Generating a real-time time scale making full use of the available frequency standards

Metrologia

Generating a real-time time scale making full use of the available frequency standards

To cite this article before publication: Lorenzo Galleani *et al* 2020 *Metrologia* in press https://doi.org/10.1088/1681-7575/ab8d7d

IOP Publishing Journal Title
Journal **XX** (XXXX) XXXXXX https://doi.org/XXXX/XXXX

Generating a real-time time scale making full use of the available frequency standards

L Galleani[1], G Signorile[2], V Formichella[2] and I Sesia[2]

[1] Department of Electronics and Telecommunications, Politecnico di Torino, Turin, Italy
[2] Quantum Metrology and Nano Technologies Division, INRiM, Turin, Italy

E-mail: i.sesia@inrim.it

Received xxxxxx
Accepted for publication xxxxxx
Published xxxxxx

Abstract

We propose a time-scale algorithm for the automated generation of a real-time time scale, making full use of the frequency standards available in a typical time laboratory. The time-scale algorithm is made by a pre-processing stage, a steering algorithm, and a post-processing stage. In particular, in this work we propose a set of three different steering algorithms, running in parallel and eventually producing a unique steering correction to be applied to a master clock. Each algorithm is based on a different steering reference, namely, a primary frequency standard, an ensemble clock, and the Coordinated Universal Time (UTC), or its rapid version, UTCr. Pre- and post-processing stages help to provide robustness and to cope with data gaps. The proposed algorithms have been extensively and successfully tested off-line, on real data from the time laboratory of the Italian National Institute of Metrological Research (INRiM), where an on-line test has also been performed in the period May-October 2019. Then, since the mid of January 2020, the time-scale algorithm has been applied for the generation of the Italian legal time scale, UTC(IT). We show here the results of the off-line tests and of the 5-month on-line test. The proposed strategy can be used wherever a stable, accurate, and robust time reference is needed, e.g. for a local realization of UTC in a laboratory k, UTC(k), or for generating the reference system time of a global navigation satellite system (GNSS).

Keywords: time scale, steering algorithm, frequency standards, UTCr, UTC(k), GNSS reference time, robustness

1. Introduction

Accurate, stable, and robust reference time scales are of fundamental importance in several applications, such as global navigation satellite systems (GNSS), smart grids, telecommunications, finance, and scientific experiments [1]. Moreover, the legal time of a country is usually a real-time local realization of the Coordinated Universal Time (UTC) generated by a laboratory k, UTC(k).

Therefore, in the past years, at the Italian National Institute of Metrological Research (INRiM), we started to design algorithms for the automated generation of a real-time time scale, capable of continuous operation and with high performance in terms of accuracy, stability, and robustness [2–5]. One of the main features of our strategy is to make full use of the frequency standards available in a time laboratory, as well as of the external information coming from the Bureau International des Poids et Mesures (BIPM), whenever available.

In this work, we propose a time-scale algorithm for the generation of a real-time hardware time scale. This time-scale algorithm is made by a pre-processing stage, a steering-algorithm stage, and a post-processing stage. The steering algorithm steers a master clock towards a suitable steering

reference. In particular, we propose a set of three steering algorithms running in parallel and based on different steering references, namely: a laboratory primary frequency standard, an ensemble clock made of commercial frequency standards, and finally UTC or its rapid version, UTCr. The pre-processing stage prepares the input data for the subsequent steering-algorithm stage, whereas the post-processing stage generates a unique steering correction, to be applied to the master clock.

The whole time-scale algorithm has been extensively tested off-line, using real clock data from INRiM's time and frequency laboratory. Promising results have been obtained, which are reported and discussed in section 3.1. Section 3.2 presents in addition the final results from a 5-month on-line test which started at INRiM in May 2019.

The proposed methodology can be used wherever a time scale with high performance and robustness is required, such as for generating a UTC(k) as well as a GNSS reference time. Moreover, its modular design allows to adapt it to the real capabilities of a given time laboratory, and to the actual requirements of a given application.

Note that the basic idea behind two out of three of the proposed steering algorithms is drafted in the conference papers [2, 3]. Specifically, the algorithm proposed in [2] is based on the INRiM's cesium fountain data and computes a frequency correction Δf_0 to be applied to a master clock, in order to correct its frequency offset. Very preliminary results of about 1 month of on-line test are also presented. In [3] we reported an upgrade of the algorithm proposed in [2] with an additional correction, Δf_2, to be applied to take into account the accuracy of the generated time scale. It is moreover presented a second algorithm based on an ensemble of commercial cesium clocks. Results of a first off-line test over about 1 year of real data are shown for the two considered algorithms.

The present article gives more details about the proposed algorithms, which were not reported in [2, 3]. Furthermore, we consider the possibility to include in the clock ensemble hydrogen masers in addition to the cesium clocks, to improve the short-term stability, then, we present a third steering algorithm based only on the use of BIPM data, and, finally, we propose a possible post-processing algorithm to generate a unique steering correction starting from the outputs of the three different steering algorithms.

Note that the results presented in this article represent the achievable timekeeping performances in both nominal and critical conditions, in order to assess the robustness of the proposed algorithms.

The paper is organized as follows: the general scheme of the time-scale algorithm and the basic steering principle are presented in section 2.1, whereas the three individual steering algorithms are described in sections 2.2 to 2.4, and the post-processing algorithm in section 2.5; the results of the off-line and on-line tests are reported and discussed in section 3; then, in section 4, the three steering algorithms are compared not only in terms of performance, but also considering cost, complexity, and autonomy. Finally, the conclusions are presented in section 5, along with an outline of our future developments.

2. The time-scale algorithm

2.1 General scheme and basic steering principle

The general scheme of the proposed time-scale algorithm is depicted in figure 1. It refers to a typical time laboratory, which may have at least some of the following equipment and resources: a set of commercial frequency standards, e.g. cesium beam clocks and Active Hydrogen Masers (AHM); a laboratory primary frequency standard, such as a cesium fountain; a link with UTC, as in the case of a UTC(k) laboratory.

One of the available clocks, typically an AHM, has to be considered as the master clock, i.e. the one to be steered in order to generate the real-time hardware time scale to be disseminated to the different timing users. The steering correction is a frequency correction, computed by the algorithm and then applied through a micro phase stepper to the master clock's output.

The algorithm's input data are the phase and frequency offsets of all the clocks measured versus a stable reference clock or a time scale within the laboratory, along with the UTC–UTC(k) and UTCr–UTC(k) phase offsets published by the BIPM on a monthly and weekly basis, respectively. First of all, the available input data enter a pre-processing stage, whose main aim is to detect and filter out possible outliers using the approach described in [6]. In an advanced version of the algorithm, the pre-processing should also include a detector of possible clock anomalies (phase and frequency jumps, frequency drift and variance changes), e.g. the one described in [7], and a decisional algorithm, which may exclude the anomalous data (if any) from the next steps of the time-scale algorithm.

Then, the pre-processed input data enter three different steering algorithms, each of which is based on a different steering reference and computes a steering correction for the master clock. These three corrections are the input of a post-processing stage, whose main aim is to combine them in order to compute a unique, final steering correction, the one which is actually applied to the master clock through the micro-stepper. Input/output of the pre- and post-processing stages are also collected for monitoring purposes, along with the log files generated by the algorithm itself, in order to check if the whole automated process is working as expected.

Before discussing the three steering algorithms in the next sections, we describe here the basic principle common to all of them. Indeed, independently of the steering reference, the

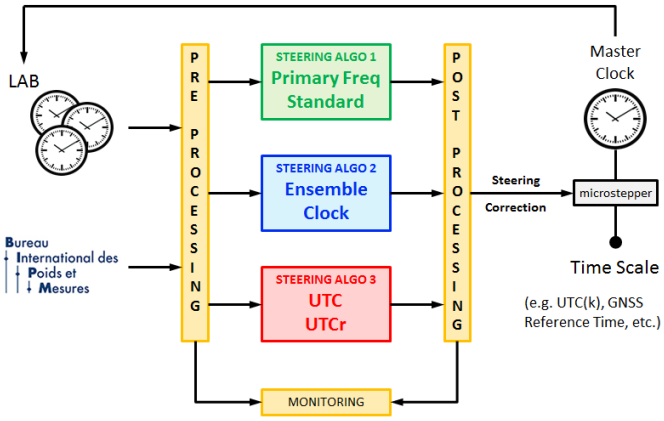

Figure 1. General scheme of the time-scale algorithm.

frequency steering correction Δf is computed as the sum of three terms, Δf_0, Δf_1, and Δf_2. The first one, Δf_0, is the main component and corrects the frequency offset of the master clock with respect to the reference. The other two terms, Δf_1 and Δf_2, correct the residual frequency and time offset of the realized time scale with respect to UTC/UTCr. In particular: Δf_0 is obtained with a linear fit on N_{fit} days of past frequency offset data, and extrapolating to the current epoch; Δf_1 is obtained as an average of the time scale's residual frequency offset data over a recent period; finally, Δf_2 is obtained by dividing the most recently measured time scale's residual time offset by N_{acc} days, which is the number of days after which that offset has to be reduced to zero. Note that N_{fit} and N_{acc} are among the most important configuration parameters of the steering algorithms and, in the first initialization phase, must be tuned in order to find a tradeoff between the two following needs: to generate a stable real-time time scale in the nominal case, but also to quickly react to possible unexpected behaviour of the master clock. For example, increasing N_{fit} may improve the stability of the time scale in nominal conditions, but also slow down the reaction of the algorithm to any possible variation of the master clock's frequency. We further discuss this topic in section 3. The choice of UTC or UTCr as a reference for computing Δf_1 and Δf_2 is also discussed in section 3.

2.2 The fountain steering algorithm

In the first steering algorithm, the laboratory primary frequency standard providing the best available realization of the SI second, e.g. a cesium fountain, is used as steering reference.

The frequency offset of the master clock is measured with respect to the primary frequency standard, and the frequency offset data are used to compute Δf_0. In principle, the batch of past data used to compute Δf_0 is regularly updated, sliding forward to include the most recent measurement results. However, if the latest measurement results are not available and, more in general, when the last N_{fit} days contain only a few data (i.e., less than a configurable threshold), the data batch actually used for the computation of Δf_0 is not updated, and hence the frequency correction is obtained from the last performed linear fit, by extrapolating up to the current epoch.

The successful generation of a real-time time scale using a primary frequency standard is already well documented, e.g. in [2] and [8, 9]. In this work, we focus on a cesium fountain as a laboratory primary frequency standard, and we compare the performance of the fountain steering algorithm with that of the other two proposed steering algorithms (see section 4). Moreover, we focus on the effects of fountain data gaps, and on the improvement which can be obtained by combining the fountain steering correction with the corrections obtained from the other two steering algorithms, in case of large gaps in the fountain data (see sections 2.5 and 3).

2.3 The ensemble-clock steering algorithm

In the ensemble-clock steering algorithm, the steering reference is an ensemble time scale, i.e. a paper time scale obtained by averaging the readings of a set of clocks. In particular, the clocks included in the ensemble are assumed to be commercial frequency standards such as cesium clocks and AHM. The ensemble time scale, henceforth often called TA (from the French "Temps Atomique"), is computed by a dedicated averaging algorithm. The aim of the averaging algorithm is to reduce the clock noise and to take the best from each clock type included in the ensemble, generating an ensemble time scale which has better stability than that of any individual clock within the ensemble. Moreover, relying on multiple clocks, the ensemble time scale is more reliable and robust than a single clock, both in the case of clock failures (as clocks can enter and leave the ensemble without discontinuities) and in the case of clock anomalies (whose amplitude is reduced by the averaging procedure). The same ensemble time scale, with its improved stability, can be used as a common reference for the detection of clock anomalies not only on the master clock, but also on all the clocks of the ensemble.

The input data of the averaging algorithm are the pre-processed phase offsets of the clocks within the ensemble, measured versus a stable reference clock or time scale. We consider two different scenarios: in the first one, only the available cesium clocks enter the ensemble, and they are averaged according to their long-term stability; in the second one, also the available AHM (excluding the master clock) enter the ensemble, improving the short-term stability of the ensemble time scale. Then, in both scenarios, TA is steered to align its frequency to that of UTC/UTCr, and the resulting time scale is the so called TA_{st}. Finally, TA_{st} is considered as the steering reference within the ensemble-clock steering algorithm.

The Δf_0 component of the steering correction is computed exactly as in the fountain steering algorithm, except for the

fact that the fitted data are the frequency offsets of the master clock with respect to TA_{st}, rather than to the primary frequency standard. These frequency offset data are obtained by differentiation from the corresponding phase offset data, which are a direct output of the averaging algorithm.

2.4 The UTC/UTCr steering algorithm

In the UTC/UTCr steering algorithm, UTC or UTCr is the steering reference. The phase offset between the master clock and the chosen reference, e.g. UTC, can be computed as [UTC−UTC(k)]+[UTC(k)−master], where the value of the first quantity is determined and published by the BIPM on a monthly basis (weekly, for UTCr), whereas the value of the second quantity is measured within the laboratory. Such phase offset data are then differentiated, and the resulting frequency offset data are used to compute the Δf_0 component of the steering correction, exactly in the same way as for the previous steering algorithms. Then, the BIPM data can be also used for the computation of Δf_1 and Δf_2.

Both UTC and UTCr can be chosen as steering references. While UTC remains the ultimate reference time scale, its rapid realization, UTCr, is expected to be comparable to UTC in terms of stability and, moreover, it is steered to UTC in order to minimize the time offset between the two time scales [10]. This means that one can use UTCr as a steering reference, instead of UTC, without degrading too much the performance of the resulting real-time time scale. On the other hand, the reduced latency in the availability of UTCr with respect to UTC, i.e. about 10 days instead of about 40 days, allows to update the steering correction more frequently, and this may result in a better stability and accuracy of the real-time time scale. According to [10], this is actually one of the reasons why UTCr has been proposed in 2011. Therefore, we present here a comparison between the performance of a UTC-based steering and of a UTCr-based steering (see section 3), showing that the UTCr-based steering provides very good performance of the realized real-time time scale.

2.5 Post-processing

The role of the post-processing algorithm is to define the actual steering correction to be applied to the master clock, based on the steering corrections computed by the different steering algorithms. It consists of two main steps: in the first one, a unique steering correction is computed from the ones resulting from the different steering algorithms; in the second step, a final check is performed on the obtained correction, before applying it.

The simplest version of the post-processing's first step, is to select one of the available steering corrections based on a priority list, depending on the prediction error associated to each steering algorithm. A better solution, which should improve the steering performance by making full use of the available frequency standards and external information, is to compute a weighted average of the three steering corrections, with weights based on the prediction error associated to each steering algorithm. Here, as an example, we present a hybrid solution which is simple and effective, as will also be shown in section 3. We call it the "mixed fountain/backup" solution.

In the mixed fountain/backup solution, the Δf_0 component of the steering correction is defined as:

$$\Delta f_0 = w_F \Delta f_{0,F} + (1-w_F)\Delta f_{0,B} \quad (1)$$

where $\Delta f_{0,F}$ and $\Delta f_{0,B}$ are the Δf_0 components of the steering corrections computed with the fountain steering algorithm and with a backup steering algorithm, respectively, where the backup can be either the ensemble clock or UTC/UTCr. The normalized weight of the fountain-based correction, w_F, is set to 1 when the fountain is available and $\Delta f_{0,F}$ is computed on an updated data batch; otherwise, when the fountain is not available and $\Delta f_{0,F}$ is extrapolated from the last performed linear fit, w_F decreases linearly in time, until it reaches 0 after a configurable time constant θ_0; when the fountain becomes available after an outage period, and a new $\Delta f_{0,F}$ is computed on an updated data batch, w_F grows linearly in time, until it reaches 1 after a configurable time constant θ_1. The ideal value of the time constant θ_0 should be assessed considering the ratio between the prediction error of the extrapolated fountain steering correction (growing with the length of the data gap) and the prediction error of the backup steering correction. The role of the time constant θ_1, instead, is just to avoid an abrupt change of Δf_0 from $\Delta f_{0,B}$ to $\Delta f_{0,F}$, and hence it can be arbitrarily chosen. As discussed in section 3, we based our choice of θ_0 and θ_1 on experimental observations.

In the second step of the post-processing algorithm, a final check on the computed steering correction is performed. The difference between the new steering correction and the current one is computed, and its absolute value is compared to a configurable threshold. If the threshold is surpassed, the value of the new steering correction is truncated, so that it corresponds to the current correction plus the threshold, and an alarm is generated to allow further investigation.

Note that the threshold value should be at least twice the value of the typical steering correction obtained in nominal conditions (5×10^{-15} in our case). The main effect of the threshold is to contain any large anomalous steering correction, while preserving the time scale stability. The consequent possible loss in accuracy due to this thresholding mechanism is largely compensated by the protection that it guarantees against such anomalous steering corrections, which rarely happen (e.g., they never happened during the tests discussed in section 3), but can have catastrophic effects on the generated time scale.

3. Test results

3.1 Off-line test results

3.1.1 Generalities

The time-scale algorithm has been coded in MATLAB® language and extensively tested off-line, on real clock data from INRiM's time and frequency laboratory.

Our laboratory generates the Italian legal time scale, UTC(IT), which was previously realized in a semi-automatic way by using data from the BIPM (but still requiring the intervention of skilled operators), whereas since the mid of January 2020 it is realized in a fully automated way, by applying the time-scale algorithm proposed in this work.

The laboratory is equipped with 4 commercial AHM (one of which is the master clock), 6 commercial cesium (Cs) clocks, and a cryogenic cesium fountain, called ITCsF2 [11]. The phase offsets of all the clocks are measured versus UTC(IT) and archived once per hour, whereas the frequency offset of the master clock with respect to ITCsF2 is made available to the algorithm as one value per day, when the fountain is operational. For testing the three steering algorithms, we used almost one year of INRiM's clocks data.

As described in section 2.1, the computation of the Δf_0 component of the steering correction is based on a batch of past data with a length of N_{fit} days. The choice of N_{fit} clearly depends on the stability of the master clock and of the steering reference. For example, if white frequency noise is the only type of noise affecting the frequency offset measurement, increasing N_{fit} would improve the steering performance. In the real case, where we may observe other types of non-stationarities affecting the frequency of the master clock, N_{fit} cannot be too large. We found a good tradeoff in the value $N_{\text{fit}} = 60$ days, that is the one used for all the tests reported in this section.

In this first step of the experimentation, the Δf_1 component of the steering correction has been set to 0, in order to focus on the characterization of the other two components, Δf_0 and Δf_2. Nevertheless, the computation of Δf_1 has been already implemented and tested on simulated data; we will report in a future work on the actual implementation of Δf_1 in a test performed on real data.

Regarding the computation of Δf_2, we compared some results obtained by using UTC or UTCr as a reference: we found them to be similar, see section 3.1.2 for an example. Therefore, considering mainly the reduced latency of UTCr, we chose it as the default reference for the computation of Δf_2 (unless differently specified). Finally, as described in section 2.1, the Δf_2 component of the steering correction depends on the configuration parameter N_{acc}: regarding the choice of its value, the only restriction is that it cannot be too small, in order to avoid an over-compensation of the residual time offset of the time scale. The quantities to be considered for this choice are the time elapsed between two consecutive updating of Δf_2 and the age of the data used for the update. In our case (UTCr as the default reference for the computation of Δf_2), N_{acc} should not be smaller than a couple of weeks, since UTCr data are weekly published and with a latency of three days with respect to the last reported value. Therefore, in order to provide a very smooth compensation of the residual time offset, we chose $N_{\text{acc}} = 30$ days as optimal tuning.

3.1.2 Fountain steering results

The fountain steering algorithm has been tested using data collected from February 2015 to February 2016. This period includes a 3-month gap in the ITCsF2 fountain data, to test the robustness and performance of the steering algorithm in case of a fountain outage. The test results are shown in figure 2, in terms of phase offset between UTC and UTC(IT): the green curve is the fountain-steered UTC(IT), i.e. the time scale obtained by using the proposed fountain steering algorithm; the grey curve is instead the official UTC(IT), as it has been realized with the semi-automatic procedure still in use at INRiM at that time, retrieved from BIPM's Circular T data and plotted for comparison. Note that, despite the 3-month data gap (which is indicated in the plot by the shaded region), the fountain-steered time scale remains well within ±5 ns from UTC for the whole 1-year test period, showing substantial improvements with respect to the official UTC(IT). Moreover, the 95$^{\text{th}}$ percentile of the phase offset over the whole period is of only 3 ns, a remarkable result.

As an example, only in the case of the fountain steering algorithm, we also report the results obtained by using UTC for the computation of Δf_2, instead of UTCr. These results are shown in figure 3, which is analogous to figure 2: the green

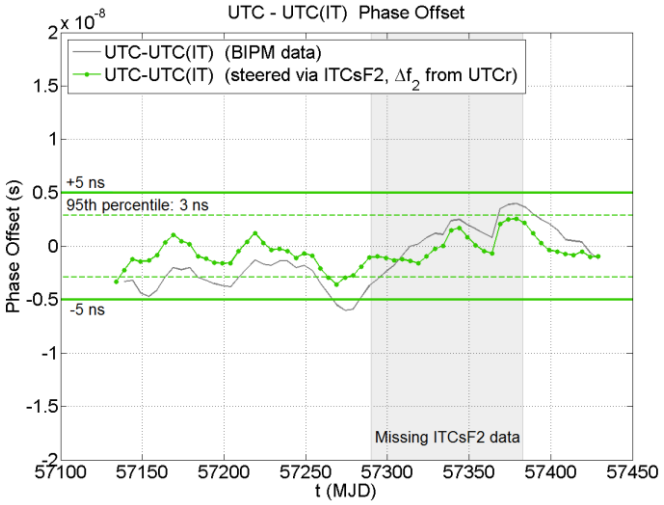

Figure 2. Test results for the fountain steering algorithm in the period from February 2015 to February 2016. The green curve is the fountain-steered UTC(IT). The grey curve is the official UTC(IT) as published by the BIPM, plotted for comparison.

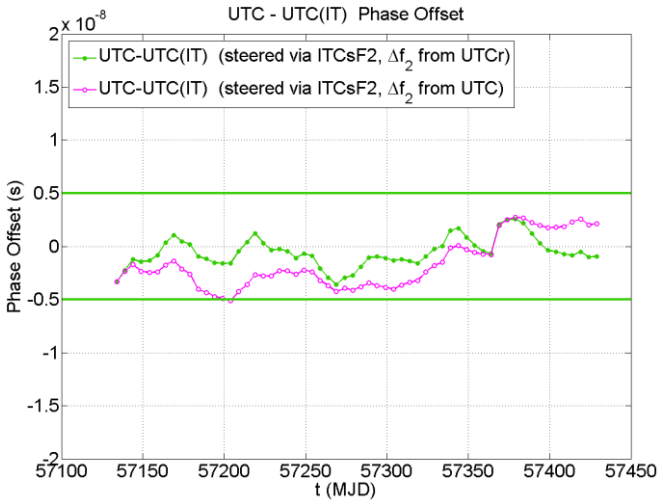

Figure 3. Test results for the fountain steering algorithm in the same period as for figure 2, comparing the use of UTCr (green curve) and UTC (purple curve) for the computation of Δf_2.

curve is the fountain-steered UTC(IT) as already reported in figure 2, obtained by using UTCr for the computation of Δf_2, whereas the purple curve is the fountain-steered UTC(IT) obtained by using UTC for computing Δf_2, with N_{acc} = 60 days. As already discussed in section 3.1.1, the two curves show similar performances and hence, due to the reduced latency allowing a faster compensation of the residual phase offset, we give our preference to the use of UTCr.

3.1.3 Ensemble-clock steering results

The ensemble-clock steering algorithm has been tested using data collected from March 2014 to May 2015. This period includes some nonstationarities on the available Cs clocks, highlighted in figure 4, in order to test the algorithm in non-nominal conditions. In a first step, only the 6 available Cs clocks enter the ensemble, and are weighted according to their stability at 30 days. Figure 4 also shows the corresponding TA_{st} (red curve), where UTCr has been chosen as steering reference. The short/mid-term stability of TA_{st} is improved with respect to that of an individual Cs clock at most of a factor $\sqrt{6}$, as expected, and even more at the long term thanks to the steering towards UTCr.

The test results for the ensemble-clock steering algorithm are shown in figure 5, always in terms of phase offset versus UTC: the blue curve is the ensemble-steered UTC(IT), i.e. the time scale obtained by using the ensemble-clock steering algorithm; the grey curve is the official UTC(IT), which is plotted for comparison. Note that, despite the non-stationarities affecting some of the Cs clocks, the ensemble-steered time scale remains well within ±10 ns from UTC for the whole period (about one year). The 95th percentile of the phase offset over the whole period is of 7 ns, therefore definitively worse than what we obtained with the fountain steering algorithm. However, on absolute grounds, this result is still very good from a performance point of view. Moreover, since the ensemble-clock steering algorithm relies on commercial clocks only, it guarantees more robustness than the fountain algorithm, as well as a much better autonomy than the UTC/UTCr algorithm.

In a second step, also the available AHM, except the master one, enter the ensemble. This improves the short-term stability of TA_{st}, which becomes better than the short-term stability of an individual AHM (at most of a factor $\sqrt{N}$, where N is the number of AHM). Then, in principle, using a very small N_{fit}, it should be possible to improve the short-term stability of the real-time time scale too. Actually, steering a clock adds some noise to the steered time scale, degrading its stability at an averaging time depending on N_{fit}. This means that the expected improvement in the short-term stability can

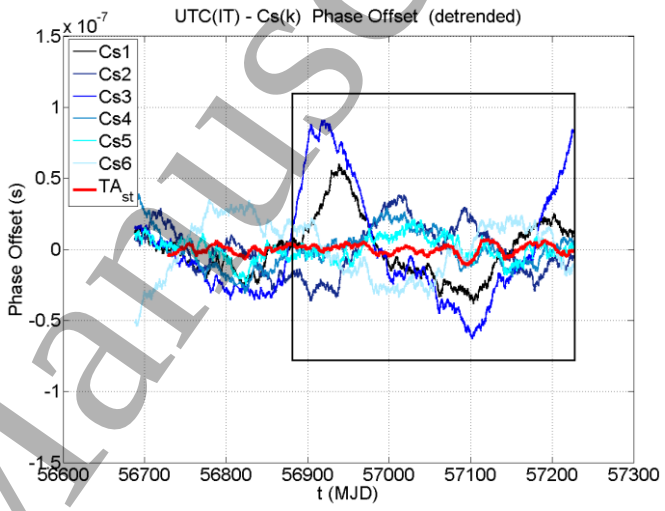

Figure 4. Phase offset of the 6 commercial Cs clocks available at INRiM with respect to the official UTC(IT), with the corresponding TA_{st} (red curve). Some Cs clock nonstationarities are highlighted in the black box.

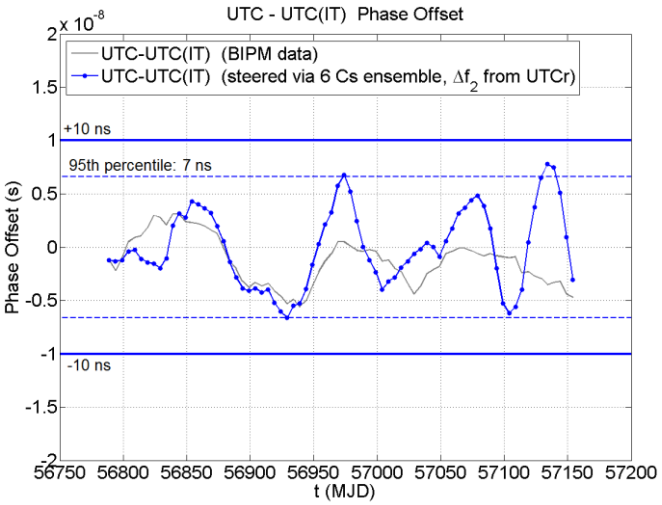

Figure 5. Test results for the ensemble-clock steering algorithm in the period from March 2014 to May 2015. The blue curve is the ensemble-steered UTC(IT). The grey curve is the official UTC(IT) as published by the BIPM, plotted for comparison.

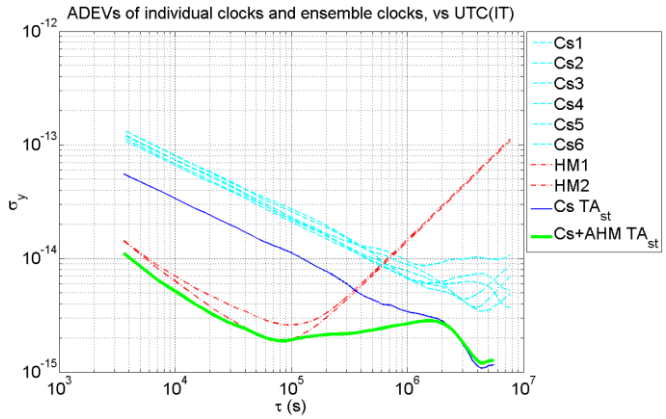

Figure 6. Stability (Allan deviation, versus the official UTC(IT)) of the individual Cs clocks and AHM (cyan and red curves), compared with the stability of the TA_{st} computed with only the Cs clocks (blue curve) or also with the AHM (green curve).

be obtained only when a large number of AHM is included in the ensemble. In our case, with 4 available AHM of which one is already used as master clock, and another one has an anomalous behaviour during the considered period, no more than 2 AHM can enter the ensemble, and this is not sufficient to improve the stability of the real-time hardware time scale. Nonetheless, we show in figure 6 the stability of the 6 individual Cs clocks (cyan curves), of the 2 individual AHM (red curves), of TA_{st} including only the Cs clocks (blue curve), and of TA_{st} including also the AHM (green curve). As expected, the latter is better than the stability of any individual clock, both at the short and at the long term. From this plot, it is also clear that the short-term improvement with only 2 AHM is very small, and hence not sufficient for improving the performance of the steering algorithm, as already explained above.

3.1.4 UTC/UTCr steering results

The UTC and UTCr steering algorithms have been tested using data collected from May 2014 to July 2015. Note that, in the UTC steering algorithm, UTC is used as reference not only for the computation of Δf_0, but also for the computation of Δf_2.

The test results for the UTCr steering algorithm are shown in figure 7 where, as in the previous cases, we plot the phase offsets versus UTC of the UTCr-steered UTC(IT) (red curve) and of the official UTC(IT) as published by the BIPM (grey curve). It can be observed that the UTCr-steered time scale remains well within ±10 ns from UTC for the whole period, covering about one year. Moreover, the 95th percentile of the phase offset over the whole period is of 6 ns, which is almost equivalent to what we obtained with the ensemble-clock steering algorithm. This is a remarkable result, considering that we are only using external information (BIPM data) and no other clock is needed except for the master one.

The test results for the UTC steering algorithm with N_{acc} = 60 days are shown in figure 8, and compared with those obtained with the UTCr steering algorithm. In particular, in figure 8 we highlighted two regions of the plot showing an interesting behaviour of the time scales. Looking at the box number 1, we note that the UTC-steered time scale (purple curve with empty circles) is more stable than the UTCr-steered one (red curve with dots). This is probably due to the superior stability of UTC along with the fact that, in the considered period, the master clock had a very stable and predictable behaviour. Instead, looking at the box number 2, we note that the UTC-steered time scale deviates from UTC much more than the UTCr-steered one. This is due to the fact that, in the period within box number 2, the master clock has

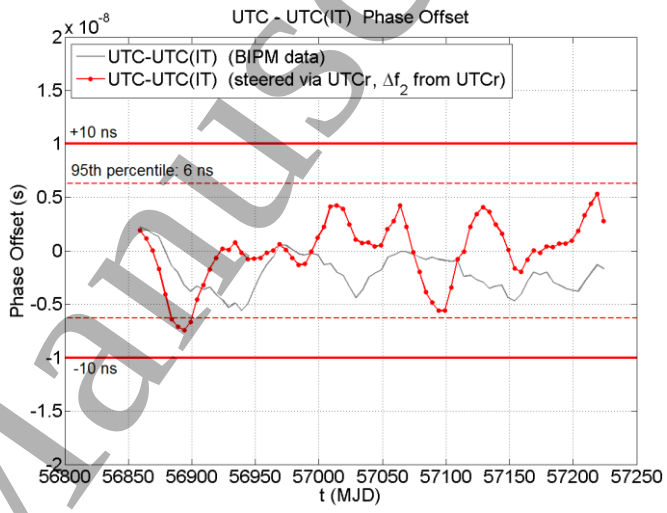

Figure 7. Test results for the UTCr steering algorithm in the test period from May 2014 to July 2015. The red curve is the UTCr-steered UTC(IT). The grey curve is the official UTC(IT) as published by the BIPM, plotted for comparison.

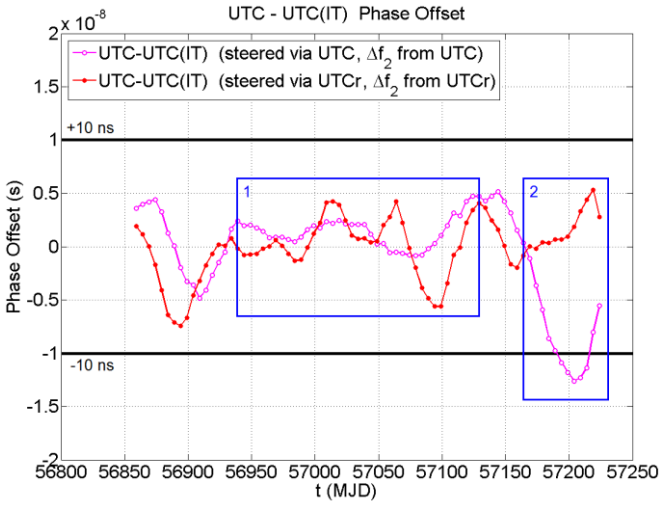

Figure 8. Comparison between the test results for the UTC (purple curve with empty circles) and UTCr (red curve with dots) steering algorithms. The behaviour of the two time scales in the regions highlighted by the blue boxes is described in the text.

been less stable and predictable than in the previous case and hence UTCr, thanks to its reduced latency, has been a better steering reference. In other words, using UTCr as a steering reference allows a faster reaction to any unexpected variation of the master clock's frequency. Therefore, looking at figure 8, we can conclude that the UTC and UTCr steering algorithms have similar overall performances. It has to be considered that the UTC steering algorithm allows to maximize the stability of the realized time scale in nominal conditions, when a very stable and predictable master clock is used. Nevertheless, the UTCr steering algorithm is a very good trade-off between the need to cope with non-nominal conditions, which are quite common in real-time operations, and the need to maximize the stability in nominal conditions.

3.1.5 Mixed fountain/backup steering results

The mixed fountain/backup post-processing algorithm has been tested using data collected from July 2016 to September 2017. This period includes two gaps in the ITCsF2 data, i.e. a large 6-month gap in the middle of the test period, and also a slightly shorter gap at the end of the considered period. With such large gaps, we can test the effectiveness of the mixed fountain/backup post-processing algorithm. To this aim, we also compare the results obtained with such post-processing algorithm to those obtained with the sole fountain steering algorithm. We consider as backup steering reference both the ensemble clock and UTCr. After some simulations, we set the values of the time constants as $\theta_0 = 90$ days and $\theta_1 = 3$ days. Note that, in this test, a value of $N_{acc} = 15$ days has been used, instead of 30 days as in the previous tests.

The test results are shown in figure 9, in terms of phase offset between UTC and UTC(IT): the green curve is the fountain-steered UTC(IT) without post-processing, i.e. the time scale obtained by using the sole fountain steering algorithm; the blue curve is UTC(IT) steered with the mixed fountain/backup post-processing, where the backup is the ensemble clock made of the available 6 Cs clocks; the red curve, instead, is obtained by using UTCr as backup steering reference; finally, as in the previous cases, the official UTC(IT) is plotted for comparison (grey curve). Looking at the first part of the green curve we note that, when the fountain data are missing (shaded region on the left side of the plot), the error in the estimation of the Δf_0 component dominates the steering correction, and the curve drifts away, reaching an offset of about 40 ns. When the fountain data are again available, the Δf_0 component is properly estimated, and the Δf_2 component dominates the steering correction, thus steering the green curve towards zero. Instead, the blue and red curves remain within ±10 ns from UTC for almost the whole period, demonstrating the effectiveness of the mixed fountain/backup post-processing algorithm.

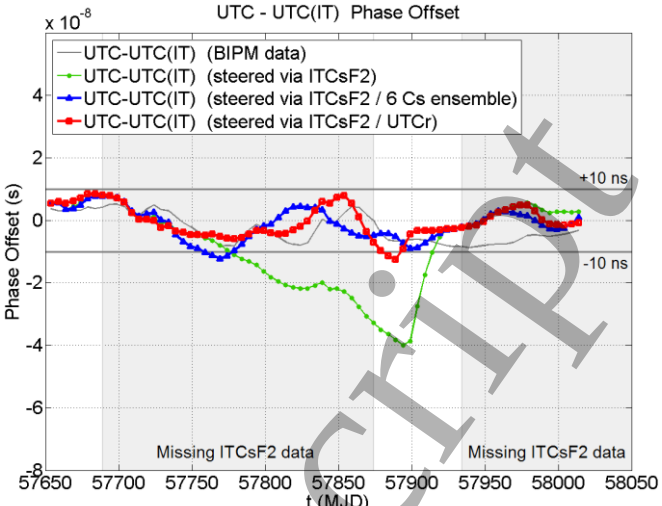

Figure 9. Test results for the mixed fountain/backup post-processing algorithm in the period from July 2016 to September 2017. The blue and red curves are the time scales obtained with the post-processing algorithm, with the ensemble clock and UTCr as backup reference, respectively. The green curve is the fountain-steered time scale obtained without post-processing, which is worse than the others due to the gaps in the fountain data (shaded regions).

3.2 On-line test results

After some recent hardware improvements, discussed in [12], the UTC(IT) generation is currently based on two parallel chains, each of which consists of an AHM and a micro-stepper. The two chains generate two independent time scales, which are then aligned in phase. The aligned time scales are then the input of a switch, whereas the output of the switch is, finally, UTC(IT). The switch can commute between the two input signals without discontinuities, in case of need, thus improving the robustness and reliability of UTC(IT). The selected signal is the master time scale, whereas the other one is the backup time scale.

Both the steering and the alignment of the two time scales were previously performed by skilled operators in a semi-automatic way. In order to improve the realization of UTC(IT), we planned a full automation of both the steering and the alignment process, which is now operational since the mid of January 2020. Before applying the new algorithms for the generation of UTC(IT), we performed an on-line test in our laboratory: we tested for 5 months, in the year 2019, the UTCr steering algorithm described in section 2.4. We used such algorithm for the generation of two independent time scales, which were then kept aligned in phase according to the procedure described in [13]. In this section, we present the final results from this 5-month on-line test, whereas only preliminary results were presented in [13]. In particular, here we focus the attention on the performance of the steering algorithm used to generate the two independent time scales (namely, the master and backup time scale), rather than on the performance of the alignment procedure. For a complete

review of the experiment, which led to the implementation of the automated procedure in January 2020, refer to [14].

The test results are shown in figure 10, in terms of phase offset between UTC/UTCr and the generated time scales. The test period goes from the mid of May 2019 to the mid of October 2019. The blue and the red curves are the master and the backup time scales, respectively, before the phase alignment of the backup to the master. Different markers and lighter colours have been used to distinguish the time offset measured with respect to UTC from the one measured with respect to UTCr. The two generated time scales intentionally start with a time offset of about 10 ns with respect to UTC/UTCr. For the first 10 days, only the Δf_0 component of the steering correction can be computed, because of the lack of previous [UTCr – time scale] phase offset data necessary for the computation of Δf_2, which is set to 0. The two timescales look almost flat with respect to UTC/UTCr within this initial 10-day period, as expected, with oscillations within 1 ns. Then, with the updated UTCr data, the steering correction becomes complete, and the Δf_2 component has the effect of increasing the frequency of the two time scales (starting from the epoch indicated by the vertical dashed line in the plot), in order to compensate the 10-ns time offset with respect to UTC/UTCr. As shown in the plot, the time offset slowly decreases in time: due to the conservative choice of N_{acc} = 30 days, the time offset comes close to zero after more than one month. After this transient, the two time scales align their frequency to that of UTCr and become again almost flat, as expected. The residual fluctuations are such that the UTCr-steered backup time scale remains well within ±5 ns from UTC/UTCr for the whole period from MJD 58650 to the end of the test, i.e. for about four consecutive months. The master time scale has some larger oscillations due to an anomalous behaviour of the master clock, which suddenly worsen its stability around MJD 58740, thus explaining the increase of the phase offset up to almost 10 ns. Nonetheless, also the UTCr-steered master time scale remains within ±5 ns from UTC/UTCr for the most of the time between MJD 58650 to the end of the test. This is in line with the results of the off-line test reported in section 3.1.4.

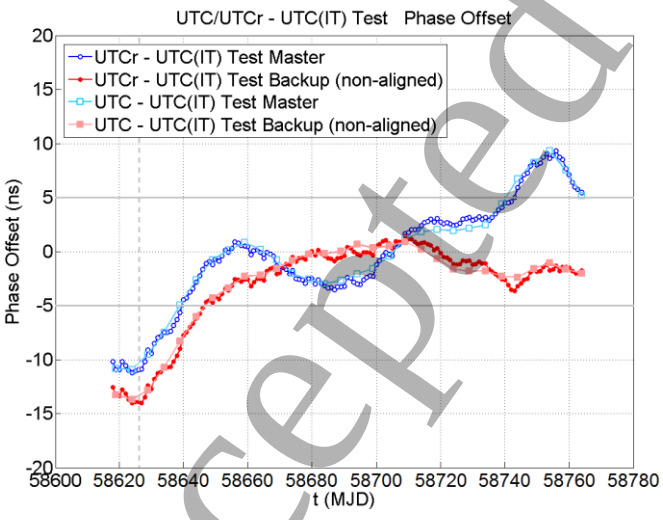

Figure 10. On-line test results for two independent time scales (master and non-aligned backup) steered with the UTCr steering algorithm, in the period from the mid of May 2019 to the mid of October 2019.

4. Steering algorithms comparison

In this section we compare the three steering algorithms presented above, by considering not only their performance, but also their cost, complexity, and autonomy.

The evaluation of the performance is based on the results presented in sections 3.1.2 to 3.1.4. In particular, we consider here: the 95th percentile of the phase offset between UTC and the generated time scale, over 1 year, reported in table 1; the stability of the generated time scale, measured with respect to UTC, reported in figure 11. The best performing algorithm is clearly the one based on the cesium fountain, both in terms of stability and phase offset with respect to UTC. The other two algorithms, based on an ensemble of 6 commercial Cs clocks and on UTCr (preferred to UTC, in this case), have similar performance, worse than those of the fountain steering algorithm. Anyway, both algorithms have advantages with respect to the cesium fountain algorithm, for example in terms of robustness and cost, two fundamental aspects for timing users in many industrial applications, such as 5G telecommunications, energy, finance, and GNSS [15]. Moreover, note that the metrics we used to quantify the performance of the algorithms not only depend on the quality of such algorithms, but also on the quality of the steered clock within the considered period, so that better results can be achieved with a better clock. As an example, consider the stability of the backup time scale realized during the on-line test discussed in section 3.2. We compute it on the whole period from MJD 58650 to the end of the test (see figure 10), and we report it in figure 12 (pink curve), along with the red curve taken from figure 11. In the considered range of averaging time values, the stability from the on-line test is clearly better than the stability plotted in figure 11, even if the same UTCr steering algorithm has been used: this is just because the stability of the steered AHM was better in the

Table 1. 95th percentile of the phase offset between UTC and the realized time scales, over 1 year, according to the results reported in sections 3.1.2 to 3.1.4.

UTC–UTC(IT) Phase Offset	
Steering Algorithm	**95th percentile over 1 year**
Cs Fountain	3 ns
Cs Ensemble Clock	7 ns
UTCr	6 ns

Table 2. A comprehensive comparison of the steering algorithms.

Steering Algorithm	Performance	Cost	Global Complexity	Autonomy
Cs Fountain	HIGH	HIGH	VERY HIGH	HIGH
Cs Ensemble Clock	MEDIUM	MEDIUM	MEDIUM/LOW	MEDIUM/LOW
UTCr	MEDIUM	LOW	LOW	LOW

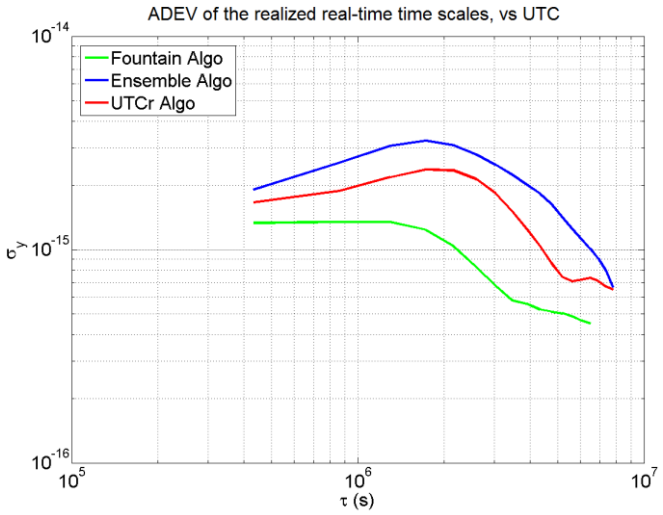

Figure 11. Stability (Allan deviation, vs UTC) of the realized time scales, according to the results reported in sections 3.1.2 to 3.1.4.

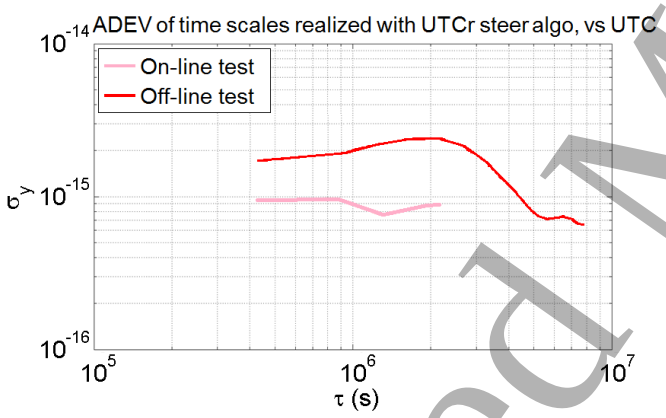

Figure 12. Stability of two time scales realized with the same UTCr steering algorithm, according to the results reported in sections 3.1.4 and 3.2 (backup time scale), where the difference is due to the different AHM behaviour.

case of the backup chain of the on-line test.

We can now compare the three steering algorithms from a more comprehensive point of view, as summarized in table 2. Besides the overall performance, we take into account: the cost, including hardware (clocks, instrumentation, etc.) and maintenance; the global complexity, i.e. the complexity of the algorithm itself and of the hardware implementation and maintenance; the autonomy, i.e. the ability to maintain good performance in absence of external information, i.e. when the link with UTC/UTCr is missing. As we noted above, the fountain steering algorithm has high performance, but it also has high cost and global complexity, since a Cs fountain is a complex prototype standard and a lot of effort is required to keep it operational in a continuous way. The ensemble-clock steering algorithm, instead, has medium performance, but in return it only has medium cost and complexity, since only a relatively small set of commercial Cs clocks is required. Finally, we note that the UTCr steering algorithm, which also has medium performance, is the one with the lowest cost and global complexity, since only a link with UTC/UTCr is required, which is anyway required also by the other two steering algorithms to guarantee the time accuracy of the generated time scale. However, in contrast with the other two algorithms, it also has a low autonomy, since it is entirely based on the external information coming from the BIPM (i.e., UTCr data).

5. Conclusions

We have presented a time-scale algorithm for the automated generation of an accurate, stable, robust, and reliable real-time time scale. It is based on three different steering algorithms, using as steering references a laboratory primary frequency standard, an ensemble clock made of commercial frequency standards, and UTC/UTCr. We have tested the three steering algorithms, and an example of post-processing algorithm which combines their outputs, on real data from INRiM's time and frequency laboratory, and we have discussed in this work the promising results we have obtained. In particular, we have shown how the realized time scale can maintain a phase offset of a few nanoseconds from UTC for long periods of time, reaching a long-term stability of some units in 10^{-16}. The three steering algorithms have been compared in terms of performance, cost, complexity and autonomy, providing helpful information to anyone who can be interested in applying this or a similar method for generating a real-time time scale. Moreover, the proposed time-scale algorithm has a modular design that allows to adapt it to the actual capabilities of a generic time laboratory, to the real needs of the possible users and to the different requirements of the several possible applications. Among them, our time-scale algorithm can be used for generating a UTC(k) or a GNSS reference time.

In a future work, we will discuss the effect of the Δf_1 component of the frequency correction, as well as a different averaging algorithm for cesium and maser clocks.

Acknowledgements

The authors warmly thank: Dr. Patrizia Tavella, for her contribution to the algorithms design when formerly INRiM; Dr. Filippo Levi and Prof. Giovanni Costanzo, for providing ITCsF2 data; all the other colleagues from INRiM's time and frequency laboratory, for providing support for the on-line test of the algorithms.

References

[1] P. Defraigne et al. 2017 Demonstrator of time services based on European GNSS signals: the H2020 DEMETRA project *Proc. 48th Annual Precise Time and Time Interval Systems and Applications Meeting*, 127-137

[2] Signorile G, Tavella P, Calonico D, Levi F, Costanzo G, Cerretto G, Costa R, Cantoni E and Sesia I 2015 Preliminary step for a UTC(IT) steering algorithm based on the ITCsF2 primary frequency standard measurements *Proc. 2015 Joint Conference of IEEE International Frequency Control Symposium & European Frequency and Time Forum*, 260-264

[3] Galleani L, Signorile G, Formichella V and Sesia I 2019 Generating a real-time time scale with an ensemble clock and a primary frequency standard *Proc. 2019 Joint Conference of IEEE International Frequency Control Symposium & European Frequency and Time Forum*

[4] Galleani L and Tavella P 2010 Time and the Kalman filter *IEEE Control Systems Magazine* **30**, 2, 44-65

[5] Farina M, Galleani L, Tavella P and Bittanti S 2010 A control theory approach to clock steering techniques *IEEE Trans. Ultrason., Ferroelect., Freq. Control* **57**, 10, 2257-2270

[6] Sesia I, Cantoni E, Cernigliaro A, Signorile G, Fantino G and Tavella P 2016 An efficient and configurable preprocessing algorithm to improve stability analysis *IEEE Trans. Ultrason., Ferroelect., Freq. Control* **63**, 4, 575-581

[7] Galleani L and Tavella P 2017 Robust detection of fast and slow frequency jumps of atomic clocks *IEEE Trans. Ultrason., Ferroelect., Freq. Control* **64**, 2, 475-485

[8] Rovera G D, Bize S, Chupin B, Guéna J, Laurent Ph, Rosenbusch P, Uhrich P and Abgrall M 2016 UTC(OP) based on LNE-SYRTE atomic fountain primary frequency standards *Metrologia* **53**, S81

[9] Bauch A, Weyers S, Piester D, Staliuniene E and Yang W 2012 Generation of UTC(PTB) as a fountain-clock based time scale *Metrologia* **49**, 180-188

[10] Petit G, Arias F, Harmegnies A, Panfilo G and Tisserand L 2014 UTCr: a rapid realization of UTC *Metrologia* **51**, 33-39

[11] Levi F, Calonico D, Calosso C E, Godone A, Micalizio S and Costanzo G 2014 Accuracy evaluation of ITCsF2: a nitrogen cooled caesium fountain *Metrologia* **51**, 270-284

[12] Sellone M, Costa R, Mura A, Cerretto G, Levi F and Siccardi M 2018 Improvements on the UTC(IT) time scale at INRIM *Proc. 2018 European Frequency and Time Forum*, 188-191

[13] Signorile G, Formichella V, Sellone M, Mura A, Siccardi M, Rovera G D, Sesia I and Levi F 2019 Reliable and robust UTC(IT) generation based on master and backup time scales alignment at INRiM *Proc. 2019 IEEE International Workshop on Metrology for AeroSpace*

[14] Formichella V, Signorile G, Thai T T, Perucca A, Cantoni E, Sellone M, Mura A, Siccardi M, Rovera G D, Sesia I and Levi F 2020 Reliable and robust real-time time scale generation: developments and experimental results at INRiM *Proc. 2020 Precise Time and Time Interval Systems and Applications Meeting*

[15] Tavella P and DEMETRA Consortium 2016 Time Dissemination Services: the Experimental Results of the European H2020 DEMETRA Project *Proc. 2016 IEEE International Frequency Control Symposium*