Time-series Imputation Algorithm

David Howe [1]

[1]NIST

October 30, 2023

Abstract

Statistical imputation is a field of study that attempts to fill missing data. It is commonly applied to population statistics whose data have no correlation with running time. For a time series, data is typically analyzed using the autocorrelation function (ACF), the Fourier transform to estimate power spectral densities (PSD), the Allan deviation (ADEV), trend extensions, and basically any analysis that depends on uniform time indexes. We explain the rationale for an imputation algorithm that fills gaps in a time series by applying a backward, inverted replica of adjacent live data. To illustrate, four intentional massive gaps that exceed 100% of the original time series are recovered. The L(f) PSD with imputation applied to the gaps is nearly indistinguishable from the original. Also, the confidence of ADEV with imputation falls within 90% of the original ADEV with mixtures of power-law noises. The algorithm in Python is included for those wishing to try it.

Time-series Imputation Algorithm

D. A. Howe, *Fellow, IEEE,* C. Champagne, and N. Schlossberger

***Abstract*— Statistical imputation is a field of study that attempts to fill missing data. It is commonly applied to population statistics whose data have no correlation with running time. For a time series, data is typically analyzed using the autocorrelation function (ACF), the Fourier transform to estimate power spectral densities (PSD), the Allan deviation (ADEV), trend extensions, and basically any analysis that depends on uniform time indexes. We explain the rationale for an imputation algorithm that fills gaps in a time series by applying a backward, inverted replica of adjacent live data. To illustrate, four intentional massive gaps that exceed 100% of the original time series are recovered. The $\mathcal{L}$(f) PSD with imputation applied to the gaps is nearly indistinguishable from the original. Also, the confidence of ADEV with imputation falls within 90% of the original ADEV with mixtures of power-law noises. The algorithm in Python is included for those wishing to try it.**

***Index Terms*— ADEV, convolution, dead-time, frequency, gaps, imputation, missing data, power-law noise models, power spectrum, Python, sparce, time, time-series.**

I. Introduction

WAVEFORMS are not continuous but are instead discrete-continuous because a sampler acts on an input signal, like signal *y(t)* of block-length *T* with τ_0 intervals, thereby creating the time series. Time-series statistics transform this sampled signal, generally decomposing parameters *vs.* frequencies in a particular descriptive way as if the input signal is passed through a tunable spectrum analyzer with a center frequency at $1/(2n\tau_0)$. This requires that measurements be evenly sampled, without interruptions or "gaps," to extract and delineate spectral frequencies. The effect of sampling in the frequency and time domains use convolution in one domain (e.g., time domain) to determine point-wise multiplication in the other domain (e.g., frequency domain) and vice-versa. Convolution determines impulse response. To improve precision, we circularize the data as a method of invoking a circular convolution to replace linear convolution of a data-run *T* [1,2]. This is the basis of our time-series imputation method. From a Fourier decomposition perspective, a *T*-length block of data is interpreted as periodic with infinite extent beyond a data run of length *T*. We reason that by a conservation principle, a data's variance does not diminish for intervals beyond the measured, observed block *T*, i.e., > *T*, thus periodic extensions are assumed [3]. This paper applies periodic extensions as a strategy to fill unmeasured data between measured data. Representations of data in either the frequency or time domains share certain properties that directly depend on the details of specific sampling mentioned above [4]. It is only important to know that these samples create stationary increments, thus are ergodic [5]. In Sec. II, we build an algorithm that fills gaps with periodically extended live data. In Sec. III, we test it using $\mathcal{L}$(f) and ADEV characterizations.

Manuscript received November 3, 2021. This work was supported in part by the Department of Commerce and Office of Naval Research.

First Author is with the National Institute of Standards and Technology, Boulder, CO 80305 USA (e-mail: David.Howe@nist.gov).

Second Author is with Space Science Division, U.S. Naval Research Lab, Wash. DC 20375 USA (email: Chloe.A.Champagne@vanderbilt.edu).

Third Author is with the University of Colorado, Boulder, CO 80303 USA (e-mail: Noah.Schlossberger@colorado.edu).

II. Imputation Algorithm

The algorithm extends a data run by adding replicas of it to both its ends. Fig. 1 shows various extensions. For example, Type 3 in Fig. 1(a) shows the middle portion as the "live" data and the left and right portions are replicas of the middle and extending the series by 3X. "Left" and "right" will henceforth denote replicas before and after the live data, respectively.

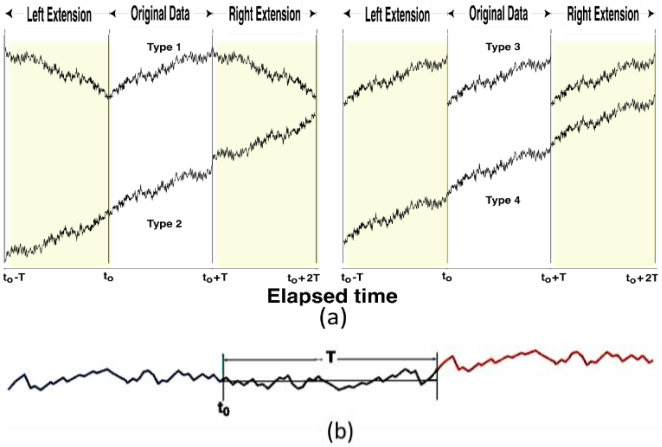

Fig. 1. (a) Four methods of extending original data between t_0 to (t_0+T) are shown. Type 1: reflect (mirror), Type 2: reflect+invert, Type 3: replica, Type 4: shifted replica, that matches ends. Gaps (dead-times) would be in intervals $(t_0\text{-}T) \rightarrow t_0$ and $(t_0+T) \rightarrow (t_0+2T)$ that are filled with surrogate data on either side of the original "live-time" data shown in between as $t_0 \rightarrow (t_0+T)$. (b) Extension of original by reflection + inversion (Type 2) with slope removed. Imputed noise extensions are *iid*, that is, independent and identically distributed at the range of lags in the autocorrelation function (ACF). We find that imputed noise is *iid* to the same degree as the original live noise is already inherently independent, which is quantified as "self-similarity" [10,11].

A. *Types of Data Extensions*

Four methods of taking the original data series and adding various replicas of it to both its ends are shown in Fig. 1(a). Type 3 is the simplest in which the point following the last point in the live data run becomes the first point in the extension. The Type 4 modifies Type 3 by removing an endpoint discontinuity. Type 1 is more interesting by creating a visual mirror image of the original data run at its first and last points by reversing the direction of the data. Type 2 inverts the sign of the Type 1 extension, thus Type 2 is a reflected plus inverted extension. Note that Types 1 and 2 have no endpoint discontinuities. Fig. 1(b) shows Type 2 left and right extended noise without the background slope. The reflected-inverted extensions of live data, a Type 2 method, is best for reasons explained next.

B. *Filling Gaps with Independent Identically Distributed (iid) Imputed Extensions*

Extensions as described are essentially extrapolations of key noise properties, i.e., surrogate data. The astute observer will note that Type 3 and 4 extensions are certainly identically distributed as the live data but not clearly independent of the live data. In fact, replicas are completely correlated with live data at lag-T. But only a T-long data run can practically be considered in analyses. Thus, T-long data runs with gaps require the imputations to be independent. We temporally reverse live data, making the data recede backwards as shown as Type 1 and 2 by reflection (creating a visual mirror image) of the original data run to the left of its first and to the right of its last points. Note that trends in the data now repeat with period $2T$ instead of T so are outside the T-block limit used in frequency-time statistics. Nevertheless, we can remove this effect by either removing the trend (a T-long slope) or naturally by inverting the sign of the reflected extension as seen in Type 2 which is visualized in Fig. 1(b). This is allowed because the frequency-time statistics we've described are invariant to either method of trend removal [6-9].

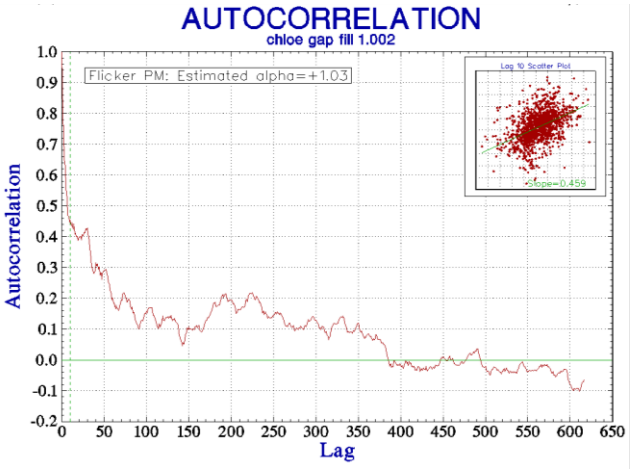

Fig. 2. Autocorrelation function (ACF) of typical Type 2 imputation of time-series data that has T = 700 increments. Inset is the cross-correlation scatter plot of on-time vs. 10-point lag. Note the significant data independence (de-correlation) using this inverted-reflection Type 2 imputation shown in Fig. 1.

C. *Autocorrelation Function and Degrees of Freedom to test Imputed Extensions are Independent Identically Distributed*

The Types 1 and 2 methods of extending the data series meet significant criteria, namely, they are *iid*, that is, independent and identically distributed but Type 2 is better as determined by the autocorrelation function (ACF). ACF is a serial correlation plot primarily used to assess the independence (i.e., randomness) of a set of observations compared to a set that lags by a constant number of samples [10]. The ACF of a time series multiplies data by a delayed version of itself, thereby showing the degree to which its value at one time is similar to its value at a certain later time. More specifically, the autocorrelation at lag k is defined as

$$\rho_k = \frac{E[(z_t - \mu)(z_{t+k} - \mu)]}{\sigma_z^2} \quad , \tag{1}$$

where z_t is the time series, μ is its mean value, σ_z^2 is its variance, and E denotes the expected value. Its estimate is the expression

$$r_k = \frac{\frac{1}{N}\sum_{t=1}^{N-k}(z_t - \bar{z})(z_{t+k} - \bar{z})}{\frac{1}{N}\sum_{t=1}^{N}(z_t - \bar{z})^2} \quad , \tag{2}$$

where $\bar{z}$ is the mean value of the time series and N is the number of data points. The ACF would be 0 starting at lag-1 for completely uncorrelated points in the time-series.

Fig. 2 is representative of a typical ACF of data in which numerous extensions were imputed to multiple-gapped live measurements. The scatter plot lag 10 of the gap-filled data is shown and no new correlation is seen compared to the ACF of the original data as indicated by slopes in the ACF. Slopes are preserved, indicating "self-similarity" or underlying long-memory properties have not changed [11]. More discussion follows in Sec. III. To review, white-noise data is uncorrelated, whereas random-walk data is correlated being an integral of white noise and flicker data is correlated being a fractional integral [12]. Superposition of slopes of the ACF is often used to delineate these and other noises. Based on the ACF, Type 2 is preferred instead of Types 1, 3 and 4 in order to decorrelate the imputed data while preserving low-frequency trends and slopes from lag-1 to lag-$2T$ [13].

Further evidence that imputed data is *iid* is also shown in that the equivalent degrees of freedom, a measure of data-point independence in a statistical average, always increases with Type 1 and 2 extensions and most dramatically increases for white noise by up to 6X, and less so for flicker and random-walk noise [14-17]. It would seem preposterous to report a reliable estimate of frequency stability with massive gaps. However, we find that with care, the procedure in the next section fills large, multiple gaps such that statistics have the same descriptive properties to within 90% confidence of the original time-series [18].

D. *Procedure*

For datasets with multiple gaps of dead time, the algorithm is as follows:

1. Find single-point gaps in the data set and fill them by taking the average of the points on either side.
2. Find the largest continuous run of data and impute the gap immediately to its right.
3. Continue until the end of the dataset is reached.
4. Then reverse the dataset and resume this algorithm until the beginning of the dataset is reached. This in essence creates double-sided extensions before and after live segments to

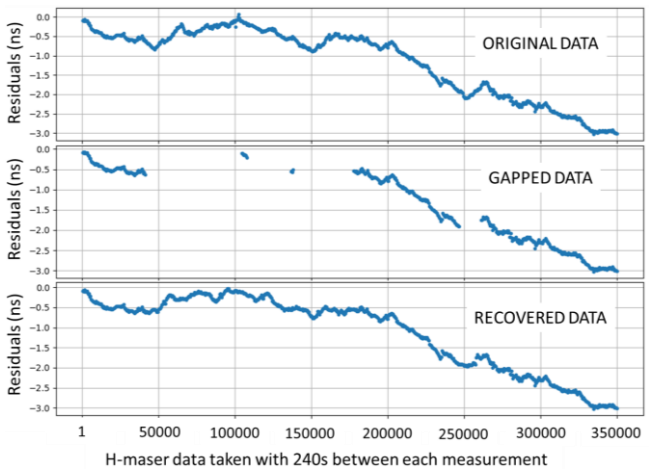

Fig. 3. TOP: Time-series measurements of NIST H-maser vs. UTC(NIST); MIDDLE: Four large gaps >100% of live data are intentionally created; BOTTOM: Test showing the gaps are filled in by the imputation algorithm.

balance and avoid only-left or only-right extensions in gaps while maximally filling them.

5. If a gap size is sufficiently small so imputation from the left fills it, then apply Type 2 extensions. If the left does not contain enough points, check the right and match averages in the middle. If there are still not enough data points, try to impute an additional half from each side. In situations where the gap size is too large for all of these, the gap is skipped and returned to after the forward and reversed imputation. This process is repeated until there are no gaps remaining.

The data with gaps is time series $\{x_n\}$, $n=1, 2 \ldots N-1, N$. The procedure above determines the number and length of gaps with zeroes, i.e., finds the i^{th} gap length in the list of gaps, designated as $n_{gap\text{-}i}$. Then we choose $n_{gap\text{-}i}$ live points that precede the gap at x_n to its left. The span $x_{n\text{-}N}$ to x_n are extended to fill in the gap from x_{n+1} to x at $n_{gap\text{-}i}$. At this point, we can apply a Type 2 (inverted-reflected) right extension of the live sequence such that $x_{n+i} = \frac{x_n + x_{n-1}}{2} + x_n - x_{n-i+1}$.

Now we add a slope to the gap sequence x_{n+1} to [x at $n_{gap\text{-}i}$] so that $x_i = [x \text{ at } n_{gap\text{-}i}] - x_n + ci$, $i = n, n+1, n+2 \ldots n_{gap\text{-}i}$. This is so that the end of the Type 2 extension doesn't present a mismatch to the beginning of the next live segment. A new sequence $\{x_i\}$ that once had live, then dead, and resumed live data should now include the imputed data in the previously dead portion. The desired slope ci is calculated by taking the endpoints of the gap, subtracting to obtain Δx, and dividing by the difference in the time. A low-pass filtered version of the data is used to match the endpoints, which is necessary to avoid the creation of a sawtooth jump in the gap [19]. This procedure can create *iid* surrogates for unprecedented gap lengths that can be greater than 100% of the live data before and after the gap.

Frequency-time variances are invariant to adding matched slope ci. The difference between the desired and current slope is the slope to apply to the reflected points. Time indexes are chosen to advance monotonically through the gap's start and end zones.

Simulations of white, flicker, and random-walk noise types show 512-length datasets *vs.* the same with 150 values removed and gap-filled yield matching ADEV(τ) within 90% confidence [19]. We illustrate the effectiveness of the algorithm in a particularly extreme example. Referring to Fig. 3, the top plot is original data that are the time differences between a NIST H-maser and the NIST time scale, UTC(NIST). Here, 8.4 x 10^6 measurements are taken with 240s between each measurement; the unit of the vertical scale is nanoseconds. Four large segments of the original data have been removed in the middle plot. The imputation algorithm applied to the middle plot produced the bottom plot whose spectral properties replicate as shown next.

III. $\mathcal{L}$(f) and ADEV

$\mathcal{L}$(f) is a ratio of the carrier-to-noise in logarithmic units of dBc/Hz versus log SSB Fourier frequency in hertz. For small phase deviations, $\mathcal{L}$(f) measures the power spectral density (PSD) of frequency noise $y(t)$, i.e., $S_y(f)$. $\mathcal{L}(f) = (\nu_0/f)^2 S_y(f)$, at frequency ν_0. The Allan deviation (ADEV) is a different spectrum analyzer that determines power-law FM noises [6-8].

The PSD in the form of $\mathcal{L}$(f) and ADEV reveal substantially equal information about autocorrelation. ADEV is a bit easier

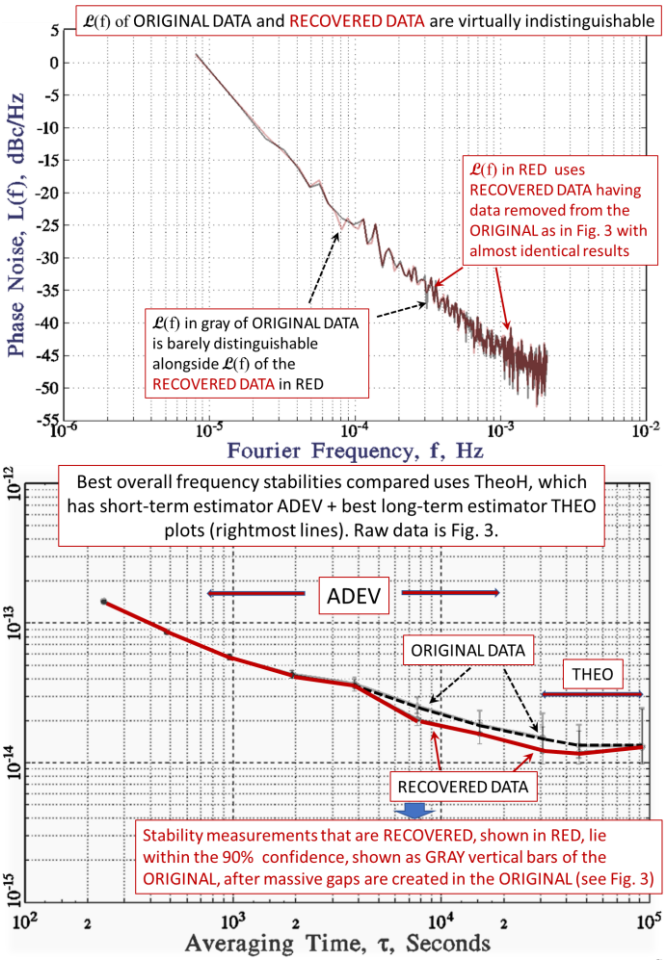

Fig. 4. TOP: Two overlaid $\mathcal{L}$(f) plots, 80 μHz < f < 2 mHz, of original and recovered time-series measurements with gaps filled by imputation as in Fig. 3; BOTTOM: Two overlaid plots of widest-τ ADEV+THEO. Note the H-maser data exhibit both white FM noise $\tau^{-1/2}$ slope is in the range 200s < τ < 3 x 10^4s, and flicker FM noise (zero-slope) at τ > 3 x 104s long-term for the T-block of data in Fig. 3 that ends at slightly over $T = 10^5$ s in the original (dashed line) and recovered (solid) data runs [19,20]. Note that even with massive gaps filled by imputation, white and flicker FM power-law slopes (clock models) are still characterized with the correct, unbiased levels and slopes with 90% confidence.

to use as an uncertainty of systematics, such as linear frequency drift, in convenient units of averaging time. In contrast, $S_y(f)$ is in units of (frac freq)^2 /Hz evaluated at f in microhertz.

Fig. 4 shows the differences between original and imputed $\mathcal{L}(f)$ and ADEV for the H-maser measurements represented in Fig. 3. A computation of THEO, which is the best long-term estimator of ADEV, is added as the longest two τ-values beyond 3 x 10^4s to see the flicker noise, i.e., zero-slope flicker FM [20-22]. Note that even with massive gaps filled by the imputation algorithm, white and flicker frequency models are still characterized with the correct, unbiased levels and slopes.

IV. Implementation in Python

A program written in Python called "fillgaps.py" is available at: https://zenodo.org/record/5594587. The following software packages are required for the code to properly execute:

- Python (developed in version 3.10.10)
- NumPy (developed with version 1.20.0) [23]
- SciPy (developed with version 1.7.1) [24]
- Matplotlib (developed with version 3.3.3) [25]

The versions listed above are not necessarily required, but the script implementation is guaranteed to be compatible with these versions. All the above are open source and, thus, free and easily accessible on the internet. The user can run the script through terminal commands or a Python-compatible IDE.

To execute, the user inputs the file containing the gap-laden data with the time stamp in one column and data in a second column. Both csv and txt input file formats are supported. Gaps are determined by finding the smallest difference between consecutive time stamps and recreating the data set with equally-spaced intervals. Since the program imputes at irregular time jumps and then reindexes the time stamp, remove pre-existing imputations of zeroes or interpolations. Otherwise, "No Gaps detected" is outputted. Finally, the user inputs an output file name with the .csv extension and adds it at the end of the command line. An example call in Linux command line is: 'python3 fillgaps.py <input_file> <output_file>.

Upon completion, the script outputs a graph of the data before and after the imputation and an output file in a csv format of the filled data in the same directory the script is run.

Intervals with no data are the 'gaps' and are filled with NumPy's nan constant (representative of 'not a number'). Both time stamps and data are stored in NumPy arrays and are manipulated in-place.

An executable version with a GUI that takes a csv input is also available at: https://zenodo.org/record/5595200. It contains a 55MB .exe file for inclusion of all libraries and features of the above non-GUI Python version.

Acknowledgment

The authors thank Paul Ray, Julia Deneva, and Andrea Lommen of NASA's NICER Working Group for x-ray Pulsar timing data and discussions that motivated the algorithm.

References

[1] C.A. Greenhall, D.A. Howe and D.B Percival, "Total variance, an estimator of long-term frequency stability", *IEEE Trans. Ultrasonics, Ferroelectrics and Freq. Control*., Vol. UFFC-46, No. 5, pp. 1183-1191, Sept. 1999.

[2] N. Schlossberger and D. Howe, "Analysis of powers-of-two calculations of the Allan variance and their relation to the standard variance," Proc. 2019 Joint Mtg. IEEE Intl. Freq. Cont. Symp. and EFTF Conf., 5 p.

[3] D. A. Howe, "Circular representation of infinitely extended sequences," Proc. 1995 IEEE Intl. Freq. Cont. Symposium, pp. 337-346.

[4] D. W. Allan, M. A. Weiss, and J. L. Jespersen, "A frequency-domain view of time-domain characterization of clocks and time and frequency distribution systems," Proc. 1991 IEEE Freq. Cont. Symp., pp. 667-678.

[5] Halmos, P. R., *Lectures on Ergodic Theory*, The Mathematical Society of Japan, Kenkyushu Printing Co., Ltd., Tokyo, 1956.

[6] Howe, D.A., Allan, D.W., Barnes, J.A., "Properties of signal sources and measurement methods," Proc IEEE International Symp on Freq Control, May, 1981, pp. A1-A47.

[7] D.B Sullivan, D.W Allan, D.A Howe, and F.L Walls, eds., *Characterization of clocks and oscillators*, Natl. Inst. Stand. Technol. Technical Note 1337, http://tf.nist.gov/timefreq/general/pdf/868.pdf (March 1990).

[8] P1139™/D1 (Draft) Standard Definitions of Physical Quantities for Fundamental Frequency and Time Metrology – Random Instabilities, IEEE-SA Standards Board, The Institute of Electrical and Electronics Engineers, New York 10016-5997, USA (in process).

[9] NIST/SEMATECH e-Handbook of Statistical Methods, http://www.itl.nist.gov/div898/handbook/

[10] G. Box and G. Jenkins, *Time Series Analysis, forecasting and control*, Holden-Day, San Francisco, 1976.

[11] D. B. Percival, "The statistics of long memory processes," Ph.D. dissertation, Department of Statistics, University of Washington, Seattle, WA, 1983.

[12] Mandelbrot, B., "Statistical self-similarity and fractional dimension," *Science* **05** May 1967, Vol. 156, Issue 3775, pp. 636-638

[13] W.J. Riley and C.A. Greenhall, "Power law noise identification using the lag 1 autocorrelation," Proc. 18th European Frequency and Time Forum (EFTF), 2004.

[14] D. A. Howe and F. Vernotte," Generalization of the Total variance approach to the modified allan variance," Proc. 1999 PTTI Mtg.

[15] D. A. Howe, R. Beard, C. A. Greenhall, F. Vernotte, and B. Riley, "Total Hadamard variance: application to clock steering by kalman filtering," Proc. 2001 EFTF Conf., pp. 423-427.

[16] F. Vernotte and D. A. Howe,"Generalization of the Total variance approach to the different classes of structure functions," Proc. 2000 EFTF Conf., pp. 375-379.

[17] D. B. Percival, D.A. Howe. "Total variance as an exact analysis of the sample variance," 29th Annual Precise Time and Time Interval (PTTI) Meeting. Dec, 1997.

[18] D. Mondal and D. B. Percival, `Wavelet variance analysis for gappy time series,' *Annals of the Institute of Statistical Mathematics*, **62**, no. 5, pp. 943-966 (2010).

[19] D. A. Howe and N. Schlossberger, "Characterizing frequency stability measurements having multiple data gaps," in process, dhowe@nist.gov.

[20] D. A. Howe, "TheoH: a hybrid, high-confidence, statistic that improves on the Allan deviation," *Metrologia*, **43** (2006) S322–S331.

[21] B. Lewis, "A fast algorithm for calculation of Thêo1", *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, 67, 10, pp. 2187-2190, 1 Oct 2020.

[22] W. J. Riley, "Handbook of Frequency and Stability Analysis," NIST Special Publication SP1065.

[23] C.R. Harris, K.J. Millman, S.J. van der Walt, et al., "Array programming with NumPy," *Nature*, 585, 357–362 (2020). DOI: 0.1038/s41586-020-2649-2.

[24] P. Virtanen, R. Gommers, T. E. Oliphant, et al., "SciPy 1.0: Fundamental algorithms for scientific computing in Python," *Nature Methods*, 17(3), 261-272 (2020).

[25] J. D. Hunter, "Matplotlib: A 2D graphics environment**,"** *Computing in Science & Engineering*, **9**, 90-95 (2007). DOI:10.1109/MCSE.2007.55.