POLITECNICO DI TORINO

MASTER's Degree in INGEGNERIA MATEMATICA

MASTER's Degree Thesis

Prediction of atomic clocks behaviour for UTC calculation: a statistical analysis based on clock type.

Supervisors

Prof. ENRICO BIBBONA

Dr. GIANNA PANFILO

Candidate

ARIANNA ABIS

MARCH 2024

Summary

This thesis presents collaborative research conducted at the Bureau International des Poids et Mesures (BIPM), specifically within its Time Department. The BIPM serves as the international authority on metrology, facilitating global uniformity in measurements and standards. In this framework, the Time Department is specialized in analyzing time differences data, sourced from atomic clocks distributed across laboratories worldwide. These data are optimally combined to establish a stable and precise time-scale, the Universal Coordinate Time (UTC) scale, used as a global time reference.
The thesis commences with a comprehensive literature review addressing the current methodologies and their evolution. Thereafter, the study focuses on the implementation and refinement of algorithms for predicting phase error terms. Notably, all clocks contributing to UTC exhibit a deviation from the nominal frequency defining the second, which results in a phase error, as the error in frequency integrates over time. Accurately predicting and correcting the deterministic component of the phase error terms for each clock is essential in maintaining the precision and stability of the UTC time scale. While the current algorithm utilizes a quadratic polynomial, the research explored alternative approaches tailored to specific types of clocks. In fact, there are three main kinds of atomic clocks: cesium (commercial clocks), hydrogen masers (very stable but not accurate) and rubidium fountains (extremely stable and accurate, lose about 3 nanoseconds in one year).
In particular, linear models of the phase deviations were proposed for fountains and cesium clocks, addressing the absence of drift which is instead accounted for in the quadratic model. Moreover, a novel method for estimating the linear coefficient, employing the least squares technique instead of the conventional last-first technique, was proposed and investigated. Experimental findings and statistical analysis, detailed within the thesis, provide insights into the efficacy of these methodologies, contributing to the optimization of global time standards.

Table of Contents

List of Tables

List of Figures

Acronyms

MJD	Modified Julian Date
UTC	Universal Coordinate Time
TAI	International Atomic Time
EAL	Free Atomic Time Scale
PSFS	Primary Secondary Frequency Standard
TT	Terrestrial Time
WPM	White Phase Modulation
FPM	Flicker Phase Modulation
WFM	White Frequency Modulation
FFM	Flicker Frequency Modulation

RWFM

Random Walk Frequency Modulation

Introduction

The concept of time has been at the center of significant philosophical and scientific discussions for centuries, with many prominent thinkers who asked themselves: "What is time?".
In his paper "Time and Frequency Characterization, Estimation and Prediction of Precision Clocks and Oscillators", published in 1987, David W. Allan gives an answer to this question: "The fact is that time as we now generate it is dependent upon defined origins, a defined resonance in the cesium atoms[...]. Hence, at a significant level time - as man generates it by the best means available to him - is an artifact."([1]).
This thesis explores the theory of time measurements and time scales, after the redefinition of the second in 1967, with the employment of atomic clocks and the related work to account for their inaccuracy and lack of stability.
In particular it focuses on the effort to maintain a global reference time scale, UTC time scale. This effort is coordinated at the Time Department of the Bureau International des Poids et Mesures (BIPM), the international authority on time metrology.

The first two chapters of this thesis will delve into the existing literature concerning current modeling standards for atomic clocks.

The first chapter will provide a comprehensive overview of the topic, elucidating the definition of time scales, and delineating the pivotal role of the BIPM within the timing community. The current algorithm employed for predicting atomic clocks errors will be thoroughly analyzed.

In the second chapter, the focus shifts to stochastic processes describing clocks random noises, both in time and frequency domains. The initial part of this chapter will explain the prevalent methodology for discerning clock errors — the Allan variance — elucidating how it is obtained as an estimator for time-dependent frequency instabilities. The latter part will center on stochastic calculus. First there will be a review on fundamental results, then it will be presented a literature case of modeling of atomic clock behavior via a generalized linear stochastic differential equation.

Finally, the third chapter embarks on original research. Initially, I will propose a mathematical framework which could be underlying the definition of fundamental metrological quantities. Subsequently, insights obtained from extensive historical data will be presented, facilitating an evaluation of the current algorithm's efficacy while identifying potential areas for enhancement. Proposed algorithm modifications, their implementation, and ensuing results will be detailed through figures and statistical analysis. In the end, a hybrid model integrating deterministic algorithms with stochastic elements will be applied to the clocks.

Chapter 1

Overview of UTC Calculation

This work started with a review and comprehensive analysis of the work carried out at the Time Department of the Bureau International des Poids et Mesures, in short BIPM. The BIPM is the International Organization, established by the Metre Convention in 1875, which guarantees the maintenance of measurements standards, through a coordinated effort with its Member States. To reach this objective, the Time Department, among other fundamental tasks, calculates and distributes every month the Universal Coordinate Time, in short UTC.
The 13th General Conference on Weights and Measures (CGPM) decided that the second is the duration of 9 192 631 770 periods of the radiation corresponding to the transition between two hyperfine levels of the ground state of caesium 133 atom. From this definition began the adoption of a time scale built by accumulating atomic seconds, which are calculated using atomic clocks. There are nowadays almost 450 atomic clocks distributed worldwide, of different types: 45% are cesium clocks, 45% are hydrogen masers, and there are 5 Rubidium fountains. There are also other types of devices which will not be considered in this work. Some of the devices are called Primary and Secondary Frequency Standards (PSFS), because they reproduce the frequency defining the second with extremely high accuracy and stability. In this work we will focus on the differences in terms of frequency accuracy and stability for cesium clocks, h-Masers and Rubidium fountains.
Each laboratory k in the world, having one or more atomic clocks, also calculates its approximated UTC value, indicated by UTC(k).
At the BIPM, clock readings comparisons provided by each laboratory contributing to UTC time scale, and clock readings comparisons between different laboratories, arrive through a system of time links, and are combined in an optimal way to guarantee reliability, long-term frequency stability and high precision of the global

reference time scale. The frequency stability of a time scale represents its capacity to maintain a fixed ratio between its unitary scale interval and its theoretical counterpart. The frequency accuracy represents the aptitude of its unitary scale interval to reproduce its theoretical counterpart.
Going in more details, the calculation of UTC is carried out in three steps:

- Computation of EAL (Free Atomic Time Scale): EAL is computed as a weighted average of atomic clocks readings, with weights to optimize the long-term frequency stability of the time scale, obtaining a level of performance beyond what can be realized by any individual clock in the ensemble. For EAL computation the BIPM tries to use as many clocks as possible, aiming for it to be as reliable as possible.
- Computation of TAI (International Atomic Time): TAI is computed by steering the frequency of EAL to maintain agreement with the definition of the second SI (by comparison with PSFS). This ensures time scale accuracy.
- Computation of UTC: UTC is computed from TAI by insertion of leap seconds. In fact the time scale computed with the current definition of the second can progressively distance itself from the time scale derived from the rotation of the Earth. When this distance exceeds 1 second, 1 second is added to UTC. Nowadays there is an offset of 37 seconds between TAI and UTC, with the last leap second added in 2017.

Another time scale which is worth mentioning is TT(BIPM). It is computed annually at the BIPM and is considered the most stable reference in frequency, given that it is obtained using only PSFS. This time scale is used, for instance, to evaluate the performance of EAL.
Each month, the work of the BIPM is synthesized in the publication of the so called Circular T, which contains for each laboratory k the differences UTC - UTC(k) at 5 days intervals over the past month (calculation in post-real time over one-month data batches). This allows the laboratories to correct their UTC(k), in order to maintain an offset from UTC whitin $100ns$.

1.1 Atomic Clocks behaviour

The content of this section is based mainly on [2].
To be compliant with the definition of the second, an atomic clock is ideally a noiseless, non-drifting oscillator. It provides as its output an electronic signal whose instantaneous voltage value, in the ideal case, is:

$$V(t) = V_0 sin\left(2\pi\nu_0 t\right) \tag{1.1}$$

where:

- V_0 is the nominal amplitude of the signal.
- ν_0 is the nominal frequency of the signal.

Nonetheless, no clock is ideal. There is thus the need to insert elements of noise in the above equation:

$$V(t) = [V_0 + \epsilon(t)]\, sin\,(2\pi\nu_0 t + \phi(t)) \tag{1.2}$$

Two terms were inserted:

- $\epsilon(t)$ is a function for the amplitude random fluctuations.
- $\phi(t)$ is a function for the phase random fluctuations.

From now on, we choose to disregard term $\epsilon(t)$ by assuming that amplitude fluctuations are negligible around V_0.
Generally speaking, if we take a wave signal: $f(t) = sin(wt + \phi_0) = sin(2\pi ft + \phi_0)$ it is clear that the angular frequency w can be obtained as the derivative of the sine argument with respect to time, while frequency f, as the derivative of the sine argument divided by 2π. If we substitute a fixed ϕ_0 with a variable $\phi(t)$, it follows that we can define the instantaneous frequency $\nu(t)$ as:

$$\nu(t) = \nu_0 + \frac{1}{2\pi}\frac{d\phi(t)}{dt} \tag{1.3}$$

The average value of $\nu(t)$ over a time interval τ beginning at t_k provides a more useful quantity directly related to an experimental result:

$$\langle \nu(t) \rangle_{t_k,\tau} = \frac{n_k}{\tau} \tag{1.4}$$

where n_k is the number of cycles of the signal during the time interval τ beginning at t_k.
The quantity $\nu(t)$ cannot be measured since infinite bandwidth measurement equipment is not available.
An useful quantity in metrology is the dimensionless frequency deviation at time t:

$$y(t) = \frac{\nu(t) - \nu_0}{\nu_0} = \frac{1}{2\pi\nu_0}\frac{d\phi_t}{dt} \tag{1.5}$$

We can rewrite 1.2 as:

$$V(t) = V_0 sin\,(2\pi\nu_0(t + x(t))) \tag{1.6}$$

where

$$x(t) = \frac{\phi_t}{2\pi\nu_0} \tag{1.7}$$

is called normalized phase deviation and has got the dimension of time. $x(t)$ is the instantaneous time error of a clock having an instantaneous frequency $\nu(t)$. In simpler terms is the reading error of the non-precise clock with respect to a perfect clock with frequency ν_0 and time reading t. We can immediately see the relationship between the normalized phase deviation and the normalized frequency deviation at time t:

$$y(t) = \frac{dx(t)}{dt} \tag{1.8}$$

which is: the frequency deviation at time t is the derivative of the normalized phase deviation at time t. The quantities $x(t)$ and $y(t)$ can indeed be measured against a reference clock by an electronic counter and they are affected by the same noises that affect clock signals.
There are two main aspects to be kept in mind for understanding clock measurements in practice:

- There is always a measured clock and a reference clock, the measured clock is both synchronized and compared to the reference clock. Therefore, the measurement represents the relative difference between the two signals. Measurement of a clock has no meaning unless it is compared to a reference.

- The error in a clock varies over time, so it is important that clock errors are measured over a sufficiently long time period to thoroughly understand the quality of a clock.

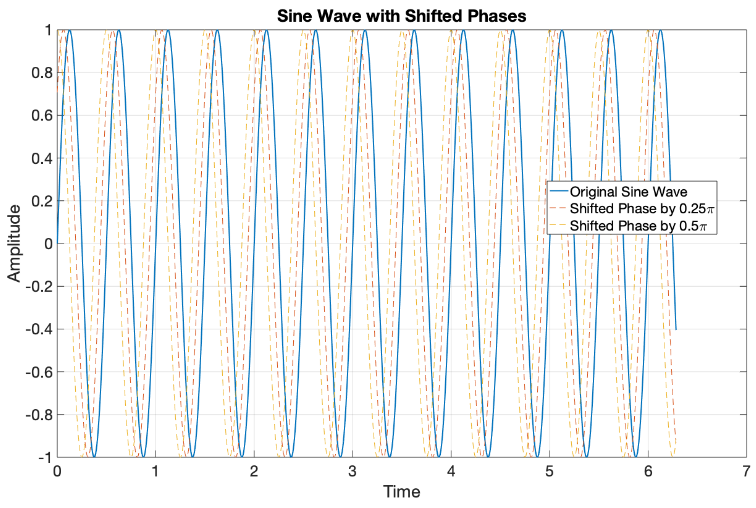


Figure 1.1: Example of sine-wave with amplitude 1 and phase 0, and its shifted versions with phase 0.25π and 0.5π.

A clock deviates from the ideal for two categories of reasons: systematic deviations and random deviations, depending on the type of clock and the considered period. Everything systematic can be modeled (and thus corrected). For example a clock can have a frequency offset y_0 at time t_0, a time offset x_0 and a constant frequency drift D. In this case we may write:

$$x(t) = x_0 + y_0 t + \frac{1}{2} D t^2 + \epsilon(t) \tag{1.9}$$

with $\epsilon(t)$ being a random deviation. Of course, the model in 1.9 does not apply in all cases e.g. in some oscillators the drift is absent or not constant.

1.2 EAL Calculation

We will focus on the calculation of EAL, which is the first step in UTC calculation, and the only step involving prediction of clocks behaviour.

1.2.1 Estimate of EAL parameters

The outline of the old algortihm is contained in [3]. Two hypothesis are made on EAL:

- EAL is calculated separately for each interval of time. Two subsequent intervals of time are denoted by $I_{k-1} = (t_{k-1}, t_{k-1} + q_{k-1})$, $I_k = (t_k, t_k + q_k)$ where $t_{k-1} + q_{k-1} = t_k$. In practice, each of these intervals is one month long, and the values t_k are Modified Julian Dates (MJD), which are integer counters of the days beginning at midnight on November 17, 1858. This implies that approximately $t_{k+1} = t_k + T$ with $T = 30$ days.

- the readings of clocks and time scales are available at instants t of an ideal, uniform time scale.

Despite the second hypothesis, $EAL(t)$ for interval I_k is always computed at discrete time instants $t_{k_j} = t_k + j\frac{T}{6}$ with $j = 0, \ldots, 6$ (or $t_{k_j} = t_k + j\frac{T}{7}$ with $j = 0, \ldots, 7$), so that $t_{k_0} = t_k$ and $t_{k_6} = t_k + T = t_{k+1}$ We will use index i for clocks and we will denote the reading of clock i at time $t = t_{k_j}$ for some j by $h_i(t)$ (in some cases this is the value $UTC(i)$). Each month every clock gets assigned a weight (we will explore the weighting system in further chapters), we denote the weight for clock i during interval I_k as $w_{i,k}$. Given that during interval I_k there are N active clocks, for $t \in I_k$ it was defined:

$$EAL(t) = \sum_{i=1}^{N} w_{i,k} h_i(t) + a_k + b_k(t - t_k) \tag{1.10}$$

In this definition the averaged readings of clocks are corrected with a linear term, under the assumption that no clock is affected by a drift (confront with section 1.1). To be more precise we are actually defining a different $EAL(t)$ depending on the interval I_k we are on. In particular, if we consider two consequent intervals we are defining a piece-wise function:

$$\begin{cases} EAL(t) = \sum_{i=1}^{N} w_{i,k-1} h_i(t) + a_{k-1} + b_{k-1}(t - t_{k-1}) & t \in (t_{k-1}, t_k) \\ EAL(t) = \sum_{i=1}^{N} w_{i,k} h_i(t) + a_k + b_k(t - t_k) & t \in (t_k, t_k + q_k) \end{cases} \tag{1.11}$$

It should be noted that although we gave a theoretical definition of $EAL(t)$, and we require it to have desirable properties, its value is never accessible (it is the same for TAI and UTC), for the same reason why it is impossible to measure absolute time. What we can measure indeed are time differences between two clocks, or the time of an event with reference to a particular clock. As stated in [1]:
"...every clock disagrees with every other clock essentially always, and no clock keeps ideal or "true" time in an abstract sense except as we may choose to define it ".
In fact the output of the calculation will not be the values of EAL calculated at equally spaced times, but the values of difference between EAL, the optimal time scale in the frequency-stability sense, and clock readings.

For a start, we want a timescale that is continuous, so we impose continuity at the common date $t = t_k$ between two intervals.

$$\sum_{i=1}^{N} w_{i,k-1} h_i(t_k) + a_{k-1} + b_{k-1} q_{k-1} = \sum_{i=1}^{N} w_{i,k} h_i(t_k) + a_k \tag{1.12}$$

Since we want a time scale that is also continuous in frequency (with continuous derivative) we get the following equation:

$$\sum_{i=1}^{N} w_{i,k-1} \left.\frac{dh_i(t)}{dt}\right|_{t_k} + b_{k-1} = \sum_{i=1}^{N} w_{i,k} \left.\frac{dh_i(t)}{dt}\right|_{t_k} + b_k \tag{1.13}$$

Nonetheless there is no way to compute analitically the derivative of the clock reading function, so it is necessary to consider mean rates elaborated over the intervals I_{k-1}, I_k. In fact we use a finite forward difference approximation of the derivative. We have to pay attention here to the fact that in I_{k-1}, t_k is the right extreme of the interval, while in I_k it is the left extreme, so that the approximation of the derivative changes between the left and right side of the equal sign. In conclusion we get:

$$b_k = b_{k-1} + \sum_{i=1}^{N} w_{i,k-1} \frac{h_i(t_k) - h_i(t_{k-1})}{q_{k-1}} - \sum_{i=1}^{N} w_{i,k} \frac{h_i(t_{k+1}) - h_i(t_k)}{q_k} \tag{1.14}$$

As already stated above, in the real world time readings alone are never accessible. For this reason we introduce time readings differences in various forms, that will be inserted in the equations. We denote

$$x_{i,j}(t) = h_j(t) - h_i(t) \tag{1.15}$$

which is the reading difference between clocks i and j at time t. These data are actually available, for some i and j, thanks to the use of time links connecting laboratories across the world.

During interval I_k we also want to define how much the reading of clock j a time $t \in I_k$ differs from the weighted average of the readings of all clocks: $z_{j,k}(t) = \sum_{i=1}^{N} w_{i,k} h_i(t) - h_j(t)$.
This definition allows us to rewrite equation 1.10 in terms of the reading of clock i (so that we get N equations with the same identical value) as:

$$EAL(t) = h_i(t) + z_{i,k}(t) + a_k + b_k(t - t_k) \qquad t \in I_k; i \in \{1, \dots, N\} \tag{1.16}$$

and time continuity condition becomes:

$$z_{i,k-1}(t_k) + a_{k-1} + b_{k-1} q_{k-1} = z_{i,k}(t_k) + a_k \qquad i \in \{1, \dots, N\} \tag{1.17}$$

By again rewriting equation 1.14 in terms of clock readings differences we obtain:

$$\begin{aligned} b_k = &b_{k-1} + \frac{z_{i,k-1}(t_k) - z_{i,k-1}(t_{k-1})}{q_{k-1}} - \frac{z_{i,k}(t_{k+1}) - z_{i,k}(t_k)}{q_k} + &(1.18)\\ &+ \frac{h_i(t_k) - h_i(t_{k-1})}{q_{k-1}} - \frac{h_i(t_{k+1}) - h_i(t_k)}{q_k} \qquad i \in \{1, \dots, N\} &(1.19) \end{aligned}$$

To simplify this formula in order to be able to work only with time differences, the following is assumed: the (real) mean frequency of any clock is the same in interval I_{k-1} as during I_k. This assumption leads us to have N different estimations $b_{i,k}$ for $i = 1, \dots, N$, each obtained for a given clock.

$$b_{i,k} = b_{k-1} + \frac{z_{i,k-1}(t_k) - z_{i,k-1}(t_{k-1})}{q_{k-1}} - \frac{z_{i,k}(t_{k+1}) - z_{i,k}(t_k)}{q_k} \qquad i \in \{1, \dots, N\} \tag{1.20}$$

At this point we can recover b_k as a weighted average of $b_{i,k}$ during I_k, with the hypothesis that the weights are the same considered in the computation of EAL. Thus we obtain:

$$b_k = \sum_{i=1}^{N} w_{i,k} b_{i,k} \tag{1.21}$$

The weighted average emphasizes the role of clocks having the best long-term frequency stability.
When computing the average, the last term in 1.20 cancels out, so that we only need quantities belonging to I_{k-1} in the computation of b_k. In fact we know that $\sum_{i=1}^{k} w_{i,k} z_{i,k}(t) = 0$
From the first definition of EAL 1.10 we can write the individual corrections of clock i as:

$$z_{i,k}(t) = EAL(t) - h_i(t) - a_k - b_k(t - t_k) \tag{1.22}$$

By substituting equation above in equation 1.17:

$$a_k = EAL(t_k) - h_i(t_k) - a_{k-1} - b_{k-1}q_{k-1} + a_{k-1} + b_{k-1}q_{k-1} - \sum_{i=1}^{N} h_i(t_k) + h_i(t) \tag{1.23}$$

and doing the necessary simplifications, we obtain a_k:

$$a_k = \sum_{i=1}^{N} w_{i,k} \left[EAL(t_k) - h_i(t_k) \right] \tag{1.24}$$

It is actually more convenient, how it is done in paper [4], to define a different value for each clock: $a_{i,k-1} = EAL(t_k) - h_i(t_k)$. This makes sense since our a_k is simply the average of $a_{i,k-1}$ for all clocks in the interval I_k. We use the suffix $k-1$ to highlight the fact that the computation happens at the end of interval I_{k-1}.
Using again equation 1.22 and substituing it in equation 1.21 we obtain:

$$b_k = \sum_{i=1}^{N} w_{i,k} \left[\frac{[EAL(t_k) - h_i(t_k)] - [EAL(t_{k-1}) - h_i(t_{k-1})]}{q_{k-1}} \right] \tag{1.25}$$

we can define:

$$b_{i,k-1} = \frac{[EAL(t_k) - h_i(t_k)] - [EAL(t_{k-1}) - h_i(t_{k-1})]}{q_{k-1}} \qquad i \in \{1, \ldots, N\} \tag{1.26}$$

with the same reasoning as before.
And thus at any date t belonging to interval I_k we get the following calculation of EAL:

$$EAL(t) = \sum_{i=1}^{N} w_{i,k} \left[h_i(t) + EAL(t_k) - h_i(t_k) + b_{i,k-1}(t - t_k) \right] \tag{1.27}$$

After defining

$$h'_i(t) = EAL(t_k) - h_i(t_k) + b_{i,k-1}(t - t_k) \qquad i \in \{1, \dots, N\}$$

We obtain:

$$EAL(t) = \sum_{i=1}^{N} w_{i,k} \left[h_i(t) + h'_i(t)\right] \tag{1.28}$$

We stress out once again how the terms $a_{i,k-1}$ and $b_{i,k-1}$ involve only quantities available on the preceding interval I_{k-1}, so that they can be calculated the preceding month, and this is what is done in practice at the BIPM. Parameters for the current month are calculated the month before. That is also intuition on why the term $h'_i(t)$ is called in literature a prediction.

1.2.2 Inclusion of the drift

By citing [2]: "Frequency drifts are systematic variations that may be, for example, due to the aging of the resonator material. These extremely slow changes are often referred to as "long-term instabilities" and expressed in terms of parts in 10^x of frequency change per hour/day/month or year. Whenever possible, systematics should be removed before statistical treatment."
Since 2011 it is decided that the model for EAL should also contain a quadratic term, taking into account that clocks, in particular H masers, are affected by a systematic drift, which means that they do not only have a frequency offset with respect to primary frequency standards, but their frequency increases (or decreases) linearly with time ([4]). Luckily, this is very predictable. Similarly to what we have done before, we define, for t in I_k:

$$EAL(t) = \sum_{i=1}^{N} w_{i,k} h_i(t) + a_k + b_k(t - t_k) + \frac{1}{2} c_k (t - t_k)^2 \tag{1.29}$$

By imposing the following three conditions:

- no time steps (continuity of EAL)
- no frequency steps (continuity of the first derivative of EAL)
- no change in frequency drift (continuity of the second derivative of EAL)

we can derive the terms a_k, b_k, c_k.
By imposing continuity of EAL in $t = t_k$:

$$\sum_{i=1}^{N} w_{i,k-1} h_i(t_k) + a_{k-1} + b_{k-1} q_{k-1} + \frac{1}{2} c_{k-1} q_{k-1}^2 = \sum_{i=1}^{N} w_{i,k} h_i(t_k) + a_k \tag{1.30}$$

We subtract $h_i(t_k)$ to get the terms $z_{i,k}(t_k)$ and $z_{i,k-1}(t_k)$ as seen before:

$$z_{i,k-1}(t_k) + a_{k-1} + b_{k-1}q_{k-1} + \frac{1}{2}c_{k-1}q_{k-1}^2 = z_{i,k}(t_k) + a_k \qquad i \in \{1, \dots, N\} \tag{1.31}$$

At this point we want to express the terms $z_{i,k}(t)$ using a_k, b_k, c_k, by subtracting $h_i(t)$ to both side of equation 1.29.

$$z_{i,k}(t) = (EAL(t) - h_i(t)) - a_k - b_k(t - t_k) - \frac{1}{2}c_k(t - t_k)^2 \qquad i \in \{1, \dots, N\} \tag{1.32}$$

And defining $x_i(t) = EAL(t) - h_i(t), \quad \forall i \in \{1, \dots, N\}$ we obtain:

$$z_{i,k}(t) = x_i(t) - a_k - b_k(t - t_k) - \frac{1}{2}c_k(t - t_k)^2 \qquad i \in \{1, \dots, N\} \tag{1.33}$$

By using the substitution above for $z_{i,k-1}(t_k)$:

$$a_k = x_i(t_k) - z_{i,k}(t_k) = EAL(t_k) - h_i(t_k) - \sum_{i=1}^{N} w_{i,k}h_i(t_k) + h_i(t_k) \tag{1.34}$$

$$= EAL(t_k) - \sum_{i=1}^{N} w_{i,k}h_i(t_k) \tag{1.35}$$

$$= \sum_{i=1}^{N} w_{i,k}\left[EAL(t_k) - h_i(t_k)\right] \qquad i \in \{1, \dots, N\} \tag{1.36}$$

Again we have re conducted ourselves to the definition of a_k as $\sum_{i=1}^{N} w_{i,k}a_{i,k-1}$ where $a_{i,k-1} = [EAL(t_k) - h_i(t_k)]$ as in [4] [5].

To derive coefficient b_k, first we need to write $EAL(t)$ in terms of clock differences:

$$EAL(t) = z_{i,k}(t) + h_i(t) + a_k + b_k(t - t_k) + \frac{1}{2}c_k(t - t_k)^2 \qquad i \in \{1, \dots, N\} \tag{1.37}$$

Now, by approximating the derivative using finite difference approximation, and by imposing continuity, we get the following:

$$\frac{z_{i,k-1}(t_k) - z_{i,k-1}(t_{k-1})}{q_{k-1}} + \frac{h_i(t_k) - h_i(t_{k-1})}{q_{k-1}} + b_{k-1} + c_{k-1}q_{k-1} = \tag{1.38}$$

$$= \frac{z_{i,k}(t_{k+1}) - z_{i,k}(t_k)}{q_{k-1}} + \frac{h_i(t_{k+1}) - h_i(t_k)}{q_k} + b_k \qquad i \in \{1, \dots, N\} \tag{1.39}$$

As before we do the assumption that the real mean frequency does not change for the clock, so that we can define, separately for each clock:

$$b_{i,k} = \frac{z_{i,k-1}(t_k) - z_{i,k-1}(t_{k-1})}{q_{k-1}} - \frac{z_{i,k}(t_{k+1}) - z_{i,k}(t_k)}{q_{k-1}} + b_{k-1} + c_{k-1}q_{k-1} \tag{1.40}$$

$$\forall i \in \{1, \ldots, N\} \tag{1.41}$$

Now $b_k = \sum_{i=1}^{N} w_{i,k} b_{i,k}$.
By applying again substitution 1.33:

$$b_{i,k} = \frac{x_i(t_k) - x_i(t_{k-1})}{q_{k-1}} + \frac{1}{2}c_{k-1}q_{k-1} - \frac{z_{i,k}(t_{k+1}) - z_{i,k}(t_k)}{q_k} \tag{1.42}$$

from which:

$$b_k = \sum_{i=1}^{N} w_{i,k} \left[\frac{x_i(t_k) - x_i(t_{k-1})}{q_{k-1}} \right] + \frac{1}{2}c_{k-1}q_{k-1} \tag{1.43}$$

given that the last term in 1.42 cancels out in the computation of the mean, as already proved.
While the term a_k staid unchanged with respect to the old algorithm, the term b_k is modified with the additional term $\frac{1}{2}c_{k-1}q_{k-1}$.
For what concerns c_k, by starting from equation 1.37 and computing the second derivative, the continuity equation states:

$$\left.\frac{d^2 z_{i,k-1}(t)}{dt}\right|_{t_k} + \left.\frac{d^2 h_i(t)}{dt}\right|_{t_k} + c_{k-1} = \left.\frac{d^2 z_{i,k}(t)}{dt}\right|_{t_k} + \left.\frac{d^2 h_i(t)}{dt}\right|_{t_k} + c_k \qquad \forall i \in \{1, \ldots, N\} \tag{1.44}$$

We decide to use again the finite difference approximation method, this time with a second order approximation:

$$\frac{z_{i,k-1}(t_{k-2}) - 2z_{i,k-1}(t_{k-1}) + z_{i,k-1}(t_k)}{q_{k-1}^2} + \frac{h_i(t_{k-2}) - 2h_i(t_{k-1}) + h_i(t_k)}{q_{k-1}^2} + c_{k-1} \tag{1.45}$$

$$= \frac{z_{i,k}(t_{k-1}) - 2z_{i,k}(t_k) + z_{i,k}(t_{k+1})}{q_k^2} + \frac{h_i(t_{k-1}) - 2h_i(t_k) + h_i(t_{k+1})}{q_k^2} + c_k \tag{1.46}$$

$$\forall i \in \{1, \ldots, N\} \tag{1.47}$$

Supposing that the real mean frequency does not change for a single clock, the two terms containing the second order finite differences for the clock readings both cancel out, leaving us with:

$$c_{i,k} = c_{k-1} + \frac{z_{i,k-1}(t_{k-2}) - 2z_{i,k-1}(t_{k-1}) + z_{i,k-1}(t_k)}{{q_{k-1}}^2} - \tag{1.48}$$

$$- \frac{z_{i,k}(t_{k-1}) - 2z_{i,k}(t_k) + z_{i,k}(t_{k+1})}{{q_k}^2} \qquad \forall i \in \{1, \ldots, N\} \tag{1.49}$$

And again defining c_k as $\sum_{i=1}^N w_{i,k} c_{i,k}$:

$$c_k = c_{k-1} + \sum_{i=1}^N w_{i,k} \frac{z_{i,k-1}(t_{k-2}) - 2z_{i,k-1}(t_{k-1}) + z_{i,k-1}(t_k)}{{q_{k-1}}^2} \tag{1.50}$$

Now substituting equation 1.33 and making all the necessary calculations, with the additional assumption that all intervals have equal length (which subsists in practice):

$$c_k = \sum_{i=1}^N w_{i,k} \left[\frac{x_i(t_{k-2}) - 2x_i(t_{k-1}) + x_i(t_k)}{q_{k-1}^2} \right] \tag{1.51}$$

So defining: $c_{i,k-1} = \left[\frac{x_i(t_{k-2}) - 2x_i(t_{k-1}) + x_i(t_k)}{q_{k-1}^2} \right]$ we can also rewrite the definition of b_k as:

$$b_k = \sum_{i=1}^N w_{i,k} \left[\frac{x_i(t_k) - x_i(t_{k-1})}{q_{k-1}} + \frac{1}{2} c_{i,k-1} q_{k-1} \right] \tag{1.52}$$

In conclusion applying the three constraints:

- $a_k = \sum_{i=1}^N w_{i,k} a_{i,k-1} = \sum_{i=1}^N w_{i,k} [EAL(t_k) - h_i(t_k)]$
- $b_k = \sum_{i=1}^N w_{i,k} \left[b_{i,k-1} + \frac{1}{2} c_{i,k-1} q_{k-1} \right] = \sum_{i=1}^N w_{i,k} \left[\frac{x_i(t_k) - x_i(t_{k-1})}{q_{k-1}} + \frac{1}{2} c_{i,k-1} q_{k-1} \right]$
- $c_k = \sum_{i=1}^N w_{i,k} c_{i,k-1} = \sum_{i=1}^N w_{i,k} \left[\frac{x_i(t_{k-2}) - 2x_i(t_{k-1}) + x_i(t_k)}{q_{k-1}^2} \right]$

Actually, the calculation of term c_k for interval I_k as predicted from theory, gave poor results. It was decided therefore that the drift should not be computed with respect to EAL, but with respect to another precise time scale, TT(BIPM).
By calling $y_{TT,h_i}(t)$ the difference at time t between the measurement of TT and the measurement of clock h_i, $c_{i,k-1}$ is estimated by taking 3 intervals $I_{k-1}, I_{k-2}, I_{k-3}$, and calculating for $j = k-3, k-2, k-1$ and $s = 0, \ldots$,5:

$$\frac{y_{TT,h_i}(t_{j_{s+1}}) - y_{TT,h_i}(t_{j_s})}{t_{j_{s+1}} - t_{j_s}} \tag{1.53}$$

Afterwards it is done a least squares interpolation of first order degree.
Once that the terms $a_{i,k-1}, b_{i,k-1}$ and $c_{i,k-1}$ are predicted for each clock, we can define

$$h'_i(t) = a_{i,k-1} + \left[b_{i,k-1} + \frac{1}{2}c_{i,k-1}(t_k - t_{k-1})\right](t - t_k) + \frac{1}{2}c_{i,k-1}(t - t_k)^2 \quad t \in I_k, i \in \{1, \dots, N\} \tag{1.54}$$

which is the prediction term for each clock. EAL at time t in I_k as:

$$EAL(t) = \sum_{i=1}^{n} w_{i,k}\left[h_i(t) + h'_i(t)\right] \tag{1.55}$$

Now, the BIPM is actually interested to report, for discrete values of t, the quantities $x_i(t) = EAL(t) - h_i(t)$ for $i \in \{1, \dots, N\}$. As pointed out before, it receives as input values $x_{i,j}(t) = h_i(t) - h_j(t)$. From this last values it can derive $N - 1$ linearly independent equations $x_i(t) - x_j(t) = x_{i,j}(t)$, so that one additional equation is required to find $x_i(t)$ for $i \in \{1, \dots, N\}$. By subtracting $\sum_{i=1}^{N} w_{i,k}h_i(t)$ to both sides of 1.55 we obtain the additional equation:

$$\sum_{i=1}^{N} w_{i,k}x_i(t) = \sum_{i=1}^{N} w_{i,k}h'_i(t) \tag{1.56}$$

In the end the system to be solved is:

$$\begin{cases} x_i(t) - x_j(t) = x_{i,j}(t) \\ \sum_{i=1}^{N} w_{i,k}x_i(t) = \sum_{i=1}^{N} w_{i,k}h'_i(t) \end{cases} \tag{1.57}$$

with solution:

$$x_j(t) = EAL(t) - h_j(t) = \sum_{i=1}^{N} w_{i,k}\left[h'_i(t) - x_{i,j}(t)\right] \quad \forall j \in \{1, \dots, N\} \tag{1.58}$$

It is then possible to compute the residuals:

$$r_j(t) = x_j(t) - h'_j(t) \quad \forall j \in \{1, \dots, N\} \tag{1.59}$$

We summarize here the procedure for the monthly computation of $x_i(t)$ at the BIPM:

- Each month, the BIPM receives data, in the form of reading differences, of all the clocks participating in the calculation of UTC. It takes the parameters estimated the preceding month ($a_{i,k-1}, b_{i,k-1}$ and $c_{i,k-1}$) and computes the prediction term for the current month.

- they solve system 1.57 and obtain the desired values.

- they compute the parameters $a_{i,k}, b_{i,k}$ and $c_{i,k}$ for the following month.

To be more precise, given that there are L laboratories, and each laboratory l contains N_l clocks, which we can denote as $h_{l,j}$ for $j \in \{1, \ldots, N_l\}$, the data arriving at the BIPM each month is :

- $UTC(l) - h_{l,j} \qquad \forall j \in \{1, \ldots, N_l\}$
- $UTC(s) - UTC(v) \qquad \forall s, v \in 1, \ldots, L$

Let us make an example.
Suppose that there are 2 laboratories, laboratory A and laboratory B. Laboratory A has got 2 clocks: clock 1 and clock 2. Laboratory B has got two clocks as well: clock 3 and clock 4. Suppose the BIPM has got to publish the Circular T related to time interval $I_2 = (t_2, t_3)$. For each $t = t_2 + k * 5$ with $k = 0, \ldots, 6$, it receives the following input vector:

$$b(t) = \begin{bmatrix} UTC(A)(t) - h_1(t) \\ UTC(A)(t) - h_2(t) \\ UTC(B)(t) - h_3(t) \\ UTC(B)(t) - h_4(t) \\ UTC(B)(t) - UTC(A)(t) \end{bmatrix}$$

and the objective is to obtain, for each $t = t_2 + k * 5$ with $k = 0, \ldots, 6$, the vector:

$$x(t) = \begin{bmatrix} EAL(t) - UTC(A)(t) \\ EAL(t) - UTC(B)(t) \\ EAL(t) - h_1(t) \\ EAL(t) - h_2(t) \\ EAL(t) - h_3(t) \\ EAL(t) - h_4(t) \end{bmatrix}$$

To find $x(t)$ this system is solved:

$$\begin{bmatrix} -1 & 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & w_{1,2} & w_{2,2} & w_{3,2} & w_{4,2} \end{bmatrix} \begin{bmatrix} EAL(t) - UTC(A)(t) \\ EAL(t) - UTC(B)(t) \\ EAL(t) - h_1(t) \\ EAL(t) - h_2(t) \\ EAL(t) - h_3(t) \\ EAL(t) - h_4(t) \end{bmatrix} = \begin{bmatrix} UTC(A)(t) - h_1(t) \\ UTC(A)(t) - h_2(t) \\ UTC(B)(t) - h_3(t) \\ UTC(B)(t) - h_4(t) \\ UTC(B)(t) - UTC(A)(t) \\ \sum_{i=1}^{4} w_{i,2} \left[a_{i,1} + \left[b_{i,1} + \frac{1}{2}c_{i,1}(t_2 - t_1)\right](t - t_2) + \frac{1}{2}c_{i,1}(t - t_2)^2\right] \end{bmatrix}$$

with

$$a_{i,1} = EAL(t_2) - h_i(t_2) \qquad \forall i \in \{1, \ldots, 4\}$$

$$b_{i,1} = \frac{[EAL(t_2) - h_i(t_2)] - [EAL(t_1) - h_i(t_1)]}{t_2 - t_1} \qquad \forall i \in \{1, \ldots, 4\}$$

calculated at the end of I_1, and $c_{i,1}$ calculated using frequency data from three months preceding. In the chapter we did not mention laboratories and we simplified the algorithm description giving only the definition of differences among clocks, nonetheless it is straightforward to obtain reading differences among all clocks in the ensemble from data above.

1.2.3 Calculation of weights

The main assumption behind the calculation of weights is that a good clock is a stable/predictable clock.
For this reason clock weights are reciprocal of a statistical quantity which characterizes their frequency stability.
For example H masers are characterized by a significant but very well predictable drift, which means that they should still contribute to the time scale with a significant weight without degrading the long term stability of EAL.
In order to avoid a small number of very stable clocks to dominate the scale, an upper limit on weight was set. From 1st November 2022 the maximum weight is set to $w_{max} = \frac{6}{N}$ where N is the total number of clocks. In the current procedure weights are computed in the following four-iteration process:

- Suppose we have initial relative weights that allow us to compute the values $EAL - h_i$, for all $i \in \{1, \dots, N\}$ and for interval I_k, as outlined in the previous section. In the following iterations, they are those obtained from the previous iteration.
- For interval I_k the value $\epsilon_{i,k} = |b_{i,k} - b_{i,k-1}|$ is computed, where $b_{i,k}$ is the real frequency on the interval and $b_{i,k-1}$ is the predicted one from previous month.
- The square of $\epsilon_{i,k}$ is evaluated for each clock.
- One year of $\epsilon_{i,k}$ is considered, which means taking the values of $\epsilon_{i,k}$ for the last computation interval I_k and those of the previous 11 months.
- A filter is used to give a predominant role to more recent measurements:

$$\sigma_i{}^2 = \frac{\sum_{j=1}^{M} \frac{M+1-j}{M} \epsilon_{i,j}{}^2}{\sum_{j=1}^{M} \frac{M+1-j}{M}} \tag{1.60}$$

  where $M = 12$ since we are considering one year.
- The relative weight of clock i is computed as the normalized inverse of the variance value σ_i^2

$$w_{i,temp} = \frac{\frac{1}{\sigma_i{}^2}}{\sum_{i=1}^{n} \frac{1}{\sigma_i{}^2}} \tag{1.61}$$

The new weight w_i of clock i is equal to $w_{i,temp}$ except for two cases:

1. The weight is bigger than the upper limit w_{max}, in which case: $w_{i,temp} = w_{max}$.
2. It is decided to exclude the clock from the ensemble because the difference between the real frequency and the predicted one is greater than a fixed threshold for the interval of calculation.

Chapter 2

Stochastic Models for Clock Noise

In the first chapter we talked about how the noise affecting atomic clocks (normalized frequency deviation with respect to the nominal frequency and related phase deviation) can be partially modeled, at least for what concerns its deterministic part, and how this is done in practice at the BIPM.
In this chapter we will focus on recognizing the random noise components affecting both frequency and phase deviations. This permits to model the clock noise as a bi-dimensional stochastic process and to compute the fundamental statistical quantities of this process. The study of clock noises is fundamental for estimation of optimal parameters.

2.1 Characterization of noise in frequency and time domains

The content of this section is mainly based on [2].
First thing to be said is that clocks can be affected by 5 types of different random noises:

- WPM: white phase modulation i.e. white noise on the phase component.
- FPM: flicker phase modulation i.e. flicker noise on the phase component.
- WFM: white frequency modulation i.e. white noise on the frequency component, inducing a random walk (or Brownian Motion) on the phase component.
- FFM: flicker frequency modulation i.e. flicker noise on the frequency component.

- $RWFM$: random walk frequency modulation i.e. random walk on the frequency.

For example a cesium clock is typically affected by WFM and $RWFM$.

We already defined in 1.1 the frequency noise given by the difference between the instantaneous frequency $\nu(t)$ (which cannot be measured) and the nominal frequency ν_0. We can call this quantity $\Delta\nu(t)$. $\Delta\nu(t)$ is a stochastic process and we suppose, for the sake of simpler modeling, that it has got zero mean and that it is stationary. The stationarity hypothesis will allow us to apply the Ergodic Theorem and use indifferently infinite time averages and statistical averages.
Now we would like to characterize the frequency noise, both in the time domain and in the frequency domain.
We have to be careful because we are now calling “ frequency” two different things: the time-dependent instantaneous frequency of an oscillator, and the time-indipendent Fourier frequency.
In the frequency domain it is common to use the spectral density, which is defined as the Fourier transform of the autocorrelation function.
If this is the definition of the autocorrelation function:

$$R_{\Delta\nu}(\tau) = \langle \Delta\nu(t)\Delta\nu(t-\tau)\rangle \tag{2.1}$$

where $\langle ... \rangle$ denotes an infinite time average, then the two-sided spectral density is:

$$S^{TS}_{\Delta\nu}(f) = \int_{-\infty}^{+\infty} R_{\Delta\nu}(\tau) exp(-i2\pi f\tau) d\tau \tag{2.2}$$

but it is more common to use the one sided spectral density:

$$S_{\Delta\nu}(f) = 2S^{TS}_{\Delta\nu}(f) \qquad 0 \leq f \leq +\infty \tag{2.3}$$

It is clear that correlation functions and spectral densities carry exactly the same information about the random process.
We actually prefer to work with the stochastic process of normalized frequency noise

$$y(t) = \frac{\Delta\nu(t)}{\nu_0} \tag{2.4}$$

, for which it is now easy to define:

$$S_y(f) = \frac{1}{\nu_0^2} S_{\Delta\nu}(f) \tag{2.5}$$

From spectral density measurements done in various laboratories, with different types of oscillators, it appeared that experimental results could be modeled by power law curves. For a given clock, the following rule was derived:

$$S_y(f) = \begin{cases} \sum_{\alpha=-2}^{2} h_\alpha f^\alpha & 0 \le f \le f_h \\ 0 & f > f_h \end{cases} \tag{2.6}$$

where f_h is an upper cutoff frequency which reflects real world limitations, such as finite bandwidth of measurement instruments. It is the value of α which indicates the type of noise affecting the clock in a particular frequency interval, as we can see from Table 2.1. For example in the case of white frequency noise the spectral density is constant, while in the case of random walk noise it varies as f^{-1}.

Given that clocks are used for time keeping, it made sense to have also a characterization of frequency instabilities in the time domain, or in other words, to be able to give a measure of instability over an interval of length τ, as a function of τ. For this reason we define the random variable $\bar{Y}_k^\tau$, which depends on the underlying stochastic process $y(t)$:

$$\bar{Y}_k^\tau = \frac{1}{\tau}\int_{t_k}^{t_{k+1}} y(t)dt = \frac{x(t_{k+1}) - x(t_k)}{\tau} \tag{2.7}$$

The true variance of $\bar{Y}_k^\tau$, assuming that the process $y(t)$ has got zero mean, is:

$$\sigma^2(\tau) = VAR\left[\bar{Y}_k^\tau\right] = \mathbb{E}(\bar{Y}_k^\tau)^2 = \langle(\bar{y}_k^\tau)^2\rangle \tag{2.8}$$

where $\langle ... \rangle$ denote an infinite time average made over one sample of $y(t)$ or, since ergodicity is assumed, a spatial average obtained by taking an infinite number of samples at a given instant t_k.
We report now the derivation of the relationship between $\sigma^2(\tau)$ and the Fourier spectral density of the process $y(t)$ as found in [2]:

$$\sigma^2(\tau) = \langle\Big(\frac{1}{\tau}\int_{t_k}^{t_{k+1}} y(t)dt\Big)^2\rangle \tag{2.9}$$

which can be rewritten as a convolution:

$$\sigma^2(\tau) = \langle\Big(\int_{-\infty}^{+\infty} y(t)h_I(t_k - t)dt\Big)^2\rangle \tag{2.10}$$

where:

$$h_I(t - t_k) = \begin{cases} 0 & t < -\tau \\ \frac{1}{\tau} & -\tau \le t \le 0 \\ 0 & t > 0 \end{cases} \tag{2.11}$$

The Fourier transform of $h_I(t)$ is given by:

$$H_I(f) = -\frac{sin(\pi\tau f)}{\pi\tau f} \tag{2.12}$$

from which we obtain, using isometric properties:

$$\sigma^2(\tau) = \int_0^{+\infty} S_y(f) \left(\frac{sin(\pi\tau f)}{\pi\tau f} \right)^2 df \tag{2.13}$$

To give an estimate of the real variance, the first intuition would be to use:

$$\sigma^2_{(1)}(N,\tau) = \frac{1}{N}\sum_{i=1}^{N}\left(\bar{y}_i - \frac{1}{N}\sum_{j=1}^{N}\bar{y}_j\right)^2 \tag{2.14}$$

where N is the number of samples, $\bar{y}_i$ for $i = 1, \dots, N$ is a sample for the interval $[t_i, t_i + \tau]$ and we assume there is no dead time between measurements. Nonetheless this sample variance turns out to be biased for all finite N. In particular ([6]):

$$\langle\sigma^2_{(1)}(N,\tau)\rangle = \sigma^2(\tau) - \sigma^2(N\tau) \tag{2.15}$$

In the special case of white noise, if we consider 2.6 and table 2.1, equation 2.13 gives:

$$\sigma^2(\tau) = \frac{h_0}{2\tau} \tag{2.16}$$

which implies

$$\langle\sigma^2_{(1)}(N,\tau)\rangle = \left(1 - \frac{1}{N}\right)\sigma^2(\tau) \tag{2.17}$$

By looking at 2.17 we infere that, at least for the case of white noise:

$$\sigma^2(N,\tau) = \frac{1}{N-1}\sum_{i=1}^{N}\left(\bar{y}_i - \frac{1}{N}\sum_{j=1}^{N}\bar{y}_j\right)^2 \tag{2.18}$$

is an unbiased estimator of variance. Since the estimator is unbiased:

$$\langle\sigma^2(N,\tau)\rangle = VAR\left[\bar{Y}_k^\tau\right] \tag{2.19}$$

From this sample variance comes the definition of the so called Allan Variance as $\langle\sigma^2(N,\tau)\rangle$ with $N = 2$. The Allan Variance is indeed defined as:

$$\sigma^2_y(\tau) = \frac{1}{2}\langle\left(\bar{y}^\tau_{k+1} - \bar{y}^\tau_k\right)^2\rangle = \frac{1}{2}\mathbf{E}\left[(\bar{Y}^\tau_{k+1} - \bar{Y}^\tau_k)^2\right] \tag{2.20}$$

If we have M frequency values a possible estimator for the Allan Variance is:

$$\hat{\sigma}^2_y(\tau) = \frac{1}{2(M-1)}\sum_{k=1}^{M-1}(\bar{y}_{k+1} - \bar{y}_k)^2 \tag{2.21}$$

Typically it has been proved that M=100 is sufficient for the estimator to converge, though of course the confidence of the estimate will typically improve as the data length increases.
We call τ_0 the minimal data spacing for the original stored dataset, for atomic clocks contributing to UTC calculation $\tau_0 = 432000$, which is the number of seconds in 5 days.
It is very useful to compute $\hat{\sigma}_y^2(\tau)$ for $\tau = n * \tau_0$ for some n positive integer. This is done by averaging n adjacent values of $\bar{y}_k^{\tau_0}$. Supposing that $k = 1, \dots, M$ and $\tau = n * \tau_0$:

$$\hat{\sigma}_y^2(\tau) = \frac{1}{(M-2n+1)} \sum_{k=1}^{M-2n+1} (\bar{y}_{k+n}^{\tau} - \bar{y}_k^{\tau})^2 \tag{2.22}$$

where

$$\bar{y}_k^{\tau} = \frac{1}{n} \sum_{i=k}^{k+n-1} \bar{y}_i^{\tau_0} = \frac{x_{k+n} - x_k}{\tau} \tag{2.23}$$

Alternately, one may write:

$$\hat{\sigma}_y^2(\tau) = \frac{1}{2\tau^2(M-2n+1)} \sum_{i=1}^{M-2n+1} (x_{i+2n} - 2x_{i+n} + x_i)^2 \tag{2.24}$$

As it happened for the Fourier spectral density, the following proportionality appears:

$$\sigma_y^2(\tau) \sim \tau^{\mu} \tag{2.25}$$

A relationship exists between α and μ:
$\mu = -\alpha - 1$ if $3 < \alpha \leq 1$ and $\mu \sim -2$ if $\alpha \geq 1$.

Depending on the value of either of these constants one can distinguish the type of random noise affecting the fractional frequency or the phase fluctuations of a clock in a particular interval, as summarized in table 2.1 and plot 2.1.

Noise Type	α	μ
White Phase	2	-2
Flicker Phase	1	-2
White Frequency	0	-1
Flicker Frequency	-1	0
Random Walk Frequency	-2	+1

Table 2.1: Power law exponents for clock noises.

More generally there exists a relationship between Fourier Spectral Density and Allan Variance, given by the following equation:

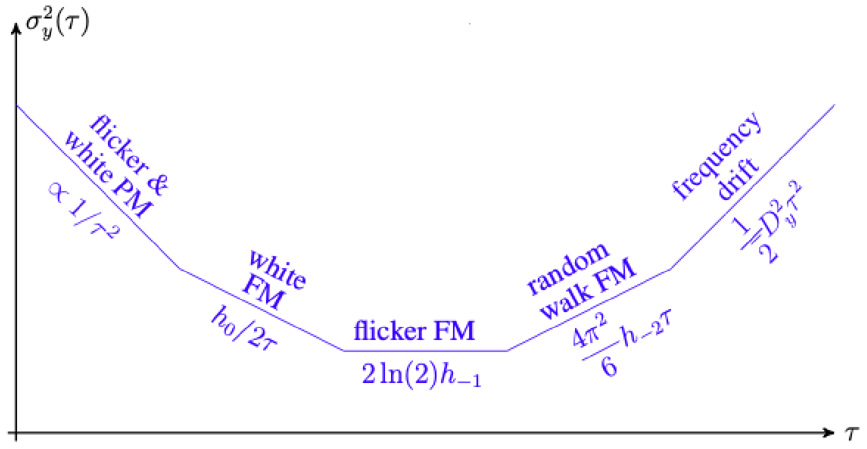


Figure 2.1: Power law curves of Allan variance to distinguish noise type ([7]).

$$\sigma_y^2(\tau) = \int_0^{+\infty} S_y(f)\frac{2sin^4(\pi\tau f)}{(\pi\tau f)^2}df \tag{2.26}$$

Proof of this in Appendix A.

Identifying the stochastic error on frequency/phase deviations is important to give theoretical rationale behind the adoption of a particular frequency estimator. For example the last - first method, which we derived from first order continuity condition, by approximating the derivative with finite forward differences, is in fact the optimum if the noise is pure white FM.

2.2 Stochastic model for the atomic clock error

Once the types of random noises affecting $x(t)$ and $y(t)$ are identified, by looking either at the Spectral density in the frequency domain or at the Allan Variance in the time domain, a stochastic differential equation can be written with stochastic state variables corresponding to the normalized phase deviation and the normalized frequency deviation.
In order to be able to solve a SDE the theory of Stochastic Calculus and Itô integral is needed. The most important results, which will prove useful in the solution, are reported in the following. They were taken from [8].
First of all a Wiener Process/Brownian Motion $W = (W_t, t \in [0, +\infty])$ is defined as a stochastic process with the following properties:

- $W_0 = 0$.

- The process increments are stationary and independent.
- Each W_t has a continuous sample path and $W_t \sim N(0,t), \quad \forall t > 0$. The sample paths are not differentiable because of unbounded variation.

In alternative a Wiener Process can be defined as a Gaussian process with mean $\mu_W(t) = 0$ and covariance function $\mathrm{E}[W_t W_s] = c_W(t,s) = min(s,t)$.
An important property of Brownian Motion is that it is a Martingale with respect to the natural filtration $\mathcal{F}_t = \sigma(W_s, s \leq t)$.
For this chapter it will prove useful to discuss the computation of stochastic integrals with respect to Brownian sample paths.
Under certain conditions on function f it it possible to compute $\int_a^b f(t)dW_t(w)$ for every Brownian sample path $W_t(w)$ as a Riemann-Stieltjes integral. It is not possible, due to the unbounded variation of the Brownian motion, to compute instead $\int_a^b W_t(w)dW_t(w)$ as a Riemann-Stieltejs integral.
Anyway, by writing the Riemann-Steltjes sum considering partition $\tau_n : 0 = t_0 < t_1 < \cdots < t_{n-1} < t_n = t$ and intermediate partition $(y_i) = t_{i-1}$, with $\Delta_i W = W_{t_i} - W_{t_{i-1}}$:

$$S_n = \sum_{i=1}^{n} W_{t_{i-1}} \Delta_i W \tag{2.27}$$

it is possible to demonstrate that it converges in the mean square sense to $\frac{1}{2}(W_t^2 - t)$.
It is thus defined:

$$\int_0^t W_s dW_s = \frac{1}{2}(W_t^2 - t) \tag{2.28}$$

which is a first important example of Itô integral.
The general Itô integral $\int_0^t C_s dW_s$, with $C = (C_t, t \in [0,T])$, is defined for integrand process C which satisfies:

- C is adapted to Brownian motion on [0,T], i.e. C_t is a function of $B_s, s \leq t$.
- The integral $\int_0^T \mathrm{E}C_s^2 ds$ is finite.

Three important properties of the Itô Stochastic integral $\int_0^t C_s dW_s$ are:

- The Itô Stochastic integral is a martingale with respect to the natural Brownian filtration.
- The Itô Stochastic integral has got expectation 0.
- It satisfies the isometry property:

$$\mathrm{E}(\int_0^t C_s dW_s) = \int_0^t \mathrm{E}C_s^2 ds, \quad t \in [0,T] \tag{2.29}$$

To solve stochastic integrals we need some kind of tool, and this tool is Itô Lemma. The Itô Lemma is the stochastic analogue of the classical chain rule of differentiation. There are four different versions of it. They are all derived using a Taylor series expansion argument which makes use of the fact that $\Delta_i W$ on the interval $[t_{i-1}, t_i]$ satisfies $\mathrm{E}\Delta_i W = 0$ and $\mathrm{E}(\Delta_i W)^2 = \Delta_i$. This heuristically means that $(\Delta_i W)^2$ is of the order of Δ_i and in terms of differentials $(dW_t)^2 = dt$.
We report hereby the four versions of Itô Lemma.

VERSION I: $f(W_x)$

$$\int_s^t df(W_x) = f(W_t) - f(W_s) = \int_s^t f'(W_x)dW_x + \frac{1}{2}\int_s^t f''(W_x)dx \qquad (2.30)$$

VERSION II: $f(x, W_x)$

$$f(t, W_t) - f(s, W_s) = \int_s^t [f_1(x, W_x) + \frac{1}{2} f_{22}(x, W_x)]dx + \int_s^t f_2(x, W_x)dW_x \quad (2.31)$$

where we use the following notation:

$$f_i(t, x) = \frac{\partial}{\partial x_i} f(x_1, x_2)|_{x_1 = t, x_2 = x}, \qquad i = 1{,}2, \qquad (2.32)$$

$$f_{i,j}(t, x) = \frac{\partial}{\partial x_i}\frac{\partial}{\partial x_j} f(x_1, x_2)_{x_1 = t, x_2 = x}, \qquad i, j = 1{,}2 \qquad (2.33)$$

From now on we will consider X process defined as:

$$X_t = X_0 + \int_0^t A_s^{(1)} ds + \int_0^t A_s^{(2)} dW_s \qquad (2.34)$$

where $A^{(1)}$ and $A^{(2)}$ are adapted to Brownian Motion.

VERSION III: $f(y, X_y)$

$$\begin{aligned} f(t, X_t) - f(s, X_s) = &\int_s^t [f_1(y, X_y)dW_y + A_y^{(1)} f_2(y, X_y) + \\ &\frac{1}{2}[A_y^{(2)}]^2 f_{22}(y, X_y)]dy + \int_s^t A_y^{(2)} f_2(y, X_y)dW_y \end{aligned}$$

Finally, given two stochastic processes $X^{(1)}$ and $X^{(2)}$ with respect to the same Brownian motion:

$$X_t^{(i)} = X_0^{(i)} + \int_0^t A_s^{(1,i)} ds + \int_0^t A_s^{(2,i)} dW_s, \quad i = 1{,}2 \qquad (2.35)$$

VERSION IV: $f(y, X_y^{(1)}, X_y^{(2)})$

$$f(t, X_t^{(1)}, X_t^{(2)}) - f(s, X_s^{(1)}, X_s^{(2)}) = \int_s^t [f_1(y, X_y^{(1)}, X_y^{(1)})dy+ \tag{2.36}$$

$$+ \sum_{i=2}^{3} \int_s^t f_i(y, X_y^{(1)}, X_y^{(2)}) X_y^{(i)} \tag{2.37}$$

$$+ \frac{1}{2} \sum_{i=2}^{3} \sum_{j=2}^{3} \int_s^t f_{i,j}(y, X_y^{(1)}, X_y^{(2)}) A_y^{(2,i)} A_y^{(2,j)} dy \tag{2.38}$$

Now we would like to discuss solutions of differential equations of the type:

$$dX_t = a(t, X_t)dt + b(t, X_t)dW_t, \qquad X_0(w) = Y(w). \tag{2.39}$$

where $a(t, x)$ and $b(t, x)$ are called coefficient functions and are deterministic functions. The randomness of $X = (X_t, t \in [0, t])$ results, on the one hand, from the initial condition, and on the other end, from the noise generated by Brownian motion. Equation 2.39 can be rewritten as:

$$X_t = X_0 + \int_0^t a(s, X_s)ds + \int_0^t b(s, X_s)dW_s \tag{2.40}$$

which is what is called a Itô stochastic differential equation. There are both strong and weak solutions to differential equations.
A strong solution $X = (X_t, t \geq 0)$ satisfies:

- X is adapted to Brownian motion, i.e. at time t it is a function of $W_s, s \leq t$.
- The integrals occurring in 2.40 are well defined as Riemann or Itô Stochastic integrals, respectively.
- X is a function of the underlying Brownian sample path and of the coefficient functions $a(t, x)$ and $b(t, x)$.

For weak solutions instead, the path behaviour is not essential and we are only interested in the distribution of X. A strong or weak solution X of the Itô stochastic integral is called a diffusion process. If the following conditions are satisfied:

- $X_0 : \mathrm{E}X_0^2 < \infty$ and is independent of $(W_t, t \geq 0)$.
- For all $t \in [0, T]$ and $x, y \in \mathbb{R}$ coefficients functions are continuous.

- For all $t \in [0,T]$ and $x, y \in \mathbb{R}$ coefficients functions satisfy a Lipschitz condition with respect to the second variable:

$$|a(t,x) - a(t,y)| + |b(t,x) - b(t,y)| \leq K|x-y| \tag{2.41}$$

then the Itô Stochastic differential equation has a unique strong solution X.
In particular we will consider the case of the General Linear Stochastic differential equation:

$$X_t = X_0 + \int_0^t [a_1(s) + a_2(s)X_s]ds + \int_0^t [b_1(s) + b_2(s)X_s]dW_s \tag{2.42}$$

which respects the conditions of existence of an unique solution. The solution for this SDE is:

$$X_t = Y_t(X_0 + \int_0^t [a_2(s) - b_1(s) * b_2(s)]Y_s^{(-1)}ds + \int_0^t b_2(s)Y_s^{(-1)}dW_s, \qquad t \in [0,T] \tag{2.43}$$

where Y is the solution of the homogeneous equation:

$$X_t = X_0 + \int_0^t a_2(s)X_s ds + \int_0^t b_2(s)X_s dW_s \tag{2.44}$$

2.2.1 Clock affected by WFM and RWFM

The content of this section is based on [9]. The objective of paper [9] was to compute the normalized phase and frequency deviation of an atomic clock as stochastic solutions of a dynamical system. To highlight the stochastic nature of these quantities $x(t)$ was renamed as $X_1(t)$, which implies $y(t) = \dot{X}_1(t)$. The following two-state model was considered:

$$\begin{cases} dX_1(t) = (X_2(t) + \mu_1)\,dt + \sigma_1 dW_1(t) \\ dX_2(t) = \mu_2 dt + \sigma_2 dW_2(t) \end{cases} \quad t \geq 0 \qquad \begin{cases} X_1(0) = c_1 \\ X_2(0) = c_2 \end{cases} \tag{2.45}$$

With the following notation:

- $X_2(t)$ is the random walk component of frequency deviation.
- W_1 is the Wiener noise acting on the phase, which is driven by a Wiener noise on the frequency. That is: a white frequency noise results in an integrated white noise on the phase X_1.
- W_2 is the so called RWFM.

- μ_1, μ_2 are the drifts (deterministic components) affecting respectively $X_1(t)$ and $X_2(t)$.
- σ_1, σ_2 are the diffusion coefficients (related to Allian variance).

Equation 2.45 can be written in a matrix form:

$$d\mathbf{X}(t) = (F\mathbf{X}(t) + \mathbf{M})dt + Qd\mathbf{W}(t), \qquad t \geq 0 \tag{2.46}$$

with:

$$F = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \; Q = \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix}, \; \mathbf{M} = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix},$$

$$d\mathbf{X}(t) = \begin{pmatrix} d\mathbf{X}_1(t) \\ d\mathbf{X}_2(t) \end{pmatrix}, \; d\mathbf{W}(t) = \begin{pmatrix} d\mathbf{W}_1(t) \\ d\mathbf{W}_2(t) \end{pmatrix}$$

Equation 2.46 is a linear SDE, hence it is possible to obtain its solution in a closed form.

The solution is:

$$\begin{cases} X_1(t) = X_1(0) + X_2(0)t + \mu_1 t + \mu_2 \frac{t^2}{2} + \sigma_1 W_1(t) + \sigma_2 \int_0^t W_2(s)ds \\ X_2(t) = X_2(0) + \mu_2 t + \sigma_2 W_2(t) \end{cases} \tag{2.47}$$

Where we used $\int_0^t dW(s) = W(t)$ which comes from the application of version I of Ito's lemma.

By using the fact that $X_1(0) = c_1$, $X_2(0) = c_2$ it is possible to rewrite 2.47 as:

$$\begin{cases} X_1(t) = c_1 + (c_2 + \mu_1)\, t + \mu_2 \frac{t^2}{2} + \sigma_1 W_1(t) + \sigma_2 \int_0^t W_2(s)ds \\ X_2(t) = c_2 + \mu_2 t + \sigma_2 W_2(t) \end{cases} \tag{2.48}$$

Now we can notice that the evolution of the process $X(t) = \begin{bmatrix} X_1(t) \\ X_2(t) \end{bmatrix}$ is driven by a stochastic innovation represented by the vector

$$G_t = \begin{bmatrix} \sigma_1 W_1(t) + \sigma_2 \int_0^t W_2(s)ds \\ \sigma_2 W_2(t) \end{bmatrix} \tag{2.49}$$

By using that:

- $W_1(t) \sim N(0,t) \longrightarrow E\left[(W_1(t))^2\right] = t$
- $W_2(t) \sim N(0,t) \longrightarrow E\left[(W_2(t))^2\right] = t$

- $\int_0^t W(s)ds \sim N(0, \frac{t^3}{3})$ (Calculations of this in A)

We get that:

$$E\left[G_t\right] = \mathbf{0} \tag{2.50}$$

$$\Sigma_t = E\left[G_t {G_t}^T\right] = \begin{bmatrix} \sigma_1^2 t + \sigma_2^2 \frac{t^3}{3} & \sigma_2^2 \frac{t^2}{2} \\ \sigma_2^2 \frac{t^2}{2} & \sigma_2^2 t \end{bmatrix} \tag{2.51}$$

In conclusion the process $X(t)$ is Gaussian with mean vector and covariance matrix given below:

$$E\left[X(t)\right] = \begin{bmatrix} c_1 + (c_2 + \mu_1)t + \mu_2 \frac{t^2}{2} \\ c_2 + \mu_2 t \end{bmatrix} \tag{2.52}$$

$$\Gamma_X(t) = \begin{bmatrix} \sigma_1^2 t + \sigma_2^2 \frac{t^3}{3} & \sigma_2^2 \frac{t^2}{2} \\ \sigma_2^2 \frac{t^2}{2} & \sigma_2^2 t \end{bmatrix} \tag{2.53}$$

For simulation purposes, in [9] the following structure was built:

- Fixed interval $[0, T]$
- Equally spaced partition $0 = t_0 < t_1 < \cdots < t_N = T$
- $\tau = t_{k+1} - t_k$

so that 2.47 could be rewritten as:

$$\begin{cases} X_1(t_{k+1}) = X_1(t_k) + (X_2(t_k) + \mu_1)\tau + \mu_2 \frac{\tau^2}{2} + J_{k,1} \\ X_2(t_{k+1}) = X_2(t_k) + \mu_2 \tau + J_{k,2} \end{cases} \tag{2.54}$$

where:

$$J_k = \begin{bmatrix} \sigma_1(W_1(t_{k+1}) - W_1(t_k)) + \sigma_2 \int_{t_k}^{t_{k+1}}((W_2(s) - W_2(t_k))ds \\ \sigma_2(W_2(t_{k+1}) - W_2(t_k)) \end{bmatrix} \tag{2.55}$$

$$\sim N\left(0, \begin{bmatrix} \sigma_1^2 \tau + \sigma_2^2 \frac{\tau^3}{3} & \sigma_2^2 \frac{\tau^2}{2} \\ \sigma_2^2 \frac{\tau^2}{2} & \sigma_2^2 \tau \end{bmatrix}\right) \tag{2.56}$$

The innovation of the process J_k, that is, the stochastic part that is added to build the process at time t_{k+1}, depends only on the increments of the Wiener process in the interval (t_k, t_{k+1}). The increments of the Wiener process are independent and identically distributed. The process J_k is a bivariate Gaussian random variable such that:

$$E\left[J_k, J_m\right] = 0 \qquad \forall k \neq m$$

These properties make it very easy to simulate J_k.

Chapter 3

Possible improvements: implementation and results.

3.1 Calculation of EAL with OLS method

From the statistical point of view there were some doubts on the rationale behind the definition of EAL as in 1.10. In particular it seems like this definition is in fact an estimator for EAL, but not EAL itself. For this reason we tried to obtain definition 1.10 as an optimal estimator for EAL given the following model:

$$h_i(t_{kj}) = EAL(t_{kj}) - a_{i,k} - b_{i,k}(t_{kj} - t_k) - \frac{1}{2}c_{i,k}(t_{kj} - t_k)^2 + \xi_i \tag{3.1}$$

where:

- $i = 1, \dots, N$
- $j = 1, \dots, 5$
- $\xi_i \sim \mathcal{N}(0, {\sigma_i}^2)$ i.i.d.

Practically each I_k is one month long, and every 5 days we get a measure of the reading of each clock and we want to estimate $EAL(t)$. We suppose that $a_{i,k}, b_{i,k}$ and $c_{i,k}$ stay unchanged in one month period. We want to give at the same time an estimate of: $EAL(t_{kj}), a_{i,k}, b_{i,k}, c_{i,k}$. In order to do that we look for the parameters that minimize the residual sum of squares RSS:

$$RSS = \sum_{i=1}^{N}\sum_{j=1}^{5}\left[h_i(t_{kj}) - EAL(t_{kj}) + a_{i,k} + b_{i,k}(t_{kj} - t_k) + \frac{1}{2}c_{i,k}(t_{kj} - t_k)^2\right]^2 \tag{3.2}$$

We compute derivatives with respect to all the variables and equal them to zero:

$$\frac{\partial RSS}{\partial EAL(t_{kj})} = -2\left[-NEAL(t_{kj}) - \sum_{i=1}^{N}\left(h_i(t_{kj}) + a_{i,k} + b_{i,k}(t_{kj} - t_k) + \frac{1}{2}c_{i,k}(t_{kj} - t_k)^2\right)\right] \tag{3.3}$$

Imposing

$$\frac{\partial RSS}{\partial EAL(t_{kj})} = 0 \tag{3.4}$$

leaves us with the following estimation of $EAL(t_{kj})$:

$$\hat{EAL}(t_{kj}) = \frac{1}{N}\left[\sum_{i=1}^{N}\left(h_i(t_{kj}) + \hat{a}_{i,k} + \hat{b}_{i,k}(t_{kj} - t_k) + \frac{1}{2}\hat{c}_{i,k}(t_{kj} - t_k)^2\right)\right] \forall k, j \tag{3.5}$$

which gives us intuition on why EAL was defined as in 1.10 in the first place.

For the other parameters we obtain:

$$\hat{a}_{i,k} = \frac{\sum_{j=1}^{5}\left(\hat{EAL}(t_{k_j}) - h_i(t_{k_j}) - \hat{b}_{i,k}(t_{k,j} - t_k) - \frac{1}{2}\hat{c}_{i,k}(t_{k_j} - t_k)^2\right)}{5} \qquad \forall i, k \tag{3.6}$$

$$\hat{b}_{i,k} = \frac{\sum_{j=1}^{5}(t_{k,j} - t_k)\left(\hat{EAL}(t_{k_j}) - h_i(t_{k_j}) - \hat{a}_{i,k} - \frac{1}{2}\hat{c}_{i,k}(t_{k_j} - t_k)^2\right)}{\sum_{j=1}^{5}(t_{k,j} - t_k)^2} \qquad \forall i, k \tag{3.7}$$

$$\hat{c}_{i,k} = \frac{\sum_{j=1}^{5}(t_{k,j} - t_k)^2\left(\hat{EAL}(t_{k_j}) - h_i(t_{k_j}) - \hat{a}_{i,k} - \hat{b}_{i,k}(t_{k_j} - t_k)\right)}{\sum_{j=1}^{5}(t_{k,j} - t_k)^3} \qquad \forall i, k \tag{3.8}$$

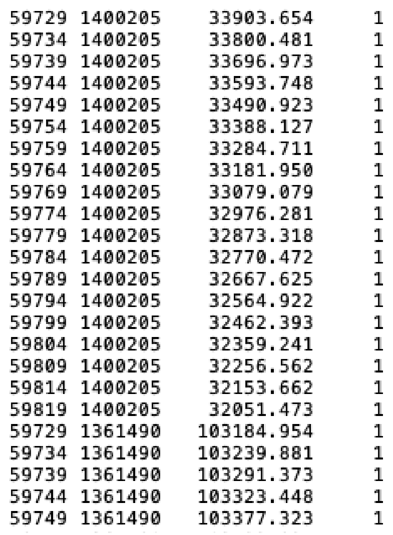
After discussion with my supervisor Gianna Panfilo, it was decided that these estimators are not compatible with the definition of a time scale. It is not admissible to optimize the prediction on each interval, without assuring the continuity conditions of the time scale. More generally, we do not want to make an estimate of the scale which is as close as possible to the available data each month, but we want instead to ensure consistency with the past. Further developments could include the imposition of continuity conditions within the OLS framework. Another thing to be noted is that, as far as it is known, there is no way to produce experimentally something which can be defined as the reading of clock i by itself, so that this model should be revised also to account for this aspect.

3.2 Implementation of deterministic models

At the $BIPM$ I was asked to evaluate the performance of the current algorithm using a large amount of past data, in order to obtain some insights on whether or not it could be improved.
In particular, since the algorithm was designed to well predict the drift of H-Masers clock and ageing of cesium clocks, there were some concerns that it could not be optimal in the case of fountains.
I started working with a data format called $TTHyymm$ where yy are two digits for the year and mm are two digits for the month. Each of these tables would contain one column of MJD values equally spaced by 5 days, one column with the clock code, and one column with the reading difference between the clock and the ultra precise TT time scale, for clock i this is what we called $y_{h_i,TT}$.
Clock codes instead are seven digit numbers that allow to recognize the clock type and laboratory. In particular

- Cesium clock codes always start with 13.
- Maser clock codes always start with 14.
- Fountains clock codes always start with 19.

There are other types of clocks with different codes, but they were not considered in this thesis work, because of poor quality.
Given that this data format is used to compute the drift, and each month we need three months of data (the current and the two preceding) for the calculation, each of the *TTHyymm* tables would contain not only the month indicated in the name of the table, *mm*, but also the two months before it.
In figure 3.1 I show a snapshot of *TTH2208* data.
As we can see the first value in the first column is 59729, which is MJD for 30 May 2022. Indeed, since the data is referred to August 2022, it should contain measurements from August, July and June 2022.
Also, in order to have a continuous prediction, the first value of each month is actually the last value from the month preceding. That is why first value for June 2022 is 30 May 2022. Taking two tables related to two subsequent months, there are two months of overlapping $MJDs$, but even though dates are the same, reading differences values can differ of few picoseconds, given that each month they are adjourned.



```
59729 1400205    33903.654     1
59734 1400205    33800.481     1
59739 1400205    33696.973     1
59744 1400205    33593.748     1
59749 1400205    33490.923     1
59754 1400205    33388.127     1
59759 1400205    33284.711     1
59764 1400205    33181.950     1
59769 1400205    33079.079     1
59774 1400205    32976.281     1
59779 1400205    32873.318     1
59784 1400205    32770.472     1
59789 1400205    32667.625     1
59794 1400205    32564.922     1
59799 1400205    32462.393     1
59804 1400205    32359.241     1
59809 1400205    32256.562     1
59814 1400205    32153.662     1
59819 1400205    32051.473     1
59729 1361490   103184.954     1
59734 1361490   103239.881     1
59739 1361490   103291.373     1
59744 1361490   103323.448     1
59749 1361490   103377.323     1
```

Figure 3.1: Example of TTH2208 data

I decided to work with 5 years of data, from January 2018 to December 2022, which means 60 batches of three months.
Using Matlab I merged all data in one table, and to overcome the problem of double measurements taken in the same date, I decided to only keep latest measurements, supposing that they are more precise then older ones.
I used Matlab to pivot the table in order to have on the rows the clock codes and on the columns the sequence of five years $MJDs$. This allowed me to select a subset of clocks with continuous measurements over the period 2018-2022 (by eliminating all clocks that had at least one NaN value in their row). After selecting the ensemble of "usable" clocks, I cleaned the table from all the remaining clocks.
All the code is found in B.1, B.2.
A snapshot of the cleaned table is found in figure 3.2.

In the end I selected 51 masers and 56 cesium coming from a total of different 24 laboratories:

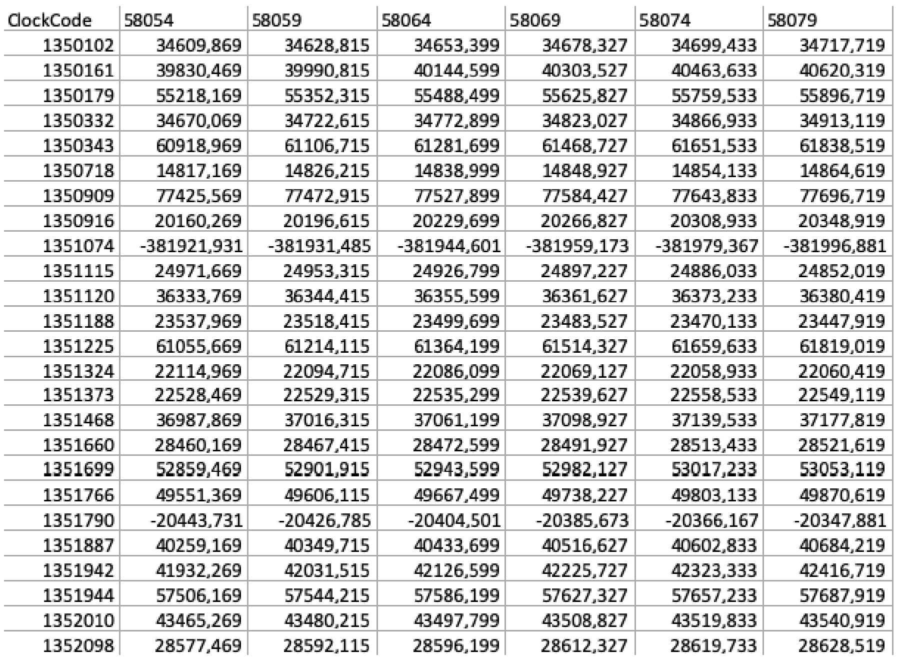
- Masers
  - NICT: 1402012
  - ROA: 1401436
  - OP: 1400810
  - NIST: 1400004, 1400205, 1400207, 1400212, 1400222, 1412015
  - PL: 1400814, 1404601
  - SP: 1407201, 1407221, 1407223
  - NPLI: 1405201

  - USNO: 1400702, 1400705, 1400708, 1400711, 1400712, 1400713, 1400718, 1400720, 1400722, 1400723, 1400724, 1400725, 1400726, 1400728, 1400729, 1400730, 1400731, 1400732, 1400736, 1400737, 1400740
  - NTSC: 1400296
  - NRC: 1400306
  - NIM: 1404832, 1404871, 1404878
  - TL: 1403011
  - BEV: 1403452
  - PTB: 1400506, 1400509
  - SU: 1403814, 1403845, 1403853
  - MIKE: 1404113, 1404180
  - NMIIJ: 1405012

- Cesium
  - ONRJ: 1350102, 1351942
  - USNO: 1350161
  - VSL: 1350179
  - NICT: 1350332, 1350343, 1350916, 1351225, 1351790, 1351887, 1351944, 1352010, 1352801, 1352903
  - ROA: 1350718, 1361488, 1361490, 1351699, 1361488, 1361490
  - OP: 1350909, 1352985
  - NIST: 1351074, 1352935
  - IT: 1351115, 1351373
  - PL: 1351120, 1351660
  - SP: 1351188, 1352166, 1362295
  - NPL: 1353167
  - NPLI: 1351324, 1353376
  - USNO: 1351468
  - SMD: 1351766, 1353564
  - NTSC: 1352098, 1352142, 1352962, 1352965, 1352980, 1353089, 1353091, 1353102, 1354936, 1354937
  - NRC: 1352150, 1352152
  - ESA: 1352353

  - NIM: 1352643, 1352769
  - IMBH: 1352909
  - TL: 1352910
  - BEV: 1353009
  - AUS: 1362269

- Fountains
  - USNO: 1930002, 1930003, 1930004, 1930005 (only fountains available for the period).



ClockCode	58054	58059	58064	58069	58074	58079
1350102	34609,869	34628,815	34653,399	34678,327	34699,433	34717,719
1350161	39830,469	39990,815	40144,599	40303,527	40463,633	40620,319
1350179	55218,169	55352,315	55488,499	55625,827	55759,533	55896,719
1350332	34670,069	34722,615	34772,899	34823,027	34866,933	34913,119
1350343	60918,969	61106,715	61281,699	61468,727	61651,533	61838,519
1350718	14817,169	14826,215	14838,999	14848,927	14854,133	14864,619
1350909	77425,569	77472,915	77527,899	77584,427	77643,833	77696,719
1350916	20160,269	20196,615	20229,699	20266,827	20308,933	20348,919
1351074	-381921,931	-381931,485	-381944,601	-381959,173	-381979,367	-381996,881
1351115	24971,669	24953,315	24926,799	24897,227	24886,033	24852,019
1351120	36333,769	36344,415	36355,599	36361,627	36373,233	36380,419
1351188	23537,969	23518,415	23499,699	23483,527	23470,133	23447,919
1351225	61055,669	61214,115	61364,199	61514,327	61659,633	61819,019
1351324	22114,969	22094,715	22086,099	22069,127	22058,933	22060,419
1351373	22528,469	22529,315	22535,299	22539,627	22558,533	22549,119
1351468	36987,869	37016,315	37061,199	37098,927	37139,533	37177,819
1351660	28460,169	28467,415	28472,599	28491,927	28513,433	28521,619
1351699	52859,469	52901,915	52943,599	52982,127	53017,233	53053,119
1351766	49551,369	49606,115	49667,499	49738,227	49803,133	49870,619
1351790	-20443,731	-20426,785	-20404,501	-20385,673	-20366,167	-20347,881
1351887	40259,169	40349,715	40433,699	40516,627	40602,833	40684,219
1351942	41932,269	42031,515	42126,599	42225,727	42323,333	42416,719
1351944	57506,169	57544,215	57586,199	57627,327	57657,233	57687,919
1352010	43465,269	43480,215	43497,799	43508,827	43519,833	43540,919
1352098	28577,469	28592,115	28596,199	28612,327	28619,733	28628,519

Figure 3.2: Snapshot of the pivoted table for data in the period Jan 2018 - Dec 2022. On the rows there are the clock codes and on the columns the sequence of MJD values. In each cell there is the reading difference measurement for that clock in that date.

First of all I looked at the frequency data, which I obtained making the first order difference of each row, and dividing all values by 5, which is the standard distance between two consecutive measurements.
By looking at the frequency data my aim was to recognize the drift which is corrected for in the algorithm.
The estimation of the drift, as reported in 1.53, was calculated using Matlab

function *polyfit*, which performs linear regression.
The frequency data and the interpolation were then plotted, along with the residuals and the qq-plot and histogram of residuals.
All the code is found in B.3 - B.5 .
Examples of interpolation for masers 1400004 and 1400702, cesium 1350102 and 1350179 and the four fountains are reported in figures 3.3, 3.4, 3.5, 3.6, 3.7, 3.8, 3.9, 3.10.

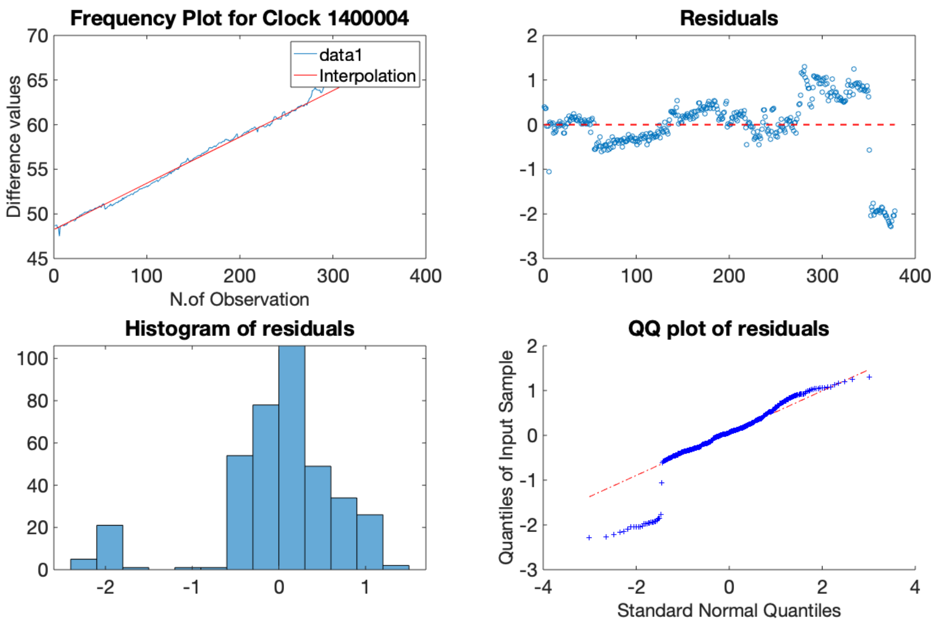


Figure 3.3: Drift and residual analysis for maser 1400004.

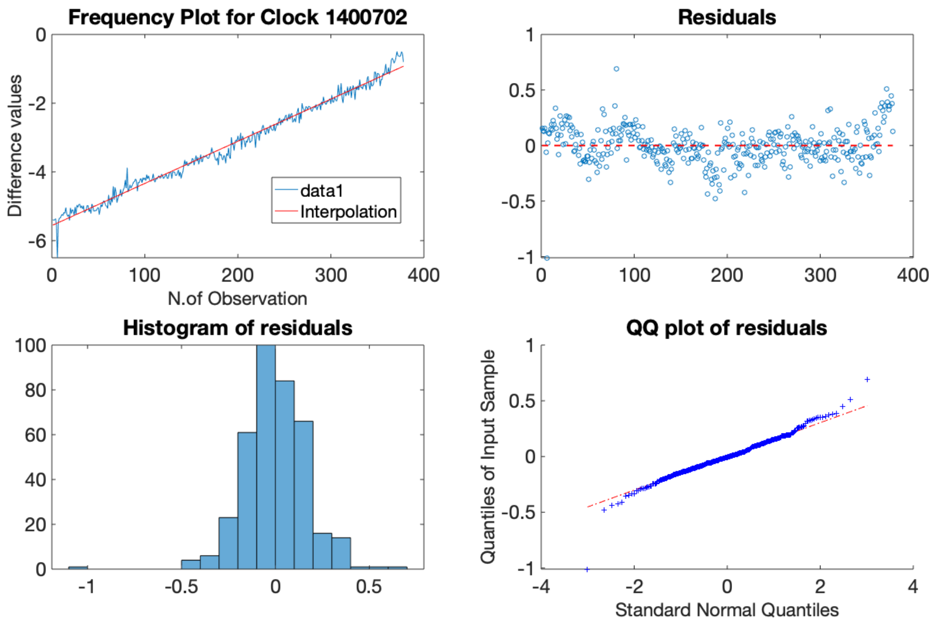


Figure 3.4: Drift and residual analysis for maser 1400702.

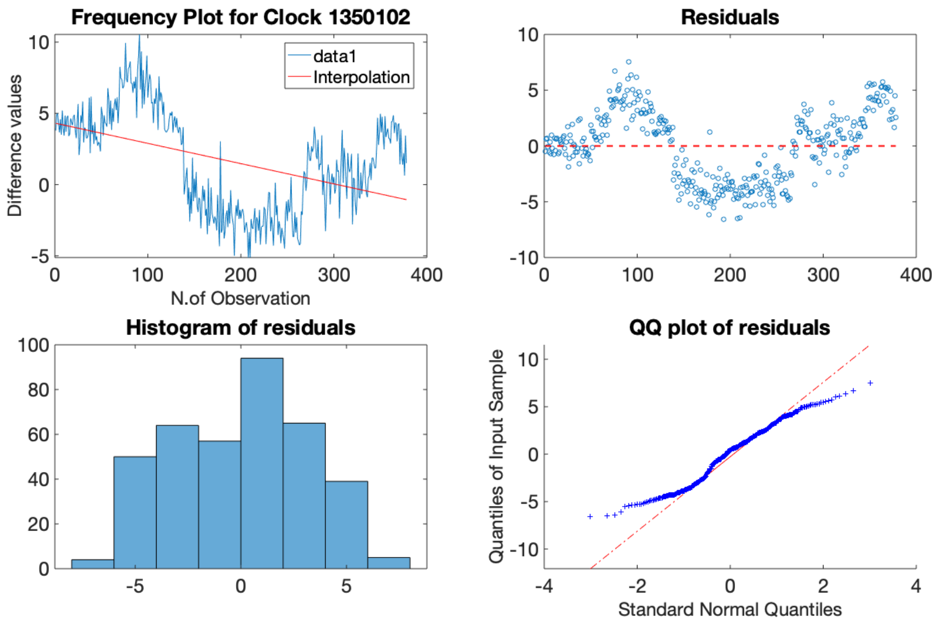


Figure 3.5: Drift and residual analysis for cesium 1350102.

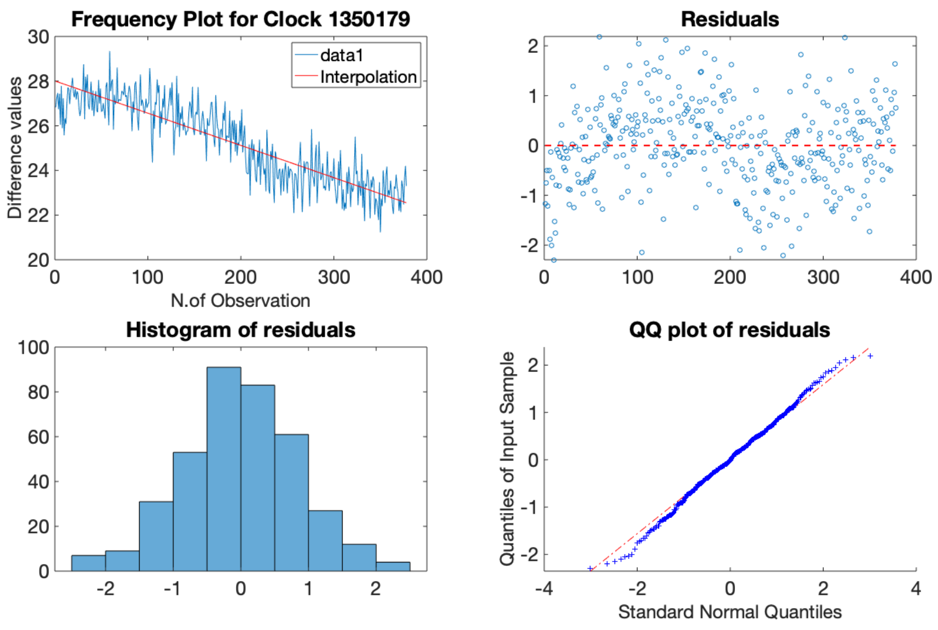


Figure 3.6: Drift and residual analysis for cesium 1350179.

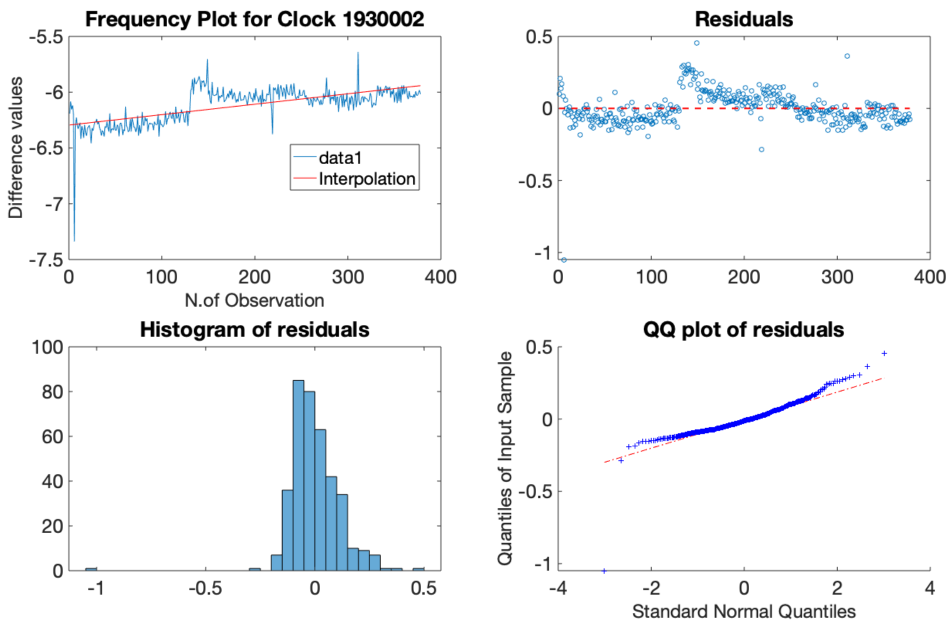


Figure 3.7: Drift and residual analysis for fountain 1930002.

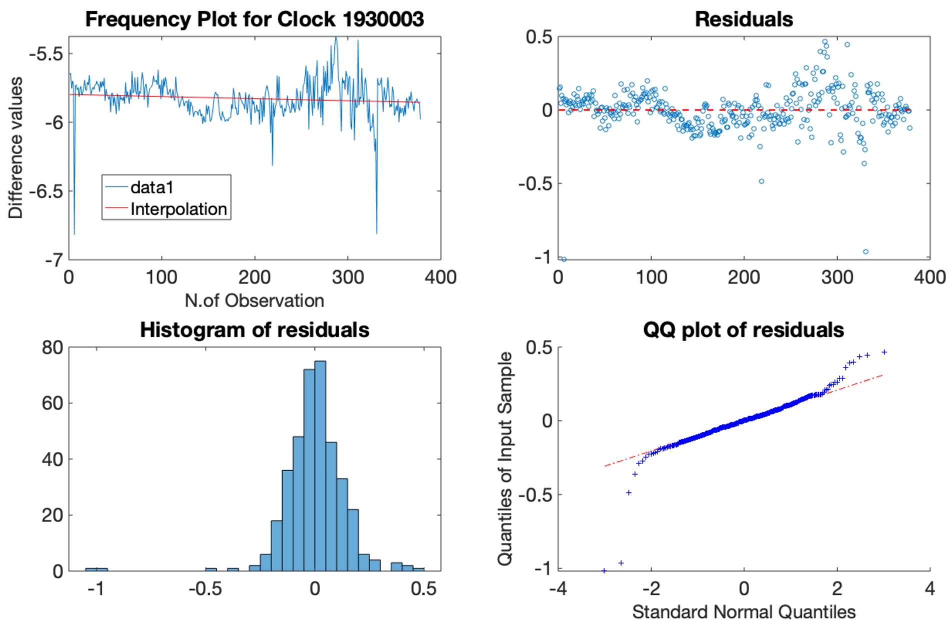


Figure 3.8: Drift and residual analysis for fountain 1930003.

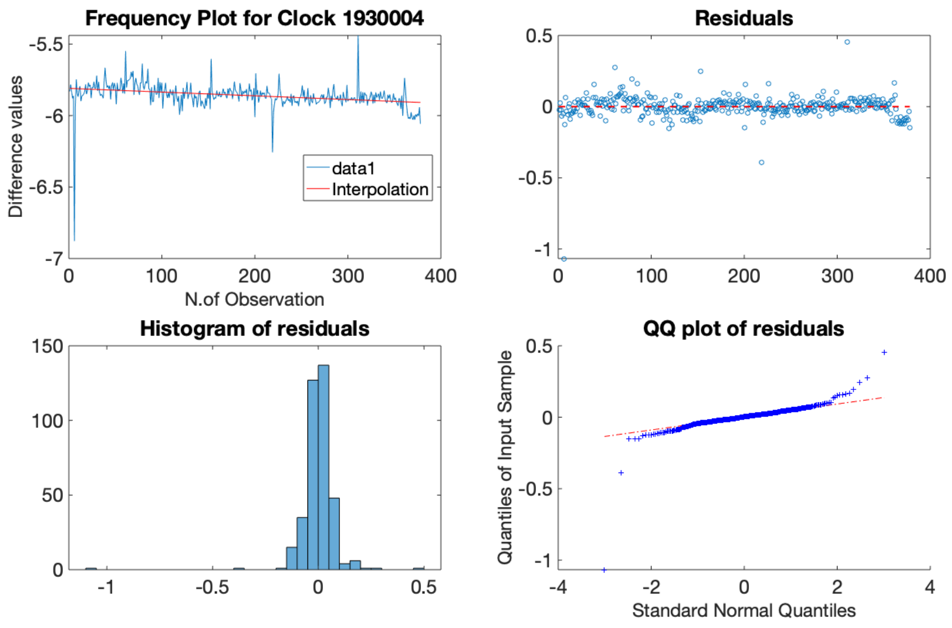


Figure 3.9: Drift and residual analysis for fountain 1930004.

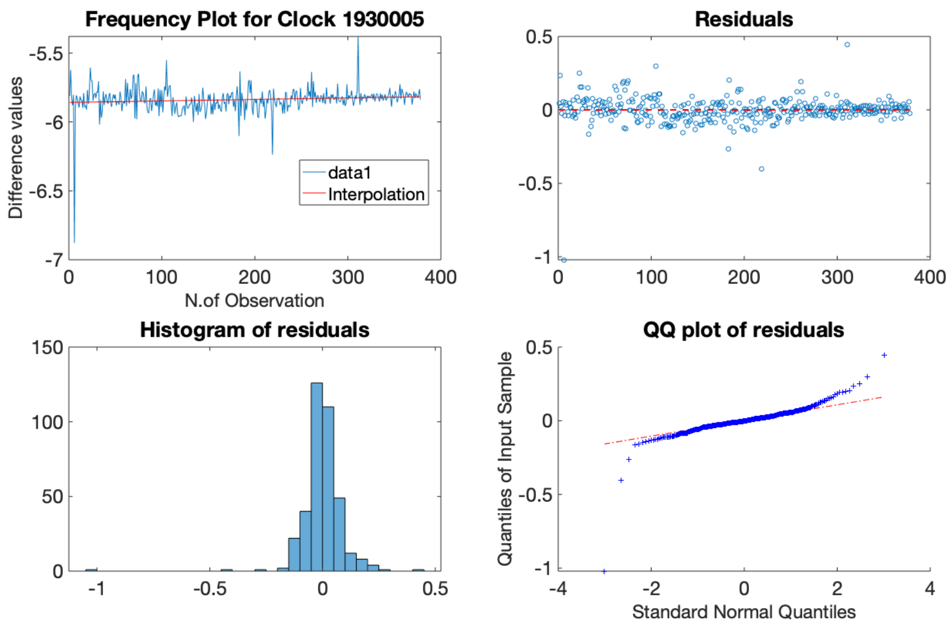


Figure 3.10: Drift and residual analysis for fountain 1930005.

From the plots above it seems that residuals are substantially normally distributed, but this was not the case for many other clocks.
The Jarque-Bera test for normality was also conducted to verify normality of residuals and the results were the following: for maser clocks the null hypothesis was accepted in the 10% of cases, for cesium in the 54% of cases, and was never accepted for the four fountains.
These results alone advocate that a simple linear regression model fails to capture some characteristics of the residuals behaviour.
Nonetheless, it is true that many masers and cesium show in some dates out of range measurements, that are probably influencing the statistics. I present below some examples of masers and cesium with critical behaviour (out of range frequency values) in some points (3.11, 3.12).

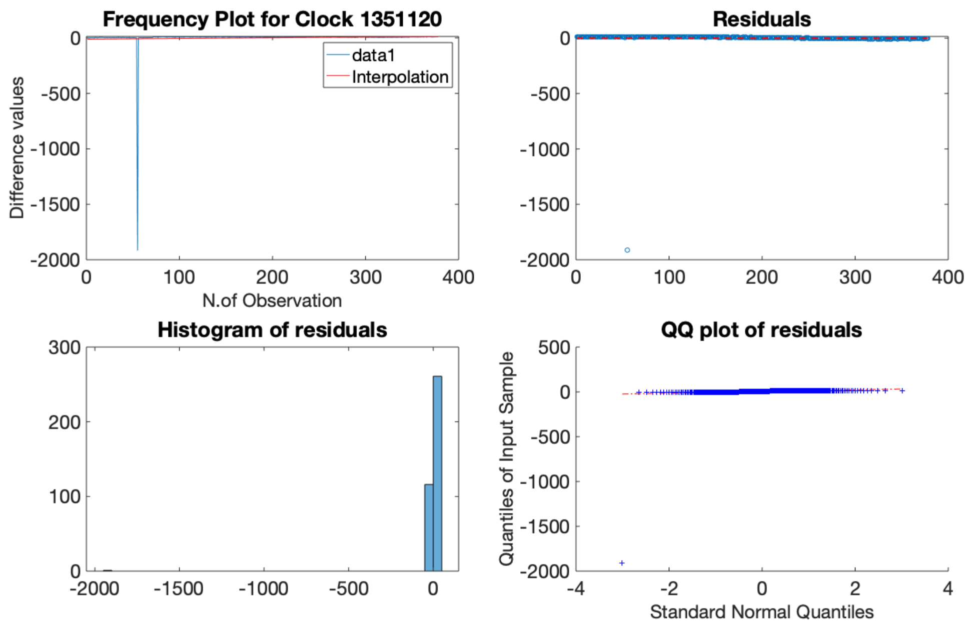


Figure 3.11: Drift and residual analysis for cesium 1351120. Presence of outliers.

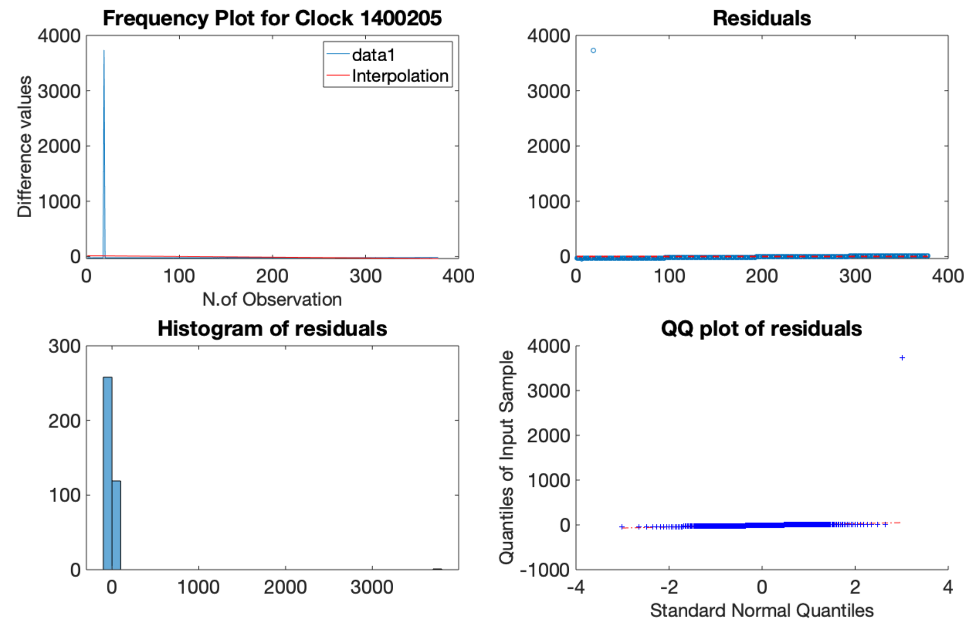


Figure 3.12: Drift and residual analysis for maser 1400205. Presence of outliers.

Most importantly, what was noted is that fountains do not seem to exhibit any drift at all. This suggested that applying to fountains a quadratic model could lead to poorer results than a simpler linear model.

On another note, plotting the drift estimation values, for each of the three months data batches form January 2018 until December 2022, as specified in 1.53, we obtained some insight on why estimation of the drift obtained assuming that it is constant is not well performing. In fact it seems that the drift is not constant for many clocks, especially masers. In figure 3.13 an example.

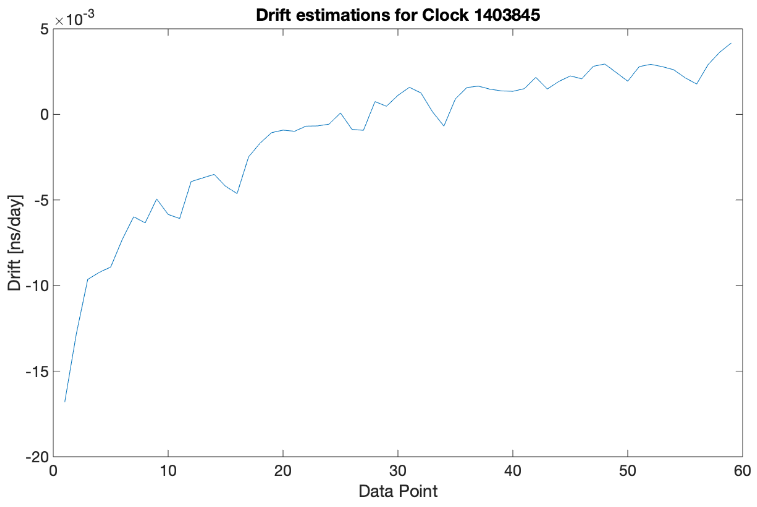


Figure 3.13: Drift subsequent estimations for clock 1403845.

3.2.1 Linear model

Linear model for fountains

In chapter 1 we outlined how the quality of EAL time scale, and consequently of UTC, depends on how well we are able to predict the phase error term for each clock $h'_i(t)$, using the following model for $t \in I_k$:

$$h'_i(t) = a_{i,k-1} + b_{i,k-1}(t-t_k) + 1/2 * c_{i,k-1} * (t_k - t_{k-1})(t-t_k) + 1/2 * c_{i,k-1}(t-t_k)^2 \quad (3.9)$$

where:

$$a_{i,k-1} = EAL(t_k) - h_i(t_k) \quad (3.10)$$

$$b_{i,k-1} = \frac{(EAL(t_k) - h_i(t_k)) - (EAL(t_{k-1}) - h_i(t_{k-1}))}{t_k - t_{k-1}} \quad (3.11)$$

and $c_{i,k-1}$ is the drift estimated with the least square technique using frequency data of the past three months of the clock with respect to TT.
In order to estimate the parameters $a_{i,k}$ and $b_{i,k}$ for each clock and time interval, we need time differences datasets of clocks with respect to EAL. I used a series of datasets from the BIPM with general name *EALH_corrmmy.* Each of these datasets contains, for every MJD and every clock code of month *mm* and year *yy*, the time difference value between EAL and the clock reading for that date.
Again I gathered 5 years of data from 2018 to 2022, and I applied the model separately for each subgroup of clocks.
I started by computing the drift $c_{i,1}$ using the first available three months of data from table *TTH1801*, then I used the last month of *EALH_corr1801* to compute $a_{i,1}$ and $b_{i,1}$. I used these parameters to compute the residuals on the next month in $TTH1802$. I iterated the algorithm until the final prediction and computation of residuals for December 2022. In the end, for a total of 59 months I computed the 5/6/7 (it depends on the length of the month) points of residuals corresponding to equally spaced dates by 5 days.

By looking at 5 years frequency data, from 2018 to 2022, for different types of clocks, we supposed that, given the extreme precision of rubidium fountains, the following model could be better:

$$h_i'(t) = EAL(t_k) - h_i(t_k) + \frac{(EAL(t_k) - h_i(t_k)) - (EAL(t_{k-1}) - h_i(t_{k-1}))}{t_k - t_{k-1}}(t - t_k) \tag{3.12}$$

In fact fountains frequency plots show nearly no drift, and a simpler model has got the advantage of eliminating the error related to the estimation of the drift, especially when there is no physical evidence of its presence.
The MATLAB code for the two models is found in B.6 and B.7.
Visual comparison of the residuals for model 3.9 and model 3.12 is shown in figures 3.16, 3.19, 3.22, 3.25.

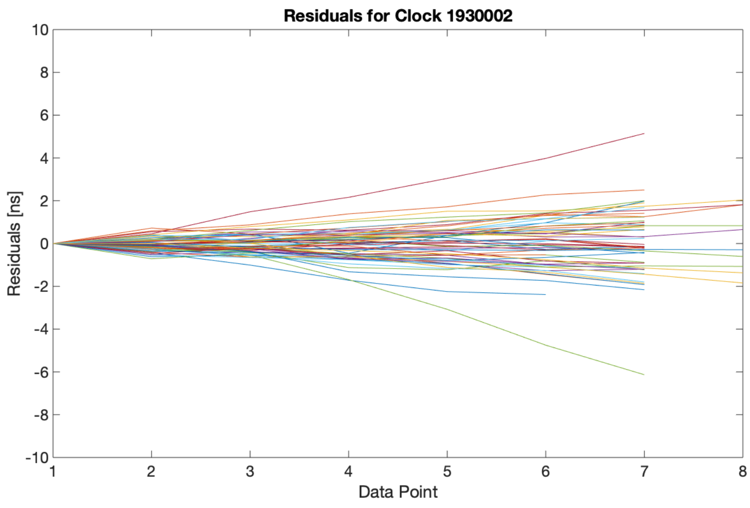


Figure 3.14: Residuals from quadratic model for clock 1930002.

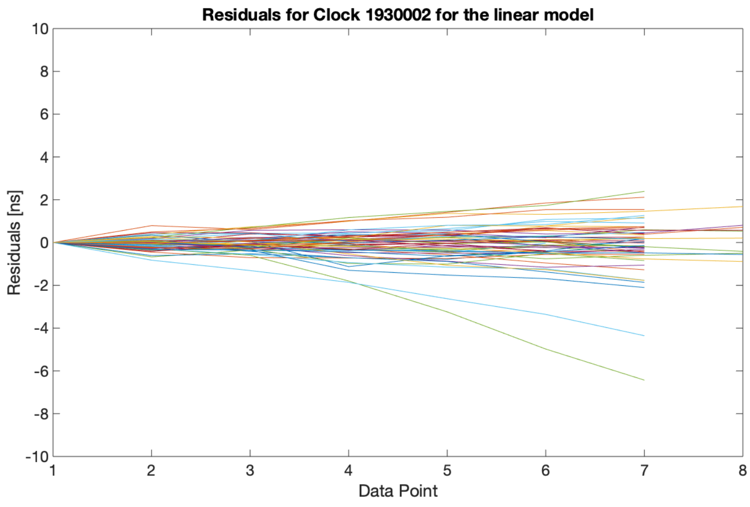


Figure 3.15: Residuals from linear model for clock 1930002.

Figure 3.16: Comparison of residuals from quadratic and linear models for clock 1930002.

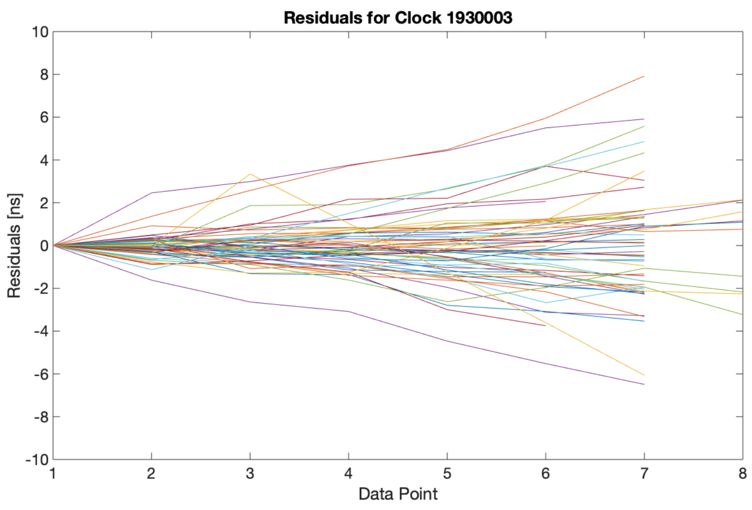


Figure 3.17: Residuals from quadratic model for clock 1930003.

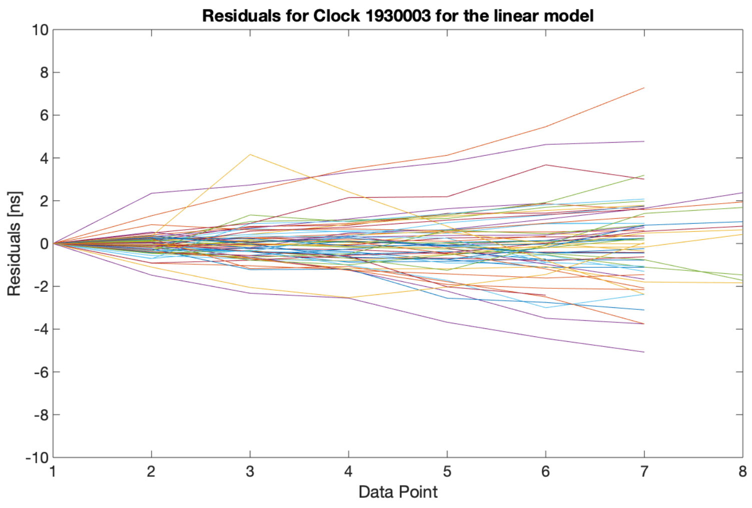


Figure 3.18: Residuals from linear model for clock 1930003.

Figure 3.19: Comparison of residuals from quadratic and linear models for clock 1930003.

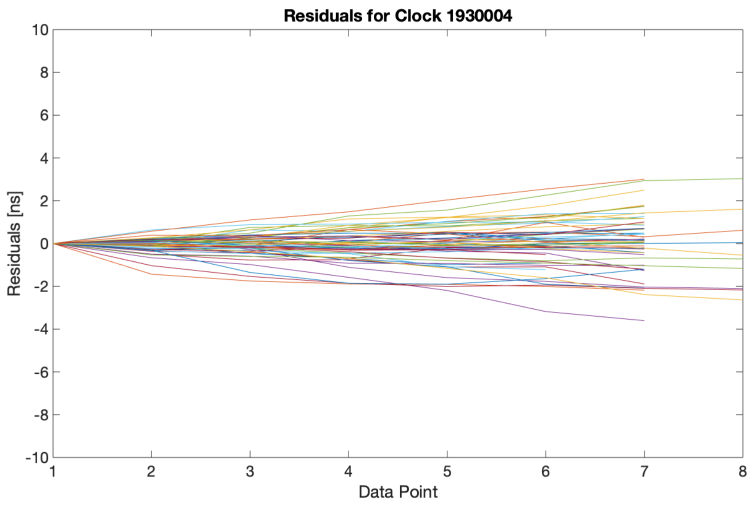


Figure 3.20: Residuals from quadratic model for clock 1930004.

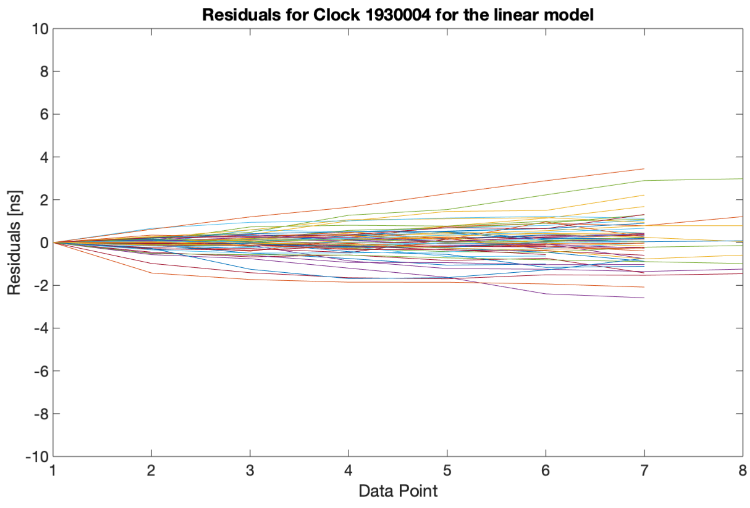


Figure 3.21: Residuals from linear model for clock 1930004.

Figure 3.22: Comparison of residuals from quadratic and linear models for clock 1930004.

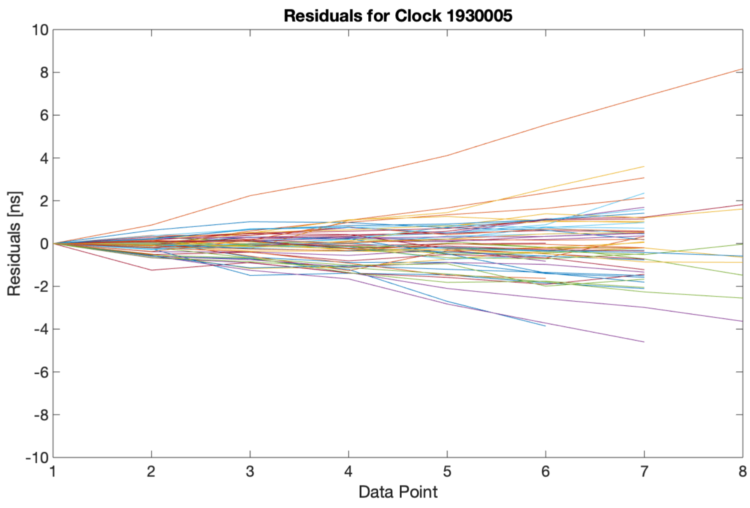


Figure 3.23: Residuals from quadratic model for clock 1930005.

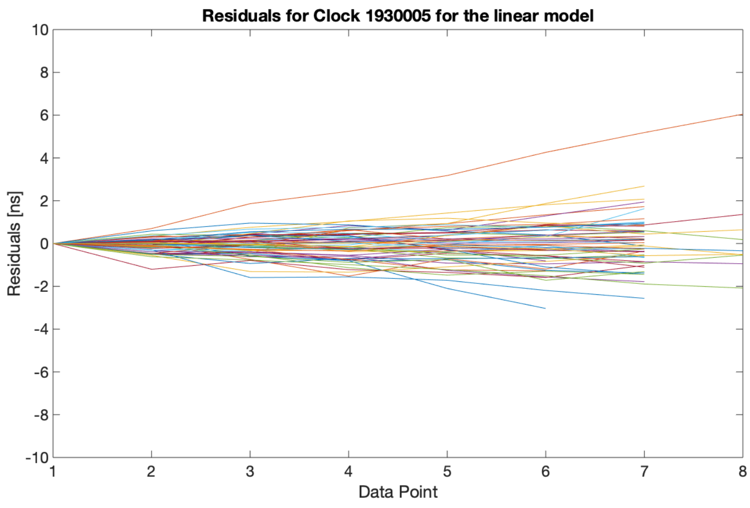


Figure 3.24: Residuals from linear model for clock 1930005.

Figure 3.25: Comparison of residuals from quadratic and linear models for clock 1930005.

I was also asked to analyze the results on a recently added fountain: 1934901 from NTSC. In fact this fountain was performing surprisingly bad compared to usual fountains standards. For this fountain I had to work on a shorter period, which was the longest period available from its inclusion in the ensemble. In particular I used data from August 2022 until October 2023.
Below I show the two-case plots for fountain 1934901 (3.28).

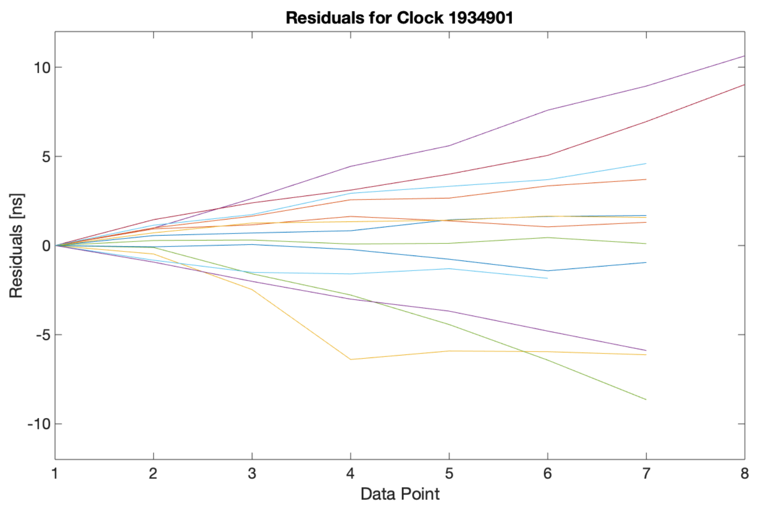


Figure 3.26: Residuals from quadratic model for clock 1934901.

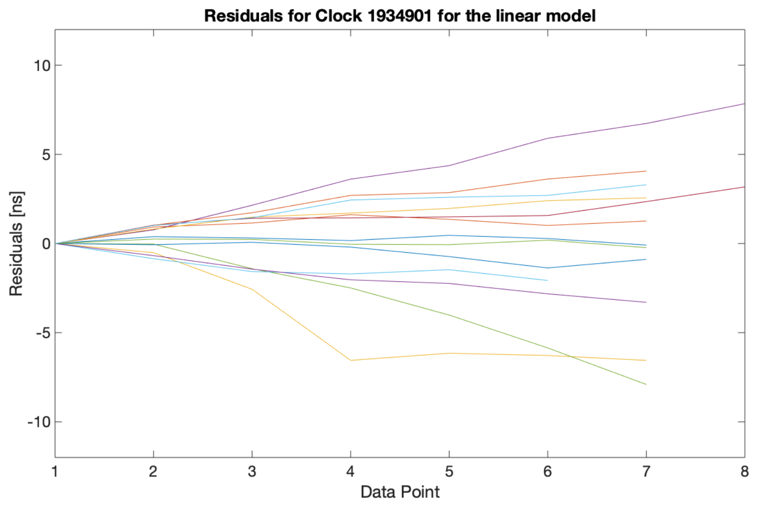


Figure 3.27: Residuals from linear model for clock 1934901.

Figure 3.28: Comparison of residuals of quadratic and linear models for clock 1934901.

It can be seen that there for all the fountains there is a decrease in residual values, at least for what concerns extreme points (maximum and minimum values of residuals).
I also did a different type of plot where, instead of showing the residuals of a clock for each month, I showed the behaviour of all fountains over each of the month. I report in figure 3.31, 3.34 this kind of plot for December 2022 (4 fountains) and for

October 2023 (5 fountains), respectively.

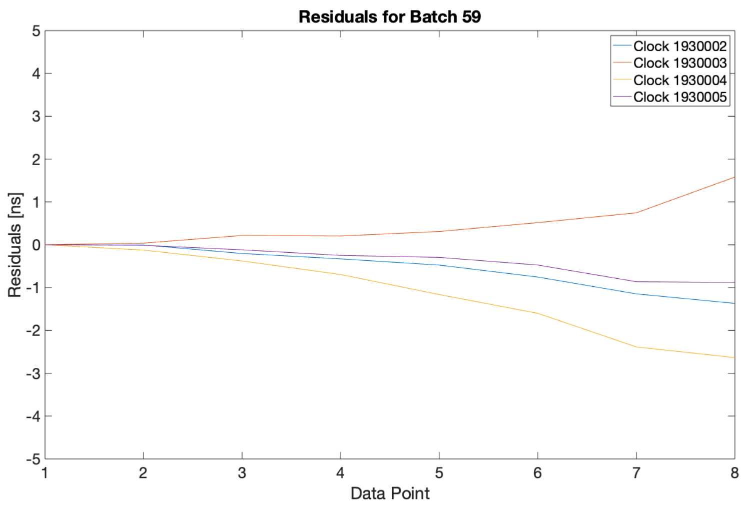


Figure 3.29: Residuals from quadratic model for December 2022, fountains.

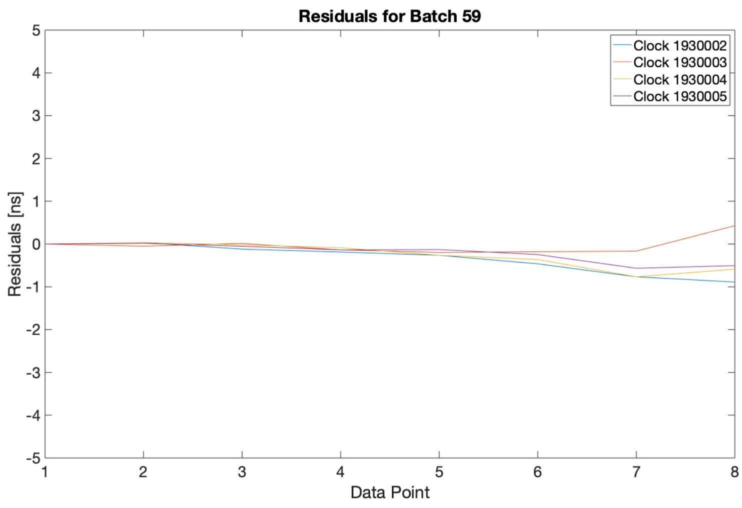


Figure 3.30: Residuals from linear model for December 2022, fountains.

Figure 3.31: Comparison of residuals over the same month for quadratic and linear model, Dec 2022, fountains.

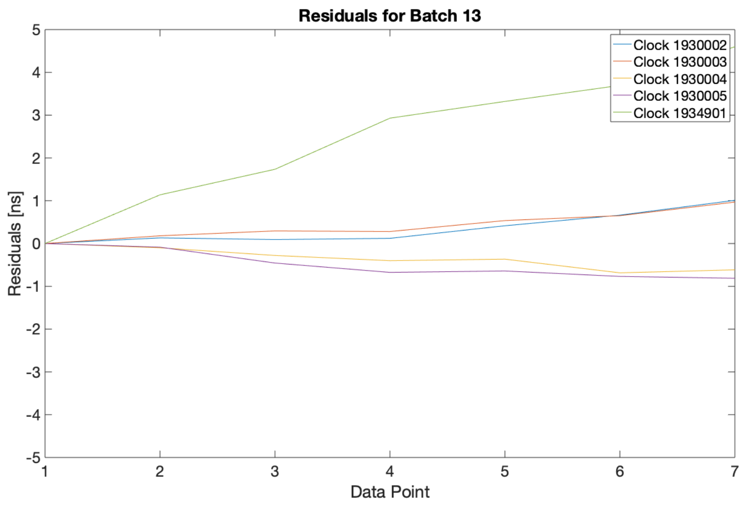


Figure 3.32: Residuals from quadratic model for October 2023, fountains.

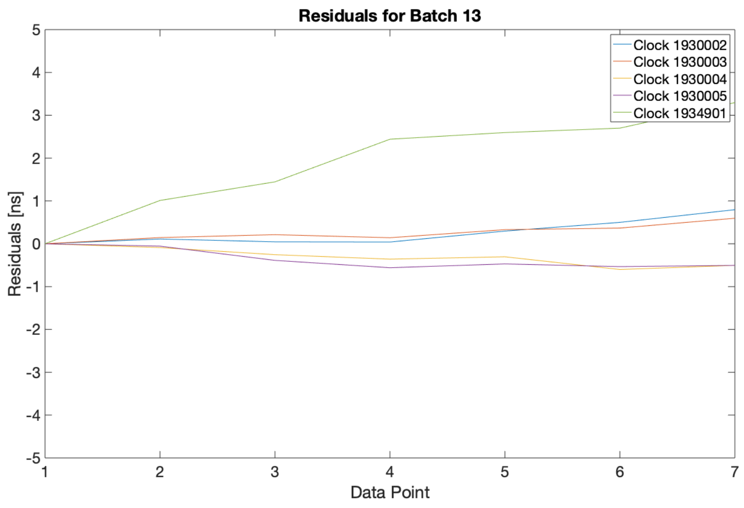


Figure 3.33: Residuals from linear model for October 2023, fountains.

Figure 3.34: Comparison of residuals over the same month for quadratic and linear model, Oct 2023, fountains.

Even with these plots it is quite clear that the linear model constitutes an improvement.

In order to gain more insights on whether the linear model is actually better than the quadratic one, I decided to look at the overall behaviour of fountains, over each month, in the two cases. I did that by averaging the residuals across all fountains for each data point, and then showing averages and standard deviation of data separately per month, as it can be seen in figures 3.35 and 3.36 . I used the period going from August 2022 to October 2023, in order to include in the comparison also fountain 1934901.

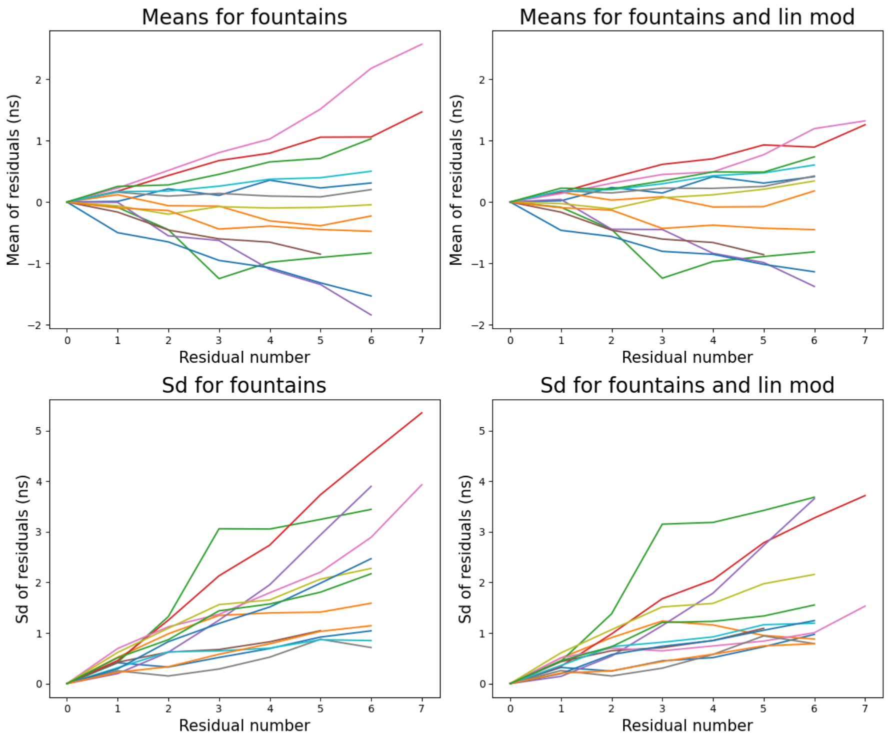


Figure 3.35: Mean of residuals and standard deviation across fountains , shown separately per month, from August 2022 to October 2023, for the two cases: linear and quadratic.

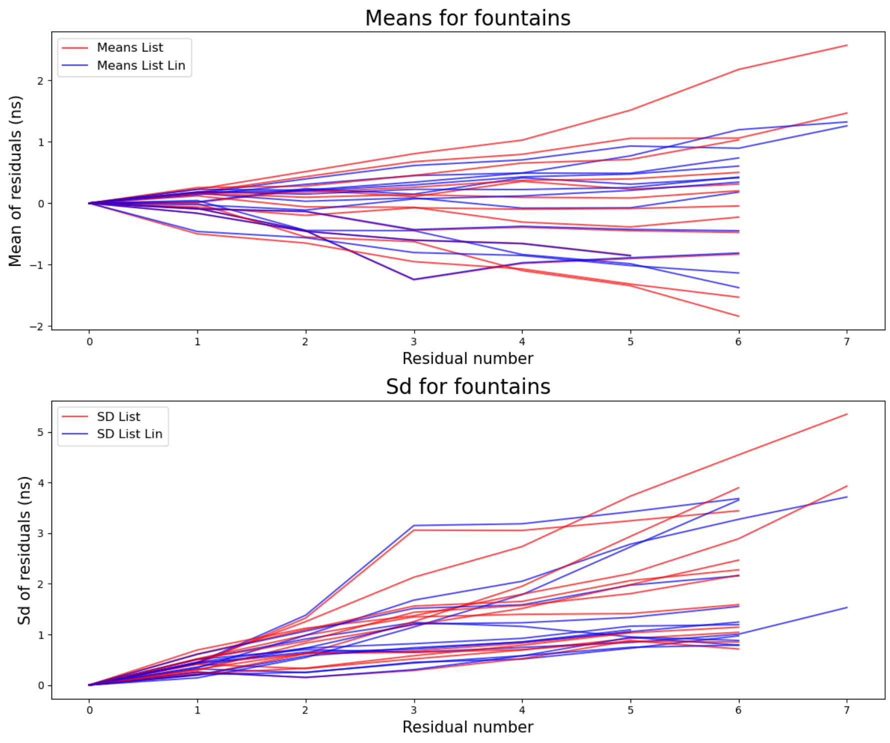


Figure 3.36: Mean of residuals and standard deviation across fountains , shown separately per month, from August 2022 to October 2023, for the two cases: linear and quadratic, superimposed.

From these plots it is clearer that there is an overall improvement of residuals with the linear model.
Then I did some intuitive visualization such as boxplots and histograms in order to better understand the global advantage of a linear model for clocks of type fountain. In particular for the boxplots:

- Figure 3.37 is a boxplot of mean residuals values.
- Figure 3.38 is a boxplot of the standard deviation values computed for every data point across fountains.
- Figure 3.39 is a boxplot of the standard deviation values computed every 7th data point of each month across fountains (in fact over one month residuals tend to grow, and the 7th day of residuals is normally the furthest from the initial value, which makes it more interesting for analyzing the effect of the linear model on standard deviation).

These plots seem to confirm that with the linear model not only the residuals of fountains are closer to zero, but they are less disperse around zero. In particular there is one nanosecond reduction in the median value of standard deviations at the 7th point of residuals, which is very promising.

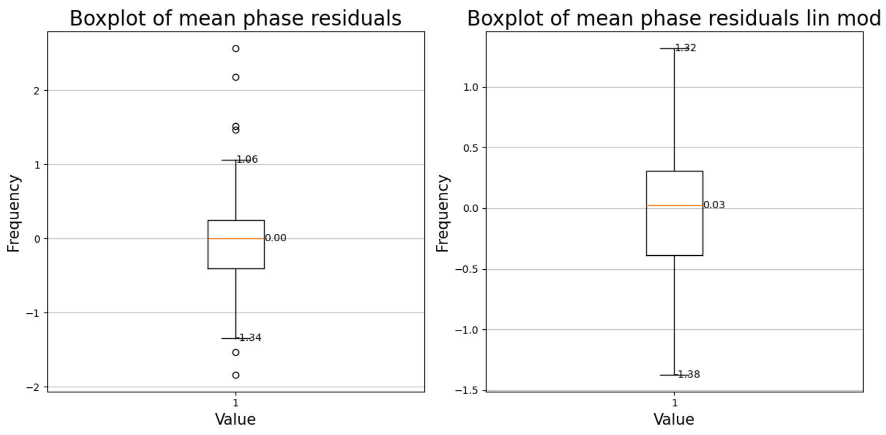


Figure 3.37: Boxplot of mean residuals for fountains, from August 2022 to October 2023, in the two cases: linear and quadratic.

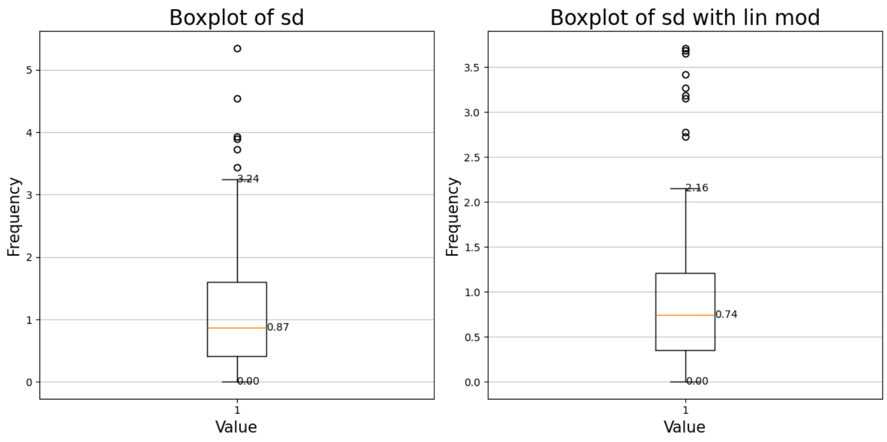


Figure 3.38: Boxplot of standard deviation of residuals for fountains, from August 2022 to October 2023, in the two cases: linear and quadratic.

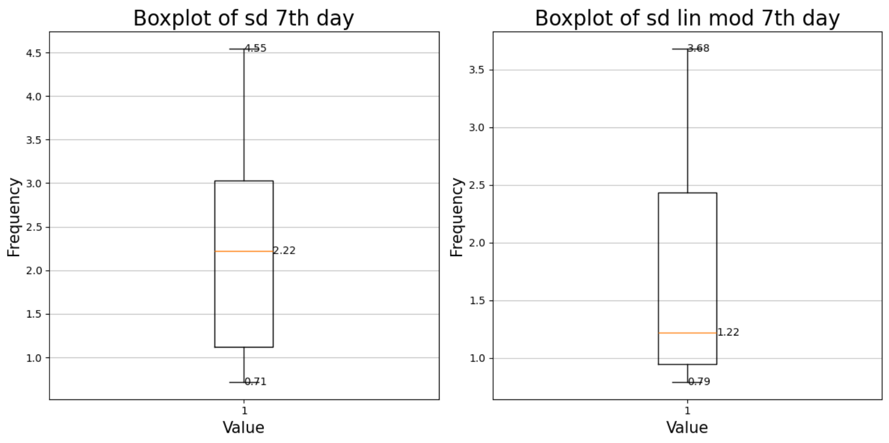


Figure 3.39: Boxplot of sd of residuals for fountains showing only the last day of residuals every month, from August 2022 to October 2023, in the two cases: linear and quadratic.

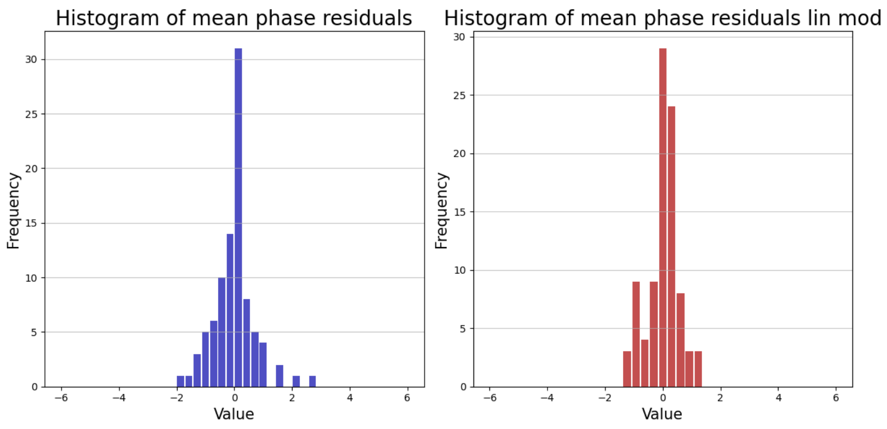


Figure 3.40: Histogram of mean residual for fountains, from August 2022 to October 2023, in the two cases: linear and quadratic.

To obtain further evidence in favor of the linear model, my supervisor Gianna Panfilo calculated EAL (using internal programs from the BIPM) by applying the linear model instead of the quadratic one. As we can see in figures 3.41 and 3.42, were the comparisons of the weights attributed to fountains are reported, the linear model describes better the behaviour of the Rubidium fountains.

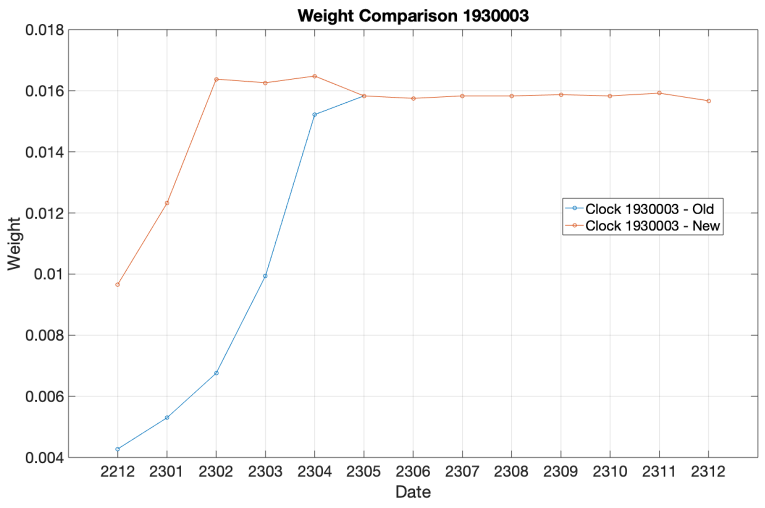


Figure 3.41: Weights from linear model vs. weights from quadratic model for clock 1930003.

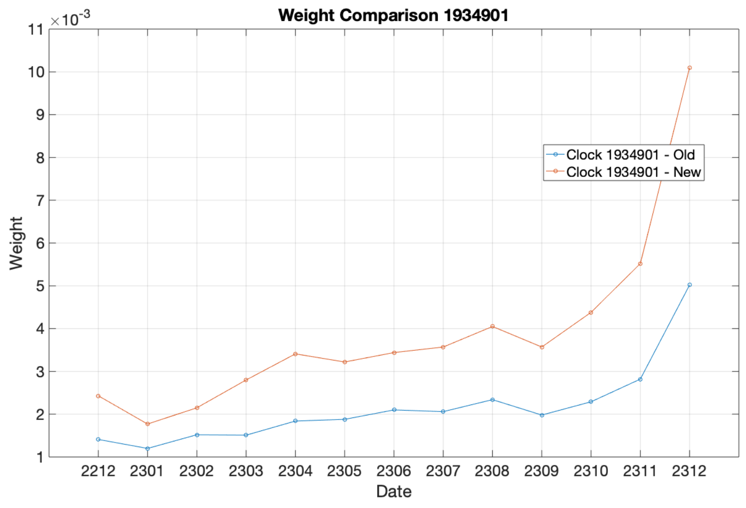


Figure 3.42: Weights from linear model vs. weights from quadratic model for clock 1934901.

Clocks 1920002, 1930004 and 1930005 keep the maximum weight in both cases, while both clock 1934901 and 1930003 obtain greater weights if the linear model is applied to correct their phase error term.
Another useful visualization, which goes in favor of the linear model for fountains, is the actual plot of the drift estimations for one fountain, one maser and one

cesium, in figure 3.43. We can see how the fountain drift estimations are extremely close to zero.

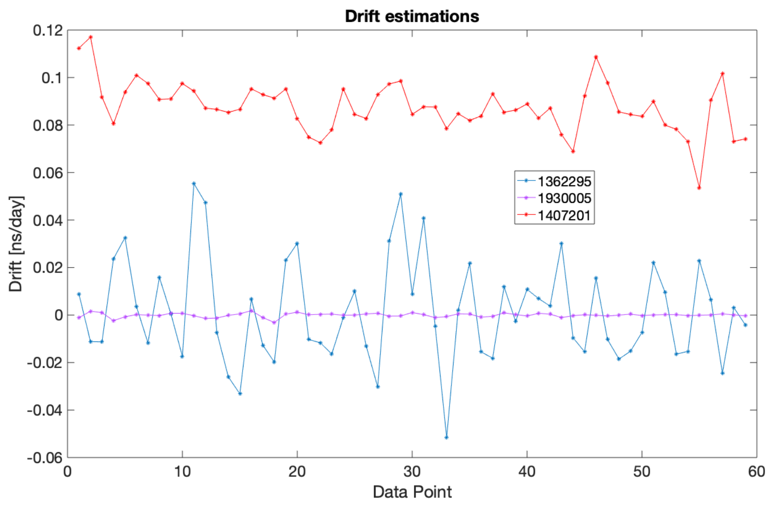


Figure 3.43: Drift estimation values over five years from Jan 2018 to Dec 2022, for one cesium, one maser, and one fountain.

Linear model for cesium clocks

Image 3.43 suggested the possibility that also for cesium clocks a linear model could be better suited, in fact drift values, although they oscillate a lot, seem to oscillate around 0.
According to the following plots, which reproduce the same analysis done for fountains, there seems indeed to be a little improvement in the use of a linear model for cesium clocks, especially in terms of standard deviation.

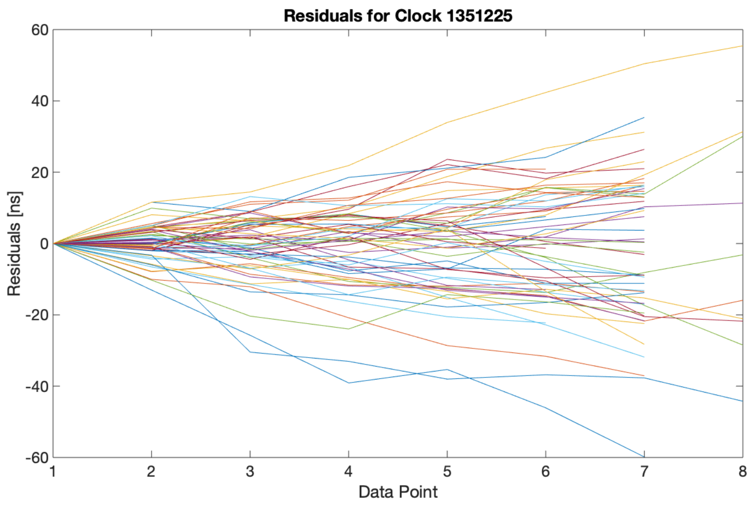


Figure 3.44: Residuals from quadratic model for clock 1351225.

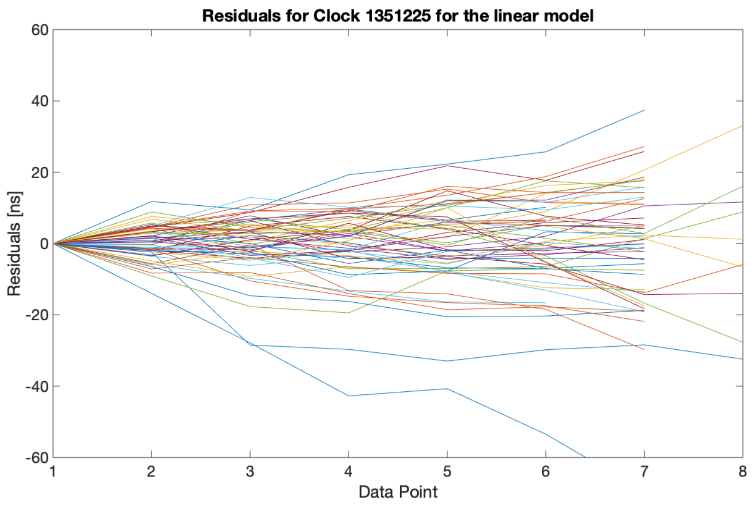


Figure 3.45: Residuals from linear model for clock 1351225.

Figure 3.46: Comparison of residuals from quadratic and linear models for clock 1351225.

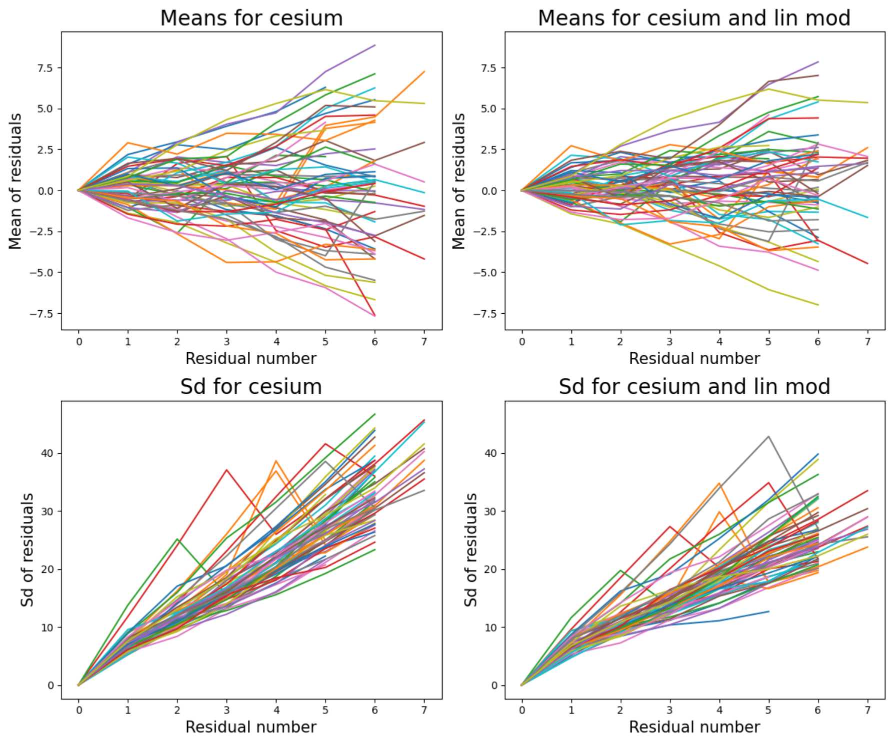


Figure 3.47: Mean of residuals and standard deviation across cesium clocks, shown separately per month, from January 2018 to December 2022, for the two cases: linear and quadratic.

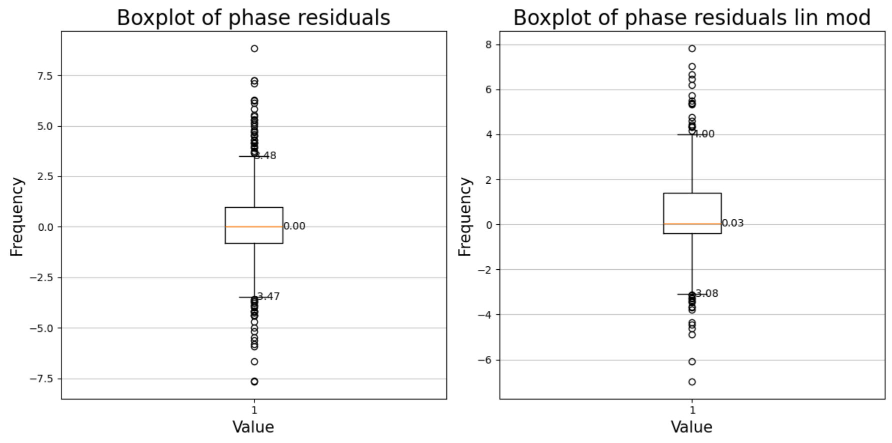


Figure 3.48: Boxplot of mean residuals for cesium clocks, from January 2018 to December 2022, in the two cases: linear and quadratic.

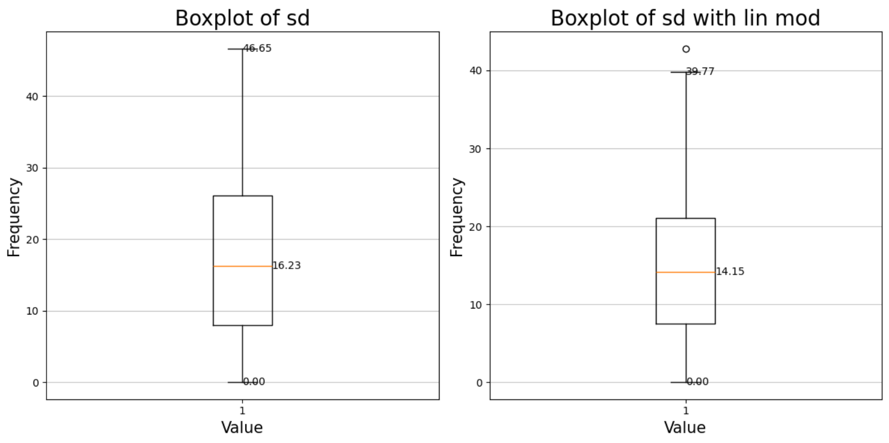


Figure 3.49: Boxplot of standard deviation of residuals for cesium clocks, from January 2018 to December 2022, in the two cases: linear and quadratic.

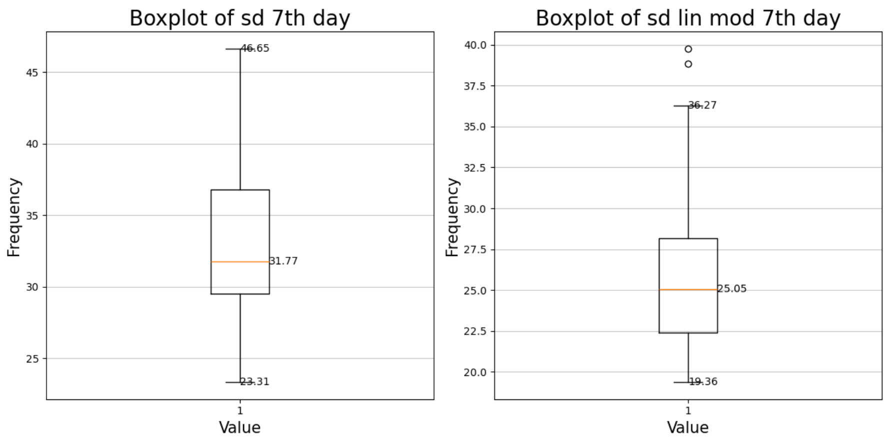


Figure 3.50: Boxplot of sd of residuals for cesium clocks showing only the last day of residuals every month, from January 2018 to December 2022, in the two cases: linear and quadratic.

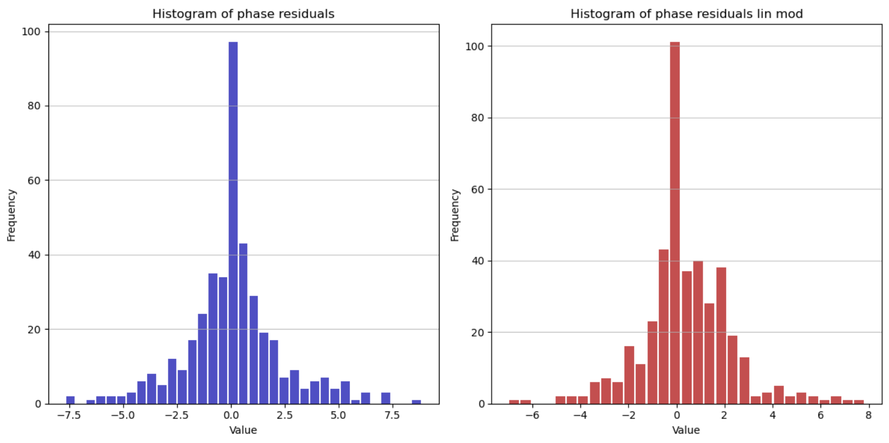


Figure 3.51: Histogram of mean residual for cesium clocks, from January 2018 to December 2022, in the two cases: linear and quadratic.

Applying a linear model to cesium clocks probably is more effective in predicting cesium clocks deviation from the ideal. This was confirmed by looking at the weights of all cesium across one year, which were obtained as described in the previous paragraph, from December 2022 to December 2023. The effect on the weights over this year is reported in figures 3.52, 3.53.

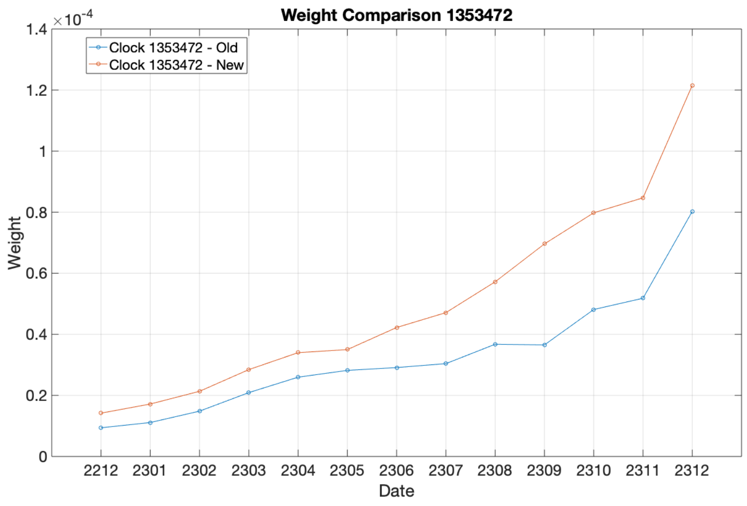


Figure 3.52: Weights from linear model vs. weights from quadratic model for clock 1353472.

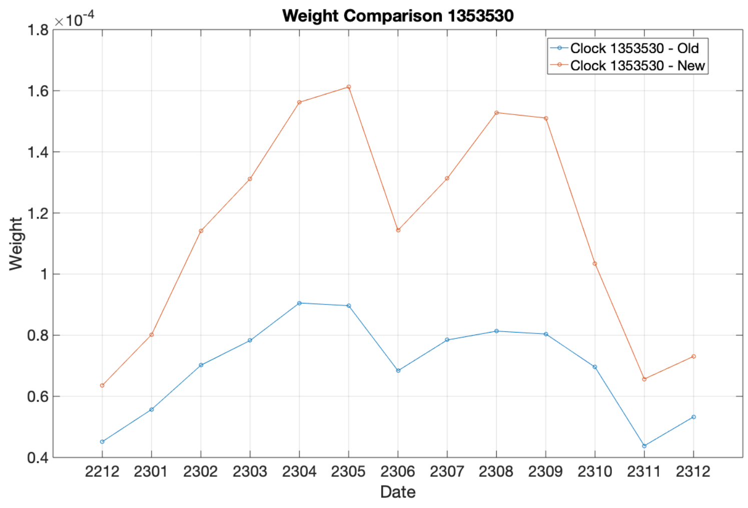


Figure 3.53: Weights from linear model vs. weights from quadratic model for clock 1353530.

3.2.2 Least squares frequency model

Another thing the BIPM wanted to investigate was the possibility of modifying the estimate of the frequency term to be used during interval I_k.
Now the model for this term is:

$$b_{i,k-1} = \frac{[EAL(t_k) - h_i(t_k)] - [EAL(t_{k-1}) - h_i(t_{k-1})]}{t_k - t_{k-1}} \tag{3.13}$$

The problem with this model is that it can oversee any time jumps of clocks happening in the middle of the interval and adjusting before the interval ends. Such frequency jumps are checked manually at the BIPM, and clocks who have them are given weight 0 and thus excluded from the calculation of UTC for the month in question.
It seemed more reasonable to compute the frequency with a least squares model. The results of applying a linear least squares technique against the standard algorithm are shown for one cesium, one maser and one fountain in figures 3.59, 3.56, 3.16 .

The code for computation of residuals on the least frequency case is found in B.6.

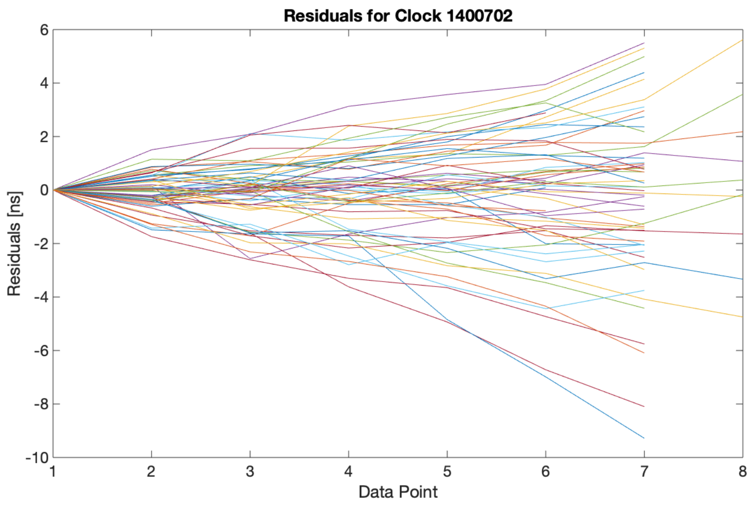


Figure 3.54: Residuals from standard frequency model for maser 1400702.

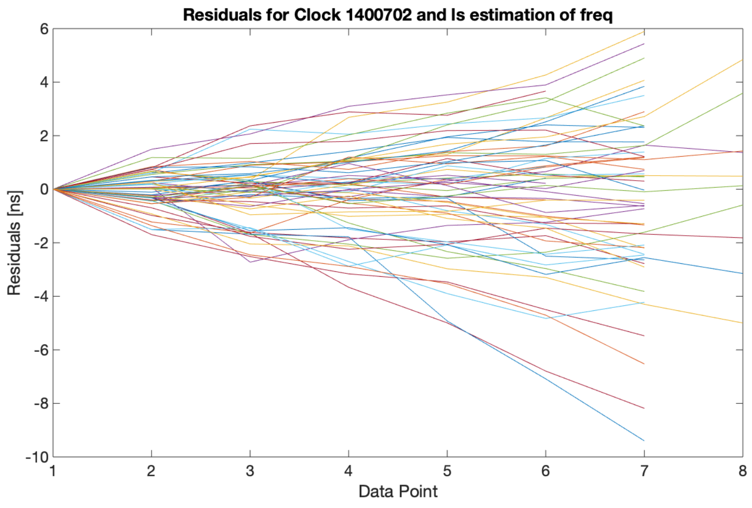


Figure 3.55: Residuals from ls frequency model for maser 1400702.

Figure 3.56: Comparison of residuals from standard and ls frequency models for maser 1400702.

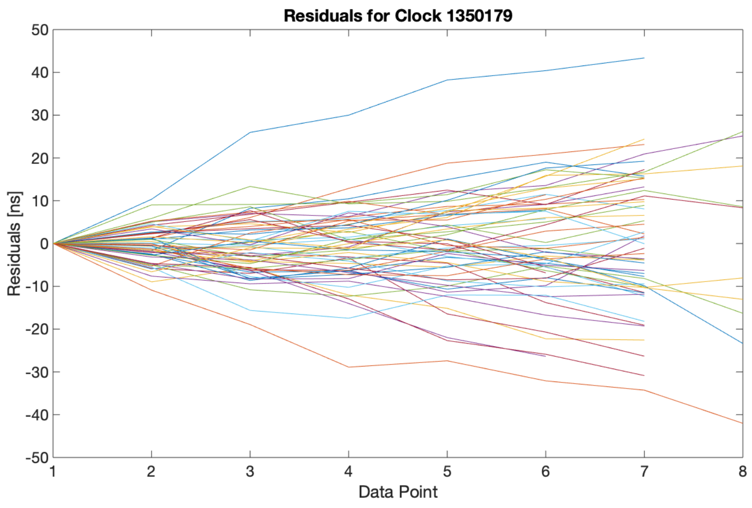


Figure 3.57: Residuals from standard frequency model for cesium 1350179.

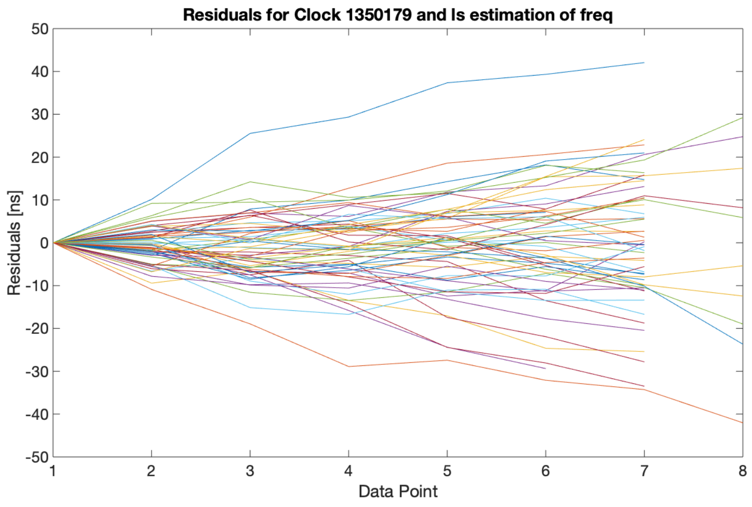


Figure 3.58: Residuals from ls frequency model for cesium 1350179.

Figure 3.59: Comparison of residuals from standard and ls frequency models for cesium 1350179.

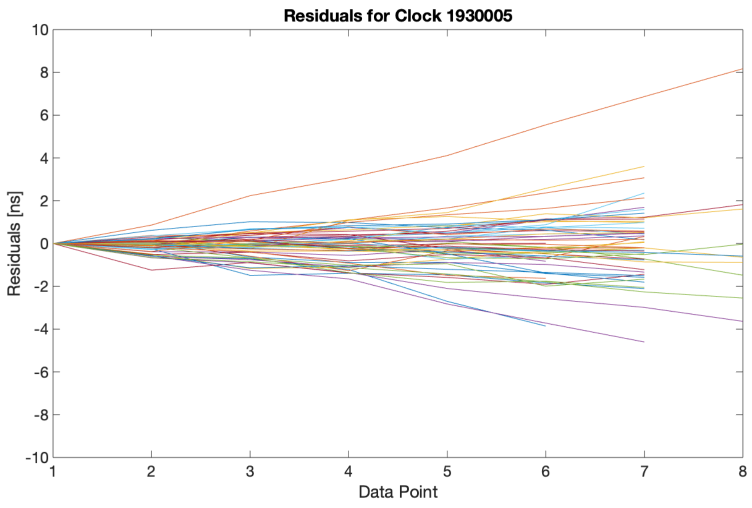


Figure 3.60: Residuals from standard frequency model for fountain 1930005.

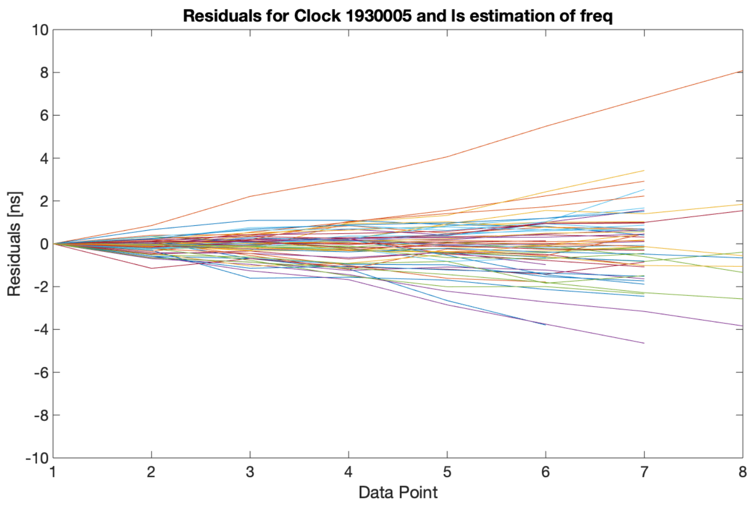


Figure 3.61: Residuals from ls frequency model for cesium 1930005.

Figure 3.62: Comparison of residuals from standard and ls frequency models for fountain 1930005.

Like was done for the linear model there is also a plot of the residual behaviour considering two selected months. In this case December 2022 for cesium (3.65), November 2022 for masers (3.68) and October 2023 (3.71) for fountains.

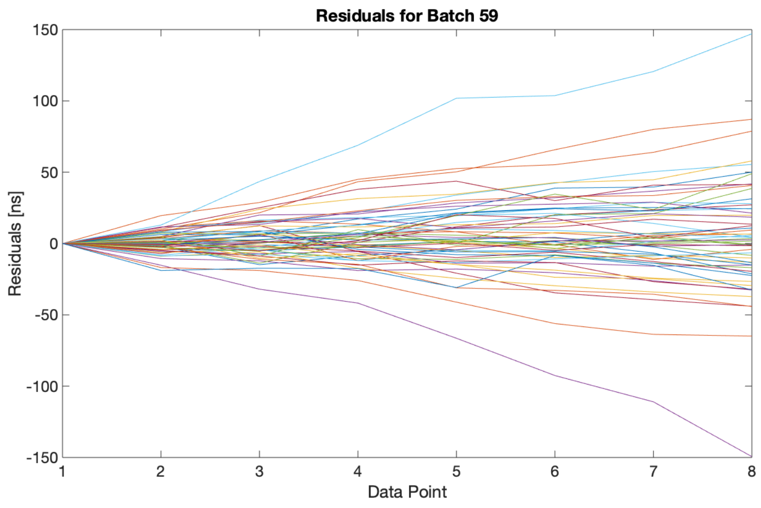


Figure 3.63: Residuals from standard frequency model for December 2022, cesium.

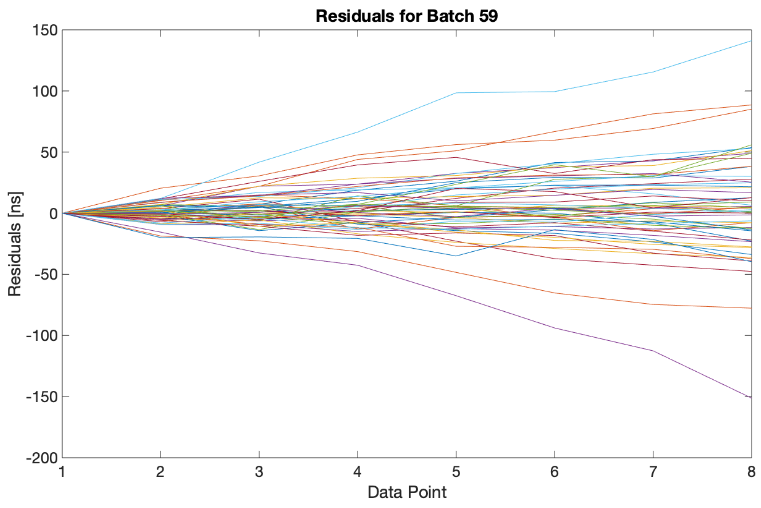


Figure 3.64: Residuals from least square frequency model for December 2022, cesium.

Figure 3.65: Comparison of residuals over the same month for the two frequency models, cesium.

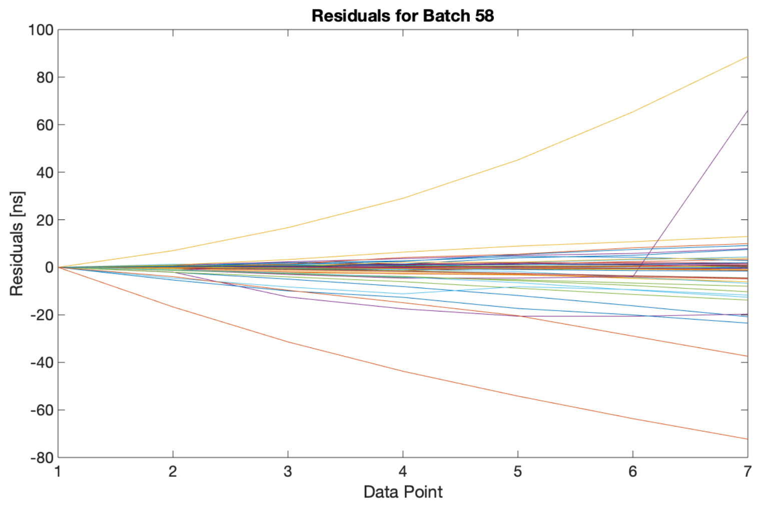


Figure 3.66: Residuals from standard frequency model for November 2022, masers.

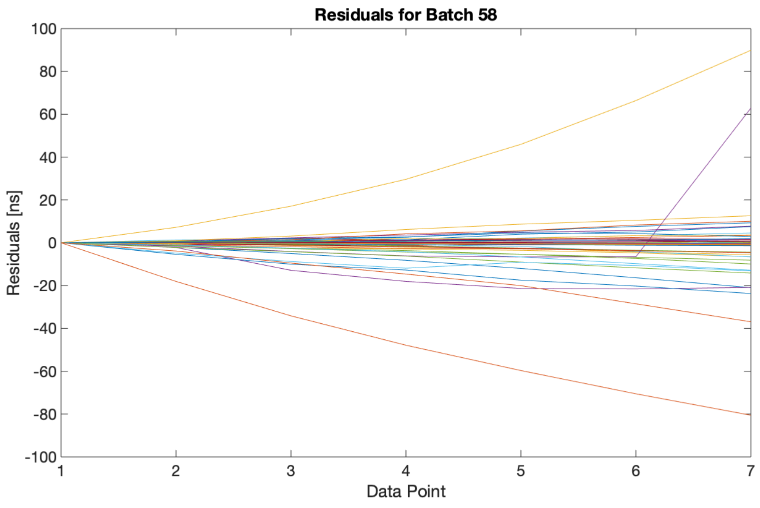


Figure 3.67: Residuals from least square frequency model for November 2022, masers.

Figure 3.68: Comparison of residuals over the same month for the two frequency models, masers.

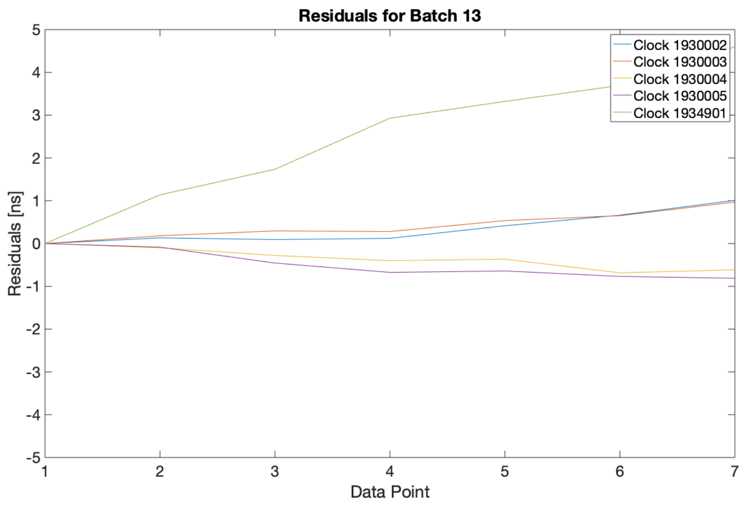


Figure 3.69: Residuals from standard frequency model for October 2023, fountains.

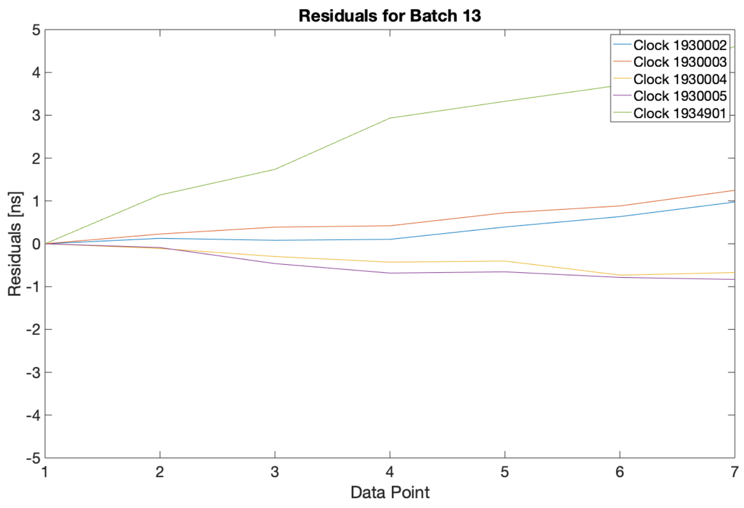


Figure 3.70: Residuals from least square frequency model for October 2023, fountains.

Figure 3.71: Comparison of residuals over the same month for the two frequency models, fountains.

Again, as I did for the linear model, I looked at the overall behaviour of clocks in the two cases. In the figures below I consider mean of residuals across all cesium, masers and fountains. In plot 3.73, 3.72, 3.74 there is a visualization of this mean values separately per month, while the boxplots 3.78, 3.75, 3.81, 3.79, 3.76, 3.82,

3.80, 3.77, 3.83, and the histograms 3.85, 3.84, 3.86 plots take into account the overall sequence of mean/sd residual values in 5 years. For the masers, every time one of them had residuals higher than 100 ns or lower than -100 ns, it was excluded from the computation of the mean in that month. In fact masers with residuals out of the range $[-100,100]$ ns clearly had some contingent problems and are not representative of masers behaviour.
The same was done for cesium clocks with residuals higher than 250 ns or lower than -250 ns.

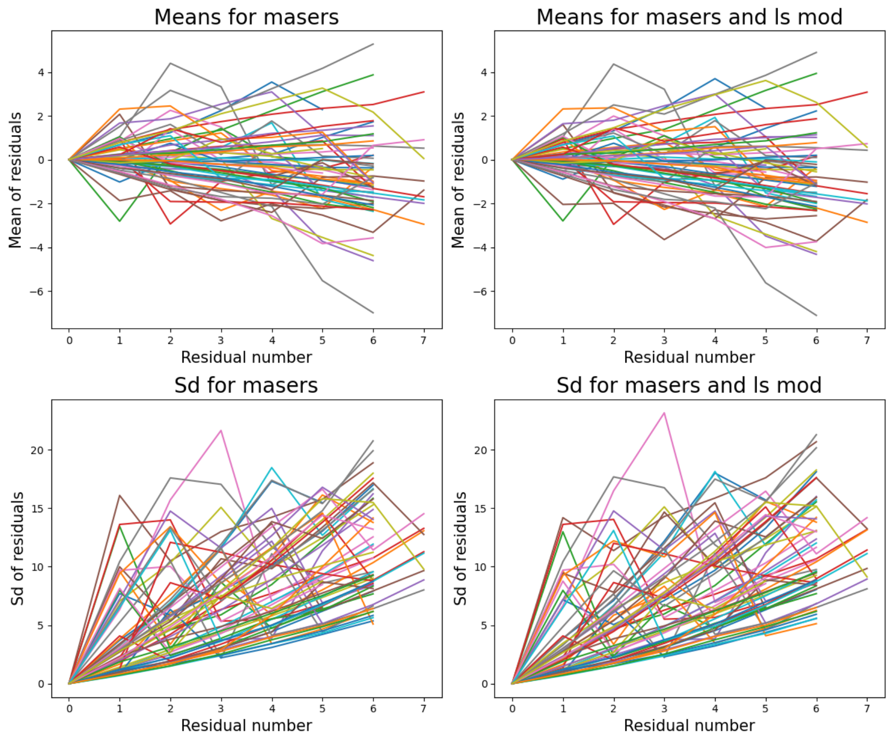


Figure 3.72: Mean of residuals and standard deviation across masers , shown separately per month, from February 2018 to December 2022, for the two cases: frequency standard and frequency ls.

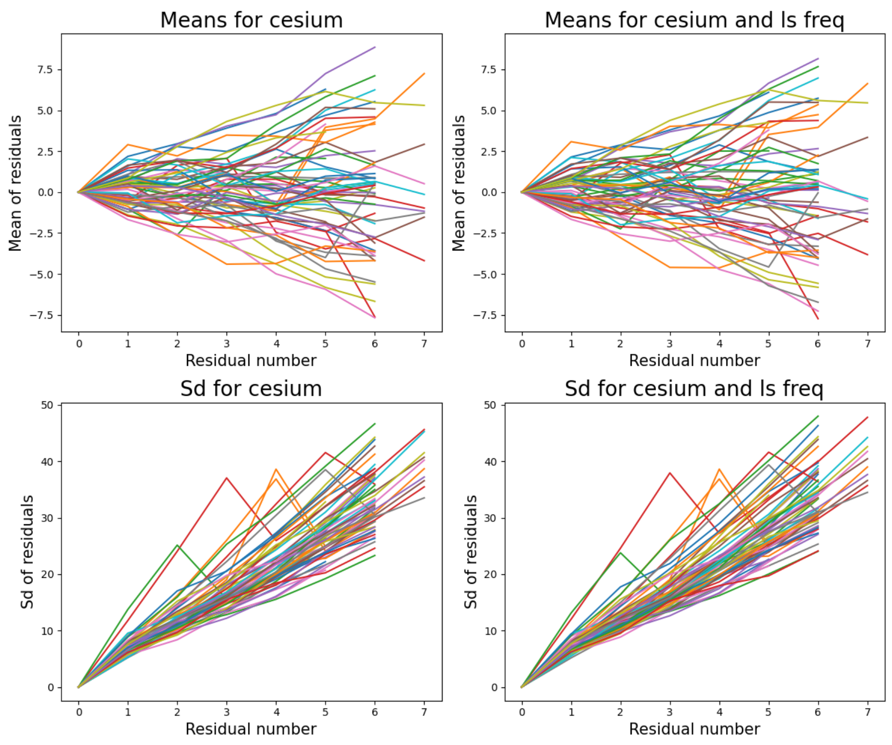


Figure 3.73: Mean of residuals and standard deviation across cesium, shown separately per month, from February 2018 to December 2022, for the two cases: frequency standard and frequency ls.

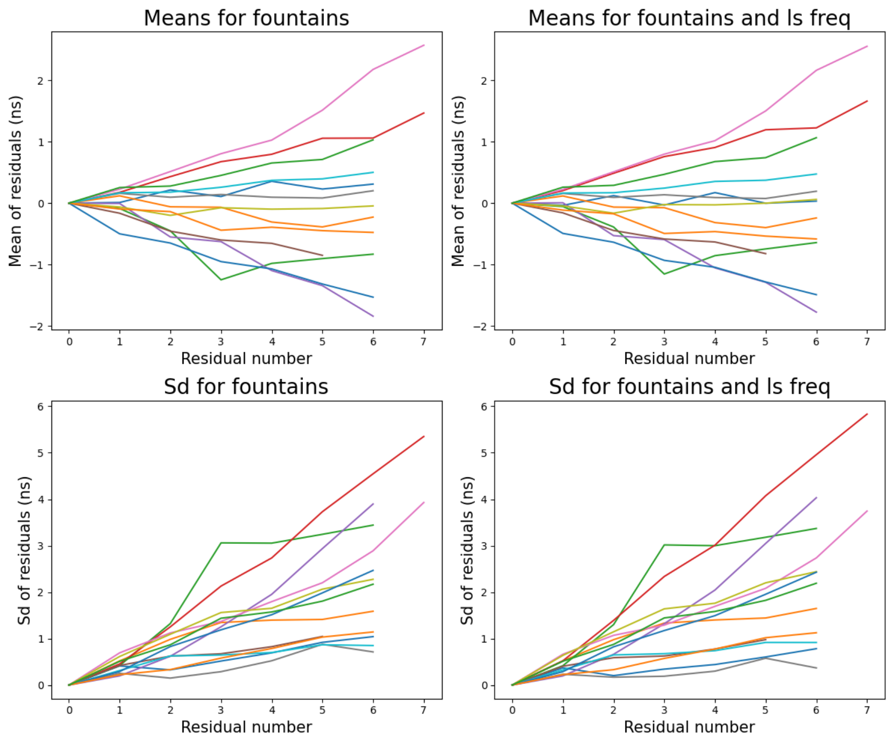


Figure 3.74: Mean of residuals and standard deviation across fountains shown separately per month, from August 2022 to October 2023, for the two cases: frequency standard and frequency ls.

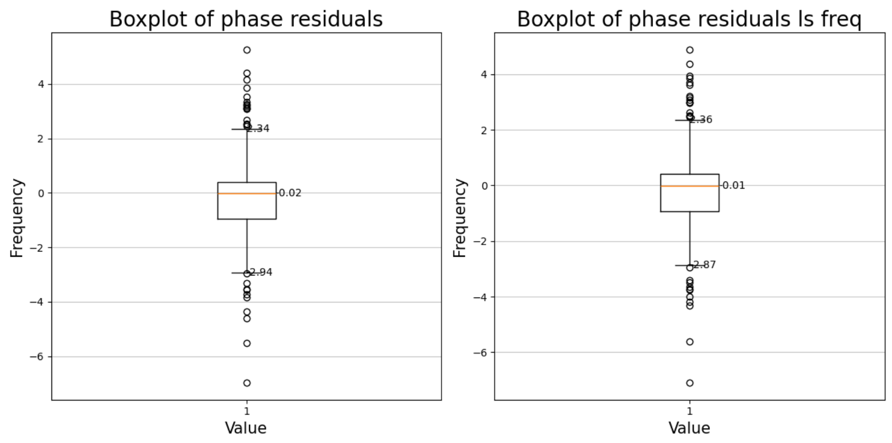


Figure 3.75: Boxplot of mean residuals for masers, from February 2018 to December 2022, in the two cases: frequency standard and frequency ls.

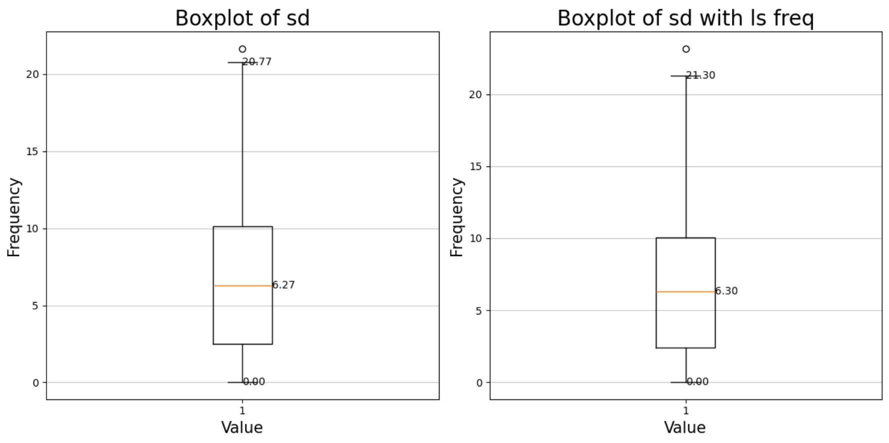


Figure 3.76: Boxplot of sd of residuals for masers, from February 2018 to December 2022, in the two cases: frequency standard and frequency ls.

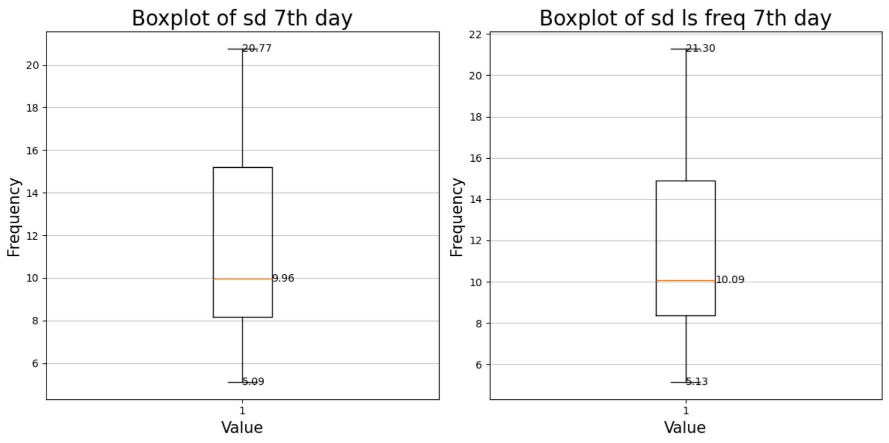


Figure 3.77: Boxplot of sd of residuals in the 7th day of residuals for masers, from February 2018 to December 2022, in the two cases: frequency standard and frequency ls.

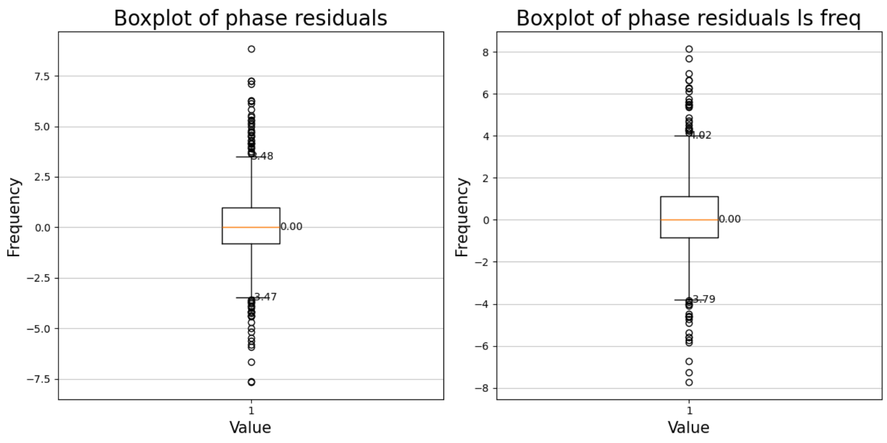


Figure 3.78: Boxplot of mean residuals for cesium, from February 2018 to December 2022, in the two cases: frequency standard and frequency ls.

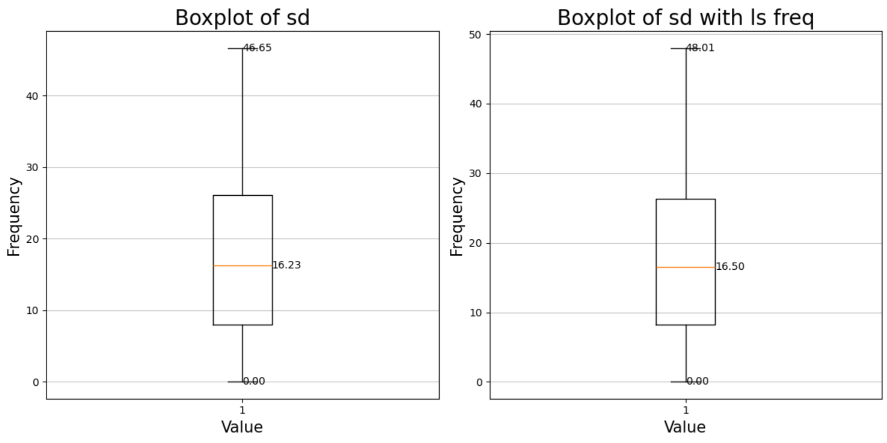


Figure 3.79: Boxplot of sd of residuals for cesium, from February 2018 to December 2022, in the two cases: frequency standard and frequency ls.

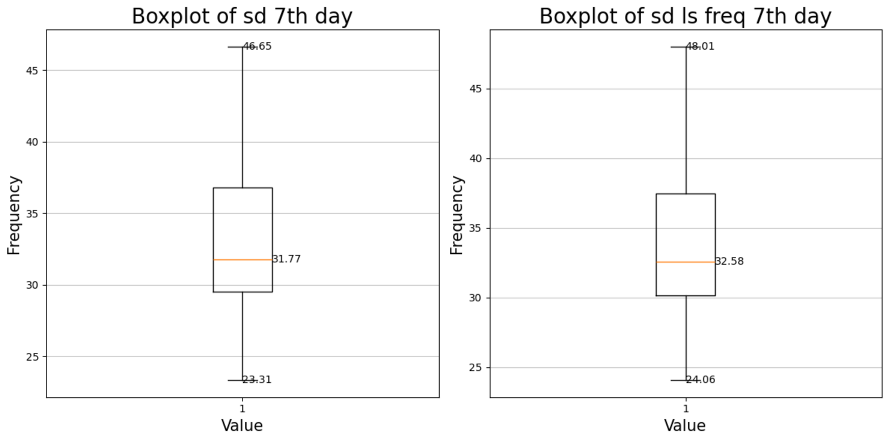


Figure 3.80: Boxplot of sd of residuals in the 7th day of residuals for cesium, from February 2018 to December 2022, in the two cases: frequency standard and frequency ls.

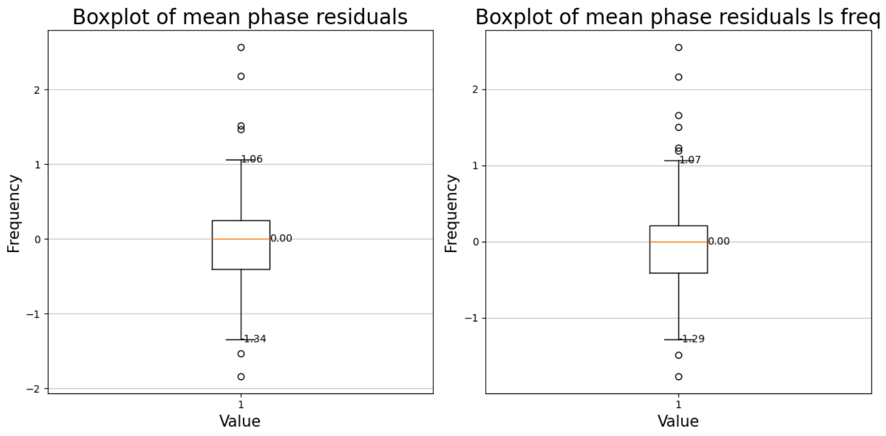


Figure 3.81: Boxplot of mean residuals for fountains, from August 2022 to October 2023, in the two cases: frequency standard and frequency ls.

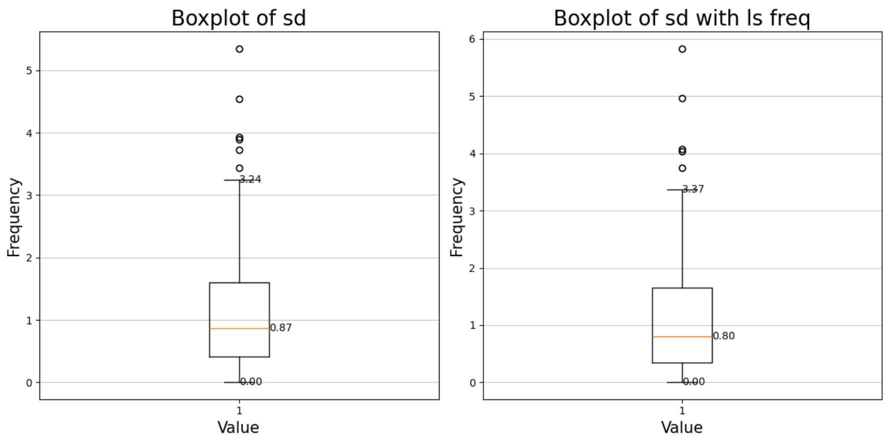


Figure 3.82: Boxplot of sd of residuals for fountains, from August 2022 to October 2023, in the two cases: frequency standard and frequency ls.

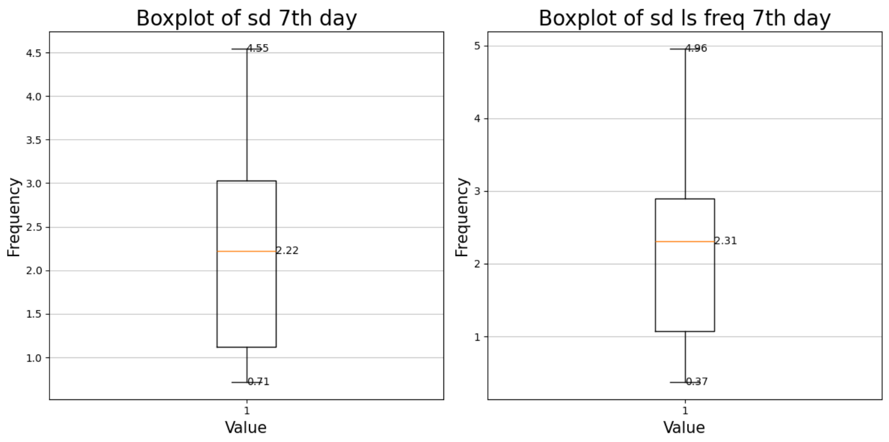


Figure 3.83: Boxplot of sd of residuals in the 7th day of residuals for fountains, from August 2022 to October 2023, in the two cases: frequency standard and frequency ls.

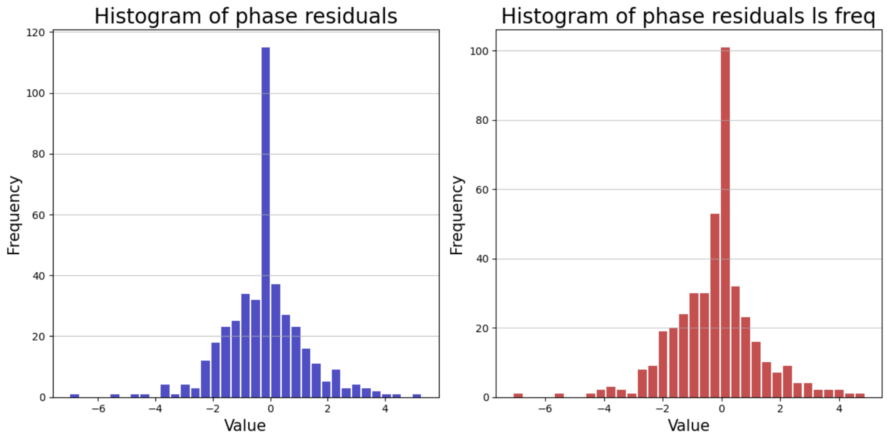


Figure 3.84: Histogram of mean residual for masers, from February 2018 to December 2022, in the two cases: frequency standard and frequency ls.

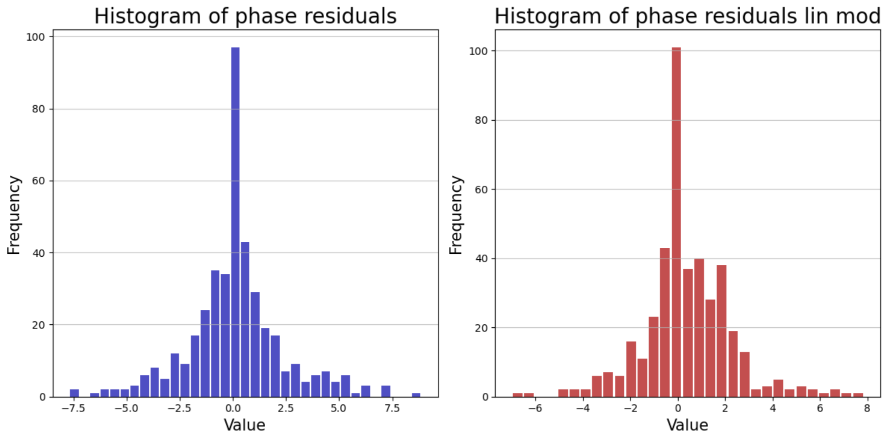


Figure 3.85: Histogram of mean residual for cesium, from February 2018 to December 2022, in the two cases: frequency standard and frequency ls.

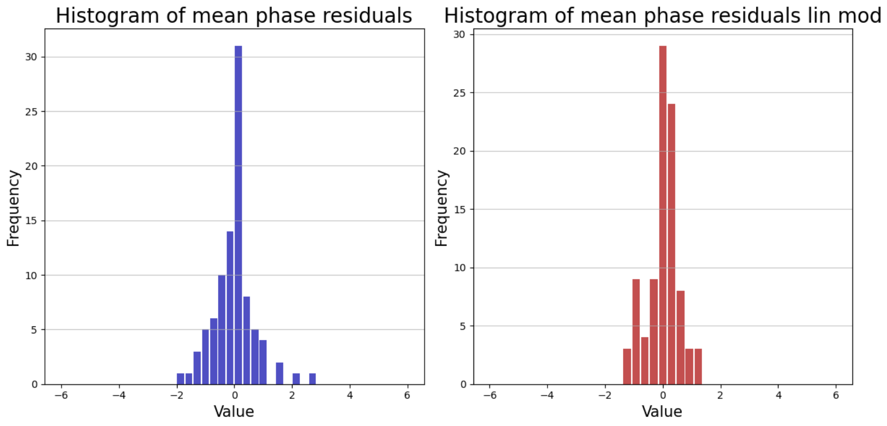


Figure 3.86: Histogram of mean residual for fountains, from August 2022 to October 2023, in the two cases: frequency standard and frequency ls.

There does not seem to be any particular improvement, for none of the clock types.
Nonetheless it has to be said that the selected clocks ensemble has been chosen to satisfy desirable properties (continuous measurements over 5 years) and this might have influenced the results.

3.2.3 Global effects on EAL time scale

In the previous sections of this chapter, we talked about the possibility of optimally predicting atomic clocks deviations from ideal, specifically for each type of clock.
We were able to determine with confidence that a linear model is indeed better at modeling fountains than the quadratic one.
For what concerns other models, in particular: linear model for cesium clocks, least squares frequency model for cesium and masers, there were still some doubts related to the choice of the clocks that we used for testing.
To try to better understand if these models could be an improvement we decided to look at how they would affect the frequency stability of the time scale if they were regularly implemented from 1st November 2021, the date in which the current maximum weight was adopted.
We always implement the linear model for fountains, while we vary the modeling of the other types of clocks. We did not make a frequency stability comparison of EAL between the original model and only the linear model for fountains, given that it is extremely hard to see any change on the global time scale by changing the prediction of only 5 clocks over 420.
To look at the frequency stability of the time scale we use $EAL - 1930005$, in fact given that 1930005 is an extremely precise clock, looking at $EAL - 1930005$ is

practically like looking at EAL itself. Frequency stability is computed by doing first order differences of phase error data and computing the Allan Deviation, which measures, for a given time interval τ, the average change in frequency. Details on the computation of Allan Deviation are found in the following chapter. The resulting plot is in figure 3.87.

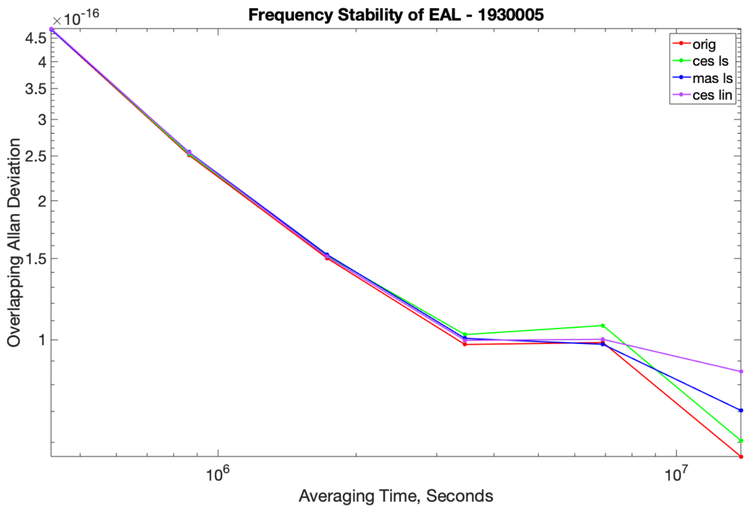


Figure 3.87: Frequency stability of EAL-1930005 for the four different models.

We conclude that there does not seem to be any improvement on the frequency stability of the scale from any of the models mentioned above. Motivation on that would need further investigation.

3.3 Implementation of stochastic model

I wanted to check the impact of adding to the deterministic model 1.54 the stochastic component 2.55, which could be easily simulated in Matlab as a bivariate Gaussian random variable.
While I knew $\tau = 432000$ s (points equally spaced by 5 days) I had to recover the values of the diffusion coefficients σ_1 and σ_2.
To do so, I used the relationship with Allan variance which was derived in paper [10], obtained by writing the Allan variance 2.20 in terms of $X_1(t)$:

$$\sigma_y^2(\tau) = \frac{1}{2\tau^2}\mathbf{E}\left[((X_1(t_{k+2}) - X_1(t_{k+1})) - (X_1(t_{k+1}) - X_1(t_k)))^2\right] \tag{3.14}$$

and making the necessary substitutions. By defining

- $\sigma_y^{WF}(\tau)$ standard Allan deviation value for τ interval where white noise on frequency is dominant.

- $\sigma_y^{RWF}(\tau)$ standard Allan deviation value for τ interval where random walk on frequency is dominant.

The relationship is the following:

$$\sigma_1 = \sigma_y^{WF}(\tau) * \sqrt{\tau} \tag{3.15}$$

$$\sigma_2 = \frac{\sigma_y^{WF}(\tau)\sqrt{3}}{\sqrt{\tau}} \tag{3.16}$$

First all I computed Allan deviation plot for cesium 1350332 (3.88), using function *allan_variance.m* in B using formula 2.24. What is done in practice is to plot the standard Allan deviation versus τ in a log-log plot, with τ varying typically between 5 and 640 days. With this plot the slopes of the broken lines reveal the underlying type of noise with the following schema:

Noise Type	Slope
White Phase	-1
Flicker Phase	-1
White Frequency	$-\frac{1}{2}$
Flicker Frequency	0
Random Walk Frequency	$\frac{1}{2}$
Drift	1

Table 3.1: Slopes of broken lines related to type of noise in a log-log plot of standard Allan deviation versu τ.

Before computing the Allan deviation, frequency values are multiplied by 10^{14} which is a famous constant in metrology for ns/day. We can see how the cesium clock is mostly affected by white frequency noise (varying as $\tau^{-1/2}$) and random walk frequency noise (varying as $\tau^{1/2}$).

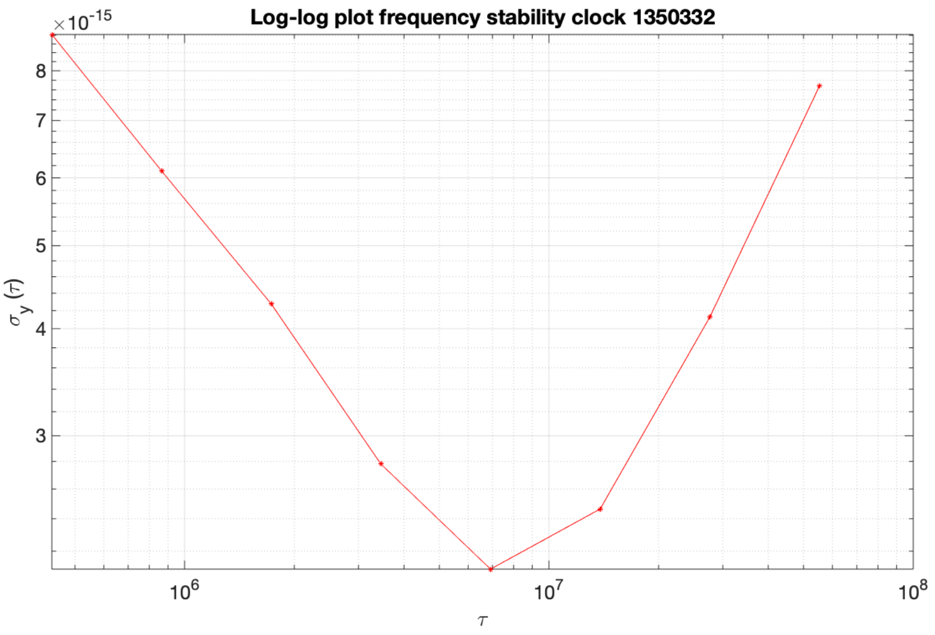


Figure 3.88: Allan deviation for cesium clock 1350332

I found values $\sigma_y^{WF}(\tau)$ and $\sigma_y^{RWF}(\tau)$ for $\tau = 1$ day for cesium 1350332 by looking at figure 3.88. and doing a projection of the broken lines at $\tau = 1$ day (since the minimal data spacing is 5 days, this value cannot be computed directly). The choice of $\tau = 1$ day is motivated by the fact that, to limit the propagation of numerical errors, the simulations are carried out with parameters in ns per day, and using $\tau = 1$ day permits some simplification.
For example in the case of clock 1350332, the calculation for σ_1 is the following: for $\tau = 432000$ (5 days) we know the Allan deviation value, so that we can write $\frac{\sqrt{h_0}}{\sqrt{2\tau}} = 8.82 * 10^{-15}$, from which it is possible to obtain $h_0 = 6.72 * 10^{-23}$. Having found h_0, $\sigma_y^{WF}(1\ \ day) = \frac{\sqrt{h_0}}{\sqrt{2*86400}} = 1.972 * 10^{-14}$. Finally, using the fact that numerically $\frac{ns}{day} = 10^{-14}$:

$$\sigma_1 = 1.972 * \frac{ns}{day} * \sqrt{day} = 1.972 \frac{ns}{\sqrt{day}}$$

Once σ_1 and σ_2 were obtained, I computed the residuals for 5 years from January 2018 to December 2022, using four simulations of 2.55 that I added to the deterministic part of the prediction term. Their plot is in figures 3.89 - 3.92. They can be compared to the standard, deterministic only, model in figure 3.93.

The code for the prediction with stochastic error term is found in B.12.

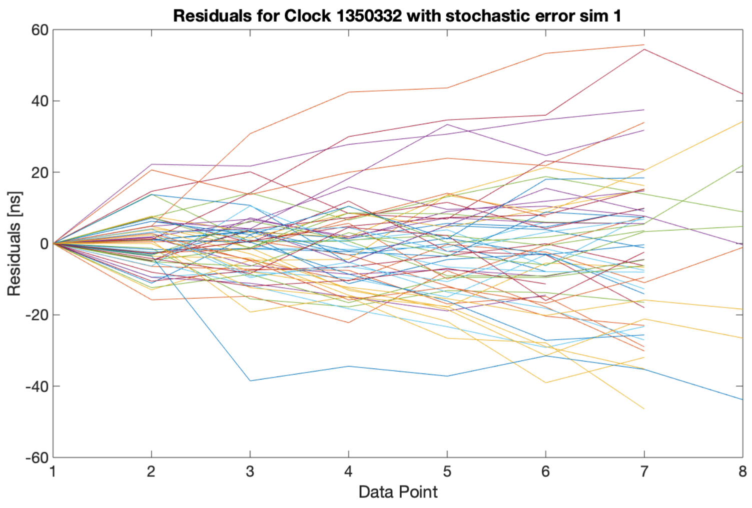


Figure 3.89: Residuals over 5 years for cesium 1350332 using deterministic plus stochastic prediction term.

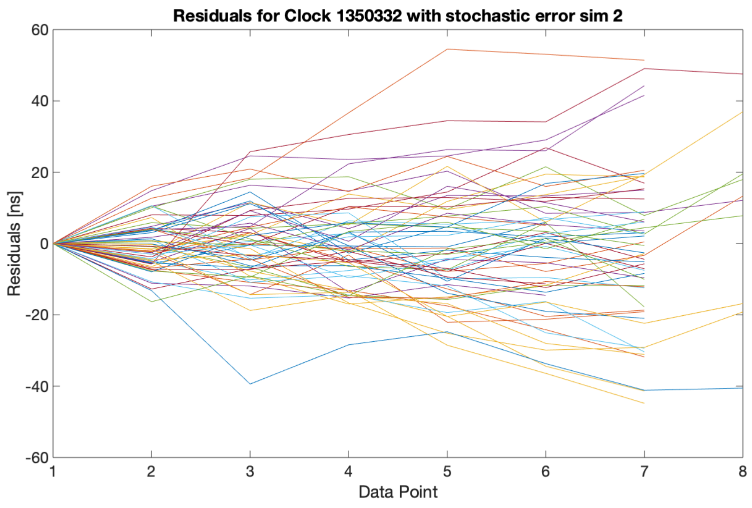


Figure 3.90: Residuals over 5 years for cesium 1350332 using deterministic plus stochastic prediction term.

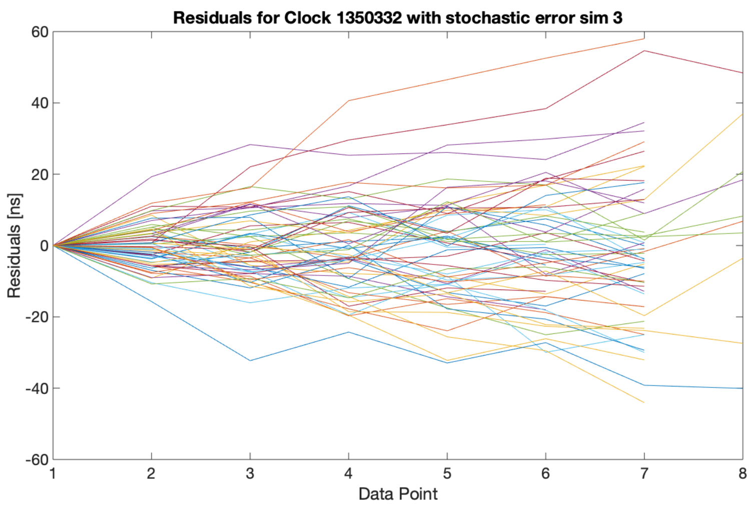


Figure 3.91: Residuals over 5 years for cesium 1350332 using deterministic plus stochastic prediction term.

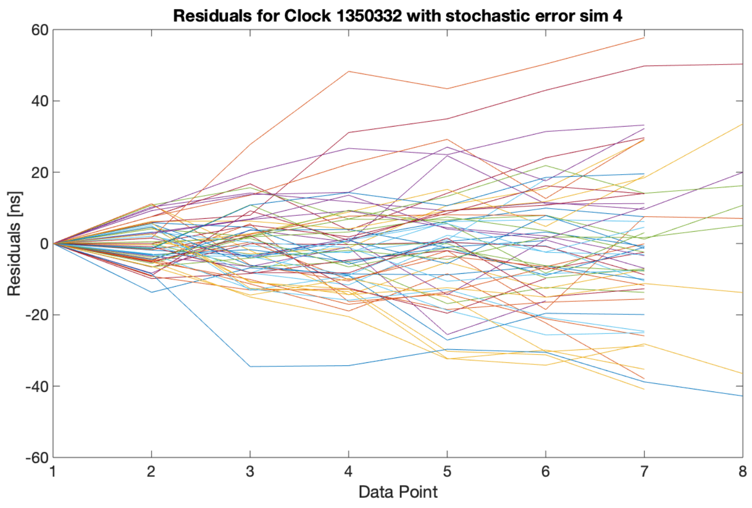


Figure 3.92: Residuals over 5 years for cesium 1350332 using deterministic plus stochastic prediction term.

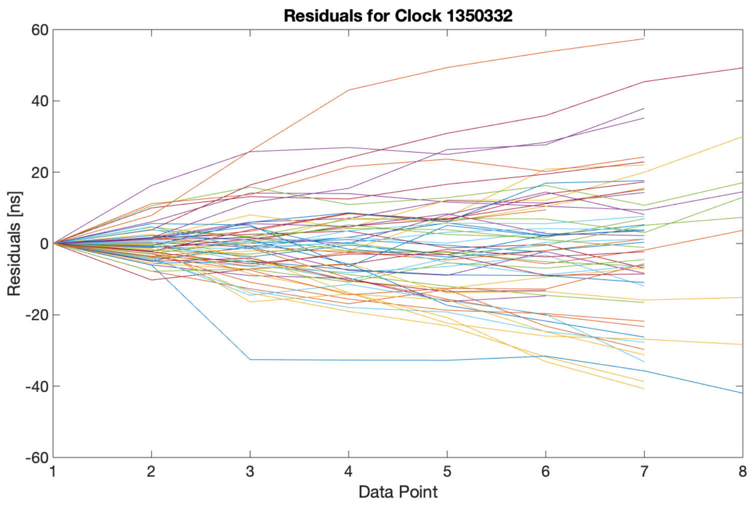


Figure 3.93: Residuals over 5 years for cesium 1350332 using deterministic prediction.

We can notice how, with the addition of the stochastic term, residuals are slightly better distributed around zero.

Conclusions

In summary, this thesis work has pieced together various aspects of time scale models, starting from the first implemented algorithm, to the current one and possible future enhancements.
The presented findings in the thesis help to conclude that the linear model should be the preferred choice for clocks of type fountain, and this conclusion was supported by physicists actively engaged in their construction.
Conversely, the investigation into the least squares frequency model did not yield any significant result, but we believe that it should be further explored, particularly in addressing cases with out-of-range phase error points within intervals, before concluding that it does not constitute an improvement.

Furthermore, this study underscores the importance of revising algorithms to account for non-constant drift, potentially through the adoption of a three-state stochastic model wherein drift varies linearly with time. Such models are already theorized in literature ([10]).

Lastly, from a comprehensive literature review it emerged the possibility of a global revision of the algorithm to ensure coherence within a statistical framework, possibly initiating with ordinary least squares (OLS) minimization supplemented by additional constraints to ensure the continuity of the time scale.

Appendix A

Calculations

A.1 Relationship between Allan Variance and Fourier Spectral Density.

$$\langle \sigma^2(N,\tau) \rangle = \frac{N}{N-1}\left(\sigma^2(\tau) - \sigma^2(N\tau)\right) \tag{A.1}$$

bu using 2.13:

$$\langle \sigma^2(N,\tau) \rangle = \frac{N}{N-1}\int_0^{+\infty} S_y(f)\left(\frac{sin(\pi\tau f)}{\pi\tau f}\right)^2\left[1-\left(\frac{sin(N\pi\tau f)}{N\pi\tau f}\right)\right] \tag{A.2}$$

making $N = 2$ gives directly

$$\sigma_y^2(\tau) = \int_0^{+\infty} S_y(f)\frac{2sin^4(\pi\tau f)}{(\pi\tau f)^2}df \tag{A.3}$$

A.2 Distribution of $\int_0^t W_s ds$.

From version II of Itô's Lemma with $f(t, W_t) = tW_t$, and $s = 0$, we get:

$$tW_t = \int_0^t W_s ds + \int_0^t s dW_s$$

$$\int_0^t W_s ds = tW_t - \int_0^t s dW_s = \int_0^t (t-s) dW_s$$

Given that the Itô stochastic integral has expectation 0 and we can compute its variance as

$$E\left[\int_0^t (t-s)dW_s\right]^2 = \int_0^t E\left[(t-s)^2\right]ds = \int_0^t (t-s)^2 ds = \frac{t^3}{3}$$

Appendix B

Matlab Code

B.1 Computation of pivoted table.

```
function pivotedTable = pivot_table(fileName)
%INPUT: a text file, in particular a file in the format TTHyyaa of
     4 columns where the first column is MJD, the second is the
    clock code, the third is the value of difference between the
    clock reading and TT. The fourth column is a column of only
    ones that gets eliminated in the execution of the function.
%OUTPUT: the pivoted table in which every line corresponds to a
    clock index and all the columns contain measurements of the
    reading difference betweeen that clock and TT for different
    times. When the measurement for a given clock is not available
    on a specific date we visualize a NaN value.

fid = fopen(fileName, 'r'); %fid is a fileID

% Read the data using textscan
data_cell = textscan(fid, '%f%f%f%f', 'Delimiter', ''); %reads
    data from an open text file
%into a cell array, the fact is that I get a nested cell array
%and this might be complicated when using cell2table. I choose
    declare
%every values as a double because it makes everything easier for
    the
%calculation.

 % Close the file
fclose(fid);

sz = size(data_cell{1,1},1) ;          %Line count
out=cell(sz,size(data_cell,2));        %I want a single cell array
    and not a nested cell array
```

```matlab
for k = 1:size(data_cell,2)
    t1 = data_cell(k); %accessing a subarray of a cell array which in our case is 1x1 cell array that contains a 8101 x 1 cell array
    t2 = [t1{:}];
    if isnumeric(t2)              % Takes care of floats
        out(:,k) = num2cell(t2);
    else
        out(:,k) = t2;
    end
end


% Convert the numeric data to a table
data = cell2table(out, 'VariableNames', {'MJD', 'ClockCode', 'ReadingDifference', 'Var4'});
data(:, end) = []; %I am leaving out the last column since it is made of only ones
[~, uniqueIdx, ~] = unique(data(:, {'MJD', 'ClockCode'}), 'rows', 'last'); %to delete rows that share
%same clock code and date. In particular I want to keep only the last
%occurrence of a measurement for a specific clock in a specific date
data = data(uniqueIdx, :);
data = sortrows(data, 'MJD'); %to order the rows based on the date


unique_clocks = unique(data.ClockCode);
unique_dates = unique(data.MJD);

length(unique_clocks) %412 clocks
length(unique_dates) %20 dates



varTypes = repmat({'double'}, 1, length(unique_dates)+1);
varNames = ['ClockCode', transpose(string(unique_dates))];


% Initialize a new table to store the pivoted data
pivotedTable = table('Size', [length(unique_clocks), length(unique_dates) + 1], 'VariableNames', varNames , 'VariableTypes', varTypes);


% Populate the pivoted table
for i = 1:length(unique_clocks)
    current_clock = unique_clocks(i);
```

```

     for k = 1:length(unique_dates)
         current_date = unique_dates(k);

         % Select rows corresponding to the current clock code
         filtered_rows = data(data.ClockCode == current_clock &
    data.MJD == current_date, :);

         % Assign values to the pivoted table
         pivotedTable{i, 1} = current_clock; % First column is the
    code of the clock

         if isempty(filtered_rows)
             pivotedTable{i, k+1} = NaN; % k+1 because the first
    column is for clock code
         else
             pivotedTable{i, k+1} = filtered_rows.ReadingDifference
    ;
         end
     end
end
end
```

B.2 Computation of cleaned table.

```
function [clean_data,good_clocks] = NaN_removal(pivoted_table)
%INPUT: a table
%OUTPUT: the same table where every row that contains NaN values
    is deleted.

rows_with_nan = any(isnan(pivoted_table{:,:}), 2);
clean_data = pivoted_table(~rows_with_nan, :);
good_clocks = clean_data.ClockCode;
end
```

B.3 Computation of freq table.

```
function frequencies_table = freq(data)
%INPUT: a table of data which should contain
```

```
%in each row a clock index and in all the columns the measurements
    for that clock over a number of years. This can be achieved
   through the use of the  function: 'pivot_table' on a text file
   of clock data. We suppose that the clock index is always in the
    first column. The data must be cleaned so that there are no
   NaN values and this can be achieved through the use of function
   : 'NaN_removal'
%OUTPUT: a table which contains for every clock the frequencies
   data over the time period (N measurements -> N-1 data)

% Extract numeric values from the table
numeric_data = table2array(data(:, 2:end));
variables = data.Properties.VariableNames;
dates = variables(2:end);
numeric_dates = str2double(dates);


% Calculate differences along the columns (dimension 2)
differences = diff(numeric_data, 1, 2);
%
mjd_differences = diff(numeric_dates,1,2);

ratios = zeros(size(data,1),size(differences,2));
for i=1:size(data,1)
ratios(i,:)  = differences(i,:) ./ mjd_differences;
end

varNames1 = compose('freq_diff%d', 1:size(ratios,2));
frequencies_table = array2table (ratios, 'VariableNames',
   varNames1);
frequencies_table.CodeClock = data.ClockCode;
code_column_index = find(strcmp(frequencies_table.Properties.
   VariableNames, 'CodeClock'));

% Rearrange the columns
new_order = [code_column_index, 1:code_column_index-1,
   code_column_index+1:size(frequencies_table, 2)];
frequencies_table = frequencies_table(:, new_order);


end
```

B.4 Computation of linear interpolation and plot.

```
function [coeff,resid] = lq1_interp(freq_table)
```

```matlab
%INPUT: a matlab table containing the frequencies values for each clock or for a preselected ensemble of clocks over a given period of time (a list of 5 days spaced MJDs)
%OUTPUT: the results of the first order polynomial interpolation with least squares technique. In particular:
    %coeff: output of the polyfit function
    %resid: residual table which contains for every clock and every
    %measurement y_i the residual vector (y_i - y_interp_i)_i and
    %also the mean of residuals for the clock and the std

resid = zeros(size(freq_table,1),size(freq_table,2)+2);
count = 0;
for i=1:size(freq_table,1)
    x = 1:(size(freq_table,2)-1);
    y = freq_table{i,2:end};
    coeff = polyfit(x,y,1);
    x_interp = linspace(1,max(x), 100); %calcolo la retta interpolante su 100 punti
    y_interp = polyval(coeff, x_interp);
    resid(i,1) = freq_table{i,1};
    resid(i,2:end-2) = y - polyval(coeff, x);
    resid(i,end-1) = round(mean(resid(i,2:end-2)));
    resid(i,end) = std(resid(i,2:end-2));
    figure;
    subplot(2, 2, 1);
    plot(x, y);
    title(sprintf('Frequency Plot for Clock %d', freq_table{i,1}));
    hold on;
    plot(x_interp, y_interp, 'r-', 'DisplayName', 'Interpolation');
    xlabel('N.of Observation');
    ylabel('Difference values');
    legend('show');
    hold off;
    subplot(2, 2, 2);
    plot(x, resid(i,2:end-2), 'o');
    title('Residuals');
    hold on;
    plot([min(x), max(x)], [0, 0], 'r--', 'LineWidth', 2);
    hold off;
    subplot(2, 2, 3);
    histogram(resid(i,2:end-2));
    title('Histogram of residuals');
    subplot(2, 2, 4);
    qqplot(resid(i,2:end-2));
    title('QQ plot of residuals')
    destinationFolder = '/Users/ariannaabis/Desktop/Tesi_Matlab/';
    filename = fullfile(destinationFolder,sprintf('%d_18_22.fig',
   freq_table{i,1} ));
    saveas(gcf, filename);
    [h_jb, p_jb] = jbtest(resid(i,2:end-2));
    message = sprintf('Jarque-Bera test p-value for clock %s: %s',
    freq_table{i,1}, num2str(p_jb));
    disp(message);
    if h_jb == 0
    count = count+1;
    disp('The null hypothesis (normality) is not rejected.');
    else
    disp('The null hypothesis (normality) is rejected.');
    end


end
varNames = ['ClockCode', compose('res%d', 1:size(freq_table,2)-1),
    'mean', 'std'];
resid = array2table(resid, 'VariableNames', varNames);

disp(sprintf('The normality test was accepted in the %d%% of cases
   ', (count/size(freq_table,1))*100));
end
```

B.5 Script for five years analysis.

```
%The purpose of this script is to test all the code process and
   functions on all the data coming from the txt file '
   clock_tt_total', which stores information for all the clocks
   over five years from 2018 to 2022 (whihc implies it stores data
    from november 2017 to december 2022, given that the data in
   THHYYMM actually stores measurements for the 5 days spaced
   intervals of MM and the two months preceding).

%tic
pivoted_18_22 = pivot_table('clock_tt_total');%function pivot
   reads a text file, it transforms it into a table and it inverts
    is so that in each row we have a clock and in the columns the
   measurements for 5 years at 5 days intervals
clean_18_22 = NaN_removal(pivoted_18_22); %function NaN_removal
   gets rid of all the clocks that have NaN values in any date
writetable(clean_18_22, 'clean_18_22.xlsx'); %we save the readings
    data for the clocks with all the measurements
good_clocks_18_22 = clean_18_22.ClockCode; %list of selected
   clocks
```

```
writematrix(good_clocks_18_22, 'good_clocks_18_22.txt');
% Extract clock codes based on starting digits
masters_18_22 = good_clocks_18_22(floor(good_clocks_18_22 / 100000)
     == 14);
fountains_18_22 = good_clocks_18_22(floor(good_clocks_18_22 /
    100000) == 19);
cesium_18_22 = good_clocks_18_22(floor(good_clocks_18_22 / 100000)
     == 13);

%Compute frequency values
freq_18_22_masers = freq(clean_18_22(ismember(clean_18_22.
    ClockCode, masers_18_22), :));
freq_18_22_fountains = freq(clean_18_22(ismember(clean_18_22.
    ClockCode, fountains_18_22), :));
freq_18_22_cesium = freq(clean_18_22(ismember(clean_18_22.
    ClockCode, cesium_18_22), :));
%toc
[coeff_fountains_18_22, resid_fountains_18_22] = lq1_interp(
    freq_18_22_fountains);
[coeff_cesium_18_22, resid_cesium_18_22] = lq1_interp(
    freq_18_22_cesium);
[coeff_masers_18_22, resid_masers_18_22] = lq1_interp(
    freq_18_22_masers);
end
```

B.6 Quadratic model for phase error, frequency computed using both techniques: first - last point of time difference and linear ls interpolation.

```
%In questo script:
%1. scelgo un sottoinsieme di orologi
%2. per ogni orologio:
%estrapolo tutti i batch di 3 mesi a partire da Nov2017 per ogni
    batch stimo il drift, mi sposto sul mese successivo (ultimo
    mese del batch successivo), calcolo frequenza come ultimo -
    primo/30 sul mese precedente e poi con i minimi quadrati sul
    mese precedente calcolo prevision come = ultimo punto mese
    scorso + frequenza*t + 1/2*drift*t^2

close all;

% Open the file 'clock_tt_total'
```

```
fid_tt = fopen('clock_tt_total', 'r');
fid_eal = fopen('clock_eal_total','r');

% Read the data using textscan
data_cell_tt = textscan(fid_tt, '%f%f%f%f', 'Delimiter', '');
data_cell_eal = textscan(fid_eal, '%f%f%f%f%f', 'Delimiter', '');
%reads data from an open text file into a cell array,
%in fact I get as an output a nested cell array
%and this might be complicated when using cell2table

 % Close the file
fclose(fid_tt);
fclose(fid_eal);

%I want to transform my nested cell array in a single cell array

sz = size(data_cell_tt{1,1},1) ;          %Line count
out=cell(sz,size(data_cell_tt,2));        %I want a single cell array
for k = 1:size(data_cell_tt,2)
    t1 = data_cell_tt(k); %accessing a subarray of a cell array which in our case is 1x1 cell array that contains another cell array
    t2 = [t1{:}];
    if isnumeric(t2)                 % Takes care of floats
        out(:,k) = num2cell(t2);
    else
        out(:,k) = t2;
    end
end

sz = size(data_cell_eal{1,1},1) ;          %Line count
out1=cell(sz,size(data_cell_eal,2));        %I want a single cell array
for k = 1:size(data_cell_eal,2)
    t1 = data_cell_eal(k); %accessing a subarray of a cell array which in our case is 1x1 cell array that contains another cell array
    t2 = [t1{:}];
    if isnumeric(t2)                 % Takes care of floats
        out1(:,k) = num2cell(t2);
    else
        out1(:,k) = t2;
    end
end


% Convert the numeric data to a table
```

```
data_tt = cell2table(out, 'VariableNames', {'MJD', 'ClockCode', 'ReadingDifference', 'Var4'});
data_tt(:, end) = [];

data_eal = cell2table(out1, 'VariableNames', {'MJD', 'ClockCode', 'Var4', 'Var5', 'ReadingDifference'});
data_eal = removevars(data_eal, {'Var4', 'Var5'});

%fountains = [1930002, 1930003, 1930004, 1930005];
%masers=[1400702, 1400222, 1400296, 1403853];
%cesium=[1350102, 1350161, 1350179, 1350332];
selected_clocks = masers_18_22; % replace with your actual clock codes
filtered_data_tt = data_tt(ismember(data_tt.ClockCode, selected_clocks), :);
filtered_data_eal = data_eal(ismember(data_eal.ClockCode, selected_clocks), :);

unique_clocks = unique(filtered_data_tt.ClockCode);
% Find the indices of selected_clocks in unique_clocks
[~, indices] = ismember(selected_clocks, unique_clocks);

% Order unique_clocks based on the indices
unique_clocks = unique_clocks(indices);
clock_tables_tt = cell(size(unique_clocks));
clock_tables_eal = cell(size(unique_clocks));
subset_tables_tt = cell(size(unique_clocks,1),60); %every clock -> 60 subtables
subset_tables_eal = cell(size(unique_clocks,1),60);

for i = 1:numel(unique_clocks)
    current_clock = unique_clocks(i);
    clock_tables_tt{i} = filtered_data_tt(filtered_data_tt.ClockCode == current_clock, :);
    clock_tables_eal{i} = filtered_data_eal(filtered_data_eal.ClockCode == current_clock, :);
    %disp(current_clock);
end

% Assuming clock_tables is the cell array containing tables for each clock

%I want, for each clock in a selected ensemble, to get as many subtables as there are three months butches of data (each batch overlaps on the preceding for two months.)
for i = 1:numel(clock_tables_tt)
    current_table_tt = clock_tables_tt{i};
    current_table_eal = clock_tables_eal{i};
    index = 1;
```

```matlab
    initial_MJD = current_table_tt.MJD(index);

    k = 1; % Reset k for each clock

    while index <= height(current_table_tt)
        % Process the subset of the table
        subset_tables_tt{i, k}(index,:) = current_table_tt(index, :);
        subset_tables_eal{i, k}(index,:) = current_table_eal(index, :);
        nonZeroRows_tt = any(table2array(subset_tables_tt{i,k}), 2);
        nonZeroRows_eal = any(table2array(subset_tables_eal{i,k}), 2);
        subset_tables_tt{i,k} = subset_tables_tt{i,k}(nonZeroRows_tt, :);
        subset_tables_eal{i,k} = subset_tables_eal{i,k}(nonZeroRows_eal, :);
        index = index + 1;
        %disp(index);

        if index <= height(current_table_tt) && current_table_tt.MJD(index) >= initial_MJD
            initial_MJD = current_table_tt.MJD(index);
            %disp(k);
        else
            if k < 60
            k = k+1;
            initial_MJD = current_table_tt.MJD(index);
            %disp(k);
            %disp(initial_MJD);
            else
                break;
            end
        end
    end
end

%FREQUENZA ULTIMO - PRIMO

resid =cell(length(selected_clocks),59);
for i = 1:length(selected_clocks)
disp(selected_clocks(i));
    previous_month = subset_tables_eal{i,1}(ismember(subset_tables_eal{i,1}.MJD,[58114:5:58149]),:).ReadingDifference;
    t0_prev = 58114;
    %disp(previous_month);
    %disp(length(previous_month));
```

```
        for k=1:59
        freq_curr = diff(subset_tables_tt{i,k}.ReadingDifference)
    /5; %I compute frequencies for the 'current' three months
        x_curr = subset_tables_tt{i,k}.MJD(2):5:subset_tables_tt{i
    ,k}.MJD(end); %I make it as if my first frequency data is
    computed in the second MJD of the first of three months
        y_curr = movmean(transpose(freq_curr),3); %making freq a
    row vector and computing moving average
        coeff = polyfit(x_curr,y_curr,1); %finding y_0 and d such
    that y(t) = y_0 + d*t, coeff(1) = d, coeff(2) = y_o
        d = coeff(1);
        x_new = x_curr(end):5:subset_tables_tt{i,k+1}.MJD(end); %
    MJD for next month
        new_month_values = subset_tables_eal{i,k+1}(
    subset_tables_eal{i,k+1}.MJD >= x_curr(end), :).
    ReadingDifference;
        %disp(new_month_values);
        x0 = new_month_values(1);
        %get measured again and we want 0 residual in the first
    point we impose it like last point of last month
        y0 = (x0 - previous_month(1))/((length(previous_month)*5)
    -5);
        x = zeros(1,length(new_month_values));
        x(1)=x0;
        for j=2:length(new_month_values)
        x(j) = x0 + y0.*(x_new(j)-x_curr(end)) + (1/2).*d.*((x_new
    (j)-x_curr(end)).^2)+(1/2).*d.*(x_new(1)-t0_prev).*(x_new(j)-
    x_curr(end));
        end
        resid{i,k} =  x - new_month_values';
        %ensamble of length(new_month_values) values
        previous_month = new_month_values;
        t0_prev = x_new(1);
        end
end

%SALVO LA TABELLA

myTable = cell2table(resid);

% % Specifica il nome del file Excel
 destinationFolder = '/Users/ariannaabis/Desktop';
 excelFileName = fullfile(destinationFolder,'
    resid_masers_18_22_eal.xlsx');
%
% % Scrivi la tabella nel file Excel
 writetable(myTable, excelFileName);

%PLOT PER OROLOGIO
```

```
for i=1:length(selected_clocks)
    LegendEntries = cell(1,59);
    figure;
    for k=1:59
        LegendEntries{k} = sprintf('Batch %d', k);
        plot(resid{i,k}) %option to only see the first six residuals for easier visualization
        hold on;
    end
    legend(LegendEntries);
    title(sprintf('Residuals for Clock %d', selected_clocks(i)));
    xlabel('Data Point');
    ylabel('Residuals');
    hold off;

%     % Salva automaticamente il plot
%     destinationFolder = '/Users/ariannaabis/Desktop/Tesi_Matlab/selected_fountains_2/resid_per_clock/freq_standard';
%     filename = fullfile(destinationFolder,sprintf('%d_resid_phase.fig', selected_clocks(i)));
%     saveas(gcf, filename);
end

%PLOT PER MESE
figure;
for k=1:59
    LegendEntries = cell(1,length(selected_clocks));
    figure;
    for i=1:length(selected_clocks)
        LegendEntries{i} = sprintf('Clock %d', selected_clocks(i));
        plot(resid{i,k}) %option to only see the first six residuals for easier visualization
        hold on;
    end
    legend(LegendEntries);
    title(sprintf('Residuals for Batch %d', k));
    xlabel('Data Point');
    ylabel('Residuals');
    hold off;

%     % Salva automaticamente il plot
%     destinationFolder = '/Users/ariannaabis/Desktop/Tesi_Matlab/selected_fountains_2/resid_per_month/freq_standard';
%     filename = fullfile(destinationFolder,sprintf('batch_%d_fountains.fig', k));
%     saveas(gcf, filename);
end

```

```
%FREQ LS

resid =cell(length(selected_clocks),59);
%for i = 1:length(selected_clocks)
i=2;
    previous_month = subset_tables_tt{i,1}(ismember(subset_tables_tt{i,1}.MJD,[58114:5:58149]),:).MJD;
    t0_prev = 58114;
        for k=1:59
        freq_curr = diff(subset_tables_tt{i,k}.ReadingDifference)/5; %I compute frequencies for the 'current' three months
        x_curr = subset_tables_tt{i,k}.MJD(2):5:subset_tables_tt{i,k}.MJD(end); %I make it as if my first frequency data is computed in the second MJD of the first of three months
        y_curr = movmean(transpose(freq_curr),3); %making freq a row vector
        coeff = polyfit(x_curr,y_curr,1); %finding y_0 and d such that y(t) = y_0 + d*t, coeff(1) = d, coeff(2) = y_o
        d = coeff(1);
        x_new = x_curr(end):5:subset_tables_eal{i,k+1}.MJD(end); % MJD for next month
        new_month_values = subset_tables_eal{i,k+1}(subset_tables_eal{i,k+1}.MJD >= x_curr(end), :).ReadingDifference;
        x0 = new_month_values(1);
        x_preceding_month = fliplr(x_curr(end):-5:previous_month(1));
        values_preceding_month = subset_tables_eal{i,k}(ismember(subset_tables_eal{i,k}.MJD,x_preceding_month),:).ReadingDifference;
        coeff1 = polyfit(x_preceding_month,values_preceding_month',1);
        y0 = coeff1(1);
        x = zeros(1,length(new_month_values));
        x(1) = x0;
        for j=2:length(new_month_values)
        x(j) = x0 + y0.*(x_new(j)-x_curr(end)) + (1/2).*d.*((x_new(j)-x_curr(end)).^2)+(1/2).*d.*(x_new(1)-t0_prev).*(x_new(j)-x_curr(end));
        end
        resid{i,k} =  x - new_month_values';
        %ensamble of length(new_month_values) values
        previous_month = x_new;
        t0_prev = x_new(1);
        end
%end

myTable = cell2table(resid);

```

```matlab
% Specifica il nome del file Excel
destinationFolder = '/Users/ariannaabis/Desktop';
excelFileName = fullfile(destinationFolder,'resid_masers_18_22_eal_ls.xlsx');

% Scrivi la tabella nel file Excel
writetable(myTable, excelFileName);

%PLOT PER CLOCK

for i=1:length(selected_clocks)
    LegendEntries = cell(1,59);
    figure;
    for k=1:59
        LegendEntries{k} = sprintf('Batch %d', k);
        plot(resid{i,k}) %option to only see the first six residuals for easier visualization
        hold on;
    end
    legend(LegendEntries);
    title(sprintf('Residuals for Clock %d and ls estimation of freq', selected_clocks(i)));
    xlabel('Data Point');
    ylabel('Residuals');
    hold off;
%     destinationFolder = '/Users/ariannaabis/Desktop/Tesi_Matlab/selected_fountains_2/resid_per_clock/freq_ls';
%     filename = fullfile(destinationFolder,sprintf('%d_resid_phase_ls.fig', selected_clocks(i)));
%     saveas(gcf, filename);
end

%PLOT PER MONTH

figure;
for k=1:59
    LegendEntries = cell(1,length(selected_clocks));
    figure;
    for i=1:length(selected_clocks)
        LegendEntries{i} = sprintf('Clock %d', selected_clocks(i));
        plot(resid{i,k})
        hold on;
    end
    legend(LegendEntries);
    title(sprintf('Residuals for Batch %d', k));
    xlabel('Data Point');
    ylabel('Residuals');
    hold off;

    % Salva automaticamente il plot
%     destinationFolder = '/Users/ariannaabis/Desktop/Tesi_Matlab/selected_fountains_2/resid_per_month/freq_ls';
%     filename = fullfile(destinationFolder,sprintf('batch_%d_fountains_ls.fig', k));
%     saveas(gcf, filename);
end
```

B.7 Linear model for phase error for fountains: frequency computed with least - first point of time differences.

```
 close all;

% Open the file 'clock_tt_total'
fid_tt = fopen('clock_tt_total', 'r');
fid_eal = fopen('clock_eal_total','r');

% Read the data using textscan
data_cell_tt = textscan(fid_tt, '%f%f%f%f', 'Delimiter', '');
data_cell_eal = textscan(fid_eal, '%f%f%f%f%f', 'Delimiter', '');
%reads data from an open text file into a cell array,
%in fact I get as an output a nested cell array
%and this might be complicated when using cell2table

 % Close the file
fclose(fid_tt);
fclose(fid_eal);

%I want to transform my nested cell array in a single cell array

sz = size(data_cell_tt{1,1},1) ;          %Line count
out=cell(sz,size(data_cell_tt,2));        %I want a single cell array
for k = 1:size(data_cell_tt,2)
    t1 = data_cell_tt(k); %accessing a subarray of a cell array which in our case is 1x1 cell array that contains another cell array
    t2 = [t1{:}];
    if isnumeric(t2)                 % Takes care of floats
        out(:,k) = num2cell(t2);
    else
        out(:,k) = t2;
    end
end

sz = size(data_cell_eal{1,1},1) ;          %Line count
out1=cell(sz,size(data_cell_eal,2));        %I want a single cell
    array
for k = 1:size(data_cell_eal,2)
    t1 = data_cell_eal(k); %accessing a subarray of a cell array
    which in our case is 1x1 cell array that contains another cell
    array
    t2 = [t1{:}];
    if isnumeric(t2)                 % Takes care of floats
        out1(:,k) = num2cell(t2);
    else
        out1(:,k) = t2;
    end
end


% Convert the numeric data to a table
data_tt = cell2table(out, 'VariableNames', {'MJD', 'ClockCode', '
    ReadingDifference', 'Var4'});
data_tt(:, end) = [];

data_eal = cell2table(out1, 'VariableNames', {'MJD', 'ClockCode',
    'Var4', 'Var5', 'ReadingDifference'});
data_eal = removevars(data_eal, {'Var4', 'Var5'});

%fountains = [1930002, 1930003, 1930004, 1930005];
%masers=[1400702, 1400222, 1400296, 1403853];
%cesium=[1350102, 1350161, 1350179, 1350332];
selected_clocks = fountains_18_22; % replace with your actual
    clock codes
filtered_data_tt = data_tt(ismember(data_tt.ClockCode,
    selected_clocks), :);
filtered_data_eal = data_eal(ismember(data_eal.ClockCode,
    selected_clocks), :);

unique_clocks = unique(filtered_data_tt.ClockCode);
% Find the indices of selected_clocks in unique_clocks
[~, indices] = ismember(selected_clocks, unique_clocks);

% Order unique_clocks based on the indices
unique_clocks = unique_clocks(indices);
clock_tables_tt = cell(size(unique_clocks));
clock_tables_eal = cell(size(unique_clocks));
subset_tables_tt = cell(size(unique_clocks,1),60); %every clock ->
     60 subtables
subset_tables_eal = cell(size(unique_clocks,1),60);
```

```matlab

for i = 1:numel(unique_clocks)
    current_clock = unique_clocks(i);
    clock_tables_tt{i} = filtered_data_tt(filtered_data_tt.ClockCode == current_clock, :);
    clock_tables_eal{i} = filtered_data_eal(filtered_data_eal.ClockCode == current_clock, :);
    %disp(current_clock);
end

% Assuming clock_tables is the cell array containing tables for each clock

%I want, for each clock in a selected ensemble, to get as many subtables as there are three months butches of data (each batch overlaps on the
preceding for two months.)
for i = 1:numel(clock_tables_tt)
    current_table_tt = clock_tables_tt{i};
    current_table_eal = clock_tables_eal{i};
    index = 1;
    initial_MJD = current_table_tt.MJD(index);

    k = 1; % Reset k for each clock

    while index <= height(current_table_tt)
        % Process the subset of the table
        subset_tables_tt{i, k}(index,:) = current_table_tt(index, :);
        subset_tables_eal{i, k}(index,:) = current_table_eal(index, :);
        nonZeroRows_tt = any(table2array(subset_tables_tt{i,k}), 2);
        nonZeroRows_eal = any(table2array(subset_tables_eal{i,k}), 2);
        subset_tables_tt{i,k} = subset_tables_tt{i,k}(nonZeroRows_tt, :);
        subset_tables_eal{i,k} = subset_tables_eal{i,k}(nonZeroRows_eal, :);
        index = index + 1;
        %disp(index);

        if index <= height(current_table_tt) && current_table_tt.MJD(index) >= initial_MJD
            initial_MJD = current_table_tt.MJD(index);
            %disp(k);
        else
            if k < 60
            k = k+1;
            initial_MJD = current_table_tt.MJD(index);
            %disp(k);
            %disp(initial_MJD);
            else
                break;
            end
        end
    end
end

%FREQUENZA ULTIMO - PRIMO

resid =cell(length(selected_clocks),59);
for i = 1:length(selected_clocks)
%i=1;
    previous_month = subset_tables_eal{i,1}(ismember(subset_tables_eal{i,1}.MJD,[58114:5:58149]),:).ReadingDifference;
        for k=1:59
        %past_month = fliplr(subset_tables{i,k}.MJD(end):-5:(subset_tables{i,k}.MJD(end)-30));
        current_month = subset_tables_eal{i,k}.MJD(end):5:subset_tables_eal{i,k+1}.MJD(end);
        x0 = subset_tables_eal{i,k+1}(subset_tables_eal{i,k+1}.MJD == current_month(1), :).ReadingDifference;
        %y = ((subset_tables{i,k}(subset_tables{i,k}.MJD == past_month(end), :).ReadingDifference) - (subset_tables{i,k}(subset_tables{i,k}.MJD == past_month(1), :).ReadingDifference))/30;
        y = (previous_month(end)-previous_month(1))/((length(previous_month)*5)-5);
        x_current_month = subset_tables_eal{i,k+1}(ismember(subset_tables_eal{i,k+1}.MJD,current_month),:).ReadingDifference;
        x_predicted = zeros(1,length(x_current_month));
        for t=1:length(x_current_month)
            x_predicted(t) = x0 + y*(current_month(t)-current_month(1));

        end
        resid{i,k} = x_predicted - x_current_month';
        previous_month = x_current_month;
        end
end

myTable = cell2table(resid);

% Specifica il nome del file Excel
destinationFolder = '/Users/ariannaabis/Desktop';
```

```
excelFileName = fullfile(destinationFolder,'resid_fountains_18_22_eal_lin.xlsx');

% Scrivi la tabella nel file Excel
writetable(myTable, excelFileName);


for i=1:length(selected_clocks)
    LegendEntries = cell(1,59);
    figure;
    for k=1:59
        LegendEntries{k} = sprintf('Batch %d', k);
        plot(resid{i,k}) %option to only see the first six residuals for easier visualization
        hold on;
    end
    legend(LegendEntries);
    title(sprintf('Residuals for Clock %d for the linear model', selected_clocks(i)));
    xlabel('Data Point');
    ylabel('Residuals');
    hold off;

    % Salva automaticamente il plot

   destinationFolder = '/Users/ariannaabis/Desktop/Tesi_Matlab/selected_fountains_2/resid_per_clock/lin_mod';
   filename = fullfile(destinationFolder,sprintf('%d_resid_phase_lin.fig', selected_clocks(i)));
   saveas(gcf, filename);
end

figure;
for k=1:59
    LegendEntries = cell(1,length(selected_clocks));
    figure;
    for i=1:length(selected_clocks)
        LegendEntries{i} = sprintf('Clock %d', selected_clocks(i));
        plot(resid{i,k}) %option to only see the first six residuals for easier visualization
        hold on;
    end
    legend(LegendEntries);
    title(sprintf('Residuals for Batch %d', k));
    xlabel('Data Point');
    ylabel('Residuals');
    hold off;

    % Salva automaticamente il plot
    destinationFolder = '/Users/ariannaabis/Desktop/Tesi_Matlab/selected_fountains_2/resid_per_month/lin_mod';
    filename = fullfile(destinationFolder,sprintf('batch_%d_fountains_lin.fig', k));
    saveas(gcf, filename);
end
```

B.8 Computation of Allan Variance.

```
function z1 = olavarf01(dataTTf,n,tau0)
%dataTTf: frequency data recordings MxN where I guess N number of clocks
%tau0: minimal data spacing (I guess tau0 = 5)
%n: moltiplicative factor for enlarging the sample time
M = size(dataTTf,1); %number of frequency point
x = nan(M+1,size(dataTTf,2)); %preallocate, x contains a value for each recording
tau = tau0*ones(M,1);
for j = 1:size(dataTTf,2)
    x(1,j) = 0;
    x(2:M+1,j) = cumsum(dataTTf(:,j).*tau);
    %prendi la j-esima colonna di dataTTf, moltiplica ogni elemento per tau
    %e poi fai la comulativa ->ottieni ultimo
end

tau = n*tau0; %stiamo cambiando il sampling time da tau0 a tau=m*tau0
tot = M+1-2*n;
z1 = zeros(1,size(dataTTf,2));
if (tot < 1)
    z1 = nan;
else
    for i=1:tot
    zt = x(i+2*n,:) - 2*x(i+n,:) + x(i,:);
    disp(size(zt));
    z1 = zt.^2/(2*tau^2*tot) +z1;
    disp(z1);
    end
end %
```

B.9 Script for computation of Allan Variance.

```matlab
close all;


%First I create an input matrix to give to function olavarf.m
%I want to compute Allan variance for a subset of clocks

% Extract relevant columns
clockCodes = freq_18_22_cesium.CodeClock;
scalingFactor = 10^-14;
diffColumns = freq_18_22_cesium(:, 2:end);
scaled_diff = varfun(@(x) x * scalingFactor, diffColumns);
% Initialize a matrix
freq_matrix = NaN(size(diffColumns, 2), numel(clockCodes));

% Populate the matrix
for i = 1:numel(clockCodes)
    currentClockCode = clockCodes(i);
    rowsForCurrentClock = (clockCodes == currentClockCode);
    freq_matrix(:, i) = table2array(scaled_diff(
   rowsForCurrentClock, :))';
end

tau0 = 432000; %numero di secondi in 5 giorni -> è la distanza
   base

avar = zeros(size(freq_matrix,2),8); %for every selected clock we
   have a matrix which contains m allan variance computation for
   each clock


for j=1:size(freq_matrix,2) %for each clock
    for i=1:8 % number of points in which I want to compute Allan
   Variance
    n=(2^(i-1)); %a.v su intervallo di cinque giorni, a.v su
   intervallo di 10 giorni, a.v. su intervallo di 20 giorni, etc.
   fino ad allan variance su intervallo di
    avar(j,i) = sqrt(allan_var(freq_matrix(:,j),n,432000));
    assex(i) = n*tau0;
    end

end


for j=1:size(freq_matrix,2)
figure;
y = avar(j,:);
```

```matlab
loglog(assex, y);
grid on;
title(sprintf('Allan Variance plot for Clock %d', clockCodes(j)));
xlabel('log(\tau)');
ylabel('log(\sigma ^2 (\tau))');
end
```

B.10 Weights extraction.

```matlab
clear all;

folderPath_orig = '/Users/ariannaabis/Desktop/Tesi_Matlab/
    Dati_confronto/Originali/';
folderPath_cesls = '/Users/ariannaabis/Desktop/Tesi_Matlab/
    Dati_confronto/cs_ls/';
folderPath_masls = '/Users/ariannaabis/Desktop/Tesi_Matlab/
    Dati_confronto/mas_ls/';
fileName = 'P2303';
filePath_orig = fullfile(folderPath_orig, fileName);
filePath_cesls = fullfile(folderPath_cesls, fileName);
filePath_masls = fullfile(folderPath_masls, fileName);
pesi_orig = readtable(filePath_orig,'FileType','text');
pesi_cesls = readtable(filePath_cesls,'FileType','text');
pesi_masls = readtable(filePath_masls,'FileType','text');
lastRowIndex = size(pesi_orig, 1);
% pesi_orig = pesi_orig(1:(lastRowIndex - 1), :);
% pesi_cesls = pesi_cesls(1:(lastRowIndex - 1), :);
% pesi_allls = pesi_allls(1:(lastRowIndex - 1), :);

% Estrai la colonna Var1 come cell array di stringhe
stringsToSplit_orig = pesi_orig.Var1;
stringsToSplit_cesls = pesi_cesls.Var1;
stringsToSplit_masls = pesi_masls.Var1;

% Inizializza celle vuote per ogni colonna
numColumns = 15;
splitColumns_orig = cell(length(stringsToSplit_orig), numColumns);
splitColumns_cesls = cell(length(stringsToSplit_cesls), numColumns
    );
splitColumns_masls = cell(length(stringsToSplit_masls), numColumns
    );


% Itera su ogni riga e assegna i valori alle colonne
for i = 1:length(stringsToSplit_orig)
```

```
    % Dividi la stringa in un array di stringhe
    splitValues_orig = strsplit(stringsToSplit_orig{i}, ' ');
    splitValues_cesls = strsplit(stringsToSplit_cesls{i}, ' ');
    splitValues_masls = strsplit(stringsToSplit_masls{i}, ' ');

    % Assegna i valori alle colonne
    for j = 1:min(length(splitValues_orig), numColumns)
        splitColumns_orig{i, j} = splitValues_orig{j};
        splitColumns_cesls{i, j} = splitValues_cesls{j};
        splitColumns_masls{i, j} = splitValues_masls{j};
    end
end

% Crea la tabella finale
columnNames = cell(1, numColumns);
for j = 1:numColumns
    columnNames{j} = ['SplitColumn' num2str(j)];
end
pesi_orig = cell2table(splitColumns_orig, 'VariableNames',
    columnNames);
pesi_cesls = cell2table(splitColumns_cesls, 'VariableNames',
    columnNames);
pesi_masls = cell2table(splitColumns_masls, 'VariableNames',
    columnNames);


for j = 1:numColumns
    % Estrai i valori dalla colonna corrente
    currentColumn_orig = pesi_orig.(columnNames{j});
    currentColumn_cesls = pesi_cesls.(columnNames{j});
    currentColumn_masls = pesi_masls.(columnNames{j});

    % Converti i valori in double
    numericValues_orig = str2double(currentColumn_orig);
    numericValues_cesls = str2double(currentColumn_cesls);
    numericValues_masls = str2double(currentColumn_masls);


    % Sostituisci la colonna corrente con i valori numerici
    pesi_orig.(columnNames{j}) = numericValues_orig;
    pesi_cesls.(columnNames{j}) = numericValues_cesls;
    pesi_masls.(columnNames{j}) = numericValues_masls;
end

pesi_orig = pesi_orig(pesi_orig.SplitColumn1 == 60034, :); %2303:
    60034 %2207: 59789
pesi_orig (:, 1) = [];
pesi_cesls = pesi_cesls(pesi_cesls.SplitColumn1 == 60034,:);
pesi_cesls (:, 1) = [];
```

```
pesi_masls = pesi_masls(pesi_masls.SplitColumn1 == 60034,:);
pesi_masls (:, 1) = [];
% Numero di colonne per gruppo
columnsPerGroup = 2;

% Inizializzazione della nuova tabella
PesiOrig = table();
PesiCesLs = table();
PesiMasLs = table();

% Iterazione attraverso ogni gruppo di colonne
for j=1:size(pesi_orig,1)
for i = 1:columnsPerGroup:size(pesi_orig, 2)
    % Estrazione delle colonne correnti
    currentValues_orig = pesi_orig(j, i:i+columnsPerGroup-1);
    currentValues_orig.Properties.VariableNames = ["ClockCode", "
   Weight"];
    currentValues_cesls = pesi_cesls(j, i:i+columnsPerGroup-1);
    currentValues_cesls.Properties.VariableNames = ["ClockCode", "
   Weight"];
    currentValues_masls = pesi_masls(j, i:i+columnsPerGroup-1);
    currentValues_masls.Properties.VariableNames = ["ClockCode", "
   Weight"];


    % Unione delle colonne correnti in una nuova riga
    newRow_orig = (currentValues_orig);
    newRow_cesls = (currentValues_cesls);
    newRow_masls = (currentValues_masls);

    % Aggiunta della nuova riga alla nuova tabella
    PesiOrig = [PesiOrig; newRow_orig];
    PesiCesLs = [PesiCesLs; newRow_cesls];
    PesiMasLs = [PesiMasLs; newRow_masls];


end
end

  PesiOrig.Weight = PesiOrig.Weight ./ 100;
  PesiCesLs.Weight = PesiCesLs.Weight ./ 100;
  PesiMasLs.Weight = PesiMasLs.Weight ./ 100;



  lastRowIndex = size(PesiOrig, 1);

  % Specify the rows to delete
  %rowsToDelete = lastRowIndex - 3:lastRowIndex;
```

```
rowsToDelete = lastRowIndex - 5:lastRowIndex;  %2303


  % Delete the specified rows
  PesiOrig(rowsToDelete, :) = [];
  PesiCesLs(rowsToDelete, :) = [];
  PesiMasLs(rowsToDelete, :) = [];
```

B.11 EAL analysis with linear model and least squares frequency model only cesium (only masers).

```
    close all;
clear all;
clc;

% Specifica il percorso della cartella contenente i file di testo
folderPath = '/Users/ariannaabis/Desktop/Tesi_Matlab/EALtimescale/
   EAL_H_ori/';

% Ottieni la lista di file nella cartella
files = dir(fullfile(folderPath, '*.out'));

% Itera attraverso i file
for i = 1:length(files)
    % Costruisci il percorso completo del file corrente
    currentFile = fullfile(folderPath, files(i).name);

    fid = fopen(currentFile, 'r');
    data_cell = textscan(fid, '%f%f%f', 'Delimiter', ''); %
   Sostituisci con il tuo formato
    fclose(fid);

    sz = size(data_cell{1,1},1) ;
    out = cell(sz,size(data_cell,2));
    for k = 1:size(data_cell,2)
    t1 = data_cell(k); %accessing a subarray of a cell array which
    in our case is 1x1 cell array that contains another cell array
    t2 = [t1{:}];
    if isnumeric(t2)              % Takes care of floats
        out(:,k) = num2cell(t2);
    else
        out(:,k) = t2;
    end
    end
    data = cell2table(out, 'VariableNames', {'MJD', 'ClockCode', '
   ReadingDifference'});
    [~, uniqueIdx, ~] = unique(data(:, {'MJD', 'ClockCode'}), '
   rows', 'last');
    data = data(uniqueIdx, :);
    data =
    % Plotta i dati
    figure(i)
    plot(data.MJD, data.ReadingDifference);

    % Aggiungi etichette agli assi e titolo
    xlabel('MJD');
    ylabel('ReadingDifference ns');
    title(sprintf('EAL-%s Ori', strrep(files(i).name, '.out', ''))
   );
    ax = gca;
    indicesToShow = 1:20:numel(data.MJD);
    ax.XTick = data.MJD(indicesToShow);
    xtickangle(90);
    ax.XTickLabel = cellstr(num2str(ax.XTick(:), '%d'));

%     % Salva la figura nella cartella
%
    figureFileName = ['eal_' strrep(files(i).name, '.out', '') '
   _ori.fig'];
    saveas(gcf, fullfile(folderPath, figureFileName));


    % Chiudi la figura corrente per passare alla successiva
    %close(gcf);
    % Assegna un nome univoco alla tabella basato sul nome del
   file
    tableName = ['eal_ori_' strrep(files(i).name, '.out', '')];

    % Assegna la tabella a una variabile con il nome univoco
    eval([tableName ' = data;']);

    % Ora la tabella è memorizzata in una variabile con un nome
   univoco
end


% Specifica il percorso della cartella contenente i file di testo
folderPath = '/Users/ariannaabis/Desktop/Tesi_Matlab/EALtimescale/
   EAL_H_lin_mas_ls/';

% Ottieni la lista di file nella cartella
files = dir(fullfile(folderPath, '*.out'));
```

```matlab

% Itera attraverso i file
for i = 1:length(files)
    % Costruisci il percorso completo del file corrente
    currentFile = fullfile(folderPath, files(i).name);

    fid = fopen(currentFile, 'r');
    data_cell = textscan(fid, '%f%f%f', 'Delimiter', ''); %
   Sostituisci con il tuo formato
    fclose(fid);

    sz = size(data_cell{1,1},1) ;          %Line count
    out=cell(sz,size(data_cell,2));        %I want a single cell
   array and not a nested cell array
    for k = 1:size(data_cell,2)
    t1 = data_cell(k); %accessing a subarray of a cell array which
     in our case is 1x1 cell array that contains a 8101 x 1 cell
   array
    t2 = [t1{:}];
    if isnumeric(t2)               % Takes care of floats
        out(:,k) = num2cell(t2);
    else
        out(:,k) = t2;
    end
    end
    data = cell2table(out, 'VariableNames', {'MJD', 'ClockCode', '
   ReadingDifference'});
    [~, uniqueIdx, ~] = unique(data(:, {'MJD', 'ClockCode'}), '
   rows', 'last');
    data = data(uniqueIdx, :);
    % Plotta i dati
    figure(i+5)
    plot(data.MJD, data.ReadingDifference);

    % Aggiungi etichette agli assi e titolo
    xlabel('MJD');
    ylabel('ReadingDifference ns');
    title(sprintf('EAL-%s New', strrep(files(i).name, '.out', ''))
   );
    ax = gca;
    indicesToShow = 1:20:numel(data.MJD);
    ax.XTick = data.MJD(indicesToShow);
    xtickangle(90);
    ax.XTickLabel = cellstr(num2str(ax.XTick(:), '%d'));

    % Aggiungere una griglia per chiarezza
    %grid on;


```

```
%      % Salva la figura nella cartella
     figureFileName = ['eal_' strrep(files(i).name, '.out', '') '_new.fig'];
     saveas(gcf, fullfile(folderPath, figureFileName));
%
%      % Chiudi la figura corrente per passare alla successiva
%      %close(gcf);
%
%      % Assegna un nome univoco alla tabella basato sul nome del file
     tableName = ['eal_new_' strrep(files(i).name, '.out', '')];
%
%      % Assegna la tabella a una variabile con il nome univoco
     eval([tableName ' = data;']);


    % Ora la tabella è memorizzata in una variabile con un nome univoco
end

% Creare un elenco di tutti i tipi dologi distinti
codes = [1930002, 1930003, 1930004, 1930005, 1934901];

% Iterare attraverso i tipi di orologi
for i = 1:length(codes)
    % Ottenere le tabelle corrispondenti sia per ori che per new
    oriTable = eval(['eal_ori_' num2str(codes(i))]);
    newTable = eval(['eal_new_' num2str(codes(i))]);

    % Calcolare la differenza tra le colonne ReadingDifference
    diffValues = newTable.ReadingDifference - oriTable.ReadingDifference ;

    % Plottare la differenza
    figure(i+10);
    plot(oriTable.MJD, diffValues);

    % Aggiungi etichette agli assi e titolo
    xlabel('MJD');
    ylabel('ReadingDifference ns');
    title(sprintf('(EALnew - %d) - (EALold - %d)', codes(i), codes(i)));
    ax = gca;
    indicesToShow = 1:20:numel(oriTable.MJD);
    ax.XTick = oriTable.MJD(indicesToShow);
    xtickangle(90);
    ax.XTickLabel = cellstr(num2str(ax.XTick(:), '%d'));

    % Aggiungere una griglia per chiarezza
   grid on;

%     % Salva la figura nella cartella (modifica il percorso
   secondo necessità)
     figureFileName = sprintf('eal_difference_plot_%d.fig', codes(
   i));
     saveas(gcf, figureFileName);

     freq_values_orig = diff(oriTable.ReadingDifference)./5;
     freq_values_new = diff(newTable.ReadingDifference)./5;
     figure(i+15)
     x_axis = oriTable.MJD(2:end);
     plot(x_axis, freq_values_orig)
     hold on ;
     plot(x_axis, freq_values_new)
     hold off;

     % Aggiungi etichette agli assi e titolo
    xlabel('MJD');
    ylabel('Frequency ns');
    title(sprintf('freq orig - freq new for clock %d', codes(i)));
    ax = gca;
    indicesToShow = 1:20:numel(oriTable.MJD);
    ax.XTick = oriTable.MJD(indicesToShow);
    xtickangle(90);
    ax.XTickLabel = cellstr(num2str(ax.XTick(:), '%d'));
    legend('Orig', 'New');

    % Aggiungere una griglia per chiarezza
    grid on;

%     % Salva la figura nella cartella (modifica il percorso
   secondo necessità)
     figureFileName = sprintf('comparison_freq_%d.fig', codes(i));
     saveas(gcf, figureFileName);
end
```

B.12 Computation of residuals with addition of stochastic term.

```
resid =cell(length(selected_clocks),59);
drift_est = table(length(selected_clocks), 59);
for i = 1:length(selected_clocks)
% disp(selected_clocks(i));
```

```
 previous_month = subset_tables_eal{i,1}(ismember(subset_tables_eal{i,1}.MJD,[58114:5:58149]),:).ReadingDifference;
 t0_prev = 58114;
 %disp(previous_month);
 %disp(length(previous_month));
     for k=1:59
     freq_curr = diff(subset_tables_tt{i,k}.ReadingDifference)/5; %I compute frequencies for the 'current' three months
     x_curr = subset_tables_tt{i,k}.MJD(2):5:subset_tables_tt{i,k}.MJD(end); %I make it as if my first frequency data is computed in the second MJD of the first of three months
     y_curr = movmean(transpose(freq_curr),3); %making freq a row vector and computing moving average
     coeff = polyfit(x_curr,y_curr,1); %finding y_0 and d such that y(t) = y_0 + d*t, coeff(1) = d, coeff(2) = y_o
     d = coeff(1);
     drift_est{i,k} = d;
     x_new = x_curr(end):5:subset_tables_tt{i,k+1}.MJD(end); %MJD for next month
     new_month_values = subset_tables_eal{i,k+1}(subset_tables_eal{i,k+1}.MJD >= x_curr(end), :).ReadingDifference;
     x0 = new_month_values(1);
     y0 = (x0 - previous_month(1))/((length(previous_month)*5)-5);
     x = zeros(1,length(new_month_values));
     x(1)=x0;
     for j=2:length(new_month_values)
     if selected_clocks(i) == 1350332
        sigma1 = 1.972;
        sigma2 = 0.05;
        tau = 5;
        covariance_matrix = [sigma1^2 * tau + sigma2^2 * (tau^3 / 3), sigma2^2 * (tau^2 / 2); ...
                  sigma2^2 * (tau^2 / 2), sigma2^2 * tau];
        num_samples = 1;
        stochastic_contribution = mvnrnd([0, 0], covariance_matrix, num_samples);
     else
         stochastic_contribution = [0;0];

     end
     x(j) = x0 + y0.*(x_new(j)-x_curr(end)) + (1/2).*d.*((x_new(j)-x_curr(end)).^2)+(1/2).*d.*(x_new(1)-t0_prev).*(x_new(j)-x_curr(end)) + stochastic_contribution(1,1);
     end
     resid{i,k} =  x - new_month_values';
     %ensamble of length(new_month_values) values
```

```
        previous_month = new_month_values;
        t0_prev = x_new(1);
        end
end
```

Bibliography

[1] D. W. Allan. «Time and Frequency (Time-Domain) characterization, estimation and prediction of precision clocks and oscillators.» In: *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control* 34.6 (1987) (cit. on pp. 1, 8).

[2] J. Rutman. «Characterization of Phase and Frequency Instabilities in Precision Frequency Sources: 15 years of progress». In: *Proceedings of the IEEE* 66.9 (1978) (cit. on pp. 4, 11, 19, 21).

[3] BIPM. «Annual report of the BIPM time section». In: (1988) (cit. on p. 7).

[4] G. Panfilo, A. Harmegnies, and L. Tisserand. «A new prediction algorithm for the generation of International Atomic Time». In: *Metrologia* 49 (2011) (cit. on pp. 10–12).

[5] G. Panfilo and F. Arias. «The Coordinated Universal Time (UTC)». In: *Metrologia* 56 (2018) (cit. on p. 12).

[6] D.W. Allan. «Statistics of atomic frequency standards». In: *Special Issue on Frequency Stability, Proc. IEEE* (1966) (cit. on p. 22).

[7] Antoine Baudiquez. «Metrology and Statistics: From clocks to millisecond pulsars». PhD Thesis. Université Bourgogne Franche - Compté, 2022 (cit. on p. 24).

[8] Thomas Mikosch. *Elementary Stochastic Calculus with Finance in view.* World Scientific, 1998 (cit. on p. 24).

[9] L. Galleani, L. Sacerdote, P. Tavella, and C. Zucca. «A mathematical model for the atomic clock error». In: *Metrologia* 40 (2003) (cit. on pp. 28, 30).

[10] C. Zucca and P. Tavella. «The Clock Model and Its Relationship with the Allan and Related Variances». In: *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control* 52.2 (2005) (cit. on pp. 78, 84).