Performance Assessment of the TurboDLL for Satellite Navigation Receivers

Fabio Dovis[†], Marco Pini[†], Paolo Mulassano[‡]

[†]Politecnico di Torino – Dipartimento di Elettronica
Corso Duca degli Abruzzi 24, 10129, Torino, Italy
Phone: +39 011 5644175, Fax: +39 011 5644099, e-mail: name.surname@polito.it
[‡]Istituto Superiore Mario Boella
via Pier Carlo Boggio 61, 10128, Torino, Italy
Phone: +39 011 2276414, Fax: +39 011 2276299, e-mail: paolo.mulassano@ismb.it

Abstract. In this paper a detailed evaluation of the performance of the architecture named "Turbo Delay Lock Loop" (TurboDLL) is presented. Such an architecture has been introduced by the authors in [1], as an innovative solution for improving the performance of satellite navigation receivers in multipath affected scenarios. The relevant innovation resides in the fact that the architecture aims at tracking each multipath component and, after a transient time, use them to wipe the multipath components off the input signal. The iterative procedure allows for a major improvement in the error induced in the code-based pseudorange measurement.

The architecture uses a preliminary estimation of the propagation channel in terms of number of not negligible reflections, and of their relative amplitude. In this paper the robustness of the TurboDLL architecture with respect to imperfect channel estimation is demonstrated.

1 The TurboDLL Architecture

In order to compute the user's location, the GNSS receiver must estimate the distances with respect to, at least, four satellites, through a fine alignment of the incoming and local codes. After the acquisition phase, such an operation, usually named "code tracking process", is carried out using a DLL for each digital channel within the receiver. In case a coherent Early-minus-Late DLL is used, a Phase Lock Loop (PLL) co-operating with the DLL is required [3]. The baseband input signal is correlated with the prompt (P), Early (E) and Late (L) versions of the locally generated code through a multiplication and an integration along a pre-detection integration period. The early and late correlation values are directly used in the code tracking process. In fact, the feedback control signal is calculated on the basis of an odd discriminator function obtained through the difference between the early and the late correlation values. The multipath presence affects such a discriminator function. In fact, if a multipath component with a delay lower than 1.5 chip with respect to the LOS is present at the input, a bias error on the code alignment is experienced.

The architecture at the basis of Turbo DLL is quite different from common rejection techniques, like the narrow correlator, where multipaths are not tracked. The new system employs a set of more than one DLL per each channel and its strategy

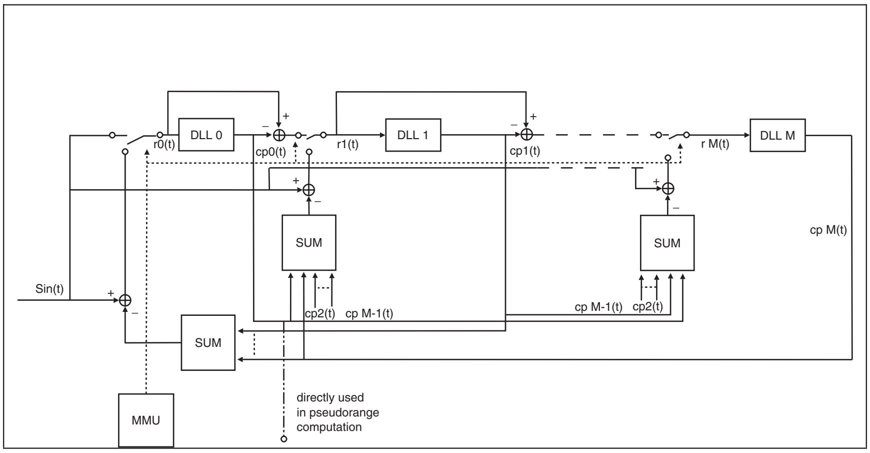

Fig. 1. Turbo DLL functional diagram.

is similar to the RAKE approach used in communication receiver. The complexity of the Turbo DLL is increased, but the proposed scheme is able to track the incoming signal replicas, since the delayed versions of the LOS are treated as additional input signal.

Referring to Fig. 1, once DLL_i is able to follow the evolution of *i-th* replica, its local code is used in order to cancel the *i-th* multipath component from the incoming signal.

In this sense, the tracking of multipath is a sort of information, which is fed back to the first DLL in order to improve the overall performance of the receiver tracking system. The word "Turbo" is thus referred as the capability of the system to boost the overall performance.

As already remarked the TurboDLL architecture is based on the multiple DLL scheme explained in [2] and two different phases can be identified:

- **Transient Time:** in which the tracking algorithm employed in the multiple DLL scheme is applied to the incoming signal and each stage of the system tracks a multipath component;
- **Steady State:** in which the code loop is closed and the DLL_0 exploits other DLLs' information to track a "cleaned" version of LOS component.

In order to allow a better understanding of the overall tracking system a detailed explanation of each phase is provided in the following.

Transient Time. As previously mentioned before in this phase the tracking algorithm developed for the Multiple DLL structure is applied. As in the case of the Multiple DLL the MMU unit is present and plays a key role. In fact it is in charge of determining whether it is necessary or not to activate the turbo architecture in presence of deleterious multipath. In particular it has to estimate the number of replicas and their amplitude with a sufficient accuracy. Furthermore, it has to compare them with a predefined threshold in order to determine how they will distort the S-curve. Otherwise if no replicas are detected or their effect can be assumed as negligible the MMU unit will force the digital channel to use standard DLL (e.g., DLL E-L narrow-correlator).

Since in this stage the aim is to study the behaviour of the Turbo architecture, a MMU being able to perfectly determine the feature (amplitude) of multi-paths that degradate the incoming signal has been assumed, while the effect of non perfect estimation will be discussed in the following sections. With reference to the general case shown in Fig. 1, in which the MMU has been able to detect M replicas of the useful signal, once the generic DLL_i is locked on the *i-th* replica then its local code, $c_{Pi}(t)$, is subtracted from the corresponding input signal $r_i(t)$ and then is fed to the next DLL, DLL_{i+1} in a sort of chain.

Steady State. As soon as the last DLL is locked on its corresponding MP then the system enters this phase and the loop is closed. This means that all the local codes generated by each DLL are subtracted from the overall incoming signal obtaining, in this way, a new input signal $r_0(t)$ for DLL_0 that will work on almost the LOS component only. The more accurate the other DLLs have tracked their corresponding MP component the better the DLL_0 works providing a local code that can be used to compute pseudorange due to the fact that multi-path distortion has been largely reduced.

As far as the other DLLs are concerned they do not have to work anymore on the signal representing the difference between the input signal of DLL_0 and the local code generated by DLL_0 itself because such a signal is just made of the residual tracking error of the LOS. To overcome this fact, the new input signal of the generic DLL_i will be the difference between the overall incoming signal and the sum of all the local codes generated by the other M DLLs; in other words if we refer, for instance, to DLL_1 in Fig. 1 its input will be given by the difference from $S_{in}(t)$ and the sum of $c_{P0}(t)$, c_{P2} (t), . . ., $c_{PM}(t)$. In this way each DLL is forced to work on the corresponding multipath component according to the philosophy of the TurboDLL.

Each DLL's contribution is iteratively employed to better the performance in pseudorange estimation.

A system of switches is then necessary to allow the DLLs to commute to the right input signal as well as one adder for each DLL_i (with $i = 1, \ldots, M$) increasing the overall complexity. The need of additional switches and adders determines a difference with the structure presented in [2], where, anyway a final DLL stage was needed.

2 A Functional Example

To better understand the way the architecture works a complete example, showing the evolution of the tracking error of each DLL, will be given in the following. In particular a situation characterized by the presence of two multi-paths in addition to the LOS will be taken into account. This assumption can be appropriate for those situation in which besides to one dominant reflective surface there is also an obstacle that causes diffraction of the direct ray generating a more delayed and attenuated replica of the LOS itself.

The simulation system taken into account is made up by three DLLs narrow-correlator with a spacing of 0.25 chips between Early and Late versions of the local code. The integration period has been assumed equal to two code periods; the amplitude of the first multipath is 0.5 (half of the LOS amplitude) while the amplitude of the second one is 0.3 and they are delayed of 0.7 chips and 1.1 chips with respect to the LOS, respectively.

Figures 2–4 show the way the system works during the *Transient Time* phase. By observing Fig. 2, which depicts the tracking error of DLL_0, it can be noticed that DLL_0, after tracking the incoming signal for about 790 periods (pointed out by the black dashed line), can be considered locked to a certain value that matches the value of 0.0779 chips that is the zero-crossing point of the distorted S-curve. Figure 3 which depicts the evolution of the tracking error of DLL_1 shows the fact that DLL_1 does not work until DLL_0 is locked; after 1090 periods DLL_1 can be considered locked and its output is used to feed up the DLL_2 whose tracking error is represented in Fig. 4. Similarly to the behaviour previously described, DLL_2 begins to work as soon as DLL_1 locks and starts tracking the second multipath.

When entering the Steady State (i.e. closing the loop), the tracking error is highly reduced as demonstrated by the behavior of DLL_0 whose tracking error is represented in Fig. 5. It is evident that after closing the loop its tracking error falls down to a value that oscillates approximately around zero with small variance, so that a significant enhancement in the pseudorange estimation can be expected.

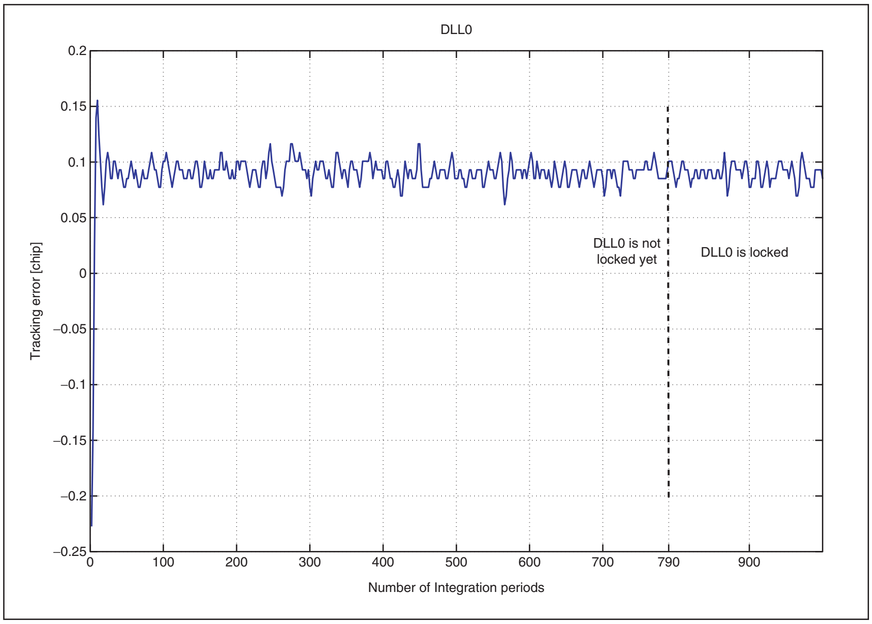

Fig. 2. Dynamics of DLL_0 in the *transient time* phase.

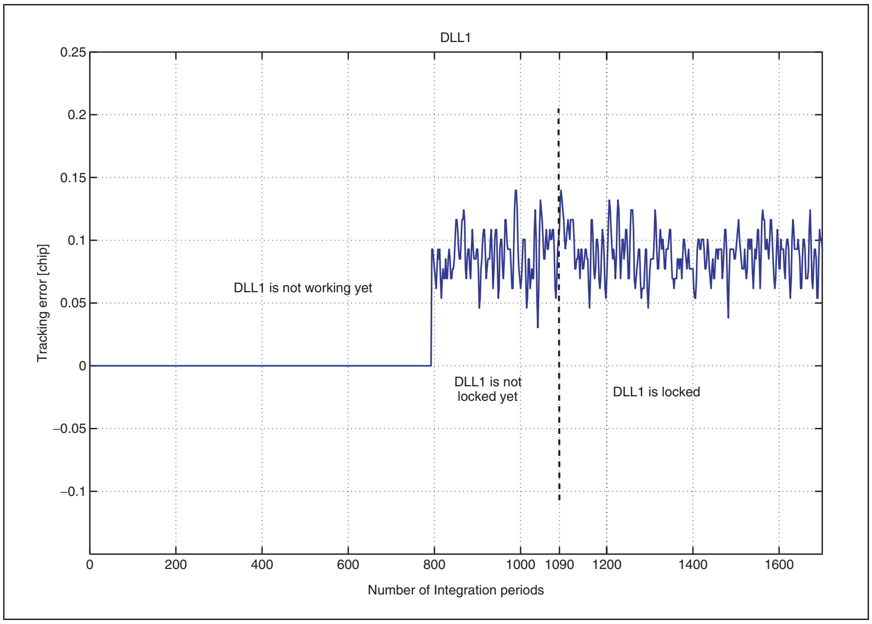

Fig. 3. Dynamics of DLL_1 in the *transient time* phase.

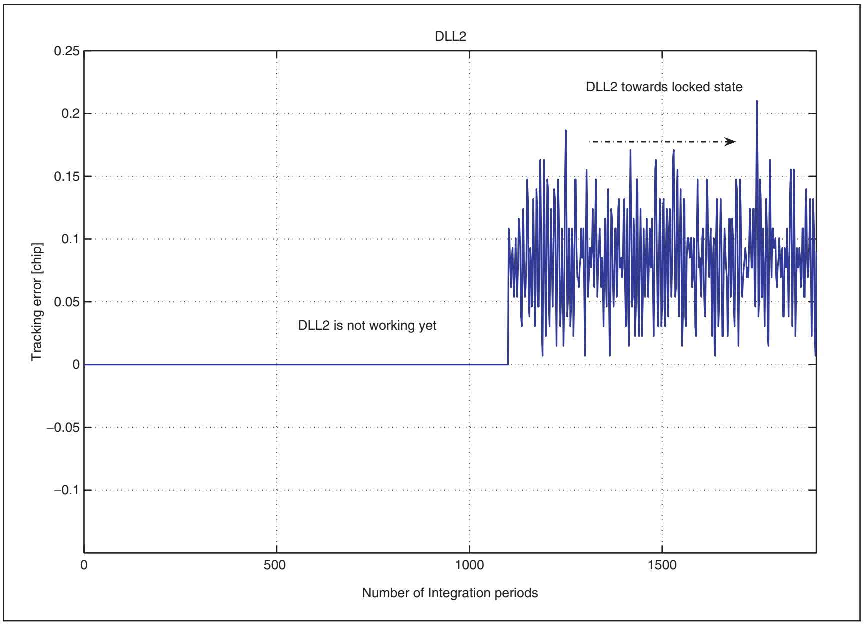

Fig. 4. Dynamics of DLL_2 in the *transient time* phase.

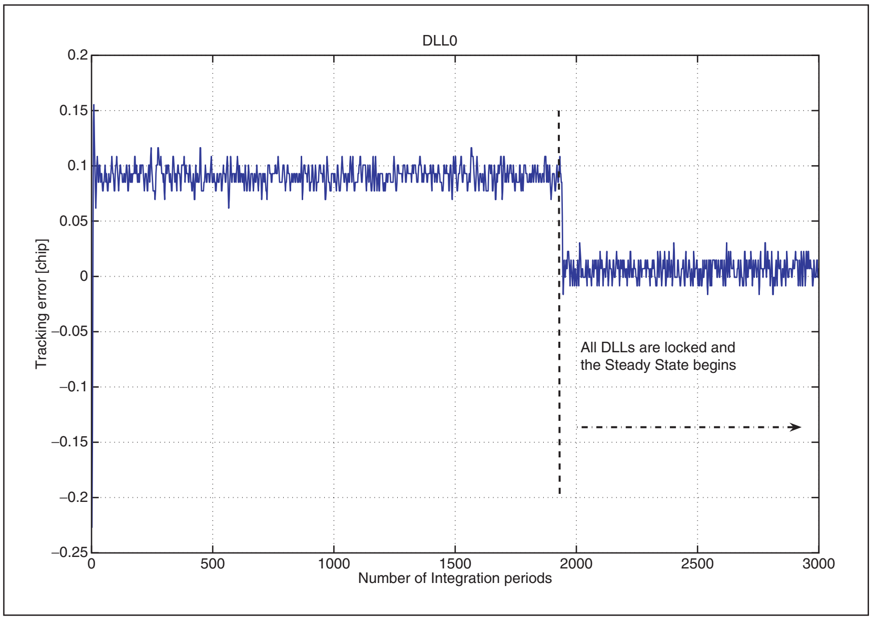

Fig. 5. Dynamics of DLL_0 in *steady state* phase.

2.1 The Propagation Channel Estimation

The complexity of the TurboDLL architecture is larger than conventional DLLs, in order to allow for a separate tracking of the input signal components. For this reason it could be desirable to start it only when multipath is conditions are detected. Furthermore, an estimation of the amplitude of each *i-th* component is required in order to adapt the dynamic of each component to the DLL_i in charge to track it. With this aim, a block in charge of the channel estimation, named Multipath Monitoring Unit (MMU) is required.

The MMU plays an important role for the functioning of the overall tracking system. In fact, it has to estimate the number of replicas and their amplitude. Furthermore, it has to compare them with a predefined threshold in order to determine their distortion effect on the S-curve. It is useful to remark that the MMU actives the Turbo DLL only in case of multipath, while, if the multipath presence is not detected, the digital channel still continue to use a standard DLL (e.g., DLL E-L narrow-correlator) as shown in Fig. 6.

The estimation performed by the MMU is affected by errors; the study presented in the paper demonstrates the ability of the iterative architecture to recover from not perfect estimation of the amplitude of the multipath components.

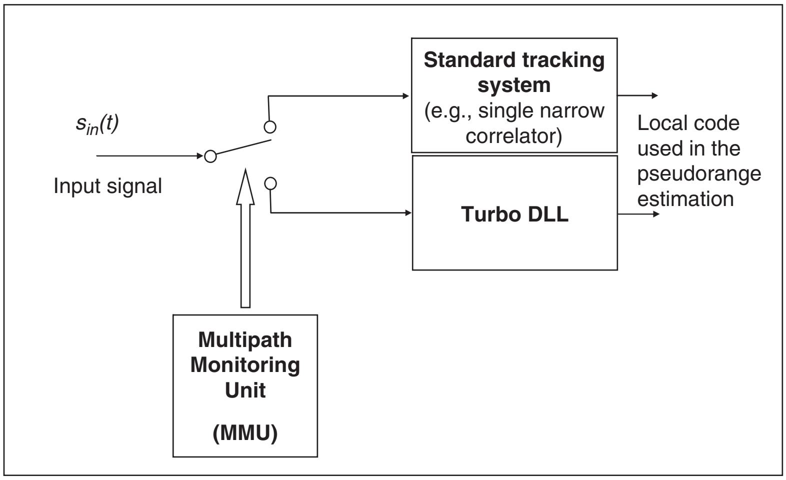
$s_{in}(t)$
Input signal
Standard tracking system
(e.g., single narrow correlator)
Turbo DLL
Local code used in the pseudorange estimation
Multipath Monitoring Unit
(MMU)

Fig. 6. Multipath monitoring unit as selector of tracking systems.

3 Performance Evaluation

Figure 7 depicts some examples of the results obtained for GPS C/A code with the TurboDLL architecture using the multipath envelope diagram. Such a plot represents the error due to the presence of one reflected ray (in fraction of chip) versus the delay of the multipath with respect to the Line-of-Sight path [3].

3.1 Imperfect Estimation of the Amplitude

The MMU plays a key role in the exploitation of the TurboDLL architecture within a receiver, since it is in charge of detecting the multipath presence and it must be able to extract some features of the received signal that are needed by the TurboDLL architecture.

The results presented for sake of example, have been obtained considering a TurboDLL made of two stages, and the presence of one single reflection with half the amplitude of the direct ray.

In Fig. 7 the theoretical curve (solid bold line) and the simulated values for a classical narrow correlator single DLL are reported. From the comparison with the values simulated for a TurboDLL with perfect estimation of the amplitude of the MP, the gain advantage of the architecture can be appreciated.

If the MMU is not precise in the estimation of the replica amplitude, providing a value which is lower than the real one, the system is still robust providing performance that are better than the single narrow correlator DLL.

Figure 7 reports the results obtained for the architecture dealing with a GPS C/A code. The considered estimation error ranges from 10% to 60% of the real

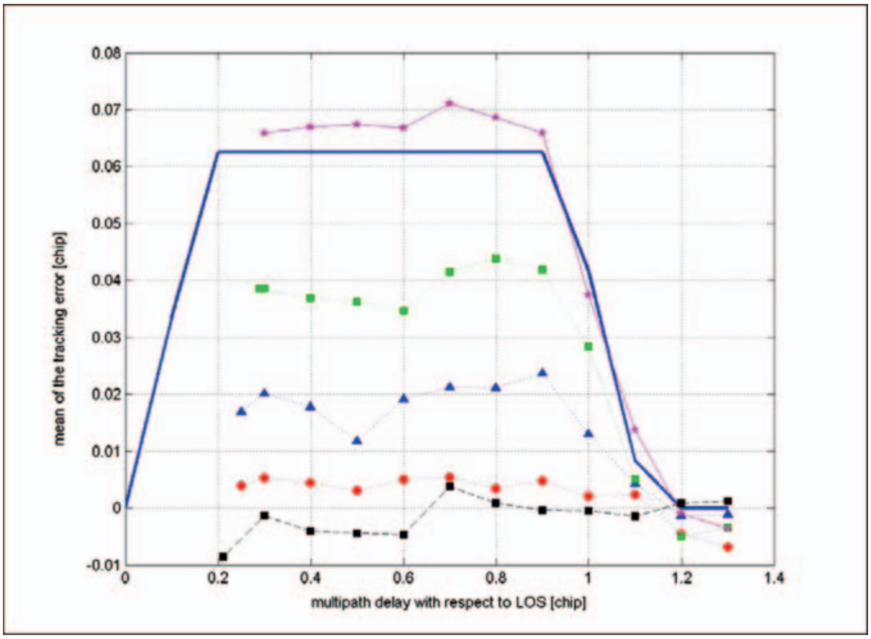

Fig. 7. GPS C/A code: theoretical MP envelope of a single DLL (solid bold line), simulated single DLL (solid line with star markers) and simulated turbo DLL architecture with perfect estimation of the MP (dashed line with square marker) versus simulated turbo DLL scheme with error of MP amplitude estimation equal to: –10% (dotted line with circle markers); –30% (dotted line with triangle markers); –60% (dotted line with square markers).

amplitude value. For an error of the 90% the system is not able to keep the tracking phase. The dotted line with square markers represents the performance in case an underestimation error of 60%. Also in this case the bias error is lower than the one of the narrow correlator single DLL. As already remarked in [1] the system performance is bounded by the ability of "separating" the multipath from the LOS contribution, and a stable state cannot be reached for very short delays of the MP with respect to the LOS (less than 0.2 chip).

A similar behavior is obtained for the Galileo BOC(1,1) signals, as depicted in Fig. 8. Also in this case the system is robust to underestimation of the multipath amplitude up to the 60% of the actual value, and it outperforms in all the cases the single narrow-correlator DLL architecture.

4 Conclusions

In this paper the performance of the TurboDLL have been discussed, showing how the architecture is able to recover the multipath contributions from the received signal, and then wiping them off the signal in an iterative procedure. The system

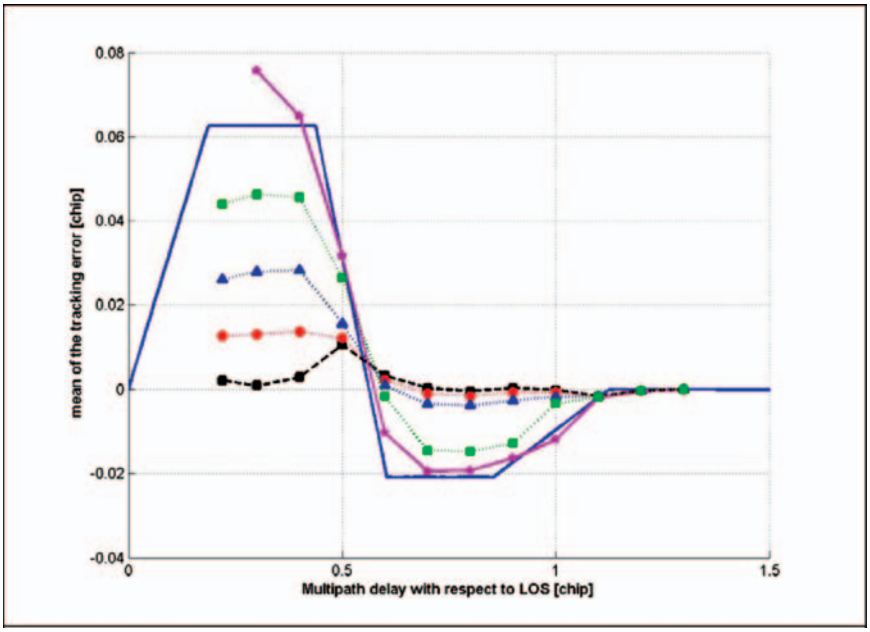

Fig. 8. Galileo BOC(1,1): theoretical MP envelope of a single DLL (solid bold line), simulated single DLL (solid line with star markers) and simulated turbo DLL architecture with perfect estimation of the MP (dashed line with square marker) versus simulated turbo DLL scheme with error of MP amplitude estimation equal to:–10% (dotted line with circle markers); –30% (dotted line with triangle markers); –60% (dotted line with square markers).

provides relevant improvements in the performance of code-based pseudoranges with respect to classical narrow-correlator discriminators, trading off the performance with the overall complexity of the tracking stage. In particular the robustness of the architecture with respect to inaccurate estimation of the multipath amplitude has been demonstrated.

Acknowledgement

Authors would like to thank the “Istituto Superiore Mario Boella” research center for its support trough the Navigation Lab. In addition, the authors would like to thank Luca Greco and Paolo Favaro for their valuable help in conducting the performance analysis.

References

[1] Fabio Dovis, Marco Pini, Paolo Mulassano, “Turbo DLL: an Innovative Architecture for Multipath Mitigation in GNSS Receivers”, *ION GNSS 2004*, Long Beach, CA (USA), 21–24 Sept., 2004.

[2] Fabio Dovis, Marco Pini, Paolo Mulassano, "Analysis of the Multiple DLL Architecture: a Novel Solution for Reducing the Multipath Effect in GNSS Receivers", *ION GNSS 2004*, Long Beach, CA (USA), 21–24 Sept., 2004.

[3] P. Misra, P. Enge, *Global Positioning System. Signals, Measurements, and Performance*, Ganga-Jamuna Press, 2004.