PU.M.A. Ser. B, Vol. 2 (1991), No. 1, pp. 53–69

A general technique for finite memories data manipulation and smooth predictions*

FERENC SZIDAROVSZKY,† CHAO–YEN WU† AND A. TERRY BAHILL†

Abstract. Filters are used to help separate signals from noise. They are also used as predictors and for signal processing; for example, they can be used to calculate present or future values of position, velocity and acceleration. In this paper a general technique for finite memories data manipulation and smooth predictions is described. In their conventional implementation a matrix inversion must be performed for each new calculation. This paper presents a general technique which is based on the difference equation using a least–squares criterion for predictions. The difference equation approach has a major advantage over the matrix approach, because it is not necessary to compute the matrix inversion.

AMS Subject Classifications. 68P05.

1. Introduction

Filters are used to separate signals from noise. They can also be used as predictors and for signal processing; for example, they can be used to calculate present or future values of position, velocity and acceleration. Correct choice of the filter coefficients will ensure the least mean squared error.

Many adaptive filters are available, for example, two–point–linear predictors [1], fivepoint–quadratic predictors [1], Kalman filters [2], least–mean–square (LMS) adaptive filters [3–4], recursive–least–square (RLS) adaptive filters [5–6], lattice–filter adaptive filters [7–8], and other forms of adaptive filters [9–10]. A tutorial review of lattice structures and their use in adaptive prediction of time series data is provided in [11]. Friedlander concludes that

*Received: November 2, 1991.

†Systems and Industrial Engineering, University of Arizona, Tucson, AZ 85721, U.S.A., Present address of the second author: Division of Industrial Engineering, School of Engineering, Alfred University, P.O. Box 576, Alfred, NY 14802.

the adaptive least squares lattice (LSL) has a number of practical advantages compared to gradient adaptive techniques. In recent years, adaptive prediction techniques have become more popular than other techniques as suggested in [12] and [10]. All these adaptive filters try to minimize the mean squared error between the actual and predicted signal; so they are called least squares adaptive filters. They are designed to minimize the sum of the squared errors:

$$\sum_{i=1}^{N}[y(t_i) - \hat{y}(t_i)]^2, \tag{1}$$

where N is the number of points that the adaptive filters use for each adaptive procedure, $y(t_i)$ is the position and $\hat{y}(t_i)$ is the predicted position at time t_i. Least squares adaptive filters use the second-order statistics of an observed process $y(t_i)$. In most practical applications these statistics are not known a priori and must be estimated from data. If the process is known to be stationary and if a large amount of data are available, it is possible to estimate the autocorrelation sequence and compute the least squares adaptive filters. In practice, signals are often nonstationary, having time-varying statistics. To account for these changing statistics, the process of estimating second-order statistics and computing adaptive filter coefficients needs to be carried out in an iterative fashion. Such a procedure for predicting a time series without prior knowledge of its statistics is called an adaptive processing.

In this paper we derive a general technique for finite memories (block processing) data manipulation and smooth prediction. In the conventional (matrix) approach, a matrix inversion must be preformed for each new set of data (finite memories or block). However, with our new method, which is based on the difference equation using a least-squares criterion for predictions, it is not necessary to compute the matrix version. Therefore, the difference equation approach is better than the matrix approach.

In section 2, our general approach is derived and verified mathematically. In section 3 we discuss some numerical examples such as the two-point linear predictor (TPLP), the fivepoint quadratic predictor (FPQP), the nine-point cubic predictor (NPCP), the first-order acceleration predictor, the second-order acceleration predictor, and the third-order acceleration predictor. A recursive identity method is included as an appendix.

2. The general approach

In traditional block processing (finite memories) techniques, a matrix approach is used to derive formulas for the adaptive filters. In this paper a different approach, which is based on the difference equation, is used to derive formulas for adaptive filters. The difference equation approach has a major advantage over the matrix approach because it does not require matrix inversion.

Consider a qth order polynomial equation as an input signal $q \leq N-1$ of the form

$$y(t_i) = d_0 + d_1 t_i + \ldots + d_q t_i^q \,. \tag{2}$$

Assume that the output signal of the predictor is given at the future time $t_{i+\alpha}$ as

$$\sum_{j=0}^{m} A_j y^{(j)}(t_{i+\alpha}) \,, \tag{3}$$

where $y^{(j)}(t_{i+\alpha})$ is the jth derivative of $y(t_{i+\alpha})$ and A_j represents the coefficients of the output signal. We want to predict this output signal α–steps ahead of time by using past observations in the form

$$\sum_{j=0}^{m} A_j y^{(j)}(t_{i+\alpha}) = \sum_{k=0}^{N-1} a_k y(t_{i-k}) \,, \tag{4}$$

where $y(t_{i-k})$ is the past signal at time t_{i-k}, a_k represents the coefficients of the difference equation, and N is the order of the difference equation. In equation (4) we must have equality, if the predictor is to produce no system error.

From the theory of digital process [13]

$$y(t_{i+\alpha}) = e^{\alpha TD} y(t_i) \,, \tag{5}$$

where D is the differential operator and T is the sampling period. Therefore $y^{(j)}(t_{i+\alpha})$ and $y(t_{i-k})$ can be rewritten as

$$y^{(j)}(t_{i+\alpha}) = e^{\alpha TD} D^j y(t_i) = \sum_{\ell=0}^{\infty} \frac{(\alpha T)^\ell}{\ell!} D^{\ell+j} y(t_i) \tag{6}$$

and

$$y(t_{i-k}) = e^{-kTD} y(t_i) = \sum_{\ell=0}^{\infty} \frac{(-kT)^\ell}{\ell!} D^\ell y(t_i) \,. \tag{7}$$

Substituting equations (6) and (7) into (4) gives the relation

$$(8) \qquad \sum_{j=0}^{m} A_j \sum_{\ell=0}^{\infty} \frac{(\alpha T)^\ell}{\ell!} D^{\ell+j} y(t_i) = \sum_{k=0}^{N-1} a_k \sum_{\ell=0}^{\infty} \frac{(-kT)^\ell}{\ell!} D^\ell y(t_i) .$$

For simplicity, define $A_j = 0$ for $j > m$, and observe that the derivative of $y(t_i)$ with order greater than q are zero as the consequence of equation (2). Hence equation (8) can be rewritten as

$$(9) \qquad \sum_{J=0}^{q} \left[A_0 \frac{(\alpha T)^J}{J!} + A_1 \frac{(\alpha T)^{J-1}}{(J-1)!} + \ldots + A_J \frac{(\alpha T)^0}{0!} \right] D^J y(t_i)$$

$$= \sum_{\ell=0}^{q} \left[\sum_{k=0}^{N-1} a_k \frac{(-kT)^\ell}{\ell!} D^\ell y(t_i) \right] ,$$

where in the left hand side of equation (9) we have introduced the new variable a $J = \ell + j$. By comparing the coefficients of the same derivatives for both sides of equation (9) we get the conditions

$$(10) \qquad \sum_{k=0}^{N-1} a_k \frac{(-kT)^J}{J!} = A_0 \frac{(\alpha T)^J}{J!} + A_1 \frac{(\alpha T)^{J-1}}{(J-1)!} + \ldots + A_s \frac{(\alpha T)^{J-s}}{(J-s)!} ,$$

where $s = \min\{q; m\}$. Simple algebra leads to equations

$$(11) \qquad \sum_{k=0}^{N-1} a_k k^J = \alpha_j , \quad J = 0, 1, 2, \ldots, q$$

with

$$(12) \qquad \alpha_j = (-1)^J \frac{J!}{T^J} \left[A_0 \frac{(\alpha T)^J}{J!} + A_1 \frac{(\alpha T)^{J-1}}{(J-1)!} + \ldots + A_s \frac{(\alpha T)^{J-s}}{(J-s)!} \right] .$$

For minimizing the mean-square-error coefficient and also reaching zero system error, we have to minimize the Lagrangian [14], which can be written as

$$(13) \qquad L(\underline{a}, \underline{\lambda}) = \sum_{k=0}^{N-1} a_k^2 - 2\lambda_0 \left[\sum_{k=0}^{N-1} a_k - \alpha_0 \right] - \ldots - 2\lambda_q \left[\sum_{k=0}^{N-1} a_k k^q - \alpha_q \right] ,$$

where

$$\underline{a} = [a_0, a_1, \ldots, a_{N-1}]^T \quad \text{and} \quad \underline{\lambda} = [\lambda_0, \lambda_1, \ldots, \lambda_q]^T .$$

Simple differentiation of equation (13) with respect to a_k shows that for $k = 0, 1, \ldots, N-1$,

$$\frac{\partial L(a,\lambda)}{\partial a_k} = 2a_k - 2\lambda_0 - 2\lambda_1 k - \ldots - 2\lambda_q k^q \,, \tag{14}$$

and for $j = 0, 1, \ldots, q$,

$$\frac{\partial L(a,\lambda)}{\partial \lambda_j} = -2\left(\sum_{k=0}^{N-1} a_k k^j - \alpha_j\right) . \tag{15}$$

Setting equations (14) and (15) to zero respectively yields the following system of linear equations:

$$-a_k + \lambda_0 + \lambda_1 k + \lambda_2 k^2 + \ldots + \lambda_q k^q = 0 \quad (0 \le k \le N-1) \tag{16}$$

and

$$\sum_{k=0}^{N-1} a_k k^j = \alpha_j \quad (0 \le j \le q) \,. \tag{17}$$

In order to simplify these equations introduce the following notation

$$\underline{V} = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \\ 1 & 1 & 1 & \ldots & 1 \\ 1 & 2 & 2^2 & \ldots & 2^q \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & (N-1) & (N-1)^2 & \ldots & (N-1)^q \end{pmatrix} , \tag{18}$$

then equations (16) and (17) can be rewritten in block form as

$$\begin{pmatrix} -\underline{I} & \underline{V} \\ \underline{V}^T & \underline{0} \end{pmatrix} \begin{pmatrix} \underline{a} \\ \underline{\lambda} \end{pmatrix} = \begin{pmatrix} \underline{0} \\ \underline{\alpha} \end{pmatrix} , \tag{19}$$

where $\underline{I}$ is the identity matrix, and

$$\underline{\alpha} = [\alpha_0, \alpha_1, \ldots, \alpha_q]^T \,. \tag{20}$$

From the first block–row of equation (19) we conclude that

$$-\underline{a} + \underline{V}\,\underline{\lambda} = \underline{0} \,, \tag{21}$$

that is,

$$\underline{a} = \underline{V}\,\underline{\lambda} \,. \tag{22}$$

The second block-row of equation (19) implies that

$$\underline{V}^T \underline{a} = \underline{\alpha}, \tag{23}$$

and by substituting equation (22) into equation (23) we have that

$$(\underline{V}^T \underline{V})\underline{\lambda} = \underline{\alpha}. \tag{24}$$

Hence the following algorithm can be proposed for the solution of the prediction problem:

Step 1. Consider matrix $\underline{U} = \underline{V}^T \underline{V}$ as

$$u_{ij} = \sum_{k=0}^{N-1} k^{i+j-2} \quad (ij = 1, 2, \ldots, q+1). \tag{25}$$

Step 2. Construct vector $\underline{\alpha} = [\alpha_0, \alpha_1, \ldots, \alpha_q]^T$ using equations (12).

Step 3. Solve the $(q+1) \times (q+1)$ size linear equation (24).

Step 4. Obtain the coefficient vector $\underline{a}$ by using equation (22).

The above method has several special properties. Before illustrating it by particular cases. these special features will be discussed.

1. Matrices $\underline{V}$ and $\underline{U}$ depend on only the selected integers q and N, and they are independent of the selected output-model and measurements. Consequently they have to be computed only once and tabulated with their inverses and (if needed) with their Cholesky's factorial form.

2. Matrix $\underline{U}$ is positive definite, since the rank of $\underline{V}$ is $q+1$, which can be seen from the fact that the $(q+1) \times (q+1)$ matrix obtained from the first $(q+1)$ row of $\underline{V}$ is a Vandermonde-matrix. This fact has two important consequences. First, it guarantees that linear equation (24) has a unique solution. Second, equation (24) can be solved by the numerically stable Cholesky's factorization method.

3. For fixed values of q and N, the output-model and measurements $y(t_i - k)$ are shown up only in the components of vector $\underline{\alpha}$. Once fixed q and N are given, matrix $(\underline{V}\,\underline{U}^{-1})$ is tabulated, then the coefficient vector $\underline{a}$ can be obtained by multiplying this matrix by vector $\underline{\alpha}$. The number of operations in this multiplication equals $N \times (q+1)$, since the type of matrix $\underline{V}\,\underline{U}^{-1}$ is $N \times (q+1)$, and vector α is $(q+1)$ dimensional. Similarly one may verify that the number of additions and subtractions equals $N \times q$.

4. In computing and/or tabulating matrix $\underline{V}\,\underline{U}^{-1}$) for fixed values of q and N useful recursive relations can be used, which make the computational procedure less expensive. In this paragraph these recursions will be introduced. We shall consider two cases:

Case 1: q is replaced by $q+1$ (for $q < N-1$) ;

Case 2: N will be increased by one.

By using these procedures the results for different values of (q, N) can be obtained according to the following scheme.

Assume first that the value of q is increased by 1. Then

$$\underline{U}_{q+1} = \begin{pmatrix} \underline{U}_q & \underline{q} \\ \underline{q}^T & q_0 \end{pmatrix} . \tag{26}$$

where $\underline{U}_q$ and $\underline{U}_{q+1}$ are matrix $\underline{U}$ with q and $q+1$, respectively. Furthermore, q_0 and $\underline{q}$ can be described as

$$q_0 = \sum_{k=0}^{N-1} k^{2q+2} \tag{27}$$

and

$$\underline{q} = \begin{pmatrix} \sum_{k=0}^{N-1} k^{q+1} \\ \cdots \\ \sum_{k=0}^{N-1} k^{2q+1} \end{pmatrix} . \tag{28}$$

And the inverse of matrix (26) can also be expressed in block form as

$$\underline{U}_{q+1}^{-1} = \begin{pmatrix} \underline{X} & \underline{y} \\ \underline{y}^T & s \end{pmatrix} . \tag{29}$$

We may assume that the inverse matrix is symmetric, since the same assumption holds for matrix $\underline{U}_{q+1}$. The definition of inverse implies that

$$\begin{pmatrix} \underline{U}_q & \underline{q} \\ \underline{q}^T & q_0 \end{pmatrix} \begin{pmatrix} \underline{X} & \underline{y} \\ \underline{y}^T & s \end{pmatrix} = \begin{pmatrix} \underline{I} & \underline{0} \\ \underline{0}^T & 1 \end{pmatrix} , \tag{30}$$

that is,

$$\begin{cases} \underline{U}_q\,\underline{X} + \underline{q}\,\underline{y}^T = \underline{I} \\ \underline{U}_q\,\underline{y} + \underline{q}s = \underline{0} \\ \underline{q}^T\,\underline{y} + q_0 s = 1 \end{cases} . \tag{31}$$

From the second equation of (31) we have

$$\underline{y} = \underline{U}_q^{-1}\underline{q}\,s . \tag{32}$$

By substituting equation (32) into the third equation of (31) we obtain

$$s = \frac{1}{q_0 - \underline{q}^T \underline{U}_q^{-1} \underline{q}}, \tag{33}$$

and by introducing vector $\underline{U}_q^{-1}\underline{q} = \underline{r}$, equation (33) can be rewritten as

$$s = \frac{1}{q_0 - \underline{q}^T \underline{r}}. \tag{34}$$

Note that $\underline{r}$ is easy to obtain, since the inverse U_q^{-1} is assumed to be known from previous computations. After the value of s is computed, from equation (32) we have

$$\underline{y} = -\underline{r}\, s, \tag{35}$$

and from the first equation of (31) we conclude that

$$\underline{X} = \underline{U}_q^{-1}(\underline{I} - \underline{q}\,\underline{y}^T) = \underline{U}_q^{-1} - \underline{r}\,\underline{y}^T. \tag{36}$$

Hence the inverse of $\underline{U}_q$ can be obtained according to the following steps:

Step 1. Compute $\underline{r} = \underline{U}_q^{-1}\underline{q}$.

Step 2. Obtain $s, \underline{y}$ and $\underline{X}$ by using equation (34), (35) and (36), respectively.

Note that the computation of $\underline{q}$ needs only integer operations, so it is inexpensive. Assuming that $\underline{q}$ is known, the entire procedure requires $O(q^2)$ floating point operations.

Now consider the second case, where N is increased by 1. Let $\underline{U}_{(N)}$ and $\underline{U}_{(N+1)}$ denote matrix $\underline{U}$ with N and $N+1$ respectively. Then

$$\underline{U}_{N+1} = \underline{U}_{(N)} + \begin{pmatrix} 1 & N & N^2 & \dots \\ N & N^2 & \dots & \dots \\ N^2 & \dots & \dots & \dots \\ \dots & \dots & \dots & N^{2q} \end{pmatrix} = \underline{U}_{(N)} + \underline{n}\,\underline{n}^T, \tag{37}$$

with

$$\underline{n} = \left[1, N, N^2, \dots, N^q\right]^T. \tag{38}$$

First we shall prove that

$$\left(\underline{U}_{(N)} + \underline{n}\,\underline{n}^T\right)^{-1} = \underline{U}_{(N)}^{-1} - \frac{\underline{U}_{(N)}^{-1}\underline{n}\,\underline{n}^T\underline{U}_{(N)}^{-1}}{1 + \underline{n}^T\underline{U}_{(N)}^{-1}\underline{n}}. \tag{39}$$

The assertion is verified by the multiplication

$$(40)\quad \left(\underline{U}_{(N)} + \underline{n}\,\underline{n}^T\right)\left(\underline{U}^{-1}_{(N)} - \frac{\underline{U}^{-1}_{(N)}\underline{n}\,\underline{n}^T\underline{U}^{-1}_{(N)}}{1+\underline{n}^T\underline{U}^{-1}_{(N)}\underline{n}}\right)$$

$$= \underline{I} + \underline{n}\,\underline{n}^T\underline{U}^{-1}_{(N)} - \frac{1}{1+\underline{n}^T\underline{U}^{-1}_{(N)}\underline{n}}\left(\underline{n}\,\underline{n}^T\underline{U}^{-1}_{(N)} + \underline{n}\left(\underline{n}^T\underline{U}^{-1}_{(N)}\underline{n}\right)\underline{n}^T\underline{U}^{-1}_{(N)}\right)$$

$$= \underline{I} + \underline{n}\,\underline{n}^T\underline{U}^{-1}_{(N)} - \frac{1}{1+\underline{n}^T\underline{U}^{-1}_{(N)}\underline{n}}\underline{n}\,\underline{n}^T\underline{U}^{-1}_{(N)}\left(1+\underline{n}^T\underline{U}^{-1}_{(N)}\underline{n}\right)$$

$$= \underline{I} + \underline{n}\,\underline{n}^T\underline{U}^{-1}_{(N)} - \underline{n}\,\underline{n}^T\underline{U}^{-1}_{(N)}$$

$$= \underline{I}.$$

Introduce vector $\underline{U}^{-1}_{(N)}\underline{n} = \underline{m}$, which can be easily computed, since $\underline{U}^{-1}_{(N)}$ is known from previous computations. Then equations (37) and (39) can be combined to yield

$$(41)\quad \underline{U}^{-1}_{(N+1)} = \underline{U}^{-1}_{(N)} - \frac{1}{1+\underline{n}^T\underline{m}}\underline{m}\,\underline{m}.$$

In applying this formula, first vector $\underline{m}$ is to be determined, which requires $(q+1)^2$ floating–point operations. The computation of the coefficient of the second term of equation (41) needs $(q+1)+2 = q+3$ operations, and the final use of equation (41) needs $2\frac{(q+1)^2+(q+1)}{2} + (q+1)$ operations, when we used the symmetry of the matrix involved. (we must compute the elements in and above the diagonal only). The total number of floating–point operations required to solve equation (41) is $O(q^2)$, namely

$$(42)\quad 2(q+1)^2 + 3(q+1) + 2\,,$$

which is again much cheaper than the direct solution of linear equations (24), which needs at least $O(q^3)$ operations.

3. Numerical examples

The above procedures will be now illustrated by numerical examples.

Example 1. Select $q = 1$, and $m = 1$ with $A_0 =)$, $A_1 = 1$. In this case the output is $\dot{y}\,(t_{i+\alpha})$, so a first order velocity predictor is to be found. By using equation (12) we have

$$(43)\quad \alpha_0 = (-1)^0\frac{0!}{T^0}\left(A_0\frac{(\alpha T)^0}{0!}\right) = 0$$

and

$$\alpha_1 = (-1)^1 \frac{1!}{T^1}\left(A_0\frac{(\alpha T^1}{1!} + A_1\frac{(\alpha T)^0}{0!}\right) = -\frac{1}{T}. \tag{44}$$

Furthermore, $\underline{U}$ can be computed from equation (25) to yield

$$\underline{U} = \begin{pmatrix} N & \frac{N(N-1)}{2} \\ \frac{N(N-1)}{2} & \frac{N(N-1)(2N-1)}{6} \end{pmatrix}, \tag{45}$$

since it is well known that

$$\sum_{k=0}^{N-1} k = \frac{N(N-1)}{2} \tag{46}$$

and

$$\sum_{k=0}^{N-1} k^2 = \frac{N(N-1)(2N-1)}{6}. \tag{47}$$

Hence equation (24) has the form

$$\underline{U}\begin{pmatrix}\lambda_0 \\ \lambda_1\end{pmatrix} = \begin{pmatrix}\alpha_0 \\ \alpha_1\end{pmatrix}, \tag{48}$$

that is,

$$\begin{cases} N\lambda_0 + \frac{N(N-1)}{2}\lambda_1 = 0 \\ \frac{N(N-1)}{2}\lambda_0 + \frac{N(N-1)(2N-1)}{6}\lambda_1 = -\frac{1}{T} \end{cases}. \tag{49}$$

Simple calculation shows that the solution is

$$\begin{cases} \lambda_0 = \frac{6}{T(N+1)N} \\ \lambda_1 = \frac{-12}{T(N^2-1)N} \end{cases}. \tag{50}$$

Hence from equation (22), we have

$$\underline{a} = \underline{V}\underline{\lambda} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ \cdots & \cdots \\ 1 & N-1 \end{pmatrix}\begin{pmatrix}\lambda_0 \\ \lambda_1\end{pmatrix} = \begin{pmatrix} \lambda_0 \\ \lambda_0+\lambda_1 \\ \lambda_0+2\lambda_1 \\ \cdots \\ \lambda_0+(N-1)\lambda_1 \end{pmatrix}. \tag{51}$$

That is, for $k = 0, 1, 2, \ldots, N-1$, equation (51) implies that

$$\begin{aligned} a_k &= \lambda_0 + k\lambda_1 = \frac{6}{T(N+1)N} + k\frac{-12}{T(N+1)N(N-1)} \\ &= \frac{6N-6-12k}{T(N+1)N(N-1)} = \frac{6(N-2k-1)}{T(N^2-1)N}. \end{aligned} \tag{52}$$

In summary, the first order velocity prediction has the form:

$$\dot{y}\,(t_{i+\alpha}) = \sum_{k=0}^{N-1} \frac{6(N-2k-1)}{T(N^2-1)N} y(t_{i-k})\,. \tag{53}$$

As a special case, set $N = 2$. Then a_0 and a_1 can be computed from equation (52) to yield

$$a_0 = \frac{6(2-0-1)}{T(3)2} = \frac{1}{T} \tag{54}$$

and

$$a_1 = \frac{6(2-2-1)}{T(3)2} = \frac{1}{T}\,. \tag{55}$$

Therefore the prediction at time $(t_{i+\alpha})$ of the first order velocity predictor is

$$\dot{y}\,(t_{i+\alpha}) = \frac{1}{T}\left[y(t_i) - y(t_{i-1})\right]\,, \tag{56}$$

which coincides with the well known backward difference estimation

Example 2. By selecting $q = 1$ and $N = 3$ from equation (45) we know that

$$\underline{U} = \begin{pmatrix} 3 & 3 \\ 3 & 5 \end{pmatrix} \tag{57}$$

and easy calculation confirms that

$$\underline{U}^{-1} = \begin{pmatrix} \frac{5}{6} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix}\,. \tag{58}$$

First we shall increase the value of q by 1 (case 1), letting $q + 1 = 2$. Using equations (58), (27) and (28), we have

$$\underline{U}_1^{-1} = \begin{pmatrix} \frac{5}{6} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix}\,,$$

$$q_0 = \sum_{k=0}^{2} k^4 = 17\,, \tag{59}$$

and

$$\underline{q} = \left[\sum_{k=0}^{2} k^2\,,\ \sum_{k=0}^{2} k^3\right]^T = [5,9]^T\,. \tag{60}$$

Then

$$\underline{r} = \underline{U}_1^{-1}\underline{q} = \begin{pmatrix} \frac{5}{6}5 - \frac{1}{2}9 \\ -\frac{1}{2}5 + \frac{1}{2}9 \end{pmatrix} = \begin{pmatrix} -\frac{1}{3} \\ 2 \end{pmatrix}\,, \tag{61}$$

and from equations (34), (59), (60) and (61)

$$s = \frac{1}{17 - [5,9]\begin{pmatrix} -\frac{1}{3} \\ 2 \end{pmatrix}} = \frac{1}{17 - \frac{49}{3}} = \frac{3}{2}. \tag{62}$$

Then from equations (35) and (36) we have,

$$\underline{y} = -\begin{pmatrix} -\frac{1}{3} \\ 2 \end{pmatrix}\frac{3}{2} = \begin{pmatrix} \frac{1}{2} \\ -3 \end{pmatrix} \tag{63}$$

and

$$\begin{aligned} \underline{X} &= \begin{pmatrix} \frac{5}{6} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix} - \begin{pmatrix} -\frac{1}{3} \\ 2 \end{pmatrix}\left(\tfrac{1}{2}, \quad -3\right) \\ &= \begin{pmatrix} \frac{5}{6} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix} - \begin{pmatrix} -\frac{1}{6} & 1 \\ 1 & -6 \end{pmatrix} = \begin{pmatrix} 1 & -\frac{3}{2} \\ -\frac{3}{2} & \frac{13}{2} \end{pmatrix}. \end{aligned} \tag{64}$$

Hence, from equations (29), (62), (63) and (64) we have

$$\underline{U}_2^{-1} = \begin{pmatrix} 1 & -\frac{3}{2} & \frac{1}{2} \\ -\frac{3}{2} & \frac{13}{2} & -3 \\ \frac{1}{2} & -3 & \frac{3}{2} \end{pmatrix}. \tag{65}$$

This result can be checked out by verifying that the product of the computed matrix $\underline{U}_2^{-1}$ and

$$\underline{U}_2 = \begin{pmatrix} 3 & 3 & 5 \\ 3 & 5 & 9 \\ 5 & 9 & 17 \end{pmatrix} \tag{66}$$

equals the identity matrix, as it actually does. Equation (66) is derived from equation (26).

Example 3. Select again $q = 1$ and $N = 3$. In this example N will be increased by 1 (case 2), and the recursive algorithm for increasing N will be illustrated. Using the notation of the algorithm and equations (66) and (38) we know that

$$\underline{U}_{(3)} = \begin{pmatrix} 3 & 3 \\ 3 & 5 \end{pmatrix}, \tag{67}$$

and

$$\underline{n} = [1,3]^T, \tag{68}$$

and since

$$\underline{U}_{(3)}^{-1} = \begin{pmatrix} \frac{5}{6} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix} \tag{69}$$

we conclude that

$$\underline{m} = \underline{U}_{(3)}^{-1}\underline{n} = \begin{pmatrix} \frac{5}{6} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} 1 \\ 3 \end{pmatrix} = \begin{pmatrix} -\frac{2}{3} \\ 1 \end{pmatrix}. \tag{70}$$

In order to apply equation (41) for updating the inverse of $\underline{U}_{(N+1)}$, we first compute

$$\underline{n}^T\underline{m} = [1,3]\begin{pmatrix} -\frac{2}{3} \\ 1 \end{pmatrix} = \frac{7}{3} \tag{71}$$

and

$$\frac{1}{1+\underline{n}^T\underline{m}} = \frac{1}{1+\frac{7}{3}} = \frac{3}{10}. \tag{72}$$

Therefore, from equation (41) we have

$$\begin{aligned} \underline{U}_{(4)}^{-1} &= \begin{pmatrix} \frac{5}{6} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix} - \frac{3}{10}\begin{pmatrix} -\frac{2}{3} \\ 1 \end{pmatrix}\left(-\tfrac{2}{3},\ 1\right) \\ &= \begin{pmatrix} \frac{5}{6} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix} - \frac{3}{10}\begin{pmatrix} \frac{4}{9} & -\frac{2}{3} \\ -\frac{2}{3} & 1 \end{pmatrix} = \begin{pmatrix} \frac{7}{10} & -\frac{3}{10} \\ -\frac{3}{10} & \frac{1}{5} \end{pmatrix}. \end{aligned} \tag{73}$$

This result can be easily checked by observing that (from equation (25))

$$\underline{U}_{(4)} = \begin{pmatrix} \sum_{k=0}^{3} k^0 & \sum_{k=0}^{3} k^1 \\ \sum_{k=0}^{3} k^1 & \sum_{k=0}^{3} k^2 \end{pmatrix} = \begin{pmatrix} 3 & 6 \\ 6 & 14 \end{pmatrix} \tag{74}$$

and verifying that the product of $\underline{U}_{(4)}$ and the computed inverse gives the identity matrix, as it actually does.

4. Computer simulation

All predictors were tested with out battery of input signals that included: sinusoids, triangular waveforms, parabolic waveforms, cubic waveforms, and saccadic eye movements [15]. The frequencies of the periodic waveforms were 10 Hz and their amplitudes were 5 degrees. The saccades were 10 degrees in amplitude. The step size was one millisecond. Computer programs were written in the C programming language and were run on a VAX 11/750 with the UNIX operating system. Mean squared errors for 340 msec of steady state tracing for the two–point linear predictor (TPLP), the five–point quadratic predictor (FPQP), and the nine–point cubic predictor (NPCP) are summarized in Tables 1,2 and 3.

Table 1. Mean squared error of the TPLP (deg^2)					
	Sinusoids	Triangular	Parabolic	Cubic	Saccades
1–step	0.00019	0.00317	0.00025	0.00032	0.00036
5–step	0.04315	0.17426	0.05420	0.06770	0.15814
10–step	0.57151	1.21980	0.68814	0.83497	0.64623

Table 2. Mean squared error of the FPQP (deg^2)					
	Sinusoids	Triangular	Parabolic	Cubic	Saccades
1–step	0.00001	0.00558	0.00005	0.00009	0.00020
5–step	0.00217	0.45051	0.01068	0.01919	0.02032
10–step	0.05967	4.68845	0.19786	0.34535	0.35592

Table 3. Mean squared error of the NPCP (deg^2)					
	Sinusoids	Triangular	Parabolic	Cubic	Saccades
1–step	0.00000	0.01014	0.00009	0.00016	0.00038
5–step	0.00014	1.34976	0.01922	0.03648	0.03618
10–step	0.00634	24.70490	0.44180	0.83840	0.78299

The sinusoid seemed to be the easiest for all predictors. While the triangular wave with its abrupt turn arrounds was the most difficult for all predictors. The NPCP predicted the sinusoids the best. The TPLP predicted the triangular (linear) wave the best. While the FPQP predicted the parabolic (quadratic), cubic and saccades the best. We think the FPQP did better on the cubic waveform than the NPCP, because of the junctions between the individual cubic segments that were joined to form the periodic waveform. None of these predictors did well for long range prediction, e.g., 20 or 100 steps; other predictors worked better in this range [14, 16].

5. Conclusion

In conclusion, we have presented a general technique to design predictors without the need for matrix inversions. Our technique can compute an arbitrary number of points into the past, present or future. It can be used for position, velocity, or acceleration. The general methodology is illustrated in particular formulae.

Appendix: A recursive identity

In deriving particular formulas (as for example in Example 1), closed form representations of the sum as

$$S_\ell = \sum_{k=0}^{N-1} k^\ell \quad (\ell \geq 0)$$

seemed to be useful. Such representations can be derived in a recursive manner as follows.

Obviously $S_0 = N - 1$, and therefore the initial sum is known. Consider now the identity

$$(k+1)^{\ell+1} = k^{\ell+1} + \binom{\ell+1}{\ell} + \ldots + \binom{\ell+1}{1} k^1 + k^0,$$

which is a simple application of the binomial theorem. By substituting $k = 1, 2, \ldots, N-1$ into the above identity we obtain the following equations

$$\begin{aligned}
2^{\ell+1} &= 1^{\ell+1} + \binom{\ell+1}{\ell} 1^\ell + \ldots + \binom{\ell+1}{1} 1^1 + 1^0 \quad (k=1) \\
3^{\ell+1} &= 2^{\ell+1} + \binom{\ell+1}{\ell} 2^\ell + \ldots + \binom{\ell+1}{1} 2^1 = 2^0 \quad (k=2) \\
&\vdots \\
(N-1)^{\ell+1} &= (N-2)^{\ell+1} + \binom{\ell+1}{\ell} (N-2)^\ell + \ldots \\
&\quad + \binom{\ell+1}{1} (N-2)^1 + (N-2)^0 \quad (k = N-2) \\
N^{\ell+1} &= (N-1)^{\ell+1} + \binom{\ell+1}{\ell} (N-1)^\ell + \ldots \\
&\quad + \binom{\ell+1}{1} (N-1)^1 + (N-1)^0 \quad (k = N-1).
\end{aligned}$$

By adding these equations and cancelling the terms $2^{\ell+1}, 3^{\ell+1}, \ldots, (N-1)^{\ell+1}$ from both sides we obtain the equation

$$N^{\ell+1} = 1 + \binom{\ell+1}{\ell} S_\ell + \binom{\ell+1}{\ell-1} S_{\ell-1} + \ldots + \binom{\ell+1}{1} S_1 + S_0,$$

from which we derive that

$$(A1) \quad S_\ell = \frac{1}{\ell+1} \left[N^{\ell+1} - 1 - \binom{\ell+1}{\ell-1} S_{\ell-1} - \ldots - \binom{\ell+1}{1} S_1 - S_0 \right].$$

This equation gives a recursive way to compute the quantities S_ℓ.

Recursion (A1) is illustrated as

$$\begin{aligned} S_1 &= \frac{1}{2}\left[N^2 - 1 - S_0\right] = \frac{1}{2}\left[N^2 - 1 - (N-1)\right] = \frac{N(N-1)}{2}; \\ S_2 &= \frac{1}{3}\left[N^3 - 1 - \binom{3}{1} S_1 - S_0\right] \\ &= \frac{1}{3}\left[N^3 - 1 - 3\frac{N(N-1)}{2} - (N-1)\right] \\ &= \frac{1}{3}(N-1)\left[N^2 + N + 1 - \frac{3N}{2} - 1\right] \\ &= \frac{N-1}{3}\frac{2N^2 - N}{2} = \frac{N(N-1)(2N-1)}{6}; \end{aligned}$$

and similarly

$$\begin{aligned} S_3 &= \frac{1}{4}\left[N^4 - 1 - \frac{4}{2}S_2 - \frac{4}{1}S_1 - S_0\right] \\ &= \frac{1}{4}\left[N^4 - 1 - 6\frac{N(N-1)(2N-1)}{6} - 4\frac{N(N-1)}{2} - (N-1)\right] \\ &= \frac{N-1}{4}\left[N^3 + N^2 + N + 1 - N(2N-1) - 2N - 1\right] \\ &= \frac{N-1}{4}\left[N^3 - N^2\right] = \frac{N^2(N-1)^2}{4}, \end{aligned}$$

and so on.

References

[1] W.E. Thompson, G.M. Flachs and T.R. Kiang, *Evaluation of filtering and prediction techniques for real-time video tracking of high performances missiles*, Proceeding of NECON, 1978, 897–904.

[2] B.D.O. Anderson and J.B. Moore, *Optimal Filtering*, Englewood Cliffs, NJ: Prentice Hall, 1979.

[3] B. Widrow, J. McCool and M. Ball, *The complex LMS algorithm*, Proc. IEEE, **63** (1975), 719–720.

[4] B. Widrow and E. Walach, *On the statistical efficiency of the LMS algorithm with nonstationary inputs*, IEEE Trans. Information Theory, Vol. IT–30, Special Issue on Linear Adaptive Filtering, 1984, 211–221.

[5] D.T.L. LEE, M. MORF and B. FRIEDLANDER, *Recursive least-squares ladder estimation algorithms/*, IEEE Trans. Circuits and Systems, Vol. CAS–28, 1981, 467–481.

[6] D.T.L. LEE, B. FRIEDLANDER and M. MORF, *Recursive ladder algorithms for ARMA modeling*, IEEE Trans. Automatic Control, Vol. AC–27, 1982. 753–764.

[7] M.L. HONIG and D.G. MESSERSCHMITT, *Convergence properties of an adaptive digital lattice filter*, IEEE Trans. Circuits and Systems, Vol. CAS–28, 1981, 482–493.

[8] B. PORAT and T. KAILATH, *Normalized lattice algorithms for least–squares FIR system identification*, IEEE Trans. Acoust., Speech and Signal Processing, Vol. ASSP–31, 1983, 122–128.

[9] T. LARITZ, *Adaptive prediction techniques that permit human like performance of eye movement simulation*, Master Thesis for the Department of Electrical Engineering, Camegie–Mellon University, 1984.

[10] M.L. HONIG and D.G. MESSERSCHMITT, *Adaptive filters: structures, algorithms and applications*, Hingham, MA: Kluwer Academic Publishers, 1984.

[11] B. FRIEDLANDER, *Lattice filter for adaptive processing*, Proc. IEEE, Vol. **70**, 1982, 829–866.

[12] D.R. HARVEY and A.T. BAHILL, *Development and sensitivity analysis of adaptive predictor for human eye movement model*, Trans. of the Society for Computer Simulation, Vol. **2** (1986), 275–292.

[13] A.J. MONROE, *Digital Processes for Sampled Data Systems*, New York: John Wiley & Sons, Inc., 1962.

[14] C.Y. WU, *Long–range predictors for saccadic eye movements*, Ph. D. Dissertation, Department of Systems and Industrial Engineering, University of Arizona, 1988.

[15] A.T. BAHILL and J.D. MCDONALD, *Smooth pursuit eye movements in response to predictable target motions*, Vision Res., Vol. **23** (1983), 1573–1583.

[16] C.Y. WU and A.T. BAHILL, *Block–processing adaptive filters for human eye movements*, Computers and Industrial Engineering, Vol. **17** (1989), 496–501.