Second-order optimization methods for time-delayed ARX models: Nature gradient descent method and its two modified methods

Jing Chen[*,a], Yan Pu[a], Liuxiao Guo[a], Junfeng Cao[a], Quanmin Zhu[b]

[a]*School of Science, Jiangnan University, Wuxi 214122, PR China*
[b]*Department of Engineering Design and Mathematics, University of the West of England, Bristol BS16 1QY, UK*

Abstract

This paper proposes several second-order optimization methods for time-delayed ARX models. Since the time-delay in the information vector makes the traditional identification algorithms be inefficient, a redundant rule based method is utilized to transform the model into a redundant model. Then, the nature gradient descent (NGD) algorithm is developed for such a model. To reduce the computational efforts of the NGD algorithm and to adaptively update each element in the parameter vector, two modified NGD algorithms are also presented. The simulation examples verify the effectiveness of the proposed algorithms.

Key words: ARX model, time-delay, nature gradient descent, adaptive gradient descent, momentum based method, convergence rate

1. Introduction

System identification plays an important role in control theory and engineering, because a robust controller or a reliable prediction for future dynamics is usually based on an accurate model of the system [1, 2, 3, 4]. Until now, the identification algorithms, both applied and theoretical, are mainly dominated by gradient descent (GD) algorithms, expectation maximization algorithms and least squares (LS) algorithms [5, 6, 7, 8, 9]. Compared with the GD algorithms, the LS algorithms have quicker convergence rates, but need to compute the inverse of a matrix. Therefore, the LS algorithms are inefficient for large scale systems, due to their prohibitive computational costs [10, 11, 12, 13].

GD algorithm is a kind of first-order optimization methods. The GD algorithm updates the parameters through a direction and a suitable step-length, that does not involve the matrix inversion calculation [14, 15]. However, the GD algorithm has slower convergence rates, that limits its wide use in application. Second-order optimization methods, which perform second-order derivatives, have quicker convergence rates for their strong theoretical properties in theory, e.g., the Newton algorithm, the LS algorithm [16, 17, 18]. Unfortunately, the second-order optimization methods require the matrix inversion calculation, thus have heavy computational efforts [19, 20, 21, 22]. Thanks to the development of computer science, the better hardware and software make the second-order optimization methods be feasible in system identification.

Recent existing second-order methods, including the Newton algorithm which involves the Hessian matrix (HM) calculation, the nature gradient descent (NGD) algorithm which is proposed in this paper, and Shampoo algorithm which bridges the gap between full matrix preconditioning and the diagonal version, are popular in deep neural networks [23, 24, 25, 26]. The NGD algorithm is based on the traditional GD algorithm and Kullback-Leibler divergence (KLD), which can estimate both the point and distribution estimates of the unknown parameters [25, 27]. The Fisher information matrix (FIM) and KLD used in the NGD algorithm can

[☆]This work is supported by the National Natural Science Foundation of China (No. 61973137), and the Natural Science Foundation of Jiangsu Province (No. BK20201339)

[*]Corresponding author

Email addresses: `chenjing1981929@126.com` (Jing Chen), `puyanedu@163.com` (Yan Pu), `guoliuxiao@jiangnan.edu.cn` (Liuxiao Guo), `caojunfeng982@163.com` (Junfeng Cao), `quan.zhu@uwe.ac.uk` (Quanmin Zhu)

guarantee the point and distribution estimates converge to their true values with the cost of more computational efforts. As we know, there exist a large amount of unknown parameters in neural network. The FIM or HM for these kinds of models is then infeasible to compute, store, or invert. This is why the second-order optimization methods are not widely used in neural network.

To get around this difficulty, one way is to use a simple matrix to approximate the FIM in the NGD algorithm or the HM in the Newton algorithm. The adaptive gradient descent (Adagrad) algorithm is a special kind of second-order optimization method, which uses a diagonal matrix to approximate the FIM [28]. Thus the computational efforts are reduced from $O(n^3)$ to $O(n)$. In the contrast, the convergence rates of the Adagrad algorithm become slower because the related elements are assumed to be independent. Inspired by the momentum method, we can use the adaptive momentum gradient descent algorithm (Adm) to increase the convergence rates, with almost the same computational costs [29]. The main contributions of our article include:

(1) Combine the redundant rule and the NGD algorithm for time-delayed ARX models, which can estimate the parameters and the time-delay in sequence.

(2) Use the Adagrad algorithm to replace the NGD algorithm, which can reduce the computational efforts.

(3) Apply the Adm algorithm for time-delayed ARX models, which has almost the same convergence rates as the NGD algorithm, and the same computational efforts as the Adagrad algorithm.

The rest of this paper is organized as follows. Section 2 provides the time-delayed ARX model. Section 3 discusses the NGD algorithm. Section 4 describes two modified NGD algorithms. In Section 5, two simulation examples are proposed to show the effectiveness of the proposed algorithms. Finally, Section 6 concludes this paper and gives future directions.

2. ARX model with time-delay

The ARX model with time-delay is written by

$$A(z)y(t) = B(z)u(t-\tau) + v(t), \tag{1}$$

where $u(t)$ and $y(t)$ are the input and output, respectively, $v(t)$ is a Gaussian white noise satisfies $v(t) \sim N(0, \sigma^2)$. τ is the unknown time-delay, $A(z)$ and $B(z)$ are polynomials, defined as

$$\begin{aligned} A(z) &= 1 + a_1 z^{-1} + \cdots + a_n z^{-n}, \\ B(z) &= b_1 z^{-1} + \cdots + b_m z^{-m}, \end{aligned}$$

where $z^{-i}u(t) = u(t-i)$.

To simplify the ARX model, let

$$\begin{aligned} \chi(t) &= [-y(t-1), \cdots, -y(t-n), u(t-1-\tau), \cdots, u(t-m-\tau)]^{\mathrm{T}}, \\ \vartheta &= [a_1, \cdots, a_n, b_1, \cdots, b_m]^{\mathrm{T}}. \end{aligned}$$

The information vector $\chi(t)$ contains unknown time-delay, which makes $\chi(t)$ be unavailable in parameter estimation. To deal with this problem, a redundant rule method is introduced [30].

Assume that the upper bound of the delay is M, then the redundant information vector and the sparse parameter vector are written by

$$\begin{aligned} \phi(t) &= [-y(t-1), \cdots, -y(t-n), u(t-1), u(t-2), \cdots, u(t-\tau-1), \cdots, u(t-\tau-m), \cdots, u(t-m-M)]^{\mathrm{T}}, \\ \alpha &= [a_1, \cdots, a_n, \alpha_1, \alpha_2, \cdots, \alpha_{\tau+1}, \cdots, \alpha_{\tau+m}, \cdots, \alpha_{m+M}]^{\mathrm{T}}. \end{aligned}$$

Clearly, the information blocks $[u(t-1), u(t-2), \cdots, u(t-\tau)]$ and $[u(t-\tau-m-1), \cdots, u(t-m-M)]$ are redundant, and the parameter blocks $[\alpha_1, \alpha_2, \cdots, \alpha_\tau]$ and $[\alpha_{\tau+m+1}, \cdots, \alpha_{m+M}]$ are zero vectors.

The time-delayed ARX model can be transformed into a redundant model

$$y(t) = \phi^{\mathrm{T}}(t)\alpha + v(t). \tag{2}$$

Using the SG algorithm to estimate the parameters yields

$$\alpha(t) = \alpha(t-1) + r(t)\phi(t)[y(t) - \phi^{\mathrm{T}}(t)\alpha(t)], \tag{3}$$

where $\phi(t)[y(t) - \phi^{\mathrm{T}}(t)\alpha(t)]$ is the direction termed as negative gradient, and $r(t)$ is the corresponding step-length.

For the sparse parameter vector, the SG algorithm has the following disadvantages:

(1) When the number of the unknown parameters is large (has little prior knowledge of the upper bound of the time-delay, thus should assign a large M), the SG algorithm would have slow convergence rates.

(2) Use one step-length for the sparse parameter vector, the smaller elements in the parameter vector will shake seriously, while the larger ones will have slow convergence rates.

3. Nature gradient descent method

To increase the convergence rates of the SG algorithm, a second-order optimization method termed as NGD algorithm is proposed in this section.

3.1. Score function and Fisher information matrix

At the sampling instant t, we aim to maximize the likelihood function

$$p(y(t)|u(t),\alpha) = \frac{1}{\sqrt{2\pi}}\exp\left[-\frac{(y(t)-\phi^{\mathrm{T}}(t)\alpha)^2}{2\sigma^2}\right].$$

Define the score function as

$$s(\alpha) = \frac{\partial \log p(y(t)|u(t),\alpha)}{\partial \alpha} = \nabla_\alpha \log p(y(t)|u(t),\alpha). \tag{4}$$

Lemma 1: The expected value of score function $s(\alpha)$ with respect to $p(y(t)|u(t),\alpha)$ is equal to zero.

Proof: Taking the expectation on $s(\alpha)$ with respect to $p(y(t)|u(t),\alpha)$ yields

$$\begin{aligned}
E_{p(y(t)|u(t),\alpha)}[s(\alpha)] &= \int p(y(t)|u(t),\alpha)s(\alpha)d\alpha = \int p(y(t)|u(t),\alpha)\nabla_\alpha \log p(y(t)|u(t),\alpha)d\alpha \\
&= \int p(y(t)|u(t),\alpha)\frac{\nabla_\alpha p(y(t)|u(t),\alpha)}{p(y(t)|u(t),\alpha)}d\alpha \\
&= \int \nabla_\alpha p(y(t)|u(t),\alpha)d\alpha = 0.
\end{aligned}$$

Therefore, Lemma 1 is obtained. ■

The Fisher information matrix is defined as

$$F = E_{p(y(t)|u(t),\alpha)}[s(\alpha)s^{\mathrm{T}}(\alpha)] = \int p(y(t)|u(t),\alpha)\nabla_\alpha \log p(y(t)|u(t),\alpha)\nabla_\alpha \log p(y(t)|u(t),\alpha)^{\mathrm{T}}d\alpha. \tag{5}$$

In application, it is difficult to compute F. We usually use $\frac{1}{L}\sum\limits_{t=1}^{L}\nabla_\alpha \log p(y(t)|u(t),\alpha)\nabla_\alpha \log p(y(t)|u(t),\alpha)^{\mathrm{T}}$ to approximate F.

3.2. Nature gradient descent algorithm

In the nature gradient descent algorithm, we define the following cost function

$$J(\nabla) = \frac{1}{2}[y(t) - \phi^{\mathrm{T}}(t)\alpha]^2 + \lambda[KL[p(y(t)|u(t),\alpha)|p(y(t)|u(t),\alpha+R)] - c],$$

where c is a small positive constant and R is the direction.

Remark 1: The above cost function can guarantee both the point estimates and the distribution estimates to converge to their true values.

To get a simplified $J(\nabla)$, the second term in the above equation should be transformed into a concise structure. Simplify $KL[p(y(t)|u(t),\alpha)|p(y(t)|u(t),\alpha+R)]$ as,

$$KL[p(y(t)|u(t),\alpha)|p(y(t)|u(t),\alpha+R)] \approx KL[p(y(t)|u(t),\alpha)|p(y(t)|u(t),\alpha)] + \\ \frac{\partial KL[p(y(t)|u(t),\alpha)|p(y(t)|u(t),\alpha)]}{\partial \alpha'}|_{\alpha'=\alpha}R + \frac{1}{2}R^{\mathrm{T}}\frac{\partial^2 KL[p(y(t)|u(t),\alpha)|p(y(t)|u(t),\alpha)]}{\partial \alpha'^2}|_{\alpha'=\alpha}R. \tag{6}$$

Since

$$KL[p(y(t)|u(t),\alpha)|p(y(t)|u(t),\alpha) = 0,$$

Equation (6) is written as

$$KL[p(y(t)|u(t),\alpha)|p(y(t)|u(t),\alpha+R) \approx \frac{\partial KL[p(y(t)|u(t),\alpha)|p(y(t)|u(t),\alpha)]}{\partial \alpha'}|_{\alpha'=\alpha}R \\ +\frac{1}{2}R^{\mathrm{T}}\frac{\partial^2 KL[p(y(t)|u(t),\alpha)|p(y(t)|u(t),\alpha)]}{\partial \alpha'^2}|_{\alpha'=\alpha}R. \tag{7}$$

The first term on the right side of the above equation is

$$\frac{\partial KL[p(y(t)|u(t),\alpha)|p(y(t)|u(t),\alpha)]}{\partial \alpha'}|_{\alpha'=\alpha}R = \nabla_{\alpha'}\int p(y(t)|u(t),\alpha)\log\frac{p(y(t)|u(t),\alpha)}{p(y(t)|u(t),\alpha')}d\alpha' R$$
$$= \left[\nabla_{\alpha'}\int p(y(t)|u(t),\alpha)\log p(y(t)|u(t),\alpha)d\alpha' - \nabla_{\alpha'}\int p(y(t)|u(t),\alpha)\log p(y(t)|u(t),\alpha')d\alpha'\right]R$$
$$= -\nabla_{\alpha'}\int p(y(t)|u(t),\alpha)\log p(y(t)|u(t),\alpha')d\alpha' R$$
$$= -\int p(y(t)|u(t),\alpha)\nabla_{\alpha'}\log p(y(t)|u(t),\alpha')d\alpha' R$$
$$= -\int p(y(t)|u(t),\alpha)\frac{\nabla_{\alpha'}p(y(t)|u(t),\alpha')}{p(y(t)|u(t),\alpha')}d\alpha' R.$$

When $\alpha' \to \alpha$, according to Lemma 1, we have

$$\frac{\partial KL[p(y(t)|u(t),\alpha)|p(y(t)|u(t),\alpha)]}{\partial \alpha'}|_{\alpha'=\alpha}R = 0. \tag{8}$$

The second term on the right side of Equation (7) is

$$\frac{1}{2}R^{\mathrm{T}}\frac{\partial^2 KL[p(y(t)|u(t),\alpha)|p(y(t)|u(t),\alpha)]}{\partial \alpha'^2}|_{\alpha'=\alpha}R = -\frac{1}{2}R^{\mathrm{T}}\nabla_{\alpha'}\int p(y(t)|u(t),\alpha)\frac{\nabla_{\alpha'}p(y(t)|u(t),\alpha')}{p(y(t)|u(t),\alpha'}d\alpha' R$$
$$= -\frac{1}{2}R^{\mathrm{T}}\int p(y(t)|u(t),\alpha)\frac{\nabla^2_{\alpha'}p(y(t)|u(t),\alpha')p(y(t)|u(t),\alpha') - \nabla_{\alpha'}p(y(t)|u(t),\alpha')\nabla_{\alpha'}p(y(t)|u(t),\alpha')^{\mathrm{T}}}{p^2(y(t)|u(t),\alpha')}d\alpha' R$$
$$= -\frac{1}{2}R^{\mathrm{T}}\int p(y(t)|u(t),\alpha)\frac{\nabla^2_{\alpha'}p(y(t)|u(t),\alpha')}{p(y(t)|u(t),\alpha')}d\alpha' R + \frac{1}{2}R^{\mathrm{T}}\int p(y(t)|u(t),\alpha)\frac{\nabla_{\alpha'}p(y(t)|u(t),\alpha')\nabla_{\alpha'}p(y(t)|u(t),\alpha')^{\mathrm{T}}}{p^2(y(t)|u(t),\alpha')}d\alpha' R.$$

When $\alpha' \to \alpha$, we have

$$\int p(y(t)|u(t),\alpha)\frac{\nabla^2_{\alpha'}p(y(t)|u(t),\alpha')}{p(y(t)|u(t),\alpha')}d\alpha' = \int \nabla^2_{\alpha'}p(y(t)|u(t),\alpha')d\alpha' = \nabla^2_{\alpha'}\int p(y(t)|u(t),\alpha')d\alpha' = 0. \tag{9}$$

According to Equation (5), it gives rise to

$$\int p(y(t)|u(t),\alpha)\frac{\nabla_{\alpha'}p(y(t)|u(t),\alpha')\nabla_{\alpha'}p(y(t)|u(t),\alpha')^{\mathrm{T}}}{p^2(y(t)|u(t),\alpha')}d\alpha' \\ = \int p(y(t)|u(t),\alpha)\nabla_{\alpha}\log p(y(t)|u(t),\alpha)\nabla_{\alpha}\log p(y(t)|u(t),\alpha)^{\mathrm{T}}d\alpha = F. \tag{10}$$

Theorem 1: For the ARX model proposed in (1), the Fisher information matrix is expressed by Equation (5) and the score function is defined in Equation (4), then the Kullback-Leibler divergence satisfies

$$KL[p(y(t)|u(t),\alpha)|p(y(t)|u(t),\alpha+R) = \frac{1}{2}RFR^{\mathrm{T}}.$$

(According to Equations (8)-(10), the proof is straightforward.)

The cost function of the NGD algorithm is transformed into

$$\begin{aligned}J(\alpha(t-1)+R) &= \frac{1}{2}[y(t)-\phi^{\mathrm{T}}(t)(\alpha(t-1)+R)]^2+\lambda[\frac{1}{2}RFR^{\mathrm{T}}-c]\\ &\approx \frac{1}{2}[y(t)-\phi^{\mathrm{T}}(t)\alpha(t-1)]^2-R^{\mathrm{T}}\phi(t)[y(t)-\phi^{\mathrm{T}}(t)\alpha(t-1)]+\lambda[\frac{1}{2}RFR^{\mathrm{T}}-c].\end{aligned}$$

Performing the derivative of $J(\alpha(t-1)+R)$ with respect to R and then equating it to zero yield

$$R=\frac{1}{\lambda}F^{-1}\phi(t)[y(t)-\phi^{\mathrm{T}}(t)\alpha(t-1)]. \tag{11}$$

Combing the step-length, the direction at the sampling instant t is normalized as

$$R(t)=F^{-1}\phi(t)[y(t)-\phi^{\mathrm{T}}(t)\alpha(t-1)]. \tag{12}$$

Then, the NGD algorithm for the time-delayed ARX model is summarized as follows

$$\begin{aligned}&\alpha(t)=\alpha(t-1)+\gamma R(t),\ \gamma=0.001,\\ &R(t)=F^{-1}(t)\phi(t)[y(t)-\phi^{\mathrm{T}}(t)\alpha(t-1)],\\ &F(t)=\frac{1}{t}\sum_{i=1}^{t}\nabla_\alpha \log p(y(i)|u(i),\alpha)\nabla_\alpha \log p(y(i)|u(i),\alpha)^{\mathrm{T}}.\end{aligned}$$

The NGD algorithm starts with the following steps:

```
Nature gradient descent algorithm
Initialize $\alpha(0)$ and threshold $\varrho$
  for t=1:L
    Collect $u(t)$, $y(t)$
    Compute $\phi(t)[y(t)-\phi^{\mathrm{T}}(t)\alpha(t-1)]$
    Compute $F(t)$
    Compute $R(t)$
    Update $\alpha(t)$
    Compare each element in $\alpha(t)$ with $\varrho$
    Get $\tau(t)$
  end
until convergence
```

Lemma 2 [25]: The negative expected Hessian of log likelihood is equal to the Fisher Information Matrix F.

Remark 2: Lemma 2 demonstrates that the NGD algorithm can be regarded as a modified Newton algorithm. Like the Newton algorithm, the computational efforts of the NGD algorithm are heavy because of the matrix inversion calculation.

Remark 3: Although the NGD algorithm can estimate the parameters, but some small elements in the parameter vector may have poor estimation accuracy, because all the elements are not adaptively/separately estimated.

4. Modified NGD algorithms

The NGD algorithm is a second-order optimization method which has quicker convergence rates but heavy computational efforts. As we know, F is a full matrix, the major computational cost of the NGD algorithm is caused by the matrix inversion calculation. If F is a diagonal matrix, the computational efforts will be decreased intensively. In this section, two modified NGD/second-order algorithms are provided which can overcome the shortcomings of the traditional NGD algorithm.

4.1. Adaptive gradient descent algorithm

Rewrite the parameter vector as

$$\alpha = [a_1, \cdots, a_n, \alpha_1, \alpha_2, \cdots, \alpha_{\tau+1}, \cdots, \alpha_{\tau+m}, \cdots, \alpha_{m+M}]^{\mathrm{T}}.$$

Some elements in the parameter vector are equal to zero, then the NGD algorithm will be inefficient for those small elements. Naturally, a question arises: can each element be updated adaptively.

Assume that the parameter vector estimate at the sampling instant $t-1$ is

$$\alpha(t-1) = [a_1(t-1), \cdots, a_n(t-1), \alpha_1(t-1), \alpha_2(t-1), \cdots, \alpha_{\tau+1}(t-1), \cdots, \alpha_{\tau+m}(t-1), \cdots, \alpha_{\tau+M}(t-1)]^{\mathrm{T}}.$$

Then, the element a_1 at the sampling instant t can be updated by the following cost function

$$J(a_1, \bar{\alpha}_{a_1(t-1)}(t-1)) = \frac{1}{2}[y(t) - a_1 y(t-1) - \bar{\phi}^{\mathrm{T}}_{y(t-1}(t-1)\bar{\alpha}_{a_1(t-1)}]^2,$$

where

$$\bar{\alpha}_{a_1(t-1)}(t-1) = [a_2(t-1), \cdots, a_n(t-1), \alpha_1(t-1), \alpha_2(t-1), \cdots, \alpha_{\tau+1}(t-1), \cdots, \alpha_{\tau+m}(t-1), \cdots, \alpha_{\tau+M}(t-1)]^{\mathrm{T}},$$
$$\bar{\phi}^{\mathrm{T}}_{y(t-1}(t-1) = [-y(t-2), \cdots, -y(t-n), u(t-1), u(t-2), \cdots, u(t-\tau-1), \cdots, u(t-\tau-m), \cdots, u(t-m-M)]^{\mathrm{T}}.$$

Then, the adaptive gradient descent (Adagrad) algorithm for the ARX model is written by

$$a_i(t) = a_i(t-1) + \frac{\gamma y(t-i)}{\sqrt{s_i(t)+c}}[y(t) - a_i(t-1)y(t-i) - \bar{\phi}^{\mathrm{T}}_{y(t-i)}(t-1)\bar{\alpha}_{a_i(t-1)}], \quad i = 1, \cdots, n, \ \gamma = 0.001, \tag{13}$$

$$\alpha_j(t) = \alpha_j(t-1) + \frac{\gamma u(t-j)}{\sqrt{s_{n+j}(t)+c}}[y(t) - \alpha_j(t-1)u(t-j) - \bar{\phi}^{\mathrm{T}}_{u(t-j)}(t-1)\bar{\alpha}_{\alpha_j(t-1)}], \quad j = 1, \cdots, m+M, \tag{14}$$

$$s_i(t) = \beta s_i(t-1) + (1-\beta)\{y(t-i)[y(t) - a_i(t-1)y(t-i) - \bar{\phi}^{\mathrm{T}}_{y(t-i)}(t-1)\bar{\alpha}_{a_i(t-1)}]\}^2, \quad \beta = 0.999, \ s_i(0) = 0, \tag{15}$$

$$s_{n+j}(t) = \beta s_{n+j}(t-1) + (1-\beta)\{u(t-j)[y(t) - \alpha_j(t-1)u(t-j) - \bar{\phi}^{\mathrm{T}}_{u(t-j)}(t-1)\bar{\alpha}_{\alpha_j(t-1)}]\}^2, \quad s_{n+j}(0) = 0, \tag{16}$$

where c is a small constant.

Remark 4: In the Adagrad algorithm, each element in the parameter vector is updated adaptively, that means each element has its own direction and step-length. Thus, the disadvantages of difference of parameter magnitude can be overcome by using the Adagrad algorithm.

4.2. The relationship between the Adagrad algorithm and the NGD algorithm

Define

$$S = \begin{bmatrix} \sqrt{s_1(t)+c} & 0 & \cdots & 0 \\ 0 & \sqrt{s_2(t)+c} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sqrt{s_{n+m+M}(t)+c} \end{bmatrix}. \tag{17}$$

According to Equations (13)-(16), the Adagrad algorithm is equivalent to

$$\alpha(t) = \alpha(t-1) + \gamma S^{-1}R(t),$$
$$R(t) = \phi(t)[y(t) - \phi^{\mathrm{T}}(t)\alpha(t-1)].$$

Clearly, the Adagrad algorithm has the same structure as the NGD algorithm. However, the Adagrad algorithm differs from the NGD algorithm in two ways:

(1) The matrix S in the Adagrad is a diagonal matrix, thus the Adagrad algorithm has less computational efforts. On the other hand, the matrix F in the NGD algorithm is a full matrix, which considers the relationships of each element, thus the NGD algorithm has quicker convergence rates than the Adagrad algorithm.

(2) The step-lengths of the Adagrad algorithm become smaller and smaller with the increased number of t. Therefore, the estimation variances of the Adagrad algorithm are smaller than those of the NGD algorithm.

4.3. The adaptive momentum gradient descent algorithm

The Adagrad algorithm can be regarded as a modified NGD algorithm. It has less computational efforts but slower convergence rates when compared with the NGD algorithm. To increase the convergence rates of the Adagrad algorithm, an adaptive momentum gradient descent (Adm) algorithm is proposed in this subsection.

The basic idea of the momentum method is to combine the previous direction with the current one, and then to update the parameters using this combined direction [31]. By using the momentum method, the convergence rates can be increased.

In the adaptive momentum gradient descent (Adm) algorithm, the direction of each element is written as

$$R_{a_i(t)}(t)=\eta R_{a_i(t-1)}(t-1)+(1-\eta)y(t-i)[y(t)-a_i(t-1)y(t-i)-\bar{\phi}^{\mathrm{T}}_{y(t-i)}(t-1)\bar{\alpha}_{a_i(t-1)}],\ \eta=0.9,$$

and then the parameter $a_i(t)$ can be updated by

$$a_i(t)=a_i(t-1)+\frac{\gamma}{\sqrt{s_i(t)+c}}R_{a_i(t)}(t),$$

where γ is usually assigned as 0.001.

The Adm algorithm starts with the following steps:

Adaptive momentum gradient descent algorithm

```
Initialize $\alpha(0)$ $s_i(0)$, $d_i(0)$, $i=1,2,\cdots,M+n+m$, and threshold $\varrho$
  for t=1:L
    Collect $u(t)$, $y(t)$
    Compute $\phi(t)[y(t)-\phi^{\mathrm{T}}(t)\alpha(t-1)]$
    Compute $R_i(t)$ and $s_i(t)$
    Compute $a_1(t),\cdots,a_n(t)$ and $\alpha_1(t),\cdots,\alpha_{m+M}(t)$
    Form $\alpha(t)$
    Compare element in $\alpha(t)$ with $\varrho$
    Get $\tau(t)$
  end
until convergence
```

Remark 5: The Adm method uses the momentum technique, thus it has quicker convergence rates than the Adagrad algorithm.

Remark 6: Both the Adm algorithm and Adagrad algorithm are second-order optimization methods, thus they have quicker convergence rates than the first-order optimization methods, such as the SG algorithm and gradient iterative algorithm. In addition, these two second-order optimization methods avoid the matrix inversion calculation, thus they have less computational efforts than the NGD algorithm and Newton algorithm.

The properties of the SG, NGD, Adagrad and Adm algorithms are summarized in Table 1.

Table 1: Properties of the four algorithms

Algorithms	SG	NGD	Adagrad	Adm
Order	First-order	Second-order	Second-order	Second-order
Computational efforts	The smallest	The heaviest	Smaller	Heavier
Convergence rates	The slowest	The fastest	Slower	Faster
Matrix inversion	None	Require	None	None
Adaptively	Unable	Unable	Enable	Enable

(**Smaller**: smaller than Adm but heavier than SG; **Heavier**: smaller than NGD but heavier than Adagrad; **Faster**: faster than Adagrad but slower than NGD; **Slower**: slower than Adm but faster than SG; **Adaptively**: can estimate each element adaptively.)

5. Examples

5.1. Example 1–time-delayed ARX model

Consider an ARX model with time-delay

$$\begin{aligned} &A(z)y(t) = B(z)u(t-\tau) + v(t),\\ &\quad A(z) = 1 + a_1z^{-1} + a_2z^{-2} + a_3z^{-3} = 1 - 0.2z^{-1} + 0.2z^{-2} - 0.3z^{-3},\\ &\quad B(z) = b_1z^{-1} + b_2z^{-2} = 0.5z^{-1} + 0.17z^{-2}, \end{aligned}$$

the input signal $u(t)$ satisfies $u(t) \sim N(0,1)$, the noise $\{v(t)\}$ is taken as a white noise sequence with zero mean and variance $\sigma^2 = 0.10^2$. Assume the time-delay $\tau = 3$ and the upper bound of τ is $M = 6$.

The redundant model is written by

$$\begin{aligned} &y(t) = \phi^{\mathrm{T}}(t)\alpha + v(t),\\ &\phi(t) = [-y(t-1), -y(t-2), -y(t-3), u(t-1), u(t-2), u(t-3), u(t-4), u(t-5), u(t-6), u(t-7), u(t-8)]^{\mathrm{T}},\\ &\quad \alpha = [a_1, a_2, a_3, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7, \alpha_8]^{\mathrm{T}} = [0.2, -0.2, 0.3, 0, 0, 0, 0.5, 0.17, 0, 0, 0]^{\mathrm{T}}. \end{aligned}$$

Apply the SG, NGD, Adagrad, Adm algorithms for this redundant model, the parameter estimates and their estimations errors $\delta = \|\alpha(t) - \alpha\|/\|\alpha\|$ are shown in Table 2 and Figure 1.

From Table 2 and Figure 1, the following findings can be obtained.

(1) The NGD algorithm has the fastest convergence rates among these four algorithms, but has the largest estimation variances.

(2) Compared with the Adagrad algorithm, the Adm algorithm is more effective for its faster convergence rates and smaller estimation variances.

(3) The SG algorithm may converge to a suboptimal point because the step-length approaches to zero with the increased number of t.

Assign different thresholds for these four algorithms. If the absolute value of the estimate is smaller than the threshold, we can regard it as a redundant element and pick it out from the parameter vector, then the time-delay can be obtained. This is illustrated in Table 3.

Since the redundant elements are α_1, α_2, α_3, α_6, α_7, α_8, Table 3 shows that Adm is the most robust algorithm, and follows the NGD algorithm, then is the Adagrad algoritm, finally is the SG algorithm. Table 3 also demonstrates that the SG algorithm cannot pick out all the redundant elements whatever the thresholds are because of its poor estimation accuracy.

5.2. Example 2–ARX model with different orders of magnitude

In this example, an element in the parameter vector is much smaller than the others. The ARX model with different orders of magnitude is written by

$$\begin{aligned} &A(z)y(t) = B(z)u(t) + v(t),\\ &\quad A(z) = 1 + a_1z^{-1} + a_2z^{-2} + a_3z^{-3} = 1 - 0.2z^{-1} + 0.2z^{-2} - 0.3z^{-3},\\ &\quad B(z) = b_1z^{-1} + b_2z^{-2} = 0.5z^{-1} + 0.0017z^{-2}, \end{aligned}$$

the input signal $u(t)$ satisfies $u(t) \sim N(0,1)$, the noise $\{v(t)\}$ is taken as a white noise sequence with zero mean and variance $\sigma^2 = 0.05^2$.

Using the SG, NGD, Adagrad and Adm algorithms for the ARX model, the parameter estimates and the estimation errors are shown in Figure 2 and Table 4. Figure 2 shows that the second-order optimization methods (NGD, Adagrad, Adm) are more effective than the first-order optimization method (SG). Table 4 demonstrates that the Adagrad and Adm algorithms can get more accurate estimates of the smallest element b_2 than those of the NGD algorithm, that is because the Adagrad and Adm algorithms estimate the parameter elements adaptively.

6. Conclusions

Several second-order optimization methods are proposed for ARX models with time-delay. By using the redundant rule, the ARX model can be turned into a redundant model whose parameter vector is sparse. Then the NGD algorithm can estimate the parameters and the unknown time-delay quickly. Since a matrix inversion is involved in the NGD algorithm, two modified NGD algorithms: Adagrad and Adm algorithms are provided, where these two modified NGD algorithms do not require matrix inversion, and can estimate the parameter elements adaptively.

The second-order optimization methods, especially the two modified NGD algorithms, are more efficient for large-scale systems, and can be further extended to structure identification. However, there remain some problems need to be further discussed, e.g., the convergence properties of the second-order optimization methods, and the selection of an optimal threshold.

References

[1] Giri F, Bai EW. Block-Oriented Nonlinear System Identification. Springer, Berlin, 2010.

[2] Söderström T, Stoica P. Systen Identification. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[3] Ding F. System Identification-Iterative Search Principle and Identification Methods. Science Press, Beijing, 2018.

[4] Xiong JX, Pan J, Chen GY, et al. Sliding mode dual-channel disturbance rejection attitude control for a quadrotor. *IEEE Trans Ind Electron.* 2022; 69(10):10489-10499.

[5] Zhang X, Ding F. Adaptive parameter estimation for a general dynamical system with unknown states. *Int J Robust Nonlinear Control.* 2020; 30(4):1351-1372.

[6] Liu XP, Yang, XQ. Identification of nonlinear state-space systems with skewed measurement noises. *IEEE Transa Circuits Syst I.* 2022. DOI: 10.1109/TCSI.2022.3193444.

[7] Ding F, Liu XP, Liu G. Identification methods for Hammerstein nonlinear systems. *Digit Signal Process.* 2021; 21(2):215-238.

[8] Wang DQ, Zhang S, Gan M, Qiu JL. A novel EM identification method for Hammerstein systems with missing output data. *IEEE Trans Ind Inform.* 2020; 16(4):2500-2508.

[9] Billings SA, Zhu QM. Rational model identification using extended least squares algorithm. *Int J Control.* 1991; 54(3):529-546.

[10] Wang DQ, Yan YR, Liu YJ, Ding JH. Model recovery for Hammerstein systems using the hierarchical orthogonal matching pursuit method. *J Comput Appl Math.* 2019; 345:135-145.

[11] Gan M, Chen GY, Chen L, Chen CLP. Term selection for a class of nonlinear separable models. *IEEE Trans Neur Net Lear Syst.* 2020; 31(2):445-451.

[12] Mu BQ, Bai EW, Zheng WX, Zhu QM. A globally consistent nonlinear least squares estimator for identification of nonlinear rational systems. *Automatica.* 2017; 77:322-335.

[13] Na J, Xing YS, Castelo RC. Adaptive estimation of time-varying parameters with application to roto-magnet plant. *IEEE Trans Syst Man Cybern Syst.* 2021; 51(2):731-741.

[14] Xu L, Ding F, Zhu QM. Separable synchronous multi-innovation gradient-based iterative signal modeling from on-line measurements. *IEEE Trans Instrumd Meas.* 2022;71: 6501313.

[15] Chen J, Liu YJ, Zhu QM. Multi-step-length gradient iterative algorithm for equation-error type models. *Syst Control Lett.* 2018; 115:15-21.

[16] Dehghan M, Hajarian M. Fourth-order variants of Newton's method without second derivatives for solving non-linear equations. *Eng Computation.* 2012; 29(4):356-365.

[17] Liu MM, Xiao YS, Ding RF. Iterative identification algorithm for Wiener nonlinear systems using the Newton method. *Appl Math Model.* 2013; 37(9):6584-6591.

[18] Ding F, Deng KP, Liu XM. Decomposition based Newton iterative identification method for a Hammerstein nonlinear FIR system with ARMA noise. *Circuits Syst Signal Process.* 2014; 33(9):2881-2893.

[19] Xu L, Chen FY, Hayat T. Hierarchical recursive signal modeling for multi-frequency signals based on discrete measured data. *Int J Adapt Control Signal Process.* 2021; 35(5):676-693.

[20] You JY, Yu CP, Sun J, Chen J. Generalized maximum entropy based identification of graphical ARMA models. *Automatica.* 2022; 141:110319.

[21] Yu CP, Li Y, Fang H, Chen J. System identification approach for inverse optimal control of finite-horizon linear quadratic regulators. *Automatica.* 2021; 129:109636.

[22] Liu SY, Zhang X, Xu L, Ding F. Expectation-maximization algorithm for bilinear systems by using the Rauch-Tung-Striebel smoother. *Automatica.* 2022; 142:110365.

[23] Xu L. Application of the Newton iteration algorithm to the parameter estimation for dynamical systems. *J Comput Appl Math.* 2015; 288:33-43.

[24] James M. New insights and perspectives on the natural gradient method. *arXiv preprint arXiv.* 2014; 1412.1193.

[25] Alexander L, Verhagen J, Grasman, R et al. A tutorial on Fisher information. *J Math Psychol.* 2017; 80:40-55.

[26] Anil R, Gupta V, Koren T, et al. Second order optimization made practical. *https://arxiv.org/abs/2002.09018.* 2020.

[27] Belov DI, Armstrong RD. Distributions of the Kullback-Leibler divergence with applications. *Brit J Math Stat Psy.* 2011; 64:291-309.

[28] Duchi J, Hazan E, Singer Y. Adaptive subgradient methods for online learning and stochastic optimization. *J Mach Learn Res.* 2011; 12:2121-2159.

[29] Kingma DP, Jimmy B. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.* 2014.

[30] Chen J, Ma JX, et al. Identification methods for time-delay systems based on the redundant rules. *Signal Process.* 2017; 137:192-198.

[31] Shen HH, Li HW. A gradient approximation algorithm based weight momentum for restricted Boltzmann machine. *Neurocomputing.* 2019; 361:40-49.

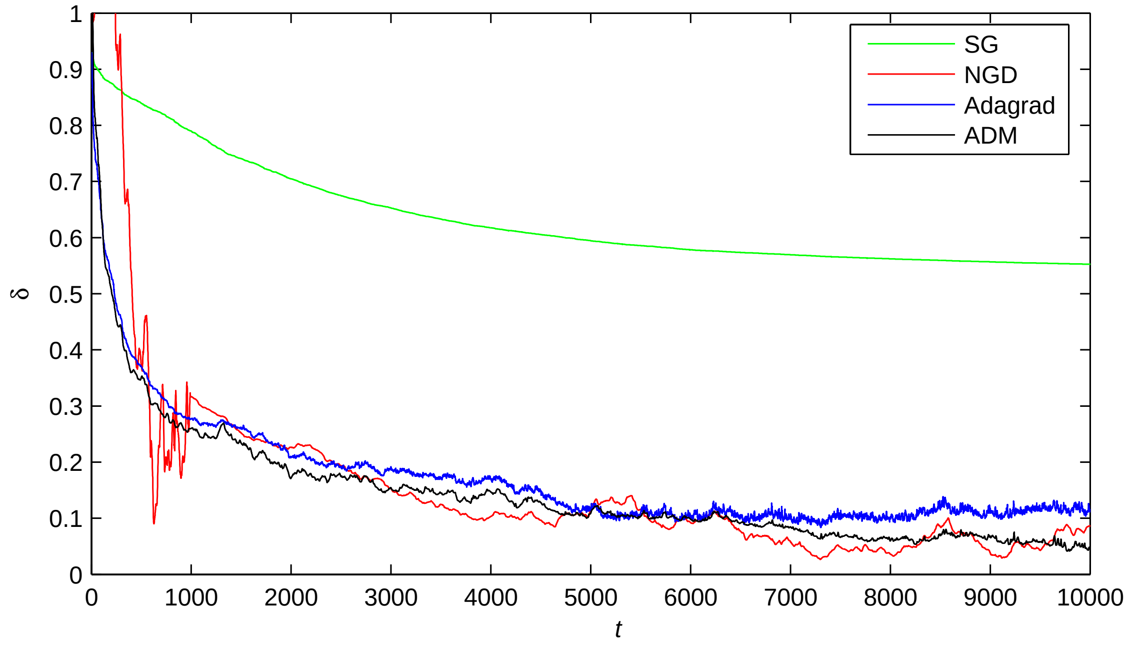

Figure 1: The parameter estimation errors

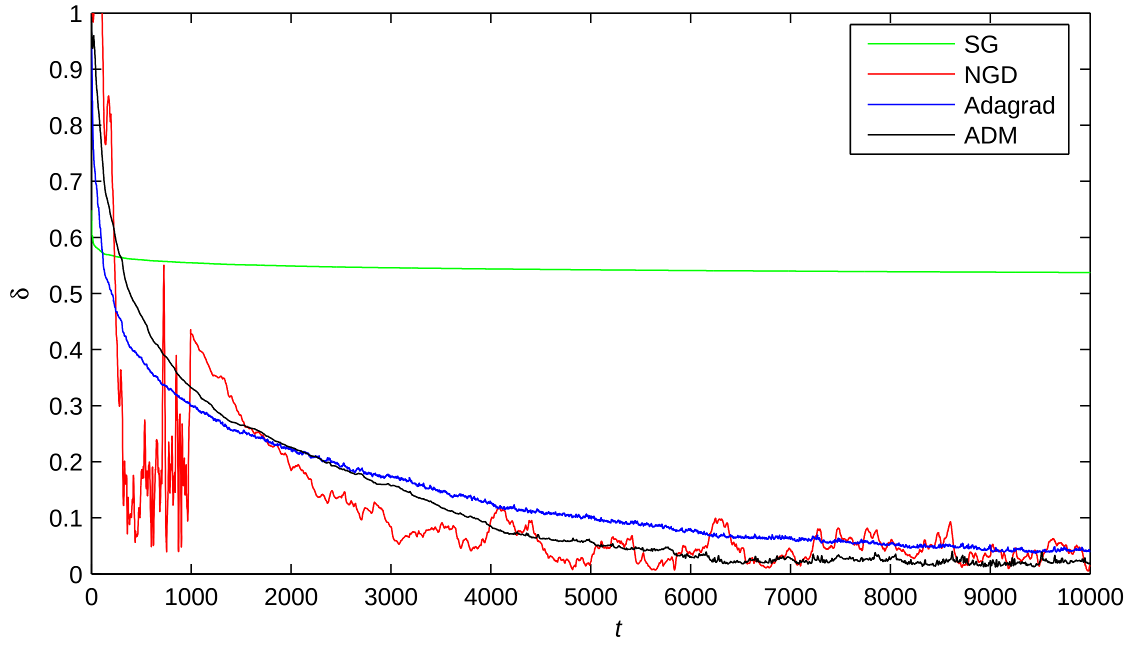

Figure 2: The parameter estimation errors

Table 2: The parameter estimates and their estimation errors

Algorithm	t	a_1	a_2	a_3	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	δ (%)
SG	1000	0.050	0.019	0.036	0.078	0.105	0.056	0.197	0.144	0.041	0.097	0.128	78.985
	2000	0.058	0.005	0.036	0.069	0.080	0.047	0.270	0.171	0.016	0.091	0.126	70.467
	4000	0.065	-0.010	0.034	0.043	0.052	0.030	0.361	0.201	-0.010	0.085	0.114	61.741
	6000	0.069	-0.020	0.034	0.024	0.032	0.022	0.417	0.217	-0.026	0.082	0.106	57.830
	8000	0.070	-0.025	0.038	0.019	0.024	0.014	0.450	0.225	-0.036	0.083	0.104	56.213
	10000	0.070	-0.028	0.040	0.012	0.017	0.010	0.470	0.230	-0.040	0.082	0.098	55.247
NGD	1000	0.228	-0.383	0.332	0.005	-0.023	-0.050	0.506	0.215	0.058	0.031	-0.009	31.647
	2000	0.237	-0.327	0.346	0.011	-0.016	-0.034	0.496	0.193	0.035	0.010	-0.014	22.649
	4000	0.243	-0.243	0.314	0.010	-0.009	-0.021	0.496	0.169	0.013	-0.002	-0.007	10.320
	6000	0.234	-0.247	0.310	0.000	-0.002	-0.003	0.503	0.157	0.010	0.007	-0.002	9.205
	8000	0.189	-0.193	0.295	0.003	0.008	-0.008	0.501	0.173	-0.010	-0.002	0.012	3.601
	10000	0.225	-0.209	0.338	-0.002	0.008	-0.001	0.502	0.147	-0.003	-0.017	-0.017	8.613
Adagrad	1000	0.254	-0.113	0.187	0.016	0.016	0.008	0.465	0.138	-0.074	0.035	0.040	27.683
	2000	0.238	-0.126	0.220	0.012	-0.002	0.004	0.505	0.148	-0.060	0.033	0.031	20.954
	4000	0.222	-0.147	0.210	0.001	-0.010	-0.005	0.501	0.165	-0.026	0.021	0.021	17.050
	6000	0.198	-0.191	0.239	-0.004	-0.005	-0.000	0.506	0.171	-0.007	0.021	0.024	10.513
	8000	0.174	-0.193	0.250	-0.002	0.008	-0.007	0.505	0.180	0.001	0.025	0.020	10.024
	10000	0.160	-0.198	0.248	-0.002	0.004	0.005	0.507	0.189	0.014	0.025	0.019	11.455
Adm	1000	0.302	-0.204	0.198	0.014	0.008	0.002	0.469	0.125	-0.038	0.060	0.034	25.996
	2000	0.265	-0.203	0.239	0.011	-0.001	0.003	0.505	0.129	-0.032	0.046	0.018	17.205
	4000	0.259	-0.198	0.231	0.003	-0.010	-0.006	0.499	0.144	-0.009	0.021	0.007	14.653
	6000	0.243	-0.222	0.266	-0.003	-0.003	0.001	0.505	0.151	0.001	0.017	0.013	9.804
	8000	0.214	-0.211	0.274	-0.002	0.006	-0.007	0.503	0.158	-0.002	0.018	0.007	5.974
	10000	0.211	-0.207	0.278	-0.002	0.001	0.003	0.503	0.165	0.005	0.013	0.005	4.533
True Values		0.200	-0.200	0.300	0.000	0.000	0.000	0.500	0.170	0.000	0.000	0.000	

Table 3: The picked out elements

Thresholds	SG	NGD	Adagrad	Adm
0.01		α_1,α_2,α_3,α_6	α_1,α_2,α_3	α_1,α_2,α_3,α_6,α_8
0.02	α_1,α_2,α_3	α_1,α_2,α_3,α_6,α_7,α_8	α_1,α_2,α_3,α_6,α_8	α_1,α_2,α_3,α_6,α_7,α_8
0.05	a_2,a_3,α_1,α_2,α_3,α_6	α_1,α_2,α_3,α_6,α_7,α_8	α_1,α_2,α_3,α_6,α_7,α_8	α_1,α_2,α_3,α_6,α_7,α_8
0.1	a_1,a_2,a_3,α_1,α_2,α_3,α_6,α_7,α_8	α_1,α_2,α_3,α_6,α_7,α_8	α_1,α_2,α_3,α_6,α_7,α_8	α_1,α_2,α_3,α_6,α_7,α_8

Table 4: The parameter estimates and their estimation errors

Algorithm	t	a_1	a_2	a_3	b_1	b_2	δ (%)
SG	1000	0.06611	-0.00593	0.06147	0.49106	0.13085	55.47355
	2000	0.06468	-0.00724	0.06555	0.49323	0.12856	54.89612
	4000	0.06342	-0.00861	0.06922	0.49471	0.12669	54.37764
	6000	0.06261	-0.00940	0.07121	0.49518	0.12539	54.08828
	8000	0.06216	-0.00999	0.07268	0.49581	0.12483	53.88581
	10000	0.06178	-0.01039	0.07377	0.49609	0.12426	53.73455
NGD	1000	0.39671	-0.35488	0.37405	0.48558	-0.09083	42.79920
	2000	0.28695	-0.25658	0.33919	0.49024	-0.04181	18.44302
	4000	0.25195	-0.21374	0.30653	0.50159	-0.02894	9.60017
	6000	0.21785	-0.20629	0.30949	0.49634	-0.00930	3.72464
	8000	0.17339	-0.20545	0.30430	0.49832	-0.02594	5.66245
	10000	0.20751	-0.20135	0.30291	0.49591	-0.02159	2.49801
Adagrad	1000	0.06434	-0.09751	0.23529	0.49438	0.07043	30.01942
	2000	0.08021	-0.15685	0.28030	0.51088	0.06572	22.26256
	4000	0.12865	-0.18172	0.29214	0.50373	0.03611	12.61632
	6000	0.15473	-0.18947	0.29987	0.50015	0.02164	7.80394
	8000	0.17055	-0.19729	0.29891	0.50073	0.01086	5.44148
	10000	0.17866	-0.19273	0.29885	0.50073	0.00416	3.18050
Adm	1000	0.06435	-0.06137	0.24481	0.46558	0.06805	33.18500
	2000	0.08033	-0.15119	0.27775	0.50818	0.06637	22.59691
	4000	0.14884	-0.18579	0.29286	0.50431	0.01464	8.53083
	6000	0.18094	-0.19539	0.30323	0.49995	0.00334	3.07739
	8000	0.18665	-0.20128	0.30177	0.50164	0.01100	2.54650
	10000	0.19014	-0.19519	0.30073	0.49964	0.00239	1.64781
True Values		0.20000	-0.20000	0.30000	0.50000	0.00170	