Multiple-Row Downdating을 수행하는 고속 최소자승 알고리즘

이 충 한† 김 석 일††

요 약

다행관측행렬을 복원하는 기존의 알고리즘은 단일행의 복원방법인 Cholesky Factor Downdating(CFD)을 이용하여 행렬 Z^T의 각 행을 순차적으로 복원하는 방법으로 필요한 실수연산의 횟수는 $\frac{5}{2}pn^2$이다. 이에 비하여 본 논문에서 제안한 HCFD(Hybrid Cholesky Factor Downdating)기법은 $p \geq n$인 크기의 다행관측행렬 Z^T를 복원하는 데 필요한 실수연산의 횟수가 $pn^2+\frac{5}{6}n^3$이다. HCFD 기법은 Z^T로부터 $Z^T=Q_Z R_Z^T$인 상삼각행렬 R_Z^T을 구하고, R_Z^T에 대해 CFD 알고리즘을 적용함으로 필요한 시간복잡도를 크게 줄일 수 있다. 또한, HCFD 기법과 기존의 CFD 기법을 Sun SPARC/2와 국산주전산기 I에서 실험한 결과, HCFD 기법이 CFD 기법에 비하여 성능이 우수함을 보여 주었으며, 특히 복원하려는 행이 많을 경우에 HCFD 기법이 CFD 기법에 비하여 성능이 크게 향상됨을 알 수 있었다.

A Fast Least-Squares Algorithm for Multiple-Row Downdatings

Chung Han Lee† and Sukil Kim††

ABSTRACT

Existing multiple-row downdating algorithms have adopted a CFD(Cholesky Factor Downdating) that recursively downdates one row at a time. The CFD based algorithm requires $\frac{3}{2}pn^2$ flops(floating point operations) downdating a $p \times n$ observation matrix Z^T. On the other hands, a HCFD(Hybrid CFD) based algorithm we propose in this paper, requires $pn^2+\frac{5}{6}n^3$ flops when $p \geq n$. Such a HCFD based algorithm factorizes Z^T at first, such that $Z^T=Q_Z R_Z^T$, and then applies the CFD onto the upper triangular matrix R_Z^T, so that the total number of floating point operations for downdating Z^T would be significantly reduced compared with that of the CFD based algorithm. Benchmark tests on the Sun SPARC/2 and the Tolerant System also show that the performance of the HCFD based algorithm is superior to that of the CFD based algorithm, especially when the number of rows of the observation matrix is large.

1. 서 론

최소자승해법은 adaptive antenna, beam-forming, 정보통신, space navigation, 음성인식 및 잡음감쇠 등과 같은 신호처리분야에서 선형예측의 방법으로 광범위하게 사용된다[1]. 그중에서도 관측치가 주기적으로 입력되는 선형시스템 $X\omega=s$에서 관측치가 행렬 X에 첨가되거나 X로 부터 일부 행이 제거된 경우에

$$\min \| s-X\omega \|_2 \qquad (1)$$

를 만족하는 최소자승해법은 신호처리분야에서 이용도가 높은 알고리즘의 하나이다. 여기서 관측치가 첨가된 경우와 제거된 경우의 최소자승해법을 각각 least-squares updating과 least-squares downdating이라고 한다.

센서로부터 연속적으로 관측치가 입력되고, 이

· 이 논문은 1994년도 학술진흥재단의 자유공모과제로 수행되었음.
† 준 회 원: 충북대학교 전자계산학과 석사과정
†† 정 회 원: 충북대학교 컴퓨터과학과 조교수
논문접수: 1994년 8월 30일, 심사완료: 1994년 12월 19일

를 토대로 기존의 정보와 함께 최소자승해를 계산하는 선형시스템을 구성함에 있어서 빠른 least-squares updating 또는 downdating 알고리즘의 구현은 시스템의 성능을 향상시킬 수 있다. 즉, 이미 입력된 관측치를 토대로 최소자승해를 구하는 동안 새로운 관측치가 입력 큐에 저장되고, 계산이 종료되면 그동안 입력 큐에 저장된 관측치를 이용하여 새로운 최소자승해를 반복해서 수행하는 시스템에서 빠른 최소자승해법 알고리즘은 시스템의 계산주기를 단축시킬 수 있으며, 빠른 선형 예측이 가능하므로 시스템의 성능을 향상시키는 데 매우 중요한 관건이 된다. 이러한 이유로 우수한 성능의 최소자승해법에 대한 연구가 활발히 진행되고있다.

식(1)의 최소자승해를 계산하기 위해서는 정규방정식(normal equation)

$$X^TX\omega = X^Ts \qquad (2)$$

의 $n\times n$ symmetric positive definite 시스템의 해를 계산하므로 가능하다. 여기서 X, s 및 ω는 각각 $m\times n$ 행렬, $m\times 1$ 및 $n\times 1$ 백터이다. 즉, 식(2)에서 Cholesky 알고리즘을 이용하여 symmetric positive definite 행렬 X^TX의 Cholesky factor인 상삼각행렬 R을 계산하고, 이어서 삼각선형대수식의 해를 구하는 알고리즘[2, 3]으로부터 ω를 계산할 수 있다. 또한 상삼각행렬 R을 계산할 때, 식(3)과 같이 직교변환(orthogonal factorization)을 이용하여 정밀한 해를 구할 수 있다.

$$QX = \begin{pmatrix} R \\ 0 \end{pmatrix},\ Q^TQ = I \qquad (3)$$

여기서 삼각선형대수식의 해를 구하는 데 필요한 실수연산의 복잡도는 $O(n^2)$이고, 식(2) 또는 식(3)에 의하여 상삼각행렬 R을 계산하는 데 필요한 실수연산의 복잡도가 각각 $O(n^3)$ 또는 $O(mn^2)$이므로 주어진 선형대수식으로부터 식(1)을 만족하는 해를 계산하는 데 필요한 실수연산의 복잡도는 R을 계산하는데 필요한 복잡도에 의하여 결정된다.

행렬 X에 새로운 관측치 z^T가 추가되거나 제거된 경우에도 변환된 선형대수식에 대한 최소자승해

$$\min \| \tilde{s} - \tilde{X}\tilde{\omega} \|_2 \qquad (4)$$

를 만족하는 새로운 해 $\tilde{\omega}$를 구할 수 있다. 여기서 $\tilde{X}$와 $\tilde{s}$는 각각 새로운 관측치의 입력에 의하여 행렬 X와 벡터 s로부터 변화된 결과이다. 새로운 해 $\tilde{\omega}$를 계산함에 있어 $\tilde{X}$로 부터 Cholesky factor $\tilde{R}$을 구하지 않고, 관측치가 입력되기 이전에 계산된 Cholesky factor R로 부터 $\tilde{R}$을 계산할 수 있으며, 이러한 방법에 의하여 실수연산의 복잡도가 $O(n^2)$인 알고리즘의 구현이 가능하며, 알고리즘의 안정성에 관한 입증은 Gill [4]에 의하여 연구된 바 있다.

신호처리의 관점에서 보면, 최소자승에 의한 적응 필터문제는

$$\sum_{j=0}^{p-1}(t^T(j)\omega(n) - s_j)^2$$

를 최소로 하는 $\omega\equiv\omega(n)$를 계산하는 것으로 요약될 수 있다. 여기서 $t^T(j)=(t_j, t_{j-1}, \cdots, t_{j-n+1})$은 연속된 n개의 관측치이며, s_j는 예상되는 반응치이다. 이때 행렬 X, ω 및 s는 각각

$$X\equiv X(n)=\begin{pmatrix} t^T(0) \\ t^T(1) \\ \vdots \\ t^T(p-1) \end{pmatrix},\ \omega\equiv\omega(n)=\begin{pmatrix} \omega_0 \\ \omega_1 \\ \vdots \\ \omega_{n-1} \end{pmatrix},$$

$$s\equiv s(n)=\begin{pmatrix} s_0 \\ s_1 \\ \vdots \\ s_p \end{pmatrix}$$

로 정의된다.

새로운 관측치 t_j가 도착하면 새로운 열 $t^T(p)$와 s_j가 각각 $X(n)$과 $s(n)$에 추가되거나 $T^T(0)$와 s_0가 각각 $X(n)$과 $s(n)$으로 부터 제거되어 $X(n+1)$과 $s(n+1)$이 생성된다. 즉, 본 논문에서 다루려고 하는 제거의 경우에 있어서 새로 생성되는 행렬은 각각

$$X(n)=\begin{pmatrix} t^T(0) \\ X(n+1) \end{pmatrix},\ s(n)=\begin{pmatrix} s_0 \\ s(n+1) \end{pmatrix}.$$

연속된 관측치의 입력에 대한 최소자승에 의한 적응 필터문제는 시점 n에서의 해 $\omega(n)$으로 부터 새로운 해 $\omega(n+1)$를 계산하는 것으로 식(3)에서 정의된 Cholesky factor R이 downdating된 행렬로 부터 새로운 $\tilde{R}$을 계산하는 문제로 귀결된다.

하나의 행에 대한 downdating 기법은 지난 몇년동안 광범위하게 연구되어 왔다[5, 6, 7, 8, 9]. 예를 들어 Bjorck [7]은 Givens rotation을 적용하여 단일 행에 대한 downdating을 연구하였으며, Alexander 등[8]은 hyperbolic rotation에 의한 downdating을 연구한 바 있다. 또한, Henkel 등[9]은 분산메모리 병렬시스템에서 hyperbolic downdating이 Givens rotation에 의한 방법에 비하여 성능이 우수함을 밝히고 있다.

관측행렬이 순차적으로 입력 큐에 저장되고, 각각의 관측행렬을 updating하는 시스템에서는 Kim [10]의 연구에서와 같이 Householder reflection를 이용하여 여러개의 행을 동시에 처리하는 기법이 연구된 바 있다. 특히 $m \times n$ 행렬의 QR-분할에 따르는 시간복잡도의 비교에서 Householder reflection이 mn^2 flops(floating point operations)를 필요로 하나 Givens rotation의 경우에는 $2mn^2$ flops가 필요하므로 Householder reflection이 Givens rotation에 의한 방법에 비하여 매우 우수하다. 그러나 아직까지 다행관측행렬을 downdating할 때에 Householder reflection에 의한 기법은 연구결과가 발표되지 않았다.

본 논문에서는 Householder reflection을 이용하여 downdating을 수행하는 기법의 개발에 앞서서 기존의 방법을 개선하여 빠른 downdating을 수행할 수 있는 알고리즘을 제안하였다. 즉, 기존의 경우에는 LINPACK에서 구현된 바와 같이 Givens rotation에 의한 CFD(Cholesky Factor Downdating)방법을 이용하여 다행관측행렬에 대한 downdating을 수행하나, 본 논문에서는 다행관측행렬을 Householder reflection에 의하여 상삼각행렬로 정렬하는 전처리과정을 거친 후 CFD의 방법을 적용하여 downdating하는 HCFD(Hybrid Cholesky Factor Downdating)방법을 제안하였다.

이를 위하여 제 2절에서는 updating과 downdating의 개념을 정리하고, 제 3절에서는 CFD에 의한 downdating과 본 논문에서 제안하는 HCFD에 의한 downdating의 이론적인 방향에서 접근하였다. 또한, 제 4절에서는 관측행렬의 크기를 변화시키면서 두가지 알고리즘을 이용한 multiple row downdating에 필요한 시간복잡도를 비교하고, 실제 환경에서 계산시간을 측정하여 비교하였다. 실험에 사용된 컴퓨터 환경은 Sun SPARC/2 워크스테이션과 국산주전산기 I 시스템(Tolerant System) 등으로, 동일한 다행관측행렬을 downdating하는데 필요한 계산시간을 측정하여 두가지 방법의 성능을 비교하였다. 마지막으로 제 5절에서는 연구결과를 밝히고 향후 연구계획에 대하여 기술하였다.

2. 다행관측행렬의 추가(Updating)와 제거(Downdating)

신호처리에 적용되는 공통적인 문제는 X가 $m \times n$ 행렬이며, s가 $m \times 1$ 벡터, ω가 $n \times 1$ 벡터일 때, 식(1)로 주어진 최소자승해를 구하는 문제로 정리될 수 있다. 이러한 최소자승해는 식(2)의 symmetric positive definite 시스템 방정식으로 부터 구하거나, 식(3)의 직교변환(orthogonalization)을 이용하여 구할 수 있다. 식(2) 또는 식(3)을 통해서 R을 계산한 후에 행렬 X와 벡터 s에 각각 행벡터 z^T와 스칼라 값 σ가 첨가되는 경우나, $\tilde{X}$로부터 하나의 행이 제거되고 이에 대응하는 값이 $\tilde{s}$에서 제거되는 경우에 변경된 행렬 X와 벡터 $\tilde{s}$에서 식(4)를 만족하는 새로운 해 $\tilde{\omega}$는 $\tilde{X}$로 부터 $\tilde{R}$을 계산하여 삼각행렬식의 해를 구하는 방법으로 계산될 수 있다. 행렬 X와 벡터 s에 각각 p개의 관측치로 구성된 행렬

$$Z^T \equiv \begin{pmatrix} z_1^T \\ z_2^T \\ \vdots \\ z_p^T \end{pmatrix}, \quad \sigma \equiv \begin{pmatrix} \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_p \end{pmatrix}$$

이 첨가되어 새로운 행렬

$\tilde{X}$와 벡터 $\tilde{s}$를 구성하면, $\tilde{X}=\begin{pmatrix} X \\ Z^T \end{pmatrix}$, $\tilde{s}=\begin{pmatrix} s \\ \sigma \end{pmatrix}$.

따라서 식(3)에 주어진 직교변환을 적용하면,

$$\tilde{X}^T\tilde{X} = R^TR + ZZ^T. \qquad (5)$$

여기서 R은 updating하기 전의 상삼각행렬이다. 그런데 식(5)에서 $\tilde{X}^T\tilde{X}$이 symmetric positive definite 행렬이므로, 갱신된 Cholesky factor $\tilde{R}$은 식(6)과 같이 R과 Z^T로 부터 직접 계산할 수 있다. 즉,

$$\tilde{R}^T\tilde{R}=R^TR+ZZ^T. \qquad (6)$$

마찬가지로 관측치를 기존의 정보로 부터 제거시키는 downdating은 X로부터 행렬 Z^T가 제거되고, s로 부터 σ가 제거된 선형대수식의 최소자승해를 계산하는 경우이다. 여기서

$$Z^T\omega=\sigma,\ X=\begin{pmatrix} \tilde{X} \\ Z^T \end{pmatrix},\ s=\begin{pmatrix} \tilde{s} \\ \sigma \end{pmatrix}. \qquad (7)$$

식(7)에 직교변환을 적용하면,

$$\tilde{X}^T\tilde{X} = R^TR - ZZ^T \qquad (8)$$

이므로 마찬가지로 식(9)와 같이 상삼각행렬 $\tilde{R}$을 R과 Z로 부터 직접 계산할 수 있다.

$$\tilde{R}^T\tilde{R} = R^TR - ZZ^T \qquad (9)$$

Kim[10]은 분산메모리 병렬처리 컴퓨터에서 $\begin{pmatrix} R \\ Z^T \end{pmatrix}$에 Householder reflection을 적용하여 $\begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}$인 상삼각행렬 $\tilde{R}$을 계산하는 다행관측행렬의 updating에 관한 연구를 수행하였으며, Givens rotation에 의한 방법에 비하여 그 결과가 우수함을 보고한 바 있다. 다행관측행렬에 대한 downdating의 경우에는 Givens rotation을 반복적으로 적용하는 방법에 대한 연구 결과만이 보고되었을 뿐이다. 본 논문에서는 Givens rotation과 Householder reflection을 혼합하여 시간복잡도가 개선된 다행관측행렬에 대한 downdating기법을 연구하였다.

3. 다행관측행렬의 Downdating 알고리즘

X를 $m\times n$ 행렬, R을 $n\times n$인 상삼각 행렬, $Z^T=\begin{pmatrix} z_1^T \\ z_2^T \\ \vdots \\ z_p^T \end{pmatrix}$,를 p개의 행벡터 z_k^T에 의하여 구성된 $p\times n$ 행렬이라 할 때, R에서 Z^T를 복원하는 과정은 단일행 z_k^T ($k=1,\ 2,\ \cdots,\ p$)에 대한 downdating을 반복하여 수행하거나 식(9)와 같이 행렬 Z^T에 대한 다행관측행렬의 downdating을 통하여 처리할 수 있다.

3.1 CFD에 의한 downdating

단일행의 downdating방법으로는 Gram-Schmidt의 QR-분할법[11,12]을 이용한 GSD (Gram-Schmidt Downdating) 기법과 LINPACK에서 이용하고 있는 CFD(Cholesky Factor Downdating) 기법[8, 13]이 광범위하게 사용된다.

GSD는 행렬 $\begin{pmatrix} R \\ Z_k^T \end{pmatrix}$를 updating하여 $\begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}$로 변환시키는 과정에서 계산된 직교변환 Q를 알고 있을 때에 Z^T의 k번째의 행인 관측치 z_k^T를 제거하는 방법으로, 직교변환 Q를 보관하고 있어야 하며, 임의의 관측치 $\tilde{z}_k^T \neq z_k^T$의 경우에는 적용할 수 없는 단점이 있다. 따라서 본 논문에서는 GSD를 다루지 않으며, 이에 대한 자세한 내용은 Bjorck [7]을 참조하기 바란다.

CFD는 X의 QR-분할의 결과로 얻어진 상삼각행렬 R과 관측치 z_k^T로 부터 downdating결과인 $\tilde{R}$을 계산하는 방법으로 GSD와는 달리 X의 직교변환 Q를 전혀 사용하지 않는 특징이 있다. Saunders [13]에 의하여 제안된 이 알고리즘은 LINPACK에 적용되고 있는 표준 알고리즘으로, downdating과정 중에서 Givens rotation이 반복적으로 적용된다. 이 알고리즘의 안정성 Stewart [14]에 의하여 연구된 바 있다.

CFD 알고리즘을 구성하는 수학적 배경은 다음과 같다. 즉, 식(10)과 같이 QR-분할에 의하여 삼각행렬 ($R\ u$)이 계산되었을 때, ($Z^T\ \sigma$)의 k번

째 행 ($z_k^T\ \sigma_k$)를 ($X\ s$)의 마지막행이라고 가정하고 ($z_k^T\ \sigma_k$)를 제거하는 downdating을 살펴보자.

$$(X\ s)=\begin{pmatrix} X & s \\ z_k^T & \sigma_k \end{pmatrix}=Q\begin{pmatrix} R & u \\ 0 & \rho \\ 0 & 0 \end{pmatrix}. \qquad (10)$$

여기서 열벡터 $e_p=(0, 0, \cdots, 0, 1)^T$을 ($X\ s$)의 좌측에 포함시키고 QR-분할을 수행하면,

$(e_p\ X\ s)=\begin{pmatrix} 0 & X & s \\ 1 & z_k^T & \sigma_k \end{pmatrix}$ 이므로 식(10)으로 부터

$$Q^T\ (e_p\ X\ s)=\begin{pmatrix} q_1 & R & u \\ \phi & 0 & \rho \\ q_2 & 0 & 0 \end{pmatrix},$$

여기서 Q의 마지막 행은 $q^T\equiv(q_1^T\ \phi\ q_2^T)$. 따라서 평면($j$, $n+2$)에서의 회전 G_j ($j=1, 2, \cdots, n+1$)을 구하면, 식(11)과 같이 ($z_k^T\ \sigma_k$)를 복원하는 변환을 구할 수 있다. 즉,

$$U_k^T\begin{pmatrix} q_1 & R & u \\ \phi & 0 & \rho \\ q_2 & 0 & 0 \end{pmatrix}=\begin{pmatrix} 0 & \tilde{R} & \tilde{u} \\ 0 & 0 & \tilde{\rho} \\ 0 & 0 & 0 \\ 1 & z_k^T & \sigma_k \end{pmatrix},$$

$$U_k^T=G_1G_2\cdots G_{n+1}. \qquad (11)$$

여기서 G_j는 q의 j번째 원소를 0으로 만들기 위한 회전변환이다.

그런데 식(11)에서 q_2는 $\bar{\gamma}e_n$, $\bar{\gamma}=\|q_2\|_2$로 치환할 수 있으므로 행렬 Q의 마지막 행의 첫 $n+1$개의 원소, 즉 ($q_1^T\phi$)와 $\bar{\gamma}\equiv\|q_2\|_2=\sqrt{1-(\|q_1\|^2+\phi^2)}$만을 가지고 ($z_k^T\ \sigma_k$)를 복원할 수 있다.

식 (10)에서 마지막 행은

$$(z_k^T\ \sigma_k)=(q_1^T\ \phi\ q_2^T)\begin{pmatrix} R & u \\ 0 & \rho \\ 0 & 0 \end{pmatrix}=(q_1^T\ \phi)\begin{pmatrix} R & u \\ 0 & \rho \end{pmatrix}.$$

여기서 q_1과 ϕ는 삼각행렬식 $\begin{pmatrix} R^T & 0 \\ u^T & \rho \end{pmatrix}\begin{pmatrix} q_1 \\ \phi \end{pmatrix}=\begin{pmatrix} z_k \\ \sigma_k \end{pmatrix}$의 해와 $u^Tq_1=u^TR^{-T}z_k=w^Tz_k$으로 부터 계산할 수 있다. 즉,

$$q_1=R^{-T}z_k,\ \phi=\frac{\sigma_k-w^Tz_k}{\rho}\quad(\rho\neq0) \qquad (12)$$

또한 식(11)로 부터 $U_k^T\begin{pmatrix} q_1 & R & u \\ \phi & 0 & \rho \\ \bar{\gamma} & 0 & 0 \end{pmatrix}=\begin{pmatrix} 0 & \tilde{R} & \tilde{u} \\ 0 & 0 & \tilde{\rho} \\ 1 & z_k^T & \sigma_k \end{pmatrix}$

를 만족하는 U_k^T는 Givens rotation 변환행렬인 $\begin{pmatrix} c & -s \\ s & c \end{pmatrix}$을 모든 열에 대해 적용함으로써 구할 수 있다. 예를 들어, 평면 ($n+1$, $n+2$)에서의 Givens rotation을 적용한 결과는 $\begin{pmatrix} c & -s \\ s & c \end{pmatrix}\begin{pmatrix} \phi & \rho \\ \bar{\gamma} & 0 \end{pmatrix}=\begin{pmatrix} 0 & \tilde{\rho} \\ \bar{\gamma} & \hat{\rho} \end{pmatrix}$이며, 각각의 값을 계산하면

$$\bar{\gamma}^2=\tilde{\gamma}^2+\phi^2=1-\|q_1\|_2^2,\quad \tilde{\rho}=\frac{\sigma_k-z_k^Tw}{\gamma},$$
$$\hat{\rho}=\sqrt{\rho^2-\tilde{\rho}^2}. \qquad (13)$$

이러한 과정을 z_1^T, z_2^T, $\cdots$, z_p^T에 대하여 순차적으로 적용하면, $Z^T=\begin{pmatrix} z_1^T \\ z_2^T \\ \vdots \\ z_p^T \end{pmatrix}$에 대한 downdating결과를 얻을 수 있다. 이상의 과정을 정리하면 다음의 알고리즘 1과 같다.

알고리즘 1 : CFD 알고리즘
input : R, u, ρ, w, and ($Z^T\ \sigma_k$);
output : $\tilde{R}$, $\tilde{u}$, $\tilde{\rho}$ and $\tilde{w}$;
1. **for** k = p **downto** 1 {
1.1. Compute q_1, γ, and $\tilde{\rho}$ from
R^Tq_1 = z_k , $\gamma=\sqrt{1-\|q_1\|_2^2}$, $\tilde{\rho}=(\sigma_k-z_k^Tw)/\gamma$.
1.2. Stop **if** $\|q_1\|$ ≥ 1 /* ill-conditioned */
1.3. Determine an orthogonal matrix U_k^T as a product of Givens rotations such that
$$\begin{pmatrix} 0 & \tilde{R} & \tilde{u} \\ 1 & z_k^T & \sigma_k \end{pmatrix}=U_k^T\begin{pmatrix} q_1 & R & u \\ \gamma & 0 & \tilde{\rho} \end{pmatrix}.$$
}
2. Compute the new solution $\tilde{w}$ and the residual norm from
$R\tilde{w}=\tilde{u}$, $\tilde{\rho}=\sqrt{\rho^2-\tilde{\rho}^2}$.

알고리즘 1은 전체적으로 pn회의 Givens rotation

을 필요로 하며, GSD 알고리즘[7]과는 달리 Q를 저장할 필요가 없으며, 관측행렬 $(Z^T\ \sigma)$로부터 Q를 생성할 수 있다. 알고리즘의 시간복잡도를 참고문헌 [15]에서와 같이 한번의 실수곱셈과 실수덧셈연산을 단위실수연산(flops: floating point operations)으로 간주하면 알고리즘 1을 이용하여 $p\times n$ 크기의 다행관측행렬 Z^T를 downdating하기 위한 시간복잡도는 $\frac{5}{2}pn^2$이다.

3.2 HCFD에 의한 downdating

CFD는 하나의 관측행에 대한 downdating시, n회의 Givens rotation이 필요하다. 따라서 Givens rotation의 횟수를 줄일 수 있다면 전체적으로 알고리즘의 성능이 개선될 수 있다. 그런데 만일 관측치 벡터 z_k^T가 첫번째 부터 몇개의 원소가 0인 벡터인 경우에는 Givens rotation을 적용할 필요가 없으므로 알고리즘의 성능이 개선될 수 있다. 즉, z_K^T가 첫번째부터 원소가 0인 부분벡터를 0_K라 하고 $*_K$를 0이 아닌 원소로 시작되는 벡터라고 하면, $z_K^T=(0_K\ *_K)$이다. 식(12)로 부터 $q_1=R^{-T}z_K=\begin{pmatrix}0\\ q_x\end{pmatrix}$. 여기서 q_x는 z_K^T의 0이 아닌 원소의 백터 $*_K$와 대응하는 상삼각행렬 R^{-T}의 부분 행렬과의 곱이다. 따라서 식(11)에서

$$U_k^T\begin{pmatrix}q_1\\ \phi\\ \gamma\end{pmatrix}=U_k^T\begin{pmatrix}0\\ q_x\\ \phi\\ \gamma\end{pmatrix}=\begin{pmatrix}0\\ \vdots\\ 0\\ 1\end{pmatrix}. \qquad (14)$$

식(14)를 만족하는 회전변환 U_k^T을 계산하는데 필요한 Givens rotation의 횟수는 q_x와 ϕ의 원소의 갯수에 영향을 받는다. 따라서 z_K^T의 원소중에서 0일 많을 수록 회전변화의 횟수가 줄어들게 된다.

이러한 관찰로부터 다행관측행렬 Z^T에 QR-분할법을 적용하면,

$$Z^T=Q_Z\,R_Z^T \qquad (15)$$

여기서 R_Z^T는 상삼각행렬이며, Q_Z는 Z^T를 R_Z^T로 변환하는 직교변환행렬이다. R_Z^T는 Z^T의 행의 수 p가 n 보다 클 경우에는 완전한 상삼각행렬이 되지만 반대의 경우에는 아래 부분이 평평한 불완전 상삼각행렬이 된다.

식(9)에 식(15)를 대입하면

$$\tilde{R}^T\tilde{R}=R^TR-ZZ^T=R^TR-R_Z^TR_Z^T. \qquad (16)$$

따라서 Z^T에 대한 downdating의 결과는 식(16)에서와 같이 상삼각행렬 R과 Z^T의 직교변환 결과로 얻어진 R_Z^T로부터 직접 구할 수 있다. 그런데 식(14)에서와 같이 R_Z^T의 k번째 행을 downdating하기 위한 회전변환을 U_k^T이라고 하면, R_Z^T를 downdating하기 위한 회전변환

$$U^T=U_1^TU_2^T\cdots U_v^T,$$

$$U_k^T\begin{pmatrix}q_1\\ \phi\\ \gamma\end{pmatrix}=U_k^T\begin{pmatrix}0\\ q_x\\ \phi\\ \gamma\end{pmatrix}=\begin{pmatrix}0\\ \vdots\\ 0\\ 1\end{pmatrix},\quad k=v,v-1,\cdots,1.$$

여기서

$$v=\begin{cases}n: p\geq n\\ p: \text{otherwise.}\end{cases}$$

또한

$$U_k^T=G_kG_{k+1}\cdots G_{v-1}.$$

여기서 G_k는 $(q_x^T\ \phi)$을 0으로, $\bar{\gamma}=\sqrt{1-(\|q_x\|^2+\phi^2)}$으로 변화시키는 Givens rotation이다.

따라서 downdating을 위하여 필요한 회전변환의 횟수는

$$\begin{cases}\dfrac{n(n+1)}{2} & p\geq n\\ \dfrac{p(2n-p+1)}{2} & \text{otherwise.}\end{cases}$$

이는 3.1절의 CFD에 비하여 회전변환의 횟수가 1/2 수준으로 줄어드는 것을 의미한다. 본 논문에서는 이 방법이 QR-분할과 CFD를 이용하고 있으므로 HCFD(Hybrid Cholesky Factor Downdating) 알고리즘이라고 명명하였다.

Z^T를 QR-분할할 때에는 Householder reflection이

Givens rotation에 비하여 우수하므로 Householder reflection에 의한 QR-분할법을 적용한다. 즉, $p\times n$ 행렬 Z^T에 대한 Householder updating에서 j번째 열을 갱신하기에 앞서 1, …, j-1번째 열이 이미 0으로 갱신되었다고 가정할 때, j번째 열의 갱신에 따른 Cholesky factor $Z^{T(j)}$는 다음과 같다.

$$Z^{T(j)} = (I - \frac{1}{\mu} u_j u_j^T) Z^{T(j-1)} = Q_Z^{(j)} Z^{T(j-1)} \quad (17)$$

여기서 u_j는 Householder 벡터

$$u_j = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ z_{jj}^{T(j-1)} + \rho \\ z_{j+1,j}^{T(j-1)} \\ \vdots \\ z_{pj}^{T(j-1)} \end{pmatrix}, \quad \rho = \sqrt{\sum_{k=j}^{p} (z_{kj}^{T(j-1)})^2},$$

$$\mu = \frac{u_j^T u_j}{2} = \rho^2 + \rho z_{jj}^{T(j-1)}.$$

j번째 열에 대한 updating은 j열에 대한 변환행렬 $Q_Z^{(j)}$를 구하는 것이며, 전체적으로 $Q_Z = Q_Z^{(n)} Q_Z^{(n-1)} \cdots Q_Z^{(1)}$ 인 직교변환 Q_Z에 의하여

$$Q_Z Z^T = Z^{T(n)} \equiv \begin{cases} \begin{pmatrix} R_n \\ 0 \end{pmatrix} & \text{if } p \geq n \\ (R_p \; Y_{n-p}) & \text{otherwise} \end{cases} \quad (18)$$

인 R_Z^T를 구할 수 있다. 여기서 R_n, R_p는 각각 $n\times n$ 및 $p\times p$ 상삼각행렬이고, Y_{n-p}는 $p\times(n-p)$ 행렬이다.

HCFD에 의한 downdating을 정리하면 알고리즘 2와 같다. 또한 HCFD 알고리즘에서 필요한 실수연산의 횟수는 p의 범위에 따라 달라진다. 즉, HCFD 알고리즘의 시간복잡도는

$$\begin{cases} pn^2 + \frac{5}{6} n^3 & \text{in } p \geq n \\ \frac{5}{2} pn^2 + \frac{1}{3} p^3 - np^2 & \text{otherwise.} \end{cases}$$

알고리즘 2: HCFD 알고리즘

input : R, u, ρ, w, and $(Z^T \; \sigma)$;
output : $\tilde{R}$, $\tilde{u}$, $\tilde{\rho}$ and $\tilde{w}$;

1. for j = 1 to min(n, p) { /* QR-factorizing Z^T */
   1.1. Determine a Householder matrix $Q_Z^{(j)}$ from $I - \frac{1}{\mu} u_j u_j^T$.
   1.2. Compute $(Z^{T(j)} \; \sigma^{(j)}) = Q_Z^{(j)} (Z^{T(j-1)} \; \sigma^{(j-1)})$.
   }
2. for k = min(n, p) downto 1 { /* Givens downdating for $Z^{T(n)} \equiv \begin{pmatrix} R_Z^T \\ 0 \end{pmatrix}$ */
   2.1. Compute q_x, γ, $\tilde{\rho}$ from
   $z_k^T = (0_k \; *_k)$, $R^{-1} z_k = \begin{pmatrix} 0 \\ q_x \end{pmatrix}$,
   $\gamma = \sqrt{1 - \|q_x\|_2^2}$, $\tilde{\rho} = (\sigma_k - z_k^T w)\gamma$.
   2.2. Stop if $\|q_x\| \geq 1$.
   2.3. Determine an orthogonal matrix U_k^T as a product of Givens rotations such that
   $$\begin{pmatrix} 0 & R & \tilde{u} \\ 1 & z_k^T & \sigma_k \end{pmatrix} = U_k^T \begin{pmatrix} q_x & R & u \\ \gamma & 0 & \tilde{\rho} \end{pmatrix}.$$
   }
3. Compute the new solution $\tilde{w}$ and the residual norm from
   $$\tilde{R}\tilde{w} = \tilde{u}, \quad \tilde{\rho} = \sqrt{\rho^2 - \tilde{\rho}^2}.$$

4. 알고리즘의 비교 및 분석

4.1 실수연산의 시간복잡도 비교

$n\times n$ 행렬 R과 $p\times n$ 행렬 Z^T에 대해 각각 CFD와 HCFD 알고리즘을 이용하여 downdating하는데 필요한 시간복잡도는 〈표 1〉과 같다. 〈표 1〉에서 HCFD 알고리즘의 복잡도를 두가지로 구분한 이유는 $Z^T = Q_Z R_Z^T$에서 R_Z^T를 계산할 때, 식(18)과 같이 p의 범위에 따라 상이한 형태의 상삼각행렬이 생성되므로 각각의 경우별로 시간복잡도의 계산이 다르기 때문이다. 〈표 1〉로 부터 관측행렬의 행의 수가 증가함에 따라 HCFD 알

고리즘의 시간복잡도가 CFD 알고리즘의 시간복잡도에 비하여 작음을 알 수 있다.

〈표 1〉 CFD와 HCFD의 시간복잡도
〈Table 1〉 Computation complexities of the algorithms based on CFD and HCFD

알고리즘	computation complexity
CFD	$\frac{5}{2}pn^2$
HCFD	$\begin{cases} pn^2+\frac{5}{6}n^3 & \text{if } p \geq n \\ \frac{5}{2}pn^2+\frac{1}{3}p^3-np^2 & \text{otherwise} \end{cases}$

4.2 실험 및 고찰

기존의 CFD와 본 논문에서 제안한 HCFD 알고리즘의 성능을 비교하기 위하여 여러가지 컴퓨터 환경하에서 관측치의 크기를 변경하면서 계산시간을 측정하고 그 값을 비교하였다. 실험에 사용된 컴퓨터로는 Sun SPARC/2 워크스테이션(메모리 16M byte)과 국산주전산기 I(Tolerant)로, 국산주전산기 I은 32 비트급 532 보드와 메인 메모리 16M byte를 장착한 개량형 시스템이다.

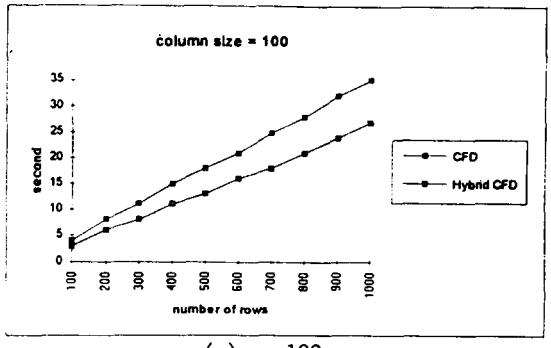
관측치 열의 크기와 행의 크기를 변화시키면서 계산에 소요된 시간을 측정한 결과는 (그림 1)~(그림 4)와 같다. (그림 1) (a)는 Sun SPARC/2 시스템에서 열의 크기가 100일 때 행의 크기를 100에서 1,000까지 변화시키면서 계산에 소요되는 시간을 측정한 것으로, HCFD에 의한 경우가 CFD를 이용한 경우에 비하여 계산시간이 적게 걸리며, 행의 크기가 증가함에 따라 성능향상이 뚜렷이 증가함을 알 수 있다. (그림 1) (b)는 열의 크기가 200인 경우에 대한 실험 결과로 열의 크기가 100인 경우와 마찬가지로 HCFD에 의한 경우가 CFD를 이용한 경우에 비하여 성능이 우수하였다.

(그림 2)는 열의 크기가 각각 100 및 200일 때, 국산주전산기 I에서의 실험결과이다. 국산주전산기 I의 경우에도 행의 크기가 변화함에 따라 HCFD에 의한 방법이 CFD에 의한 방법에 비하여 성능이 우수함을 알 수 있다.

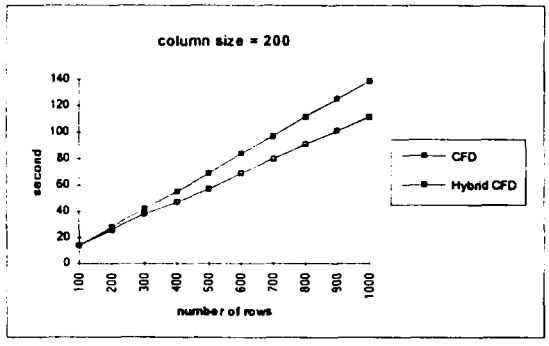
본 논문에서 제안한 HCFD 알고리즘의 성능이 CFD 알고리즘에 비하여 어느정도 향상되었는 지를 측정하기 위하여 동일한 다행관측행렬에 대한 계산소요시간을 토대로 다음과 같이 성능향상률(performance improvement rate)을 정의하였다. 즉, 성능향상률

$$\xi \equiv 1-\psi, \quad \psi = \frac{t_{HCFD}}{t_{CFD}}.$$

여기서 t_{HCFD}와 t_{CFD}는 각각 HCFD 및 CFD 알고



(a) $n=100$



(b) $n=200$

(그림 1) Sun SPARC/2상에서 다행관측행렬의 downdating 소요시간
(Fig. 1) Computation times for downdating multiple rows on Sun SPARC/2

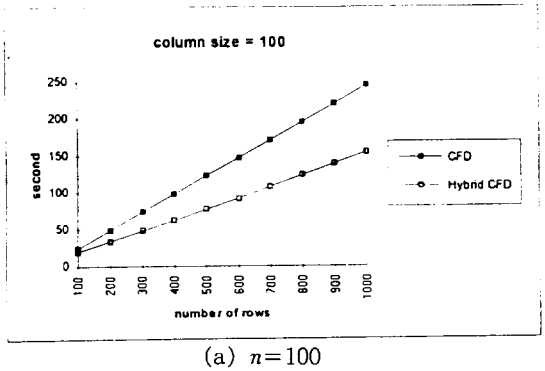
리즘에 의하여 다행관측행렬을 downdating하는 데 걸리는 시간이다. 즉,

$$\xi = \frac{t_{CFD} - t_{HCFD}}{t_{CFD}}.$$

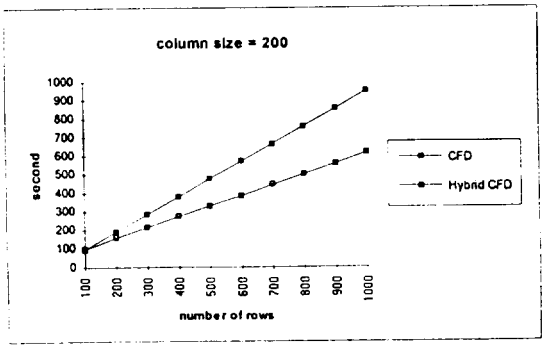
(그림 3)과 (그림 4)는 여러가지 컴퓨터 시스템하에서의 성능향상률을 보여준다. 열의 크기가 100인 경우 Sun SPARC/2 시스템에서는 관측행렬의 크기가 증가하여도 성능향상률의 변화폭이 작으나, 국산주전산기 I의 경우에는 관측행렬의 크기가 증가함에 따라 큰 폭의 성능향상이 있음을 알 수 있다. 또한 열의 크기가 200인 경우에도 관측행렬의 크기가 증가함에 따라 성능향상률이 점차 증가함을 알 수 있다. 이러한 사실로 보아 HCFD 알고리즘이 CFD 알고리즘에 비하여 우수한 성능을 지니고 있음을 알 수 있으며, 특히 관측행의 크기가 큰 경우에는 HCFD 알고리즘이 CFD 알고리즘에 비하여 성능이 우수한 것을 알 수 있다. 또한 이러한 결과는 시간복잡도의 비교결과와도 부합됨을 알 수 있다.

5. 결 론

본 논문에서는 신호처리의 공통문제라 할 수 있는 최소자승해를 구하는 과정에서 반드시 필요한 updating 및 downdating 알고리즘을 연구하고, 기존의 CFD 알고리즘을 개선한 HCFD 알고리즘을 제안하였다. 기존의 CFD 알고리즘은 R과 Z^T로부터 Givens rotation으로 이루어진 변환행렬 U^T를

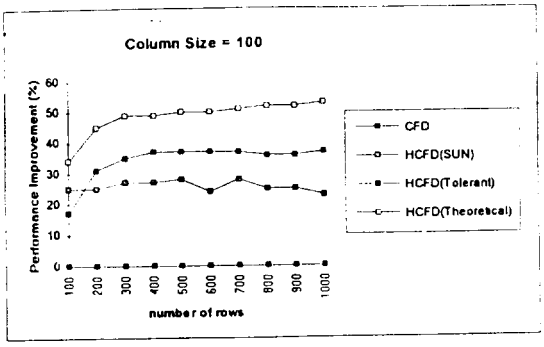


(a) $n=100$

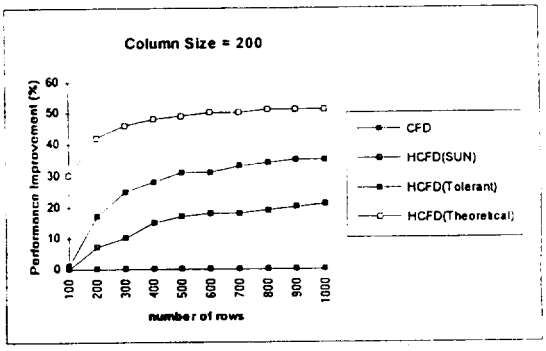


(b) $n=200$

(그림 2) 국산주전산기 I에서 다행관측행렬의 downdating 소요시간
(Fig. 2) Computation times for downdating multiple rows on Tolerant System



(그림 3) 열의 크기가 100인 경우의 성능향상률 비교
(Fig. 3) Comparison of performance improvement ($n=100$)



(그림 4) 열의 크기가 200인 경우의 성능향상률 비교
(Fig. 4) Comparison of performance improvement ($n=200$).

구하는 방법으로 전체적으로 $\frac{5}{2}pn^2+\frac{1}{3}p^3-np^2$회의 실수연산이 필요하다. 이에 비하여 HCFD 알고리즘에서는 Z^T로 부터 Householder reflection을 이용한 QR-분할법을 적용하여 R_Z^T를 계산한 후, R_Z^T에 CFD 알고리즘을 적용하므로 시간복잡도를 크게 개선하였다. 그 결과, HCFD 알고리즘은 $p \geq n$인 경우 $pn^2+\frac{5}{6}n^3$회의 실수연산이, 그리고 $p < n$인 경우에는 $\frac{5}{2}pn^2$회의 실수연산이 각각 필요하므로 기존의 CFD 알고리즘에 비하여 실수연산의 횟수가 크게 감소하였음을 알 수 있다.

Sun SPARC/2 워크스테이션과 국산주전산기 I (Tolerant) 상에서 관측치의 행과 열의 크기를 변화시키면서 계산시간을 측정하여 비교한 결과, 실제 계산에서도 제안된 알고리즘이 기존 알고리즘에 비하여 우수함을 알 수 있었다. 특히 Sun SPARC/2의 경우에는 10%~30%, 국산주전산기 I의 경우에는 25%~40%의 성능 향상이 있음을 확인할 수 있었다.

이러한 특징은 여러가지 구조적 특징을 지닌 컴퓨터 시스템, 예를 들면, 파이프라인 프로세서 또는 병렬처리 컴퓨터 등에서도 유지될 수 있을 것으로 보이며, 앞으로는 이들 시스템에서도 제안된 알고리즘을 적용하여 성능향상이 가능한 지에 대한 연구를 계속할 예정이다.

참 고 문 헌

[1] S. T. Alexander, 'Adaptive Signal Processing,' Springer-Verlag, New York, 1986.

[2] M. T. Heath and C. H. Romine, "Parallel solution of triangular systems on distributed-memory multiprocessors," SIAM Jour. Sci. Statist. Comput., Vol. 9, No. 3, May 1988.

[3] 김석일, 김홍기, "PALM시스템 네트웍하에서의 효과적인 삼각선형시스템," 한국정보과학회 91 충청지회 추계논문발표집, pp. 54-58, 1991.

[4] P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders, "Methods for modifying matrix factorizations," Math. Comput., Vol. 28, pp. 505-535, 1974.

[5] A. W. Bojanczyk, R. P. Brent, P. Van Dooren, and F. R. de Hoog, "A note on downdating the Cholesky factorization," SIAM Jour., Sci. Statist. Comput., Vol. 8, No. 3, pp. 201-221, 1987.

[6] C. Bishof and C. F. Van Loan, "The WY representation for products of Householder matrices," SIAM Jour. Sci. Statist. Comput., Vol. 8, No. 1, s2-s13, 1987.

[7] A. Bjork, H. Park, and L. Elden, "Accurate downdating of least squares solutions," SIAM Jour. Matrix Anal. Appl., Vol. 15, No. 2, pp. 549-568, April 1994.

[8] S. T. Alexander, C. T. Pan, and R. J. Plemmons, "Analysis of a recursive least squares hyperbolic rotations algorithm for signal processing," Linear Algebra Appl., Vol. 98, pp. 3-40, 1988.

[9] C. S. Henkel, M. T. Heath, and R. J. Plemmons, "Cholesky downdating on a hypercube," In Fox. G. (Ed.), Proc. Third Conference on Hypercube Concurrent Comput. Appl., ACM Press, New York, NY., pp. 1592-1598, 1989.

[10] Sukil. Kim, D. P. Agrawal and R. J. Plemmons, "Least-squares multiple updating algorithms on a hypercube," Jour. Par. and Dist. Computing, Vol. 8, pp. 80-88, 1990.

[11] L. Reichel and W. B. Gragg, "FORTRAN subroutines for updating the QR decomposition," ACM Trans. Math. Software, Vol. 16, pp. 369-377, 1990.

[12] B. N. Parlett, 'The Symmetric Eigenvalue Problem,' Prentice-Hall, Englewood Cliffs, NJ, 1980.

[13] M. A. Saunders, "Large-scale linear programming using the Cholesky factorization," Tech. Report CS252, Computer Science Dept., Stanford University, Stanford, CA, 1972.

[14] G. W. Stewart, "The effects of rounding error on an algorithm for downdating a Cholesky factorization," Jour. Inst. Math. Appl., Vol. 23, pp. 203-213, 1979.

[15] G. H. Golub and C. F. Van Loan, 'Matrix Computations,' Johns Hopkins Univ. Press, Baltimore, MD, 1983.

이 충 한

1994년 충북대학교 컴퓨터과학과(학사)
1994년~현재 충북대학교 전자계산학과 석사과정
관심분야: 병렬처리 컴퓨터구조, 병렬처리 알고리즘.

김 석 일

1975년 서울대학교 전기공학과 졸업(학사)
1984년 연세대학교 전자계산학(공학석사)
1989년 North Carolina State University 전기 및 컴퓨터공학(공학박사)
1975년~90년 국방과학연구소 연구원, 선임연구원
1990년~현재 충북대학교 자연과학대학 컴퓨터과학과 조교수
관심분야: 병렬처리 컴퓨터 구조, 병렬처리 알고리즘, Supercomputing, Advanced factory automation 등.