FIXED POINT IMPLEMENTATIONS OF FAST KALMAN ALGORITHMS

Louis L. Scharf
Department of Electrical Engineering
University of Rhode Island

Sigurdur Sigurdsson
Department of Electrical Engineering
Colorado State University

November 1983

Prepared for

OFFICE OF NAVAL RESEARCH (Code 411 SP)
Statistics and Probability Branch
Arlington, Virginia 22217
under Contract N00014-82-K-0300
Louis L. Scharf, Principal Investigator.

DISTRIBUTION STATEMENT A
Approved for public release; Distribution Unlimited.

83 11 15 027
In this paper we study scaling rules and round-off noise variances in a fixed point implementation of the Kalman predictor for an ARMA time series observed noise-free. The Kalman predictor is realized in a fast form that uses the so-called fast Kalman gain algorithm. The algorithm for the gain is fixed point.

Scaling rules and expressions for rounding error variances are derived. The numerical results show that the fixed point realization performs very
closely to the floating point realization for relatively low-order ARMA time series that are not too narrowband.

The predictor has been implemented in 16-bit fixed point arithmetic on an INTEL 8086 microprocessor, and in 16-bit floating point arithmetic on an INTEL 8080. Fixed point code was written in ASSEMBLY language and floating point code was written in FORTRAN. Experimental results were obtained by running the fixed and floating point filters on identical data sets. All experiments were carried out on an INTEL MDS 230 Development System.
FIXED POINT IMPLEMENTATIONS OF FAST KALMAN ALGORITHMS

Louis L. Scharf

Department of Electrical Engineering
University of Rhode Island
Kingston, RI 02881

Siguurður Sigurðsson

Department of Electrical Engineering
Colorado State University
Fort Collins, CO 80523

SUMMARY

In this paper we study scaling rules and round-off noise variances in a fixed point implementation of the Kalman predictor for an ARMA time series observed noise-free. The Kalman predictor is realized in a fast form that uses the so-called fast Kalman gain algorithm. The algorithm for the gain is fixed point.

Scaling rules and expressions for rounding error variances are derived. The numerical results show that the fixed point realization performs almost as well as the floating point realization for relatively low-order ARMA time series that are not too narrow-band.

The predictor has been implemented in 16-bit fixed point arithmetic on an INTEL 8086 microprocessor, and in 16-bit floating point arithmetic on an INTEL 8080. Fixed point code was written in ASSEMBLY language and floating point code was written in FORTRAN. Experimental results were obtained by running the fixed and floating point filters on identical data sets. All experiments were carried out on an INTEL MDS 230 Development System.

Introduction

Finite-dimensional Gaussian time series have stationary Markovian state-space descriptions. In such descriptions the initial conditions are multivariate normal and state variables are predicted values of the series based on an infinite past of observations. The linear filtering problem is one of estimating the state at time t based on observations up to time t and the prediction problem is one of predicting the state at time t+1 based on observations up to time t.

Corresponding to the Markovian representation is the innovations representation. The essential characteristic of this nonstationary representation is that it may be used to synthesize a time-series, starting from zero initial conditions, whose second-order statistics match the statistics of the original time series. The states are predicted values of the time series based on a finite past of observations. Using this representation, the Kalman predictor may be written down from inspection as the causal and stable inverse of the representation. The so-called Kalman gain is the innovations representation (or equivalently in the Kalman predictor) may be associated either with the Levinson recursions for factoring the inverse of the correlation matrix for the time series or with the Le Roux-Gueguen recursions for factoring the correlation matrix itself. The latter association leads to a fixed point algorithm for computing Kalman gains. This so-called fast algorithm produces a fast Kalman filter.

In this paper we present results from a study of fast Kalman predictors, implemented in fixed point and in fixed point arithmetic, for autoregressive moving average time series. More extensive results of this study, for noisy and noise-free filtering and prediction, may be found in the thesis of Sigurðsson [1].


Signal Models for Stationary ARMA Time Series

A zero-mean, second-order, stationary, time series \( y(t) \) is said to be autoregressive moving average (ARMA) if the entries \( y(t) \) in the time series obey this recursion for all \( t \):

\[
\sum_{n=0}^{p} a(n) y(t-n) = \sum_{n=0}^{q} b(n) u(t-n) + \epsilon(t)
\]

where \( u(t) \) is the exogenous input sequence, \( \epsilon(t) \) is time series white noise, and the \( a(n) \) and \( b(n) \) are polynomials in the backward shift operator \( \delta(k) \). The transfer function of the ARMA model is

\[
H(z) = \frac{\sum_{n=0}^{q} b(n) z^{-n}}{\sum_{n=0}^{p} a(n) z^{-n}} \quad |z| > r
\]

The coefficients \( a(n) \) and \( b(n) \) are the autoregressive and moving average parameters of the model, respectively.

Markovian Representation

The Markovian representation for \( y(t) \) is summarized in the following equations, where \( a = \text{max}(p,q) \), and where for purposes of illustration we have assumed \( q > p \).

This paper was presented at the 22nd IEEE Conference on Decision and Control, San Antonio, TX (December 14-16, 1983). The work was supported by Office of Naval Research, Contract N N0014-82-K-0300.
State Equations

\[ g(t+1/t) = A(t)g(t/t) + h(t) + u(t) \]
\[ g(0/-1) : N(0, Q) \]
\[ y(t) = g(t)g(t/t) + u(t) \]

Unit Pulse Response

\[ h(t) = u(t) \eqno{0, t<0} \]
\[ h(t) = 1, t \geq 0 \]
\[ g^A(t/t-1), t>0 \]

State Covariance

\[ R(t) = A^TQ \]
\[ Q = AQA^T + \sigma^2 h(t)^2(t) \]

Output Covariance

\[ r(t) = g(t)R(t) \]
\[ r(t) = \sigma^2u(t) \]

Design Equations

\[ \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} h(t) = \begin{bmatrix} h(t) \\ h(t) \end{bmatrix} \]
\[ \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \end{bmatrix} \]
\[ a(1) \quad h(1) \quad a(2) \quad h(2) \quad \ldots \quad a(p) \quad h(p) \quad 0 \quad a(p+1) \quad h(p+1) \quad 0 \quad \ldots \quad 0 \quad b(q) \]

Innovations Representation

The idea behind the innovations representation for \( y(t) \) is to replace the stationary initial conditions, distributed as \( h(0/-1) = N(0, Q) \), with the nonstationary initial condition \( x(0/-1) = 0 \), to replace the stationary input vector \( g(t) \) by a nonstationary Kalman gain vector \( h(t) \), and to replace the stationary i.i.d. sequences \( u(t) \), distributed as \( u(t) : N(0, \sigma^2) \), with the nonstationary i.i.d. innovations sequences \( u(t) \), distributed as \( u(t) : N(0, v(t)) \). The trick is to choose the Kalman gain \( h(t) \) and the innovations variance \( v(t) \) correctly. The innovations representation is summarized below. It is worth noting that \( h(t) \rightarrow h(t) \), \( v(t) \rightarrow 0 \), and \( u(t) \rightarrow 0 \) as \( t \rightarrow \infty \).

State Equations

\[ g(t+1/t) = A(t)g(t/t) + h(t) + u(t) \]
\[ g(0/-1) = 0 \]
\[ u(t) : N(0, v(t)) \]
\[ y(t) = g(t)g(t/t) + u(t) \]

Unit Pulse Response

\[ h(t) = u(t) \]
\[ g(t) = 1, u(t) \]

State Covariance

\[ R(t) = A^TQ \]
\[ Q = AQA^T + \sigma^2 h(t)^2(t) \]

Output Covariance

\[ r(t) = g(t)R(t) \]
\[ r(t) = \sigma^2u(t) \]

Design Equations

\[ \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} h(t) = \begin{bmatrix} h(t) \\ h(t) \end{bmatrix} \]
\[ \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \end{bmatrix} \]
\[ a(1) \quad h(1) \quad a(2) \quad h(2) \quad \ldots \quad a(p) \quad h(p) \quad 0 \quad a(p+1) \quad h(p+1) \quad 0 \quad \ldots \quad 0 \quad b(q) \]

In these equations, \( h(t) \) is the response of the system of equations at time \( t \) to an impulse applied at time \( t \), \( R(t) \) is the state covariance between \( y(t) \) and \( y(t+1) \), and \( r(t) \) is the output covariance between \( y(t) \) and \( y(t+1) \). If \( \sigma^2(t) \) is to equal \( r(t) \), to match the second order properties, \( h(t) \) and \( v(t) \) must be chosen as above in the design equations.

Kalman Predictor

In the innovations representation, identify \( g(t+1/t) \) as the prediction \( y(t+1/t) \). Then the Kalman filter equations become simply a rewriting of the innovations representation:

\[ y(t+1/t) = a(t)g(t+1/t) \]
\[ g(t+1/t) = Ag(t+1/t-1) + h(t)g(t+1/t-1) \]
\[ y(t) = y(t) \]
Past Algorithm for $h(t)$

The gain $h(t)$ may be calculated as outlined under the innovations equations by solving a system of Riccati equations. An alternative is to note that the gain is related to the time varying impulse response of the innovations representation:

$$h^t(n) = rA^{n-1}h(t), \ n \geq 0$$

This means the elements of $h(t)$ may be read out as follows:

$$
\begin{bmatrix}
  h^t(1) \\
  h^t(2) \\
  \vdots \\
  h^t(n)
\end{bmatrix}
= 

\begin{bmatrix}
  h^t(1) \\
  h^t(2) \\
  \vdots \\
  h^t(n)
\end{bmatrix}
\times

\begin{bmatrix}
  h(t) \\
  y(t)
\end{bmatrix}
$$

As the correlation matrix for $y(t)$ is related to the time varying impulse responses, we may write

$$R = H^TVH^T$$

where the matrices $R$, $V$, and $H$ are defined as follows:

$$R = \begin{bmatrix} r_{11} & \ldots & r_{1n} \\ \vdots & \ddots & \vdots \\ r_{n1} & \ldots & r_{nn} \end{bmatrix}$$

$$V = \text{diag}(v(1), \ldots, v(n))$$

$$H = \begin{bmatrix}
  h^0(t) & h^1(t) & \ldots & h^n(t)
\end{bmatrix}$$

This means the correlation matrix may be factored with the fast impulse response algorithm of LoRenzo-Guevens [6] to obtain $H$, and $n$-dimensional columns of $H$ may be picked off to obtain $h(t)$.

Scaling and Rounding in the Kalman Predictor

Let $\delta$ denote quantization step size and $[\delta/2]$ the bound on the maximum magnitude that may be represented in an $m$-bit, signed binary representation. The problem in a finite-word length realization of the Kalman predictor is to scale variables so that the probability of overflow is small.

Scaling

A filter is said to be $l_2$ scaled if

$$o^2 \sum_0 f_k(\delta) = \| \delta^{2n-1/2} \|^2$$

where $\delta$ is a parameter that may be increased to force the norm of $f_k$ to be small, and thereby decrease the probability of overflow. The parameter $\delta$ is the variance of the noisy excitation of the filter.

In the Kalman predictor equation we may scale the state vector $g(t|t-1)$ by a diagonal scaling matrix $S$:

$$S = \text{diag}(S(1), \ldots, S(n))$$

The scaled Kalman predictor equations are then

$$g(t|t-1) = S^{-1}A_S g(t|t-1) + S^{-1}h(t) u(t)$$

$$y(t|t-1) = S \cdot\ S g(t|t-1)$$

The $l_2$ scaling rules for the states of the Kalman predictor are

$$S^{-1}(Q(t))_k S^{-T} \cdot \| \delta^{2n-1/2} \|^2$$

The state variance $Q(t)$ may be computed recursively as outlined previously, or recursively as

$$Q(t) = v(t) \sum_{n=1}^{t} \delta^{n-1} h(t-n)^T h(t-n)$$

The diagonal terms of $Q(t)$ converge monotonically to the upper limit $Q = \delta \delta^T \cdot \delta^{n-1} h(t-n)^T h(t-n)$.

A practical procedure is to replace $Q(t)$ by $Q$ to obtain the scaling rule

$$S(k) = [Q]^{1/2} \delta/2^{n-1} \quad (k=1,2, \ldots, n)$$

This result for stationary state space filters is due to Mullis and Roberts [5].

The algorithm for the gain need not be scaled because it is fixed point: all internal variables are bounded by unity in magnitude.

Rounding

The updating of the state vector in the Kalman predictor requires one multiply for the first $(n-1)$ elements and $(n+1)$ multiplies for the $n$th element. By generating a sequence of i.i.d., variance $\delta^2/12$, random variables with each fixed point multiply, we generate a mean zero, variance $N$ random vector $g(t)$ each time we update.
\[
g(t+1) = g(t)(t+1) + \frac{1}{2} g(t-1) + g(t)
\]

\[
E g(t) g(t+h) = N(h)
\]

\[
N = \begin{bmatrix}
1 & 1 \\
\frac{1}{2} & 1
\end{bmatrix}
\]

The state variance is now

\[
Q(t+1) = (g(t+1)^2) Q(t) (g(t+1)^2) + \frac{1}{4} Q(t-1)
\]

\[
= (g(t+1)^2) Q(t) (g(t+1)^2) + N
\]

**Numerical Experiments**

Refer to the abstract for a summary of how all experiments were conducted. What follows is a brief annotation of Figures 1 through 4.

Experiments were conducted by generating realizations of a stationary time series from the model

\[
E(z) = \frac{1-1.75z^{-1} + 0.5z^{-2}}{1-1.5z^{-1} + 1.21z^{-2} - 0.4590z^{-3}}
\]

Figure 1 illustrates \(h(t)\), a 31 vector, computed with floating point arithmetic (curves 1-3) and with fixed point arithmetic (curves 4-6). Figure 2 illustrates the innovation variance \(v(t)\) computed in floating point (curve 1) and in fixed point. These two curves illustrate that the data-independent fast Kalman gain calculation may be practically computed in fixed point using 16-bits.

Figure 3 illustrates predictions \(y(t|t-1)\) in the floating point realization (curve 1) and in the scaled, fixed point realization of the Kalman predictor.

Figure 4 illustrates the corresponding innovation sequences. The floating point and fixed point predictors were run over the same realization of the time series.

**References**


Figure 2. The innovations variance $v(t)$ calculated using floating point arithmetic (curve 1) and fixed point arithmetic (curve 2).

Figure 3. The predicted output $y(t(t-1))$ calculated using floating point arithmetic (curve 1) and fixed point arithmetic (curve 2).

Figure 4. The innovations sequence $u(t)$ calculated using floating point arithmetic (curve 1) and fixed point arithmetic (curve 2).