Abstract: Encrypted global navigation satellite system signals (GNSS) can be tracked using codeless techniques that do not require the knowledge of the spreading code used for signal generation. These techniques can also be applied to binary offset carrier (BOC) modulated signals whose unknown code sequence can be removed through squaring. In this study, an alternative codeless approach based on the cross-correlation principle is considered. Cross-correlation codeless processing is commonly used for tracking signal components on different frequencies, such as the global positioning system (GPS) L1 and L2 P(Y) signals, and it is adapted here to BOC modulations broadcast on a single frequency. A cross-correlation codeless framework is proposed where the BOC signal is split into two data streams that are cross-multiplied to remove the unknown code sequence. Two architectures, open-loop and closed-loop processing, are proposed and analysed. The codeless cross-correlation function is introduced and open-loop processing is used to reconstruct it from the received samples. Closed-loop processing based on cross-correlation phase lock loop (PLL) and delay lock loop (DLL) are used to estimate the signal parameters. The proposed cross-correlation framework is thoroughly analysed theoretically, through simulations and using real data. The analysis shows the effectiveness of the cross-correlation framework.

1 Introduction

Codeless and semi-codeless techniques have been traditionally used to generate measurements from the global positioning system (GPS) L2 P(Y) signal [1, 2] that is encrypted and requires approaches able to operate without prior knowledge of the signal pseudo-random noise (PRN) code. In codeless and semi-codeless techniques, a non-linear operation, such as squaring, is used to remove the impact of the unknown code that is treated as a nuisance component.

More recently, codeless techniques have been adapted to binary offset carrier (BOC) signals [3–6], where the structure of these modulations is exploited for codeless tracking. Codeless techniques can be used to monitor the signal correlation function [3, 4], to assess the code/subcarrier divergence [5, 6] and to generate signal measurements. Moreover, codeless techniques can be used to introduce additional layers of security [7, 8]. In this respect, short data segments of encrypted global navigation satellite system (GNSS) signals can be recorded and used to sign GNSS observables. Techniques based on codeless processing can then be used to verify the authenticity of the signed signals.

Codeless techniques developed for BOC modulations usually track the subcarrier component [9] and integrate it coherently over the duration of a code chip. The impact of the unknown PRN code is then removed through squaring.

In this paper, an alternative approach based on the cross-correlation principle is considered. Cross-correlation codeless processing has been widely used for jointly processing the GPS L1 and L2 P(Y) signals [2], on the 1575.42 and 1227.60 MHz frequencies. In this respect, the L1 P(Y) signal is used to remove the unknown code on the L2 P(Y) component. This principle can also be used for BOC modulated signals broadcast on a single frequency. More specifically, BOC signals are characterised by two symmetric spectral lobes and can be decomposed in two side-band components. This principle is used by different types of side-band BOC processing techniques [10–12], which split BOC signals into two separate data streams. These techniques assume the knowledge of the signal code and use it to despread the received samples. In this way, long coherent integrations can be implemented. This approach cannot be adopted here since the code is unknown and consequently coherent processing cannot be implemented. In this work, side-band processing is used to obtain two BOC side-band components that are treated as the GPS L1 and L2 P(Y) signal pair. In particular, these side-band components are cross-multiplied to remove the impact of the unknown code. Using this principle, a cross-correlation codeless phase lock loop (PLL) is designed along with two code tracking architectures. The first architecture is open-loop and allows one to reconstruct and monitor the signal codeless cross-correlation function (CCCF), which is a function of the received code chips. The second approach is closed-loop and consists of a cross-correlation delay lock loop (DLL) able track, in a codeless way, the code component.

In addition to adapt cross-correlation codeless technique to BOC signals, the paper provides a thorough characterisation of the BOC cross-correlation codeless framework. Open-loop processing is characterised from a theoretical point of view and a closed-form expression for the CCCF is derived. The cross-correlation codeless PLL is also analysed and a closed-form expression for the PLL tracking jitter is provided.

Simulations and real data processing support theoretical findings and show the feasibility of cross-correlation codeless processing. In this respect, a fully software MATLAB receiver has been developed and used to process encrypted Galileo E1a signals with centre frequency equal to 1575.42 MHz. The receiver processes the open Galileo E1c component in a standard way and uses Doppler measurements to aid codeless processing. In this way, long integrations can be implemented in the cross-correlation codeless PLL and DLL.

Real data processing demonstrates the feasibility of cross-correlation codeless techniques that can be effectively used as an alternative to squaring PLL approaches.

The reminder of this paper is organised as follows: the BOC signal model is introduced in Section 2 while cross-correlation codeless processing is described in Section 3. The tracking jitter of the cross-correlation codeless PLL is analysed in Section 4 and simulation results are provided in Section 5. Section 6 describes the experimental results obtained processing real Galileo E1a signals and Section 7 finally concludes the paper.
2 BOCSignals

After amplification, filtering and down-conversion, the analogue BOCSignal modulated
can be modelled as [10, 2]

\( y(t) = C \exp \left[ j2\pi f_{d,0} t + j\phi_0 \right] \)

(1)

where \( C \) is the received signal power, \( \exp (\cdot) \) is the signal spreading
code and \( \exp (\cdot) \) is the BOCSubcarrier. \( \exp (\cdot) \) is a noise term usually
modelled as an additive white Gaussian noise (AWGN). All the
signal components in (1) are functions of the time variable, \( t \). The
communication channel delays the useful signal component and
introduces a Doppler effect. In (1), the delay introduced by the
communication channel is modelled by \( t_0 \), which shifts the signal
code and subcarrier, and by \( \phi_0 \), the additional phase experienced by
the carrier component. In this context, the Doppler effect is the
signal frequency shift perceived by the receiver and caused by the
relative motion between the receiver and the transmitting satellite.

Thus, the subcarrier term in (1) can be written as

\( \exp (\cdot) \) is the signal spreading

code and subcarrier, and by \( \phi_0 \), the additional phase experienced by
the carrier component. In this context, the Doppler effect is the
signal frequency shift perceived by the receiver and caused by the
relative motion between the receiver and the transmitting satellite.

\( y(t) = C \exp \left[ j2\pi f_{d,0} t + j\phi_0 \right] \)

(2)

and

\( \exp (\cdot) \) is the signal spreading

code and subcarrier, and by \( \phi_0 \), the additional phase experienced by
the carrier component. In this context, the Doppler effect is the
signal frequency shift perceived by the receiver and caused by the
relative motion between the receiver and the transmitting satellite.

If only the first term in the Fourier series expansion in (2) is
retained, the subcarrier can be effectively approximated as a pure
sinusoid of fundamental frequency, \( f_{sub} \)

\[ s_{y}(t) = \frac{4}{\pi} \sin(2\pi f_{sub} t) \]

(3)

Thus, the subcarrier term in (1) can be written as

\[ s_{y}(t) = \frac{4}{\pi} \sin(2\pi f_{sub} t) \]

(4)

where \( f_{sub} = \Delta f_{sub} \) is the Doppler frequency on the subcarrier
expressed in Hz and \( \phi_0 = -2\pi f_{sub} \Delta T_s = 2\pi f_{sub} T_s / 2 \)

Using (4), it is possible to express (1) as

\( y(t) = C \exp \left[ j2\pi f_{d,0} t + \phi_0 \right] \)

(5)

\[ y(t) = C \exp \left[ j2\pi f_{d,0} t + \phi_0 \right] \]

(6)

Equation (5) implies that the received baseband BOCSignal modulated
can be made of two symmetric lobes shifted in frequency with
respect to the subcarrier fundamental frequency, \( f_{sub} \) and the
Doppler shifts on the carrier and on the subcarrier. A schematic
spectral representation of (5) is provided in Fig. 1.

The analogue signal, \( y(t) \), is sampled with a frequency \( f_s = 1 / T_s \)
and the digital sequence, \( y[n] = y(n T_s) \), is obtained

\[ y[n] = \begin{cases} 1 & \text{if } n = 0 \\ 0 & \text{otherwise} \end{cases} \]

(7)

Signal \( y[n] \) is obtained by shifting the upper lobe of the BOCSignal
modulation close to baseband. Similarly, \( y[n] \) is obtained by up-
converting the lower BOCSide lobe close to baseband.

In the following, it will be assumed that the code values, \( c[n] \)
are unknown at the receiver side. For this reason, codeless
techniques are adopted. The signals, \( y[n] \) and \( y[n] \), form a pair
similar to that formed by the L1 and L2 P(Y) signals and thus can be
processed using cross-correlation codeless processing [2].

3 Cross-correlation codeless processing

In order to implement cross-correlation codeless techniques, it is
necessary, at first, to integrate data streams (7) over the code chip
duration. It is assumed that the code component in (6) is binary
phase shifting keying (BPSK) modulated and that \( c[n] \) takes values in \{-1, 1\}.
Moreover, \( c[n] \), is assumed approximately constant over a code chip.

A BOCSignal modulation is defined by two parameters [14]: the
subcarrier frequency, \( f_{sub} \) introduced in Section 2, and the code
frequency, \( f_{sc} \), that defines the duration of an individual element of
the spreading code usually denoted as chip. For example, in the
BOC(10, 5) adopted by the M-code modulation [6, 15], the
subcarrier frequency is \( f_{sub} = 10 - 1.023 \) MHz, whereas the code
rate is \( f_{sc} = 5 - 1.023 \) MHz. When the BOCSignal is split into two
separate streams, each stream can be approximated as a BPSK
modulation with code rate equal to \( f_{sc} \). The chip duration is \( T_s = 1 / f_{sc} \).
BOCSignals can be assumed to be approximately constant over this time period.
It is assumed that the start of a code chip is effectively recovered using the cross-correlation DLL described in Section 3.3. In addition to this, it is assumed that $K$, the number of samples integrated over each code chip, is approximately constant. In reality, $K$ varies over time oscillating between $[T_u/T_s]$ and $[T_u/T_f]$ and, for each code chip, the DLL described in Section 3.3 is able to determine the exact value of $K$. The assumption of a constant $K$ is made only to simplify derivations and does not compromise the generality of the results.

Using these assumptions, it is finally possible to compute the integrated data streams

$$
\tilde{y}_d[m] = \frac{1}{K} \sum_{n=mK}^{(m+1)K-1} y[n]e^{-j2\pi f_d n T_s},
$$

$$
\tilde{y}_y[m] = \frac{1}{K} \sum_{n=mK}^{(m+1)K-1} y[n].
$$

The integration/averaging over $K$ samples acts as a low-pass filter that preserves only the signal components close to baseband. This is the integration process mentioned in Section 2 that isolates the two BOC side lobes. In particular

$$
\tilde{y}_d[m] = A \sum_{n=mK}^{(m+1)K-1} c_0[n]e^{j2\pi f_d n T_s},
$$

$$
\tilde{y}_y[m] = A \sum_{n=mK}^{(m+1)K-1} c_0[n]e^{j2\pi f_d n T_s} + j\phi_0 + \eta_0[m].
$$

The second signal component from $y[n]$ is shifted to high frequencies by (7) and it is averaged out by the summation in (8). The noise term, $\eta_0[m]$, is given by

$$
\eta_0[m] = \frac{1}{K} \sum_{n=mK}^{(m+1)K-1} \eta(nT_s) \exp \{-j2\pi f_d n T_s\}
$$

and defines an AWGN with variance reduced by a factor $K$ with respect to $\eta(nT_s)$. The properties of $\eta_0[m]$ are better analysed in Section 4.

Similarly

$$
\tilde{y}_y[m] = A \sum_{n=mK}^{(m+1)K-1} c_0[n]e^{j2\pi f_d n T_s} + j\phi_0 + \eta_0[m].
$$

$\eta_0[m]$ has the same statistical properties of $\eta_0[m]$.

Under the assumption that $c_0[n]$ is constant over the integration interval, it is possible to show that $\tilde{y}_y[m]$ and $\tilde{y}_y[m]$ are given by

$$
\tilde{y}_d[m] = A c_0[mK] \sin(\pi f_d m KT_s) / K \sin(\pi f_d m T_s),
$$

$$
\tilde{y}_y[m] = A c_0[mK] \sin(\pi f_d m KT_s - \pi \phi_0 + \eta_0[m]).
$$

The product between the chip integration time, $KT_s$, and the maximum Doppler frequencies $f_{d,t}$ is small and closed to zero. Thus, the Dirichlet kernels in (13) are approximately equal to

$$
\sin(\pi f_d m KT_s) / K \sin(\pi f_d m T_s) \approx 1.
$$

For the same reason, the additional phase introduced during the integration process can be neglected as in (14).

Cross-correlation codeless processing is obtained by multiplying $\tilde{y}_d[m]$ and $\tilde{y}_y[m]$

$$
d[m] = \tilde{y}_d[m] \tilde{y}_y[m] = A^2 c_0[mK] \exp \{j4\pi f_d m KT_s + j2\phi_0 + \eta_0[m] + j\phi_0 + \eta_0[m].
$$

where the BPSK assumption, $\tilde{c}_0[m] = 1$, has been used. The signal, $d[m]$, contains a pure tone at twice the carrier Doppler frequency of the input signal, $\eta[n]$. The signal carrier phase, $\phi_0$, is preserved. Thus, it is possible to use a PLL to recover the signal parameters, $f_d, D, \phi_0$. The noise term, $\eta_0[m]$, is given by the sum of $\eta_0[m]\eta_0[m]$ with the cross-terms between signal and noise components. The properties of $\eta_0[m]$ are better analysed in Section 4.

A schematic representation of cross-correlation codeless processing for tracking the carrier parameters is shown in Fig. 2. The residual carrier error is removed from $d[m]$ through the multiplication by the local carrier generated by the numerically controlled oscillator (NCO). The local carrier is generated using the previous estimates of the Doppler frequency and carrier phase. After carrier removal, $d[m]$ is integrated over $M$ epochs and a cross-correlation correlator is obtained.

In particular, the prompt cross-correlation codeless correlator is obtained as

$$
P_k = \frac{1}{M} \sum_{m=0}^{M-1} d[m]
$$

where subscript $k$ denotes different time epochs. Each epoch has duration $KT_s$. $P_k$ is used to compute the discriminator output

$$
\hat{\phi}_{d,k} = \frac{1}{2} \tan^{-1}(\Re\{P_k\}, \Im\{P_k\}).
$$

The error signal, $\hat{\phi}_{d,k}$, is filtered and used to update the NCO that produces a new estimate of the signal carrier frequency and generates a new local carrier for the next processing epoch. In Fig. 2, $H(z)$ denotes the transfer function of the filter used to process the discriminator output (18).

3.1 Frequency aiding from unencrypted channels

Cross-correlation codeless processing requires long integration times, $MKT_s$, and thus the PLL used for the frequency and phase recovery needs a small loop bandwidth [13, 16]. While this requirement ensures loop stability, it also reduces the ability of the
loop to track fast changes in the signal parameters. Thus, only reduced phase and Doppler frequency dynamics can be tracked. This condition can be achieved by implementing frequency aiding [17] from unencrypted channels. The Galileo Public Regulated Service (PRS) and GPS M-code are broadcast along with open signals that are affected by the same Doppler shift. Frequency aiding consists in properly scaling the Doppler frequency estimated on an open channel and injecting it in the codeless processing loop. In this way, the dynamics sensed by the codeless loop is significantly reduced. Frequency aiding is commonly used in codeless loops [2, 3, 5] and it is adopted here for cross-correlation processing.

3.2 Open-loop processing

In the previous sections, it was assumed that the start of the code chips was known or recovered using a DLL. When the start of the code chip is unknown, it is possible to use open-loop processing where a bank of correlators is adopted to test several delay values in parallel. Open-loop processing is used for monitoring the correlation function of encrypted signals [4–6] and it is adapted here to cross-correlation processing.

Open-loop cross-correlation processing of BOC modulated signals is depicted in Fig. 3. Several branches are present in Fig. 3: for each branch a different delay is tested and the coherent accumulation process on the code chips of \( y_0[n] \) and \( y_1[n] \) is delayed by a different value. The outputs of the two coherent accumulation blocks are then multiplied and further integrated after Doppler removal. The Doppler frequency can be estimated using the PLL in Fig. 2. In this way, the empirical CCCF is sampled at different delay instants. The delays should be selected in the \([0, T_{sc}]\) interval.

In Appendix, the ideal CCCF, determined in the presence of perfect carrier recovery, is derived. The CCCF is a periodic function of period \( T_{sc} \) and it is given by (Appendix)

\[
R_d(\tau) = \frac{A^2}{2} \left[ C_c(T_{sc}) + C_c(T_{sc}) - 2C_c(\tau) \right] \quad (19)
\]

where

\[
C_c(\tau) = \frac{1}{T_{sc}} \int_0^{T_{sc}} c(t) \, dt \quad (20)
\]

Under ideal conditions, i.e. when the code chip is modelled as a rectangular pulse, \( R_d(\tau) \) becomes

\[
R_d(\tau) = \frac{A^2}{2} \left[ 1 + \left( 1 - 2 \frac{\tau}{T_{sc}} \right)^2 \right] \quad (21)
\]

The block processing scheme depicted in Fig. 3 allows one to estimate \( R_d(\tau) \) and monitor the properties of the filtered code chip, \( c(t) \).

3.3 Cross-correlation DLL

The CCCF obtained using real data is a noisy and delayed version of (19). It has a maximum for \( \tau = \tau_0 \) and exhibits a symmetric peak that can be maximised using a DLL [18]. The principle is the same as that used in standard tracking loops: early and late correlators can be used to maximise the CCCF. A schematic representation of the DLL adopted to recover the start of a code chip in cross-correlation processing is shown in Fig. 4. A nominal delay, \( \tau \), is used to compute the prompt correlator. More specifically, \( \tau \) determines the start of the prompt accumulation process. The early and late correlators are computed in a similar way considering \( \tau \pm \frac{d_s}{2} \) as start of their accumulation processes. \( d_s \) is the early-minus-late code spacing in seconds. The early and late correlators are obtained according to the same processing described above for the prompt correlator and are denoted as \( E_k \) and \( L_k \).

The discriminator output is computed from the early and late correlators as

\[
D(\tau) = \frac{1}{G(d_s, T_{sc})} \frac{[E_0] - [L_0]}{[E_0] + [L_0]} \quad (22)
\]

where

\[
G(d_s, T_{sc}) = \frac{1}{\tau_0 T_{sc}} \int_0^{\tau_0} \left[ 1 + \left( 1 - 2 \frac{\tau}{T_{sc}} \right)^2 \right] \, d\tau
\]
\[ G(d_s, T_{sc}) = \frac{4}{T_{sc}} \left( 1 - (d_s/T_{sc}) \right) \]  

(23)

is a normalisation constant selected to have a discriminator function with unit gain. \( G(d_s, T_{sc}) \) has been computed assuming an ideal rectangular chip and a CCCF equal to (21).

The discriminator output passes through the loop filter, denoted in Fig. 4 by the transfer function \( H(z) \), and a new estimate of the code rate, \( \tilde{f}_{d,sc} \), is obtained. Finally, the NCO updates, \( \tau \), which indicates the start of the accumulation process and determines the number of samples used in the coherent integration process, \( K \). In particular, \( \tau \) is updated as

\[
\tau_{m+1} = \text{mod}(\tau_m + K_m \tilde{T}_s, T_s) + 1.
\]  

(24)

where the index \( m \) has been added to indicate the different integration epochs and ‘mod’ denotes the modulus operator with respect to the code chip duration, \( T_{sc} \). \( \tilde{T}_s \) is given by

\[
\tilde{T}_s = \left( 1 + \frac{\tilde{f}_{d,sc}}{T_{sc}} \right) T_s
\]  

(25)

and represents the duration of a sampling interval scaled in order to take into account the effect of the code Doppler, \( \tilde{f}_{d,sc} \), obtained at the output of the loop filter.

Finally, \( K_m \) is the number of samples accumulated during the \( m \)th epoch. While in the previous sections, \( K \) was considered constant to simplify mathematical derivations, in a real implementation, it is updated at each accumulation epoch and it is given by

\[
K_m = \left[ \frac{T_{sc} - \tau_m}{T_s} \right] + 1.
\]  

(26)

4 Tracking jitter

In this section, the tracking jitter obtained for cross-correlation codeless processing is provided. Only the codeless PLL described in Section 3 is considered.

The tracking jitter determines the amount of noise transferred from the input signal, \( y[n] \), to the final phase estimate and it is defined as \( \sigma_\phi \)

\[
\sigma_\phi = \frac{\sigma_d}{G_d} \sqrt{\frac{K M T_s}{B_{eq}}}
\]  

(27)

where \( \sigma_\phi \) is the tracking jitter, \( \sigma_d \) is the standard deviation of the discriminator output (18) and \( G_d \) is the discriminator gain. \( T_s = K M T_s \) is the loop update rate and \( B_{eq} \) is the loop equivalent bandwidth. \( B_{eq} \) is a loop design parameter and is used to determine the coefficients of the loop filter [13]. \( G_d \) is defined as

\[
G_d = \frac{\partial E_{\phi_{d,k}}}{\partial \phi} \bigg|_{\phi = 0} = \frac{\partial S(\phi)}{\partial \phi} \bigg|_{\phi = 0}
\]  

(28)

where \( S(\cdot) \) is the loop discriminator input–output function. The correlator outputs, \( P_{k} \), are functions of the residual carrier phase error, which is denoted here as \( \phi \). Discriminator (18) is used to extract an error signal from the correlator output. In particular, \( \phi_{d,k} \) in (18) is, in general, a non-linear function of the input residual phase, \( \phi \), and it is affected by noise

\[
\phi_{d,k} = S(\phi + \eta_{\phi}) \approx S(\phi) + \eta_{S} \approx G_d \cdot \phi + \eta_{S}
\]  

(29)

where \( \eta_{\phi} \) is the noise at the input of the discriminator function propagated from \( y[n] \) through the prompt correlator and \( \eta_{S} \) is the discriminator output noise. \( G_d \) is used to linearise the discriminator function and approximate the PLL as a linear system.

![Fig. 4 Schematic representation of the DLL adopted to recover the start of a code chip in cross-correlation codeless processing](http://creativecommons.org/licenses/by/3.0/)

2002  
IET Radar Sonar Navig., 2019, Vol. 13 Iss. 11, pp. 1998-2007  
This is an open access article published by the IET under the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0/)
For the derivations, it is assumed that the input noise, \( n(t) \) in (6) is an AWGN with independent real and imaginary parts each with variance [3]

\[
\sigma^2 = N_0 B_{RX}\tag{30}
\]

where \( N_0 \) is the power spectral density of the input noise and \( B_{RX} \) is the one-sided receiver equivalent bandwidth. In the following

\[
B_{RX} = \frac{f_s}{2}\tag{31}
\]

is assumed. Since the operations performed to obtain \( \eta_n[m] \) and \( \eta_i[m] \) are linear, it follows that these noise terms are also Gaussian. Moreover, they have real and imaginary parts each with variance \( \sigma^2/2 \).

The noise affecting the product signal, \( d[m] \), is given by

\[
\eta_n[m] = \eta_r[m] \eta_i[m] + \eta_i[m] s[m] + s[m] \eta_r[m]\tag{32}
\]

where

\[
\begin{align*}
s[m] &= \mathcal{E}\{\tilde{d}[m]\} = A C_0 e^{j \omega f_s t + j \Omega_0 t} + j \omega f_s t \nonumber \\
        &\quad + \mathcal{E}\{d[m]\} \\
\tilde{s}[m] &= \mathcal{E}\{\tilde{d}[m]\} = A C_0 e^{j \omega f_s t + j \Omega_0 t} + j \omega f_s t
\end{align*}
\]

Using these results, it follows:

\[
\begin{align*}
\mathcal{E}\{\eta_n[m]\} &= 0 \\
\text{Var}\{\eta_n[m]\} &= \text{Var}\{\eta_r[m] \eta_i[m]\} + \text{Var}\{\eta_i[m] s[m]\} + \text{Var}\{s[m] \eta_r[m]\} \\
&= \text{Var}\{\eta_r[m]\} \text{Var}\{\eta_i[m]\} + \text{Var}\{\eta_i[m]\} \text{Var}\{s[m]\} + \text{Var}\{s[m]\} \text{Var}\{\eta_r[m]\} \\
&= 4 \sigma^2 + A^2 \sigma^2 \nonumber \\
&= 2 \sigma^2 + 2 A^2 \nonumber \\
&\approx 4 \sigma^2/\mathcal{K} + 2 A^2/\mathcal{K} = 2 \sigma^2 (A^2 + 2 \sigma^2/\mathcal{K}) \approx 4 \sigma^2/\mathcal{K}.
\end{align*}
\]

The variance of \( \eta_n[m] \) has been computed taking into account the fact that the three terms in (32) are uncorrelated since their products lead to monomials of odd order of zero mean Gaussian random variables. Moreover, the variance in (34) is the total variance of \( \eta_n[m] \) and it is equally split between the real and imaginary parts of \( \eta_n[m] \). The product signal, \( d[m] \), is multiplied by the local carrier at twice the Doppler frequency and is integrated over \( M \) samples. This process leads to the correlator output, \( P_i \). Under the assumption that the PLL perfectly recovers the Doppler frequency, \( f_s t + \Omega_0 t \), \( P_i \) is characterised by

\[
\begin{align*}
\mathcal{E}\{P_i\} &= A^2 e^{j \Omega_0 t} \\
\text{Var}\{P_i\} &= \frac{1}{M} \text{Var}\{\eta_n[m]\} \approx 4 \sigma^2 / M \mathcal{K}^2.
\end{align*}
\]

Using the results in (35), it is finally possible to compute the post-correlation signal-to-noise power ratio (SNR)

\[
R_i = \frac{\mathcal{E}\{P_i\}^2}{\text{Var}\{P_i\}} = \frac{A^4}{8 \sigma^2 / M \mathcal{K}^2} = \frac{16 \mathcal{C}^2 MK^2}{\pi^2} \mathcal{K}^2 = \frac{32 MK^2 \mathcal{C}^2}{\pi^2 N_0 K T} \mathcal{L}_d = L_d \left( \frac{C}{N_0} K T \right)^2 \tag{36}
\]

with

\[
L_d = \frac{32}{\mathcal{K}} = 0.3182. \tag{37}
\]

The tracking jitter can be directly computed using (36) and the results obtained in [2, 3]. In particular, in [2] it is shown that for a four-quadrant discriminator as in (18), the normalised standard deviation of the discriminator output is given by

\[
\sigma_d = \frac{1}{C_d} \left( \frac{R_0 + 1}{R_i} \right)^{1/2}. \tag{38}
\]

In this way, the tracking jitter of the cross-correlation codeless PLL is given by

\[
\sigma_d = \frac{\sqrt{2 P_{B0} K M T} + L_d M (C/N_0 K T)^2}{2L_d C/N_0 (C/N_0 K T)^2} \tag{39}
\]

### 5 Simulation results

In this section, simulations are used to support the theoretical results developed in Section 4. Two types of simulations are used: full Monte Carlo simulations have been adopted to verify the validity of (36) whereas semi-analytic simulations [19] have been used to analyse the tracking jitter and (39).

#### 5.1 Post-correlation SNR

Full Monte Carlo simulations have been used to estimate the post-correlation SNR and support the validity of (36). A synthetic GNSS signal was generated using the parameters reported in Table 1. The signal was modulated by a cosine BOC(15, 2.5) that is the modulation adopted by the Galileo E1a component. The signal was modulated with a random code and a sampling frequency, \( f_s = 40 \text{ MHz} \), was considered. The simulations were performed as a function of the input carrier-to-noise power spectral density ratio \( C/N_0 \). For each \( C/N_0 \) value, AWGN was added to the simulated GNSS signal samples. The noise variance was determined from the input \( C/N_0 \) and (30). For each \( C/N_0 \) value, \( 10^5 \) simulation runs were performed. For each simulation run, a prompt correlator, \( P_k \), was evaluated. The mean and standard deviation of the correlators were finally estimated from the \( P_k \) obtained through Monte Carlo simulations. The correlators were computed using the processing schemes described in Section 3 and assuming perfect carrier, subcarrier and code synchronisation. Indeed the goal of these type of simulations was to verify the validity of (36) and not to test the proper functioning of the cross-correlation codeless loops. These loops are tested through simulations in Section 5.2 and by processing real GNSS data in Section 6.

Simulation results related to the post-correlation SNR are provided in Fig. 5, which compares simulation and theoretical findings. A good agreement between theoretical and simulation results is obtained supporting the validity of (36): the post-integration SNR increases linearly as a function of the input \( C/N_0 \). The slope of the SNR curve in Fig. 5 is equal to 2 and it is determined by the squaring factor in (36).
Considered along with a total integration time, the validity of the theory developed in Section 4. In this framework, simulations were conducted as a function of the input results were used only for the evaluation of perfect code alignment was assumed for the analysis of the PLL jitter. For the simulation, a cosine BOC(15, 2.5) was adopted by the Galileo E1a signal. The characteristics of the BOC signal used for the analysis are summarised in Table 3 along with the different processing parameters adopted in the software receiver.

### 6 Experimental results

In order to demonstrate the feasibility of cross-correlation codeless processing, real Galileo E1a data have been collected and processed using the approaches detailed in Section 3. A fully software MATLAB receiver able to process Galileo E1 signals has been developed. Standard processing has been used to acquire and track the E1c pilot signal that has been used to aid codeless processing.

Data have been collected using a National Instruments (NI) PXIe-5663 Radio Frequency Signal Analyser configured according to the parameters listed in Table 2. A sampling frequency equal to 40 MHz was selected in order to be able to collect the full cosine BOC(15, 2.5) adopted by the E1a signal [9].

### Table 2 Parameters used for the collection of Galileo E1a data

| Parameter          | Value                  |
|--------------------|------------------------|
| sampling frequency | $f_s = 40\text{MHz}$  |
| signals            | Galileo E1             |
| signal centre frequency | 1575.42 MHz            |
| intermediate frequency | 0 Hz                   |
| sample type        | Complex I/Q            |
| bits per sample    | 16                     |

Jitter through simulations.

**Fig. 5** Post-correlation SNR: comparison between simulation and theoretical results

**Fig. 6** Tracking jitter of the cross-correlation codeless PLL: comparison between theoretical and simulation results

### 5.2 Tracking jitter

Semi-analytic simulations [19] have been used to support the validity of the theory developed in Section 4. In this framework, analytic results on the statistical properties of the prompt correlators (35) were used to reduce the computational load that full Monte Carlo would have required. The statistical properties of $P_k$ were assessed in Section 5.1 and used to simulate the correlator outputs that are a function of the input $C/N_0$, of the assumed signal Doppler frequency and phase, and of the NCO output. Analytic results were used only for the evaluation of $P_k$, whereas all the other elements of the cross-correlation codeless PLL described in Section 3 were implemented and used to determine the tracking jitter through simulations.

A second-order PLL with different loop bandwidths was considered along with a total integration time, $T_c$, equal to 400 ms. Perfect code alignment was assumed for the analysis of the PLL tracking jitter. For the simulation, a cosine BOC(15, 2.5) was considered since this modulation is adopted by the Galileo E1a signal. Simulations were conducted as a function of the input $C/N_0$ and for each $C/N_0$ value, $10^6$ simulation runs were conducted. During the simulations, a constant Doppler frequency was assumed and a linearly varying carrier phase was generated. For each epoch, the phase estimated by the loop was compared with that generated by the simulation scenario and used to compute the residual phase error that, in turn, was used to estimate the tracking jitter.

Results are provided in Fig. 6, which compares the tracking jitter obtained through simulations with (39). A good agreement between theoretical and simulation results is obtained for high/moderate $C/N_0$ values. This fact is expected since theoretical results have been obtained assuming signal lock conditions. Using this assumption, it is possible to linearise the effects of the loop discriminator (see (29)) and obtain a closed-form expression for the tracking jitter. Thus, (39) is unable to predict non-linear effects, such as loss of lock, which occur at low $C/N_0$ values. The practically vertical dashed lines observable in Fig. 6 represent the tracking thresholds of the cross-correlation codeless PLL, i.e. the lowest $C/N_0$ values for which the PLL is able to maintain signal lock. The tracking threshold is a function of the integration time and of the loop bandwidth. Lower loop bandwidths reduce the tracking jitter and lead to lower tracking threshold.

Similar results were obtained by considering different combinations of integration times and loop bandwidths. They are not reported here to avoid the repetition of similar results. In all cases and under lock conditions, a good agreement between simulations and theoretical results was obtained. The semi-analytic framework used here supports the validity of the theoretical findings provided in Section 4 and is able to predict the non-linear behaviour of the cross-correlation tracking PLL.

### Table 3 Signal characteristics and processing parameters adopted for the Galileo E1a signal

| Parameter          | Value                  |
|--------------------|------------------------|
| signal type        | cosine BOC(15, 2.5)    |
| subcarrier frequency | 15.1023 MHz            |
| code rate          | 2.5⋅1023 MHz           |
| code chip duration | 0.391 μs               |
| total integration time | 400 ms                |
| PLL order          | 1                      |
| PLL bandwidth      | 0.3 Hz                 |
| DLL order          | 1                      |
| DLL bandwidth      | 0.1                    |
| early-minus-late spacing | $T_{el}$              |

$$f_{sc} = 2.5 \cdot 1.023 \cdot 10^6, T_c = 400 \text{ ms}$$

$$T_c = MT_{sc} = 400 \text{ ms}$$

### References

2004 IET Radar Sonar Navig., 2019, Vol. 13 Iss. 11, pp. 1998-2007

This is an open access article published by the IET under the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0/)
The developed software receiver is able to process the Galileo E1a signal in both open-loop and closed-loop mode. When operating in open-loop mode, a bank of correlators is computed for different delay values. The correlators are then used to estimate the Galileo E1a CCCF. Sample results for cross-correlation codeless open-loop processing are provided in Fig. 7 that shows the CCCF computed using a real Galileo E1a signal from Satellite Vehicle (SV) 12. In this case, 21 correlators were used to span a code chip period. The reference delay, corresponding to the zero in the code delay axis in Fig. 7, is the delay estimated from the E1c channel that is used for frequency/delay aiding. The total integration time was equal to 400 ms.

In this case, the CCCF is S-shaped and its amplitude oscillates between a minimum value of about 700 and a maximum of about 1400. Note that the amplitude scaling of the CCCF depends on the quantisation applied to the input samples and on the specific implementation of the accumulation/correlation process. In this respect, the CCCF amplitudes are numerical values without a specific unit of measurement. It was however important to remark that the CCCF maximum and minimum values are approximately related by a factor 2. This result was predicted by the theory developed in Section 3.2 and in Appendix.

The cross-correlation codeless PLL was detailed along with open- and closed-loop processing. Open-loop processing can be adopted to monitor the CCCF of encrypted signals, which is a function of the received code chips. Open-loop processing was also characterised from a theoretical point of view and a closed-form expression for the CCCF was derived. Closed-loop processing was obtained by implementing a cross-correlation codeless DLL based on early/late processing.

The cross-correlation codeless PLL was analysed from a theoretical point of view and a closed-form expression for the PLL parameters used for the tracking loops are provided in Table 3. Since Doppler aiding from the E1c signal was implemented, it was possible to adopt first-order loops with narrow bandwidths.

Results related to cross-correlation codeless tracking are provided in Fig. 8, which shows the cross-correlation correlator outputs as a function of time. Fig. 8a shows the absolute values of the early, prompt and late correlators. After an initial transient, the DLL is able to balance the early and late components that assume similar amplitudes. The prompt correlator has an amplitude greater than that of the other two correlators showing the proper functioning of the codeless DLL.

The prompt correlator is further analysed in Fig. 8b, which shows the in-phase and quadrature components of the of prompt correlators. After an initial transient, the codeless PLL is able to recover the residual phase of the E1a signal carrier and concentrate the signal energy on the in-phase component of the correlator. The real part of the prompt correlator always assumes positive values. This fact is expected and is due to filtering effects at both transmission and reception side. Under real conditions, the code chips cannot be approximated as perfect rectangular pulses and the CCCF is less sharp than (21). In the limit case, when only the first harmonic term in the Fourier series expansion of a rectangular pulse is retained, the CCCF assumes the following form:

\[ R_c(\tau) \approx \frac{A_1}{\pi} \left[ 1 + \cos\left(\frac{\pi}{T_{sc}} \tau\right) \right]. \tag{40} \]

The results in Fig. 7 are in agreement with (40) and the theoretical findings presented in Section 3.2.

The MATLAB software receiver used to demonstrate cross-correlation codeless techniques also implements closed-loop processing with the PLL and DLL detailed in Section 3. The parameters used for the tracking loops are provided in Table 3. Since BOC signals can be decomposed into two independent data streams that can be used in a way similar to that commonly adopted for the GPS L1 and L2 P(Y) signals, it was adapted here to BOC modulations broadcast on a single frequency. It was shown that BOC signals can be handled by the codeless PLL in the same way as that commonly adopted for the GPS L1 and L2 P(Y) signal pair.

The cross-correlation codeless PLL was detailed along with open- and closed-loop processing. Open-loop processing can be adopted to monitor the CCCF of encrypted signals, which is a function of the received code chips. Open-loop processing was also characterised from a theoretical point of view and a closed-form expression for the CCCF was derived. Closed-loop processing was obtained by implementing a cross-correlation codeless DLL based on early/late processing.

The cross-correlation codeless PLL was analysed from a theoretical point of view and a closed-form expression for the PLL tracking jitter was provided.

Monte Carlo and semi-analytic simulations supported the theoretical findings and provided additional insights on the characteristics of cross-correlation codeless processing.

Finally, real Galileo E1 data were used to demonstrate the feasibility of cross-correlation processing that can be effectively used to track encrypted BOC modulations.

7 Conclusions

In this paper, cross-correlation codeless processing was considered as an alternative to the squaring PLL usually adopted for monitoring encrypted BOC signals. While cross-correlation codeless processing is commonly used for tracking signal components on different frequencies, such as the GPS L1 and L2 P(Y) signals, it was adapted here to BOC modulations broadcast on a single frequency. It was shown that BOC signals can be handled by the codeless PLL in the same way as that commonly adopted for the GPS L1 and L2 P(Y) signal pair.

The cross-correlation codeless PLL was detailed along with open- and closed-loop processing. Open-loop processing can be adopted to monitor the CCCF of encrypted signals, which is a function of the received code chips. Open-loop processing was also characterised from a theoretical point of view and a closed-form expression for the CCCF was derived. Closed-loop processing was obtained by implementing a cross-correlation codeless DLL based on early/late processing.

The cross-correlation codeless PLL was analysed from a theoretical point of view and a closed-form expression for the PLL tracking jitter was provided.

Monte Carlo and semi-analytic simulations supported the theoretical findings and provided additional insights on the characteristics of cross-correlation codeless processing.

Finally, real Galileo E1 data were used to demonstrate the feasibility of cross-correlation processing that can be effectively used to track encrypted BOC modulations.
The ideal CCCF is defined as
\[ R(\tau) = E[\tilde{y}_c[m] \tilde{y}_c[m]] = \mathbb{E}\left[ \sum_{n=0}^{mK-1} c(nT_s - \tau) \sum_{n=0}^{mK-1} c(nT_s - \tau) \right] \]

where the definitions of \( \tilde{y}_c[m] \) and \( \tilde{y}_c[m] \) in (9) and (11) have been used taking into account that the carrier/subcarrier Doppler frequencies and phases are perfectly recovered. The noise terms in (9) and (11) are zero mean and thus are removed when computing the expected value in (41). The accumulation process is depicted in Fig. 9: continuous lines represent the code chips whereas the dashed rectangle represents the accumulation interval. The signal code, \( c_c[m] \), is unknown and can be considered as a sequence of pulses modulated by independent random variables taking value in \([-1, 1]\). The models the total delay between the start of a code chip and the start of the accumulation interval.

When the accumulation over a chip duration is considered, two possibilities can occur: either the two chips over which the accumulation is performed have the same sign or a chip transition occurs. The first case, which occurs with a 0.5 probability, is depicted in Fig. 9a. In this case, the accumulation process is independent from \( \tau \) and

\[ \frac{1}{K} \sum_{n=0}^{mK-1} c(nT_s - \tau) \]

where \( nT_s \) is defined in (20) and \( C(\cdot) \) represents the random sign of the two consecutive chips. \( C(\cdot) \) is defined in (20) and \( C(T_w) \) represents the average height of a chip. Under ideal conditions, i.e. in the absence of filtering, \( C(T_w) = 1 \).

The second case, when a chip transition occurs, is depicted in Fig. 9b. In this case, the accumulation process is independent from \( \tau \) and

\[ \frac{1}{K} \sum_{n=0}^{mK-1} c(nT_s - \tau) \]

\[ \approx \frac{1}{T_w} \int_{-T_w/2}^{T_w/2} c(\tau)d\tau = b[m]C(T_w) \]

\[ = b[m]C(T_w) - 2C_\tau \]

where \[ \approx \frac{1}{T_w} \int_{-T_w/2}^{T_w/2} c(\tau)d\tau = b[m]C(T_w) \]

\[ = b[m]C(T_w) - 2C_\tau \]

In the absence of filtering, when a chip is approximated as a perfect square pulse,

\[ \approx \frac{1}{T_w} \int_{-T_w/2}^{T_w/2} c(\tau)d\tau = b[m]C(T_w) \]

\[ = b[m]C(T_w) - 2C_\tau \]
The delay, $\tau$, should be always considered in the $[0, T_{sc}]$ range. This is due to the fact that when $\tau$ exceeds $T_{sc}$, the subsequent chip should be considered to measure the delay. Since the spreading code is unknown, it is possible to measure delays only modulo $T_{sc}$.

Results (42) and (43) can be written as

$$
\sum_{n=0}^{K} c(nT_s - \tau) = b[m](C_c(T_{sc}) - 2B[m]C_c(\tau))
$$

where $B[m]$ is a Bernoulli random variable taking value in $\{0, 1\}$ with equal probability. $B[m]$ models the fact that chip transitions can occur or not with equal probability. Using (45), (41) finally becomes

$$
R_d(\tau) = A^2E[b[m](C_c(T_{sc}) - 2d[m]C_c(\tau))^2]
= A^2E[C_c^2(T_{sc}) - 4d[m]C_c(T_{sc})C_c(\tau) + 4d^2[m]C_c^2(\tau)]
= A^2[C_c^2(T_{sc}) - 2C_c(T_{sc})C_c(\tau) + 2C_c^2(\tau)]
$$

(46)