FIR ADAPTIVE FILTERS BASED ON THE HIRSCHMAN OPTIMAL TRANSFORM

Osama Alkhouli, Victor DeBrunner, Yan Zhai, and Mark Yeary
School of Electrical and Computer Engineering
The University of Oklahoma
{osama_k,vdebrunn}@ou.edu, {yan zhai,yeary}@ieee.org

ABSTRACT

In this paper, we derive a "convolution theorem" suitable for the Hirschman optimal transform (HOT), a unitary transform derived from a discrete-time, discrete-frequency version of the entropy-based uncertainty measure first described by Hirschman. We use the result to develop transform domain adaptive filters. First, we show how our method can be used to implement a fast block-LMS adaptive filter that we call the HOT block-LMS adaptive filter. This filter requires slightly less than half of the computations that are required in an FFT-based block-LMS adaptive filter. We also develop another transform-based adaptive filter algorithm that uses a sliding window instead of a block of data. The HOT version of these sliding algorithms is also significantly computationally more efficient (by $\sqrt{N}$, where N is the filter order) than the sliding DFT version. Because our work is at an early stage, we develop simulations that explore basic convergence characteristics.

1. INTRODUCTION

The HOT is a recently developed discrete unitary transform that uses the orthonormal minimizers of the entropy-based Hirschman uncertainty measure [1]. This measure is different from the energy-based Heisenberg uncertainty measure that is only suited for continuous time signals. The Hirschman uncertainty measure uses entropy to quantify the spread of discrete-time signals in time and frequency [2]. Since the HOT bases are among the minimizers of the uncertainty measure, they have the novel property of being the most compact in discrete time and frequency. The fact that the HOT basis sequences have many zero-valued samples, and their resemblance to the DFT basis sequences, makes the HOT computationally attractive. Furthermore, it has been shown recently that a thresholding algorithm using the HOT yields superior frequency resolution of a pure tone in additive white noise to a similar algorithm based on the DFT [3].

The main theorem in [1] describes a method to generate an $N = K^2$-point orthonormal HOT basis, where K is an integer. A HOT basis sequence of length K^2 is the most compact bases in the time-frequency plane. The 3^2-point HOT matrix is

$$\begin{bmatrix} I_3 & I_3 & I_3 \\ I_3 & e^{-j\frac{2\pi}{3}} I_3 & e^{-j\frac{4\pi}{3}} I_3 \\ I_3 & e^{-j\frac{4\pi}{3}} I_3 & e^{-j\frac{8\pi}{3}} I_3 \end{bmatrix} \quad (1)$$

Eq. (1) indicates that that the HOT of any sequence is related to the DFT of some polyphase components of the signal. In fact, we called this property the "1 and ½ dimensionality" of the HOT in [2]. Consequently, for this paper, we will use the terms HOT and DFT of the polyphase components interchangeably. The K^2-point HOT requires fewer computations than does the K^2-point DFT. We used this computational efficiency of the HOT to implement fast convolution algorithms in [4]. When K is a power-of-2 integer, then $K^2 \log_2 K$ (complex) multiplications are needed to compute the HOT, which is half that required when computing the DFT.

In this paper, we use the computational efficiency of the HOT to implement a fast block-LMS adaptive filter. The fast block-LMS adaptive filter was first proposed [5] to reduce computational complexity. Our proposed HOT block-LMS adaptive filter requires less than half of the computations required in the corresponding FFT block-LMS adaptive filter. This significant complexity reduction could be important in many real time applications. We also present a self-orthogonalizing adaptive filter (as in [6]) that uses the sliding window concept with the HOT, which we call the "sliding HOT-LMS." We build a transform domain adaptive filter in this case that is extremely efficient, requiring only $1/K$ as many computations as the corresponding FFT-based sliding window algorithm. Of course, in both cases, the complexity reduction comes at some performance cost.

In Section 2, we develop the notion of convolution with the HOT. In Section 3, we develop the HOT block-LMS algorithm. In Section 4, we develop the sliding HOT-LMS algorithm. Simulations are provided in Section 5. There we also examine the performance and costs in some detail. Finally, we conclude.

2. HOT AND CIRCULAR CONVOLUTION

Suppose that *u* and *h* are two signals of length $N = K^2$ and the signal y is their circular convolution. Then, in the frequency domain we can write $Y(k) = U(k)H(k)$. To replace the DFT with the HOT we need the relationship between the DCT and HOT of the signals *u*, *h*, and *y*. Define $u_i(l) = u(lK + i)$, where *i,l = 0, 1, 2, …… , K-1*. Then

$$U(k) = \sum_{i=0}^{K-1} e^{-j\frac{2\pi}{K^2}ki} U_i(k), \quad (2)$$

where $U_i(k)$ is the DFT of $u_i(l)$. In matrix form, Eq. (2) can be written

$$U = \sum_{i=0}^{K-1} D_{0,K^2-1}(i) \begin{bmatrix} F_K \\ \vdots \\ F_K \end{bmatrix} u_i, \tag{3}$$

where $D_{i,l}(k) = \mathrm{diag}\left\{ e^{-j\frac{2\pi}{K^2}ki}, \ldots, e^{-j\frac{2\pi}{K^2}kl} \right\}$ and F_K is the K-point DFT matrix. Eq. (3) relates the DFT of the signal u to its HOT. Therefore, circular convolution in the HOT domain is given by

$$\begin{aligned} &\sum_{i=0}^{K-1} D_{0,K^2-1}(i) \begin{bmatrix} F_K \\ \vdots \\ F_K \end{bmatrix} y_i \\ &= \left(\sum_{i=0}^{K-1} D_{0,K^2-1}(i) \begin{bmatrix} F_K \\ \vdots \\ F_K \end{bmatrix} h_i \right) \bullet \left(\sum_{i=0}^{K-1} D_{0,K^2-1}(i) \begin{bmatrix} F_K \\ \vdots \\ F_K \end{bmatrix} u_i \right) \\ &= \sum_{i=0}^{K-1} \sum_{j=0}^{K-1} D_{0,K^2-1}(i+j) \left(\begin{bmatrix} F_K \\ \vdots \\ F_K \end{bmatrix} h_i \right) \bullet \left(\begin{bmatrix} F_K \\ \vdots \\ F_K \end{bmatrix} u_j \right). \end{aligned} \tag{4}$$

The $\cdot$ means element-wise multiplication. Eq (4) is not in the form that we require. To find the j^{th} polyphase component of the output $F_K y_j$, we divide it into the following K equations:

$$\begin{aligned} &\sum_{i=0}^{K-1} D_{rK,(r+1)K-1}(i) F_K y_i \\ &= \sum_{i=0}^{K-1} \sum_{j=0}^{K-1} D_{rK,(r+1)K-1}(i+j) F_K h_i \bullet F_K u_j, \end{aligned} \tag{5}$$

where $r = 0, 1, 2, \ldots, K-1$. But since

$$D_{rK,(r+1)K-1}(i) = e^{-j\frac{2\pi}{K}ri} D_{0,K-1}(i) \tag{6}$$

Eq. (5) simplifies to

$$\begin{aligned} &\sum_{i=0}^{K-1} e^{-j\frac{2\pi}{K}ri} D_{0,K-1}(i) F_K y_i \\ &= \sum_{i=0}^{K-1} \sum_{j=0}^{K-1} e^{-j\frac{2\pi}{K}r(i+j)} D_{0,K-1}(i+j) F_K h_i \bullet F_K u_j \end{aligned} \tag{7}$$

Because the DFT matrix is unitary we have

$$\begin{aligned} &F_K y_s \\ &= \sum_{i=0}^{K-1} \sum_{j=0}^{K-1} \delta(i+j-s) D_{0,K-1}(i+j-s) F_K h_i \bullet F_K u_j \end{aligned} \tag{8}$$

Let H_i be the diagonal matrix corresponding to $F_K h_i$. Then the element-wise multiplication in Eq. (8) can be replaced with conventional matrix multiplication. Finally we combine the K equations in (8)

$$\begin{bmatrix} F_K y_0 \\ F_K y_1 \\ F_K y_2 \\ \vdots \\ F_K y_{K-2} \\ F_K y_{K-1} \end{bmatrix} = \begin{bmatrix} H_0 & DH_{K-1} & DH_{K-2} & \cdots & DH_2 & DH_1 \\ H_1 & H_0 & DH_{K-1} & \cdots & DH_3 & DH_2 \\ H_2 & H_1 & H_0 & \cdots & DH_4 & DH_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ H_{K-2} & H_{K-3} & H_{K-4} & \cdots & H_0 & DH_{K-1} \\ H_{K-1} & H_{K-2} & H_{K-3} & \cdots & H_1 & H_0 \end{bmatrix} \begin{bmatrix} F_K u_0 \\ F_K u_1 \\ F_K u_2 \\ \vdots \\ F_K u_{K-2} \\ F_K u_{K-1} \end{bmatrix}, \tag{9}$$

where $D = D_{0,K-1}(K)$. From Eq. (9) one can see that calculating the HOT of the output y requires $K^3 + 2K^2 \log_2 K + K^2 - K$ multiplications. When we require only one polyphase component, then only $K^2 + 2K^2 \log_2 K + K \log_2 K$ multiplications are necessary. By way of comparison, if the DFT is used to calculate the output then $K^2 + 6K^2 \log_2 K$ multiplications are required. Asymptotically in K, we see that the HOT could be three times more efficient than the DFT.

3. THE HOT BLOCK-LMS ADAPTIVE FILTER

In a block adaptive filter, the adaptation proceeds block-by-block, with the weight update equation

$$w_{k+1} = w_k + \frac{\mu}{L} \sum_{s=0}^{L-1} e(kL+s) u(kL+s), \tag{10}$$

where d and y are the desired and output signals, respectively, u is the vector that contains the input samples in the k[th] block, L is the block size or the filter length, and the error is $e(n) = d(n) - y(n)$. The FFT is commonly used to efficiently calculate the output of the filter and the sum in the update equation. For our proposed HOT block-LMS algorithm, we use the update equation:

$$w_{k+1} = w_k + \mu \frac{2}{K} \sum_{i=0}^{\frac{K}{2}-1} e(k\frac{K^2}{2} + s) u(k\frac{K^2}{2} + s), \tag{11}$$

and $s = iK + j$. We assume that the block size or the filter length is $K^2/2$ (our reason for this assumption will become clear) and the block size is equal to the filter order. The parameter j determines which polyphase component of the error signal is being used in the adaptation. This parameter can be changed from block to block. Therefore, in the adaptation we are only using one polyphase component, rather than the whole signal. The output and the sum in this update equation are efficiently calculated:

a) Append the weight vectors with $K^2/2$ zeros (the resulting vector is now K^2 points long as required in the HOT definition) and find its HOT

b) Compute the HOT of the input vector $[u((k-1)\frac{K^2}{2}),\ldots,u(k\frac{K^2}{2}),\ldots,(u(k\frac{K^2}{2}+\frac{K^2}{2}-1)]^t$. Note that this vector contains the input samples for the current and previous block.

c) Use the inverse HOT and Eq. (9) to calculate the j[th] polyphase component of the circular convolution. The j[th] polyphase component of the output can be found by discarding the first half of the j[th] polyphase component of the circular convolution.

d) Calculate the j[th] polyphase component of the error, insert a block of $K/2$ zeros, up-sample by K, then calculate its HOT.

e) Circularly flip the vector in b) and then compute its HOT.

f) Compute the sum in the update equation using Eq. (9) – this sum is that of the first half elements of the circular convolution between the vectors in part e) and d).

Now we look at the computational cost of the algorithm and compare it to that of the FFT block adaptive algorithm. Parts a), b), and e) require $3K^2\log_2 K$ multiplications, part c) requires $K\log_2 K + K^2$, part d) requires $K\log_2 K$, and part f) requires $K^2 + K^2\log_2 K$. The total number of multiplications is thus $4K^2\log_2 K + 2K\log_2 K + 2K^2$. The corresponding FFT block adaptive algorithm requires $10K^2\log_2 K + 2K^2$ multiplications – asymptotically more than twice as many. Therefore, by using only one polyphase component for the adaptation in a block, the computational cost can be reduced by a factor of 2.5. While this complexity reduction comes at the cost of not using all available information, our proposed algorithm provides better estimates than does the LMS filter. The reduction of the computational complexity in our algorithm comes from using the polyphase components of the input signal to calculate one polyphase component of the output and using HOT. It is worth mentioning that the fast exact LMS adaptive algorithm (FELMS) [7] also reduces the computational complexities by finding the output via processing the polyphase components of the input. However, the computational complexity reduction of the FELMS is less than that found in the FFT and HOT block adaptive algorithms because the FELMS is designed to have exact mathematical equivalence to, and hence the same convergence properties as, the conventional LMS.

4. SLIDING HOT AND HOT-LMS ADAPTIVE FILTERS

In this section we propose the transform domain LMS adaptive filter HOT-LMS that is significantly more efficient than the analogous DFT-LMS algorithm. The sliding DFT of the signal u with a K^2-long sliding window is [8]

$$U_k(n) = e^{j\frac{2\pi}{K^2}k}U_k(n-1) + u(n) - u(n-K^2). \qquad (12)$$

This recursion requires K^2 multiplications to find the updates for all of the K^2 DFT points. If we redefine the HOT using two indices

$$U_{k,l}(n) = \sum_{i=0}^{K-1} U(n-iK-l)e^{-j\frac{2\pi}{K}ki} \qquad (13)$$

where $k,l = 0,1,2,\ldots,K-1$, then the sliding HOT is

$$\begin{aligned} U_{k,0}(n) &= e^{j\frac{2\pi}{K}k}U_{k,K-1}(n-1) + u(n) - u(n-K^2) \\ U_{k,1}(n) &= U_{k,0}(n-1) \\ &\vdots \\ U_{k,K-1}(n) &= U_{k,K-2}(n-1) \end{aligned} \qquad (14)$$

Since the sliding HOT requires only K multiplications, it is K times more efficient than the sliding DFT that requires $N = K^2$ multiplications! The transform domain adaptive filer was proposed to increase the convergence speed of adaptive filters [9]. The transform domain adaptive filter uses an orthogonal basis as an approximation to the data dependent Karhunen-Loève transform (KLT) to decompose the incoming signal into a set of partially uncorrelated components. The following theorem provides the second order statistics whereby the HOT is optimal in this sense:

Theorem 1: let $\{u_o(n), u_1(n),\ldots, u_{K-1}(n)\}$ be a set of K uncorrelated wide-sense stationary (WSS) periodic, with periodicity K, random processes with the corresponding set of autocorrelation sequences $\{r_o(n), r_1(n),\ldots, r_{K-1}(n)\}$. Then the random process

$$u(Kr+l) = u_l(r) \qquad (15)$$

has its $K^2 x\ K^2$ autocorrelation matrix diagonalizable by the HOT. (A proof is included in the appendix). Note that the random process in (15) is in general not stationary. Using this theorem, we propose the HOT-LMS

$$\begin{aligned} y(n) &= \sum_{k=0}^{K-1}\sum_{l=0}^{K-1} U_{k,l}(n)W_{k,l}(n) \\ e(n) &= d(n) - y(n) \\ \lambda_{k,l}(n) &= \lambda_{k,l}(n) + \frac{1}{n}(|U_{k,l}(n)|^2 - \lambda_{k,l}(n)) \\ W_{k,l}(n+1) &= W_{k,l}(n) + \frac{\mu}{\lambda_{k,l}(n)}e(n)U^*_{k,l}(n) \end{aligned} \qquad (16)$$

Although in most applications the random processes are not periodic, the DFT is asymptotically optimal as the order of the filter increases [10], and hence the HOT optimality follows when the process consists of uncorrelated sub-processes.

5. SIMULATIONS

Our simulations take the form of a system identification experiment. Three input types are generated. The first type is white noise. The second type is generated by coloring the white noise with a high-pass FIR filter of order 31. The third input type is generated by polyphasing three uncorrelated signals of the second type. To generate the simulated signals, we further add measurement noise with power -40 dB to the outputs corresponding to the (noise-free) three different inputs types. The identified system is assumed to be 9[th] order.

The HOT and FFT block algorithms are simulated using the first input type – white noise. The learning curves are shown in Fig. 1 with the learning curve of the conventional LMS. The step sizes in Eqs. (10), (11), and the conventional algorithm are chosen to be the same. The higher mean square error of the HOT algorithm, compared to FFT algorithm, shows the trade-off for complexity reduction by more than half. The similar convergence characteristics of the HOT and FFT algorithms is expected because we use the same sliding block in the implementations. The learning curves of the sliding window algorithms are shown in Figs. 2 and 3. As expected the HOT-LMS and the other transform domain LMS methods converge faster with less mean square error than the LMS due to their ability to decorrelate the samples at the input of the filter. While the performances of the three transform algorithms are very similar in this example, our algorithm has a complexity cost reduction of K .

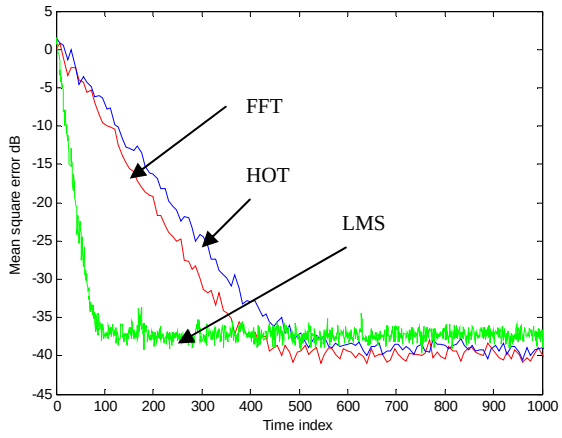

Figure 1. The learning curves of the block algorithms with the conventional LMS. The input is type one.

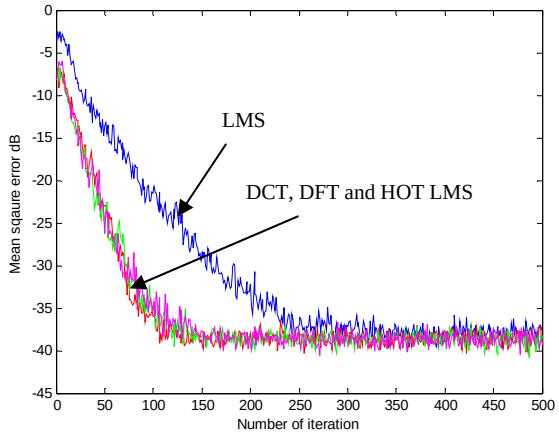

Figure 2. The learning curves of the sliding window algorithms. The input signal is type three.

5. CONCLUSIONS

This paper has proposed two computationally efficient transform domain adaptive filters – the HOT block-LMS and HOT sliding window adaptive filters. In addition to their efficiency, our preliminary simulations show that the performance of the HOT-based algorithms is competitive with those of the corresponding DCT and DFT algorithms. Our next goal is to quantitatively determine the performances of these HOT-based adaptive filters.

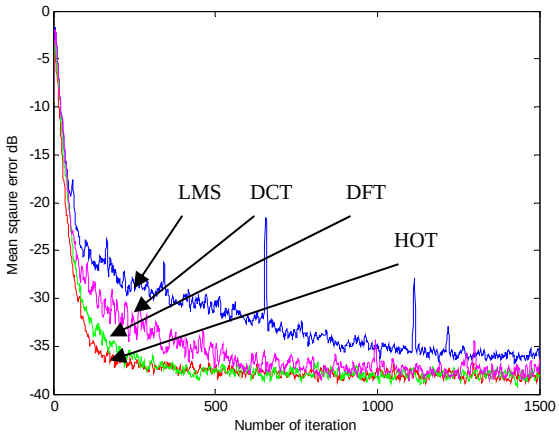

Figure 3. The learning curves of the sliding window algorithms. The input signal is type two.

6. REFERENCES

[1] T. Przebinda, V. DeBrunner, and M. Özaydın, "The optimal transform for the discrete Hirschman uncertainty principle," *IEEE Trans. Infor. Theory*, pp. 2086-2090, Jul 2001.

[2] V. DeBrunner, M. Özaydın, and T. Przebinda, "Resolution in time-frequency," *IEEE Trans. SP*, pp. 783-788, Mar 1999.

[3] V. E. DeBrunner, J. P. Havlicek, T. Przebinda, and M. Özaydın, "Entropy-based uncertainty measures for $L^2\left(\mathbb{R}^n\right)$, $\ell^2\left(\mathbb{Z}\right)$, and $\ell^2\left(\mathbb{Z}/N\mathbb{Z}\right)$ with a Hirschman optimal transform for $\ell^2\left(\mathbb{Z}/N\mathbb{Z}\right)$," *IEEE Trans. SP*, scheduled to appear Aug 2005.

[4] V. DeBrunner and E. Matusiak, "An algorithm to reduce the complexity required to convolve finite length sequences using the Hirschman optimal transform (HOT)," *ICASSP'03*, Hong Kong, China, pp. II-577-580, Apr 2003.

[5] G. Clark, S. Mitra, and S. Parker, "Block implementation of adaptive digital filters," *IEEE Trans. ASSP*, pp. 744-752, Jun 1981.

[6] D. F. Marshall, W. K. Jenkins, and J. J. Murphy, "The use of orthogonal transforms for improving performance of adaptive filters", *IEEE Trans. Cir.&Sys.*, pp. 474-484, Apr 1989.

[7] J. Benesty and P. Duhamel, "A fast exact least mean square adaptive algorithm," *IEEE Trans. SP*, pp. 2904-2920, Dec 1992.

[8] E. Jacobsen and R. Lyons, "The sliding DFT," *IEEE SP Mag.*, pp. 74-80, Mar 2003.

[9] S. Narayan, A. Peterson, and M. Narasimha, "Transform domain LMS algorithm," *IEEE Trans. ASSP*, pp. 609-615, Jun 1983.

[10] J. Pearl, "On coding and filtering stationary signals by discrete Fourier transforms," *IEEE Trans. Infor. Theory*, pp. 229-232, Mar 1973.

APPENDIX

Proof of Theorem 1: To simplify the notation we prove the theorem for $K=3$. Generalizing to any other value of K is straight forward. The 32x32 autocorrelation matrix R of the random process u can be shown to have the form

$$R=\begin{bmatrix} r_0(0) & 0 & 0 & r_0(2) & 0 & 0 & r_0(1) & 0 & 0 \\ 0 & r_1(0) & 0 & 0 & r_1(2) & 0 & 0 & r_1(1) & 0 \\ 0 & 0 & r_2(0) & 0 & 0 & r_2(2) & 0 & 0 & r_2(1) \\ r_0(1) & 0 & 0 & r_0(0) & 0 & 0 & r_0(2) & 0 & 0 \\ 0 & r_1(1) & 0 & 0 & r_1(0) & 0 & 0 & r_1(2) & 0 \\ 0 & 0 & r_2(1) & 0 & 0 & r_2(0) & 0 & 0 & r_2(2) \\ r_0(2) & 0 & 0 & r_0(1) & 0 & 0 & r_0(0) & 0 & 0 \\ 0 & r_1(2) & 0 & 0 & r_1(1) & 0 & 0 & r_1(0) & 0 \\ 0 & 0 & r_2(2) & 0 & 0 & r_2(1) & 0 & 0 & r_2(0) \end{bmatrix} \tag{17}$$

The zeros in R correspond to the vanishing cross-correlations between the sub-processes. To write the autocorrelation matrix R in terms of the 3x3 autocorrelation matrices R_1, R_2, R_3 of the sub processes u_1, u_2, u_3 we define the following matrix

$$T=\begin{bmatrix} J_{11} & J_{21} & J_{31} \\ J_{12} & J_{22} & J_{32} \\ J_{13} & J_{23} & J_{33} \end{bmatrix} \tag{18}$$

where J_{ab} is a 3x3 matrix where all of the entries are zeros except the $(J_{ab})_{ab}$ which are equal to one. The matrix T rearranges the elements of R into blocks where each block is the autocorrelation matrix of one of the sub-processes, *i.e.*

$$R=T\tilde{R}T \tag{19}$$

where

$$\tilde{R}=\begin{bmatrix} R_1 & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & R_2 & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & R_3 \end{bmatrix} \tag{20}$$

The matrix 0_{3x3} is a 3x3 matrix of zero entries. To show that R is diagonalizable by the 3^2-point HOT matrix H, see Eq. (17), we only need to show that $H^H RH$ or $TH^H RHT$ is in fact a diagonal matrix. Note that we can write

$$TH^H RHT = TH^H T\tilde{R}THT = (THT)^H \tilde{R}(THT) \tag{21}$$

The matrix T also rearranges H into a block diagonal matrix where each block is the 3-point DFT matrix. Therefore,

$$THT=\begin{bmatrix} F_3 & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & F_3 & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & F_3 \end{bmatrix} \tag{22}$$

Hence,

$$\begin{aligned} & TH^H RHT \\ &= \begin{bmatrix} F_3^H & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & F_3^H & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & F_3^H \end{bmatrix} \begin{bmatrix} R_1 & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & R_2 & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & R_3 \end{bmatrix} \begin{bmatrix} F_3 & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & F_3 & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & F_3 \end{bmatrix} \\ &= \begin{bmatrix} F_3^H R_1 F_3 & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & F_3^H R_2 F_3 & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & F_3^H R_3 F_3 \end{bmatrix} \end{aligned} \tag{23}$$

which is a diagonal matrix since the 3-point DFT matrix diagonalizes the autocorrelation matrices R_1, R_2, and R_3 of the periodic sub-processes.