IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 25, 2020, accepted July 10, 2020, date of publication August 7, 2020, date of current version August 21, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3014884

Blind Estimation of Spreading Sequence and Data Bits in Direct-Sequence Spread Spectrum Communication Systems

HOESANG CHOI AND HICHAN MOON, (Member, IEEE)
Department of Electrical Engineering, Hanyang University, Seoul 04763, South Korea

Corresponding author: Hichan Moon (hcmoon@hanyang.ac.kr)

This work was supported by the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korean government (KNPA), LTE-based accurate positioning technique for emergency rescue, under Grant 2019001291.

ABSTRACT At the transmitter of a Direct Sequence Spread Spectrum (DSSS) communication system, data bits are spread using a spreading sequence. To demodulate the transmitted data bits, the received signal should be de-spread using the same spreading sequence. However, there are many non-cooperative communication systems, where a receiver does not have the information on the spreading sequence. In such systems, to demodulate the transmitted data bits across a wide area, it is necessary to blindly estimate the spreading sequence at low Signal-to-Noise Ratio (SNR). Furthermore, for real-time processing, the computational complexity for the blind estimation should be minimized. In this paper, a novel algorithm is proposed to blindly estimate a spreading sequence and data bits based on turbo processing for both synchronous and asynchronous cases. The proposed algorithm can significantly improve the error rates of estimated data bits and the chips in the spreading sequence compared to an Eigen-Value Decomposition (EVD) based algorithm. Furthermore, compared to the EVD-based algorithm, the proposed algorithm drastically reduces the number of required multiplications.

INDEX TERMS Blind estimation, direct sequence spread spectrum, turbo processing.

I. INTRODUCTION

Direct Sequence Spread Spectrum (DSSS) has been widely used in both military and commercial communication systems [1]–[6]. At the transmitter of a DSSS communication system, to generate a wideband signal, data bits are multiplied by a spreading sequence, which comprises multiple chips [2]. The transmitted data bits are demodulated after the received signal is despread with the same sequence. The ratio of the chip rate to the data bit rate is defined as processing gain. Due to the processing gain, the received signal has low probability of interception and can be demodulated even below the level of Additive White Gaussian Noise (AWGN) power [2], [5]–[7].

There are several non-cooperative communication systems, where the receiver does not have information on a transmitted DSSS signal. The applications of these systems include eavesdropping, spectrum surveillance and source localization [8]–[11]. In non-cooperative communication systems, it is common for the receiver to possess information on the modulation parameters, including data rate and a spreading sequence used in the transmitter. Furthermore, there are cases where the receiver does not even have information of the presence of a DSSS signal.

The receiver should blindly estimate several modulation parameters and a spreading sequence to recover transmitted data bits. Without the information on a DSSS signal, the usual procedure for blind estimation is implemented as follows [12]: The receiver detects whether there is a DSSS signal. If there exists a DSSS signal, the receiver estimates the modulation parameters such as the data bit rate and the length of a spreading sequence from the received signal. After parameter estimation, a spreading sequence is estimated to demodulate the transmitted signal.

To detect the presence of a DSSS signal, several studies have been performed based on energy radiometry [13] and the autocorrelation of received signals [7], [14]. In addition, several studies have investigated the blind estimation of modulation parameters [15]–[18].

To recover transmitted data bits in non-cooperative communication systems, critical challenge is the estimation of a spreading sequence from a received DSSS signal.

The associate editor coordinating the review of this manuscript and approving it for publication was Zilong Liu.

This estimation problem is extremely difficult even with the information of the data bit rate and the length of a spreading sequence. Several studies have examined the blind estimation of a spreading sequence [9], [10], [19]–[25]. In [9], a spreading sequence was estimated in a multi-path environment based on a multi-channel identification technique [26], [27]. Neural network algorithms were applied for the blind estimation of a spreading sequence in [22]–[24]. In [10] and [25], an Eigen-Value Decomposition (EVD)-based algorithm was proposed for the blind estimation of a spreading sequence. This algorithm estimates a spreading sequence based on the EVD of the autocorrelation matrix of the received signal. The EVD-based algorithm is one of the best performing algorithms for blind estimation of a spreading sequence. However, the EVD-based algorithm requires high computational complexity. Furthermore, there exists a partial-encoding problem in an asynchronous case [25]. To solve the partial-encoding problem, several modified algorithms have been proposed based on the EVD-based algorithm in [15], [28]–[30]. However, these algorithms result in increased computation complexity and performance degradation.

In non-cooperative communication systems, it is essential to blindly estimate a spreading sequence and data bits from DSSS signals at low SNR. Furthermore, for real-time processing, computational complexity should be minimized for the blind estimation.

Therefore, in this paper, a novel algorithm is proposed to blindly estimate a spreading sequence and data bits based on turbo processing. With the proposed algorithm, a receiver calculates extrinsic information based on received signals and updates Log-Likelihood Ratios (LLRs), iteratively [31], [32]. Turbo processing has been applied to decoding problems, equalization, coded modulation, and joint source and channel estimation [31]–[33]. However, there has been no previous research on the blind estimation of a spreading sequence and data bits based on turbo processing.

The contributions of this paper are as follows:

- A novel algorithm is proposed to simultaneously blindly estimate a spreading sequence and data bits simultaneously. This is the first approach based on turbo processing in estimating a spreading sequence for non-cooperative communication systems.
- The proposed algorithm can be implemented for both synchronous and asynchronous cases. In other words, even when the receiver does not have prior information on time synchronization, the proposed algorithm provides a method to jointly estimate a spreading sequence and data bits and to acquire time synchronization.
- The performance of the proposed algorithm is evaluated via simulations. From the simulation, it is shown that the proposed algorithm can achieve better estimation performance than that of an EVD-based algorithm. Especially, a performance gain of approximately 3.0 dB performance gain is observed in an asynchronous case.
- The computational complexity is analyzed for the proposed algorithm. As the proposed algorithm drastically reduces the number of multiplications compared with the EVD-based algorithm, it is less computationally complex.
- With the proposed algorithm, a spreading sequence is estimated without a partial-encoding problem.

The remainder of this paper is organized as follows. In Section II, the system model is presented for this paper. In Section III, a novel algorithm is proposed to blindly estimate a spreading sequence and data bits. Then, simulation results are shown and discussed in Section IV. Finally, the conclusions of this paper are presented in Section V.

II. SYSTEM MODEL

In this section, a system model is presented for the proposed blind estimation algorithm. In the system, a transmitter sends a baseband DSSS signal, which is the product of data bits and a spreading sequence $\boldsymbol{c} = \{c_1, c_2, \cdots, c_L\}$, where L is the length of a spreading sequence. The spreading sequence is a random binary sequence of length L and a data bit is modulated by Binary Phase Shift Keying (BPSK). It is assumed that the length of the spreading sequence is equal to spreading gain and that the spreading sequence is repeatedly used to transmit multiple data bits.

The transmitted baseband signal $s(t)$ is expressed as

$$s(t) = \sum_{k=1}^{\infty} a_k \sum_{l=1}^{L} c_l \, p(t - lT_c - kT_s), \tag{1}$$

where a_k is the kth data bit ($a_k \in \{-1, 1\}$), c_l is the lth chip in a spreading sequence ($c_l \in \{-1, 1\}$), T_c is the chip duration of a spreading sequence ($T_c = T_s/L$), T_s is the period of a data bit, and $p(t)$ is a pulse-shaping filter.

In non-cooperative communication systems [8]–[11], a receiver does not have the information of the spreading sequence $\boldsymbol{c}$. Therefore, in this paper, the receiver simultaneously estimates a spreading sequence $\boldsymbol{c}$ and data bits. It is assumed that the data bit period T_s and the length L of a spreading sequence are known to the receiver before the blind estimation of a spreading sequence as in previous studies [9], [10], [19].

At a receiver, a received signal $r(t)$ is expressed as

$$r(t) = s(t - \tau) + w(t), \quad t \geq 0, \tag{2}$$

where $w(t)$ is the complex AWGN with zero mean and variance N_0 and τ is the unknown desynchronization value between the receiver and a transmitter.[1] In this paper, without loss of generality, it is assumed that $\tau = 0$, but the receiver does not have the information of the τ value. In addition, N_0 is assumed to be known to the receiver and channel estimation is not required since the transmitted signal is a baseband signal. The received signal is sampled at a rate T_c. The sampled signal $z[i]$ can be expressed as

$$z[i] = r(iT_c + \zeta), \quad i = 0, 1, 2 \cdots, \tag{3}$$

where ζ is an arbitrary value in the rage of $[0, T_c)$.

[1] In non-cooperative systems, since a receiver does not have information of the desynchronization value, the boundary of a data bit is not known at the receiver.

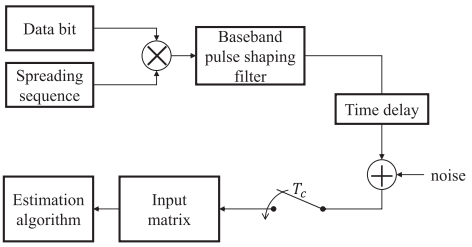


FIGURE 1. System block diagram.

Let us define two different types of blind estimation of a spreading sequence and data bits. One is a synchronous case, where a receiver has the information of the desynchronization value τ. The other is an asynchronous case, where the receiver does not have the information of the desynchronization value τ.

With the proposed algorithm, a spreading sequence and M spread data bits are simultaneously estimated based on an input matrix, which is constructed by sampling the received signal. The block diagram of the system is shown in Fig. 1.

Let us define $\boldsymbol{y}_{m,\Delta}$ as follows:

$$\boldsymbol{y}_{m,\Delta} = \begin{bmatrix} z[(m-1)L+\Delta] \\ z[(m-1)L+\Delta+1] \\ \cdots \\ z[mL+\Delta] \end{bmatrix}^T, \tag{4}$$

where $(\cdot)^T$ is a transpose operator and Δ denotes a time offset. Here, the time offset Δ is the number of shifted samples to construct an input matrix. Subsequently, with the time offset Δ, an input matrix $\boldsymbol{Y}_\Delta$ is expressed as follows:

$$\boldsymbol{Y}_\Delta = \begin{bmatrix} \boldsymbol{y}_{1,\Delta} \\ \boldsymbol{y}_{2,\Delta} \\ \vdots \\ \boldsymbol{y}_{M,\Delta} \end{bmatrix}. \tag{5}$$

The element in the kth row and the lth column of $\boldsymbol{Y}_\Delta$ is denoted by $y^{\Delta}_{(k,l)}$. For a time offset Δ, let us define a time synchronization error ϵ_Δ as the time difference between the beginning of a data bit and the first sample in an input matrix, where ϵ_Δ is a value in the range of $[0, T_s)$. In this paper, since τ is assumed to be zero, ϵ_Δ is the same as ΔT_c.

In a synchronous case, it is sufficient to consider only an input matrix $\boldsymbol{Y}_\Delta$ with $\Delta = 0$. Then, the input matrix $\boldsymbol{Y}_0$ is expressed as

$$\boldsymbol{Y}_0 = \begin{bmatrix} a_1c_1 & a_1c_2 & \cdots & a_1c_L \\ a_2c_1 & a_2c_2 & \cdots & a_2c_L \\ \vdots & \vdots & \ddots & \vdots \\ a_Mc_1 & a_Mc_2 & \cdots & a_Mc_L \end{bmatrix} + \boldsymbol{W}_0, \tag{6}$$

where $\boldsymbol{W}_0$ is the noise part of the input matrix $\boldsymbol{Y}_0$ and is expressed as

$$\boldsymbol{W}_0 = \begin{bmatrix} w[0] & w[1] & \cdots & w[L-1] \\ w[L] & w[L+1] & \cdots & w[2L-1] \\ \vdots & \vdots & \ddots & \vdots \\ w[ML-L] & w[ML-L+1] & \cdots & w[ML-1] \end{bmatrix}, \tag{7}$$

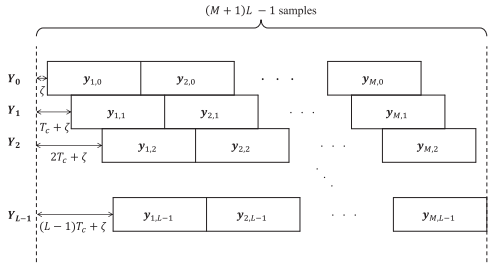


FIGURE 2. Input matrices generated from received samples.

where $w[i] = w(iT_c + \zeta)$, which is a sample of AWGN $w(t)$. In the mth row of $\boldsymbol{Y}_0$, a_k is multiplied with the spreading sequence $\boldsymbol{c}$. In addition, in the lth column of $\boldsymbol{Y}_0$, each data bit is multiplied by the same chip c_l of the spreading sequence. Then, the element in the kth row and the lth column $y^0_{(k,l)}$ of $\boldsymbol{Y}_0$ is

$$y^0_{(k,l)} = a_kc_l + w^0_{(k,l)}, \tag{8}$$

where $w^0_{(k,l)}$ is the element in the kth row and the lth column element of $\boldsymbol{W}_0$.

In an asynchronous case, since a receiver does not have the information of the desynchronization value, the proposed algorithm constructs L input matrices. The L input matrices are constructed from the $(M+1)L-1$ samples with L different time offsets $\Delta = 0, 1, \cdots, L-1$. Fig. 2 shows the construction of L input matrices from the samples for the proposed algorithm.

Therefore, for the proposed algorithm, ML and $(M+1)L-1$ consecutive samples are required in synchronous and asynchronous cases, respectively.

If ϵ_Δ is not zero, the input matrix $\boldsymbol{Y}_\Delta$ is expressed as

$$\boldsymbol{Y}_\Delta = \left[\boldsymbol{S}_{\Delta,0}, \boldsymbol{S}_{\Delta,1}\right] + \boldsymbol{W}_\Delta, \tag{9}$$

where $\boldsymbol{W}_\Delta$ is the noise part of the $\boldsymbol{Y}_\Delta$ with a time offset Δ,

$$\boldsymbol{S}_{\Delta,0} = \begin{bmatrix} a_1c_{\Delta+1} & \cdots & a_1c_L \\ a_2c_{\Delta+1} & \cdots & a_2c_L \\ \vdots & \vdots & \vdots \\ a_Mc_{\Delta+1} & \cdots & a_Mc_L \end{bmatrix} \tag{10}$$

and

$$\boldsymbol{S}_{\Delta,1} = \begin{bmatrix} a_2c_1 & \cdots & a_2c_\Delta \\ a_3c_1 & \cdots & a_3c_\Delta \\ \vdots & \vdots & \vdots \\ a_{M+1}c_1 & \cdots & a_{M+1}c_\Delta \end{bmatrix}. \tag{11}$$

In the kth row of $\boldsymbol{Y}_\Delta$, a_k is multiplied by the last $L-\Delta$ chips in the spreading sequence and a_{k+1} is multiplied by the first Δ chips in the spreading sequence due to the synchronization error.

III. PROPOSED ALGORITHM

In this section, the proposed algorithm is presented for blind estimation of a spreading sequence and data bits simultaneously. This proposed algorithm is based on the turbo processing principle for a maximum *a posteriori* probability decision. In the proposed algorithm, the channel LLRs of chips in a spreading sequence and those of data bits are iteratively calculated to obtain *a posteriori* LLRs.

At each iteration, after the channel LLRs of chips in a spreading sequence are first calculated, *a priori* probabilities of chips in a spreading sequence are updated with the calculated channel LLRs. Subsequently, the channel LLRs of data bits are calculated with the updated *a priori* probabilities of chips in a spreading sequence. Moreover, *a priori* probabilities of data bits are updated with the calculated channel LLRs.[2] The extrinsic information of the calculated channel LLRs is used in the next iteration.

A. LOG-LIKELIHOOD RATIO

Let us consider a synchronous case. In this case, the input matrix $\boldsymbol{Y}_\Delta$ is constructed with a time offset $\Delta = 0$.

With a time offset Δ, *a posteriori* LLR $L^\Delta(c_l)$ of c_l, is defined as

$$\begin{aligned} L^\Delta(c_l) &\triangleq L\left(c_l|\boldsymbol{v}_{(l,\Delta)}\right) \\ &= L_{\text{ch}}\left(\boldsymbol{v}_{(l,\Delta)} \,|\, c_l\right) + L_{\text{pr}}\left(c_l\right) \\ &= \sum_{k=1}^{M} L_{\text{ch}}\left(y^\Delta_{(k,l)} \,|\, c_l\right) + L_{\text{pr}}\left(c_l\right), \end{aligned} \tag{12}$$

where $\boldsymbol{v}_{(l,\Delta)}$ is the lth column of $\boldsymbol{Y}_\Delta$, $L_{\text{ch}}(y^\Delta_{(k,l)} \,|\, c_l)$ is the channel LLR of $y^\Delta_{(k,l)}$ for c_l and $L_{\text{pr}}(c_l)$ is *a priori* LLR of c_l. Here,

$$L_{\text{ch}}\left(y^\Delta_{(k,l)} \,|\, c_l\right) = \log\left[\frac{\Pr\left(y^\Delta_{(k,l)} \,|\, c_l = 1\right)}{\Pr\left(y^\Delta_{(k,l)} \,|\, c_l = -1\right)}\right], \tag{13}$$

and

$$L_{\text{pr}}\left(c_l\right) = \log\left[\frac{\Pr\left(c_l = 1\right)}{\Pr\left(c_l = -1\right)}\right]. \tag{14}$$

With a time offset Δ, *a posteriori* LLR $L(a_k)$ of a_k is defined as

$$\begin{aligned} L^\Delta(a_k) &\triangleq L(a_k|\boldsymbol{u}_{(k,\Delta)}) \\ &= L_{\text{ch}}(\boldsymbol{u}_{(k,\Delta)}|a_k) + L_{\text{pr}}(a_k) \\ &= \sum_{l=1}^{L} L_{\text{ch}}\left(y^\Delta_{(k,l)} \,|\, a_k\right) + L_{\text{pr}}(a_k), \end{aligned} \tag{15}$$

[2]It is possible to change the order of the LLR calculation between chips in a spreading sequence and data bits. For convenience, in this paper, the LLR calculation of chips in a spreading sequence is followed by the LLR calculation of data bits.

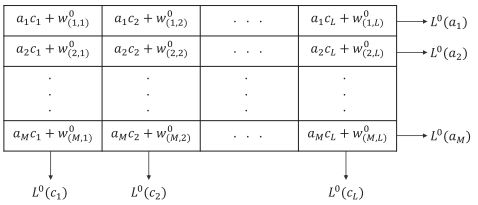


FIGURE 3. LLR calculation from an input matrix $\boldsymbol{Y_0}$.

where $\boldsymbol{u}_{(k,\Delta)}$ is the kth row of $\boldsymbol{Y}_\Delta$, $L_{\text{ch}}(y^\Delta_{(k,l)} \,|\, a_k)$ is the channel LLR of $y^\Delta_{(k,l)}$ for a_k and $L_{\text{pr}}(a_k)$ is *a priori* LLR of a_k. Here,

$$L_{\text{ch}}\left(y^\Delta_{(k,l)} \,|\, a_k\right) = \log\left[\frac{\Pr\left(y^\Delta_{(k,l)} \,|\, a_k = 1\right)}{\Pr\left(y^\Delta_{(k,l)} \,|\, a_k = -1\right)}\right], \tag{16}$$

and

$$L_{\text{pr}}\left(a_k\right) = \log\left[\frac{\Pr\left(a_k = 1\right)}{\Pr\left(a_k = -1\right)}\right]. \tag{17}$$

From (12) and (15), it is observed that *a posteriori* LLR of c_l can be calculated with the lth column of $\boldsymbol{Y}_\Delta$ and *a posteriori* LLR of a_k can be calculated with the kth row of $\boldsymbol{Y}_\Delta$. Fig. 3 shows how to calculate *a posteriori* LLRs from an input matrix.

In a synchronous case, as $y^\Delta_{(k,l)}$ is expressed as $y^\Delta_{(k,l)} = a_k c_l + w^\Delta_{(k,l)}$, the channel LLR of $y^\Delta_{(k,l)}$ for c_l is calculated as (18), shown at the bottom of the page where $\gamma(x|y) = -x^2 + \log \Pr(y)$, $\text{Re}[x]$ is a real part of x and the approximation (a) is obtained from [34]

$$\log[\exp(x_1) + \exp(x_2)] \approx \max(x_1, x_2). \tag{19}$$

As $\max(x_1, x_2)$ is the same as $(x_1 + x_2 + |x_1 - x_2|)/2$, the channel LLR of $y^\Delta_{(k,l)}$ for c_l is expressed as

$$\begin{aligned} &L_{\text{ch}}\left(y^\Delta_{(k,l)} \,|\, c_l\right) \\ &\approx \left|\frac{4\text{Re}[y^\Delta_{(k,l)}]}{N_0} + L_{\text{pr}}(a_k)\right| - \left|\frac{-4\text{Re}[y^\Delta_{(k,l)}]}{N_0} + L_{\text{pr}}(a_k)\right|, \\ &= \begin{cases} \text{sgn}\left(\text{Re}[y^\Delta_{(k,l)}]\right) 2L_{\text{pr}}(a_k), \\ \qquad\qquad \text{if } \left|\text{Re}[y^\Delta_{(k,l)}]\right| > \dfrac{N_0}{4} L_{\text{pr}}(a_k), \\ \dfrac{8\text{Re}[y^\Delta_{(k,l)}]}{N_0}, \qquad \text{otherwise}, \end{cases} \end{aligned} \tag{20}$$

where $\text{sgn}(x)$ is a sign function [31].

$$\begin{aligned} L_{\text{ch}}\left(y^\Delta_{(k,l)} \,|\, c_l\right) &= \log\left[\frac{\sum_{a_k\in\{-1,1\}} \Pr\left(y^\Delta_{(k,l)} \,|\, c_l = 1, a_k\right)\Pr(a_k)}{\sum_{a_k\in\{-1,1\}} \Pr\left(y^\Delta_{(k,l)} \,|\, c_l = -1, a_k\right)\Pr(a_k)}\right], \\ &\overset{(a)}{\approx} \max\left[\gamma\left(\frac{\text{Re}[y^\Delta_{(k,l)}] - 1}{\sqrt{N_0}}\middle| a_k{=}1\right), \gamma\left(\frac{\text{Re}[y^\Delta_{(k,l)}] + 1}{\sqrt{N_0}}\middle| a_k{=}-1\right)\right] \\ &\quad - \max\left[\gamma\left(\frac{\text{Re}[y^\Delta_{(k,l)}] + 1}{\sqrt{N_0}}\middle| a_k{=}1\right), \gamma\left(\frac{\text{Re}[y^\Delta_{(k,l)}] - 1}{\sqrt{N_0}}\middle| a_k{=}-1\right)\right], \end{aligned} \tag{18}$$

Similarly, $L_{\text{ch}}\left(y^{\Delta}_{(k,l)} \mid a_k\right)$, the channel LLR of $y^{\Delta}_{(k,l)}$ for a_k, can be expressed as

$$\begin{aligned} & L_{\text{ch}}\left(y^{\Delta}_{(k,l)} \mid a_k\right) \\ &= \log\left[\frac{\sum_{c_l\in\{-1,1\}} \Pr\left(y^{\Delta}_{(k,l)} \mid a_k = 1, c_l\right)\Pr(c_l)}{\sum_{c_l\in\{-1,1\}} \Pr\left(y^{\Delta}_{(k,l)} \mid a_k = -1, c_l\right)\Pr(c_l)}\right], \\ &\approx \left|\frac{4\text{Re}[y^{\Delta}_{(k,l)}]}{N_0} + L_{\text{pr}}(c_l)\right| - \left|\frac{-4\text{Re}[y^{\Delta}_{(k,l)}]}{N_0} + L_{\text{pr}}(c_l)\right|, \\ &= \begin{cases} \text{sgn}\left(\text{Re}[y^{\Delta}_{(k,l)}]\right) 2L_{\text{pr}}(c_l), & \\ \qquad\qquad \text{if } \left|\text{Re}[y^{\Delta}_{(k,l)}]\right| > \dfrac{N_0}{4}L_{\text{pr}}(c_l), & \\ \dfrac{8\text{Re}[y^{\Delta}_{(k,l)}]}{N_0}, \quad \text{otherwise.} & \end{cases} \end{aligned} \tag{21}$$

For the turbo processing, the extrinsic information for c_l and a_k are denoted as $\lambda^{\Delta}_{c_l}$ and $\lambda^{\Delta}_{a_k}$, respectively. The extrinsic information is expressed as follows

$$\begin{aligned} \lambda^{\Delta}_{c_l} &= \frac{1}{2^{n_1}} L_{\text{ch}}(\boldsymbol{v}_{(l,\Delta)} | c_l), \\ &= \frac{1}{2^{n_1}} \sum_{k=1}^{M} L_{\text{ch}}(y^{\Delta}_{(k,l)} | c_l), \end{aligned} \tag{22}$$

$$\begin{aligned} \lambda^{\Delta}_{a_k} &= \frac{1}{2^{n_2}} L_{\text{ch}}(\boldsymbol{u}_{(k,\Delta)} | a_k), \\ &= \frac{1}{2^{n_2}} \sum_{l=1}^{L} L_{\text{ch}}(y^{\Delta}_{(k,l)} | a_k), \end{aligned} \tag{23}$$

where n_1 and n_2 are the smallest integer greater than or equal to $\log_2 M$ and $\log_2 L$, respectively.[3] The extrinsic information is used to update *a priori* probability.

B. SYNCHRONOUS CASE

Let us consider a synchronous case, where the receiver has the information of a desynchronization value τ. Then, the *aposteriori* LLRs of chips in the spreading sequence $L^0(c_l)$ and those of data bits $L^0(a_k)$ are calculated from (12) and (15) for the input matrix $\boldsymbol{Y}_0$, respectively. At each iteration, $L^0(c_l)$ and $L^0(a_k)$ are calculated for $l = 1, 2, \cdots, L$ and $k = 1, 2, \cdots, M$. To calculate $L^0(c_l)$ and $L^0(a_k)$, the *a priori* probabilities $\Pr(c_l = 1)$ and $\Pr(a_k = 1)$ are updated based on the extrinsic information obtained from the previous iteration.

Before the first iteration, the *a priori* probabilities $\Pr(c_l = 1)$ and $\Pr(a_k = 1)$ are initialized for all l and k values. After the initialization, $L^0(c_l)$ is first calculated from (12). Subsequently, the extrinsic information $\lambda^0_{c_l}$ for c_l is fed back to update the *a priori* probability of $\Pr(c_l = 1)$ as follows:

$$\Pr(c_l = 1) \leftarrow f_{\text{atv}}\left(\lambda^0_{c_l}\right), \tag{24}$$

[3]As division by a power of two can be easily implemented by a shift operation, its computational complexity is negligible compared with a multiplication operation.

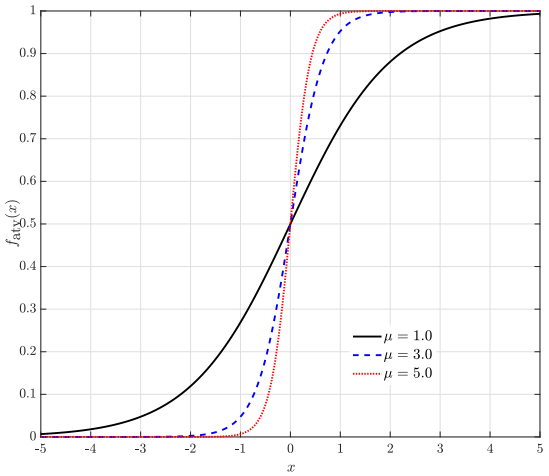


FIGURE 4. Plots of activation functions.

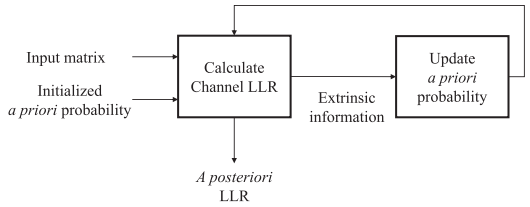


FIGURE 5. Iterative calculation of *a posteriori* LLR.

where $f_{\text{atv}}(x)$ is an activation function. In this paper, as in [35]–[37], a logistic function is used for the activation function. This function is expressed as [38]:

$$f_{\text{atv}}(x) = \frac{1}{1 + e^{-\mu x}}, \tag{25}$$

where μ is a positive coefficient. Fig. 4 shows plots of the activation function (25) for different coefficient μ values.

After $\Pr(c_l = 1)$ is updated, $L^0(a_k)$ is calculated from (15). Then, the extrinsic information $\lambda^0_{a_k}$ for a_k is also fed back to update the *a priori* probability of $\Pr(a_k = 1)$ as follows:

$$\Pr(a_k = 1) \leftarrow f_{\text{atv}}\left(\lambda^0_{a_k}\right). \tag{26}$$

The updated *a priori* probabilities are used at the next iteration to calculate $L^0(c_l)$ and $L^0(a_k)$. Fig. 5 summarizes the overall iterative procedure for the proposed algorithm.

After the end of the last iteration, the estimated chips in a spreading sequence and data bits are obtained based on the sign of $L^0(c_l)$ and $L^0(a_k)$ as follows:

$$\widehat{c_l} = \text{sgn}\left(L^0(c_l)\right), \quad l = 1, 2, \cdots, L, \tag{27}$$

$$\widehat{a_k} = \text{sgn}\left(L^0(a_k)\right), \quad k = 1, 2, \cdots, M. \tag{28}$$

C. ASYNCHRONOUS CASE

Let us consider an asynchronous case, where the receiver does not have the information of the desynchronization value τ. In this case, the receiver estimates a spreading sequence and data bits for all possible desynchronization values. Therefore, L input matrices are generated from the consecutive $(M + 1)L - 1$ samples with L time offsets Δ.

Subsequently, among the L estimations of chips in a spreading sequence and data bits, the receiver selects the most reliable one.

For each time offset Δ, the *a posteriori* LLRs of chips in a spreading sequence and those of data bits are calculated from (12) and (15). After the end of the last iteration for each Δ value, tentative estimations of a spreading sequence and data bits are obtained as

$$\widehat{c}_{l(\Delta)} = \operatorname{sgn}\left(L^{\Delta}(c_l)\right), \quad l = 1, 2, \cdots, L, \tag{29}$$

$$\widehat{a}_{k(\Delta)} = \operatorname{sgn}\left(L^{\Delta}(a_k)\right), \quad k = 1, 2, \cdots, M. \tag{30}$$

Then, from the tentative estimation, an input matrix $\widehat{\boldsymbol{Y}_{\Delta}}$ is regenerated as

$$\widehat{\boldsymbol{Y}_{\Delta}} = \operatorname{sgn}\left(\begin{bmatrix} \widehat{c}_{1\Delta} \\ \widehat{c}_{2\Delta} \\ \vdots \\ \widehat{c}_{L\Delta} \end{bmatrix} \cdot \left[\widehat{a}_{1\Delta}, \widehat{a}_{2\Delta}, \cdots, \widehat{a}_{M\Delta}\right]\right). \tag{31}$$

The receiver computes the Euclidean distance between an input matrix $\boldsymbol{Y}_{\Delta}$ and a regenerated input matrix $\widehat{\boldsymbol{Y}_{\Delta}}$. The computed Euclidean distance is used as a reliability measure for a tentative estimation. Let us define Θ as the time offset Δ, with which the minimum Euclidean distance is obtained. Then, Θ is expressed as

$$\begin{aligned}
\Theta &= \arg\min_{\Delta}\left(\left|\boldsymbol{Y}_{\Delta} - \widehat{\boldsymbol{Y}_{\Delta}}\right|^2\right) \\
&= \arg\min_{\Delta}\left(\sum_{k=1}^{M}\sum_{l=1}^{L} z^2[(k-1)L + l - 1 + \Delta]\right. \\
&\quad \left. - 2\widehat{y}^{\Delta}_{(k,l)} z[(k-1)L + l - 1 + \Delta] + (\widehat{y}^{\Delta}_{(k,l)})^2\right) \\
&\overset{(a)}{\equiv} \arg\min_{\Delta}\left(\sum_{i=\Delta}^{L-1}(z[i])^2 + \sum_{i=ML-1}^{ML+\Delta}(z[i])^2\right. \\
&\quad \left. - 2\sum_{k=1}^{M}\sum_{l=1}^{L}\widehat{y}^{\Delta}_{(k,l)} z[(k-1)L + l - 1 + \Delta]\right),
\end{aligned} \tag{32}$$

where $\widehat{y}^{\Delta}_{(k,l)}$ is the kth row and the lth column element of $\widehat{\boldsymbol{Y}_{\Delta}}$. (a) is derived as follows:

(i) In the second line of (32), with $i = L, L+1, \cdots, ML-1$, the first term $(z[i])^2$ is included for all Δ. Therefore, this term can be simplified as the first and the second terms of the third line in (32).

(ii) As the value of $\widehat{y}^{\Delta}_{(k,l)}$ is 1 or -1, $(\widehat{y}^{\Delta}_{(k,l)})^2$ is always one for all l and k.

Therefore, the final estimation of a spreading sequence and data bits can be obtained as follows:

$$\widehat{c}_l = \widehat{c}_{l(\Theta)}, \quad l = 1, 2, \cdots, L, \tag{33}$$

$$\widehat{a}_k = \widehat{a}_{k(\Theta)}, \quad k = 1, 2, \cdots, M. \tag{34}$$

Algorithm 1 presents the proposed algorithm as a pseudo code for an asynchronous case.

Algorithm 1 Blind Estimation of a Spreading Sequence and Data Bits

Input : $(M+1)L - 1$ consecutive samples
Output: Estimated chips in a spreading sequence and data bits

for $\Delta = 0$ **to** $L - 1$ **do**
  Initialization of $\Pr(c_l)$ and $\Pr(a_k)$;
  for $n = 1$ **to** N_{iter} **do**
    for $l \leftarrow 1$ **to** L **do**
      calculate $L_{\text{ch}}(y^{\Delta}_{(k,l)}|c_l)$, $L^{\Delta}(c_l)$, $\lambda^{\Delta}_{c_l}$;
      $\Pr(c_l) \leftarrow f_{\text{atv}}(\lambda^{\Delta}_{c_l})$;
      $\widehat{c}_{l(\Delta)} = \operatorname{sgn}(L^{\Delta}(c_l))$;
    end
    for $k \leftarrow 1$ **to** M **do**
      calculate $L_{\text{ch}}(y^{\Delta}_{(k,l)}|a_k)$, $L^{\Delta}(a_k)$, $\lambda^{\Delta}_{a_k}$;
      $\Pr(a_k) \leftarrow f_{\text{atv}}(\lambda^{\Delta}_{a_k})$;
      $\widehat{a}_{k(\Delta)} = \operatorname{sgn}(L^{\Delta}(a_k))$;
    end
  end
end
$\Theta = \arg\min_{\Delta}\left(\left|\boldsymbol{Y}_{\Delta} - \widehat{\boldsymbol{Y}_{\Delta}}\right|^2\right)$;
$\widehat{c}_l = \widehat{c}_{l(\Theta)}, \quad l = 1, 2, \cdots, L$;
$\widehat{a}_k = \widehat{a}_{k(\Theta)}, \quad k = 1, 2, \cdots, M$;

D. COMPUTATIONAL COMPLEXITY

In this subsection, the computational complexity is analyzed for the proposed algorithm. In addition, the computational complexity is compared with that of the conventional EVD-based algorithm.

Let us first consider a synchronous case. The computational complexity is summarized as follows:

- Normalization of an $M \times L$ input matrix by N_0 (signal-to-noise ratio normalization):
  - ML multiplications
- Calculation of channel LLRs as in (20) and (21):
  - $2MLN_{\text{iter}}$ additions[4]
- Calculation of extrinsic information based on the calculated channel LLRs as in (22) and (23):
  - $(2ML - M - L)N_{\text{iter}}$ additions
  - $(M + L)N_{\text{iter}}$ bit shift operations
- Update of *a priori* probabilities using a logistic function as in (24) and (26), that can be implemented with a look-up table [39]:
  - $(M + L)N_{\text{iter}}$ memory accesses for a one-dimensional look-up table of a logistic function
- Calculation of *a priori* LLRs similar to that in (14) and (17), which can be implemented with a look-up table:

[4]The calculation of a channel LLR comprises of a magnitude comparison of an input value and multiplications by a power of two. The complexity of magnitude comparison is the same as that of an addition. In addition, multiplication by a power of two can be implemented with a shift operation, which has negligible computational complexity compared with addition.

TABLE 1. Computational complexity.

	Proposed Algorithm		EVD-based Algorithm
	Sync.	Async.	
Multiplication	ML	$\approx (M+1)L$	ML^2+L^3
Addition	$4N_{\text{iter}}ML$	$\approx 4N_{\text{iter}}ML^2$	ML^2+L^3
Lookup table	$2N_{\text{iter}}(M+L)$	$2N_{\text{iter}}L(M+L)$	0

  - $(M+L)N_{\text{iter}}$ memory accesses for an one-dimensional look-up table of a logarithm function
- Calculation of *a posteriori* LLRs as in (12) and (15):
  - $(M+L)N_{\text{iter}}$ additions

Therefore, the proposed algorithm requires approximately ML multiplications and $4N_{\text{iter}}ML$ additions for the synchronous case, as the shift operation and memory access for a look-up table are negligible compared with an addition or a multiplication.

On the other hand, in the asynchronous case, the proposed algorithm requires L input matrices of size $M \times L$ obtained from $(M+1)L-1$ samples. In addition, to calculate the Euclidean distance between an input matrix $\boldsymbol{Y}_\Delta$ and its regenerated matrix $\widehat{\boldsymbol{Y}}_\Delta$, $2L-1$ multiplications and $L(ML-1)$ additions are required. Therefore, the proposed algorithm requires $(M+1)L+2L-2$ multiplications and $4N_{\text{iter}}ML^2+L(ML-1)$ additions. Thus, for the asynchronous case, computational complexity is approximately L times that of the synchronous case.

For the conventional EVD-based algorithm [10], [20], [25], it is necessary to compute the average of M covariance matrices of size $L \times L$. Subsequently, after the calculation of eigen-decomposition of the averaged covariance matrix, the spreading sequence is estimated based on two largest eigenvalues and the corresponding eigenvectors. Therefore, ML^2+L^3 additions and ML^2+L^3 multiplications are required for this algorithm [21], [40], [41]. With the EVD-based algorithm, almost the same computational complexity is required for both synchronous and asynchronous cases.

Table 1 summarizes the computational complexity of the EVD-based algorithm and proposed algorithm in synchronous and asynchronous cases.

IV. NUMERICAL RESULT

In this section, simulation results of the proposed algorithm are presented. A simulator was built to obtain the results for the proposed algorithm in an AWGN channel. The performance of the proposed algorithm was compared with that of an EVD-based algorithm.

The estimation performances were obtained from 10,000 runs of Monte Carlo simulations for an SNR value between -20.0 dB and -10.0 dB. For the simulation, 100 data bits were spread with a spreading sequence of length 31 and the coefficient μ for an activation function is fixed to 3.0. The results were obtained after N_{iter} iterations, where N_{iter} is an integer in the range $[1, 25]$. In the asynchronous cases, the desynchronization value was set to iT_c, where i is an integer, uniformly distributed between 0 and $L-1$. Table 2 summarizes the parameters used for the simulation.

TABLE 2. Simulation parameters.

Parameter	Value
Number M of data bits	100
Length L of a spreading sequence	31
Number N_{iter} of iterations	$[1, 25]$
Coefficient μ for an activation function	3.0
SNR (dB)	$[-20, -10]$
Desynchronization value	iT_c $(i = 0, 1, \cdots, L-1)$
Number of simulations	10,000

Let us define $P_e(x)$, which is an ideal BER (bit error rate) performance of BPSK for SNR of x after despreading. $P_e(x)$ is expressed as

$$P_e(x) = \frac{1}{2}\text{erfc}\left(\sqrt{Gx}\right), \tag{35}$$

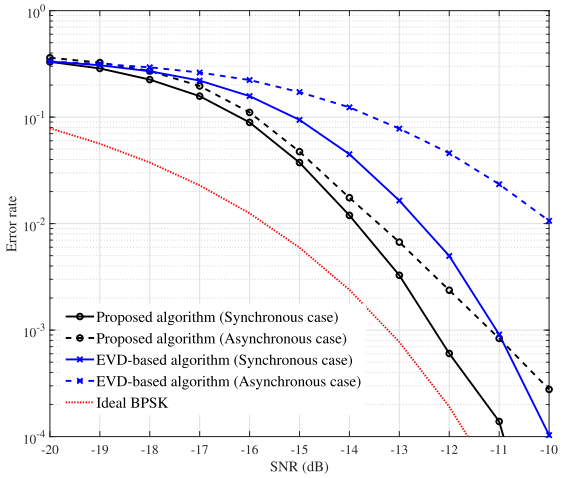
where $\text{erfc}(x)$ is an error function and G is the processing gain [42]. As the proposed estimation algorithm cannot outperform the ideal BER $P_e(x)$, $P_e(x)$ is used as a lower bound for estimation performance. For the estimation of a data bit, as a data bit is spread by a spreading sequence of length L, G becomes L. On the other hand, for the estimation of a chip in a spreading sequence, as it can be regarded that a chip in a spreading sequence is spread by a sequence of length M, G becomes M.

Fig. 6 shows the error rate versus SNR for a chip in a spreading sequence and a data bit. The solid and dashed lines represent the performances of the algorithms in synchronous and asynchronous cases, respectively. The results were obtained after 10 iterations with the proposed algorithm.

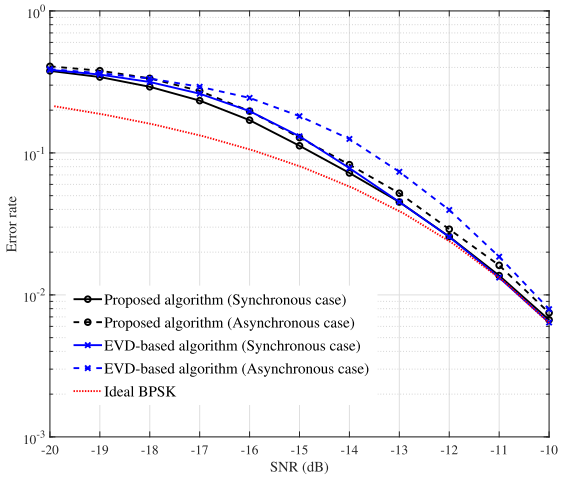
Fig. 6.(a) shows the plots of the error rate of a chip in a spreading sequence for the proposed and EVD-based algorithms. In the synchronous case, for the error rate of 10^{-2}, -13.9 dB and -12.6 dB are required with the proposed and EVD-based algorithms, respectively. Furthermore, in the asynchronous case, for the error rate of 10^{-2}, -13.4 dB and -10.0 dB are required with the proposed and EVD-based algorithms, respectively. Compared to the EVD-based algorithm, the proposed algorithm achieves better performance with respect to chips in a spreading sequence. Particularly, in the asynchronous case, the proposed algorithm has performance gain of approximately 3.0 dB compared with the EVD-based algorithm.

Fig. 6.(b) shows plots of the error rate of a data bit for a proposed and EVD-based algorithms. In the synchronous case, for the error rate of 10^{-1}, -14.7 dB and -14.5 dB are required with the proposed and EVD-based algorithms, respectively. Furthermore, in the asynchronous case, for the error rate of 10^{-1}, -14.4 dB and -13.6 dB are required with the proposed and EVD-based algorithms, respectively. Compared to the EVD-based algorithm. the proposed algorithm achieves better performance with respect to data bit error rate.

From the Fig. 6.(a) and Fig. 6.(b), it is evident that the estimation performance of a chip in a spreading sequence is



(a) Error rate of a chip in a spreading sequence.



(b) Error rate of a data bit.

FIGURE 6. Error rate vs SNR.

better than that of a data bit for a fixed SNR. This is because the size of column M is smaller than that of row L with an $M \times L$ input matrix.

The proposed algorithm is based on the iterative calculations of *a posteriori* LLRs. Therefore, the estimation performance is dependent on the number N_{iter} of iterations. To investigate the influence of N_{iter} on the estimation performance, Fig. 7 shows the plots of the error rate for different N_{iter} with the proposed algorithm. For the results, SNR is fixed at −15.0 dB. As N_{iter} increases, the error rate of a chip in a spreading sequence converges to approximately 2.1×10^{-2} or 4.2×10^{-2} in a synchronous or an asynchronous case, respectively. In addition, the error rate of a data bit converges to approximately 9.8×10^{-2} or 12.5×10^{-2} in synchronous or asynchronous case, respectively. Fig. 7 show that the performances saturate after 10 and 20 iterations for an asynchronous and a synchronous cases, respectively. In the asynchronous case, as estimation performance degrades due to the time synchronization error, the error rate converges to a higher value compared with that in the synchronous case.

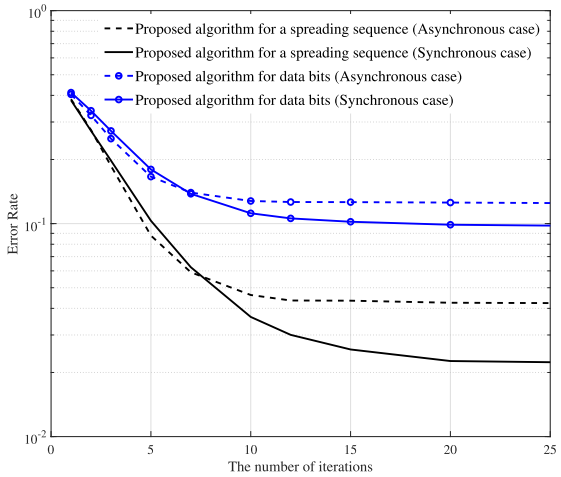


FIGURE 7. Error rate vs number N_{iter} of iterations.

Therefore, in the asynchronous case, the error rate saturates after fewer iterations than in the synchronous case.

V. CONCLUSION

In this paper, a novel algorithm is proposed to blindly estimate chips in a spreading sequence and data bits based on turbo processing principle in non-cooperative baseband communication systems. With the proposed algorithm, the *a posteriori* LLRs of chips in a spreading sequence and data bits are calculated and *a priori* probabilities are updated based on channel LLRs. The proposed algorithm can be implemented for both synchronous and asynchronous cases. Therefore, regardless of the information of a desynchronization value, chips in a spreading sequence and data bits can be estimated simultaneously. The simulation results show that the proposed algorithm can achieve better estimation performance than an EVD-based algorithm. Especially, in the asynchronous case, a performance gain of approximately 3.0 dB was achieved compared to the EVD-based algorithm to estimate the chips in a spreading sequence.

In addition, this paper analyzed the computation complexity of the proposed algorithm. The complexity of the proposed algorithm is compared with that of an EVD-based algorithm. As the proposed algorithm considerably reduced the number of multiplications required, it is less computationally complex than the EVD-based algorithm.

REFERENCES

[1] R. L. Peterson, D. E. Borth, and R. E. Ziemer, *An Introduction to Spread-Spectrum Communications*, 1st ed. Upper Saddle River, NJ, USA: Prentice-Hall, 1995.

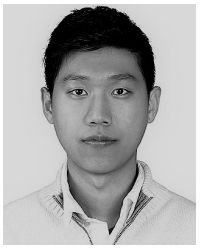
[2] A. Viterbi, *CDMA: Principles of Spread Spectrum Communication* (Addison-Wesley Wireless Communications Series). Reading, MA, USA: Addison-Wesley, 1995.

[3] D. T. Magill, F. D. Natali, and G. P. Edwards, "Spread-spectrum technology for commercial applications," *Proc. IEEE*, vol. 82, no. 4, pp. 572–584, Apr. 1994.

[4] A. Tayebi, S. Berber, and A. Swain, "Security enhancement of fix chaotic-DSSS in WSNs," *IEEE Commun. Lett.*, vol. 22, no. 4, pp. 816–819, Apr. 2018.

[5] Y. Xing, A. Hu, J. Zhang, L. Peng, and G. Li, "On radio frequency fingerprint identification for DSSS systems in low SNR scenarios," *IEEE Commun. Lett.*, vol. 22, no. 11, pp. 2326–2329, Nov. 2018.

[6] A. Masmoudi, F. Bellili, S. Affes, and A. Ghrayeb, "Maximum likelihood time delay estimation from single-and multi-carrier DSSS multipath MIMO transmissions for future 5G networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 4851–4865, Aug. 2017.
[7] G. Burel, C. Bouder, and O. Berder, "Detection of direct sequence spread spectrum transmissions without prior knowledge," in *Proc. IEEE GLOBECOM*, vol. 1, Nov. 2001, pp. 236–239.
[8] H. Zhang, P. Wei, and Q. Mou, "A semidefinite relaxation approach to blind despreading of long-code DS-SS signal with carrier frequency offset," *IEEE Signal Process. Lett.*, vol. 20, no. 7, pp. 705–708, Jul. 2013.
[9] M. K. Tsatsanis and G. B. Giannakis, "Blind estimation of direct sequence spread spectrum signals in multipath," *IEEE Trans. Signal Process.*, vol. 45, no. 5, pp. 1241–1252, May 1997.
[10] G. Burel and C. Bouder, "Blind estimation of the pseudo-random sequence of a direct sequence spread spectrum signal," in *Proc. MILCOM*, vol. 2, Oct. 2000, pp. 967–970.
[11] C. French and W. Gardner, "Spread-spectrum despreading without the code," *IEEE Trans. Commun.*, vol. 34, no. 4, pp. 404–407, Apr. 1986.
[12] L. Chang, F. Wang, and Z. Wang, "Detection of DSSS signal in non-cooperative communications," in *Proc. Int. Conf. Commun. Technol.*, Nov. 2006, pp. 1–4.
[13] S. Davidovici and E. G. Kanterakis, "Radiometric detection of direct-sequence spread-spectrum signals using interference excision," *IEEE J. Sel. Areas Commun.*, vol. 7, no. 4, pp. 576–589, May 1989.
[14] A. Polydoros and C. Weber, "Detection performance considerations for direct-sequence and time-hopping LPI waveforms," *IEEE J. Sel. Areas Commun.*, vol. 3, no. 5, pp. 727–744, Sep. 1985.
[15] C. Bouder, S. Azou, and G. Burel, "A robust synchronization procedure for blind estimation of the symbol period and the timing offset in spread spectrum transmissions," in *Proc. IEEE 7th Int. Symp. Spread Spectr. Techn. Appl.*, vol. 1, Sep. 2002, pp. 238–241.
[16] G. Burel, "Detection of spread spectrum transmissions using fluctuations of correlation estimators," in *Proc. IEEE Int. Symp. Intell. Signal Process. and Commun. Syst.*, Honolulu, HI, USA, vol. 11, Nov. 2000, p. B8.
[17] Q. Wang and Q. Ge, "Blind estimation algorithm of parameters in PN sequence for DSSS-BPSK signals," in *Proc. Int. Conf. Wavelet Act. Media Technol. Inf. Process.*, Dec. 2012, pp. 371–376.
[18] Z. Xiaoming and Z. Zhongzhao, "Parameter estimation of DSSS signals in non-cooperative communication system," *J. Syst. Eng. Electron.*, vol. 18, no. 1, pp. 14–21, Mar. 2007.
[19] Z. Qiu, H. Peng, and T. Li, "A blind despreading and demodulation method for QPSK-DSSS signal with unknown carrier offset based on matrix subspace analysis," *IEEE Access*, vol. 7, pp. 125700–125710, Sep. 2019.
[20] P. Y. Qui, Z. T. Huang, W. L. Jiang, and C. Zhang, "Improved blind-spreading sequence estimation algorithm for direct sequence spread spectrum signals," *IET Signal Process.*, vol. 2, no. 2, pp. 139–146, Jun. 2008.
[21] S. Mehboodi, A. Jamshidi, and M. Farhang, "Spreading sequence estimation algorithms based on ML detector in DSSS communication systems," *IET Signal Process.*, vol. 12, no. 6, pp. 802–809, Aug. 2018.
[22] C. Bouder and G. Burel, "Spread spectrum codes identification by neural networks," in *Proc. Syst. Control, Theory Appl.*, 2000, pp. 257–262.
[23] Y. Wei, S. Fang, X. Wang, and S. Huang, "Blind estimation of the PN sequence of a DSSS signal using a modified online unsupervised learning machine," *Sensors*, vol. 19, no. 2, p. 354, Jan. 2019.
[24] Z. Tianqi, L. Xiaokang, and Z. Zhengzhong, "Neural network approach to blind-estimation of PN spreading sequence in lower SNR DS/SS signals," *J. Syst. Eng. Electron.*, vol. 16, no. 4, pp. 756–760, Dec. 2005.
[25] C. Bouder, S. Azou, and G. Burel, "Performance analysis of a spreading sequence estimator for spread spectrum transmissions," *J. Franklin Inst.*, vol. 341, no. 7, pp. 595–614, Nov. 2004.
[26] E. Moulines, P. Duhamel, J.-F. Cardoso, and S. Mayrargue, "Subspace methods for the blind identification of multichannel FIR filters," *IEEE Trans. Signal Process.*, vol. 43, no. 2, pp. 516–525, Feb. 1995.
[27] B. Yang, "Projection approximation subspace tracking," *IEEE Trans. Signal Process.*, vol. 43, no. 1, pp. 95–107, Jan. 1995.
[28] T. Zhang, S. Dai, W. Zhang, and G. Ma, "Blind estimation of the PN sequence for weak DSSS signals in dynamic environments," in *Proc. 11th IEEE Singap. Int. Conf. Commun. Syst.*, Nov. 2008, pp. 470–474.
[29] L. Wu, Z. Li, J. Li, and C. Chen, "A blind algorithm for estimating pseudo-noise sequence of DSSS signal in lower SNR conditions," in *Proc. 3rd Int. Congr. Image Signal Process.*, vol. 9, Oct. 2010, pp. 4286–4289.
[30] T. Zhang, "Blind estimation of the PN sequence in lower SNR DS/SS signals," *IEICE Trans. Commun.*, vol. E88-B, no. 7, pp. 3087–3089, Jul. 2005.
[31] J. Hagenauer, "The turbo principle: Tutorial introduction and state of the art," in *Proc. Int. Symp. Turbo Codes, Rel. Topics*, Sep. 1997, pp. 1–11.
[32] C. Berrou, J. Hagenauer, M. Luise, C. Schlegel, and L. Vandendorpe, "Special issue on turbo-information processing: Algorithms, implementations & applications," *Proc. IEEE*, vol. 95, no. 6, pp. 1146–1149, Jun. 2007.
[33] X. Wang and H. V. Poor, "Iterative (turbo) soft interference cancellation and decoding for coded CDMA," *IEEE Trans. Commun.*, vol. 47, no. 7, pp. 1046–1061, Jul. 1999.
[34] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *Proc. IEEE Int. Conf. Commun.*, vol. 2, Jun. 1995, pp. 1009–1013.
[35] P. Robertson, "Illuminating the structure of code and decoder of parallel concatenated recursive systematic (turbo) codes," in *Proc. IEEE GLOBECOM*, vol. 3, Nov. 1994, pp. 1298–1303.
[36] G. Colavolpe, G. Ferrari, and R. Raheli, "Extrinsic information in iterative decoding: A unified view," *IEEE Trans. Commun.*, vol. 49, no. 12, pp. 2088–2094, Dec. 2001.
[37] R. Asvadi, A. H. Banihashemi, M. Ahmadian-Attari, and H. Saeedi, "LLR approximation for wireless channels based on Taylor series and its application to BICM with LDPC codes," *IEEE Trans. Commun.*, vol. 60, no. 5, pp. 1226–1236, May 2012.
[38] M. I. Jordan *et al.* (1995). *Why the Logistic Function? A Tutorial Discussion on Probabilities and Neural Networks*. [Online]. Available: https://www.ics.uci.edu/~smyth/courses/cs274/readings/jordan_logistic.pdf
[39] P. Kumar Meher, "An optimized lookup-table for the evaluation of sigmoid function for artificial neural networks," in *Proc. 18th IEEE/IFIP Int. Conf. VLSI System-Chip*, Sep. 2010, pp. 91–95.
[40] S. Mehboodi, A. Jamshidi, and M. Farhang, "A low-complexity near-optimal algorithm for blind estimation of pseudo-noise sequences in DSSS communication systems," in *Proc. 8th Int. Symp. Telecommun. (IST)*, Sep. 2016, pp. 218–221.
[41] C. Liang, W. Fuping, and W. Zanji, "Low complexity method for spreading sequence estimation of DSSS signal in non-cooperative communication systems," *J. Syst. Eng. Electron.*, vol. 20, no. 1, pp. 41–49, 2009.
[42] B. Sklar, *Digital Communications: Fundamentals and Applications*. Englewood Cliffs, NJ, USA: Prentice-Hall, 2016.



HOESANG CHOI received the B.S. and M.S. degrees in electronics engineering from Hanyang University, Seoul, South Korea, in 2014 and 2016, respectively, where he is currently pursuing the Ph.D. degree with the Department of Electronic Engineering. His research interests include wireless communication systems and communication theory.

HICHAN MOON (Member, IEEE) received the B.S. *(summa cum laude)* and M.S. degrees in electronics engineering from Seoul National University, Seoul, South Korea, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, USA, in 2004. He was with Samsung Electronics Company, South Korea, from 1994 to February 2011. In Samsung, he was involved in development of mobile station modems and standardization of wireless cellular systems including 3GPP W-CDMA, LTE, and LTE-Advanced. Since March 2011, he has been an Associate Professor with the Department of Electronic Engineering, Hanyang University, Seoul. His research interests include wireless communication systems, resource allocation, and communication theory. He holds 50 issued U.S. patents, issued 80 Korean patents and several pending.