$$

$$w(t)=\boldsymbol{\varphi}_{\rm n}^{\rm T}(t)\boldsymbol{\theta}_{\rm n}+v(t),\tag{74}$$

$$y(t)=\boldsymbol{\varphi}^{\rm T}(t)\boldsymbol{\theta}+v(t).\tag{75}$$

3.1 梯度迭代辨识方法

考虑从 $i=t-p+1$ 到 $i=t$ 最新的 p 组数据,定义堆积输出向量 $\boldsymbol{Y}(t)$,堆积信息矩阵 $\boldsymbol{\Phi}(t)$,堆积白噪声向量 $\boldsymbol{V}(t)$ 如下:

$$\boldsymbol{Y}(t):=\begin{bmatrix}y(t)\\ y(t-1)\\ \vdots\\ y(t-p+1)\end{bmatrix}\in\mathbf{R}^{p},$$

$$\boldsymbol{\Phi}(t):=\begin{bmatrix}\boldsymbol{\varphi}^{\rm T}(t)\\ \boldsymbol{\varphi}^{\rm T}(t-1)\\ \vdots\\ \boldsymbol{\varphi}^{\rm T}(t-p+1)\end{bmatrix}\in\mathbf{R}^{p\times n},\tag{76}$$

$$\boldsymbol{V}(t):=\begin{bmatrix}v(t)\\ v(t-1)\\ \vdots\\ v(t-p+1)\end{bmatrix}\in\mathbf{R}^{p}.\tag{77}$$

注意:如果取 $p=L,t=L(L$ 为数据长度),那么 $\boldsymbol{Y}(t)$ 和 $\boldsymbol{\Phi}(t)$ 就包含了所有量测输入输出数据 $\{u(t),y(t):t=0,1,2,\cdots,L\}$. 由式(75)可得

$$\boldsymbol{Y}(t)=\boldsymbol{\Phi}(t)\boldsymbol{\theta}+\boldsymbol{V}(t).\tag{78}$$

由于 $\boldsymbol{V}(t)$ 是一个零均值白噪声向量,定义准则函数:

$$J_5(\boldsymbol{\theta}):=\|\boldsymbol{Y}(t)-\boldsymbol{\Phi}(t)\boldsymbol{\theta}\|^2.$$

令 $k=1,2,3,\cdots$ 是一个迭代变量,$\hat{\boldsymbol{\theta}}_k(t)$ 为 $\boldsymbol{\theta}$ 的迭代估计. 对于优化问题 J_5,使用负梯度搜索算法(16)可得迭代算法:

$$\hat{\boldsymbol{\theta}}_k(t)=\hat{\boldsymbol{\theta}}_{k-1}(t)-\frac{\mu_k(t)}{2}{\rm grad}[J_5(\hat{\boldsymbol{\theta}}_{k-1}(t))]=\hat{\boldsymbol{\theta}}_{k-1}(t)+\mu_k(t)\boldsymbol{\Phi}^{\rm T}(t)[\boldsymbol{Y}(t)-\boldsymbol{\Phi}(t)\hat{\boldsymbol{\theta}}_{k-1}(t)].\tag{79}$$

式中 $\mu_k(t)$ 为迭代步长(iterative step-size)或收敛因子(convergence factor). 对于 $\boldsymbol{\Phi}(t)$(也就是 $\boldsymbol{\varphi}(t)$)中包含未知变量 $x(t-i),w(t-i)$ 和 $v(t-i)$,仍采用递阶辨识原理:这些未知变量用其第 $k-1$ 次迭代估计值 $\hat{x}_{k-1}(t-i),\hat{w}_{k-1}(t-i)$ 和 $\hat{v}_{k-1}(t-i)$ 代替,代替后的 $\boldsymbol{\varphi}(t)$ 记作

$$\hat{\boldsymbol{\varphi}}_k(t):=\begin{bmatrix}\hat{\boldsymbol{\varphi}}_{{\rm s},k}(t)\\ \hat{\boldsymbol{\varphi}}_{{\rm n},k}(t)\end{bmatrix}\in\mathbf{R}^{n_a+n_b+n_c+n_d},$$

$$\hat{\boldsymbol{\varphi}}_{{\rm s},k}(t):=[-\hat{x}_{k-1}(t-1),-\hat{x}_{k-1}(t-2),\cdots,-\hat{x}_{k-1}(t-n_a),u(t-1),u(t-2),\cdots,u(t-n_b)]^{\rm T}\in\mathbf{R}^{n_a+n_b},$$

$$\hat{\boldsymbol{\varphi}}_{{\rm n},k}(t):=[-\hat{w}_{k-1}(t-1),-\hat{w}_{k-1}(t-2),\cdots,-\hat{w}_{k-1}(t-n_c),\hat{v}_{k-1}(t-1),\hat{v}_{k-1}(t-2),\cdots,\hat{v}_{k-1}(t-n_d)]^{\rm T}\in\mathbf{R}^{n_c+n_d}.$$

令 $k=1,2,3,\cdots$ 是一个迭代变量,$\hat{\boldsymbol{\theta}}_k(t):=\begin{bmatrix}\hat{\boldsymbol{\theta}}_{{\rm s},k}(t)\\ \hat{\boldsymbol{\theta}}_{{\rm n},k}(t)\end{bmatrix}$ 是 $\boldsymbol{\theta}=\begin{bmatrix}\boldsymbol{\theta}_{\rm s}\\ \boldsymbol{\theta}_{\rm n}\end{bmatrix}$ 的迭代估计. 用 $(t-i)$ 代替式(73)中 t 得到

$$x(t-i)=\boldsymbol{\varphi}_{\rm s}^{\rm T}(t-i)\boldsymbol{\theta}_{\rm s}.$$

用 $\hat{\boldsymbol{\varphi}}_{{\rm s},k}(t-i)$ 和 $\hat{\boldsymbol{\theta}}_{{\rm s},k}(t)$ 分别代替上式中 $\boldsymbol{\varphi}_{\rm s}(t-i)$ 和 $\boldsymbol{\theta}_{\rm s}$,可得到 $x(t-i)$ 的第 k 次迭代估计:

$$\hat{x}_k(t-i)=\hat{\boldsymbol{\varphi}}_{{\rm s},k}^{\rm T}(t-i)\hat{\boldsymbol{\theta}}_{{\rm s},k}(t).\tag{80}$$

由式(72)可得

$$w(t-i)=y(t-i)-x(t-i).$$

用 $\hat{x}_k(t-i)$ 代替上式中 $x(t-i)$,可得 $w(t-i)$ 的第 k 次迭代估计:

$$\hat{w}_k(t-i)=y(t-i)-\hat{x}_k(t-i)=y(t-i)-\hat{\boldsymbol{\varphi}}_{{\rm s},k}^{\rm T}(t-i)\hat{\boldsymbol{\theta}}_{{\rm s},k}(t).\tag{81}$$

由式(74)可得

$$v(t-i)=w(t-i)-\boldsymbol{\varphi}_{\rm n}^{\rm T}(t-i)\boldsymbol{\theta}_{\rm n}.$$

用 $\hat{w}_k(t-i),\hat{\boldsymbol{\varphi}}_{{\rm n},k}(t-i)$ 和 $\hat{\boldsymbol{\theta}}_{{\rm n},k}(t)$ 分别代替上式中 $w(t-i),\boldsymbol{\varphi}_{\rm n}(t-i)$ 和 $\boldsymbol{\theta}_{\rm n}$,可得到 $v(t-i)$ 的第 k 次迭代估计:

$$\hat{v}_k(t-i)=\hat{w}_k(t-i)-\hat{\boldsymbol{\varphi}}_{{\rm n},k}^{\rm T}(t-i)\hat{\boldsymbol{\theta}}_{{\rm n},k}(t).\tag{82}$$

当然,$\hat{v}_k(t-i)$ 也可以这样求:由式(75)可得

$$v(t-i)=y(t-i)-\boldsymbol{\varphi}^{\rm T}(t)\boldsymbol{\theta}.$$

用 $\hat{\boldsymbol{\varphi}}_k(t)$ 和 $\hat{\boldsymbol{\theta}}_k(t)$ 分别代替上式中 $\boldsymbol{\varphi}(t)$ 和 $\boldsymbol{\theta}$,得到 $v(t)$ 的第 k 次迭代估计:

$$\hat{v}_k(t-i)=y(t-i)-\hat{\boldsymbol{\varphi}}_k^{\mathrm{T}}(t-i)\hat{\boldsymbol{\theta}}_k(t). \tag{83}$$

用 $\hat{\boldsymbol{\varphi}}_k(t-i)$ 代替 $\boldsymbol{\Phi}(t)$ 中 $\boldsymbol{\varphi}(t-i)$,代替后的 $\boldsymbol{\Phi}(t)$ 记作:

$$\hat{\boldsymbol{\Phi}}_k(t):=\begin{bmatrix}\hat{\boldsymbol{\varphi}}_k^{\mathrm{T}}(t)\\ \hat{\boldsymbol{\varphi}}_k^{\mathrm{T}}(t-1)\\ \vdots\\ \hat{\boldsymbol{\varphi}}_k^{\mathrm{T}}(t-p+1)\end{bmatrix}\in\mathbf{R}^{p\times n}. \tag{84}$$

用 $\hat{\boldsymbol{\Phi}}_k(t)$ 代替式(79)中 $\boldsymbol{\Phi}(t)$,得到 Box-Jenkins 模型的梯度迭代辨识算法(BJ-GI,Gradient based Iterative identification algorithm for Box-Jenkins models)(BJ-GI 算法):

$$\hat{\boldsymbol{\theta}}_k(t)=\hat{\boldsymbol{\theta}}_{k-1}(t)+\mu_k(t)\hat{\boldsymbol{\Phi}}_k^{\mathrm{T}}(t)[\boldsymbol{Y}(t)-\hat{\boldsymbol{\Phi}}_k(t)\hat{\boldsymbol{\theta}}_{k-1}(t)],\quad k=1,2,3,\cdots, \tag{85}$$

$$\hat{\boldsymbol{\Phi}}_k(t)=[\hat{\boldsymbol{\varphi}}_k(t),\hat{\boldsymbol{\varphi}}_k(t-1),\cdots,\hat{\boldsymbol{\varphi}}_k(t-p+1)]^{\mathrm{T}}, \tag{86}$$

$$\boldsymbol{Y}(t)=[y(t),y(t-1),\cdots,y(t-p+1)]^{\mathrm{T}}, \tag{87}$$

$$\hat{\boldsymbol{\varphi}}_k(t)=\begin{bmatrix}\hat{\boldsymbol{\varphi}}_{\mathrm{s},k}(t)\\ \hat{\boldsymbol{\varphi}}_{\mathrm{n},k}(t)\end{bmatrix}, \tag{88}$$

$$\hat{\boldsymbol{\varphi}}_{\mathrm{s},k}(t)=[-\hat{x}_{k-1}(t-1),-\hat{x}_{k-1}(t-2),\cdots,-\hat{x}_{k-1}(t-n_a),u(t-1),u(t-2),\cdots,u(t-n_b)]^{\mathrm{T}}, \tag{89}$$

$$\hat{\boldsymbol{\varphi}}_{\mathrm{n},k}(t)=[-\hat{w}_{k-1}(t-1),-\hat{w}_{k-1}(t-2),\cdots,-\hat{w}_{k-1}(t-n_c),\hat{v}_{k-1}(t-1),\hat{v}_{k-1}(t-2),\cdots,\hat{v}_{k-1}(t-n_d)]^{\mathrm{T}}, \tag{90}$$

$$\hat{\boldsymbol{\theta}}_k(t)=\begin{bmatrix}\hat{\boldsymbol{\theta}}_{\mathrm{s},k}(t)\\ \hat{\boldsymbol{\theta}}_{\mathrm{n},k}(t)\end{bmatrix}, \tag{91}$$

$$\hat{x}_k(t-i)=\hat{\boldsymbol{\varphi}}_{\mathrm{s},k}^{\mathrm{T}}(t-i)\hat{\boldsymbol{\theta}}_{\mathrm{s},k}(t),i=1,2,\cdots,n_a, \tag{92}$$

$$\hat{w}_k(t-i)=y(t)-\hat{x}_k(t-i)=y(t-i)-\hat{\boldsymbol{\varphi}}_{\mathrm{s},k}^{\mathrm{T}}(t-i)\hat{\boldsymbol{\theta}}_{\mathrm{s},k}(t),\quad i=1,2,\cdots,n_c, \tag{93}$$

$$\hat{v}_k(t-i)=\hat{w}_k(t-i)-\hat{\boldsymbol{\varphi}}_{\mathrm{n},k}^{\mathrm{T}}(t-i)\hat{\boldsymbol{\theta}}_{\mathrm{n},k}(t)=y(t-i)-\hat{\boldsymbol{\varphi}}_k^{\mathrm{T}}(t-i)\hat{\boldsymbol{\theta}}_k(t),\quad i=1,2,\cdots,n_d, \tag{94}$$

$$0<\mu_k(t)\leqslant\frac{2}{\lambda_{\max}[\hat{\boldsymbol{\Phi}}_k^{\mathrm{T}}(t)\hat{\boldsymbol{\Phi}}_k(t)]}. \tag{95}$$

BJ-GI 算法的计算步骤如下.

1) 令 $t=1$,确定 p,给定参数估计精度 ε,置 $\hat{\boldsymbol{\theta}}_0(t)=\mathbf{1}_n$,$p_0=10^6$.

2) 收集输入输出数据 $u(t)$ 和 $y(t)$,用式(87)构造 $\boldsymbol{Y}(t)$.

3) 令 $k=1$,置 $\hat{x}_0(t-i)=1/p_0$,$\hat{w}_0(t-i)=1/p_0$,$\hat{v}_0(t-i)=1/p_0$,$i=1,2,\cdots,\max[n_a,n_c,n_d]$.

4) 用式(89)构造 $\hat{\boldsymbol{\varphi}}_{\mathrm{s},k}(t)$,用式(90)构造 $\hat{\boldsymbol{\varphi}}_{\mathrm{n},k}(t)$,用式(88)构造 $\hat{\boldsymbol{\varphi}}_k(t)$,用式(86)构造 $\hat{\boldsymbol{\Phi}}_k(t)$.

5) 根据式(95)选择一个大 $\mu_k(t)$,用式(85)刷新参数估计 $\hat{\boldsymbol{\theta}}_k(t)$.

6) 用式(92)计算 $\hat{x}_k(t-i)$,用式(93)计算 $\hat{w}_k(t-i)$,用式(94)计算 $\hat{v}_k(t-i)$.

7) 比较 $\hat{\boldsymbol{\theta}}_k(t)$ 与 $\hat{\boldsymbol{\theta}}_{k-1}(t)$:如果 $\|\hat{\boldsymbol{\theta}}_k(t)-\hat{\boldsymbol{\theta}}_{k-1}(t)\|>\varepsilon$,$k$ 增 1,转到步骤 4;否则,获得迭代次数 k 和参数估计向量 $\hat{\boldsymbol{\theta}}_k(t)$,令 $\hat{\boldsymbol{\theta}}_0(t+1):=\hat{\boldsymbol{\theta}}_k(t)$,$t$ 增 1,转到步骤 2.

有限量测数据 BJ-GI 算法在式(76)—(77)中取 $p=L$,$t=L$(L 为数据长度),有

$$\boldsymbol{Y}(L):=\begin{bmatrix}y(L)\\ y(L-1)\\ \vdots\\ y(1)\end{bmatrix}\in\mathbf{R}^L,\boldsymbol{\Phi}(L):=\begin{bmatrix}\boldsymbol{\varphi}^{\mathrm{T}}(L)\\ \boldsymbol{\varphi}^{\mathrm{T}}(L-1)\\ \vdots\\ \boldsymbol{\varphi}^{\mathrm{T}}(1)\end{bmatrix}\in\mathbf{R}^{L\times n},$$

$$\boldsymbol{V}(L):=\begin{bmatrix}v(L)\\ v(L-1)\\ \vdots\\ v(1)\end{bmatrix}\in\mathbf{R}^L,$$

$\boldsymbol{Y}(L)$ 和 $\boldsymbol{\Phi}(L)$ 就包含了所有量测输入输出数据 $\{u(t),y(t):t=0,1,2,\cdots,L\}$. 由式(75)可得

$$\boldsymbol{Y}(L)=\boldsymbol{\Phi}(L)\boldsymbol{\theta}+\boldsymbol{V}(L). \tag{96}$$

由于 $\boldsymbol{V}(L)$ 是一个零均值白噪声向量,定义准则函数:

$$J_6(\boldsymbol{\theta})=\|\boldsymbol{Y}(L)-\boldsymbol{\Phi}(L)\boldsymbol{\theta}\|^2.$$

按照 BJ-GI 算法的推导思路,可以得到有限量测数据 Box-Jenkins 模型的梯度迭代辨识算法(BJ-GI,Gradient based Iterative identification algorithm for BJ models with finite mea-surement data):

$$\hat{\boldsymbol{\theta}}_k=\hat{\boldsymbol{\theta}}_{k-1}+\mu_k\hat{\boldsymbol{\Phi}}_k^{\mathrm{T}}(L)[\boldsymbol{Y}(L)-\hat{\boldsymbol{\Phi}}_k(L)\hat{\boldsymbol{\theta}}_{k-1}],\quad k=1,2,3,\cdots, \tag{97}$$

$$\hat{\boldsymbol{\Phi}}_k(L)=[\hat{\boldsymbol{\varphi}}_k(L),\hat{\boldsymbol{\varphi}}_k(L-1),\cdots,\hat{\boldsymbol{\varphi}}_k(1)]^{\mathrm{T}}, \tag{98}$$

$$\boldsymbol{Y}(L)=[y(L),y(L-1),\cdots,y(1)]^{\mathrm{T}}, \tag{99}$$

$$\hat{\boldsymbol{\varphi}}_k(t)=\begin{bmatrix}\hat{\boldsymbol{\varphi}}_{\mathrm{s},k}(t)\\ \hat{\boldsymbol{\varphi}}_{\mathrm{n},k}(t)\end{bmatrix},\quad t=1,2,\cdots,L, \tag{100}$$

$$\hat{\boldsymbol{\varphi}}_{\mathrm{s},k}(t)=[-\hat{x}_{k-1}(t-1),-\hat{x}_{k-1}(t-2),\cdots,-\hat{x}_{k-1}(t-n_a),u(t-1),u(t-2),\cdots,u(t-n_b)]^{\mathrm{T}}, \tag{101}$$

$$\hat{\boldsymbol{\varphi}}_{\mathrm{n},k}(t)=[-\hat{w}_{k-1}(t-1),-\hat{w}_{k-1}(t-2),\cdots,-\hat{w}_{k-1}(t-n_c),\hat{v}_{k-1}(t-1),\hat{v}_{k-1}(t-2),\cdots,\hat{v}_{k-1}(t-n_d)]^{\mathrm{T}}, \tag{102}$$

$$\hat{\boldsymbol{\theta}}_k=\begin{bmatrix}\hat{\boldsymbol{\theta}}_{\mathrm{s},k}\\ \hat{\boldsymbol{\theta}}_{\mathrm{n},k}\end{bmatrix}, \tag{103}$$

$$\hat{x}_k(t)=\hat{\boldsymbol{\varphi}}_{\mathrm{s},k}^{\mathrm{T}}(t)\hat{\boldsymbol{\theta}}_{\mathrm{s},k},\quad k=1,2,\cdots,L, \tag{104}$$

$$\hat{w}_k(t)=y(t)-\hat{x}_k(t)=y(t)-\hat{\boldsymbol{\varphi}}_{\mathrm{s},k}^{\mathrm{T}}(t)\hat{\boldsymbol{\theta}}_{\mathrm{s},k}, \tag{105}$$

$$\hat{v}_k(t)=\hat{w}_k(t)-\hat{\boldsymbol{\varphi}}_{\mathrm{n},k}^{\mathrm{T}}(t)\hat{\boldsymbol{\theta}}_{\mathrm{n},k}, \tag{106}$$

$$0<\mu_k\leqslant\frac{2}{\lambda_{\max}[\hat{\boldsymbol{\Phi}}_k^{\mathrm{T}}(L)\hat{\boldsymbol{\Phi}}_k(L)]}. \tag{107}$$

有限量测数据的 BJ-GI 算法的计算步骤如下.

1) 收集输入输出数据 $\{u(t)$ 和 $y(t):i=1,2,\cdots,L\}$,用式(98)构造 $\boldsymbol{Y}(t)$.

2) 令 $k=1$,置 $\hat{\boldsymbol{\theta}}_0=\mathbf{1}_n/p_0$,$\hat{x}_0(t)=1/p_0$,$\hat{w}_0(t)=1/p_0$,$\hat{v}_0(t)=1/p_0$,$p_0=10^6$.

3) 用式(101)构造 $\hat{\boldsymbol{\varphi}}_{\mathrm{s},k}(t)$,用式(102)构造 $\hat{\boldsymbol{\varphi}}_{\mathrm{n},k}(t)$,用式(100)构造 $\hat{\boldsymbol{\varphi}}_k(t)$,用式(98)构造 $\hat{\boldsymbol{\Phi}}_k(t)$.

4) 根据式(107)选择一个大 μ_k,用式(97)刷新参数估计 $\hat{\boldsymbol{\theta}}_k$.

5) 用式(104)计算 $\hat{x}_k(t)$,用式(105)和式(106)分别计算 $\hat{w}_k(t)$ 和 $\hat{v}_k(t)$,

6) 比较 $\hat{\boldsymbol{\theta}}_k$ 与 $\hat{\boldsymbol{\theta}}_{k-1}$:如果 $\|\hat{\boldsymbol{\theta}}_k-\hat{\boldsymbol{\theta}}_{k-1}\|\leqslant\varepsilon$,中断循环过程,获得迭代次数 k 和参数估计向量 $\hat{\boldsymbol{\theta}}_k$;否则,$k$ 增 1,转到步骤 3.

有限量测数据 BJ-GI 算法参数估计 $\hat{\boldsymbol{\theta}}_k$ 的流程如图 15 所示.

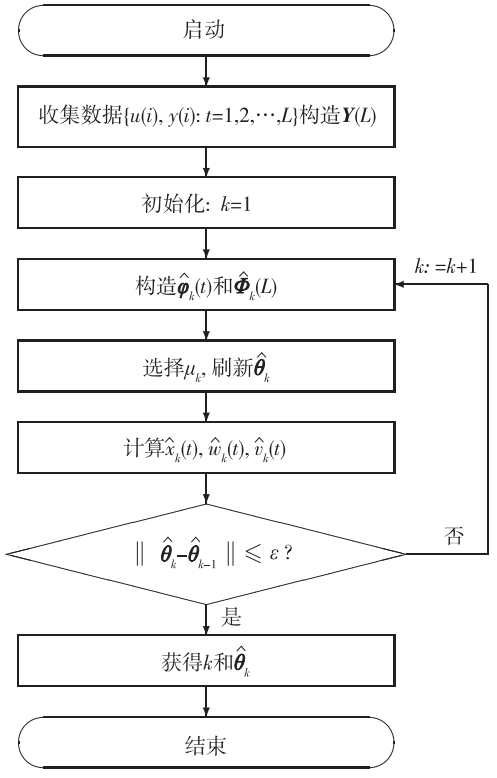

图 15 有限量测数据 BJ-GI 算法参数估计 $\hat{\boldsymbol{\theta}}_k$ 的流程

Fig. 15 The flowchart for computing the iterative estimate $\hat{\boldsymbol{\theta}}_k$

3.2 最小二乘迭代方法

假设信息向量 $\boldsymbol{\varphi}(t)$ 是持续激励的,即 $[\boldsymbol{\Phi}^{\mathrm{T}}(t)\boldsymbol{\Phi}(t)]$ 是可逆矩阵. 极小化准则函数 $J_6(\hat{\boldsymbol{\theta}})$ 给出下列最小二乘估计:

$$\hat{\boldsymbol{\theta}}(t)=[\boldsymbol{\Phi}^{\mathrm{T}}(t)\boldsymbol{\Phi}(t)]^{-1}\boldsymbol{\Phi}^{\mathrm{T}}(t)\boldsymbol{Y}(t). \tag{108}$$

式(108)无法计算 $\hat{\boldsymbol{\theta}}(t)$,因为 $\boldsymbol{\Phi}(t)$(也就是 $\boldsymbol{\varphi}(t)$)中包含了未知中间变量 $x(t-i)$,$w(t-i)$,$v(t-i)$. 类似于 BJ-GI 算法的处理方法:$x(t-i)$,$w(t-i)$,$v(t-i)$ 都分别用它们的估计 $\hat{x}_{k-1}(t-i)$,$\hat{w}_{k-1}(t-i)$,$\hat{v}_{k-1}(t-i)$ 代替,$\boldsymbol{\varphi}(t)$ 用 $\hat{\boldsymbol{\varphi}}_k(t)$ 代替,用 $\hat{\boldsymbol{\Phi}}_k(t)$ 代替式(108)中 $\boldsymbol{\Phi}(t)$,可得 Box-Jenkins 模型的最小二乘迭代辨识算法(BJ-LSI, Least Squares based Iterative identifcation algorithm for Box-Jenkins models)(BJ-LSI 算法):

$$\hat{\boldsymbol{\theta}}_k(t)=[\hat{\boldsymbol{\Phi}}_k^{\mathrm{T}}(t)\hat{\boldsymbol{\Phi}}_k(t)]^{-1}\hat{\boldsymbol{\Phi}}_k^{\mathrm{T}}(t)\boldsymbol{Y}(t),\quad k=1,2,3,\cdots \tag{109}$$

$$\hat{\boldsymbol{\Phi}}_k(t)=[\hat{\boldsymbol{\varphi}}_k(t),\hat{\boldsymbol{\varphi}}_k(t-1),\cdots,\hat{\boldsymbol{\varphi}}_k(t-p+1)]^{\mathrm{T}}, \tag{110}$$

$$\boldsymbol{Y}(t)=[y(t),y(t-1),\cdots,y(t-p+1)]^{\mathrm{T}}, \tag{111}$$

$$\hat{\boldsymbol{\varphi}}_k(t)=\begin{bmatrix}\hat{\boldsymbol{\varphi}}_{\mathrm{s},k}(t)\\ \hat{\boldsymbol{\varphi}}_{\mathrm{n},k}(t)\end{bmatrix}, \tag{112}$$

$$\hat{\boldsymbol{\varphi}}_{\mathrm{s},k}(t)=[-\hat{x}_{k-1}(t-1),-\hat{x}_{k-1}(t-2),\cdots,-\hat{x}_{k-1}(t-n_a),u(t-1),u(t-2),\cdots,u(t-n_b)]^{\mathrm{T}}, \tag{113}$$

$$\hat{\boldsymbol{\varphi}}_{\mathrm{n},k}(t)=[-\hat{w}_{k-1}(t-1),-\hat{w}_{k-1}(t-2),\cdots,-\hat{w}_{k-1}(t-n_c),\hat{v}_{k-1}(t-1),\hat{v}_{k-1}(t-2),\cdots,\hat{v}_{k-1}(t-n_d)]^{\mathrm{T}}, \tag{114}$$

$$\hat{\boldsymbol{\theta}}_k(t)=\begin{bmatrix}\hat{\boldsymbol{\theta}}_{\mathrm{s},k}(t)\\ \hat{\boldsymbol{\theta}}_{\mathrm{n},k}(t)\end{bmatrix}, \tag{115}$$

$$\hat{x}_k(t-i)=\hat{\boldsymbol{\varphi}}_{\mathrm{s},k}^{\mathrm{T}}(t-i)\hat{\boldsymbol{\theta}}_{\mathrm{s},k}(t),i=1,2,\cdots,n_a, \tag{116}$$

$$\hat{w}_k(t-i)=y(t)-\hat{x}_k(t-i)=y(t-i)-\hat{\boldsymbol{\varphi}}_{\mathrm{s},k}^{\mathrm{T}}(t-i)\hat{\boldsymbol{\theta}}_{\mathrm{s},k}(t),\quad i=1,2,\cdots,n_c, \tag{117}$$

$$\hat{v}_k(t-i)=\hat{w}_k(t-i)-\hat{\boldsymbol{\varphi}}_{\mathrm{n},k}^{\mathrm{T}}(t-i)\hat{\boldsymbol{\theta}}_{\mathrm{n},k}(t)=y(t-i)-\hat{\boldsymbol{\varphi}}_k^{\mathrm{T}}(t-i)\hat{\boldsymbol{\theta}}_k(t),\quad i=1,2,\cdots,n_d. \tag{118}$$

BJ-LSI 算法的计算步骤如下.

1) 确定 p,令 $t=p$,收集输入输出数据 $\{u(i),y(i):i=0,1,\cdots,p-1\}$,给定参数估计精度 ε.

2) 收集输入输出数据 $u(t)$ 和 $y(t)$,用式(111)构造 $\boldsymbol{Y}(t)$.

3) 令 $k=1$,置 $\hat{x}_0(t-i)=$随机数,$\hat{w}_0(t-i)=$随机数,$\hat{v}_0(t-i)=$随机数,$i=1,2,\cdots,\max[n_a,n_c,n_d]$.

4) 用式(113)构造 $\hat{\boldsymbol{\varphi}}_{\mathrm{s},k}(t)$,用式(114)构造 $\hat{\boldsymbol{\varphi}}_{\mathrm{n},k}(t)$,用式(112)构造 $\hat{\boldsymbol{\varphi}}_k(t)$,用式(110)构造 $\hat{\boldsymbol{\Phi}}_k(t)$.

5) 用式(109)刷新参数估计 $\hat{\boldsymbol{\theta}}_k(t)$.

6) 用式(116)计算 $\hat{x}_k(t-i)$,用式(117)计算 $\hat{w}_k(t-i)$,用式(118)计算 $\hat{v}_k(t-i)$.

7) 比较 $\hat{\boldsymbol{\theta}}_k(t)$ 与 $\hat{\boldsymbol{\theta}}_{k-1}(t)$:如果 $\|\hat{\boldsymbol{\theta}}_k(t)-$

$\hat{\boldsymbol{\theta}}_{k-1}(t) \| > \varepsilon$,$k$ 增 1,转到步骤 4;否则,获得迭代次数 k 和参数估计向量 $\hat{\boldsymbol{\theta}}_k(t)$,$t$ 增 1,转到步骤 2.

BJ-LSI 算法计算参数估计 $\hat{\boldsymbol{\theta}}_k(t)$ 的流程如图 16 所示.

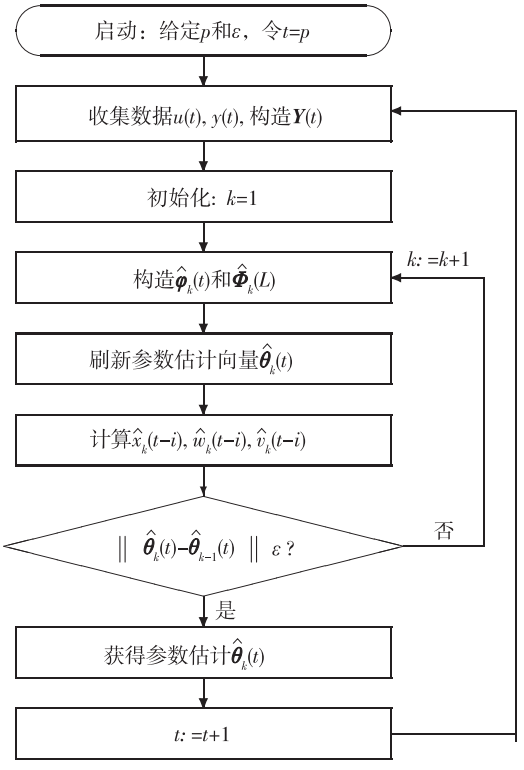

图 16 计算 BJ-LSI 参数估计 $\hat{\boldsymbol{\theta}}_k(t)$ 的流程

Fig. 16 The flowchart for computing the BJ-LSI parameter estimate $\hat{\boldsymbol{\theta}}_k(t)$

BJ-LSI 算法是利用数据窗长度为 p 的有限数据窗内的数据极小化准则函数得到的,因此具有跟踪时变参数的能力,能用于在线辨识,BJ-GI 算法也具有这一性质.

4 非线性系统的迭代辨识方法

线性系统模型结构比较简单,可以用统一的模型来描述,如差分方程、状态空间模型等,而非线性系统结构通常很复杂,难以用统一的模型来描述. 研究最多的是比较简单的块结构非线性系统,如输入非线性系统(简称 N-L. N:Nonlinear,非线性之意;L:Linear,线性之意),它是由一个静态非线性环节串连一个线性动态子系统构成的;另一种是输出非线性系统(简称 L-N),它是由一个线性动态子系统串连一个静态非线性环节构成的. 如果静态非线性环节是一个多项式或一个已知基的线性组合函数,则这类输入非线性系统称为 Hammerstein 非线性系统,这类输出非线性系统称为 Wiener 非线性系统. 当然,还有 N-L-N 非线性系统(有时称为 Hammerstein-Wiener 非线性系统),L-N-L 非线性系统(称为 Wiener-Hammerstein 非线性系统). 如果非线性环节在反馈回路上,其前向通道是一个线性子系统,或反之,这样的非线性系统称为反馈非线性系统. 值得指出的是,非线性环节也可以是一个非线性动态子系统,这样的非线性系统块结构研究得不是很多.

文献[12]研究了 Hammerstein 非线性方程误差系统的牛顿迭代辨识方法和牛顿递推辨识方法等,文献[6]提出了 Hammerstein 非线性 ARMAX 系统的最小二乘迭代辨识方法和递推增广最小二乘辨识方法,文献[13]讨论了一类特殊 Wiener 非线性系统的最小二乘迭代辨识方法和梯度迭代辨识方法,所采用的是双线性参数向量输出非线性系统. 这里简单讨论一类 Wiener 非线性系统的迭代辨识方法,如图 17 所示. 其中 $u(t)$ 为系统输入,$x(t)$ 为线性部分的输出(不可测中间变量),$\bar{y}(t)$ 为非线性部分的输出(未知的),$v(t)$ 为量测噪声(可以假定是零均值和有限方差的),$y(t)$ 是系统输出,即 $\bar{y}(t)$ 的含噪量测,$f(\cdot)$ 是输出端的非线性函数,$G(z)$ 是线性部分的传递函数. 假设一个 FIR 模型:

$$G(z) := b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_n z^{-n}.$$

这个输出非线性 FIR 系统可以表示为

$$x(t) = G(z)u(t), \tag{119}$$

$$\bar{y}(t) = f(x(t)), \tag{120}$$

$$y(t) = \bar{y}(t) + v(t) = f(x(t)) + v(t). \tag{121}$$

定义参数向量 $\boldsymbol{\theta}$ 和信息向量 $\boldsymbol{\varphi}(t)$ 如下:

$$\boldsymbol{\theta} := [b_0, b_1, b_2, \cdots, b_n]^{\mathrm{T}} \in \mathbf{R}^{n+1},$$

$$\boldsymbol{\varphi}(t) := [u(t), u(t-1), u(t-2), \cdots, u(t-n)]^{\mathrm{T}} \in \mathbf{R}^{n+1}.$$

则有

$$x(t) = \boldsymbol{\varphi}^{\mathrm{T}}(t)\boldsymbol{\theta}. \tag{122}$$

对于这样一个简单的输出非线性输出误差系统,即使假定非线性函数是一个二次函数

$$\bar{y} = f(x) = x^2,$$

或

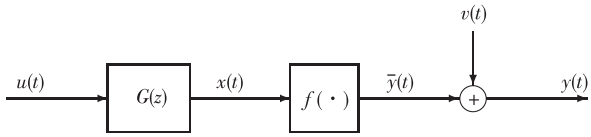

图 17 输出非线性输出误差系统(ON-OE)

Fig. 17 The output nonlinear output error system

$$\bar{y}(t)=f(x(t))=x^2(t),$$

其辨识问题也是不容易的. 当然,也可以假设这个输出非线性特性 $\bar{y}=f(x)$ 是已知非线性基 $\boldsymbol{f}=(f_1,f_2,\cdots,f_m)$ 的线性函数,即非线性特性可以写为参数 $(\vartheta_1,\vartheta_2,\cdots,\vartheta_m)$ 的线性组合形式

$$\bar{y}(t)=f(x(t))=$$
$$\vartheta_1 f_1(x(t))+\vartheta_2 f_2(x(t))+\cdots+\vartheta_m f_m(x(t))=$$
$$\boldsymbol{\psi}^{\mathrm{T}}(x(t))\boldsymbol{\vartheta}, \tag{123}$$

其中 $\boldsymbol{\psi}(x(t)):=[f_1(x(t)),f_2(x(t)),\cdots,f_m(x(t))]^{\mathrm{T}}\in\mathbf{R}^m$ 是基函数构成的向量,$\boldsymbol{\vartheta}:=[\vartheta_1,\vartheta_2,\cdots,\vartheta_m]^{\mathrm{T}}\in\mathbf{R}^m$ 是非线性部分的参数向量.

非线性系统模型中一般会出现系统参数的乘积项,为了得到唯一的参数估计,需要规范化模型参数[6,12]. 常用的规范化方法有:1) 固定 b_i 中的一个,或者固定 ϑ_j 中的一个;2) 设 $(b_0,b_1,b_2,\cdots,b_n)$ 或 $(\vartheta_1,\vartheta_2,\cdots,\vartheta_m)$ 的模为 1,即 $b_0^2+b_1^2+b_2^2+\cdots+b_n^2=1$,或 $\vartheta_1^2+\vartheta_2^2+\cdots+\vartheta_m^2=1$;3) 设线性子系统的增益为 1,即 $G(1)=b_0+b_1+b_2+\cdots+b_n=1$,或非线性函数的系数和为 1,即 $\vartheta_1+\vartheta_2+\cdots+\vartheta_m=1$.

联立式(121)—(123)给出输出非线性 FIR 系统的辨识模型:

$$y(t)=f(x(t))+v(t)=\boldsymbol{\psi}^{\mathrm{T}}(x(t))\boldsymbol{\vartheta}+v(t), \tag{124}$$
$$\bar{y}(t)=f(x(t))=\boldsymbol{\psi}^{\mathrm{T}}(x(t))\boldsymbol{\vartheta}, \tag{125}$$
$$x(t)=\boldsymbol{\varphi}^{\mathrm{T}}(t)\boldsymbol{\theta}, \tag{126}$$
$$\boldsymbol{\psi}(x(t))=[f_1(x(t)),f_2(x(t)),\cdots,f_m(x(t))]^{\mathrm{T}}, \tag{127}$$
$$\boldsymbol{\varphi}(t)=[u(t),u(t-1),u(t-2),\cdots,u(t-n)]^{\mathrm{T}}. \tag{128}$$

这个非线性系统参数向量 $\boldsymbol{\vartheta}$ 和 $\boldsymbol{\theta}$ 辨识的困难在于中间变量 $x(t)$ 是未知的. 这里采用迭代最小二乘来研究 $\boldsymbol{\vartheta}$ 和 $\boldsymbol{\theta}$ 的辨识方法.

设数据长度为 L. 在式(125)约束下,定义最小二乘准则函数:

$$J_7(\boldsymbol{\vartheta},\boldsymbol{\theta}):=\sum_{t=1}^{L}[y(t)-f(x(t))]^2=$$
$$\sum_{t=1}^{L}[y(t)-\boldsymbol{\psi}^{\mathrm{T}}(x(t))\boldsymbol{\vartheta}]^2,$$
$$\bar{y}(t)=f(x(t))=\boldsymbol{\psi}^{\mathrm{T}}(x(t))\boldsymbol{\vartheta},$$
$$x(t)=\boldsymbol{\varphi}^{\mathrm{T}}(t)\boldsymbol{\theta}. \tag{129}$$

$J_7(\boldsymbol{\vartheta},\boldsymbol{\theta})$ 分别对 $\boldsymbol{\vartheta}$ 和 $\boldsymbol{\theta}$ 求偏导数(梯度):

$$\mathrm{grad}_{\boldsymbol{\vartheta}}[J_7(\boldsymbol{\vartheta},\boldsymbol{\theta})]:=\frac{\partial J_7(\boldsymbol{\vartheta},\boldsymbol{\theta})}{\partial\boldsymbol{\vartheta}}=$$
$$-2\sum_{t=1}^{L}\boldsymbol{\psi}(x(t))[y(t)-\boldsymbol{\psi}^{\mathrm{T}}(x(t))\boldsymbol{\vartheta}]=$$
$$-2\sum_{t=1}^{L}[\boldsymbol{\psi}(x(t))y(t)-\boldsymbol{\psi}(x(t))\boldsymbol{\psi}^{\mathrm{T}}(x(t))\boldsymbol{\vartheta}],$$
$$\mathrm{grad}_{\boldsymbol{\theta}}[J_7(\boldsymbol{\vartheta},\boldsymbol{\theta})]:=\frac{\partial J_7(\boldsymbol{\vartheta},\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}=$$
$$-2\sum_{t=1}^{L}\boldsymbol{\varphi}(t)f'(x(t))[y(t)-f(x(t))]=$$
$$-2\sum_{t=1}^{L}[\boldsymbol{\varphi}(t)f'(x(t))y(t)-\boldsymbol{\varphi}(t)f'(x(t))f(x(t))].$$

令 $\mu_1(k)\geqslant 0$ 和 $\mu_2(k)\geqslant 0$ 是收敛因子,$k=1,2,3,\cdots$ 是迭代变量,$\hat{\boldsymbol{\vartheta}}_k$ 和 $\hat{\boldsymbol{\theta}}_k$ 分别是 $\boldsymbol{\vartheta}$ 和 $\boldsymbol{\theta}$ 的迭代估计,使用梯度搜索,可得下列基于梯度的迭代算法:

$$\hat{\boldsymbol{\vartheta}}_k=\hat{\boldsymbol{\vartheta}}_{k-1}-\frac{\mu_1(k)}{2}\mathrm{grad}_{\boldsymbol{\vartheta}}[J_7(\hat{\boldsymbol{\vartheta}}_{k-1},\hat{\boldsymbol{\theta}}_{k-1})]=$$
$$\hat{\boldsymbol{\vartheta}}_{k-1}+\mu_1(k)\sum_{t=1}^{L}[\boldsymbol{\psi}(\hat{x}_{k-1}(t))y(t)-$$
$$\boldsymbol{\psi}(\hat{x}_{k-1}(t))\boldsymbol{\psi}^{\mathrm{T}}(\hat{x}_{k-1}(t))\hat{\boldsymbol{\vartheta}}_{k-1}], \tag{130}$$
$$\hat{\boldsymbol{\theta}}_k=\hat{\boldsymbol{\theta}}_{k-1}-\frac{\mu_2(k)}{2}\mathrm{grad}_{\boldsymbol{\theta}}[J_7(\hat{\boldsymbol{\vartheta}}_{k-1},\hat{\boldsymbol{\theta}}_{k-1})]=$$
$$\hat{\boldsymbol{\theta}}_{k-1}+\mu_2(k)\sum_{t=1}^{L}[\boldsymbol{\varphi}(t)f'(\hat{x}_{k-1}(t))y(t)-$$
$$\boldsymbol{\varphi}(t)f'(\hat{x}_{k-1}(t))f(\hat{x}_{k-1}(t))], \tag{131}$$
$$\hat{x}_k=\boldsymbol{\varphi}^{\mathrm{T}}(t)\hat{\boldsymbol{\theta}}_k, \tag{132}$$
$$\boldsymbol{\psi}(\hat{x}_k(t))=$$
$$[f_1(\hat{x}_k(t)),f_2(\hat{x}_k(t)),\cdots,f_m(\hat{x}_k(t))]^{\mathrm{T}}, \tag{133}$$
$$f(\hat{x}_k(t))=\boldsymbol{\psi}^{\mathrm{T}}(\hat{x}_k(t))\hat{\boldsymbol{\vartheta}}_k, \tag{134}$$
$$f'(\hat{x}_k(t))=[\boldsymbol{\psi}'(\hat{x}_k(t))]^{\mathrm{T}}\hat{\boldsymbol{\vartheta}}_k=$$
$$[f_1'(\hat{x}_k(t)),f_2'(\hat{x}_k(t)),\cdots,f_m'(\hat{x}_k(t))]\hat{\boldsymbol{\vartheta}}_k, \tag{135}$$
$$\boldsymbol{\varphi}(t)=[u(t),u(t-1),\cdots,u(t-n)]^{\mathrm{T}}. \tag{136}$$

当然,也可以利用最小二乘迭代搜索原理、牛顿迭代搜索等来求解优化问题 J_7,导出输出非线性系统的最小二乘迭代辨识方法和牛顿迭代辨识方法;也可以定义不同的准则函数,从而导出输出非线性系统的多新息辨识方法,输出非线性系统的递推最小二乘辨识方法,牛顿递推辨识方法等. 非线性系统辨识方法,以及上述算法收敛因子的选择都是极其困难的研究课题.

5 结语

系统辨识是研究建立系统数学模型的理论与方法. 迭代辨识方法是系统辨识的一个重要分支. 本文主要讨论了 CARMA 和 Box-Jenkins 伪线性系统的最小二乘迭代辨识方法与梯度迭代辨识方法. 这些方法也可推广到其他所有方程误差类系统和输出误差类系统. 对于非线性系统,特别是输入非线性系统、输出非线性系统和反馈非线性系统,如果是双线性参数非线性模型结构,那么它们就通过过参数化(o-

ver-parameterization)方法,化为过参数化线性回归模型,线性系统的辨识方法都可使用,否则必须使用梯度搜索方法、最小二乘搜索方法、牛顿搜索方法来研究其辨识问题.

参考文献
References

[1] 丁锋. 系统辨识(1):辨识导引[J]. 南京信息工程大学学报:自然科学版,2011,3(1):1-22
DING Feng. System identification. Part A:Introduction to the identification [J]. Journal of Nanjing University of Information Science & Technology:Natural Science Edition,2011,3(1):1-22

[2] 丁锋. 系统辨识(2):系统描述的基本模型[J]. 南京信息工程大学学报:自然科学版,2011,3(2):97-117
DING Feng. System identification. Part B:Basic models for system description [J]. Journal of Nanjing University of Information Science & Technology:Natural Science Edition,2011,3(2):97-117

[3] 丁锋. 系统辨识(3):辨识精度与辨识基本问题[J]. 南京信息工程大学学报:自然科学版,2011,3(3):193-226
DING Feng. System identification. Part C:Identification accuracy and basic problems [J]. Journal of Nanjing University of Information Science & Technology:Natural Science Edition,2011,3(3):193-226

[4] 丁锋. 系统辨识(4):辅助模型辨识思想与方法[J]. 南京信息工程大学学报:自然科学版,2011,3(4):289-318
DING Feng. System identification. Part D:Auxiliary model identification idea and methods [J]. Journal of Nanjing University of Information Science & Technology:Natural Science Edition,2011,3(4):289-318

[5] 丁锋. 基于输出估计的多输入系统随机梯度估计算法[J]. 南京信息工程大学学报:自然科学版,2010,2(6):481-488
DING Feng. Stochastic gradient estimation algorithm for multiple-input systems based on the output estimation [J]. Journal of Nanjing University of Information Science & Technology:Natural Science Edition,2010,2(6):481-488

[6] Ding F,Chen T. Identification of Hammerstein nonlinear ARMAX systems [J]. Automatica,2005,41(9):1479-1489

[7] Ding F, Shi Y, Chen T. Gradient-based identification methods for Hammerstein nonlinear ARMAX models [J]. Nonlinear Dynamics,2006,45(1/2):31-43

[8] 丁锋. 系统辨识理论方法[M]. 北京:电力出版社,2012
DING Feng. System identification theory and methods [M]. Beijing:China Electric Power Press,2012

[9] Ding F,Chen T. Hierarchical gradient-based identification of multivariable discrete-time systems [J]. Automatica,2005,41(2):315-325

[10] Ding F,Chen T. Hierarchical least squares identification methods for multivariable systems [J]. IEEE Transactions on Automatic Control,2005,50(3):397-402

[11] Ding F,Liu X P,Liu G. Gradient based and least-squares based iterative identification methods for OE and OEMA systems [J]. Digital Signal Processing,2010,20(3):664-677

[12] Ding F,Liu X P,Liu G. Identification methods for Hammerstein nonlinear systems [J]. Digital Signal Processing,2011,21(2):215-238

[13] Wang D Q,Ding F. Least squares based and gradient based iterative identification for Wiener nonlinear systems [J]. Signal Processing,2011,91(5):1182-1189

[14] Ding F,Chen T. Gradient based iterative algorithms for solving a class of matrix equations [J]. IEEE Transactions on Automatic Control,2005,50(8):1216-1221

[15] Ding F,Liu X P,Ding J. Iterative solutions of the generalized Sylvester matrix equations by using the hierarchical identification principle [J]. Applied Mathematics and Computation,2008,197(1):41-50

[16] Ding F,Chen T. On iterative solutions of general coupled matrix equations [J]. SIAM Journal on Control and Optimization,2006,44(6):2269-2284

[17] Ding F,Chen T. Iterative least squares solutions of coupled Sylvester matrix equations [J]. Systems & Control Letters,2005,54(2):95-107

[18] 袁平. 多变量系统辨识方法比较研究[D]. 无锡:江南大学物联网工程学院,2008
YUAN Ping. Comparisions and studies of identification methods for multivariable systems[D]. Wuxi:School of Internet of Things Engineering,Jiangnan University,2008

[19] 王金海,丁锋. CARMA 模型离线最小二乘迭代辨识方法[J]. 科学技术与工程,2007,7(23):5998-6003
WANG Jinhai,DING Feng. Least-squares-iterative identification algorithms for CARMA models [J]. Science Technology and Engineering,2007,7(23):5998-6003

[20] 陈晓伟,丁锋. 动态调节模型的最小二乘迭代辨识方法[J]. 科学技术与工程,2007,7(23):5994-5997
CHEN Xiaowei,DING Feng. Least-squares-iterative identificationmethods for dynamical adjusting models [J]. Science Technology and Engineering,2007,7(23):5994-5997

[21] Bao B,Xu Y Q,Sheng J,Ding R F. Least squares based iterative parameter estimation algorithm for multivariable controlled ARMA systems modelling with finite measurement data [J]. Mathematical and Computer Modelling,2011,53(9/10):1664-1669

[22] Ding F,Liu Y J,Bao B. Gradient based and least squares based iterative estimation algorithms for multi-input multi-output systems [J]. Journal of Systems and Control Engineering,2011,DOI:10. 1177/0959651811409491

[23] Wang D Q. Least squares-based recursive and iterative estimation for output error moving average (OEMA) systems using data filtering [J]. IET Control Theory and Applications,2011,5(14):1648-1657

[24] Liu Y J,Wang D Q,Ding F. Least-squares based iterative algorithms for identifying Box-Jenkins models with finite measurement data [J]. Digital Signal Processing,2010,20(5):1458-1467

[25] Wang D Q, Yang G W, Ding F. Gradient-based iterative parameter estimation for Box-Jenkins systems with finite measurement data [J]. Computers & Mathematics with Applications, 2010, 60(5): 1200-1208

[26] Wang L Y, Ding F, Liu X P. Consistency of HLS estimation algorithms for MIMO ARX-like systems [J]. Applied Mathematics and Computation, 2007, 190(2): 1081-1093

[27] 蒋红霞,王金海,丁锋. 一类非均匀采样系统的迭代最小二乘辨识[J]. 系统工程与电子技术,2008,30(8):1535-1539
JIANG Hongxia, WANG Jinhai, DING Feng. Least-squares-iterative identification for a class of non-uniformly sampled-data systems [J]. Systems Engineering and Electronics, 2008, 30(8): 1535-1539

[28] 陈晓伟,丁锋. 有色噪声系统的迭代辨识与递推辨识方法仿真比较研究[J]. 系统仿真学报,2008,20(21):5758-5762
CHEN Xiaowei, DING Feng. Comparison of iterative and recursive identification for systems with colored noises [J]. Journal of System Simulation, 2008, 20(21): 5758-5762

[29] 陆静,张彩霞,丁锋. 双输入多率输出误差系统最小二乘迭代辨识[J]. 科学技术与工程,2008,8(16):4683-4686
LU Jing, ZHANG Caixia, DING Feng. Lest-squares-iterative identification for two-input multirate output-error systems [J]. Science Technology and Engineering, 2008, 8(16): 4683-4686

[30] Liu X G, Lu J. Least squares based iterative identification for a class of multirate systems [J]. Automatica, 2010, 46(3): 549-554

[31] Xie L, Yang H Z. Gradient based iterative identification for non-uniform sampled output error systems [J]. Journal of Vibration and Control, 2011, 17(3): 471-478

[32] Han H Q, Xie L, Ding F, et al. Hierarchical least squares based iterative identification for multivariable systems with moving average noises [J]. Mathematical and Computer Modelling, 2010, 51(9/10): 1213-1220

[33] Zhang Z N, Ding F, Liu X G. Hierarchical gradient based iterative parameter estimation algorithm for multivariable output error moving average systems [J]. Computers & Mathematics with Applications, 2011, 61(3): 672-682

System identification. Part E: Iterative search principle and identification methods

DING Feng[1,2,3]

1 School of Internet of Things Engineering, Jiangnan University, Wuxi 214122

2 Control Science and Engineering Research Center, Jiangnan University, Wuxi 214122

3 Key Laboratory of Advanced Process Control for Light Industry (Ministry of Education), Jiangnan University, Wuxi 214122

Abstract Recursive identification and iterative identification are two important parameter estimation methods. The recursive index in the recursive identification is a time variable and the recursive identification can be used for on-line estimating system parameters; the iterative index in the iterative identification is a natural number and independent of time and the iterative identification is generally used for off-line estimating system parameters. The auxiliary model identification idea, multi-innovation identification theory, hierarchical identification principle and coupling identification concept based methods can be realized through recursive algorithms and iterative algorithms. Iterative methods can be traced to hundreds of years ago Jacobi iteration and Guass-Seidel iteration for solving the matrix equations $\boldsymbol{A}\boldsymbol{x} = b$. Iterative identification methods are based on the gradient search, least-squares search and Newton search principle. This paper studies the least squares based and gradient based iterative identification methods for CARMA systems and Box-Jenkins systems. The propsed methods can also be extended to other equation error type systems, output error type systems and nonlinear systems. Iterative methods usually apply system identification with finite data and their convergence analysis is very difficult and is a challenging research topic.

Key words iterative indentification; recursive identification; parameter estimation; FIR model; CAR model; CARMA model; CARAR model; CARARMA model; output error models; OEMA model; OEAR model; auxiliary model identification; multi-innovation identification; hierarchical identification; coupled identification