Real-Time Implementation of a Narrow-Band Kalman Filter With a Floating-Point Processor DSP32

HEN-GEUL YEH, SENIOR MEMBER, IEEE

Abstract—This paper presents the experimental results of two studies. First, a real-time narrow-band Kalman filter is implemented with a floating-point digital signal processor DSP32. The real-time capability of this narrow-band filter is investigated by varying parameters $Q$ and $R$. The covariance matrices $Q$ and $R$ of the dynamic and measurement noise sequences are found to exhibit duality in the real-time tuning process and have a direct effect on system stability. If the value of $Q$ used is smaller (with fixed $R$), the tracking time and the narrower tracking bandwidth of the filter will be longer. In addition, if the value of $R$ used (with fixed $Q$) is smaller, the tracking time will be smaller, and the tracking bandwidth of the filter will be larger. The results are tabulated.

Second, two optimal codes (in the sense of the executional speed), straight-line code and general matrix-based code, have been developed for implementing the narrow-band Kalman filter. These two codes are compared in terms of program memory size, data memory size, and speed of execution. With the matrix-based code, the DSP32 performance is evaluated in terms of speed and memory size by varying the number of states of a Kalman filter. The results are also tabulated.

I. INTRODUCTION

FOR MORE than two decades, Kalman filters have been applied extensively in many signal processing applications; these include target tracking, adaptive control, radar signal processing, and navigation systems. The applicability of the Kalman filter to real-time signal processing problems is generally limited by the relatively complex mathematical operations necessary in computing the Kalman filtering algorithm. However, with the rapid development of VLSI integrated circuits, it has become technologically feasible to implement Kalman filters in real-time with DSP processors.

The processing of a Kalman filter requires matrix/vector operations such as multiplications, additions, subtractions, and inversions. Among these, the matrix inversion is the most difficult to implement in terms of the execution speed and numerical accuracy required due to a recursive feedback loop in the structure of Kalman filters. Any computational error can easily accumulate and may result in overflow. Thus, both the system dynamics and measurement system must be accurately modeled numerically. These problems are less critical when a floating-point digital signal processor is employed. Later in this paper, we will discuss the performance of the DSP32, which is used in this experiment, and its advantages over fixed-point digital signal processor chips in implementing a real-time recursive Kalman filter.

In this paper, the discussion will go as follows: Section II gives the Kalman filter algorithm and an intuitive reason for $Q$-$R$ duality. Section III describes the formulation of a narrow-band tracking filter using both direct and normal form realizations. Section IV discusses the implementation consideration with a floating-point processor DSP32. The software implementation techniques, both straight-line code and general matrix-based code, are discussed and compared in detail in Section V. The evaluation of the real-time Kalman filter performance is described in Section VI. The conclusion is given in Section VII.

II. KALMAN FILTER ALGORITHM AND REASON FOR
$Q$-$R$ DUALITY [1]

Kalman filters have been shown to be the optimal linear estimator in the least-square sense for estimating dynamic system states in linear systems. The Kalman filter updates state estimation based on prior estimates and observed measurements. The filter consists of the model of the dynamic process that performs the function of the prediction and a feedback correction scheme. The measurements can be processed as they occur, and there is no need to store any measurement data. However, all the associated matrices that describe the system dynamic, measurement system, and noise sequences are assumed to be known. The discrete time-varying Kalman filtering process involves the propagation of state estimates and error covariance matrices from one time sample to the next. Discussion and applications on Kalman filters can be found widely in the literature [1]-[4].

The following equations define a general dynamic system and a measurement system:

\[ x(k + 1) = A(k) \ast x(k) + B(k) \ast w(k) \]  
\[ y(k + 1) = C(k + 1) \ast x(k + 1) + v(k + 1) \]  

In (1), $A(k)$ is a $n \times n$ state transition matrix, which describes the dynamic plant; $x(k)$ is the state vector with $n$ dimensions; $w(k)$ is a $p$-dimensional system dynamic noise sequence, and $B(k)$ is a $n \times p$ matrix, which describes the impact of $w(k)$ on the system dynamic. In (2), $y(k)$ is an $m$ vector called the measurement vector. $C(k)$ is an $m \times n$ matrix that describes the measurement system, and $v(k)$ is an $m$ vector known as...
the measurement noise sequence. The noise sequences \( w(k) \) and \( v(k) \) are assumed to be independent, zero mean, and white noise sequences with covariance matrices \( Q(k) \) and \( R(k) \), respectively. The Kalman filter is described by the following equations under the assumptions that matrices \( A(k), B(k), C(k), Q(k), \) and \( R(k) \) are available. For simplicity, we consider a time-invariant case and use \( A, B, C, Q, \) and \( R \) as the matrix notations. The following equations describe a Kalman filter that estimates the state vector \( x(k) \) of (1):

\[
P_1(k) = A \ast P_1(k-1) \ast A^T + B \ast Q \ast B^T \]

(3)

\[
K(k) = P_1(k) \ast C^T \ast [C \ast P_1(k) \ast C^T + R]^{-1}
\]

(4)

\[
\dot{x}(k) = A \ast \dot{x}(k-1)
\]

(5)

\[
\dot{x}(k) = \dot{x}(p(k)) + K(k) \ast [y(k) - C \ast \dot{x}(p(k))]
\]

(6)

\[
P(k) = P_1(k) - K(k) \ast C \ast P_1(k)
\]

(7)

\( k = 1, 2, \ldots \)

Initial conditions \( \dot{x}(0) \) and \( P(0) \) are given in general.

Notice that matrices \( P_1(k) \) and \( P(k) \) are commonly called error covariance matrices. The vector \( \dot{x}(k) \) represents the optimal estimate of the state \( x(k) \) based on the measurement sequences \( \{y(1), y(2), \ldots, y(k)\} \). Equations (3) and (5) are referred to as time updates, and (6) and (7) are referred to as measurement updates. These filter equations are computed in order as listed.

A. An Intuitive Reason for the Q-R Duality

Recall (4) for the expression of the Kalman gain. Apparently, variation of the gain \( K(k) \) is related to variations in \( P_1(k) \) and \( R \). However, if we recall (3) for the error covariance matrix \( P_1(k) \), we see that the uncertainty in the dynamic model can be characterized by the process noise covariance matrix \( Q \). For large \( Q, P_1(k) \) is large, which indicates high uncertainty or an inadequate model of the dynamic system. For small \( Q, P_1(k) \) is small, which indicates an adequate model. Hence, we can heuristically conceive of the Kalman gain as a ratio of dynamic process to measurement noise (for \( C = I \)), that is

\[
K \text{ is proportional to } \frac{Q}{R}.
\]

Intuitively, we can perceive the Kalman filter as a deterministic filter with a time-varying bandwidth determined by Kalman gain. This can be seen by rewriting (6) as

\[
\dot{x}(k) = [I - K(k)C] \dot{x}(k-1) + K(k)y(k).
\]

(6.1)

Clearly, we see that as \( Q \) increases, \( K(k) \) increases as the filter bandwidth increases. Thus, the filter transient performance is fast, and the tracking time is shorter. The tracking time is defined as the time that starts from the start of the signal tracking to the time when the filter has steady-state estimates at its output. As \( R \) increases, \( K(k) \) decreases, and the filter bandwidth decreases. Thus, the filter transient performance is slow, and tracking time is long. Consequently, the Q-R duality affects both the tracking time and tracking bandwidth of the Kalman filter in real time. This agrees with what was observed in the experiment and is discussed in the following sections.

III. PROBLEM FORMULATION

To experimentally study the real-time narrow-band tracking capability of a Kalman filter, a second-order digital cosine wave generator is developed and used as the system dynamic model. The received signal is a sinusoidal wave corrupted by noise. The Kalman filter is used to track the narrow-band signal. A block diagram of the laboratory setup is shown in Fig. 1.

Two realization (normal and direct) forms of the system model, which represent both dynamic and measurement systems with noise sequences, are shown in Fig. 2. Mathematical models of the dynamic system and measurement are obtained from realization forms, which are illustrated in Fig. 2(a) and (b), respectively. They include the state vector of both realization forms

\[
\begin{bmatrix}
    x_1(k) \\
    x_2(k)
\end{bmatrix}
\]

the normal form

\[
A = \begin{bmatrix}
    a_1 & a_2 \\
    -a_2 & a_1
\end{bmatrix}, \quad B = \begin{bmatrix}
    1 & 0 \\
    0 & 1
\end{bmatrix}, \quad C = \begin{bmatrix}
    1 & 0 \\
    0 & 0
\end{bmatrix}
\]

\[
Q = \begin{bmatrix}
    Q_0 & 0 \\
    0 & Q_0
\end{bmatrix}, \quad R = R_0 (\text{scalar})
\]
and the direct form

\[ A = \begin{bmatrix} 2 & a1 & -1 \\ 1 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 \end{bmatrix} \]

where

\[ Q = Q0 \text{ (scalar)} = 2.368475786 \times 10^{-15} \]
\[ R = R0 \text{ (scalar)} = 7.761021458 \times 10^{-11} \]
\[ a1 = \cos 9^\circ = .9876883406 \]
\[ a2 = \sin 9^\circ = .1564344650. \]

Notice that the Kalman filter is designed to track a 200-Hz sinusoidal signal with an 8000-Hz sampling rate. Consequently, parameters \( a1 \) and \( a2 \) are determined based on 40 samples per cycle. Parameters \( Q0 \) and \( R0 \) are based on the rounding errors (addition) of the 24-bit mantissa of the DSP32 processor and the quantization error of the 16-bit D/A quantizer, respectively. Both normal and direct form models are implemented as the dynamic system of the Kalman filter. In this experiment, a PCM codec (M7708E) with a 16-bit A/D and D/A conversion is used in the DSP32 development system as an interface device.

The transfer function of the dynamic system is

\[ X(z) = H1(z)W(z) = (Iz - A)^{-1}BW(z) \]

where \( X(z) \) and \( W(z) \) are the \( z \) transform of the state vector \( x(k) \) and the noise sequence \( w(k) \), respectively.

The transfer function of the measurement system is (assume \( v(k) = 0 \))

\[ Y(z) = H2(z)X(z) = CX(z) \]

where \( Y(z) \) is the \( z \) transform of the output \( y(k) \). The overall transfer function is

\[ H(z) = H2(z)H1(z) = C(Iz - A)^{-1}B. \]

Obviously, the characteristic equation of the overall system, obtained directly from \( H(z) \), to have a sinusoidal signal, one can choose pole location on an unit circle as a complex pair.

The following initial conditions are employed for the real-time testing of both normal and direct forms:

\[ \hat{x}(0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \]

\[ P(0) = \begin{bmatrix} 5 & 0 \\ 0 & 1 \end{bmatrix} \]

IV. IMPLEMENTATIONAL CONSIDERATIONS WITH A DSP32 [5]-[7]

It is necessary to study and choose an appropriate DSP chip for implementing the Kalman filter, as described in Section II, in terms of memory size, speed, numerical accuracy, and easy use of the assembly language. AT&T's DSP32 digital signal processor has been employed for the Kalman filter implementation in this paper.

A. Memory and Performance Consideration

The memory space is important. In general, one may prefer to have a large on-chip ROM/RAM for both program and data memory to reduce board complexity and size and to shorten the data and instruction access time. Typically, instructions and fixed operands are stored in ROM and variable operands in RAM. A large-dimension Kalman filter with multiple measure sensors can be both instruction and data-storage intensive. In a high-performance global positioning system (GPS) navigation [8], for example, the Kalman filter may need to update as many as 11 state variables (position, velocity, clock bias, clock drift, and acceleration/platform tilts). The DSP32 provides on-chip memory that includes a 512-by-32-bit ROM and a 1024-by-32-bit RAM.

B. Speed

The time required for a multiply/accumulate operation is generally between 250 and 30 ns for all DSP chips. All instructions are executed in one instruction cycle of a DSP32. A multiply/accumulate with a data move operation takes a single instruction cycle (250 or 160 ns due to different clock rates) to execute. As described in [5], the DSP32 memory is divided into two banks: an upper bank and a lower bank. Memory accesses can be made to either of these two banks. However, to achieve maximum throughput, memory access must alternate between the upper bank and the lower bank. In general, the real-time minimal throughput must satisfy the Nyquist rate.

C. Numerical Accuracy

Finite word-length registers of the DSP chip affect the accuracy of the filter [9]. These effects are found in quantization errors and limit cycles. Quantization errors such as those due to analog-to-digital data conversion are influenced by the numbering system used to encode data (e.g., 2's complement, floating point, etc.). With recursive algorithms and infinite impulse response digital filters, the roundoff or truncation errors can build up and hurt the performance of the filter, whereas large-scale overflow limit cycles are caused by, as the name implies, a system state (or variable) exceeding a prespecified dynamic range; this source of error can be disastrous if it is not successfully treated.

To minimize the effects of finite word lengths, there are several approaches available. One approach is to properly scale the filter variables and parameters for each sample point. Another approach is to use the floating-point processor with large dynamic range. For example, the dynamic range of TMS320C25 (fixed-point processor) and DSP32 (floating-point processor) is \( 2^{-15} \approx 1 \) and \( 10^{-30} \approx 10^{38} \), respectively. For this paper, the latter option has been chosen [5]; the 8-bit exponent in the 32-bit DSP32 chip, floating-point word yields a large dynamic range that makes overflow unlikely, whereas a 24-bit normalized mantissa gives high precision, which is independent of magnitude. In addition, the floating-point pro-
second processor contributes to its ease of use because it frees programmers from concern about intermediate scaling to avoid overflow or loss of precision. In fact, computer simulations with single precision can easily and closely predict the numerical results of the real-time Kalman filter since both DSP32 and single precision have the same number of bits of both exponent and mantissa. Therefore, the analysis for numerical errors is not necessary when using a DSP32 chip.

D. DSP32 Language

Most DSP devices are coded by using assembly language. However, AT&T's DSP16 and 32 are programmed by using C-like syntax language. In general, the use of high-level language for coding can actually shorten the developmental time and simplify the debugging process.

V. SOFTWARE IMPLEMENTATION TECHNIQUES AND PERFORMANCE COMPARISON

Two different programs (namely, straight-line and matrix-based codes) are developed. The straight-line code is a software coding method, which implements straightforward algorithms with limited subroutines or without subroutines. An alternative coding method is to take the module-based approach, which forms and calls subroutines for matrix operations. The consideration of implementing a Kalman filter involves the total executional speed, program memory size, and data memory size. In a general sense, a straight-line code may speed up the executional time due to the nature of a straight code. It is, however, difficult to adjust the assembly program to fit a different number of the states of a Kalman filter. This is due to the need for a new addressing arrangement for implementation. However, it is very easy to adjust the matrix-based code to fit a different number of states of a Kalman filter because the number of states can be a variable that is passed between main program and subroutines. In the matrix-based code, the following subroutines are employed:

1) MATAS: matrix addition/subtraction (5)
2) MATRAN: matrix transpose (4)
3) MATMUL: matrix multiplication (11)
4) DIVE: fast divide routine (1)

The (.) indicates the number of calls on each subroutine during one iteration of the Kalman filter execution. The signal flowchart of a general Kalman filter is given in Fig. 3. Notice that if a large-size system is implemented, an additional subroutine that forces symmetry of the P(k) matrix may be needed. Two Kalman filter codes (the straight-line and matrix-based approaches) have been developed for the problem of tracking a narrow-band signal, as discussed in Section III, for both direct and normal forms. The performance of a Kalman filter with two different realizations shows almost identical numerical results. This is due to the fact that a floating-point processor is used for implementing the two-state Kalman filter. The performance of the DSP32 programs based on direct form realization is summarized in Tables I and II. Notice that the Kalman filter with a scalar measurement is considered in both Tables I and II.

In Table I, a comparison in terms of speed and memory sizes is made between straight-line and matrix-based codes. Clearly, the straight-line code for a two-state Kalman filter with a scalar measurement is better in both program memory size and execution time as compared with the matrix-based code. In Table II, the DSP32 performance is evaluated by varying the number of states of a Kalman filter with scalar measurement. Apparently, the speed decreases when the number of states increases. However, the program size is kept constant due to the module-based approach. On the other hand, if there are more states, the program size is required to be larger in the straight-line code.

VI. REAL-TIME KALMAN FILTER PERFORMANCE

We summarize the results of this experimental study as follows:

1) The results from using two different realization forms (direct form and normal form) are numerically similar. This
Table III

<table>
<thead>
<tr>
<th>Matrices $Q$ and $R$ Values</th>
<th>Tracking Time (ms)</th>
<th>2 dB Tracking Bandwidth (Hz)</th>
<th>Variation Around (20 Hz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_0$</td>
<td>150</td>
<td>$10$</td>
<td>$10$</td>
</tr>
<tr>
<td>$R_1$</td>
<td>250</td>
<td>$10$</td>
<td>$10$</td>
</tr>
<tr>
<td>$R_2$</td>
<td>150</td>
<td>$10$</td>
<td>$10$</td>
</tr>
<tr>
<td>$R_3$</td>
<td>150</td>
<td>$10$</td>
<td>$10$</td>
</tr>
<tr>
<td>$R_4$</td>
<td>150</td>
<td>$10$</td>
<td>$10$</td>
</tr>
</tbody>
</table>

is due to the fact that DSP32 is a floating-point processor, which naturally provides large dynamic range.

2) Fig. 4 shows a 200-Hz sine wave corrupted with additive white noise ($S/N = 3.279$ dB), and Fig. 5 shows the output of the Kalman filter $x(k)$. It is clear that the Kalman filter tracks the 200-Hz sine wave very well. This is due to the precise modeling on the signal and on the quantization noise. Fig. 6 shows the spectrum of the received signal $y(k)$ (before $A/D$), and Fig. 7 shows the spectrum of the output signal $x(k)$ (after $D/A$) of the Kalman filter using a spectrum analyzer. Obviously, the signal-to-noise ratio is improved by about 30 dB.

3) Table III shows the duality principle of varying matrices $Q$ and $R$ of the Kalman filter. It is obvious that both the tracking time and tracking bandwidth (variation around 200 Hz at steady state) of the tracking filter are changed by varying either $Q$ or $R$ or both. If the value of $Q$ used (with fixed $R$) is smaller, then the tracking time will be longer, and the tracking bandwidth of the filter will be narrower. By duality, if the value of $R$ used (with fixed $Q$) is smaller, then the tracking time will be smaller, and the tracking bandwidth of the filter will be larger.

4) A fourth-order Kalman filter is also developed and tested in real time. It performs the same duality property as the second-order one.

VII. Conclusion

This paper demonstrated that a simple second-order Kalman filter can be modeled to track a sinusoidal signal corrupted by noise. The modeling problems involved are the determination of the transition matrix (covariance matrices $Q$ and $R$). However, the transition matrix is found by employing second-order realization forms of the desired signal. Both the $Q$ and $R$ matrices are determined by quantization errors. Furthermore, experiments show the duality property between matrices $Q$ and $R$ of the Kalman filter in real time. By varying parameters $Q$ and $R$, both the tracking time and the tracking bandwidth of the filter are significantly affected.

Two optimal codes (a straight-line code and a matrix-based code) have been developed, tested, and discussed in this paper. The straight-line code for a two-state vector with a scalar measurement is better in both program memory and execution time as compared with the matrix-based code. However, the matrix-based code can easily be adjusted to fit a different number of states of the Kalman filter.
With the matrix-based code, the DSP32 performance is evaluated in terms of speed and memory size by varying the number of the states of a Kalman filter from 2 to 9. The results are tabulated.

REFERENCES


