논문 01-26-4B-6 한국통신학회논문지 '01-4 Vol.26 No.4B

4-점 리버스 자켓 변환을 이용한 N-점 고속 푸리에 변환

정회원 이 승 래*, 성 굉 모*

N-Point Fast Fourier Transform Using 4 x 4 Fast Reverse Jacket Transform

Seung-Rae Lee*, Koeng-Mo Sung* *Regular Members*

요 약

4-점 리버스 자켓 변환 (4-point Reverse Jacket transform)의 장점 중의 하나는 4-점 fast Fourier transform (FFT)시 야기되는 실수 또는 복소수 곱셈을 행렬분해 (matrix decomposition)를 이용, 곱셈인자를 모두 대각행렬에만 집중시킨, 매우 간결하고 효율적인 알고리즘이라는 점이다. 본 논문에서는 이를 N 점 FFT에 적용하는 알고리즘을 제안한다. 이 방법은 기존의 다른 변환형태보다 확장하거나 구조를 파악하기에 매우 용이하다.

ABSTRACT

One of the advantages using the 4x4 Reverse Jacket transform is the fact that it is very simple and efficient algorithm with which all entries including the real or complex-multiplicity arising in a 4-point FFT are positioned into a diagonal matrix. In this paper, we propose the algorithm applied to N-point FFT. By this method it is very easy to extend and understand the structure of the FFT. Z

Ⅰ. 서 론

신호처리와 분석에서 사용되는 기본 변환들 중의 하나는 푸리에 변환이다. 푸리에 변환에 대한 고속 알고리즘을 개발하는 목적으로 하다마드 변환이 활용되어 왔다[1,7]. 만을 행렬요소로 갖는 하다마드 행렬과는 달리, 일부 혹은 모든 요소들에 대해 임의의 실수 또는 복소수를 사용할 수 있다는 장점을 갖는 리버스 자켓 행렬은 참고문헌 [5]에서 최초로 소개되었다. 리버스 자켓 행렬의 특이한 현상은 그의 역행렬을 구할 때 본 행렬 (original matrix) 의 요소들이 그룹을 지어 위치들이 이동한 형상으로 나타나 마치 등산복을 뒤집어 입은 것처럼 보인다 하여 그의 이름을 Reverse Jacket 행렬이라 칭하였다. 개선된 리버스 자켓 행렬의 역행렬은 그 구조가 처음 행렬을 구성할 때와 동일한 구조를 갖는 장점을 갖고 있어 매우 쉽게 구할 수 있다[6]. 리버스 자켓 행렬을 분해 (decomposition)하여 개발한 고속 리버스 자켓 변환 (FRJT)은 중앙가중치 하다마드 변환 (Center-weighted Hadamard transform)[3] 보다 훨씬 더 빠른 알고리즘이다[6]. 4-점 리버스 자켓 변환 (4-point Reverse Jacket transform)의 장점 중의 하나는 4-점 fast Fourier transform (FFT)시 야기되는 실수 또는 복소수 곱셈을 행렬분해 (matrix decomposition)를 이용, 곱셈인자를 모두 대각행렬에만 집중시킨, 매우 간결하고 효율적인 알고리즘이라는 점이다. 본 논문에서는 [4, 5]에서 다루지 않았던 4-점 리버스 자켓 변환을 N-점 FFT에 적용하는 알고리즘을 제안한다.

본 논문은 다음과 같이 구성되었다. 제 2장에서는 개선된 리버스 자켓 행렬의 정의와 역행렬을 구하는 문제를 해결한 정리 그리고 고속 푸리에 변환과의

* 서울대학교 전기공학부 (srlee@acoustics.snu.ac.kr)
논문번호: K01020-0109, 접수일자: 2000년 1월 9일

관계를 나타내는 정리들을 학습하였다. 제3장에서는 이 행렬을 분해하여 얻은 고속 리버스 자켓 변환을 소개하며 제4장에서는 4-점 FRJT를 이용한 N-점 FFT에 적용하는 알고리즘을 제안한다. 끝으로, 제5장에서는 결론과 향후 전망에 대하여 논의한다.

Ⅱ. 리버스 자켓 행렬과 예비학습

이 장에서는 역행렬이 뒤집어지는 기하학적 구조를 갖는 리버스 자켓 행렬을 구성한다. 이 행렬은 하다마드 행렬의 요소들을 일반화한다. 안팍으로 뒤집어 입을 수 있는 등산복처럼 리버스 자켓의 역행렬을 구할 때 적어도 두 개의 행렬요소의 위치가 바뀐다. 예를 들어, 중앙 원으로부터 바깥쪽으로, 그리고 바깥쪽에서 안쪽으로 뒤바뀌는 형태가 된다. 이러한 기이한 현상으로 말미암아 이 행렬을 리버스 자켓 행렬이라 칭하였다[5].

주어진 2×2 기본행렬 $R_2 \in \square^{2\times 2}$에 대하여 가역 (invertible)인 4×4 행렬을 구성한다.

I_{2^k}와 O_{2^k}, $j \in \square \cup 0$은 각각 $(2^j \times 2^j)$ 단위행렬과 영행렬이며, □, □은 각각 실수와 자연수의 집합을 칭한다.

정의 1: [6] a, b, c, d를 영이 아닌 실수 또는 복소수라 하자. 리버스 자켓 행렬은 귀납적으로 (recursively) 다음과 같이 정의된다.

$$R_{2^{k+1}} = \begin{bmatrix} R_{2^k} & Z_{2^k} R_{2^k} S_{2^k} \\ S_{2^k} R_{2^k} Z_{2^k} & -J_{2^k} R_{2^k} J_{2^k} \end{bmatrix} \tag{2.1}$$

여기서 초기조건은 $1 \le k$ 에 대하여

$$R_2 := \begin{bmatrix} a & b \\ c & -d \end{bmatrix}, \quad Z_{2^k} := \begin{bmatrix} I_{2^{k-1}} & O_{2^{k-1}} \\ O_{2^{k-1}} & -I_{2^{k-1}} \end{bmatrix},$$

$$S_{2^k} := \begin{bmatrix} O_{2^{k-1}} & I_{2^{k-1}} \\ I_{2^{k-1}} & O_{2^{k-1}} \end{bmatrix} \quad \text{그리고} \quad J_{2^k} := \begin{bmatrix} O_{2^{k-1}} & I_{2^{k-1}} \\ -I_{2^{k-1}} & O_{2^{k-1}} \end{bmatrix}$$

이다. (2.2)

Remark 2: 우리는 여기서 $S_2S_2 = Z_2Z_2 = I_2$, $J_2S_2 = Z_2$ 그리고 $S_2J_2 = -Z_2$ 임을 쉽게 알 수 있다.

Remark 3: $A^{(k)}: R_{2k}$, $1 \le k$이라 하면 $1 \le k$에 대하여 다음 등식이 성립한다.

$$A^{k+1} := \left[\begin{array}{c|c} A^k & B^k \\ \hline C^k & -D^k \end{array}\right],$$

$$:= \left[\begin{array}{cc|cc} A^{(k-1)} & B^{(k-1)} & B^{(k-1)} & A^{(k-1)} \\ C^{(k-1)} & -D^{(k-1)} & D^{(k-1)} & -C^{(k-1)} \\ \hline C^{(k-1)} & D^{(k-1)} & -D^{(k-1)} & -C^{(k-1)} \\ A^{(k-1)} & -B^{(k-1)} & -B^{(k-1)} & A^{(k-1)} \end{array}\right] = R_{2^{k+1}} \tag{2.3}$$

여기서 초기조건은

$A^{(0)} = a$, $B^{(0)} = b$, $C^{(0)} = c$, $D^{(0)} = d$, 즉,

$A^{(1)} := \begin{bmatrix} a & b \\ c & -d \end{bmatrix}$, $B^{(k)} := Z_{2^k} A^{(k)} S_{2^k}$,

$C^{(k)} := S_{2^k} A^{(k)} Z_{2^k}$, 그리고 $D^{(k)} := -J_{2^k} A^{(k)} J_{2^k}$ 이다. (2.4)

정리 4: [6] R_2가 정칙행렬이라 하자. 리버스 자켓 행렬

$$R_{2^{k+1}} = \begin{bmatrix} R_{2^k} & Z_{2^k} R_{2^k} S_{2^k} \\ S_{2^k} R_{2^k} Z_{2^k} & -J_{2^k} R_{2^k} J_{2^k} \end{bmatrix} \tag{2.5}$$

이면 $R_{2^{k+1}}$는 정칙행렬이며

$$R^{-1}_{2^{k+1}} = \begin{bmatrix} \check{R}^{-1}_{2^k} & Z_{2^k} \check{R}^{-1}_{2^k} S_{2^k} \\ S_{2^k} \check{R}^{-1}_{2^k} Z_{2^k} & -J_{2^k} \check{R}^{-1}_{2^k} J_{2^k} \end{bmatrix} \tag{2.6}$$

이다. 이 때,

$$\check{R}_{2^k} = R_{2^k} + Z_{2^k} R_{2^k} S_{2^k} J_{2^k} R^{-1}_{2^k} J_{2^k} S_{2^k} R_{2^k} Z_{2^k}, \quad 1 \le k$$

정리 5: [6] R_2가 정칙행렬이라 하자. 다음 행렬

$$W_{2^{k+1}} = \begin{bmatrix} R_{2^k} & Z_{2^k} R_{2^k} \\ R_{2^k} Z_{2^k} & Z_{2^k} R_{2^k} Z_{2^k} \end{bmatrix} \tag{2.7}$$

이면 $W_{2^{k+1}}$는 정칙행렬이며

$$W^{-1}_{2^{k+1}} = \begin{bmatrix} \hat{R}^{-1}_{2^k} & Z_{2^k} \hat{R}^{-1}_{2^k} \\ \hat{R}^{-1}_{2^k} Z_{2^k} & Z_{2^k} \hat{R}^{-1}_{2^k} Z_{2^k} \end{bmatrix} \tag{2.8}$$

이다. 이 때,

$$\hat{R}_{2^k} = R_{2^k} - Z_{2^k} R_{2^k} Z_{2^k} R^{-1}_{2^k} Z_{2^k} R_{2^k} Z_{2^k}, \quad 1 \le k$$

따름정리 6: [6] $1 \le k$ 에 대하여 직교행렬 $U_{2^{k+1}}$ 이 존재하면

$$R_{2^{k+1}} = U_{2^{k+1}} W_{2^{k+1}} U_{2^{k+1}} \tag{2.9}$$

Ⅲ. 고속 리버스 자켓 변환

이 장에서는 고속 하다마드 변환이나 고속 중앙 가중치 하다마드 변환을 일반화한 알고리즘을 소개하기로 한다[6]. 이 알고리즘을 위해 우선 두 개의 치환행렬을 정의하자.

$$P_4 := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix},$$

$$P_{2^k} := P_4 \otimes I_{2^{k-2}},\ 2 \le k,$$

$$Q_{2^k} := [I_{2^{k-1}} \oplus (S_2 \otimes I_{2^{k-2}})] \tag{3.1}$$

여기서 $\otimes$는 Kronecker product를 $\oplus$는 direct sum을 각각 나타낸다.

$\otimes$와 $\oplus$등을 이용한 행렬분해를 통해서 개발한 고속 리버스 자켓 변환 (FRJT) 알고리즘은 다음과 같은 수식으로 표현된다.

$$R_{2^k} = P_{2^k}^T \underbrace{(H_{2^{k-1}} \oplus H_{2^{k-1}})}_{\text{스테이지 3}} \underbrace{((U_A \otimes I_{2^{k-3}}) \oplus (U_B \otimes I_{2^{k-3}}))}_{\text{스테이지 2}} \underbrace{(H_2 \otimes I_{2^{k-1}})}_{\text{스테이지 1}} Q_{2^k}^T \tag{3.2}$$

여기서 $U_A := \mathrm{diag}(a,b)$ 이고 $U_B := \mathrm{diag}(c,d)$이다.

[6]에서 밝혔듯 $a=b=c=1$ 일 때 $N \log_2 N$ 덧셈과 $\frac{N}{4}$의 곱셈이 소요된다.

Ⅳ. 4-점 FRJT를 이용한 N-점 FFT의 구성

N-점 고속 푸리에 변환은 입력 데이터의 집합 x(0),x(1), ... ,x(N-1) 에 대한 푸리에 계수 X(0),X(1), ... ,X(N-1) 와의 관계

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ \vdots \\ X(N-1) \end{bmatrix} = \begin{bmatrix} f_N^0 & f_N^0 & f_N^0 & \cdots & f_N^0 \\ f_N^0 & f_N^1 & f_N^2 & \cdots & f_N^{N-1} \\ f_N^0 & f_N^2 & f_N^4 & \cdots & f_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ f_N^0 & f_N^{N-1} & f_N^{2(N-1)} & \cdots & f_N^{(N-1)(N-1)} \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ \vdots \\ x(N-1) \end{bmatrix} \tag{4.1}$$

로 나타낸다. 여기서 x(n)은 신호열이고

$$f_N = e^{-j\frac{2\pi}{N}} = \cos(2\pi/N) - j \cdot \sin(2\pi/N)$$

매개변수 f_n 은 $f_n^n=1$ 이므로 n차 근이다. 식 (4.1)은

$$X = F_N \cdot x \tag{4.2}$$

로 나타낼 수 있다. 여기서

$X = [X(0), X(1), \ldots X(N-1)]^T$ 이고

$x = [x(0), x(1), \ldots, x(N-1)]^T$ 이다. (4.2)의 역변환은

$$x = \frac{1}{N} F_N^* \cdot X \tag{4.3}$$

이며 이 때 *는 복소켤레를 나타내며 F_N은 FFT 행렬이다.

따름 정리 6은 리버스 자켓 행렬과 4-점 FFT 행렬과의 관계를 보여 주었다. 이제 우리는 4-점 FFT를 어떻게 FRJT로 전환하여 연산할 수 있는지를 다음의 예제를 통해 살펴보자.

예제 7:

$$W_4 := \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

는 널리 알려진 4-점 FFT 행렬이다.

$$R_2 := \begin{bmatrix} 1 & 1 \\ 1 & -j \end{bmatrix}, \qquad U_4 := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

는 각각 기본행렬과 리버스 자켓 행렬을 위한 치환 행렬이다. 따름정리 6에 의해 $R_4 = U_4 W_4 U_4$ 이고 $W_4 = U_4 R_4 U_4$ 이다. 따라서 FRJT 알고리즘을 적용하고 $T_4 := U_4 P_4$, $Q_4^T U_4 = (U_4)^2 = I_4$ 이면 W_4와 R_4를 다음과 같이 나타낸다.

$$\begin{aligned} W_4 &= U_4 P_4^T (H_2 \oplus H_2)(U_A \oplus U_B)(H_2 \otimes I_2) Q_4^T U_4 \\ &= T_4 (H_2 \oplus H_2)(U_A \oplus U_B)(H_2 \otimes I_2) \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & j \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \end{aligned}$$

$$R_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & j & -1 \\ 1 & j & -j & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

이 예제의 각 스테이지에서의 총 연산량은 a=b=c =1이고 d=j∈□ 일 때 8번의 덧셈과 한번의 복소수 곱셈이 소요된다. N=8 일 경우에는 매개변수의 주기성과 대칭성을 고려하면

$f_4^1 = f_8^2 = -j$, $f_4^2 = f_8^4 = -1$ 이며,

$$F_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & f_8^1 & f_8^2 & f_8^3 & f_8^4 & f_8^5 & f_8^6 & f_8^7 \\ 1 & f_8^2 & f_8^4 & f_8^6 & 1 & f_8^2 & f_8^4 & f_8^6 \\ 1 & f_8^3 & f_8^6 & f_8^1 & f_8^4 & f_8^7 & f_8^2 & f_8^5 \\ 1 & f_8^4 & 1 & f_8^4 & 1 & f_8^4 & 1 & f_8^4 \\ 1 & f_8^5 & f_8^2 & f_8^7 & f_8^4 & f_8^1 & f_8^6 & f_8^3 \\ 1 & f_8^6 & f_8^4 & f_8^2 & 1 & f_8^6 & f_8^4 & f_8^2 \\ 1 & f_8^7 & f_8^6 & f_8^5 & f_8^4 & f_8^3 & f_8^2 & f_8^1 \end{bmatrix} \quad (4.4)$$

인덱스 벡터[0 2 4 6 1 3 5 7]를 치환행렬 M_8로 표시하면

$$\tilde{F}_8 := F_8 \square M_8 = \left[\begin{array}{cccc|cccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & f_8^2 & f_8^4 & f_8^6 & f_8^1 & f_8^3 & f_8^5 & f_8^7 \\ 1 & f_8^4 & 1 & f_8^4 & f_8^2 & f_8^6 & f_8^2 & f_8^6 \\ 1 & f_8^6 & f_8^4 & f_8^2 & f_8^3 & f_8^1 & f_8^7 & f_8^5 \\ \hline 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & f_8^2 & f_8^4 & f_8^6 & -f_8^1 & -f_8^3 & -f_8^5 & -f_8^7 \\ 1 & f_8^6 & f_8^4 & f_8^2 & -f_8^2 & -f_8^6 & -f_8^2 & -f_8^6 \\ 1 & f_8^7 & f_8^6 & f_8^5 & -f_8^3 & -f_8^1 & -f_8^7 & -f_8^5 \end{array}\right] \quad (4.5)$$

이 된다.

각 블록이 4×4 행렬로 이루어진 2×2 행렬 $\tilde{F}_8$ 을 생각해보자. $f_8^2 = f_4^1$ 임을 주목하면

$$\tilde{F}_8 = \begin{bmatrix} F_4 & \Lambda F_4 \\ F_4 & -\Lambda F_4 \end{bmatrix} \quad (4.6)$$

이 성립함을 곧 알 수 있다. 여기서

$\Lambda_4 := \mathrm{diag}(1, f_8^1, f_8^2, f_8^3)$ 이다.

I_4 와 O_4 가 각각 4×4 단위행렬과 영행렬이라면 다음등식이 성립한다.

$$\tilde{F}_8 = \begin{bmatrix} F_4 & \Lambda_4 F_4 \\ F_4 & -\Lambda_4 F_4 \end{bmatrix} = \begin{bmatrix} I_4 & \Lambda_4 \\ I_4 & -\Lambda_4 \end{bmatrix} \begin{bmatrix} F_4 & O_4 \\ O_4 & F_4 \end{bmatrix} = \begin{bmatrix} I_4 & I_4 \\ I_4 & -I_4 \end{bmatrix} \begin{bmatrix} F_4 & O_4 \\ O_4 & \Lambda_4 F_4 \end{bmatrix}$$

$$= (H_2 \otimes I_4) \begin{bmatrix} I_4 & O_4 \\ O_4 & \Lambda_4 \end{bmatrix} \begin{bmatrix} F_4 & O_4 \\ O_4 & F_4 \end{bmatrix} = (H_2 \otimes I_4)(I_4 \oplus \Lambda_4)(I_2 \otimes F_4) \quad (4.7)$$

F_4를 $U_4 R_4 U_4$ 로 치환하고 (4.7)을 적용하면 등식

$$\tilde{F}_8 = (H_2 \otimes I_4)(I_4 \oplus \Lambda_4)(I_2 \otimes F_4)$$
$$= (H_2 \otimes I_4)(I_4 \oplus \Lambda_4)(I_2 \otimes (U_4 R_4 U_4)) \quad (4.8)$$

를 얻는다. M_8이 치환행렬이므로 $M_8^T = M_8^{-1}$ 이다. 따라서

$$F_8 = \tilde{F}_8 M_8^T = \underbrace{(H_2 \otimes I_4)}_{\text{8번 덧셈}} \underbrace{(I_4 \oplus \Lambda_4)}_{\text{3번 곱셈}} \underbrace{(I_2 \otimes (U_4 R_4 U_4))}_{\text{2x(8번 덧셈+한번 곱셈)}} M_8^T \quad (4.9)$$

유사한 방법으로 N 점 FFT 행렬 F_N 을 다음과 같이 표현할 수 있다.

$$F_N = \tilde{F}_N M_N^T = \underbrace{(H_2 \otimes I_{N/2})}_{\text{N번 덧셈}} \underbrace{(I_{N/2} \oplus \Lambda_{N/2})}_{\frac{N}{2}-1\text{번 곱셈}} \underbrace{(I_2 \otimes F_{N/2})}_{2x(F_{N/2}\text{의 연산량})} M_N^T \quad (4.10)$$

여기서 $\Lambda_{N/2} := \mathrm{diag}(1, f_N^1, f_N^2, \ldots, f_N^{(N/2)-1})$ 이며 치환행렬에 M_N 상응하는 인덱스 벡터는 [0 2 ... N-2 1 3 ... N-1] 이다. 이 방법은 (4.10)의 두번째 스테이지가 항상 대각행렬이고 다른 스테이지에서는 단지 Kronecker product 만을 사용하였기 때문에 [2]에 있는 방법보다 더 명백한 행렬분해를 가지고 있으며 확장하거나 구조를 이해하기에 용이하다.

V. 결 론

더욱 일반화된 리버스 자켓 행렬의 정의와 정리들을 학습하였다. 고속 리버스 자켓 변환 알고리즘을 소개하였고 아울러 4-점 고속 푸리에 변환과의 관계를 나타내는 정리와 예제를 보여주었다. 4-점 리버스 자켓 변환 (4-point Reverse Jacket transform)의 장점 중의 하나는 4-점 fast Fourier transform (FFT)시 야기되는 실수 또는 복소수 곱셈을 행렬분해 (matrix decomposition)를 이용, 곱셈

인자를 모두 대각행렬에만 집중시킨, 매우 간결하고 효율적인 알고리즘이라는 점이다. 본 논문에서는 4-점 리버스 자켓 변환을 N-점 FFT에 적용하는 알고리즘을 제안하였다. 이 방법은 기존의 다른 변환형태보다 확장하거나 구조를 파악하기에 매우 용이하다. 향후 N-점 FRJT와 N-점 FFT와의 관계를 밝히는 연구과제가 남아있다.

참 고 문 헌

[1] O. ERSOY, Fourier-Related Transforms, *Fast Algorithms and Applications*, Prentice Hall International Editions, 1997.

[2] G. H. GOLUB AND C. VAN LOAN, Matrix Computations, Third Edition, The Johns Hopkins University Press 36, No. 9, pp. 1247-1249, 1996.

[3] M. H. LEE, *The Center-weighted Hadamard Transform*, IEEE Trans. on Circuits and Systems-II, Vol. 36, No. 9, pp. 1247-1249, 1989.

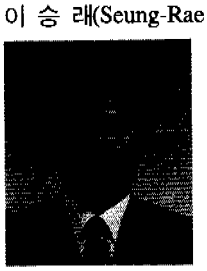
[4] M. H. LEE, A New Reverse Jacket Transform and Its Fast Algorithm, IEEE Trans. on Circuits and Systems-II, Vol. 47, No. 1, pp. 39-47, 2000.

[5] S. -R. LEE AND M. H. LEE, *On the Reverse Jacket Matrix for Weighted Hadamard Transform*, IEEE Trans. on Circuits and Systems-II, Vol. 45, No. 3, pp. 436-441, 1998.

[6] S.-R. LEE AND K.-M. SUNG, *고속 리버스 자켓 변환과 그의 응용*, 한국통신학회논문지에 제출, 2001.

[7] R. K. R. YARLAGADDA AND J. E. HERSHEY, *Hadamard Matrix Analysis and Synthesis*, Kluwer Academic Publishers, 1997.

이 승 래(Seung-Rae Lee) 정회원



1991년 : 독일 아헨공대 수학과 학사 (VorDiplom)

1993년 : 독일 아헨공대 수학과 석사 (Diplom Mathematiker)

1997년 : 독일 아헨공대 수학과 이학박사 (Dr. rer. Nat.)

1995년 7월~1996년 12월 : 독일 두이스부르그 대학 수학과 강사

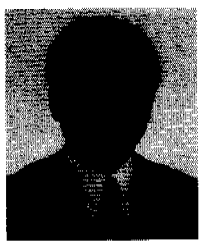
1998년 9월~2000년 3월 : 서울대학교 제어계측신기술연구센터 Post Doc. 연구원

2000년 4월~2000년 11월 : 성균관대학교 전기전자 및 컴퓨터공학부 연구조교수

2000년 12월~현재 : 서울대학교 음향공학연구실 BK21 Post Doc. 연구원

<주관심 분야> 미분게임, 최적제어이론, 신호처리, 부호이론, 음향공학

성 굉 모(Koeng-Mo Sung) 정회원



1971년 : 서울대학교 공과대학 전자공학과 학사

1977년 : 독일 아헨공대 전자공학과 석사 (Diplom Ingenieur)

1982년 : 독일 아헨공대 전자공학과 공학박사 (Dr. Ing.)

1983년 7월~현재 : 서울대학교 전기공학부 교수

1997년 7월~1998년 9월 : 뉴미디어 통신공동연구소장

1998년 9월~2000년 9월 : 서울대학교 전기공학부 학부장

2000년 1월~현재 : 한국음향학회 회장

<주관심 분야> ultrasonics, musical acoustics, underwater acoustics, electro-acoustics, speech recognition and synthesis