# A DELTA LEAST SQUARES LATTICE ALGORITHM FOR FAST SAMPLING

*Parthapratim De and H.(Howard) Fan*

Dept. of Electr. & Comp. Engr. & Comp. Science,
University of Cincinnati, Cincinnati, OH 45221-0030, USA

## ABSTRACT

Most shift operator-based adaptive algorithms exhibit poor numerical behavior when the input discrete time process is obtained from a continuous time process by fast sampling. This includes the shift operator based least squares lattice algorithm. In this paper, we develop a delta least squares lattice algorithm. This algorithm has low computational complexity compared to the delta Levinson RLS algorithm and shows better numerical properties compared to the shift least squares lattice algorithm. Computer simulations show that the new algorithm also outperforms an existing delta least squares lattice algorithm.

## 1. INTRODUCTION

As the demand arises for faster information transmission, fast sampled processes and systems are becoming a necessity. The shift operator, i.e. $qx(t) = x(t+1)$, based methods often yield ill-conditioned processes and systems for fast sampling. This is a very pressing problem, since recently there has been a tremendous upsurge of research and development activities in communications, particularly wireless and mobile communications. Ever increasing capability of today's hardware has paved the way for high speed processing and communication. In such a scenario, the shift operator based algorithms would produce numerically inferior results.

In order to alleviate this problem, the concept of $\delta$-operator based algorithms where $\delta = \frac{q-1}{\Delta}$ has been recently introduced [2] which are numerically superior. Thus $\delta x(n) = \frac{1}{\Delta}[x(n+1) - x(n)]$. Typically if $\{x(n)\}$ is samples of a continuous time signal, $\Delta$ is then chosen as the sampling interval. Obviously as $\Delta \rightarrow 0$ $\delta$ approaches the differentiation operator. Thus new information other than what is apparent in $\{x(n)\}$ is obtained.

Least squares lattice (LSL) on-line algorithm based on the shift operator have been developed [1]. A similar approach will be used to obtain a LSL algorithm based on the $\delta$ operator in this paper. The parameters of the lattice from one stage of the lattice to the next stage will be updated. Also the parameters will be updated in time. Since all the updates of the least squares lattice algorithm will be made in terms of scalar quantities only, the computational complexity of the algorithm is $O(N)$ instead of the $O(N^2)$ which is the computational complexity of the $\delta_b$-Levinson algorithm of [3], where $N$ is the order of the system.

Up to date, the only work on $\delta$-LSL is [4]. However, the $\delta$-LSL of [4] has no limit as the sampling period converges to zero. Therefore, its improvement over the shift LSL is very limited. We present a different formulation to come up with a LSL algorithm in the fast sampling scenario. The new algorithm has a limit as $\Delta \rightarrow 0$ and is seen to be numerically superior to [4].

This work is supported by the Office of Naval Research under Grant N00014-96-1-0241

## 2. BACKGROUND

First let us introduce two difference operators which will be used in this work. The forward $\delta$ operator is defined as $\delta = \frac{q-1}{\Delta}$ and the backward $\delta_b$ operator is defined as $\delta_b = \frac{1-q^{-1}}{\Delta}$ where $q$ is the shift operator. The data vector $\mathbf{x}(k)$ is defined as

$$\mathbf{x}(k) \triangleq [x(k), x(k-1), \cdots, x(k-L+1)]^T \tag{1}$$

where $L$ is the window length. The data matrix corresponding to the $\delta$ operator is defined as

$$Z^{\delta}_{n+1}(k) \triangleq [\delta^n \mathbf{x}(k-n), \delta^{n-1}\mathbf{x}(k-n), \cdots, \mathbf{x}(k-n)] \tag{2}$$

From the definition of the $\delta$ operator, it can be seen that the $Z^{\delta}_{n+1}(k)$ matrix can be decomposed as

$$Z^{\delta}_{n+1}(k) = [\delta^n \mathbf{x}(k-n) \mid Z^{\delta}_n(k-1)] \tag{3}$$

For the $\delta_b$ operator, the data matrix is given by

$$Z_{n+1}(k) \triangleq [\mathbf{x}(k), \delta_b \mathbf{x}(k), \delta_b^2 \mathbf{x}(k), \cdots, \delta_b^n \mathbf{x}(k)] \tag{4}$$

Similarly, this matrix can be decomposed as

$$Z_{n+1}(k) = [Z_n(k) \mid \delta_b^n \mathbf{x}(k)], \tag{5}$$

Also, it can be shown that $Z^{\delta}_{n+1}(k)$ and $Z_{n+1}(k)$ are related by

$$Z^{\delta}_{n+1}(k) = Z_{n+1}(k) M_n \tag{6}$$

where

$$M_n \triangleq \begin{bmatrix} 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & -n\Delta \\ 0 & 0 & 0 & \cdots & \binom{n}{2}\Delta^2 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 1 & -2\Delta & \cdots & (-1)^{n-1} n \Delta^{n-1} \\ 1 & -\Delta & \Delta^2 & \cdots & (-1)^n \Delta^n \end{bmatrix}$$

In [4], only the forward delta operator is used and the $n$th forward prediction error $f_n(k)$ at iteration $k$ is defined as the error involved in predicting $\delta^n \mathbf{x}(k-n)$ using

$Z_n^\delta(k-1)$. The $n$th order backward prediction error $b_n(k)$ at iteration $k$ is defined as the error in predicting $\mathbf{x}(k-n)$ using $Z_n^\delta(k)$ [4]. In the limit as the sampling period $\Delta$ approaches zero, the limiting backward prediction error becomes predicting $\mathbf{x}(t)$ from $\{\mathbf{x}(t), \mathbf{x}^{(1)}(t), \mathbf{x}^{(2)}(t), \cdots, \mathbf{x}^{(n-1)}(t)\}$ where $t$ is the continuous time index. Thus in the limit, $b_n(t)$ tends to zero, irrespective of the underlying system. The dynamics of the system is lost. As a result, the finite precision implementation of the algorithm in [4] does not give much improvement over that of the shift LSL for fast sampling.

In this work, both forward and backward delta operators are used. While the forward prediction error $f_n(k)$ is the same as that of [4], the backward prediction error $b_n(k)$ is the error in predicting $\delta_b^n \mathbf{x}(k)$ from $Z_n(k)$. In the limit as the sampling period approaches zero, both forward and backward prediction errors involve predicting $\mathbf{x}^{(n)}(t)$ from $\{\mathbf{x}(t), \mathbf{x}^{(1)}(t), \mathbf{x}^{(2)}(t), \cdots, \mathbf{x}^{(n-1)}(t)\}$. This is in consonance with [2].

The autocorrelation matrix for the forward prediction (see [1] for the shift operator version) is given by

$$\Phi_{n+1}^{z^\delta}(k) \triangleq Z_{n+1}^{\delta T}(k) Z_{n+1}^\delta(k)$$

Similarly, for the backward prediction, the autocorrelation matrix is defined by

$$\Phi_{n+1}^{z}(k) \triangleq Z_{n+1}^{T}(k) Z_{n+1}(k)$$

Using (3) and defining $\boldsymbol{\alpha}_n(k) = [\alpha_{n,0}(k), \cdots, \alpha_{n,n}(k)]^T$ and $\boldsymbol{\alpha}_n^{'}(k) = [\alpha_{n,0}^{'}(k), \cdots, \alpha_{n,n}^{'}(k)]^T$ where $\alpha_{n,0}(k) = 1$ and $\alpha_{n,n}^{'}(k) = (-1)^n$, the forward augmented normal equation can be written as

$$\begin{aligned} &\Phi_{n+1}^{z^\delta}(k)\boldsymbol{\alpha}_n(k) \\ &= \left[\begin{array}{c|c} \delta^n\mathbf{x}^T(k-n)\delta^n\mathbf{x}(k-n) & \delta^n\mathbf{x}^T(k-n)Z_n^\delta(k-1) \\ \hline Z_n^{\delta T}(k-1)\delta^n\mathbf{x}(k-n) & \Phi_n^{z^\delta}(k-1) \end{array}\right] \\ &\cdot\boldsymbol{\alpha}_n(k) = \begin{bmatrix} R_n^f(k) \\ \mathbf{0} \end{bmatrix} \end{aligned} \tag{7}$$

where $R_n^f(k) = \delta^n\mathbf{x}^T(k-n)Z_{n+1}^\delta(k)\boldsymbol{\alpha}_n(k)$ is the forward prediction error energy. Similarly, it can be shown that the augmented normal equation for the backward linear prediction is

$$\begin{aligned} &\Phi_{n+1}^{z}(k)\boldsymbol{\alpha}_n^{'}(k) \\ &= \left[\begin{array}{c|c} \Phi_n^z(k) & Z_n^T(k)\delta_b^n\mathbf{x}(k) \\ \hline \delta_b^n\mathbf{x}^T(k)Z_n(k) & \delta_b^n\mathbf{x}^T(k)\delta_b^n\mathbf{x}(k) \end{array}\right]\boldsymbol{\alpha}_n^{'}(k) \\ &= \begin{bmatrix} \mathbf{0} \\ (-1)^n R_n^b(k) \end{bmatrix} \end{aligned} \tag{8}$$

where $R_n^b(k)$ is similarly defined as the backward prediction error energy.

## 3. ORDER UPDATE RECURSIONS

For the forward prediction parameters, substituting (6) in the augmented normal equation for the forward part (7), we obtain

$$\Phi_{n+1}^{z}(k)\bar{\boldsymbol{\alpha}}_n(k) = M_n^{-T}\begin{bmatrix} R_n^f(k) \\ \mathbf{0} \end{bmatrix} \tag{9}$$

where $\bar{\boldsymbol{\alpha}}_n(k) = M_n\boldsymbol{\alpha}_n(k)$. Using the decomposition of $\Phi_{n+1}^z(k)$, we obtain

$$\Phi_{n+1}^{z}(k)\begin{bmatrix} \bar{\boldsymbol{\alpha}}_{n-1}(k) \\ 0 \end{bmatrix} = \left[\begin{array}{c} \Phi_n^z(k)\bar{\boldsymbol{\alpha}}_{n-1}(k) \\ \hline c_{n-1}^f(k) \end{array}\right]$$

where $c_{n-1}^f(k) \triangleq \delta_b^n\mathbf{x}^T(k)Z_n(k)\bar{\boldsymbol{\alpha}}_{n-1}(k)$. The above equation can further be written as

$$\begin{aligned} &\Phi_{n+1}^{z}(k)\begin{bmatrix} M_{n-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}\begin{bmatrix} \boldsymbol{\alpha}_{n-1}(k) \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} M_{n-1}^{-T} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}\begin{bmatrix} R_{n-1}^f(k) \\ \mathbf{0} \\ c_{n-1}^f(k) \end{bmatrix} \end{aligned} \tag{10}$$

The above equation can be expressed in terms of $\Phi_{n+1}^{z^\delta}(k)$, by recalling that $\Phi_{n+1}^{z^\delta}(k) = M_n^T\Phi_{n+1}^z(k)M_n$, as

$$\begin{aligned} &\Phi_{n+1}^{z^\delta}(k)M_n^{-1}\begin{bmatrix} M_{n-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}\begin{bmatrix} \boldsymbol{\alpha}_{n-1}(k) \\ 0 \end{bmatrix} \\ &= M_n^T\begin{bmatrix} M_{n-1}^{-T} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}\begin{bmatrix} R_{n-1}^f(k) \\ \mathbf{0} \\ c_{n-1}^f(k) \end{bmatrix} \end{aligned} \tag{11}$$

For the backward prediction parameters, the augmented normal equation (8) can be rewritten as

$$\Phi_{n+1}^{z^\delta}(k)\boldsymbol{\alpha}_n^{''}(k) = M_n^T\begin{bmatrix} \mathbf{0} \\ (-1)^n R_n^b(k) \end{bmatrix} \tag{12}$$

where $\boldsymbol{\alpha}_n^{''}(k) = M_n^{-1}\boldsymbol{\alpha}_n^{'}(k)$. From the above, utilizing the decomposition of $\Phi_{n+1}^{z^\delta}(k)$ yields

$$\Phi_{n+1}^{z^\delta}(k)\begin{bmatrix} 0 \\ \boldsymbol{\alpha}_{n-1}^{''}(k-1) \end{bmatrix} = \left[\begin{array}{c} c_{n-1}^b(k-1) \\ \hline \Phi_n^{z^\delta}(k-1)\boldsymbol{\alpha}_{n-1}^{''}(k-1) \end{array}\right]$$

where $c_{n-1}^b(k-1) \triangleq \delta^n\mathbf{x}^T(k-n)Z_n^\delta(k-1)\boldsymbol{\alpha}_{n-1}^{''}(k-1)$. The above equation is equivalent to

$$\begin{aligned} &\Phi_{n+1}^{z^\delta}(k)\begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & M_{n-1}^{-1} \end{bmatrix}\begin{bmatrix} 0 \\ \boldsymbol{\alpha}_{n-1}^{'}(k-1) \end{bmatrix} \\ &= \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & M_{n-1}^{T} \end{bmatrix}\begin{bmatrix} c_{n-1}^b(k-1) \\ \mathbf{0} \\ (-1)^{n-1}R_{n-1}^b(k-1) \end{bmatrix} \end{aligned} \tag{13}$$

Similarly, in terms of $\Phi_{n+1}^z(k)$, the above equation can be rewritten as

$$\begin{aligned} &\Phi_{n+1}^{z}(k)M_n\begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & M_{n-1}^{-1} \end{bmatrix}\begin{bmatrix} 0 \\ \boldsymbol{\alpha}_{n-1}^{'}(k-1) \end{bmatrix} \\ &= M_n^{-T}\begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & M_{n-1}^{T} \end{bmatrix}\begin{bmatrix} c_{n-1}^b(k-1) \\ \mathbf{0} \\ (-1)^{n-1}R_{n-1}^b(k-1) \end{bmatrix} \end{aligned} \tag{14}$$

Letting $d_1$ and $d_2$ be some constants to be determined, multiplying (11) by $d_1$ and (13) by $d_2$ and then adding them together, we obtain

$$\begin{aligned}
&\Phi^{z^\delta}_{n+1}(k)\{d_1 \begin{bmatrix} \Delta & 0 & 0 & \cdots & 1 \\ 1 & \Delta & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & & 0 \\ 0 & & 1 & \Delta & 0 \\ 0 & 0 & & 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha_{n-1}(k) \\ 0 \end{bmatrix} \\
+ \; & d_2 \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & M^{-1}_{n-1} \end{bmatrix} \begin{bmatrix} 0 \\ \alpha'_{n-1}(k-1) \end{bmatrix}\} \\
= \; & d_1 \begin{bmatrix} 0 & 0 & \cdots & \cdots & 1 \\ 1 & 0 & 0 & \cdots & -\Delta \\ -\Delta & 1 & 0 & \cdots & \Delta^2 \\ \vdots & \vdots & \vdots & & \vdots \\ \frac{\Delta^{n-1}}{(-1)^{n-1}} & \frac{\Delta^{n-2}}{(-1)^{n-2}} & \cdots & 1 & \frac{\Delta^n}{(-1)^n} \end{bmatrix} \begin{bmatrix} R^f_{n-1}(k) \\ \mathbf{0} \\ c^f_{n-1}(k) \end{bmatrix} \\
+ \; & d_2 \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & M^T_{n-1} \end{bmatrix} \begin{bmatrix} c^b_{n-1}(k-1) \\ \mathbf{0} \\ (-1)^{n-1} R^b_{n-1}(k-1) \end{bmatrix} \qquad (15)
\end{aligned}$$

We want the above equation to equal the augmented forward normal equation (7) of order $n$. Equating the right hand sides of these equations and after some calculations, it entails that the following equation holds

$$\begin{aligned}
& d_1 \begin{bmatrix} 0 & 0 & \cdots & \cdots & 1 \\ 1 & 0 & 0 & \cdots & -\Delta \\ -\Delta & 1 & 0 & \cdots & \Delta^2 \\ \vdots & \vdots & \vdots & & \vdots \\ \frac{\Delta^{n-1}}{(-1)^{n-1}} & \frac{\Delta^{n-2}}{(-1)^{n-2}} & \cdots & 1 & \frac{\Delta^n}{(-1)^n} \end{bmatrix} \begin{bmatrix} R^f_{n-1}(k) \\ \mathbf{0} \\ c^f_{n-1}(k) \end{bmatrix} \\
+ \; & d_2 \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & M^T_{n-1} \end{bmatrix} \begin{bmatrix} c^b_{n-1}(k-1) \\ \mathbf{0} \\ (-1)^{n-1} R^b_{n-1}(k-1) \end{bmatrix} \\
= \; & \begin{bmatrix} R^f_n(k) \\ \mathbf{0} \end{bmatrix} \qquad (16)
\end{aligned}$$

From the second row in the above matrix equation, we require

$$d_1[R^f_{n-1}(k) - \Delta c^f_{n-1}(k)] + d_2(-1)^{n-1} R^b_{n-1}(k-1) = 0$$

Similarly, the first row of equation (16) gives

$$R^f_n(k) = d_1 c^f_{n-1}(k) + d_2 c^b_{n-1}(k-1)$$

The update of the parameter $\alpha_n(k)$ can be obtained easily by equating the left hand sides of (7) and (15), from where it can be obtained that $d_1 = \frac{1}{\Delta}$. Then

$$d_2 = -\frac{[R^f_{n-1}(k) - \Delta c^f_{n-1}(k)]}{\Delta[(-1)^{n-1} R^b_{n-1}(k-1)]}$$

So, the forward prediction error energy becomes

$$\begin{aligned}
R^f_n(k) &= \frac{1}{\Delta} c^f_{n-1}(k) \\
&- \frac{[R^f_{n-1}(k) - \Delta c^f_{n-1}(k)]}{\Delta[(-1)^{n-1} R^b_{n-1}(k-1)]} c^b_{n-1}(k-1) \qquad (17)
\end{aligned}$$

Similar calculations can be performed for the backward prediction part to yield

$$\begin{aligned}
R^b_n(k) &= -\frac{1}{\Delta} \frac{[\Delta c^b_{n-1}(k-1) + (-1)^{n-1} R^b_{n-1}(k-1)]}{(-1)^n R^f_{n-1}(k)} \\
&\cdot c^f_{n-1}(k) + \frac{1}{\Delta} \frac{c^b_{n-1}(k-1)}{(-1)^n} \qquad (18)
\end{aligned}$$

Next, updates for the forward and backward prediction errors are obtained in terms of scalar quantities. With $\mathbf{z}_{n+1}(k) \triangleq [x(k), \delta_b x(k), \cdots, \delta_b{}^n x(k)]^T$ and $\mathbf{z}^\delta_{n+1}(k) \triangleq [\delta^n x(k-n), \delta^{n-1} x(k-n), \cdots, x(k-n)]^T$, we define the backward prediction error $b_n(k) \triangleq \mathbf{z}^T_{n+1}(k)\alpha'_n(k)$ and the forward prediction error as $f_n(k) = \mathbf{z}^{\delta^T}_{n+1}(k)\alpha_n(k)$. Also, we define

$$\begin{aligned}
\Gamma^f_n(k) &\triangleq \frac{[R^f_{n-1}(k) - \Delta c^f_{n-1}(k)]}{[(-1)^{n-1} R^b_{n-1}(k-1)]} \\
\Gamma^b_n(k) &\triangleq \frac{[\Delta c^b_{n-1}(k-1) + (-1)^{n-1} R^b_{n-1}(k-1)]}{R^f_{n-1}(k)}
\end{aligned} \qquad (19)$$

as the forward and the backward reflection coefficients. From the order update recursions for the forward and the backward prediction parameters, multiplying by the respective data vectors we obtain

$$\begin{aligned}
f_n(k) &= \frac{f_{n-1}(k)}{\Delta} - \frac{\Gamma^f_n(k)}{\Delta} b_{n-1}(k-1) \\
b_n(k) &= \frac{b_{n-1}(k-1)}{\Delta} - \frac{\Gamma^b_n(k)}{\Delta} f_{n-1}(k) \qquad (20)
\end{aligned}$$

The recursions for $R^f_n(k)$ and $R^b_n(k)$ from (17) and (18) can also be rewritten as

$$\begin{aligned}
R^f_n(k) &= R^f_{n-1}(k) \frac{[1 - \Gamma^b_n(k)\Gamma^f_n(k)]}{\Delta^2} \\
R^b_n(k) &= R^b_{n-1}(k-1) \frac{[1 - \Gamma^b_n(k)\Gamma^f_n(k)]}{\Delta^2}
\end{aligned} \qquad (21)$$

## 4. TIME UPDATES

The scalar quantities $c_{n-1}$'s need to be updated in time. In that direction, we define $c'_{n-1}(k) \triangleq [R^f_{n-1}(k) - \Delta c^f_{n-1}(k)]$. It can also be shown that $c'_{n-1}(k) = (-1)^{n-1}[\Delta c^b_{n-1}(k-1) + (-1)^{n-1} R^b_{n-1}(k-1)]$. Note that, except a $(-1)^{n-1}$ factor, $c'_{n-1}(k)$ is the numerators for (19). From (10) after some manipulations, it can be shown that $c'_{n-1}(k)$ can be written as

$$\begin{aligned}
c'_{n-1}(k) &= (-1)^{n-1}[0, \alpha'^T_{n-1}(k-2)] \begin{bmatrix} 0 & 0 & \cdots & 1 \\ 1 & -\Delta & \cdots & 0 \\ & \ddots & \ddots & 0 \\ 0 & \cdots & 1 & -\Delta \end{bmatrix} \\
&\cdot \Phi^z_{n+1}(k) \begin{bmatrix} M_{n-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \alpha_{n-1}(k) \\ 0 \end{bmatrix} \qquad (22)
\end{aligned}$$

From the definition, the autocorrelation matrix $\Phi^z_{n+1}(k)$ for prewindowing ($L = k$) can be time updated as

$$\Phi^z_{n+1}(k) = \lambda \Phi^z_{n+1}(k-1) + \mathbf{z}_{n+1}(k)\mathbf{z}^T_{n+1}(k) \qquad (23)$$

where $\lambda$ is the forgetting factor. Using (23), (22), (14) and the definitions of the forward and the backward prediction errors, after some algebraic manipulations, it can be shown that the time update of $c'_{n-1}(k)$ is given by

$$\begin{aligned} c'_{n-1}(k) &= \lambda\{\Delta c^b_{n-1}(k-2) + (-1)^{n-1} \\ &\quad \cdot R^b_{n-1}(k-2)\} + (-1)^{n-1} f_{n-1}(k) pr^b_{n-1}(k-1) \\ &= \lambda c'_{n-1}(k-1) + (-1)^{n-1} f_{n-1}(k) pr^b_{n-1}(k-1) \end{aligned} \tag{24}$$

where $pr^b_{n-1}(k) \triangleq \mathbf{z}^T_n(k)\alpha'_{n-1}(k-1)$ is the *a priori* backward prediction error and is related to the backward prediction error $b_{n-1}(k)$, defined earlier, through the conversion factor $\gamma_{n-1}(k)$ by

$$pr^b_{n-1}(k) = \frac{b_{n-1}(k)}{\gamma_{n-1}(k)} \tag{25}$$

The conversion factor $\gamma_{n-1}(k)$ can be updated, as in [1], by

$$\gamma_n(k) = \gamma_{n-1}(k) - \frac{b^2_{n-1}(k)}{R^b_{n-1}(k)} \tag{26}$$

The recursions for the prediction error energy in (21) will be numerically stable as it has been shown in [2] that the quantity $\frac{[1-(\Gamma^b_n(k)\Gamma^f_n(k))]}{\Delta^2}$ has a limit as the sampling period $\Delta \to 0$. Similarly, from (20), it can be shown that

$$\begin{aligned} b_{n+1}(k) &= \frac{1}{\Delta}[b_n(k-1) + \frac{\Gamma^b_{n+1}(k)}{\Gamma^b_n(k)} b_n(k)] \\ &- b_{n-1}(k-1)\frac{\Gamma^b_{n+1}(k)}{\Gamma^b_n(k)} \frac{[1-\Gamma^b_n(k)\Gamma^f_n(k)]}{\Delta^2} \end{aligned} \tag{27}$$

By the definitions of the quantities $\Gamma$'s, it follows that $\lim_{\Delta\to 0} \Gamma^{f,b}_n(k) = (-1)^{n-1}$. Thus the first term on the right hand side of (27) approaches the derivative of $b_n(t)$, and (27) has a limiting recursion as $\Delta \to 0$. Therefore, (19), (21), (24)-(27) form the new $\delta$-LSL that has a limit as $\Delta \to 0$.

## 5. SIMULATION RESULTS

The example of [3] is used here. A 4th order AR process is used as the input. The poles of this model are chosen so that they correspond to a continuous-time model with double poles at $-3 \pm j3$ with a sampling period $\Delta = 0.0026$. They are driven by a zero-mean, unit variance white noise process. Prewindowing is used on the data, and the forgetting factor is set at $\lambda = 0.997$. The double precision provided by Sun ULTRA 1 is considered as infinite precision. Computer experiments are written so that finite precision in terms of binary bits can be simulated. Floating point arithmetic is considered, and quantization is performed after each arithmetic operation. The relative error of the prediction error energy is defined as

$$Re(k) = \frac{|(R^f_4(k))_{fp} - (R^f_4(k))_\infty|}{(R^f_4(k))_\infty}$$

where $fp$ refers to finite precision and $\infty$ means infinite precision. $\bar{Re}(k)$ is the ensemble average of $Re(k)$ over 30 computer experiments. Figure 1 is a plot of $\bar{Re}(k)$ for the $q$-LSL algorithm [1], the $\delta$-LSL algorithm of [4] and the new algorithm presented in this paper versus iteration number $k$ when the number of mantissa bits is 50. Figure 2 is a plot of time averaged $\bar{Re}(k)$ from $k = 250$ to $k = 2000$ versus the number of mantissa bits for the three algorithms. In both cases, the new algorithm shows superior numerical behavior under finite precision. In fact, the algorithm in [4] shows no improvement over the $q$-LSL.

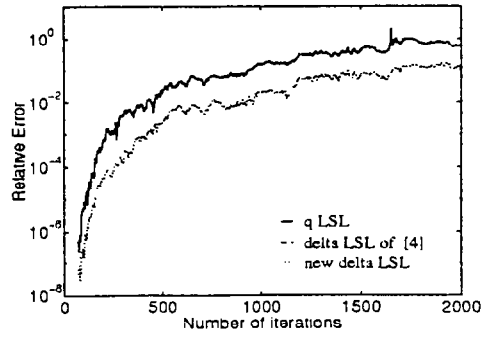


Figure 1: Ensemble averaged (over 30 runs) relative error, $\bar{Re}(k)$, versus iteration number. 50 *bits*, $\lambda = 0.997$

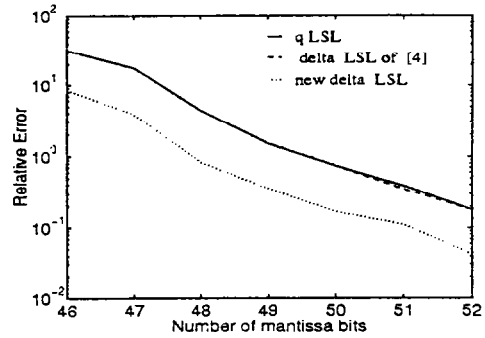


Figure 2: Time averaged relative error $\bar{Re}(k)$ versus number of mantissa bits. $\lambda = 0.997$

## 6. REFERENCES


[1] S. Haykin, *Adaptive Filter Theory*, 2nd ed., Englewood Cliffs, NJ: Prentice-Hall, 1991.

[2] R. Vijayan, H. V. Poor, J. B. Moore, and G. C. Goodwin, "A Levinson-type algorithm for modeling fast-sampled data," *IEEE Trans. on Automatic Control*, vol. AC-36, no. 3, pp. 314-321, March 1991.

[3] H. Fan and X. Liu, "Delta Levinson and Schur-type RLS algorithms for adaptive signal processing," *IEEE Trans. on Signal Processing*, vol. 42, no. 7, pp. 1629-1639, June 1994.

[4] F. Jabbari, "Lattice filters for RLS estimation of a Delta Operator-Based Model, "*IEEE Trans. on Automatic Control*, vol. AC-36, no. 7, pp. 869-875, July 1991.