UDU 행렬분해법을 이용한 재귀적 TLS 알고리즘

임준석 * , 최낙진** 성굉모**
*세종대학교 전자공학과, * 서울대학교 전기공학부

A UDU decomposition based recursive total least square method

Jun-seok Lim*, Nakjin Choi**, KoengMo Sung**
*Dept. Electroncis Eng., Sejong Univ., **School of Electrical Eng., Seoul National Univ.
*jslim@sejong.ac.kr

본 논문은 IDEC의 부분 지원을 받았음을 밝힙니다.

요약

본 논문은 시스템 인식에서 RLS의 성능을 높이기 위한 한 방법으로 UDU 행렬 분해법을 바탕으로 한 recursive total least squares (RTLS) algorithm을 제안한다. 기존의 RTLS는 Power Method에 의거해서 recursive하게 만든 형태이어서 RLS와 거의 같은 구조이다. 그러나 본 논문에서는 일반적인 Power Method가 rank-1 update를 이용하기 때문에 ill-condition에 빠질 가능성이 높은 점을 감안하여, UDU 행렬 분해법을 사용한 RTLS방법을 제안하고, 그를 시스템 인식에 적용한다.

1. 서론

시스템 인식에서 RLS의 쓰임은 잘 정립되어 있는 방법이다. 그러나 RLS로 시스템 인식을 할 때 입력측과 출력측에 작은 오차가 함께 존재할 경우도 있다. 이런 때 Least Square풀이법의 한 방법인 RLS는 좋은 성능을 낼 수 없다. 일반적으로 TLS(total least square)는 입/출력측에 모두 오류가 존재할 경우 좋은 결과를 내는 것으로 알려져 있다. 그러나 재귀적 구조를 가지고 있지 않다.

본 논문은 시스템 인식에서 RLS의 성능을 높이기 위한 한 방법으로 새로운 recursive total least squares (RTLS) algorithm을 제안한다. 이는 TLS를 Power Method에 의거해서 recursive하게 만든 형태이어서 RLS와 거의 같은 구조이다. 그리고 본 논문에서는 일반적인 Power Method가 rank-1 update를 이용하기 때문에 ill-condition에 빠질 가능성이 높은 점을 감안하여, UDU 행렬 분해법을 사용한 RTLS방법을 제안하고, 그를 MFNN의 훈련방법으로 적용한다.

2. Recursive TLS method for MFNN

그림1(a)와 같은 시스템 인식에서 그림1(b)과 같이 입력과 출력에 잡음이 동시에 존재할 경우 일반적인 TLS 문제는 다음과 같은 식의 최소화로 나타낼 수 있다.

$$\min \left\| [\mathbf{E}|\mathbf{r}] \right\|_F \text{ s.t. } (\mathbf{A}+\mathbf{E})\mathbf{x} = \mathbf{b}+\mathbf{r} \qquad (1)$$

여기서 A는 MxN의 입력 행렬, b는 Mx1의 출력 벡터이다. E와 r은 각각 입력 행렬의 잡음과 출력 벡터의

잡음을 의미한다. Golub 등은 위 식을 최소하기위해서 다음과 같은 확장된 행렬을 정의하고

$$\overline{\mathbf{A}} = [\mathbf{A}|\mathbf{b}] \tag{2}$$

그의 최소 크기 고유치에 해당하는 고유 벡터V_{N+1}를 구하여 그로부터 해, x_{tls}를 얻는 방법을 제안하였다[1].

$$\mathbf{X}_{TLS} = -\frac{1}{v_{N+1,N+1}} \begin{pmatrix} v_{N+1,1} \\ \vdots \\ v_{N+1,N} \end{pmatrix}, \tag{3}$$

여기서 $\mathbf{V}_{N+1} = \begin{pmatrix} v_{N+1,1} & \cdots & v_{N+1,N+1} \end{pmatrix}^T$ 이다.

이 최소 고유치에 해당하는 고유 벡터를 구하기 위해서는 여러 방법을 사용할 수 있으나 실시간 응용을 위해서 다음과 같은 Power Method에 의한 재귀적인 방법에 의한 최소 고유치에 대한 고유벡터를 구할 수 있다.

표1. recursive total least square 방법[1]

$\mathbf{P}(0) = \delta^{-1}\mathbf{I}$, $V(0) = \left(\frac{1}{\sqrt{N+1}} \quad \cdots \quad \frac{1}{\sqrt{N+1}} \right)^T$,

여기서 δ 는 임의의 작은 값이고, I는 단위 행렬이다.

n=1 ... N

$$\overline{\mathbf{a}(\mathbf{n})} = \begin{pmatrix} \mathbf{a}(\mathbf{n})^T & \mathbf{b}(\mathbf{n})^T \end{pmatrix}^T$$

$$\mathbf{k}(\mathbf{n}) = \frac{\lambda^{-1}\mathbf{P}(\mathbf{n}-1)\bar{\mathbf{a}}(\mathbf{n})}{1+\lambda^{-1}\bar{\mathbf{a}}^{H}(\mathbf{n})\mathbf{P}(\mathbf{n}-1)}$$

$$\mathbf{P}(\mathbf{n}) = \lambda^{-1}\mathbf{P}(\mathbf{n}-1) - \lambda^{-1}\mathbf{k}(\mathbf{n})\bar{\mathbf{a}}^{H}(\mathbf{n})\mathbf{P}(\mathbf{n}-1)$$

$$\mathbf{v}(\mathbf{n}) = \mathbf{P}(\mathbf{n})\hat{\mathbf{v}}_{N+1}(\mathbf{n}-1)$$

$$\hat{\mathbf{v}}_{N+1}(\mathbf{n}) = \mathbf{v}(\mathbf{n})/\|\mathbf{v}(\mathbf{n})\|$$

$$\hat{\mathbf{x}}_{TLS}(n) = -\frac{1}{\hat{v}_{N+1,N+1}(n)}\left(\hat{v}_{N+1,1} \cdots \hat{v}_{N+1,N+1}\right)^T$$

3. UDU decomposed Recursive TLS method

RTLS에서 핵심은 P(n)을 구하는 것이라고 할 수 있다. 그러나 컴퓨터를 이용하여 계산할 경우 유한 자리 수로 인한 여러 가지 오류로 인하여, P(n)은 대칭성을 잃게 되고 또 Positive definite한 성질도 상실되게 되는 경우가 발생한다. 이른 극복하기 위해 이용할 수 있는 방법에는 SVD, square-root, UDU 분해법 등이 있다. 이들 중에서 UDU 분해법이 계산상의 효율성 때문에 많이 선호하고 있다. 본 절에서는 앞 절에서 제안한 RTLS의 계산 방법의 안정성을 제고하는 방법으로 UDU 행렬분해법을 적용한 알고리즘을 제안한다[2].

앞 절에서 ,

$$\begin{aligned}\mathbf{P}(\mathbf{n}) &= \lambda^{-1}\mathbf{P}(\mathbf{n}-1) - \lambda^{-1}\mathbf{k}(\mathbf{n})\bar{\mathbf{a}}^{H}(\mathbf{n})\mathbf{P}(\mathbf{n}-1) \\ &= \lambda^{-1}\mathbf{P}(\mathbf{n}-1) - \lambda^{-1}\mathbf{P}(\mathbf{n}-1)\bar{\mathbf{a}}(\mathbf{n})\bar{\mathbf{a}}^{H}(\mathbf{n})\mathbf{P}(\mathbf{n}-1)\end{aligned} \tag{4}$$

에 UDU 행렬 분해법을 $\mathbf{P}(\mathbf{n}-1) = \mathbf{U}(\mathbf{n}-1)\mathbf{D}(\mathbf{n}-1)\mathbf{U}^{H}(\mathbf{n}-1)$ 적용하면 다음과 같다.

$$\mathbf{P}(\mathbf{n}) = \lambda^{-1}\mathbf{U}(\mathbf{n}-1)\left[\mathbf{D}(\mathbf{n}-1) - \frac{\mathbf{gg}}{\boldsymbol{\beta}}\right]\mathbf{U}(\mathbf{n}-1) \tag{5}$$

여기서

$$\mathbf{g} = \mathbf{D}(\mathbf{n}-1)\mathbf{e}, \mathbf{e} = \mathbf{U}(\mathbf{N}-1)\bar{\mathbf{a}}(\mathbf{n})$$

$$\boldsymbol{\beta} = \mathbf{e}^{T}\mathbf{D}(\mathbf{n}-1)\mathbf{e} + \lambda$$

실제로 UDU행렬 분해를 효율적으로 수행하는 여러가지 알고리즘이 있으나 그 중 Bierd의 알고리즘을 사용한다면 앞 절의 RTLS와 거의 비슷한 복잡도를 갖는 알고리즘을 구성할 수 있다.

표1. recursive total least square 방법

$$\mathbf{P(0)} = \delta^{-1}\mathbf{I},\ V(0) = \left(\frac{1}{\sqrt{N+1}} \quad \cdots \quad \frac{1}{\sqrt{N+1}}\right)^T,$$

여기서 δ 는 임의의 작은 값이고, I는 단위 행렬이다.

n=1 ... N

$$\overline{\mathbf{a(n)}} = \left(\mathbf{a(n)}^{\mathbf{T}} \quad \mathbf{b(n)}^{\mathbf{T}}\right)^{\mathbf{T}}$$

$$\mathbf{e} = \mathbf{U(n-1)}\bar{\mathbf{a}}^{\mathbf{H}}\mathbf{(n)}, \mathbf{g} = \mathbf{D(n-1)e}, \alpha_0 = \lambda$$

j=1,...,n_e

$$\alpha_j = \alpha_{j-1} + e_j g_j$$
$$D_{jj}(n) = D_{jj}(n-1)\alpha_{j-1}/\alpha_j\lambda$$
$$\nu_j = g_j$$
$$\mu_j = -e_j/\alpha_{j-1}$$

i=1,...,j-1

$$U_{ij}(n) = U_{ij}(n-1) + \nu_i\mu_j$$
$$\nu_i = \nu_i + U_{ij}(n-1)\nu_j$$

$$\mathbf{P(n)} = \mathbf{U(n)D(n)U(n)}$$
$$\mathbf{v(n)} = \mathbf{P(n)}\hat{\mathbf{v}}_{\mathbf{N+1}}\mathbf{(n-1)}$$

$$\hat{\mathbf{v}}_{\mathbf{N+1}}\mathbf{(n)} = \mathbf{v(n)}/\|\mathbf{v(n)}\|$$

$$\hat{\mathbf{x}}_{\mathbf{TLS}}(n) = -\frac{1}{\hat{v}_{N+1,N+1}(n)}\left(\hat{v}_{N+1,1}\cdots\hat{v}_{N+1,N+1}\right)^T$$

4. Simulation

제안된 알고리즘의 성능을 보이기 위해서 그림2와 같은 MFNN를 이용한 시스템 인식을 가정하고 이를 RLS와 표1과 같은 RTLS 및 표2와 같은 UDU-RTLS로 각각 훈련한 결과를 그림 3와 그림4에 각각 보였다. 이 때 다음 식과 같은 가상의 모델을 세우고 MFNN이 이를 바탕으로 인식하도록 하였다.

$$y = \sin(3x),$$

여기서 y는 출력이고 x는 입력이다. 위 시스템을 잡음하에서 인식하기 위해서 인식 훈련할 때 MFNN의 입력과 출력에 각각 10^{-4}의 표준편차를 갖는 잡음이 부가하였다.

그림 3에서 보면 훈련 알고리즘의 성능을 알 수 있다. 즉 UDU-RTLS가 가장 좋고, 그 다음이 RTLS이어 RLS 순이 됨을 알 수 있다. 이를 좀 더 명확히 보기 위해서 그림 4에서는 오차를 나타내었다. 이를 보면 그림 3에서 관찰 했던 성능의 우위가 잘 나타나고 있다.

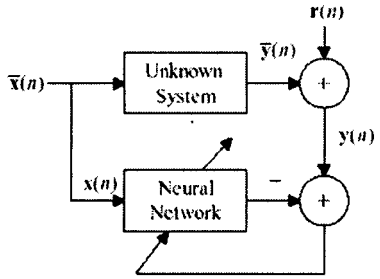
5. 결론

본 논문에서는 좀 더 안정한 시스템 인식을 위한 알고리즘의 하나로 새로운 RTLS를 이용하는 방법을 제안하였다. 본 알고리즘의 안정성 확보을 위해서 UDU 행렬 분해법을 적용한 알고리즘으로 발전 시켜보았다.

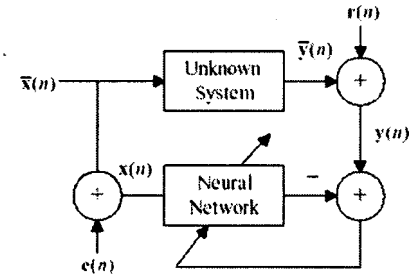
참고문헌

[1] 5. Nakjin Choi, J.S. Lim, J. Song, K.M. Sung, "Adaptive System Identi-¯cation Using an E±cient Recursive Total Least Squares Algorithm," Journal of the Acoustical Society of Korea, Vol. 22, No3E, pp.93-100, 2003.9.

[2] Ljung and Soderstrom, *Theory and Practice of Recursive Identification*, MIT press, Cambridge, MA, U.S.A., 1987



(a) Output measurement noise



(b) Input and output measurement noise

그림1. Training of Neural Networks

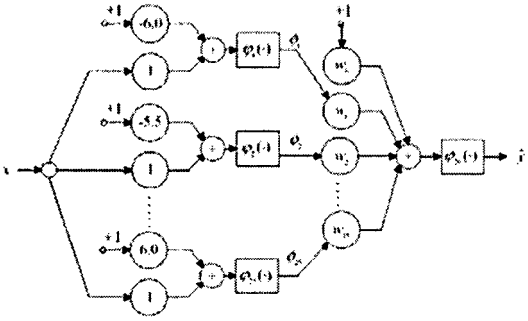


그림2. MFNN model

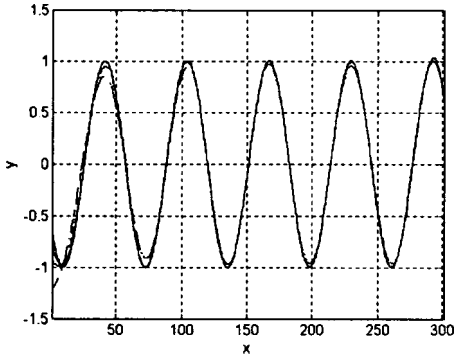


그림3. 훈련 성능 비교(파선:RLS, 일점쇄선:RTLS, 실선:UDU-RTLS)

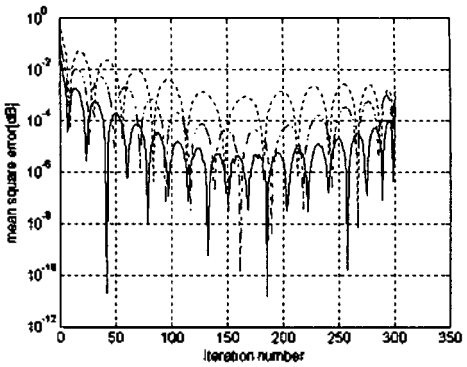


그림4. 훈련 오차 비교(파선:RLS, 일점쇄선:RTLS, 실선:UDU-RTLS)