論 文
39～10～6

이산 칼만 필터의 병렬처리 구조

A Parallel Processing Structure for the Discrete Kalman Filter

金 瑢 俊*·李 章 揆**·金 炯 中***
(Yong-Joon Kim · Jang-Gyu Lee · Hyoung-Joong Kim)

요 약

현대제어, 신호처리, 그리고 통신 등의 분야에서 널리 적용되는 필터기법 중의 하나인 칼만필터에 대한 병렬처리 알고리즘을 제안하였다. 칼만필터의 많은 계산량을 줄이기 위하여, 병렬성을 도입한 기존의 알고리즘들은 주로 측정치의 분산처리에 의한 계층적 구조의 형태이거나 행렬의 계산속도를 향상시키기 위하여 시스톨릭 구조를 사용한 형태이다. 그러나 본 논문이 제안한 병렬 칼만필터는 반복이분법(recursive doubling)과 유사한 방법을 사용한 새로운 병렬 알고리즘이다. 이 알고리즘은 칼만필터의 최적성을 유지하면서도 일반적인 순차적 칼만필터보다 2배의 빠르기로 측정치를 처리하므로 상태변수의 추정치를 참값에 더 빨리 수렴시킨다.

Abstract- A parallel processing algorithm for the discrete Kalman filter, which is one of the most commonly used filtering techniques in modern control, signal processing, and communication, is proposed. To decrease the number of computations critical in the Kalman filter, previously proposed parallel algorithms are of the hierarchical structure by distributed processing of measurements, or of the systolic structure to disperse the computational burden. In this paper, a new parallel Kalman filter employing a structure similar to recursive doubling is proposed. Estimated values of state variables by the new algorithm converge faster to the true values because the new algorithm can process data twice faster than the conventional Kalman filter. Moreover, it maintains the optimality of the conventional Kalman filter.

1. 서 론

칼만필터[1]는 현대제어, 신호처리, 그리고 통신 등의 분야에서 널리 적용되는 필터기법 중의 하나이다. 이것은 백색 가우스 잡음을 가지고 있는 시스템에 있어서 최적필터이며, 현재와 과거의 측정치$\{\underline{z}_1,\cdots,\underline{z}_k\}$로부터 가능한 한 가장 정확한 시스템 상태변수 $\underline{x}_k$의 추정치를 순환적으로 얻는 기술이다. 그러나 칼만필터의 실시간 응용은 복잡한 행렬 계산식으로 구성된 필터 알고리즘의 계산량으로 인하여 제한되고 있다. 이의 해결 방법으로

*正 會 員 : 金星社 中央硏究所 勤務
**正 會 員 : 서울大 工大 制御計測工學科 副敎授 · 工博
***正 會 員 : 江原大 工大 制御計測工學科 專任講師 · 工博
接受日字 : 1989年 12月 28日
1次修正 : 1990年 8月 23日

는 모델의 크기를 줄여서 계산량을 줄이는 방법이 있으나, 칼만필터는 모델의 정확도에 민감하여 오차가 발생할 여지가 많다. 그러므로 충분한 크기의 정확한 모델에 대하여 실시간 칼만필터링을 하기 위해서는 병렬성의 도입이 필요하며, 이에 따라 병렬성을 도입하여 계산량을 줄이려는 방법들[3-7]이 제안되었다.

지금까지 이산 칼만필터에 병렬성을 도입한 연구들은 크게 계층적 구조의 알고리즘과 시스톨릭 구조를 사용한 알고리즘으로 나눌 수 있다. 계층적 구조의 알고리즘은 측정치를 여러개의 작은 차수의 그룹으로 나누거나 시간적으로 여러개의 그룹으로 나누어서, 각각을 아래의 하부프로세서에서 처리하여 각각의 추정치를 구하고, 이 값들을 상부프로세서로 전송하여 이곳에서 최적 추정치를 얻는 구조를 가지고 있다. Hassan[3], Desai와 Das[4], Hashemipour[5]등이 이러한 계층적 구조의 알고리즘을 제안하였다. 최근에 VLSI기술의 발달과 더불어 시스톨릭 구조[8]라는 새로운 구조의 병렬처리 기법이 발달하였다. 시스톨릭 구조를 사용하면 행렬의 연산(행렬과 벡터의 곱셈, 행렬과 행렬의 곱셈, 행렬의 삼각화, 행렬의 회전)을 고속으로 처리할 수 있다[7, 9]. 이러한 장점때문에 칼만필터의 복잡한 행렬의 연산에 시스톨릭 구조를 이용하여 성능향상을 얻으려는 연구들이 있었다. 이러한 연구는 Andrews[5]가 Bierman의 LDU분할에 적용한 것이 최초이고, Jover와 Kailath[6], Sung과 Hu[7]등이 이러한 연구들을 하였다. 이들은 제곱근필터의 특징을 이용하여 시스톨릭 구조의 병렬 칼만필터들을 제안하였다.

본 논문에서는 이러한 기존 알고리즘의 형태와는 다르게, Kogge와 Stone[10]이 제안한 순환방정식에 대한 병렬처리 알고리즘의 기본개념을 사용하여 새로운 병렬 칼만필터 알고리즘을 제안한다. Kogge와 Stone은 반복이분법을 사용하여 일반적인 순환방정식에 대한 병렬 알고리즘을 제안하였다. 이들의 알고리즘에 의하면 연속된 N개의 해를 구하는 순환방정식 문제를, N개의 프로세서를 사용할 경우 시간복잡도(time complexity)는 $0(\log_2 N)$이다. 그러나 이들의 알고리즘을 적용하려면 모든 프로세서가 각각 자기의 데이타를 동시에 가지고 있어야 한다. 따라서 만일 데이타가 시간에 따라 순차적으로 입력되는 문제라면, 이들의 알고리즘을 적용할 수 없게 된다. 이산 칼만필터는 순환방정식의 형태이나, 실시간 운영에 적용할 때 시간에 따라 순차적으로 입력되는 측정치가 포함되므로, 실시간으로 운영되는 칼만필터에는 이들의 알고리즘을 적용할 수 없다. 본 논문은 Kogge와 Stone의 알고리즘이 반복이분법에 의하여 순환방정식을 병렬로 해결한 것과 유사하지만, 실시간으로 운영되는 칼만필터의 제약조건에 알맞게 이산 칼만필터에 대한 새로운 병렬처리 알고리즘을 제안한다.

2. 병렬 칼만필터 알고리즘

본 절에서는 새로운 병렬 칼만필터 알고리즘을 제안한다. 다음과 같은 이산 선형시스템(discrete linear system)과 측정방정식(observation equation)을 생각해 보자.

$$\underline{x}_k = \Phi_{k-1}\underline{x}_{k-1} + \underline{w}_{k-1} \tag{1}$$

$$\underline{z}_k = H_k\underline{x}_k + \underline{\nu}_k \tag{2}$$

식 (1)과 식 (2)는 이산 선형시스템과 측정방정식의 가장 일반적인 식으로, 식에서 $\underline{w}_k$와 $\underline{\nu}_k$는 정규분포를 갖는 백색잡음으로, 평균은 $\underline{0}$이고 공분산 행렬은 각각 Q_k, R_k이다. 칼만필터는 상태변수 $\underline{x}_k$와 추정상태변수 $\underline{\hat{x}}_k$의 오차분산 $E\{(\underline{x}_k-\underline{\hat{x}}_k)(\underline{x}_k-\underline{\hat{x}}_k)^T\}$을 최소로 하는 알고리즘으로 다음과 같은 두 단계에 의하여 최적 추정치를 찾게 된다.

time propagation

$$\underline{\hat{x}}_k(-) = \Phi_{k-1}\underline{\hat{x}}_{k-1}(+) \tag{3}$$

$$P_k(-) = \Phi_{k-1}P_{k-1}(+)\Phi_{k-1}^T + Q_{k-1} \tag{4}$$

measurement update

$$\underline{\hat{x}}_k(+) = \underline{\hat{x}}_k(-) + K_k[\underline{z}_k - H_k\underline{\hat{x}}_k(-)] \tag{5}$$

$$P_k(+) = [I - K_kH_k]P_k(-) \tag{6}$$

$$K_k = P_k(-)H_k^T[H_kP_k(-)H_k^T + R_k]^{-1} \tag{7}$$

알고리즘 유도의 편의상 우선 칼만필터 전개식에서 상태변수에 관한 전개식만을 고려하기로 한다. 식 (3)과 식 (5)를 이용하여 $\underline{\hat{x}}_{h+i+1}(+)$를 $\underline{\hat{x}}_h(+)$의 함수로 표시하면 식 (8)과 같이 표현된다.

$$\underline{\hat{x}}_{h+i+1}(+) = S_{i+1}\underline{\hat{x}}_h(+) + T_{i+1} \tag{8}$$

식 (8)에서 S_{i+1}과 T_{i+1}은 다음과 같이 구해진다.

$$S_0 = I, T_0 = \underline{0}$$

for $i = 0, 1, 2, \cdots$

$$S_{i+1} = [\Phi_{h+i} - K_{h+i+1}H_{h+i+1}\Phi_{h+i}]S_i \tag{9}$$

$$T_{i+1} = [\Phi_{h+i} - K_{h+i+1}H_{h+i+1}\Phi_{h+i}]T_i + K_{h+i+1}\underline{z}_{h+i+1} \tag{10}$$

이상으로부터 $\underline{\hat{x}}_{h+i+1}(+)$는 S_{i+1}, T_{i+1}을 먼저 계산하고, 식 (8)에 의하여 $\underline{\hat{x}}_h(+)$로부터 직접 구할 수 있음을 알 수 있다.

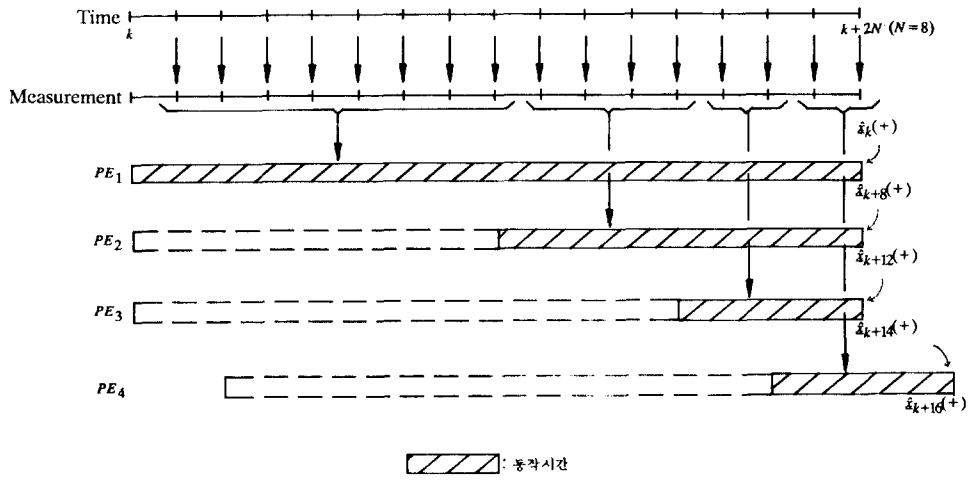


그림 1 병렬 칼만필터 시나리오
Fig. 1 Parallel Kalman Filter Scenario

식 (8)에서부터 식 (10)까지의 관계로부터 새로운 병렬 칼만필터 알고리즘을 유도할 수 있다. 그림 1은 병렬 칼만필터의 시나리오를 나타내고 있는데, 먼저 그림을 사용하여 제안하는 알고리즘을 예를 들어 설명하고, 다음에 일반적으로 정리한다. 본 논문에서 제안하는 병렬 알고리즘은 일정 시간 구간을 한 계산단계로 하여, 칼만필터를 실시간으로 운영한다. 그림 1에서 첫번째 선은 시간축을 나타내며, $2N$은 병렬 알고리즘이 한 단계 실행될 때 소요되는 시간을, k는 그 단계의 시작 시간을 나타낸다. $2N$의 시간에 측정치는 $2N$개 입력된다고 가정하고, 각 프로세서들의 계산능력은 하나의 측정치를 2의 단위시간에 처리할 수 있다고 가정한다. 그러므로 일반적인 순차적 칼만필터를 사용하면, $2N$의 시간에 N개의 측정치만을 처리할 수 있다. 그러나 본 논문은 $2N$의 시간에 $2N$개의 측정치를 처리할 수 있는 새로운 병렬 알고리즘을 제안한다. 설명을 쉽게 하기 위하여 $N=8$의 경우에 대하여 병렬 칼만필터 알고리즘을 전개해 본다.

PE_1(processing element 1)은 시간 k부터 $k+8$까지에 입력되는 8개의 측정치를 시간 k부터 $k+16$까지의 시간에 처리한다. 식 (9)과 식 (10)에서 $h=k$로 하여, S_8, T_8을 계산하고, 식 (8)의 관계에 의하여 $\hat{x}_k(+)$로부터 $\hat{x}_{k+8}(+)$를 계산한다. 그 다음 $\hat{x}_{k+8}(+)$를 PE_2에 전송한다. PE_2는 9번째 측정치부터 12번째 측정치까지 4개의 측정치를 시간 $k+8$부터 $k+16$까지의 시간에 처리한다. $h=k+8$로 하여, S_4, T_4를 계산하고, PE_1에서 받은 $\hat{x}_{k+8}(+)$로부터 $\hat{x}_{(k+8)+4}(+)$를 계산한다. 그 다음 $\hat{x}_{k+12}(+)$를 PE_3에 전송한다. 비슷한 과정에 의하여 PE_3는 PE_2에서 받은 $\hat{x}_{k+12}(+)$로 부터 $\hat{x}_{k+14}(+)$를 계산하여 PE_4에 전송하며, PE_4는 이것으로부터 $\hat{x}_{k+16}(+)$를 계산하며, 이 값을 다음 계산단계에서 사용하기 위하여 PE_1으로 전송한다.

이상과 같이 예를 들어 설명한 병렬 칼만필터 알고리즘을 수행하기 위하여 필요한 프로세서 갯수를 p라고 하면, $p=(\log_2 N)+1$이다. 제안한 병렬 칼만필터 알고리즘을 일반적으로 정리하면 다음과 같다.

$PE_i (i=1,\cdots,(\log_2 N)+1)$:

계산시작시간 : $h=k+2\sum_{j=1}^{i-1}\frac{N}{2^j}$

given $P_{h+1}(-)$

$S_0=I, T_0=\underline{0}$

$l=N-\sum_{j=1}^{i-1}\frac{N}{2^j}$ (단, $i=(\log_2 N)+1$이면, $l=2$)

$K_{h+1}=P_{h+1}(-)H_{h+1}^T[H_{h+1}P_{h+1}(-)H_{h+1}^T + R_{h+1}]^{-1}$

$P_{h+1}(+)=[I-K_{h+1}H_{h+1}]P_{h+1}(-)$

$S_1=[\Phi_h-K_{h+1}H_{h+1}\Phi_h]S_0$

$T_1=[\Phi_h-K_{h+1}H_{h+1}\Phi_h]T_0+K_{h+1}\underline{z}_{h+1}$

for $j=2,\cdots,l$

$P_{h+j}(-)=\Phi_{h+j-1}P_{h+j-1}(+)\Phi_{h+j-1}^T+Q_{h+j-1}$

$$K_{h+j}=P_{h+j}(-)H_{h+j}^T[H_{h+j}P_{h+j}(-)\,H_{h+j}^T + R_{h+j}]^{-1}$$
$$P_{h+j}(+)=[I-K_{h+j}H_{h+j}]P_{h+j}(-)$$
$$S_j=[\Phi_{h+j-1}-K_{h+j}H_{h+j}\Phi_{h+j-1}]S_{j-1}$$
$$T_j=[\Phi_{h+j-1}-K_{h+j}H_{h+j}\Phi_{h+j-1}]\,T_{j-1}+K_{h+j}\underline{z}_{h+j}$$

$(i-1)$번째 프로세서에서 $\hat{x}_h(+)$를 전송받는다.

$\hat{x}_{h+i}(+)=S_i\hat{x}_h(+)+T_i$

$\hat{x}_{h+i}(+)$를 $(i+1)$번째 프로세서로 전송한다.

단, 첫번째 프로세서는 상태변수의 초기치를 전 계산단계에서 마지막 프로세서로부터 전송받으며, 마지막 프로세서는 계산된 상태변수를 다음 계산단계에서 사용하기 위하여 첫번째 프로세서로 전송한다.

3. 공분산행렬의 병렬 알고리즘

2절에서 실시간 운영의 칼만필터에 대한 병렬처리 알고리즘을 유도하였다. 이 알고리즘에서 보면, 각 프로세서가 S_j, T_j를 계산할 때 칼만이득 K_{h+j}를 필요로 하며, 이것은 공분산행렬 P_{h+j}로부터 계산된다. 따라서 P_{h+j}의 전달을 위하여 각 프로세서는 알고리즘의 매 계산단계 시작전에 각자의 공분산행렬의 초기치를 가지고 있어야 함을 알 수 있다. 즉, 그림 1의 경우 PE_1은 $P_{k+1}(-)$를, PE_2는 $P_{k+9}(-)$를, PE_3는 $P_{k+13}(-)$를, PE_4는 $P_{k+15}(-)$를 초기값으로 계산시작 전에 가지고 있어야 한다. 이러한 문제점을 해결하기 위하여, 매 계산단계에서 필요로 하는 공분산행렬의 초기치들을 한 계산단계 전에 미리 구하는 방법을 제안한다. 즉, 전 계산단계에서 병렬 칼만필터 알고리즘을 수행하는 프로세서들과는 별도로 다른 프로세서들에 의하여 다음 계산단계에서 사용할 공분산행렬의 초기치들을 구한다.

일반적인 이산 칼만필터에서 공분산행렬의 전개식은 이산 리카티방정식으로 나타나며 역행렬이 포함된 복잡한 비선형방정식이다. 따라서 계산량이 매우 많은데, 한 계산단계 시간내에 계산을 끝마치기 위해서는 병렬계산기법을 도입해야 한다. 그런데 이산 리카티방정식은 비선형방정식이므로 이 식에 직접 병렬기법을 도입하기에는 문제점이 있다. 따라서 이산 리카티방정식을 변형시켜, 병렬기법을 적용하기에 알맞게 선형화된 형태로 만들어야 한다. 식 (4), (6), (7)로부터 식 (11)이 유도된다[11].

$$\begin{aligned}P_{k+1}(-)&=\Phi_k[P_k(-)-P_k(-)H_k^T(H_kP_k(-)H_k^T+R_k)^{-1}H_kP_k(-)]\Phi_k^T+Q_k\\&=[Q_k\Phi_k^{-T}\pm(\Phi_k+Q_k\Phi_k^{-T}H_k^TR_k^{-1}H_k)P_k(-)]\cdot[\Phi_k^{-T}+\Phi_k^{-T}H_k^TR_k^{-1}H_kP_x(-)]^{-1}\\&\equiv[C_k+D_kP_k(-)][A_k+B_kP_k(-)]^{-1}\end{aligned}\quad(11)$$

식 (11)에서의 A_k, B_k, C_k, D_k는 모두 $n\times n$행렬이다. 이를 이용하여 식 (12)와 같이 표현되는 선형 시스템을 가정하자.

$$\begin{bmatrix}X_{k+1}\\Y_{k+1}\end{bmatrix}=\begin{bmatrix}A_k & B_k\\C_k & D_k\end{bmatrix}\begin{bmatrix}X_k\\Y_k\end{bmatrix}\quad(12)$$

식 (12)에서 X_k, Y_k는 각각 $n\times n$행렬로서 이것으로부터 식 (13)과 같은 관계식을 얻을 수 있다.

$$\begin{aligned}Y_{k+1}X_{k+1}^{-1}&=[C_kX_k+D_kY_k][A_kX_k+B_kY_k]^{-1}\\&=[C_kX_k+D_kY_k]X_k^{-1}\cdot[A_k+B_kY_kX_k^{-1}]^{-1}\\&=[C_k+D_kY_kX_k^{-1}][A_k+B_kY_kX_k^{-1}]^{-1}\end{aligned}\quad(13)$$

식 (11)과 식 (13)으로부터 $Y_kX_k^{-1}=P_k(-)$를 가정하면, $Y_{k+1}X_{k+1}^{-1}=P_{k+1}(-)$의 관계가 성립함을 알 수 있다. 따라서 식 (11)로 표현되는 이산 비선형방정식을 식 (12)와 같은 선형방정식의 전개로 변형시킬 수 있으며, X_k, Y_k로부터 $P_k(-)$를 계산할 수 있음을 알 수 있다. 실제로 $X_0=I, Y_0=P_0(-)$로 하면 $P_0(-)=Y_0X_0^{-1}$이며, 이 X_0, Y_0로부터 식 (12)를 이용하여 선형차분방정식으로 X_k, Y_k를 순환적으로 구할 수 있으며, 이로부터 $P_k(-)$를 구할 수 있다.

식 (12)의 선형차분방정식을 사용하여 공분산행렬의 병렬 알고리즘을 유도할 수 있다. 식 (14)와 같이 M_k를 정의하고 X_{h+i}, Y_{h+i}를 X_h, Y_h의 함수로 표현하면 식 (15)와 같이 된다.

$$M_k\equiv\begin{bmatrix}A_k & B_k\\C_k & D_k\end{bmatrix}\quad(14)$$

$$\begin{bmatrix}X_{h+i}\\Y_{h+i}\end{bmatrix}=\Psi_i\begin{bmatrix}X_h\\Y_h\end{bmatrix}\quad(15)$$

식 (15)에서 Ψ_i는 다음과 같이 계산된다.

$$\Psi_0=I$$
$$\text{for } i=1,2,3,\cdots$$
$$\Psi_i=M_{h+i-1}\Psi_{i-1}\quad(16)$$

공분산행렬의 병렬 알고리즘에서 사용되는 프로세서의 갯수는, 공분산행렬의 병렬 알고리즘의 계산량과 앞에서 유도한 병렬 칼만필터 알고리즘의 계

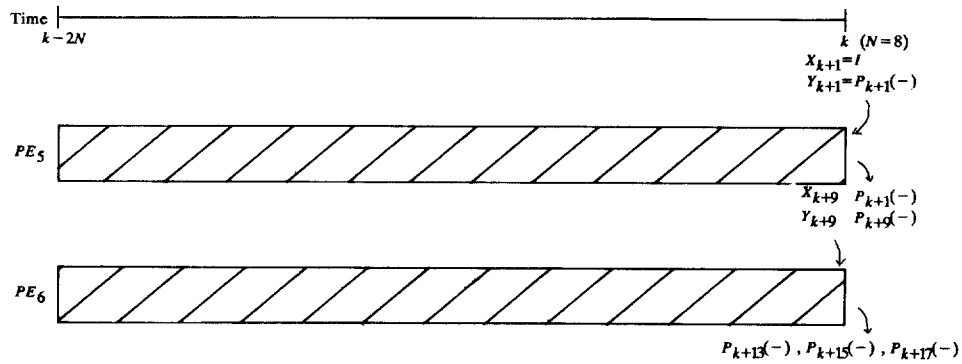


그림 2 공분산행렬의 병렬 알고리즘
Fig. 2 Parallel Algorithm for Covariance Equation

산량에 의하여 결정된다. 자세한 내용은 공분산행렬의 알고리즘을 유도한 다음에 설명하고, 여기서는 편의상 프로세서는 2개라고 가정한다. 그림 2는 공분산행렬의 병렬 알고리즘을 보여준다. 2절에서 유도한 병렬 칼만필터 알고리즘은 매 계산단계 $2N$의 시간에 $2N$개의 측정치를 처리하므로, 공분산행렬의 병렬 알고리즘은 $2N$시간 내에 $2N$개의 M_{k+i}행렬을 처리하여, 다음 계산단계에서 필요한 공분산행렬의 초기치들을 구해야 한다. 2절에서와 같이 $N=8$이라고 가정하고 알고리즘을 설명한다. 그림 1에서 $N=8$일때, 필요한 공분산행렬의 초기치들은 $P_{k+1}(-)$, $P_{k+9}(-)$,$P_{k+13}(-)$, $P_{k+15}(-)$로 모두 4개이다. 그러나 공분산행렬의 병렬 알고리즘에서는 $P_{k+1}(-)$가 전 계산단계에서 얻어지고, 대신에 다음 계산단계에서 필요한 $P_{k+17}(-)$를 계산한다.

PE_5는 $h=k+1$로 하여, 식 (16)에 의하여 Ψ_8를 계산하고, 이것과 $X_{k+1}=I, Y_{k+1}=P_{k+1}(-)$로부터 X_{k+9}, Y_{k+9}를 계산하여, 이 값들을 PE_6으로 전송한다. 그리고 X_{k+9}, Y_{k+9}로부터 $P_{k+9}(-)$를 구하고, $P_{k+1}(-)$를 PE_1로 $P_{k+9}(-)$를 PE_2로 전송한다. PE_6은 $h=k+9$로 하여, 식 (16)에 의하여 Ψ_4, Ψ_6, Ψ_8을 계산하고, 이들과 X_{k+9}, Y_{k+9}로부터 X_{k+13}, $Y_{k+13}, X_{k+15}, Y_{k+15}, X_{k+17}, Y_{k+17}$을 계산하고, 이들로부터 $P_{k+13}(-)$, $P_{k+15}(-)$, $P_{k+17}(-)$를 구한 뒤에 이 값을 각각 PE_3, PE_4, PE_5로 전송한다.

프로세서 갯수를 $q, X_{k+1}=I, Y_{k+1}=P_{k+1}(-)$라 하고 위에서 설명한 공분산행렬의 병렬 알고리즘을 정리하면 다음과 같다.

$PE_i(i=1,\cdots,q)$:
계산시작시간 : $k-2N$
$\Psi_o=I$, $l=k+2N(i-1)/q+1$
for $j=1,\cdots,2N/q$
$\Psi_j=M_{l+j-1}\Psi_{j-1}$
$(i-1)$번째 프로세서에서 X_l, Y_l을 전송받는다. (단, $i=1$인 경우는 제외)

$$\begin{bmatrix} X_{l+2N/q} \\ Y_{l+2N/q} \end{bmatrix} = \Psi_{2N/q} \begin{bmatrix} X_l \\ Y_l \end{bmatrix}$$

$X_{l+2N/q}, Y_{l+2N/q}$을 $(i+1)$번째 프로세서로 전송한다. (단, $i=q$인 경우는 제외)

병렬 칼만필터 알고리즘의 다음 계산단계에서 필요로 하는 공분산행렬의 초기치(예를들어 P_{l+j})를 계산해야 하는 프로세서는 다음의 계산과정이 더 첨가된다.

$$\begin{bmatrix} X_{l+j} \\ Y_{l+j} \end{bmatrix} = \Psi_j \begin{bmatrix} X_l \\ Y_l \end{bmatrix}$$

$P_{l+j}(-) = Y_{l+j}X_{l+j}^{-1}$
$P_{l+j}(-)$를 이 값을 필요로 하는 프로세서로 전송한다.

지금까지 설명한 공분산행렬의 병렬 알고리즘을 첨가하여 구성한 병렬 칼만필터 알고리즘의 전체적인 구조는 그림 3과 같다. 그림에서는 알고리즘의 계산단계 2개를 나타내고 있다. $k-2N$시간부터 k시간까지의 계산단계 Ⅰ에서 병렬 칼만필터가 2절에서 설명한 알고리즘을 수행하는 동안, 아래쪽의 프로세서들은 본 절에서 설명한 공분산행렬의 병렬 알고리즘을 수행하여 계산단계 Ⅱ에서 병

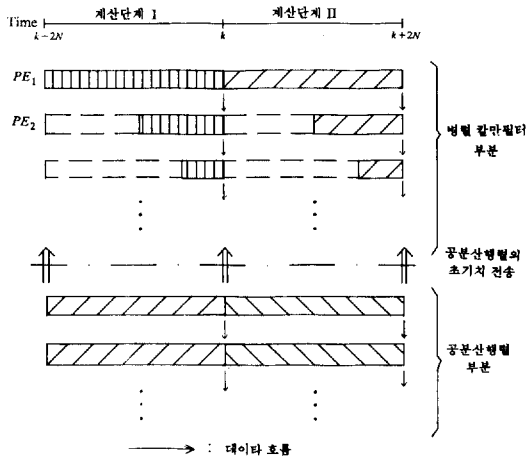


그림 3 병렬 칼만필터
Fig. 3 Parallel Kalman Filter

렬 칼만필터가 필요로 하는 공분산행렬 값들을 계산하고, k시간 전에 이들을 병렬 칼만필터로 전송한다.

본 절에서 설명한 공분산행렬의 병렬 알고리즘에서 필요한 프로세서의 갯수를 구하기 위하여, 지금까지 설명한 병렬 알고리즘의 계산량을 구한다. 알고리즘의 계산량은 뺄셈은 덧셈으로, 나눗셈은 곱셈으로 간주하고 덧셈과 곱셈의 수행횟수로 나타내본다. 그림 3에서 보면, 본 논문에서 제안된 병렬 칼만필터 알고리즘은 PE_1의 한 단계의 계산시간이 전체 계산시간을 결정하고, 결과적으로 이 알고리즘의 효율을 결정하고 있음을 볼 수 있다. 따라서 각 계산단계에서 공분산행렬의 초기값을 결정하기 위한 프로세서들은 PE_1의 한 단계의 계산시간 내에 그들의 계산을 끝마쳐야 알고리즘의 다음 계산단계를 계속 수행할 수 있다. 그러므로 공분산행렬의 초기값을 결정하기 위한 프로세서들의 갯수는 이들이 수행하는 전체 계산량을 PE_1이 수행하는 계산량으로 나눔으로써 결정된다. 상태변수 $\underline{x}_k$의 차수를 n, 측정변수 $\underline{z}_k$의 차수를 m, 그리고 공분산행렬은 대칭이라고 가정하고, PE_1이 한 계산단계에서 수행하는 계산량을 구하면 덧셈과 곱셈이 각각 다음과 같다.

$$\text{덧셈} : N\{4n^3-\frac{1}{2}n^2-\frac{3}{2}n+\frac{3}{2}n^2m+\frac{3}{2}nm^2+m^3-2m^2+m\}-2n^3+n^2$$

$$\text{곱셈} : N\{4n^3+2n^2+\frac{3}{2}n^2m+2nm+\frac{3}{2}nm^2+m^3\}-2n^3$$

공분산행렬의 병렬 알고리즘에서 전체 프로세서가 한 계산단계에서 수행하는 총계산량은 덧셈과 곱셈이 각각 다음과 같다.

$$\text{덧셈} : 2N\{8n^3-4n^2\}+\{(\log_2 N)+1\}\{6n^3-5n^2+n\}$$

$$\text{곱셈} : 16Nn^3+6n^3\{(\log_2 N)+1\}$$

일반적으로 컴퓨터가 덧셈보다 곱셈에 더 많은 시간이 소비되므로 곱셈량을 가지고 공분산행렬의 병렬 알고리즘에 필요한 프로세서 갯수를 결정한다. 공분산행렬의 병렬알고리즘에서 수행되는 곱셈의 총계산량을 PE_1이 수행하는 곱셈의 계산량으로 나눈 값을 s라 하면 s는 식 (17)과 같이 주어진다.

$$\frac{16Nn^3+6n^3\{(\log_2 N)+1\}}{N\{4n^3+2n^2+\frac{3}{2}n^2m+2nm+\frac{3}{2}nm^2+m^3\}-2n^3} \tag{17}$$

그리고 공분산행렬의 병렬 알고리즘에 필요한 프로세서 갯수를 q라 하면 q는 s보다 작지 않은 최소의 정수이다.

4. 시뮬레이션

본 절에서는 2절과 3절에서 유도한 병렬 칼만필터 알고리즘을 시뮬레이션 한다. 본 논문이 제안한 병렬 칼만필터 알고리즘은 일반적인 칼만필터보다 2배의 측정치를 처리하기 때문에, 일반적인 칼만필터보다 상태변수의 추정치가 더욱 빠르게 수렴한다. 시뮬레이션을 통하여 이 점을 검토해 본다. 또한 제안된 병렬 알고리즘은 일반적인 칼만필터와 다른 계산방법을 사용한 관계로 수치적인 오차가 다를 수 있는데, 이런 점도 검토해 본다.

먼저 시뮬레이션을 하기 위한 대상 시스템을 설명한다. 여기서 사용한 대상 시스템은 Candy의 저서[12] 예제 6.2.1에 있는 시스템이다.

$$\underline{x}_k=\begin{bmatrix}0.97 & 0\\ 0 & 0.9\end{bmatrix}\underline{x}_{k-1}+\begin{bmatrix}1\\ 0\end{bmatrix}u_{k-1}+\underline{w}_{k-1}$$

$$z_k=[2 \quad 1.5]\underline{x}_k+\nu_k$$

여기에서 $\underline{w}_k$와 ν_k는 백색잡음으로 다음의 성질을 가진다.

$$E\{\underline{w}_k\}=\underline{0}, E\{\nu_k\}=0$$

$$E\{\underline{w}_k\underline{w}_j^T\}=\begin{bmatrix}1\times10^{-4} & 0\\ 0 & 4\times10^{-3}\end{bmatrix}\delta_{kj}$$

$$E\{\nu_k\nu_j\}=1.0\delta_{kj}$$

그리고 시스템 상태변수의 초기치, 필터 상태변수의 초기치, 그리고 오차의 공분산행렬의 초기치는 다음과 같이 가정한다.

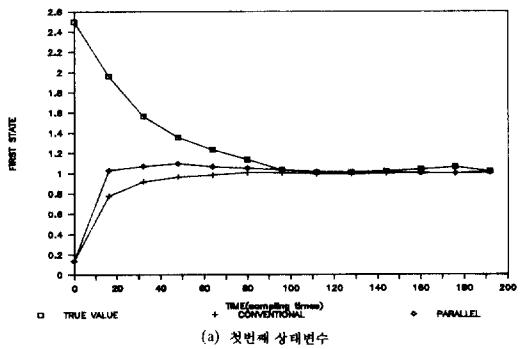
$$\underline{x}_0^T(-)=[2.5\quad 2],\ \underline{\hat{x}}_0^T(-)=[0\quad 0]$$

$$P_0(-)=\begin{bmatrix}10^{-2} & 0\\ 0 & 10^{-2}\end{bmatrix}$$

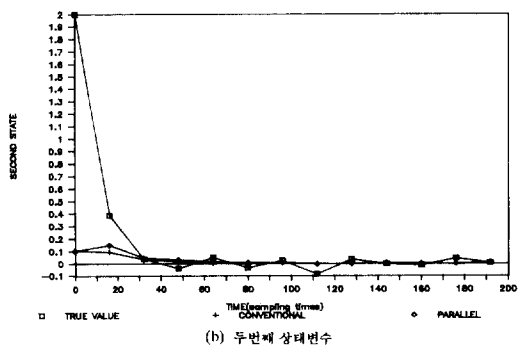
입력은 $u_k=0.03$으로 하고, 샘플링 시간은 0.1로 가정한다. 2절에서 유도한 병렬 칼만필터 알고리즘에는 입력에 대한 식은 포함되지 않았다. 그러나 이것은 칼만필터의 식 (3)에 쉽게 첨가할 수 있고, 병렬 칼만필터 알고리즘에 이것에 대한 항을 첨가시키면 된다. 시뮬레이션 프로그램은 이것을 첨가하여 작성되었다.

일반적인 칼만필터는 측정치 한 개를 처리하는데 0.2가 걸린다고 가정한다. 또 병렬 칼만필터 알고리즘의 한 계산단계 시간은 16샘플링 시간으로, 즉 1.6이라고 가정한다. 따라서 일반적인 칼만필터는 한 계산단계 시간, 1.6동안에 8개의 측정치를 처리할 수 있다. 그러나 본 논문이 제안한 병렬 칼만필터는 한 계산단계 시간동안 일반적인 칼만필터가 처리하는 측정치의 2배인 16개의 측정치를 처리할 수 있다. 병렬 칼만필터가 한 계산단계 시간동안 처리하는 측정치가 16개이므로, 이 병렬 알고리즘을 수행하는 프로세서 갯수는 공분산행렬의 병렬 알고리즘을 수행하는 프로세서 갯수를 제외하면 4개이다. 앞에서 설명한 예제로부터 상태변수의 차수 n은 2이고, 측정치의 차수 m은 1이다. 그리고 가정에서 $N=8$이다. n,m,N들을 3절의 식 (17)에 대입하면 $s\approx2.92$이므로, 공분산행렬의 병렬 알고리즘에서 필요한 프로세서 갯수는 $q=3$이 된다.

그림 4는 시뮬레이션 결과를 16샘플링 시간마다 그린 것이다. (a)는 첫번째 상태변수에 대하여, (b)는 두번째 상태변수에 대하여 보여준다. 네모 표시는 시스템의 상태변수의 참값을 나타내며, + 표시는 일반적인 칼만필터로 추정한 상태변수의 추정치를 나타내며, 마름모 표시는 병렬 칼만필터로 추정한 상태변수의 추정치를 나타낸다. 병렬 칼만필터는 일반적인 칼만필터보다 2배의 측정치

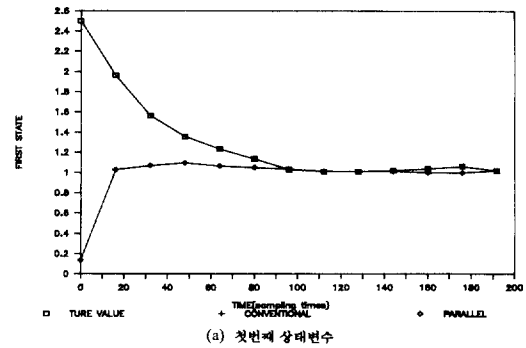


(a) 첫번째 상태변수

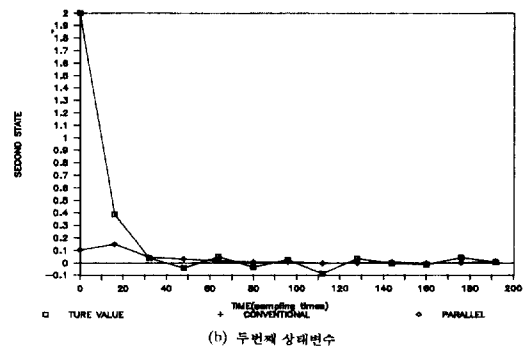


(b) 두번째 상태변수

그림 4 상태변수의 추정치
Fig. 4 Estimates of State



(a) 첫번째 상태변수



(b) 두번째 상태변수

그림 5 상태변수의 추정치
Fig. 5 Estimates of State

를 처리하므로 더 빠르게 수렴하는데, 그림에서 보면, 병렬 칼만필터의 추정치가 일반적인 칼만필터보다 더 빠르게 참값에 접근하는 것을 볼 수 있다.

병렬 칼만필터 알고리즘의 수치적인 오차를 일반적인 칼만필터와 비교하기 위하여, 일반적인 칼만필터도 한 계산단계 시간 내에 병렬 칼만필터가 처리하는 측정치의 갯수만큼 처리할 수 있다고, 즉 한 계산단계 시간 내에 16개의 측정치를 처리할 수 있다고 가정하고 시뮬레이션하였는데, 그림 5는 이 결과를 나타낸다. (a)는 첫번째 상태변수에 대하여, (b)는 두번째 상태변수에 대하여 보여준다. 일반적인 칼만필터(+표시)와 병렬 칼만필터(마름모 표시)의 상태변수 추정치가 일치함을 볼 수 있는데, 이로부터 본 논문이 제안한 병렬 칼만필터의 수치적인 안정도는 일반적인 칼만필터와 같음을 알 수 있다.

5. 결 론

칼만필터는 잡음이 섞인 측정값에서 오차의 분산이 최소가 되게 상태변수를 추정하는 최적 필터이나, 복잡한 행렬 계산식으로 인해 계산량이 매우 많다. 따라서 실시간 적용을 위하여 많은 계산량을 효과적으로 처리할 수 있는 병렬성의 도입이 요구되었다. 칼만필터를 병렬처리한 기존 알고리즘은 주로 분산처리에 의한 계층적 구조의 형태이거나, 행렬계산에 시스톨릭 구조를 사용하여 계산속도를 향상시키는 형태이다. 그러나 본 논문이 제안한 병렬 칼만필터는 Kogge와 Stone[3]이 제안한 순환방정식에 대한 병렬처리 알고리즘의 기본개념을 사용한 새로운 형태의 알고리즘이다. 본 논문이 제안한 병렬 칼만필터는 일정시간 구간을 한 계산단계로 하여, 칼만필터를 실시간으로 운영한다. 그리고 일반적인 칼만필터보다 두배의 빠르기로 측정치를 처리하기 때문에 상태변수 추정치가 참값에 더욱 빨리 수렴한다.

병렬 칼만필터 알고리즘을 수행하기 위하여 각 프로세서는 매 계산단계 시작 전에 각각 자신의 공분산행렬의 초기치를 가지고 있어야 하는데, 이를 위하여 바로 앞의 계산단계에서 공분산행렬의 초기치를 결정하기 위한 공분산행렬의 병렬 알고리즘이 제안되었다. 공분산행렬의 순환방정식은 리카티 방정식이라는 비선형방정식으로 병렬화가 어렵기 때문에 우선 이 방정식을 변형시켜 선형화된 식을 유도하고 선형화된 식을 사용하여 병렬 알고리즘이 구성되었다.

본 논문이 제안한 병렬 칼만필터 알고리즘은 실시간 운영이지만, 실제로 실시간으로 사용될 수 있는 상태변수는 각 계산단계의 마지막 상태변수뿐이다. 이러한 이유로 본 논문이 제안한 병렬 칼만필터는 매 샘플링 시간마다 계속적으로 상태변수 추정치를 구해야 하는 문제에는 적용 불가능하고, 일정한 시간간격으로 정확한 상태변수 추정치를 필요로 하는 문제에 적합하다.

만일 대상시스템이 시불변이거나 또는 시스템 잡음이 없는 경우에는, 공분산행렬의 병렬 알고리즘을 병렬 칼만필터 알고리즘을 수행하는 프로세서들의 비동작시간에 수행할 수 있기 때문에 프로세서 사용효율을 증대시킬 수 있다. 또한 본 논문이 제안한 병렬 칼만필터 알고리즘의 기본개념은 순환적 형태의 선형방정식에 확장 적용될 수도 있다.

참 고 문 헌

[1] A. Gelb, *Applied Optimal Estimation*, M.I.T. Press, 1974.

[2] Mohamed F. Hassan, G. Salut, Madan G. Singh, and Andre Titli, "A Decentralized Computational Algorithm for the Global Kalman Filter," *IEEE Trans. Automat. Contr.*, Vol. AC-23, No. 2, pp. 262~268, 1978.

[3] U.B. Desai and B. Das, "Parallel Algorithms for Kalman Filtering," in *Proc. 1985 Amer. Contr. Conf.*, Boston, MA, pp. 920~921, 1985.

[4] Hamid R. Hashemipour, Sumit Roy, and Alan J. Laub, "Decentralized Structures for Parallel Kalman Filtering," *IEEE Trans. Automat. Contr.*, Vol. 33, No. 1, pp. 88~94, 1988.

[5] A. Andrews, "Parallel Precessing of the Kalman Filter," in *Proc. Int. Conf. Parallel Processing*, Columbus, Ohio, pp. 216~220, 1981.

[6] J.M. Jover and T. Kailath, "A Parallel Architecture for Kalman Filter Measurement Update and Parameter Estimation," *Automatica*,, Vol. 22, No. 1, pp. 43~57, 1986.

[7] Tze-Yun Sung and Yu-Hen Hu, "Parallel VLSI Implementation of the Kalman Filter," *IEEE Trans. Aero. Elec. Syst.*, vol. AES-23,

No. 2, pp. 215~224, 1987.

[8] H.T. Kung, "Why Systolic Architecture?," *IEEE Computer*, pp. 37~46, Jan., 1982.

[9] C.Mead and L.Conway, *Introduction to VLSI Systems*, Addison Wesley, 1980.

[10] Peter M. Kogge and Harold S. Stone. "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations, *IEEE Trans. Computers*, Vol. C-22, No. 8, pp. 786~793, 1973.

[11] Brian D.O. Anderson and John B. Moore, *Optimal Filtering*, Prentice-Hall, 1979.

[12] James V. Candy, *Signal Processing (the Model-based Approach)*, McGraw Hill, 1986.