Combining secondary code correlations for fast GNSS signal acquisition

Jérôme Leclère, René Jr Landry
LASSENA, Electrical Engineering Department
École de Technologie Supérieure (ÉTS)
Montréal, Canada
jerome.leclere@lassena.etsmtl.ca

***Abstract*—The secondary codes of the modern GNSS signals bring some notable advantages, however they constitute a challenge for the acquisition process. Indeed, it becomes much more difficult to extend the coherent integration time with these codes. Several methods have been proposed for increasing the coherent integration time when there is a secondary code. Basically, the methods fall into two categories : 1) Methods with long coherent integration times, which require synchronization with the secondary code and imply a significant computational burden, and 2) Methods with short coherent integration times, which test all possible combinations for the secondary code. Since this leads to an exponential increase in the number of combinations, the coherent integration time remains limited, while non-coherent integrations are not usable. Therefore, there is currently no effective solution with intermediate coherent integration time, which would enable moderate to high sensitivity, while maintaining a reasonable level of complexity. In this paper, a method is proposed to address this problem. The method combines secondary code correlations to reduce the number of possible secondary code delays and reduce the complexity. In exchange, there is a loss in the signal-to-noise ratio as compared to the full secondary code correlation. It is shown that the proposed method offers similar or better performance than the short integration times method, in addition to offering the possibility of using non-coherent integrations, and offers lower complexity than the traditional long integration times method with greater sensitivity.**

Keywords—acquisition; correlation; complexity reduction; GNSS, secondary code; SNR

I. INTRODUCTION

Modern GNSS signals introduce new features over the legacy GPS L1 C/A signal, such as subcarriers, a pilot channel, and secondary codes. The secondary code makes synchronization with the data easier, offers better cross-correlation between satellites or constellations; and better interference mitigation. However, secondary codes complicate the acquisition for two reasons : 1) There is now a potential sign transition between every period of the primary code; 2) They make the extension of the coherent integration time more difficult [1-5]. It is the second problem that interests us in this paper. Current literature proposed two main methods for dealing with the extension of the coherent integration time in the presence of a secondary code : 1) Synchronization with the secondary code by computing the correlation with this code, thus enabling long coherent integration times and high sensitivity, but involving a significant computational burden [5-9]; 2) Testing all possibilities for sign transitions between consecutive primary code correlations, but since the number of possibilities grows exponentially, this solution is typically limited to short coherent integration times (maximum of 4 or 5 primary code periods) [10-12]. Moreover, non-coherent integration cannot be performed, or more accurately, it is not interesting since the number of possibilities has a double exponential grows, as will be demonstrated later.

Therefore, there is currently no effective solution with intermediate coherent integration time, to enable moderate to high sensitivity, while maintaining reasonable complexity. In this paper, a method is thus proposed to address this problem. The main idea is to combine secondary code correlation results in order to :

- Reduce complexity by testing fewer delays;
- Reduce complexity by combining inputs, which means fewer fast Fourier transforms (FFTs) for the primary code correlation to compute and a shorter local secondary code;
- Potentially reduce complexity because the local secondary code (modified by the combinations) may contain some zeros, which may eliminate some operations.

Moreover, non-coherent integrations can be used with this method.

However, this complexity reduction does not come without cost. Since the result is modified, there is a loss in the signal-to-noise ratio (SNR) as compared to the full secondary code correlation. This loss can be easily expressed through an equivalent coherent integration time. In a nutshell, the same amount of input signal is used as with a full secondary code correlation (e.g., 20 ms), however the number of operations, or the processing time, or the required resources, will be much lower (e.g., 4 times lower), but the coherent integration time is also lower (e.g., 12 ms). The goal is for the complexity reduction to be greater than the coherent integration time reduction. Note that since results are combined, there is still some ambiguity in the secondary code delay. However it is not difficult to resolve once the primary code delay and the Doppler frequency are correctly estimated.

Also, these combinations cannot be made randomly. Let us illustrate this without going into too much detail, with a simple example that compares two different combinations with the

following 6-chip code [1 1 1 –1 1 –1], whose autocorrelation is $[y_0\ y_1\ y_2\ y_3\ y_4\ y_5]$ = [6 –2 2 –2 2 –2]. If two correlation results are added, the combinations $[y_0+y_3\ \ y_1+y_4\ \ y_2+y_5]$ or $[y_0+y_1\ \ y_2+y_3\ \ y_4+y_5]$ will both have [4 0 0] as result (with a certain shift, depending on the delay of the incoming code). However, if three correlation results are added, the combination $[y_0+y_2+y_4\ \ y_1+y_3+y_5]$ gives [10 –6], whereas the combination $[y_0+y_1+y_2\ \ y_3+y_4+y_5]$ gives [6 –2] or [2 2] according to the delay of the incoming code. In the last case, this means e.g. that if the incoming code is [1 –1 1 1 1 –1], the result will be [6 –2], whereas if the incoming code is [–1 1 1 1 –1 1], the result will be [2 2]. This is problematic because the maximum correlation value is not always the same, and thus the performance depends on the delay of the incoming code, which is not desired. This small example shows that not every combination can be performed if fixed performance is desired. This paper will thus give a set of combination rules that allow a constant maximum correlation value regardless of the delay of the incoming secondary code, in addition to a high level of complexity reduction.

Section II describes the acquisition scheme considered, recalls some properties of the GPS L5 secondary code, provides the implementation with short and long coherent integration times, and evaluate their complexity. Section III explains why it is interesting to combine secondary code correlation and how it should be done, and demonstrates the impact on the SNR. Then, Section IV applies this to the GPS L5 signal by showing the possible combinations and their performances, evaluates the complexity, and compares it with the two traditional solutions.

II. Secondary Code Correlation

A. Acquisition of GNSS signals

To illustrate the proposed method, the parallel code search (PCS) acquisition is considered [13-16]. Note, however, that the method proposed in this article can be applied to other search methods as well (serial search [14], parallel frequency search [16,17] two-dimension search [18,19,20]), since it is related to the correlation with the secondary code. The PCS is chosen because it is an efficient method that provides a good balance between performance and resource/memory requirements [16,21,5], and enable us to manipulate simple equations.

The PCS, as discussed in [7,5], is illustrated in Fig. 1, where the correlation with the primary code is performed by FFTs, followed by the correlation with the secondary code, possibly followed by non-coherent integration. First, the incoming signal is stored into a memory of size $N_\mathrm{P}N_\mathrm{C}N_\mathrm{NC}$ for fast processing, N_P being the number of samples in one period of the primary code, N_C the number of coherent integrations (i.e. the number of primary code correlation results coherently accumulated) and N_NC the number of non-coherent integrations [21,5]. The incoming signal is then multiplied by a local carrier. It is then separated in blocks of $2N_\mathrm{P}$ samples, which are zero-padded to reach a length that is a power of two, i.e., $N_\mathrm{FFT} = 2^{\lceil \log_2(2N_\mathrm{P}) \rceil}$. Consecutive blocks overlap by N_P samples. Each of these blocks, denoted $\boldsymbol{x}_i$, is correlated with the local zero-padded code $\boldsymbol{p}_\mathrm{Z}$ using FFTs, giving the correlation results, denoted $\boldsymbol{r}_i$. Only the first N_P samples of the correlation results are kept, since the others may be impacted by a sign transition [1,3]. Further non-coherent integration can be performed to increase the sensitivity. Note that for hardware implementations, the accumulators in Fig. 2 are memory-based accumulators, each one using a memory of N_P elements and one logic adder [5]. Therefore, such an accumulator mainly consumes memory.

If the residual carrier is completely removed, the ith correlation result can be expressed as

$$\boldsymbol{r}_i = a\, s_{i-m_\mathrm{S}} \boldsymbol{r}_\mathrm{P} + \boldsymbol{\eta}_i, \tag{1}$$

where a is the signal amplitude, s_{i-m_S} is the $(i-m_\mathrm{S})$th chip of the secondary code (the subscript of s is modulo N_S, where N_S is the secondary code length in chip), m_S is the unknown delay of the incoming secondary code, $\boldsymbol{r}_\mathrm{P}$ is the autocorrelation of the primary code of length N_P with an unknown shift, and $\boldsymbol{\eta}_i$ is the noise.

The circular correlation with the secondary code for one period is then given by

$$\boldsymbol{y}_k = \sum_{i=0}^{N_\mathrm{S}-1} s_{i-k} \boldsymbol{r}_i, \tag{2}$$

with $k = 0, 1, \ldots, N_\mathrm{S}-1$, and the subscript of s is modulo N_S (the same applies for the next equations when a subscript may be lower than 0 or higher than $N_{\mathrm{S}-1}$). The correlation can also be written using matrices as

$$\begin{bmatrix} \boldsymbol{y}_0 \\ \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_{N_\mathrm{S}-1} \end{bmatrix} = \begin{bmatrix} s_0 & s_1 & \cdots & s_{N_\mathrm{S}-1} \\ s_{N_\mathrm{S}-1} & s_0 & \cdots & s_{N_\mathrm{S}-2} \\ \vdots & \vdots & \ddots & \vdots \\ s_1 & s_2 & \cdots & s_0 \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_0 \\ \boldsymbol{r}_1 \\ \vdots \\ \boldsymbol{r}_{N_\mathrm{S}-1} \end{bmatrix} \tag{3}$$
$$\boldsymbol{y} = \boldsymbol{S}\,\boldsymbol{r},$$

where $\boldsymbol{S}$ is a right-circulant matrix with s^T as the first row.

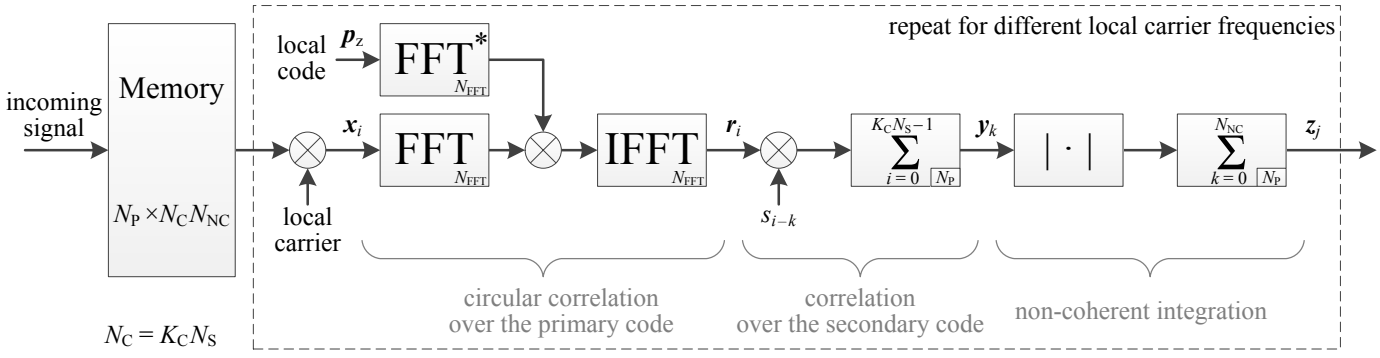

Fig. 1. PCS acquisition with secondary code correlation and non-coherent integration.

TABLE I L5 SECONDARY CODE AND ITS AUTOCORRELATION

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Secondary code	1	1	1	1	1	–1	1	1	–1	–1	1	–1	1	–1	1	1	–1	–1	–1	1
Auto-correlation	20	0	0	0	0	0	–4	0	4	0	–4	0	4	0	–4	0	0	0	0	0

B. Secondary code

The GPS L5 secondary code of the pilot channel and its autocorrelation are given in TABLE I.

C. Traditional secondary code processing

This section details the two traditional ways to deal with the secondary code, using short coherent integration time and testing all the possibilities, or using long coherent integration time by computing the secondary code correlation.

1) Short coherent integration time

When short integration times are used, all the possibilities for the accumulation of several primary code correlation results should be tested. Fig. 2 shows this when 3 primary code correlation results are accumulated ($N_\mathrm{C} = 3$), i.e., the coherent integration time is three times the primary code period, and there is no non-coherent integration.

If non-coherent integration is performed in addition, all the possibilities for the accumulation of the coherent integrations should be tested as well. Fig. 3 shows this when $N_\mathrm{C} = 2$ and 2 coherent integrations are non-coherently accumulated ($N_\mathrm{NC} = 2$), i.e., the coherent integration time is two times the primary code period, and the total integration time is four times the primary code period.

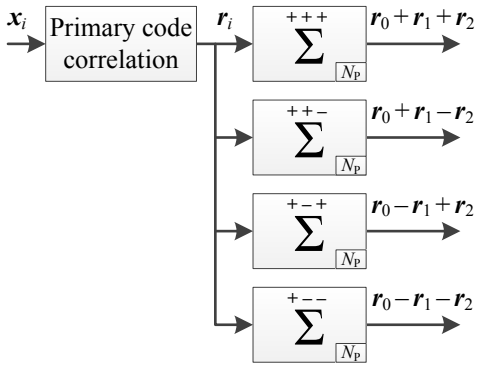

Fig. 2. Acquisition with a coherent integration time of three primary code periods ($N_\mathrm{C} = 3$).

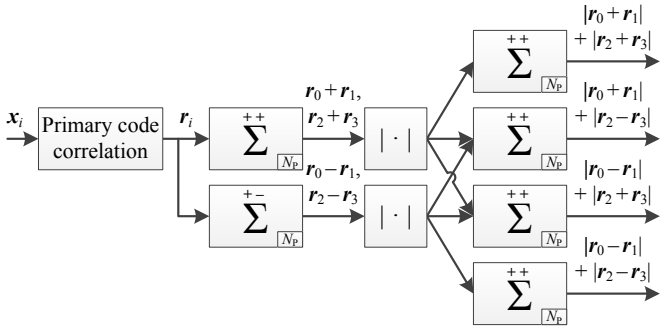

Fig. 3. Acquisition with a coherent integration time of two primary code periods, and two non-coherent accumulations ($N_\mathrm{C} = 2$, $N_\mathrm{NC} = 2$).

TABLE II NUMBER OF ACCUMULATORS REQUIRED ACCORDING TO THE NUMBER OF COHERENT ACCUMULATIONS (N_C) AND NON-COHERENT ACCUMULATIONS (N_NC).

Number of coherent accumulators

N_C \ N_NC	1	2	3	4	5
1	1	1	1	1	1
2	2	2	2	2	2
3	4	4	4	4	4
4	8	8	8	8	8
5	16	16	16	16	16

Number of non-coherent accumulators

N_C \ N_NC	1	2	3	4	5
1	0	1	1	1	1
2	0	4	8	16	32
3	0	16	64	256	1024
4	0	64	512	4096	32 768
5	0	256	4096	65 536	1 048 576

Total number of accumulators

N_C \ N_NC	1	2	3	4	5
1	1	2	2	2	2
2	2	6	10	18	34
3	4	20	68	260	1028
4	8	72	520	4104	32 776
5	16	272	4112	65 552	1 048 592

In summary, the number of coherent accumulators is $2^{N_\mathrm{C}-1}$, and the number of non-coherent accumulators is $(2^{N_\mathrm{C}-1})^{N_\mathrm{NC}}$ if $N_\mathrm{NC} > 1$. TABLE II shows the number of accumulators according to N_C and N_NC, and it can clearly be seen that non-coherent integration is not interesting because the number of accumulators rapidly explodes. Therefore, only low sensitivity can be achieved with this method.

2) Long coherent integration time

To obtain a long coherent integration time, the correlation with the secondary code should be computed. Fig. 4 and Fig. 5 show this when the secondary code correlation is performed in parallel and serially, respectively [5]. The second one uses much less memory, but it is also much slower. These figures show $N_\mathrm{C} = N_\mathrm{S}$, but of course the coherent integration time can be longer.

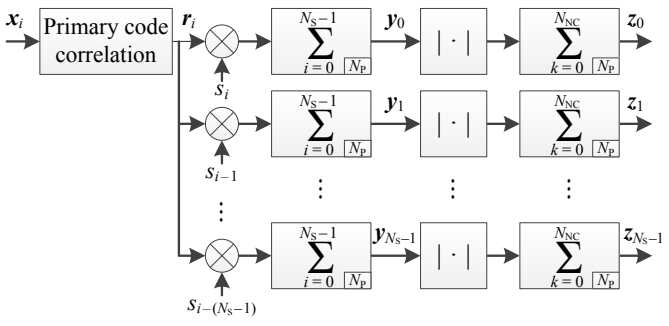

Fig. 4. PCS acquisition with parallel secondary code correlation ($N_\mathrm{C} = N_\mathrm{S}$).

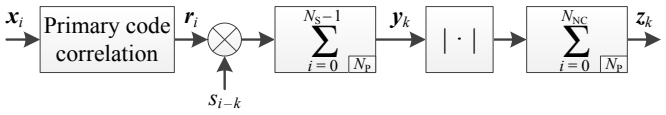

Fig. 5. PCS acquisition with serial secondary code correlation ($N_C = N_S$).

3) Complexity

TABLE III and TABLE V give the complexity for a standard computation when the coherent integration time is respectively N_C and N_S times the primary code length [5]. However, there are more efficient implementations. For the first case, looking at Fig. 2, it is seen that the second output can be obtained from the first one by subtracting $2\boldsymbol{r}_2$, then the fourth output can be obtained from the second one by subtracting $2\boldsymbol{r}_1$, and finally, the third output can be obtained from the fourth one by adding $2\boldsymbol{r}_2$. Therefore, once the first output is computed, each new output requires only one operation instead of N_C, following a kind of Gray code scheme with recursion. TABLE IV shows the complexity when recursion is used, and it can be seen that it does not change anything for the parallel implementation. For the second case, several methods have been proposed to reduce the complexity by decomposing the secondary code, where the processing time for the serial implementation and the number of additions are approximately halved [8][9].

TABLE III COMPLEXITY FOR SHORT INTEGRATION TIMES TESTING ALL POSSIBILITES FOR N_C ACCUMULATIONS.

Number of	Hardware		Software
	Parallel	***Serial***	
memories	2^{N_C-1}	1	$N_C + 1$
primary code correlations	N_C	$N_C\ 2^{N_C-1}$	N_C
cycles (processing time)	$N_C\ N_{FFT}$	$N_C\ 2^{N_C-1}\ N_{FFT}$	-
additions	-	-	$(N_C - 1)\ 2^{N_C-1}$

TABLE IV COMPLEXITY FOR SHORT INTEGRATION TIMES TESTING ALL POSSIBILITES FOR N_C ACCUMULATIONS USING RECURSION.

Number of	Hardware		Software
	Parallel	***Serial***	
memories	2^{N_C-1}	1	$N_C + 1$
primary code correlations	N_C	$N_C + 2^{N_C-1} - 1$	N_C
cycles (processing time)	$N_C\ N_{FFT}$	$(N_C + 2^{N_C-1} - 1)\ N_{FFT}$	-
additions	-	-	$(N_C - 1) + 2^{N_C-1} - 1$

TABLE V COMPLEXITY FOR LONG INTEGRATION TIMES FOR N_S ACCUMULATIONS.

Number of	Hardware		Software
	Parallel	***Serial***	
memories	N_S	1	$N_S + 1$
primary code correlations	N_S	N_S^2	N_S
cycles (processing time)	$N_S\ N_{FFT}$	$N_S^2\ N_{FFT}$	-
additions	-	-	$(N_S - 1)\ N_S$

III. PROPOSED METHOD : COMBINING SECONDARY CODE CORRELATIONS

This section aims to present the proposed method and demonstrate the impact of combining secondary code correlations on the SNR. For this second point, only few simple rules regarding the expected value and the variance of random variables are used.

A. Why and how combining secondary code correlations ?

If N_{RC} secondary code correlation results are combined, the number of secondary code delays to search is divided by N_{RC}. In the case of serial correlation, this implies that the processing time will be divided by N_{RC}. In the case of parallel correlation, the number of memory-based accumulators will be divided by N_{RC} (the next subsection discusses this in more detail).

However, this is not the only element reducing the complexity. It is possible to have a greater reduction depending on the combinations performed. To illustrate this, let us consider an example with a 4-chip long code, with the following correlation :

$$\begin{bmatrix}\boldsymbol{y}_0\\ \boldsymbol{y}_1\\ \boldsymbol{y}_2\\ \boldsymbol{y}_3\end{bmatrix} = \begin{bmatrix}s_0 & s_1 & s_2 & s_3\\ s_3 & s_0 & s_1 & s_2\\ s_2 & s_3 & s_0 & s_1\\ s_1 & s_2 & s_3 & s_0\end{bmatrix}\begin{bmatrix}\boldsymbol{r}_0\\ \boldsymbol{r}_1\\ \boldsymbol{r}_2\\ \boldsymbol{r}_3\end{bmatrix}. \tag{4}$$

The combination of one result with the next gives

$$\begin{bmatrix}\boldsymbol{y}_0 + \boldsymbol{y}_1\\ \boldsymbol{y}_2 + \boldsymbol{y}_3\end{bmatrix} = \begin{bmatrix}s_0 + s_3 & s_1 + s_0 & s_2 + s_1 & s_3 + s_2\\ s_2 + s_1 & s_3 + s_2 & s_0 + s_3 & s_1 + s_0\end{bmatrix}\begin{bmatrix}\boldsymbol{r}_0\\ \boldsymbol{r}_1\\ \boldsymbol{r}_2\\ \boldsymbol{r}_3\end{bmatrix}, \tag{5}$$

where nothing special happens. Now, the combination of one result with the next but one gives

$$\begin{aligned}\begin{bmatrix}\boldsymbol{y}_0 + \boldsymbol{y}_2\\ \boldsymbol{y}_1 + \boldsymbol{y}_3\end{bmatrix} &= \begin{bmatrix}s_0 + s_2 & s_1 + s_3 & s_2 + s_0 & s_3 + s_1\\ s_3 + s_1 & s_0 + s_2 & s_1 + s_3 & s_2 + s_0\end{bmatrix}\begin{bmatrix}\boldsymbol{r}_0\\ \boldsymbol{r}_1\\ \boldsymbol{r}_2\\ \boldsymbol{r}_3\end{bmatrix}\\ &= \begin{bmatrix}s_0 + s_2 & s_1 + s_3\\ s_3 + s_1 & s_0 + s_2\end{bmatrix}\begin{bmatrix}\boldsymbol{r}_0 + \boldsymbol{r}_2\\ \boldsymbol{r}_1 + \boldsymbol{r}_3\end{bmatrix},\end{aligned} \tag{6}$$

where it can be seen that some columns of the matrix become similar, which allows us to combine some primary code correlation results $\boldsymbol{r}_i$. This implies that the inputs $\boldsymbol{x}_i$ can be combined prior to the primary code correlation (since it is a linear operation), which will reduce the number of FFTs to compute.

In summary, if N_{RC} secondary code correlation results are combined where the results are separated by a delay N_S/N_{RC} (as in Eq. (6) with $N_S = 4$, $N_{RC} = 2$ and a delay of 2, or as in the introduction with $N_S = 6$, $N_{RC} = 3$ and a delay of 2), the number of secondary code delays to test is divided by N_{RC}, and N_{RC} inputs can be combined. This implies that the complexity is approximately divided by N_{RC}^2. The complexity can be further reduced due to the fact that the local combined secondary code may contain zeros, as shown in Section IV, but this is specific to the code and combination used.

Regarding the sign of the combinations, only two patterns are possible for the inputs combinations to allow a reduction by N_{RC}^2 : only +, or alternatively + and –; and the pattern should be circular (e.g., + – + – + does not work). If other patterns are

used, either some inputs may still be combined (e.g., + + – –) or none of them may be combined (e.g., + + + –).

Considering only positive combinations, the output can be expressed as

$$\begin{aligned} \boldsymbol{y}'_k &= \sum_{l=0}^{N_{\mathrm{RC}}-1} \boldsymbol{y}_{k+\frac{N_\mathrm{S}}{N_{\mathrm{RC}}}l} \\ &= \sum_{i=0}^{N_\mathrm{S}/N_{\mathrm{RC}}-1} s'_{i-k}\boldsymbol{r}'_i, \end{aligned} \tag{7}$$

with $k = 0, 1, \ldots, N_\mathrm{S}/N_{\mathrm{RC}} - 1$, and

$$\boldsymbol{r}'_i = \sum_{l=0}^{N_{\mathrm{RC}}-1} \boldsymbol{r}_{i+\frac{N_\mathrm{S}}{N_{\mathrm{RC}}}l}, \tag{8}$$

$$s'_i = \sum_{l=0}^{N_{\mathrm{RC}}-1} s_{i+\frac{N_\mathrm{S}}{N_{\mathrm{RC}}}l}. \tag{9}$$

This can also be written using matrices for the two patterns mentioned previously as

$$\begin{bmatrix} \boldsymbol{y}'_0 \\ \boldsymbol{y}'_1 \\ \vdots \\ \boldsymbol{y}'_{\frac{N_\mathrm{S}}{N_{\mathrm{RC}}}-1} \end{bmatrix} = \begin{bmatrix} s'_0 & s'_1 & \cdots & s_{\frac{N_\mathrm{S}}{N_{\mathrm{RC}}}-1} \\ s_{\frac{N_\mathrm{S}}{N_{\mathrm{RC}}}-1} & s_0 & \cdots & s_{\frac{N_\mathrm{S}}{N_{\mathrm{RC}}}-2} \\ \vdots & \vdots & \ddots & \vdots \\ s_1 & s_2 & \cdots & s_0 \end{bmatrix} \begin{bmatrix} \boldsymbol{r}'_0 \\ \boldsymbol{r}'_1 \\ \vdots \\ \boldsymbol{r}'_{\frac{N_\mathrm{S}}{N_{\mathrm{RC}}}-1} \end{bmatrix} \tag{10}$$
$$\boldsymbol{y}' = \boldsymbol{S}'\,\boldsymbol{r}',$$

where $\boldsymbol{S}'$ is a right-circulant if only positive combinations are done, and is right skew-circulant if alternative positive and negative combinations are done. Fig. 6 shows the parallel implementation scheme of the proposed method. Note that the proposed method needs to store more signal than the actual integration time, and needs to access several portions $\boldsymbol{x}_i$ simultaneously, and the carrier removal should also be adapted because of the simultaneous access of $\boldsymbol{x}_i$.

B. Complexity

Following the same methodology as in Section II.C, the complexity for the proposed method is given in TABLE VI. In the hardware parallel implementation, the number of memories and the processing time are both divided by N_{RC} as compared to the traditional parallel correlation (Fig. 4, TABLE V). In the serial implementation, the number of memories is unchanged and the processing time is divided by $N_\mathrm{S}N_{\mathrm{RC}}/N'_{\mathrm{S,NZ}}$ compared to the traditional serial correlation (Fig. 5, TABLE IV), where $N'_{\mathrm{S,NZ}}$ is the number of chips in $\boldsymbol{s}'$ that are different from zero.

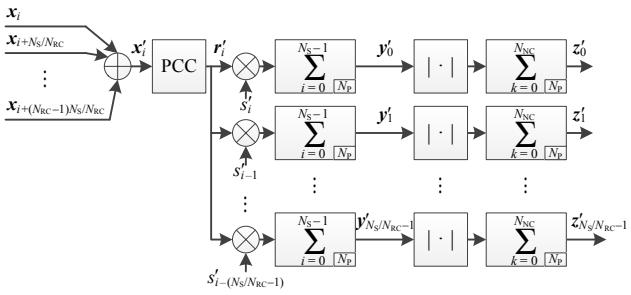

Fig. 6. Parallel implementation of the proposed method, where N_{RC} inputs are combined, giving $N_\mathrm{S}/N_{\mathrm{RC}}$ output delays. PCC : primary code correlation.

TABLE VI COMPLEXITY OF THE PROPOSED METHOD USING N_S PRIMARY CODE PERIODS. IN THE WORST CASE, $N'_{\mathrm{S,NZ}} = N_\mathrm{S}/N_{\mathrm{RC}}$.

Number of	Hardware		Software
	Parallel	*Serial*	
memory	$N_\mathrm{S}/N_{\mathrm{RC}}$	1	$N_\mathrm{S}/N_{\mathrm{RC}} + 2$
primary code correlation	$N_\mathrm{S}/N_{\mathrm{RC}}$	$N_\mathrm{S}/N_{\mathrm{RC}}\, N'_{\mathrm{S,NZ}}$	$N_\mathrm{S}/N_{\mathrm{RC}}$
cycle (processing time)	$N_\mathrm{S}/N_{\mathrm{RC}}\, N_{\mathrm{FFT}}$	$N_\mathrm{S}/N_{\mathrm{RC}}\, N'_{\mathrm{S,NZ}}\, N_{\mathrm{FFT}}$	-
additions	-	-	$N_\mathrm{S} + N_\mathrm{S}/N_{\mathrm{RC}}N'_{\mathrm{S,NZ}}$

In the worst case, there is no zero in $\boldsymbol{s}'$, i.e., $N'_{\mathrm{S,NZ}} = N_\mathrm{S}/N_{\mathrm{RC}}$. Therefore, in the serial implementation, the processing time is divided by N_{RC}^2 in the worst case. In software, $N_\mathrm{S}/N_{\mathrm{RC}}$ memories are needed to store the $\boldsymbol{r}'_i$, one for $\boldsymbol{x}'_i$ and one for $\boldsymbol{y}'_k$, and the number of additions in the worst case is $N_\mathrm{S} + N_\mathrm{S}^2/N_{\mathrm{RC}}^2$ (the first N_S is for the computation of the $\boldsymbol{x}'_i$), and therefore, they are roughly divided by N_{RC}^2 as compared to the traditional implementation.

C. SNR in the traditional case

By inserting Eq. (1) into Eq. (2), the correlation can be expressed as

$$\boldsymbol{y}_k = a\sum_{i=0}^{N_\mathrm{S}-1} s_{i-k}s_{i-m_\mathrm{S}}\boldsymbol{r}_\mathrm{P} + \sum_{i=0}^{N_\mathrm{S}-1} s_{i-k}\boldsymbol{\eta}_i. \tag{11}$$

The samples of $\boldsymbol{\eta}_i$ follow a normal distribution with a zero mean and a variance of σ^2, and $\boldsymbol{\eta}_i$ and $\boldsymbol{\eta}_j$ are independent for $i \neq j$. Multiplying $\boldsymbol{\eta}_i$ by +1 or –1 does not change its mean or variance, therefore the samples of $s_{i-k}\boldsymbol{\eta}_i$ also follow a normal distribution with a zero mean and a variance of σ^2.

Considering the sample of $\boldsymbol{r}_\mathrm{P}$ with the correct delay, denoted m, and that $r_{\mathrm{P},m} = 1$ as normalization to simplify the equations (this has no impact in the results), before the secondary code correlation, the expected value is

$$\mathrm{E}\big(r_{i,m}\big) = \mathrm{E}\big(as_{i-m_\mathrm{S}}r_{\mathrm{P},m} + \eta_{i,m}\big) = as_{i-m_\mathrm{S}}, \tag{12}$$

the variance is

$$\mathrm{Var}\big(r_{i,m}\big) = \mathrm{Var}\big(as_{i-m_\mathrm{S}}r_{\mathrm{P},m} + \eta_{i,m}\big) = \sigma^2, \tag{13}$$

and therefore, the SNR is

$$SNR_\mathrm{r} = \frac{\mathrm{E}\big(r_{i,m}\big)^2}{\mathrm{Var}\big(r_{i,m}\big)} = \frac{a^2}{\sigma^2}, \tag{14}$$

since the value of s_{i-m_S} is +1 or –1.

Still considering the sample of $\boldsymbol{r}_\mathrm{P}$ with the correct delay, after the secondary correlation, the expected value is

$$\begin{aligned} \mathrm{E}\big(y_{k,m}\big) &= \mathrm{E}\left(\sum_{i=0}^{N_\mathrm{S}-1} s_{i-k}\, r_{i,m}\right) = \sum_{i=0}^{N_\mathrm{S}-1} \mathrm{E}\big(s_{i-k}\, r_{i,m}\big) \\ &= \sum_{i=0}^{N_\mathrm{S}-1} s_{i-k}\,\mathrm{E}\big(r_{i,m}\big) = \sum_{i=0}^{N_\mathrm{S}-1} s_{i-k}\, a\, s_{i-m_\mathrm{S}} \\ &= a\sum_{i=0}^{N_\mathrm{S}-1} s_{i-k}\, s_{i-m_\mathrm{S}}, \end{aligned} \tag{15}$$

the variance is

$$\begin{aligned}\mathrm{Var}(y_{k,m}) &= \mathrm{Var}\left(\sum_{i=0}^{N_S-1} s_{i-k}\, r_{i,m}\right) = \sum_{i=0}^{N_S-1} \mathrm{Var}(s_{i-k}\, r_{i,m}) \\ &= \sum_{i=0}^{N_S-1} s_{i-k}^2\, \mathrm{Var}(r_{i,m}) = \sum_{i=0}^{N_S-1} s_{i-k}^2 \sigma^2 \\ &= \sigma^2 \sum_{i=0}^{N_S-1} s_{i-k}^2,\end{aligned} \tag{16}$$

and therefore, the SNR is

$$\begin{aligned}SNR_y &= \frac{\mathrm{E}(y_{k,m})^2}{\mathrm{Var}(y_{k,m})} = \frac{a^2\left(\sum_{i=0}^{N_S-1} s_{i-k}\, s_{i-m_S}\right)^2}{\sigma^2 \sum_{i=0}^{N_S-1} s_{i-k}^2} \\ &= SNR_r \frac{\left(\sum_{i=0}^{N_S-1} s_{i-k}\, s_{i-m_S}\right)^2}{\sum_{i=0}^{N_S-1} s_{i-k}^2}.\end{aligned} \tag{17}$$

Consequently, the gain brought by the secondary code correlation is

$$G = \frac{SNR_y}{SNR_r} = \frac{\left(\sum_{i=0}^{N_S-1} s_{i-k}\, s_{i-m_S}\right)^2}{\sum_{i=0}^{N_S-1} s_{i-k}^2}. \tag{18}$$

In the traditional case, s_{i-k} and s_{i-m_S} are binary sequences having as value $\{-1, +1\}$, which implies that $\sum_{i=0}^{N_S-1} s_{i-k}^2 = N_S$. If, in addition, the alignment is correct, i.e., $k = m_S$, we have :

- $\sum_{i=0}^{N_S-1} s_{i-m_S}\, s_{i-m_S} = N_S$;
- $G = \frac{SNR_y}{SNR_r} = \frac{N_S^2}{N_S} = N_S$.

This is of course expected since when the alignment is correct, the code "disappears" from the signal, and the operation then becomes a simple accumulation. Since N_S uncorrelated samples are accumulated, there is a gain of N_S in the SNR [22].

In the next section, secondary code correlation results are combined, which means that s_{i-k} will be replaced by another code that is no longer binary.

D. SNR when combining secondary code correlations

If two secondary code correlations results of two different delays are added or subtracted, we obtain

$$\begin{aligned}\boldsymbol{y}_k \pm \boldsymbol{y}_j &= \sum_{i=0}^{N_S-1} s_{i-k}\, \boldsymbol{r}_i \pm \sum_{i=0}^{N_S-1} s_{i-j}\, \boldsymbol{r}_i \\ &= \sum_{i=0}^{N_S-1} \left(s_{i-k} \pm s_{i-j}\right) \boldsymbol{r}_i = \sum_{i=0}^{N_S-1} s_{i,(k,j,\pm)}\, \boldsymbol{r}_i,\end{aligned} \tag{19}$$

where $s_{i,(k,j,\pm)} = s_{i-k} \pm s_{i-j}$ can take as value $\{-2, 0, 2\}$. Then, the integration gain becomes

$$\begin{aligned}G = \frac{SNR_y}{SNR_r} &= \frac{\left(\sum_{i=0}^{N_S-1} s_{i,(k,j,\pm)} s_{i-m_S}\right)^2}{\sum_{i=0}^{N_S-1} s_{i,(k,j,\pm)}^2} \\ &= \frac{\left(\sum_{i=0}^{N_S-1} s_{i-k} s_{i-m_S} \pm \sum_{i=0}^{N_S-1} s_{i-j} s_{i-m_S}\right)^2}{\sum_{i=0}^{N_S-1} \left(s_{i-k} \pm s_{i-j}\right)^2}.\end{aligned} \tag{20}$$

Since the goal is to increase the gain as much as possible, the numerator should be increased. To do so, the correlation results should be combined such that they add positively together, as predicted by intuition. However, it is not that simple, because the denominator should be taken into account as well.

Eq. (20) can easily be generalized. The numerator is the square of the combinations of the secondary code correlations, and the denominator is the sum of the square of the combinations of the secondary code chips :

$$\begin{aligned}G = \Bigg(&\sum_{i=0}^{N_S-1} s_{i-k_1} s_{i-m_S} \pm \sum_{i=0}^{N_S-1} s_{i-k_2} s_{i-m_S} \pm \cdots \\ &\pm \sum_{i=0}^{N_S-1} s_{i-k_{N_{RC}}} s_{i-m_S}\Bigg)^2 \\ &\Big/ \sum_{i=0}^{N_S-1} \left(s_{i-k_1} \pm s_{i-k_2} \pm \cdots \pm s_{i-k_{N_{RC}}}\right)^2.\end{aligned} \tag{21}$$

IV. Application to the L5 signal

A. Possible combinations

The L5 secondary code has 20 chips (20 = 1×20 = 2×10 = 4×5), thus there are the following possibilities to respect the rule mentioned previously :

- 2 results combined with a delay of 10, to get 10 results;
- 4 results combined with a delay of 5, to get 5 results;
- 5 results combined with a delay of 4, to get 4 results;
- 10 results combined with a delay of 2, to get 2 results;
- 20 results combined with a delay of 1, to get 1 result.

TABLE VII summarizes these combinations, detailing the combined secondary code (which is the local modified secondary code), the combined secondary code autocorrelation, and the SNR gain. For example, the second combination has a gain of 12, which means that the equivalent coherent integration time is 12 ms since the primary code is 1 ms.

Comparing the combinations, combination 2 offers a better gain than combination 1, and is therefore preferred. Combinations 3 and 4 have the same performance in terms of SNR gain and correlation side lobes, but combination 4 has a slightly more interesting combined code (it is composed of four 2s, which may make it easier to further reduce the complexity). Combination 5 seems better than 3 and 4 because it offers a higher gain, while combining more results. Combination 6 is not at all efficient. Combination 7 offers the same gain as combinations 3 and 4, but it offers a much greater complexity reduction since it combines 10 results. Finally, combinations 8 and 9 are clearly not interesting, since the SNR would be lower compared to one input. Therefore, the rest of the section will focus only on combinations 2, 5, and 7.

B. Complexity

TABLE VIII gives the complexity for the selected combinations following TABLE VI.

TABLE VII POSSIBLE COMBINATIONS WITH THE L5 SECONDARY CODE, AND CORRESPONDING SNR GAIN COMPARED TO A ONE PRIMARY CODE INTEGRATION. *MOST INTERESTING COMBINATIONS CONSIDERED FOR THE PERFORMANCE COMPARISON.

	Combination	N_{RC}	Resulting code and correlation										Numerator of G	Denominator of G	SNR gain G
1	$\sum_{k=0}^{1} \boldsymbol{y}_{10k}$	2	2	0	2	0	2	0	0	0	–2	0	$16^2 = 256$	32	8
			16	0	4	0	–4	0	–4	0	4	0			
2*	$\sum_{k=0}^{1} (-1)^k \boldsymbol{y}_{10k}$	2	0	2	0	2	0	–2	2	2	0	–2	$24^2 = 576$	48	12
			±24	0	∓4	0	±4	0	∓4	0	±4	0			
3	$\sum_{k=0}^{3} \boldsymbol{y}_{5k}$	4	2		0		2		–2		2		$16^2 = 256$	64	4
			16		–4		4		4		–4				
4	$\sum_{k=0}^{3} (-1)^k \boldsymbol{y}_{5k}$	4	2		0		2		2		2		$16^2 = 256$	64	4
			±16		±4		±4		∓4		∓4				
5*	$\sum_{k=0}^{4} \boldsymbol{y}_{4k}$	5	1		–3		3		3				$28^2 = 784$	140	5.6
			28		0		-12		0						
6	$\sum_{k=0}^{9} \boldsymbol{y}_{2k}$	10	4				0						$16^2 = 256$	160	1.6
			16				0								
7*	$\sum_{k=0}^{9} (-1)^k \boldsymbol{y}_{2k}$	10	–2				–6						$40^2 = 1600$	400	4
			±40				0								
8	$\sum_{k=0}^{19} \boldsymbol{y}_{k}$	20	4										$16^2 = 256$	320	0.8
			16												
9	$\sum_{k=0}^{19} (-1)^k \boldsymbol{y}_{k}$	20	4										$16^2 = 256$	320	0.8
			±16												

TABLE VIII COMPLEXITY OF THE PROPOSED METHOD FOR THE L5 SIGNAL.

Number of	N_{RC}	T_C (ms)	$\frac{N_S}{N_{RC}}$	$N'_{S,NZ}$	Hardware		Software
					Parallel	*Serial*	
memory	2	12	10	6	10	1	12
	5	5.6	4	4	4	1	6
	10	4	2	2	2	1	4
primary code correlation	2	12	10	6	10	60	10
	5	5.6	4	4	4	16	4
	10	4	2	2	2	4	2
additions	2	12	10	6	-	-	80
	5	5.6	4	4	-	-	36
	10	4	2	2	-	-	24

C. Proposed method vs short integration time

TABLE IX summarizes the complexity for short coherent integration times between 1 ms and 5 ms, and TABLE X compares the short integration time implementation using recursion for 4 ms and 5 ms of coherent integration time, and the proposed method for combinations 7 and 5, which have a coherent integration time of 4 ms and 5.6 ms, respectively.

TABLE IX COMPLEXITY FOR SOME SHORT INTEGRATION TIMES TESTING ALL POSSIBILITES FOR N_C ACCUMULATIONS USING RECURSION.

Number of	T_C (ms)	Hardware		Software
		Parallel	*Serial*	
memory	1	1	1	1
	2	2	1	3
	3	4	1	4
	4	8	1	5
	5	16	1	6
primary code correlation	1	1	2	1
	2	2	3	2
	3	3	6	3
	4	4	11	4
	5	5	20	5
additions	1	-	-	1
	2	-	-	2
	3	-	-	5
	4	-	-	10
	5			19

TABLE X COMPARISON OF COMPLEXITY BETWEEN THE PROPOSED METHOD AND SOME SHORT INTEGRATION TIMES.

Number of	N_{RC}	T_C	$\frac{N_S}{N_{RC}}$	$N'_{S,NZ}$	Hardware		Software
					Parallel	*Serial*	
memory	-	4	-	-	8	1	5
	10	4	2	2	2	1	4
	-	5	-	-	16	1	6
	5	5.6	4	4	4	1	6
primary code correlation	-	4	-	-	4	11	4
	10	4	2	2	2	4	2
	-	5	-	-	5	20	5
	5	5.6	4	4	4	16	4
additions	-	4	-	-	-	-	10
	10	4	2	2	-	-	24
	-	5	-	-	-	-	19
	5	5.6	4	4	-	-	36

TABLE X shows that the proposed method :

- uses less memory for the hardware parallel implementation (75 % reduction), and a similar amount in the other cases;
- computes fewer primary code correlations;
- requires more additions.

Therefore, overall, the proposed method is more interesting, with some reserve for software implementations due to the higher number of additions. But remember that the proposed method can use with non-coherent integrations, which opens the way to higher sensitivities.

D. Proposed method vs long integration time

TABLE XI summarizes the complexity for a long coherent integration time of 20 ms (the duration of one secondary code period), and TABLE XII compares it with the proposed method. Two cases are considered for the proposed method : 1) Using N_S primary code periods, which gives a coherent integration time of 12 ms, with the goal of significantly reducing the complexity; 2) Using $2N_S$ primary code periods, by computing two results and adding them together to get a coherent integration time of 24 ms, with the goal of slightly reducing the complexity. TABLE XII shows that the proposed method :

- uses about half the memory in both cases. However, for the second case, 40 ms of input data is needed instead of 20 ms; therefore, overall, the amount of memory is not halved. Nevertheless. the memory saved contains high resolution samples (> 14 bits), whereas the input signal is saved with low resolution (< 4 bits), and therefore, in the end, the total amount of memory is still lower with the second case of the proposed method;
- computes less or the same number of primary code correlations. The reduction is especially significant for serial hardware implementation, even considering the smart approach of [8,9] that could approximately halve it;

TABLE XI COMPLEXITY FOR A LONG COHERENT INTEGRATION TIME ($T_C = N_S T_P = 20$ MS).

Number of	Hardware		Software
	Parallel	*Serial*	
memory	20	1	21
primary code correlation	20	400	20
additions	-	-	380

TABLE XII COMPLEXITY FOR A COHERENT INTEGRATION TIME OF $N_S T_P$.

Number of	N_{RC}	T_C	$\frac{N_S}{N_{RC}}$	$N'_{S,NZ}$	Hardware		Software
					Parallel	*Serial*	
memory	-	20	-	-	20	1	21
	2	12	10	6	10	1	12
	2	24	10	6	10	1	12
primary code correlation	-	20	-	-	20	400	20
	2	12	10	6	10	60	10
	2	24	10	6	20	120	20
additions	-	20	-	-	-	-	380
	2	12	10	6	-	-	80
	2	24	10	6	-	-	160

- requires fewer additions, even considering the smart approach of [8,9] that could approximately halve it.

The estimates for a hardware receiver are very accurate, because the processing time is directly proportional to the number of primary code correlations, as shown in [5]. Therefore, TABLE XII shows that for a better sensitivity (coherent integration time of 24 ms instead of 20 ms), the processing time can be reduced by 70 % compared to the traditional implementation, and by about 25 % compared to the method 1 proposed in [8] that was requiring 160 correlations instead of 400 (with an additional memory).

For a software, the values of TABLE XII are still very representative, because the additions are performed on vectors of large size, it can thus be expected that the processing time for the secondary code correlation is proportional to the number of additions given in TABLE XII. To verify this, simulations have been performed on Matlab to measure 100 times the processing time of 200 iterations of the algorithms. Two configurations have been tested, one including both the FFT-based primary code correlations and the secondary code correlation, and another including only the secondary code correlation, to have a better evaluation of the impact on each of this part. Figures 7 and 8 show the reduction in the processing time obtained for these two configurations, using N_S primary code periods, i.e. the coherent integration time is 12 ms. For the case of 24 ms of coherent integration time, the processing time would be simply doubled. When the primary code correlations are taken into account, the proposed method reduces the processing time by about 45 % (this is because the number of primary code correlations is halved and these correlations are the most consuming operations). Therefore, with 24 ms of integration time the processing time would be 10 % higher than the traditional implementation, while the coherent

integration time is 20 % longer, therefore the overall performance are slightly better for the proposed method. As for the method proposed in [8], it offers very low reduction. When only the secondary code correlations are calculated, the reduction is about 70 %. Therefore, with 24 ms of integration time the processing time would be 40 % lower than the traditional implementation, which is similar to the method proposed in [8] (which has an integration time of 20 ms).

Therefore, the proposed method can either greatly reduce the complexity for a smaller integration time, or slightly reduce the complexity for a slightly longer integration time, i.e., it is possible to win both in terms of complexity and of sensitivity.

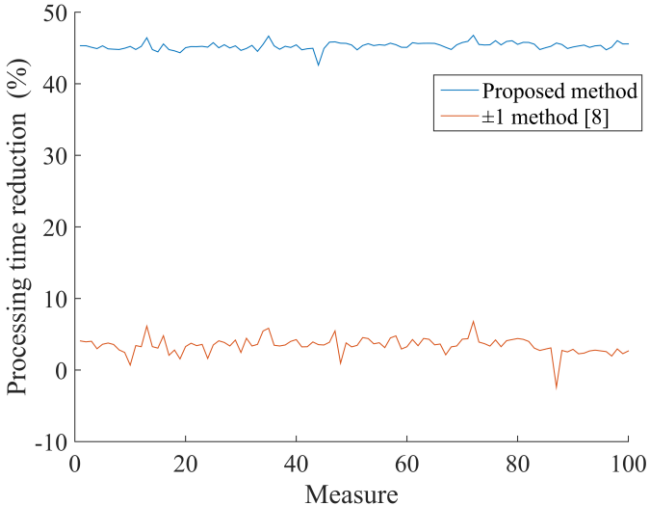

Fig. 7. Reduction of the processing time in Matlab compared to the traditional implentation, including both primary code correlations and secondary code correlations.

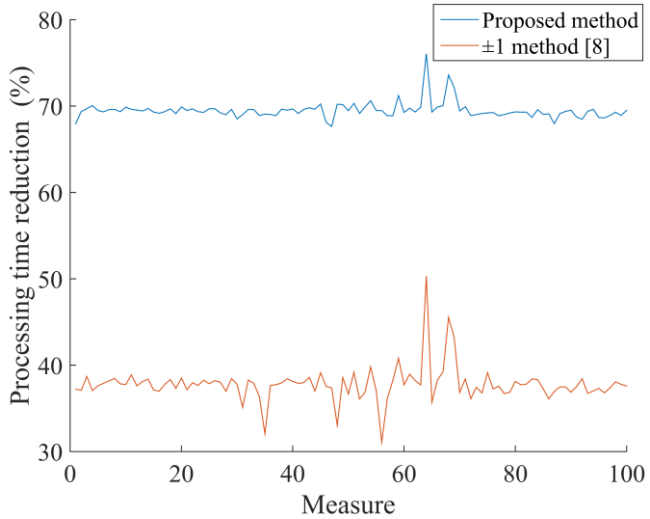

Fig. 8. Reduction of the processing time in Matlab compared to the traditional implentation, including only the secondary code correlations.

V. Conclusion

The secondary code of modern GNSS signals makes the extension of the coherent integration time difficult. Existing solutions were not sensitive enough (use of short coherent integration times by testing all possibilities for the accumulations, and non-coherent integrations not possible) or had a high computational burden (computing the correlation with the secondary code). This article proposes a method that fills the gap between these two solutions, enabling moderate to high sensitivity acquisition of GNSS signals with a secondary code, with low to moderate complexity.

The idea behind the proposed method is to combine secondary correlation results, which reduces the number of code delays to search. Moreover, with well selected combinations, the inputs can be combined instead of the outputs, thereby reducing the complexity by computing fewer primary code periods and shortening the local secondary code. In return, there is a loss in the SNR and it requires more data than the obtained coherent integration time. It also requires subsequently finding the correct secondary code delay from among a few possibilities, but this is not a big problem once the signal is detected, and the primary code delay and the Doppler frequency are correctly estimated.

It has been demonstrated that the complexity of the proposed method is overall lower than the short coherent integration time method (statement less true for software implementations regarding the number of additions), while the proposed method allows the use of non-coherent integration, which enables higher sensitivities. It has also been shown that the complexity of the proposed method is lower than the long coherent integration time. A significant reduction of complexity can be obtained against a small loss of sensitivity, or a slight reduction of complexity can be obtained with a slight increase of sensitivity, i.e., it is possible to have a win-win. For example, it is possible to have a coherent integration time of 24 ms instead of 20 ms while reducing the processing time by 70 % for a hardware implementation, and by 45 % or 70 % for a software implementation according to if both if the primary code and secondary codes correlations are calculated or only secondary code correlations, respectively. The only drawback is that 40 ms of data are needed instead of 20 ms.

In a future research, a study of the combinations for other signals can be performed. Other combinations can also be studied to determine if the currently proposed combinations are optimal. Finally, a study of the impact of the residual Doppler could be performed as well since it is known that it affects the secondary code correlation.

References

[1] L. Lo Presti, X. Zhu, M. Fantino, P. Mulassano, “GNSS signal acquisition in the presence of sign transition”, IEEE Journal of Selected Topics in Signal Processing, vol. 3, no. 4, pp. 557-570, 2009.

[2] D. Borio, M. Fantino, L. Lo Presti, “The impact of the Galileo signal in space in the acquisition system,” in Satellite Communications and Navigation Systems, Springer US, 2008.

[3] J. Leclère, C. Botteron, P.-A. Farine, “Acquisition of modern GNSS signals using a modified parallel code-phase search architecture”, Signal Processing, vol. 95, pp. 177-191, 2014.

[4] M. Foucras, O. Julien, C. Macabiau, B. Ekambi, F. Bacard, “Probability of detection for GNSS signals with sign transitions”, IEEE Transactions on Aerospace and Electronic Systems, vol. 52, no. 3, pp. 1296-1308, 2016.

[5] J. Leclère, C. Botteron, P.-A. Farine, “High sensitivity acquisition of GNSS signals with secondary code on FPGAs”, IEEE Aerospace and Electronic Systems Magazine, vol. 32, no. 8, pp. 46-63, 2017.

[6] C. Hegarty, M. Tran, A. Van Dierendonck, “Acquisition algorithms for the GPS L5 signal”, ION GPS/GNSS 2003, Portland, OR, USA, Sept. 2003, pp. 165–177.

[7] C. Yang, C. Hegarty, M. Tran, “Acquisition of the GPS L5 signal using coherent combining of I5 and Q5”, ION GNSS, pp. 2184-2195, Sept. 2004.

[8] J. Leclère, R. Jr, Landry, “Complexity reduction for high sensitivity acquisition of GNSS signals with a secondary code”, ION GNSS+ 2016, Portland, Oregon, USA, Sept. 2016, pp. 436-443.

[9] J. Leclère, M. Andrianarison, R. Jr, Landry, “Efficient GNSS secondary code correlations for high sensitivity acquisition”, ENC 2017, Lausanne, Switzerland, May 2017.

[10] G. Corazza, C. Palestini, R. Pedone, M. Villanti, “Galileo primary code acquisition based on multi-hypothesis secondary code ambiguity elimination”, ION GNSS 2007, Fort Worth, TX, USA, Sept. 2007, pp. 2459–2465.

[11] N. Shivaramaiah, A. Dempster, C. Rizos, “Exploiting the secondary codes to improve signal acquisition performance in Galileo receivers”, ION GNSS 2008, Savannah, GA, USA, Sept. 2008, pp. 1497–1506.

[12] D. Borio, “M-sequence and secondary code constraints for GNSS signal acquisition”, IEEE Transactions on Aerospace and Electronic Systems, vol. 47, no. 2, pp. 928-945, 2011.

[13] D. van Nee, A. Coenen, “New fast GPS code-acquisition technique using FFT”, Electronics Letters, vol. 27, no. 2, pp. 158-160, 1991.

[14] K. Borre, D. Akos, N. Bertelsen, P. Rinder, S. Jensen, “A software-defined GPS and Galileo receiver. Single-frequency approach”, Applied and Numerical Harmonic Analysis. Boston, MA: Birkhäuser, 2007.

[15] C. O'Driscoll, “Performance analysis of the parallel acquisition of weak GPS signals”, Ph.D. thesis, University College Cork, Ireland, 2007.

[16] J. Leclère, “Resource-efficient parallel acquisition architectures for modernized GNSS signals”, Ph.D. thesis, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland, 2014.

[17] H. Mathis, P. Flammant, A. Thiel, “An analytic way to optimize the detector of a postcorrelation FFT acquisition algorithm”, ION GPS/GNSS, pp. 689-699, Sept. 2003.

[18] N.I. Ziedan, J.L. Garrison, “Unaided acquisition of weak GPS signals using circular correlation or double-block zero padding”, IEEE PLANS, April 2004, pp. 461–470.

[19] M. Foucras, O. Julien, C. Macabiau, B. Ekambi, “A novel computationally efficient Galileo E1 OS acquisition method for GNSS software receiver”, ION GNSS, pp. 365-383, Sept. 2012.

[20] D. Akopian, Fast FFT based GPS satellite acquisition methods, IEE Proceedings Radar, Sonar and Navigation, vol. 152, no. 4, pp. 277-286, August 2005.

[21] J. Leclère, C. Botteron, P.-A. Farine, “Comparison framework of FPGA-based GNSS signals acquisition architectures”, IEEE Transactions on Aerospace and Electronic Systems, vol. 49, no. 3, pp. 1497-1518, 2013.

[22] F. van Diggelen, “A-GPS: Assisted GPS, GNSS, and SBAS”, GNSS Technology and Applications Series, Artech House, 2009.