


RESEARCH

Open Access



# Nearest advocate: a novel event-based time delay estimation algorithm for multi-sensor time-series data synchronization

Christoph Schranz<sup>1\*</sup> , Sebastian Mayr<sup>1</sup>, Severin Bernhart<sup>1</sup> and Christina Halmich<sup>1</sup>

\*Correspondence:  
christoph.  
schranz@salzburgresearch.at

<sup>1</sup> Human Motion Analytics,  
Salzburg Research,  
Jakob-Haringer-Strasse 5/3,  
5020 Salzburg, Austria

## Abstract

Estimating time delays in event-based time-series is a crucial task in signal processing as it affects the data quality and is a prerequisite for many subsequent analyses. In particular, data acquired from wearable devices often suffer from a low timestamp precision or clock drift. Current state-of-the-art methods such as Pearson Cross-Correlation are sensitive to typical data quality issues, e.g. misdetected events, and Dynamic Time Warping is computationally expensive. To overcome these limitations, we propose Nearest Advocate, a novel event-based time delay estimation method for multi-sensor time-series data synchronisation. We evaluate its performance using three independent datasets acquired from wearable sensor systems, demonstrating its superior precision, particularly for short, noisy time-series with missing events. Additionally, we introduce a sparse variant that balances precision and runtime. Finally, we demonstrate how Nearest Advocate can be used to solve the problem of linear as well as non-linear clock drifts. Thus, Nearest Advocate offers a promising opportunity for time delay estimation and post-hoc synchronization for challenging datasets across various applications.

**Keywords:** Event-based time-series, Time delay estimation, Synchronization, Clock drift, Cross-correlation, Kernel cross-correlation, Dynamic time warping, Wearable devices

## 1 Introduction

Time delay estimation is the process of quantifying the relative time shift between two time-series that observe the same quantities [1, 2]. This technique is critical in various domains, including signal processing, control systems, and multimedia analysis. This paper focus on the time delay estimation of event-based time-series data. Unlike continuous signals with a constant sample rate, event-based time-series consist of a sequence of timestamps that indicate the re-occurrence of similar events in time. Common examples of such data are heartbeat, stride, or fault events.

In general, datasets created by multi-sensor data collections contain time offsets if no automated time synchronization device is applied in advance. Therefore, manually induced synchronization points at data collection, i.e. events easy to detect,

enable a manual time offset removal in the data post-processing by identifying the synchronization events in all sensor systems and shifting the different time-series on one mutual timeline. However, this method is often not feasible, leads to a more complex experiment setup, and also additional work in post-processing [3–6].

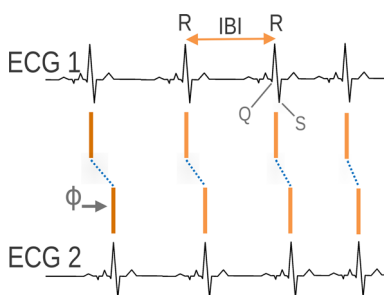
Clock drifts are timeline deviations from an ideal timeline that cause varying additional time delays and are usually measured in parts per million (ppm) or seconds per second, e.g. clocks with crystal oscillators have an accuracy of 10–100 ppm [3, 7]. They occur in custom sensor hardware as well as in high-end devices. Whereas inter-device clock drifts between different devices can be relatively large, inter-device clock drifts between devices of the same type can also occur, but are more likely to be negligible. Constant clock drifts are caused by oscillator irregularities that are triggered by crystal production errors or aging. Moreover, temperature fluctuations, concussions, or power supply alterations interfere regular oscillating clocks and raise erratic clock drifts [4, 7].

The standard procedure for synchronizing event-based time-series is to interpolate the temporal differences of all subsequent events to a constant sampling rate and a subsequent time delay estimation. Common methods for time delay estimation on a signal include Pearson Cross-Correlation (PCC), Kernel Cross-Correlation (KCC), and Dynamic Time Warping (DTW), each with its own strengths and weaknesses [8, 9].

By shifting one signal across various time offsets and computing the correlation at each shift, PCC considers the time delay that yields the highest cross-correlation as the most likely estimate [10–12]. This can be computationally expensive ( $\mathcal{O}(n \cdot T)$ ), but a performance gain to  $\mathcal{O}(n \cdot \log(n))$  can be achieved by applying the Fast Fourier Transform (FFT) [13, 14]. However, the application of PCC has several drawbacks: The precision of the determined time offset decreases if the inherent frequency of the signal variability is lower than that of the occurring events [12]. In addition, PCC also shows low robustness in case of too short time-series as well as missing events or imprecise timestamps, which often requires an expensive correction [12, 15].

Kernel Cross-Correlation (KCC) is a time delay estimation method that builds on the concept of PCC while incorporating a kernel function to enhance accuracy. The KCC algorithm for event-based time-series data converts the events into equidistant arrays with binary event-hot encoding and convolves this discrete representation with the kernel function before computing their Cross-Correlation. Finally, it shifts one input across multiple candidate time delays to find the delay that maximizes the Cross-Correlation similar to PCC [16, 17]. The type of kernel function is chosen based on the specific characteristics of the event-based time-series being analyzed, such as noise level or frequency content. In the case of event-based time-series, a triangular kernel can be used, which reduces noise and improves time delay estimation precision [16].

Dynamic Time Warping (DTW) is a method for aligning two similar time-series signals based on a distance-like similarity measure. This measure is calculated by finding the minimal sum of distances in a cross-distance matrix containing the differences between all elements of the test and reference sequences. The advantage of DTW lies in its ability to allow a nonlinear mapping between signals, accounting for local time domain distortions. However, this flexibility comes at the cost of significant computational demands of  $\mathcal{O}(n \cdot m)$  [18]. DTW is particularly suited for systems or objectives that experience severe clock drifts or irregular connection issues [19].



**Fig. 1** Two identical ECGs with a relative time delay of  $\phi$  (gray) with their characteristic beats, i.e., R-peaks depicted in orange

We identified a gap in methods and research on how to estimate time delays in event-based multi-sensor time-series data, such as two identical measurements shifted relatively in time (see example in Fig. 1). Additionally, we state that the transformation of inter-event interpolation can result in information loss, which makes existing methods less accurate. Despite the growing importance of accurately estimating time delays due to the rise of wearable devices and the combination of data from multiple sensors, the body of research dedicated to this subject remains disproportionately scant.

This paper introduces the Nearest Advocate (NAd) algorithm, a novel approach designed for event-based multi-sensor time-series data synchronization. Building on an initial idea from [20], the NAd algorithm has been significantly improved and expanded, and it has been rigorously validated. It is compared against current state-of-the-art methods to demonstrate its effectiveness for varying magnitudes of typical data quality issues in three event-based time-series. To the best of our knowledge, this work represents the first comparison of time delay estimation methods for event-based time-series data. Additionally, the NAd algorithm is evaluated for the challenge of non-linear clock drift correction, introduces advancements in event-specific weightings, and provides a proof-of-concept for measuring the synchronicity of different observation quantities. These contributions address critical gaps found in the existing literature and constitute advancements in the field of time-series analysis.

Section 2 formulates the problem of time delay estimation and clock drift for event-based time-series. Then, Sect. 3 proposes the Nearest Advocate algorithm and its properties. Section 4 presents the three elaborated datasets on which the algorithms were evaluated in Sect. 5. Section 6 presents an empirical evaluation of the Nearest Advocate algorithm for clock drift correction, introduces advancements in event-specific weightings, and provides a proof-of-concept for measuring the synchronicity of different observation quantities. These three problems still pose a challenge for state-of-the-art algorithms. The findings are discussed in Sect. 7 and concluded in Sect. 8.

## 2 Problem formulation

We differentiate three interconnected problems of time synchronization: time offset estimation (i.e. time delay), linear clock drift (i.e. skew) correction, and non-linear clock drift correction, each being a subset of the subsequent. This terminology is

consistent with existing definitions of Meier and Holz [4], Tirado-Andrés and Araujo [7], and Mills [21].

While the basic implementation of the NAd algorithm aims to solve the time offset issue in Sects. 5, 6.1 will demonstrate how it can also be utilized for a clock drift correction. Additionally, Sect. 6.2 will show how to incorporate the events' confidences in NAd using event-specific weighting and Sect. 6.3 illustrates a proof-of-concept of how to process distinct observations.

### 2.1 Time offset estimation

Consider an event-based time-series  $\mathcal{T}_n = \{t_i : t_i \in \mathbb{R} \wedge t_i - t_{i-1} > 0 \forall i = 1, 2, \dots, n\}$ , representing strictly increasing sets, and  $\mathcal{S}_m = \{s_j : s_j \in \mathbb{R} \wedge s_j - s_{j-1} > 0 \forall j = 1, 2, \dots, m\}$ , the measurement of  $\mathcal{T}_n$  with a time delay  $\phi_s$  between the event-based time-series. The relationship between the two series is given by:

$$\forall j \leq m : \exists i \leq n : s_j = t_i + \phi_s + \varepsilon_j \quad (1)$$

Assuming the distribution of errors  $\varepsilon$  is centered around zero, i.e.  $\mathbb{E}[\varepsilon] = 0$ , the time delay  $\phi_s$  can be estimated for  $m = n$  and  $i = j$  using the Mean Absolute Error (MAE):

$$\hat{\phi}_s = \arg \min_{\phi \in \mathbb{R}} \sum_{i=1}^n |s_i - t_i - \phi| \quad (2)$$

In cases where events in  $\mathcal{S}$  are undetected or unmatched, this estimation becomes invalid. Assuming  $m \leq n$  and a known injective and strictly monotonically increasing mapping function between events  $f : j \mapsto i$ , the time delay  $\phi$  can be estimated by:

$$\hat{\phi}_s = \arg \min_{\phi \in \mathbb{R}} \sum_{j=1}^m |s_j - t_{f(j)} - \phi| \quad (3)$$

However, the mapping function  $f$  is not known in practical scenarios, complicating the time delay estimation problem. To address this, we propose the NAd algorithm, which operates independently of the mapping function  $f$ .

### 2.2 Linear clock drift

There is a linear clock drift between two event-based time-series, if there is a function  $f : \mathcal{S} \rightarrow \mathcal{R}$  such that  $s_j \mapsto (1 + \alpha)s_j + \phi + \varepsilon_j \quad \forall j = 1, 2, \dots, m$  with  $\alpha \in (-1, \infty)$ . There is no clock drift if  $\alpha = 0$ , but only a constant time offset  $\phi$ .

### 2.3 Non-linear clock drift

There is a non-linear clock drift between two event-based time-series, if there is a non-linear monotonically increasing function  $f : \mathcal{S} \rightarrow \mathcal{R}$ , i.e.  $\forall f(s_j) \geq f(s_i) \Rightarrow s_j > s_i$ .

In many practical scenarios, the non-linear portion of clock drift is negligible, especially for short measurements at nearly constant temperatures [7]. Therefore, the need for the correction of non-linear clock drifts has to be assessed. E.g., our measurements spanning up to eight hours revealed that the non-linearity's magnitude was about a second, while the linear part reached up to a minute.

### 3 Nearest advocate algorithm

#### 3.1 Algorithm

The Nearest Advocate (NAd) algorithm, based on a zipper principle adapted from a stream-stream join algorithm [22], estimates the time delay between two event-based time-series data. The core concept involves calculating the distance between each event in one time-series and the nearest event (i.e., advocate) in another time-series for a given time offset  $\phi$ . The average of these distances indicates the synchrony between the two time-series.

The method estimates the time delay between two event-based time-series by calculating the synchrony measure for multiple potential time offsets in a one-dimensional search space,  $\mathcal{T} = \{\phi : \phi \in \mathbb{R}\}$ . The evaluation of a specific time delay is the inner part of the algorithm, while looping through the search space  $\mathcal{T}$  is the outer part. After skipping leading reference events (see Pseudocode 1, phase 1), the inner part is based on the zipper principle to match each test event with the nearest advocate event (see pseudocode 1, phase 3). Leading and trailing test events must be considered separately (see pseudocode 1, phases 2 and 4, respectively). A simplified depiction of the inner part is shown in Algorithm 1, with the full code available at <https://github.com/iot-salzburg/nearest-advocate>.

**Algorithm 1** Nearest Advocate, inner part

---

```

Require:  $a_r$  array of sorted reference timestamps
Require:  $a_s$  array of sorted test timestamps, shifted by a time offset to evaluate
Require:  $d_{\max} > 0$  maximal accepted distance between advocate events
1:  $i_r \leftarrow 0$  // index for the reference array
2:  $i_s \leftarrow 0$  // index for the test array
3:  $d_{\text{cum}} \leftarrow 0$  // cumulative distance of advocate events
4: // Phase 1: Skip indices in reference array until first matching
5: while  $i_r + 1 < |a_r| \wedge a_r[i_r + 1] \leq a_s[i_s]$  do
6:    $i_r = i_r + 1$ 
7: end while
8: if  $i_r + 1 < |a_r|$  then
9:   // Phase 2: Match leading test events
10:  while  $i_s < |a_s| \wedge a_s[i_s] < a_r[i_r]$  do
11:    // Calculate distance to the nearest (advocate) event:
12:     $d_{\text{cum}} \leftarrow d_{\text{cum}} + \min(a_r[i_r - a_s[i_s]], d_{\max})$ 
13:     $i_s = i_s + 1$ 
14:  end while
15:  // Phase 3: Regular matching case
16:  while  $i_s < |a_s|$  do
17:    if  $a_s[i_s] < \text{last}(a_r)$  then
18:      while  $a_r[i_r + 1] \leq a_s[i_s]$  &  $i_r < |a_r| - 1$  do
19:         $i_r \leftarrow i_r + 1$  // forward reference until invariant holds
20:      end while
21:      if  $i_r < |a_r|$  then
22:        // Invariant:  $a_r[i_r] \leq a_s[i_s] < a_r[i_r + 1]$ 
23:        // Calculate distance to the nearest (advocate) event:
24:         $\text{distance} \leftarrow \min(a_s[i_s] - a_r[i_r], a_r[i_r + 1] - a_s[i_s], d_{\max})$ 
25:         $d_{\text{cum}} \leftarrow d_{\text{cum}} + \text{distance}$ 
26:      end if
27:    end if
28:     $i_s \leftarrow i_s + 1$  // forward test array
29:  end while
30: end if
31: // Phase 4: Match trailing test events
32: while  $i_s < |a_s|$  do
33:  // Calculate distance to the nearest (advocate) event:
34:   $d_{\text{cum}} \leftarrow d_{\text{cum}} + \min(a_s[i_s - \text{last}(a_r)], d_{\max})$ 
35:   $i_s = i_s + 1$ 
36: end while
37: return  $d_{\text{cum}} / |a_s|$  // mean distance of advocate events

```

---

In the first phase, leading reference events are skipped by incrementing the reference array indices until the first test element lies between two reference events, expressed by the inequalities  $a_r[i_r] \leq a_s[i_s] < a_r[i_r + 1]$ . In this phase the cumulative distance  $d_{\text{cum}}$  is not incremented.

The second phase processes leading test events. This is relevant when the test time-series start precedes the reference time-series. Each test event is matched with the first reference element as long as the respective invariant holds.

In the third phase, the loop invariant  $a_r[i_r] \leq a_s[i_s] < a_r[i_r + 1]$  is exploited and maintained. In each iteration, the average distance between each event in the test array and its nearest counterpart in the reference array is calculated and cumulated.

The final phase processes trailing test events, a similar case to the second phase but for trailing events instead of leading ones. Finally, the mean of all matched distances is returned, given by the fraction of cumulative distance and the cardinality of the test array.

This inner algorithm has a linear runtime complexity. For the overall time delay estimation, the inner part is evaluated for multiple time offsets  $\phi \in \mathcal{T}$  which are by default equally spaced within a given interval. Therefore, the runtime complexity is  $|\mathcal{T}| \cdot (|a_r| + |a_s|)$ .

The granularity, i.e. the interval between these evaluated time offsets  $\mathcal{T}$  affects the quality of the characteristic curve. This curve oscillates approximately with the maximal event frequency of the reference and test time-series. To achieve a characteristic curve that is not aliased and suitable for accurately detecting its minimum—which serves as the time offset estimator—it is advisable to use a frequency at least ten times higher than the expected Nyquist frequency [23].

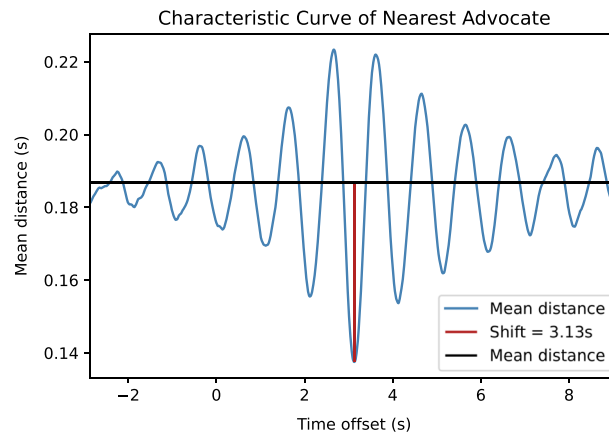
### 3.2 Characteristic synchronisation curve

In Fig. 2, the characteristic curve of the NAd method is shown in blue. The x-axis depicts the evaluated time delay  $\phi \in \mathbb{R}$ , where the detected offset is marked as a vertical red line in the proximity of the true time delay  $\pi$ . The y-axis shows the NAd synchrony measure with the black horizontal line representing the mean over all time shifts. It can be seen that the curve oscillates noticeably in proximity to the estimated time shift.

*The global minimum of the characteristic curve constitutes the estimator of the test array's time delay* The mean distance metric increases during test array shifting and reaches a maximum when the majority of test events are approximately halfway between the reference events. A secondary (or minor) minimum occurs when the shift is one mean inter-event interval (or a multiple of it), but it's less significant due to the inherent pattern's variability. Increased variability of event intervals, like heart rate variability or varying running speeds, improves the time delay estimation's accuracy by dampening local minima.

### 3.3 Nearest advocate as a metric space

The NAd algorithm is proposed as a novel approach to address the challenge of time delay estimation in event-based time-series. This section investigates under which constraints the NAd's mean distance measure constitutes a metric.



**Fig. 2** Characteristic curve of NAd, global minimum indicates the time offset

### 3.3.1 Definition of a metric space

Formally, a metric space is an ordered pair  $(M, D)$  where  $M$  is a set and  $D$  is a function  $D : M \times M \rightarrow \mathbb{R}$  satisfying the following axioms for all elements  $x, y, z \in M$ :

1. **Positive definiteness** The distance from a point to any other point is non-negative  $D(x, y) \geq 0$  and the distance is only zero if and only if the two elements are equal:  $D(x, y) = 0 \iff x = y$ .
2. **Symmetry** The distance from  $x$  to  $y$  is always the same as the distance from  $y$  to  $x$ :  $D(x, y) = D(y, x)$ .
3. **Triangle Inequality** The distance between two elements is always less than or equal to the sum of the distances of a third element from each of these points:  $D(x, z) \leq D(x, y) + D(y, z)$

### 3.3.2 Application to nearest advocate

Here we want to show if (or under which constraints)  $(M, D_{\text{NAd}})$  is a metric space, where  $M = \{\mathcal{T} \subset \mathbb{R} : \mathcal{T} = \{t_i : t_i \in \mathbb{R} \wedge t_i - t_{i-1} > 0 \forall i \leq n\} \subset \mathcal{P}(\mathbb{R})$  and  $\mathcal{X}, \mathcal{Y}, \mathcal{Z} \in M$  are sets with strict increasing order with their elements in  $\mathbb{R}$ . The function  $D_{\text{NAd}} : M \times M \rightarrow \mathbb{R}$  is the inner part of the NAd algorithm as illustrated in the pseudocode 1 depended of it's single parameter  $d_{\text{max}} > 0$ .

### 3.3.3 Symmetric nearest advocate

Similarly to the Kullback–Leibler Divergence [24], the NAd requires also a symmetric version which is simply constructed by averaging the function with both orders of their arguments.

$$D_{\text{NAd-symmetric}}(\mathcal{X}, \mathcal{Y}) := \frac{1}{2}(D_{\text{NAd}}(\mathcal{X}, \mathcal{Y}) + D_{\text{NAd}}(\mathcal{Y}, \mathcal{X}))$$

To shorten the notation we call  $D_{\text{NAd-symmetric}} = D$  and for some cases, we want to emphasize the single parameter  $d_{\text{max}}$  with a subscript  $D_{d_{\text{max}}}$ .



### 3.3.4 Theorem

The Symmetric NAd  $D_{d_{\max}}$  over  $M_{\Delta t_{\min}} = \{\mathcal{T} \subset \mathbb{R} : \mathcal{T} = \{t_i : t_i \in \mathbb{R} \wedge t_i - t_{i-1} \geq \Delta t_{\min} \forall i = 1, 2, \dots, n\}\}$  is a metric space for all  $\Delta t_{\min} > 0$  if the inequality  $d_{\max} \leq \Delta t_{\min}$  holds.

Specifically, the following properties can be shown:

1.  $(M, D_{\text{NAd}})$  is **not positive definite**: We show that the non-symmetric NAd distance function is in general not positive definite.
2.  $(M, D)$  is **positive-definite**: We show that the Symmetric NAd distance function is positive-definite.
3.  $(M, D_{\text{NAd}})$  is **not symmetric**: We show that the symmetry property does not hold in general.
4.  $(M, D)$  is **symmetric**: We show that the Symmetric NAd distance function is symmetric.
5. **Contradiction of the Triangle Inequality for  $(M, D_{\text{NAd}})$** : We show that the NAd distance function  $D_{\text{NAd}}$  does not satisfy the triangle inequality in general.
6. **Proof of the Triangle Inequality for  $(M_{\Delta t_{\min}}, D_{\text{NAd}})$** : We prove that under a certain constraint, the NAd distance function  $D_{\text{NAd}}$  satisfies the triangle inequality.

Thus, we conclude that  $(M_{\Delta t_{\min}}, D_{d_{\max}})$  constitutes a metric space. The proofs for all statements are provided in the ‘‘Appendix A.’’ Additionally, simulations are provided in the same appendix and under [https://github.com/iot-salzburg/nearest-advocate/blob/main/experiments/metric\\_proof\\_simulation.ipynb](https://github.com/iot-salzburg/nearest-advocate/blob/main/experiments/metric_proof_simulation.ipynb) for an empirical demonstration of the theorem.

### 3.4 Implementation details

The NAd implementation sets the maximum distance  $d_{\max}$  to a quarter of the median inter-event interval of the reference array ( $\text{median}(\text{diff}(a_r))$ ). The search space ranges from -300 s to 300 s, with increments of 0.1 s each. Various NAd versions (‘NAd-dense’, ‘NAd-sparse10’, and ‘NAd-sparse100’) with different *sparse\_factor* parameters were implemented to balance precision and computational efficiency. The method was implemented in *Python 3.10* and the just-in-time (JIT) compiler *Numba 0.55* [25] to optimize performance.

Both Cross-Correlation methods utilized the *scipy* package (version 1.7.3), while the Dynamic Time Warping (DTW) method employed the *dtw-python* package (version 1.3.0) [19].

## 4 Evaluation

The previously described methods (PCC, KCC, DTW, and NAd) are compared using three distinct datasets to evaluate their efficacy in diverse applications characterized by variable inter-event intervals and long-term event patterns. Each dataset comprises pairs of events, denoted by their timestamps:

1. Heartbeat (HB) Dataset: The Heartbeat (HB) Dataset consists of two series of R-peaks from the characteristic QRS-complex in electrocardiogram (ECG) signals,



collected from 17 participants during sleep as part of the Virtual Sleep Lab project [26]. The reference series was captured using the laboratory-standard *BrainVision BrainAmp ExG* polysomnograph (Brain Products GmbH, Germany), and the comparative series was recorded with the *Movesense HR+* (Suunto, Finland) sensor. Figure 1 illustrates two ECG signals with a relative time delay and their corresponding R-peaks as event timestamps. Each session lasted approximately eight hours with an average inter-event interval (inter-beat interval (IBI)) of 1.04 s.

2. **Breath Rate (BR) Dataset:** This dataset comprises breathing data recorded during running using a custom smart textile chest sensor [27]. A custom flow reversal detection algorithm detected expiration and inspiration events, marking the beginning of the exhalation and inhalation phases, respectively. Additionally, reference flow reversal events were obtained from a spirometry system. Participants were instructed to maintain two different average breathing rates ( $39 \pm 3$  bpm and  $27 \pm 2.4$  bpm). The average inter-event interval was 1.07 s.
3. **Step Rate (SR) Dataset:** Chest-worn inertial measurement units (IMUs) captured acceleration data during the same running experiment as (BR), with a custom algorithm identifying step events [27]. Reference step events were determined using a validated algorithm on data from tibia-mounted IMUs. Participants ran on a treadmill at two different speeds, resulting in average step rates of  $154.2 \pm 10.8$  spm and  $158.4 \pm 10.2$  spm. The average inter-event interval was 0.467 s. Both the BR and SR datasets included 19 pairs of measurements, each with a duration of approximately 42 min.

As no ground truth time delay can be obtained for the given data pairs, the methods are evaluated using simulated, semi-simulated, and real data pairs for a rigorous comparison.

1. Simulated reference event-based time-series data were generated using normally distributed inter-event intervals to assess the effect of time-series lengths on accuracy and runtime. Additionally, the errors of detected events in the test signals can be assumed to be sampled from a stationary normal distribution. Therefore, the test array was cloned from the reference array, with constant Gaussian noise added to each event timestamp.
2. The impact of noise, missing events, and clock drift on time delay estimation accuracy was explored for both reference and test arrays of all three datasets HB, BR, and SR. Assuming a stationary normal distribution of the event detection in the test signal, a standard deviation of approximately 0.1s was found to be typical across the datasets by evaluating against the respective reference signal. Therefore, the default Gaussian noise was set to 0.1s, default clock drift to zero, and the test array length to 1000, before removing a fraction of events.
3. Real heartbeat data (HB) were used to compare methods, after manual investigations of their time delays. To increase the result's robustness, the symmetric NAd was assessed. The impact of length on estimation precision was examined by selecting different test array lengths.

NAd is evaluated with three different sparseness factors, PCC with DFT (discrete FT) and FFT (fast FT), KCC with two triangular kernel widths, and DTW with an asymmetric step pattern. The median mean absolute error (MAE) and runtime for each method and setting were calculated over 25 runs and methods exceeding 3 s were omitted for subsequently increased lengths.

## 5 Results

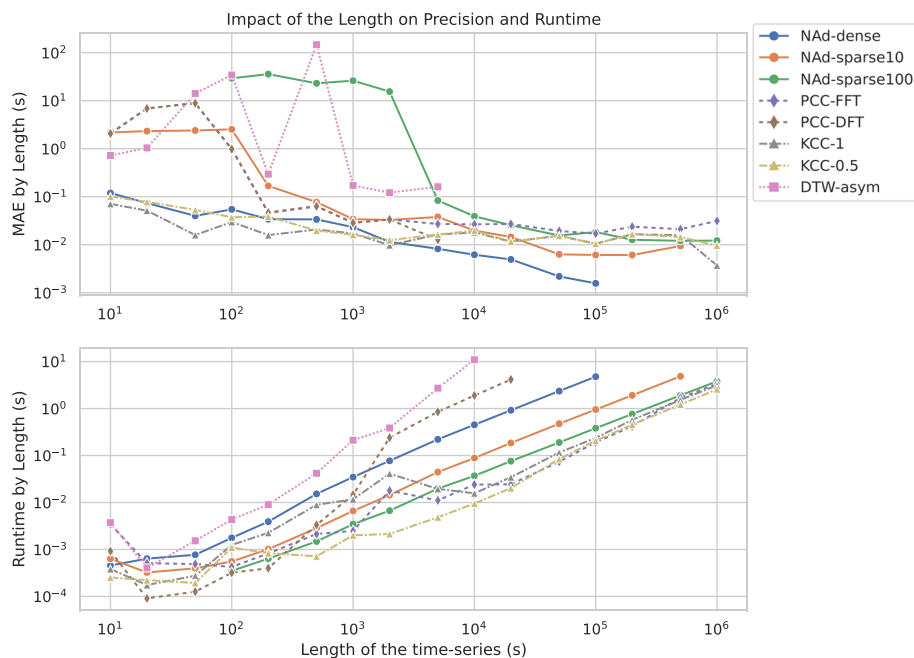
### 5.1 Method comparison on simulated data

Figure 3 compares the methods on simulated data, demonstrating the impact of time-series length on precision and computation time.

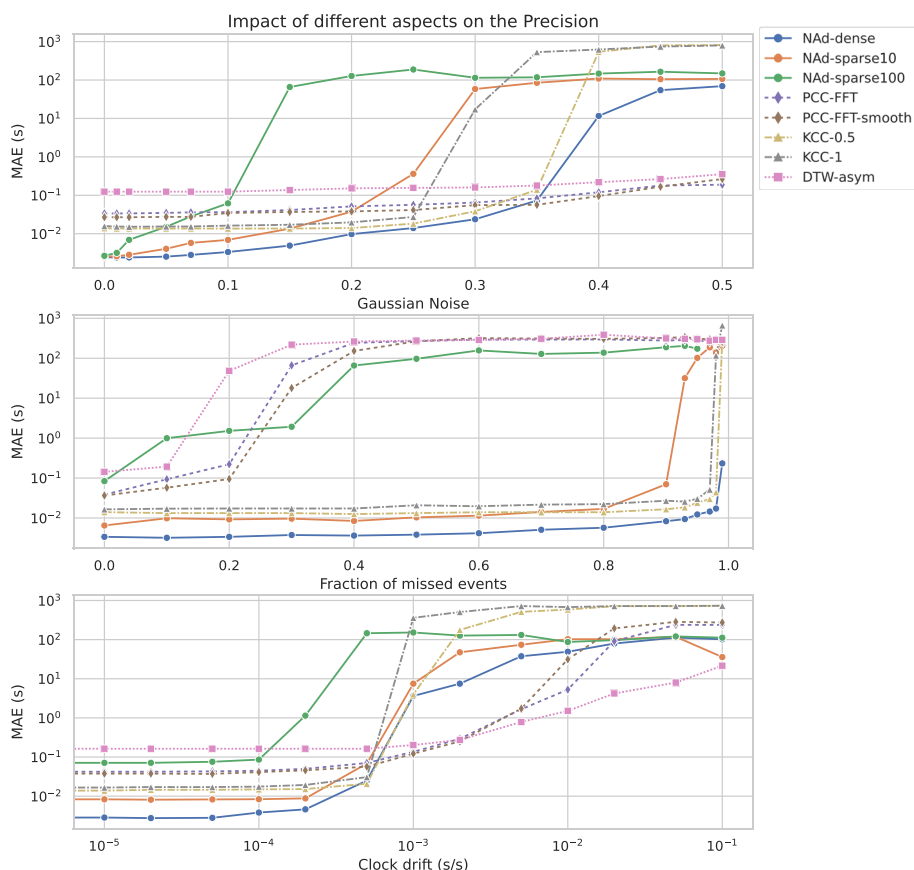
NAd sparsity affects precision and computation time, offering an adjustable tradeoff. PCC-FFT performs faster than PCC-DFT with the same precision. KCC methods show comparable precision to NAd and computational time to PCC-FFT. DTW exhibits unstable results for time-series with less than 1000 events and longer time-series have rapidly increasing runtime complexity.

### 5.2 Method comparison on semi-simulated data

Figure 4 illustrates the effects of varying Gaussian noise, missing event fractions, and linear clock-drift intensities (i.e. skews) on the precision. Each point shows the median MAE of 25 runs. The top plot shows that the MAE increases with the noise magnitude, as noise represents imprecisely detected timestamps. PCC and DTW methods, along with NAd-dense, are robust against noise. The center plot shows that a lower retention fraction leads to lower precision, with NAd and KCC performing well at high missing



**Fig. 3** Impact of time-series length on precision and runtime of time delay estimation methods on simulated data



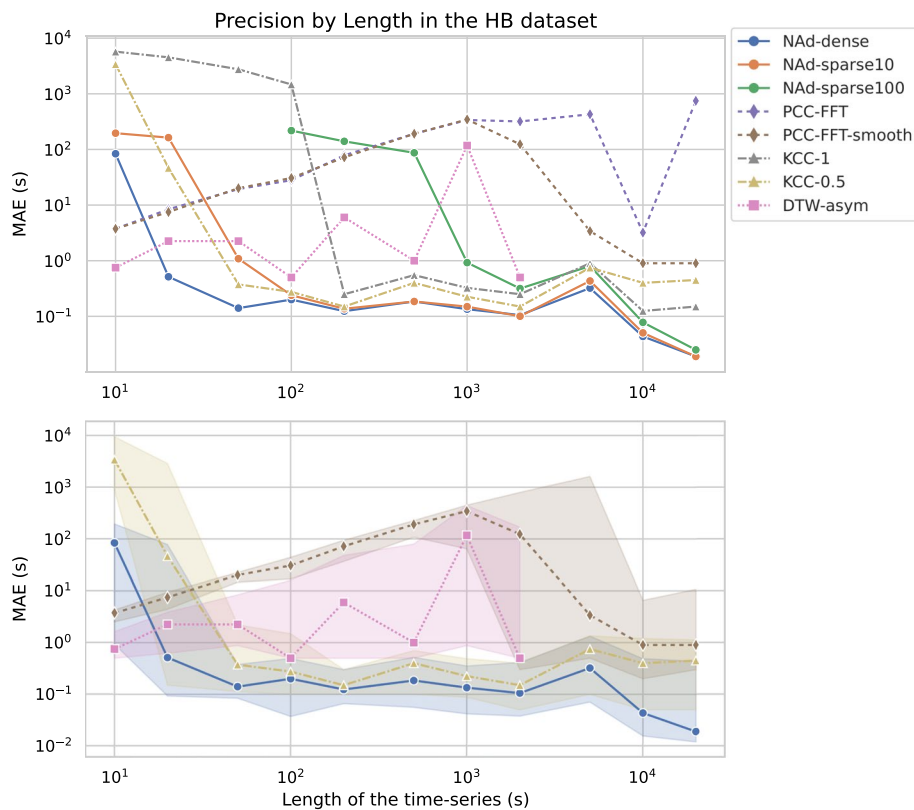
**Fig. 4** Impact of Gaussian noise, missing events, and clock drift on the precision of methods on semi-simulated data based on HB, BR, and SR

event fractions. The bottom plot indicates that higher clock drift results in a larger MAE. PCC and particularly DTW show high robustness against clock drifts.

### 5.3 Method comparison on real data

Figure 5 displays each method’s performance on real HB dataset measurement pairs. A subsequence of variable length is selected from the test array to demonstrate the influence of different lengths of real data pairs on the precision. The upper plot shows median precision as a function of length, while the bottom plot indicates the interquartile interval for the most precise method variant.

It can be seen, that NAd followed by KCC yield precise time delay estimations, even for short sub-sequences. In contrast, PCC estimation is mostly random until a length of about 1000 due to the subsequence length constraint, as indicated by its linear increase. DTW estimates have high variance, and the method’s runtime exceeds 3 s for time-series with more than 2000 events.



**Fig. 5** Impact of the length of real heartbeat dataset pairs on precision

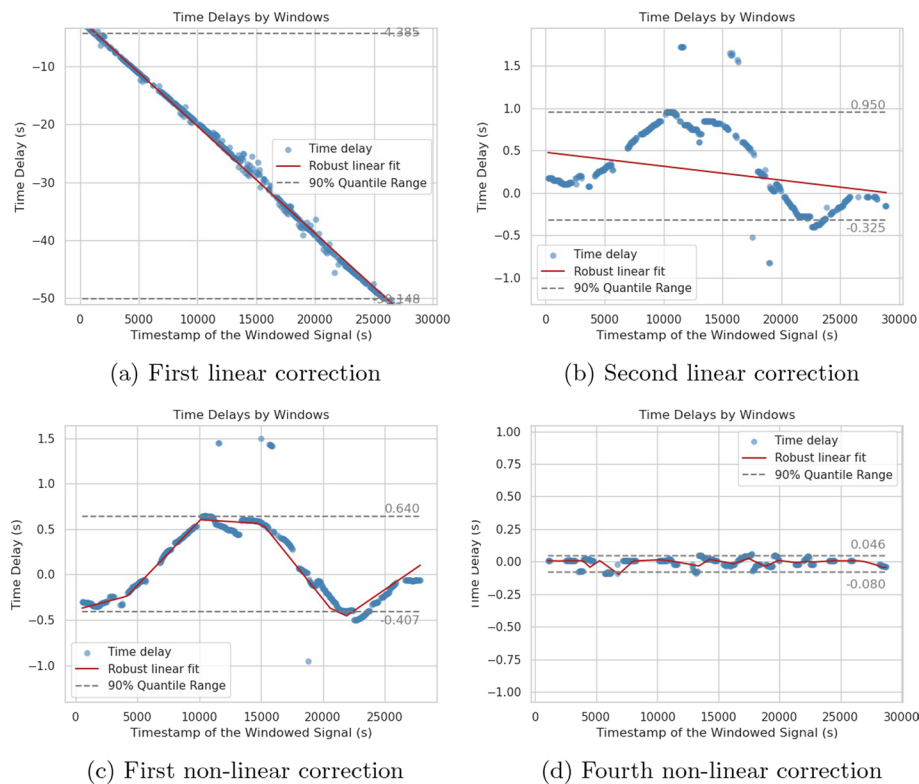
## 6 Applications of nearest advocate

As the NAd algorithm proved its ability for data of low quality and has algorithmical flexibility, this section investigates the application of NAd for various advanced time delay estimation problems.

### 6.1 Linear and non-linear clock drift correction

Besides the application for eliminating constant time offsets, NAd can be applied to detect and solve the clock drift issue between sensor devices. This process exploits the investigated property of NAd to be very precise for short time-series and robust against some degree of clock drift (see Figs. 4 and 5), by applying it on a high number of short (overlapping) windows of the whole time-series. Afterwards, a regression is used to estimate the progress of the clock drift.

For the linear clock drift correction, the robust Theil-Sen estimator [28] is used to detect (1) the optimal time delay ( $\hat{\phi}(t)$ ) per window, (2) the relative factor of the sample rate ( $F_s$ ) deviation ( $\Delta F_s$ ) compared to the reference clock for clock drift removal and (3) the MAE and interquartile width as indicators of the accuracy of the resulting regression. After the regression, the test array's timestamp are adapted based on the estimated  $\hat{\phi}_0$  and  $\Delta F_s$ .



**Fig. 6** Iterative linear and subsequent non-linear corrections of the clock drift using a robust linear regression and MLP regression

The non-linear clock drift requires the estimation of the clock drift course, which is assumed to be continuous. Therefore, a multi-layer perceptron (MLP) with a single input, single output neuron, one hidden layer, and a *ReLU* activation function [29] succeeds in estimating this unknown continuous clock drift function [30]. After each function estimation, the timestamps of the test array are interpolated using the trained function estimator of the MLP.

Figure 6 presents two iterations of a linear and a non-linear clock drift correction, respectively. Each plot depicts a scatterplot of estimated time delays for the individual windows along a respective regression. The interquartile range indicates the accuracy of the correction. Figure 6a depicts a strong linear clock drift: The estimated time offset of each window (blue dot) is aligned approximately on a line, that is calculated by the robust linear regression.

As we can see in Fig. 6b the non-linear clock-drift wanders (nearly continuously) between  $-0.5$ s and  $1.0$ s within the temporal span of about eight hours. This non-linear portion of the clock drift cannot be further corrected by a linear regression. However, in Fig. 6c we can see that the non-linear approach incorporating a MLP for the function approximation can capture the trend and therefore decreases the residuals substantially. After three iterations of non-linear clock drift corrections, as shown in Fig. 6d, the 90% interquartile range was reduced to less than  $0.13$ s, indicating a precise time delay

synchronization. The median residual is  $11ms$ , which is less than half of the median error of  $33.4ms$  reported by Meier and Holz [4] using BMAR.

In experiments, it was shown that one or two linear corrections suffice. Heuristically, starting with a smaller window width in the first iteration is recommended, because narrow windows are more robust against a high magnitude of linear clock drift (skews). A high initial sample rate deviation could lead to incoherence in wide windows.

A weight decay helped successfully to regularize the estimated function. We experienced high slopes in the beginning and end of the function's definition space, as strong neuron activations in this part are hardly regularized. To mitigate this effect, two dummy points were added 20 % outside the range of the times both before and after the measurement with the value of zero. This led to more conservative function estimations at the borders and also allowed extrapolations to some extent. The full implementation is available under [https://github.com/iot-salzburg/nearest-advocate/blob/main/experiments/application\\_nonlinear\\_correction.ipynb](https://github.com/iot-salzburg/nearest-advocate/blob/main/experiments/application_nonlinear_correction.ipynb).

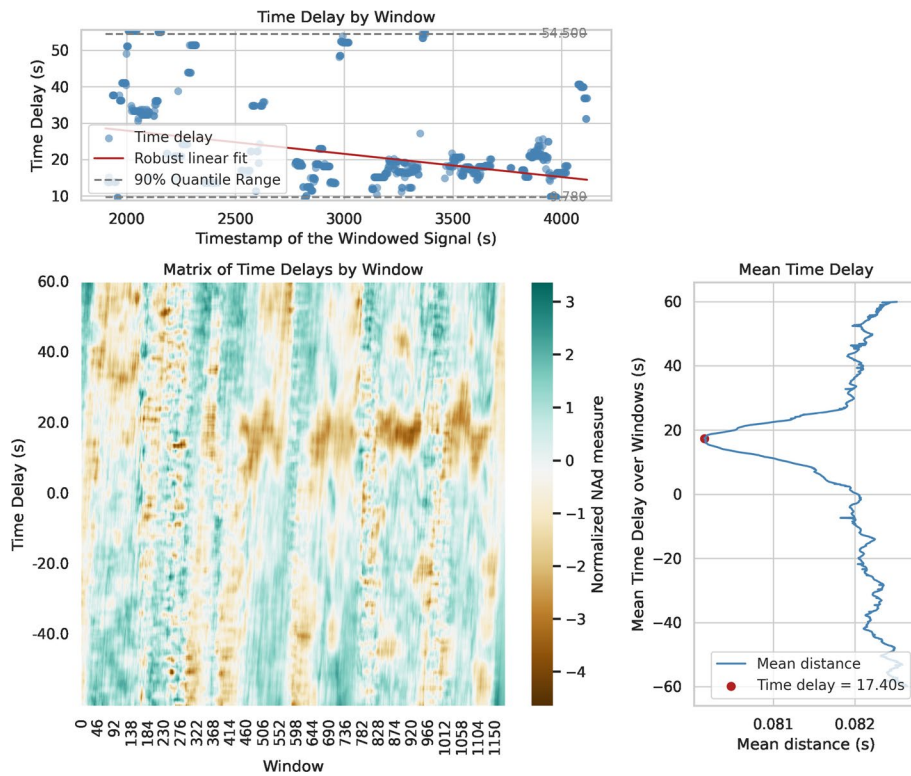
## 6.2 Event-weighted nearest advocate

In many event detection algorithms, some kind of confidence or precision can be assigned to each detected event. In contrast to algorithms based on the Pearson correlation (PCC and KCC), this additional information can be incorporated into NAd. A weighted NAd aims to weigh events that were confidently detected higher while weakening the influence of temporal areas with poor data quality or uncertain event detection. Therefore, any match of test and reference event can be weighted by multiplying their respective distance as proposed in the original algorithm (1) with a predefined weighting factor related to the test event. A pseudocode for the weighted NAd is provided in "Appendix 8".

## 6.3 Synchronicity of different observations quantities

There is a decent interdependency between breath and stride during running, the so-called locomotor-respiratory (LRC) coupling [31]. The LRC-ratio describes the number of strides per breath cycle, e.g. common human LRC-ratios are 2:1, 2.5:1, 3:1, or 4:1 [32, 33]. The BR and SR datasets were acquired simultaneously during the same experiment. Within this experiment, the participants were instructed to accomplish four runs and breathe with LRC ratios of 2:1 and 3:1, i.e., four and six steps per breath cycle, respectively [27]. The authors raised the question of whether it would be possible to use this LRC to estimate the time delay based on breathing and stride events and, therefore on observations of different quantities.

Figure 7 shows the time delay estimation between a breathing and a step event-based time-series. The NAd algorithm was applied with a  $d_{\max}$  of a quarter of the minimal median inter-event difference of the arrays. The basic time delay estimation was performed for multiple highly overlapping windows with a window length of 250 s and a sliding rate of 5 s. Each optimal time delay is illustrated in the upper scatter plot in Fig. 7 as blue point, with an additional robust linear regression showing a linear trend over time.



**Fig. 7** Windowed time delay estimation between breathing and step events: linear regression, heatmap for estimated time offset and window, and mean time offset across offsets

In the heatmap of Fig. 7 the windows' characteristic curves within the search space of  $\pm 60$ s were stacked horizontally for each window. The brown colors in the heatmap matrix indicate lower mean distances of the NAd runs, thus higher synchronicity. Within the heatmap, the four runs can be recognized by four batches of more intense brown between windows 460 and 1104 and around the time offset of 10 to 25 s.

Finally, the respective time delays were aggregated over all windows and summarized in the bottom right plot in Fig. 7. The mean aggregation utilizes the full information of all time delay estimations, instead of using only one optimum for the regression. The resulting mean time delay shows a clear minimum at 17.4s, constituting a robust estimation of the test measurement's relative shift in time. However, this estimation might suffer from a time delay that originates from a relative phase shift between their frequencies.

Using the described process, 22 out of 24 pairs of corresponding BR and SR time-series could be synchronized successfully, emphasizing a successful proof-of-concept of the NAd method for time offset estimation on different observations. The full implementation is available under [https://github.com/iot-salzburg/nearest-advocate/blob/main/experiments/application\\_different\\_observations.ipynb](https://github.com/iot-salzburg/nearest-advocate/blob/main/experiments/application_different_observations.ipynb).



**Table 1** Performance comparison of NAd implementations

Implementation	Mean with std.
Python	213 ms $\pm$ 1.25 ms
Numba	907 $\mu$ s $\pm$ 3.98 $\mu$ s
Cython	905 $\mu$ s $\pm$ 776 ns

## 7 Discussion

### 7.1 Method comparison

The NAd method demonstrates precision and stability in time delay estimation for event-based time-series, excelling when few events are present. With only 1% retained events (10 out of 1000), the method estimates time delays with a median MAE of under 0.5s. Our results are consistent with Vio and Wamsteker [12] that PCC yields unstable results for short time-series.

NAd's drawback is its higher runtime compared to FFT-based cross-correlation methods. However, adjusting the sparseness parameter trades runtime for precision, achieving higher precision and similar runtimes compared to PCC and KCC for long time-series.

Given a number of  $n$  respectively  $m$  events and a search space  $\mathcal{T} = \{\phi : \phi \in \mathbb{R}\}$ , NAd's runtime complexity is  $\mathcal{O}((n + m) \cdot |\mathcal{T}|)$ . Based on interpolated arrays of  $t_{max}$  events, PCC and KCC using FFT are  $\mathcal{O}(t_{max} \cdot \log(t_{max}))$  [13]. NAd's runtime complexity becomes lower than the runtimes of PCC and KCC for long measurements because the search space is independent of the measurement length. DTW has  $\mathcal{O}(n \cdot m)$  complexity [18], making it suitable only for short time-series where it has a significant higher MAE.

### 7.2 Performance test

Table 1 summarizes the performance results of three implementations of the NAd method. To obtain statistically significant results, each implementation was measured seven times. The algorithm was executed 1000 times for both the *Numba* [25] and *Cython* [34] implementations, while it was only executed once for the *Python* implementation for runtime reasons.

The pure *Python* implementation required an execution time more than 200 times longer than the optimized versions, due to the intensive use of increment operations which are not efficiently handled by dynamically typed programming languages like Python. On the other hand, the JIT-compiled version of *Numba* required only marginally more time compared to the pre-compiled C-code generated by the *Cython* compiler.

### 7.3 Applications

In Sect. 6.1, the ability of the NAd method to correct both the linear and also non-linear clock drifts of real pairs of measurements is demonstrated. This is particularly important for real-world applications where clock drift comes in combination with other data quality issues, including imprecisely detected event timestamps and missed events. The median residual of NAd is 11 ms for a non-linear clock-drift, which is less than half of the median error of 33.4 ms reported by Meier and Holz [4] using BMAR.

Typically, existing synchronization algorithms prioritize minimizing error over reducing power consumption. The power efficiency, has not been extensively assessed in prior research, especially in the context of online synchronization algorithms. Our method offers a distinct advantage for wearable devices. These devices regularly collect events that are either extracted from time-series signals or detected directly on the embedded device, which further reduces power-intensive data traffic. Consequently, our approach does not incur extra power usage during operation and may even offer a promising possibility to minimize data transfer. Moreover, it is well suited for use with devices not specifically designed to function within the same wireless network.

Additionally, the algorithmic flexibility of NAd allows to weight events separately and thereby, allowing the incorporation of the event detection algorithm's confidence into the process of time delay estimation. Furthermore, even events of different but dependent observations can be processed, thus emphasizing the robustness of this method against poor data quality.

#### 7.4 Limitations

Experiments in this study focus on event-based time-series data which requires transforming event time-series into continuous signals for methods like PCC. Future research could compare methods on continuous signals, whereby NAd is applied on events detected in those signals to make the experiment setup more challenging and broaden its area of application.

Further investigation of time-series properties affecting NAd precision and stability is needed. Time-domain features like entropy or autocorrelation and the power density distribution might explain NAd's oscillation width and magnitude around the estimated time offset and its estimate's confidence interval.

The study's datasets consist of nearly periodic events with similar sampling frequencies. Examining NAd's robustness for non-periodic events and time delay estimation on different types of measurements could further broaden its applicability.

## 8 Conclusion

We proposed NAd, a novel algorithm for synchronizing sensor data across various devices, eliminating the necessity for communication or direct user engagement during runtime while minimizing data traffic. The NAd method offers improved precision in time delay estimation compared to existing methods, particularly for short time-series and those with imprecise or missing events. The sparse variant provides computational efficiency at a slight precision tradeoff. NAd may not be optimal for applications prioritizing runtime but excels in precision for data with quality issues.

Overall, NAd provides a robust and precise solution for time delay estimation in event-based time-series data. The algorithm's efficiency and robustness against outliers make it suitable for time delay estimation in noisy event-based data with challenging event detection, where it can also incorporate the event detection's confidence. It also allows the correction of even non-linear clock drifts and the time delay estimation based on events of different but dependent observations.

## Appendix A: Proof of $(M', D)$ being a metric space

In this appendix, the statements from Sect. 3.3 are proven.

### A.1 $(M, D_{\text{NAd}})$ is not positive definite

Consider sets  $\mathcal{X} = \{1, 2, 3\}$  and  $\mathcal{Y} = \{1, 3\}$ . For any  $d_{\text{max}} > 0$ , from the regular case, i.e. step two, of the Pseudocode 1, we have (it can be analogously shown for the other steps):

$$D_{\text{NAd}}(\mathcal{X}, \mathcal{Y}) = \frac{1}{2}(\min(|1 - 1|, d_{\text{max}}) + \min(|3 - 3|, d_{\text{max}})) = 0$$

Each element in  $\mathcal{Y}$  is matched with its nearest advocate in  $\mathcal{X}$ . Therefore, we find that  $D_{\text{NAd}}(\mathcal{X}, \mathcal{Y}) = 0$  while  $\mathcal{X} \neq \mathcal{Y}$ , contradicting the positive-definiteness of  $(M, D_{\text{NAd}})$ .

### A.2 $(M, D)$ is positive-definite

**Showing  $D(\mathcal{X}, \mathcal{Y}) \geq 0$  indirectly**

Suppose  $\exists \mathcal{X}, \mathcal{Y} \in M : D(\mathcal{X}, \mathcal{Y}) < 0$

$$\begin{aligned} & \text{w.l.o.g.} \Rightarrow D_{\text{NAd}}(\mathcal{X}, \mathcal{Y}) < 0 \\ & \text{dist}_{\text{cum}} < 0 \text{ (as the pseudocode 1)} \end{aligned}$$

We show the matching Case 3 as it is the most complex one and the other can be shown analogously. The invariant we have in this case is  $x_i \leq y_j < x_{i+1}$ :

As we know  $\text{dist}_{\text{cum}}$  is initialized with 0, there must exist at least one loop iteration for which holds:

$$x_i \in \mathcal{X}, y_j \in \mathcal{Y} : \text{distance} = \min(y_j - x_i, x_{i+1} - y_j, d_{\text{max}}) < 0$$

Case  $y_j - x_i < 0 \wedge x_{i+1} - y_j < 0$  contradicts the loop invariant of the matching Case 3.

Case  $d_{\text{max}} < 0$  contradicts the definition.

Therefore,  $D(\mathcal{X}, \mathcal{Y}) \geq 0$  holds  $\forall \mathcal{X}, \mathcal{Y} \in M, \forall d_{\text{max}} \in \mathbb{R}$ .

**Showing  $\mathcal{X} = \mathcal{Y} \Rightarrow D(\mathcal{X}, \mathcal{Y}) = 0$**

$$\mathcal{X} = \mathcal{Y} \Rightarrow \forall y_j \in \mathcal{Y} : \exists x_i \in \mathcal{X} : x_i = y_j$$

Therefore, in each loop iteration  $\text{distance} = \min(y_j - x_i, x_{i+1} - y_j, d_{\text{max}}) = 0$

$$\Rightarrow \text{dist}_{\text{cum}} = 0 \Rightarrow D_{\text{NAd}}(\mathcal{X}, \mathcal{Y}) = 0 \Rightarrow D(\mathcal{X}, \mathcal{Y}) = 0$$

**Showing  $D(\mathcal{X}, \mathcal{Y}) = 0 \Rightarrow \mathcal{X} = \mathcal{Y}$  indirectly**

Suppose  $\mathcal{X} \neq \mathcal{Y}$

$$\text{w.l.o.g.} \Rightarrow \exists y_j \in \mathcal{Y} : \forall x_i \in \mathcal{X} : x_i \neq y_j$$

As we have the invariant  $x_i \leq y_j < x_{i+1}$ , it follows:

$$\begin{aligned} \Rightarrow \text{distance} &= \min(y_j - x_i, x_{i+1} - y_j, d_{\text{max}}) > 0 \\ &\Rightarrow D_{\text{NAd}}(\mathcal{X}, \mathcal{Y}) = 0 \Rightarrow D(\mathcal{X}, \mathcal{Y}) = 0 \end{aligned}$$

Therefore,  $D$  is symmetric for all  $\mathcal{X}, \mathcal{Y} \in M$

### A.3 ( $M, D_{\text{NAd}}$ ) is not symmetric

Let  $\mathcal{X} = \{1, 2, 3\}$ ,  $\mathcal{Y} = \{1, 3\}$ , then for any  $d_{\text{max}} > 0$  :

$$\begin{aligned} D_{\text{NAd}}(\mathcal{X}, \mathcal{Y}) &= \frac{1}{2}(\min(|1-1|, d_{\text{max}}) + \min(|3-3|, d_{\text{max}})) = 0 \\ D_{\text{NAd}}(\mathcal{Y}, \mathcal{X}) &= \frac{1}{3}(\min(|1-1|, d_{\text{max}}) + \min(2-1, 3-2, d_{\text{max}}) \\ &\quad + \min(|3-3|, d_{\text{max}})) = \frac{1}{3} \min(1, d_{\text{max}}) > 0 \end{aligned}$$

$D_{\text{NAd}}(\mathcal{X}, \mathcal{Y}) \neq D_{\text{NAd}}(\mathcal{Y}, \mathcal{X})$  contradicts the symmetry property.

### A.4 ( $M, D$ ) is symmetric

$\forall \mathcal{X}, \mathcal{Y} \in M, \forall d_{\text{max}} > 0$  :

$$\begin{aligned} D(\mathcal{X}, \mathcal{Y}) &:= \frac{1}{2}(D_{\text{NAd}}(\mathcal{X}, \mathcal{Y}) + D_{\text{NAd}}(\mathcal{Y}, \mathcal{X})) \\ &= \frac{1}{2}(D_{\text{NAd}}(\mathcal{Y}, \mathcal{X}) + D_{\text{NAd}}(\mathcal{X}, \mathcal{Y})) \\ &= D(\mathcal{Y}, \mathcal{X}) \end{aligned}$$

Therefore, the symmetric Nearest Advocate distance  $D$  is symmetric over  $M$ .

### A.5 Contradiction of the Triangle Inequality for ( $M, D_{\text{NAd}}$ )

Let  $\mathcal{X} = \{1, 2\}$ ,  $\mathcal{Y} = \{2.1, 2.9\}$ ,  $\mathcal{Z} = \{3, 4\}$ , then for any  $d_{\text{max}} \geq 2$  :

$$\begin{aligned} 2 \cdot D_{\text{NAd}}(\mathcal{X}, \mathcal{Z}) &= 2 + 1 = 3.0 \\ 2 \cdot D_{\text{NAd}}(\mathcal{X}, \mathcal{Y}) &= 0.1 + 0.9 = 1.0 \\ 2 \cdot D_{\text{NAd}}(\mathcal{Y}, \mathcal{Z}) &= 0.1 + 1.1 = 1.2 \end{aligned}$$

$D_{\text{NAd}}(\mathcal{X}, \mathcal{Z}) > D_{\text{NAd}}(\mathcal{X}, \mathcal{Y}) + D_{\text{NAd}}(\mathcal{Y}, \mathcal{Z}) \quad \forall d_{\text{max}} \geq 2$  contradicts the triangle inequality.

Analogously, we can show that for this case  $D_{\text{NAd}}(\mathcal{Z}, \mathcal{X}) > D_{\text{NAd}}(\mathcal{Z}, \mathcal{Y}) + D_{\text{NAd}}(\mathcal{Y}, \mathcal{X})$  also does not hold and therefore the triangle equation is violated for the symmetric  $D$ .

However, we see that we could constrain  $D(d_{\text{max}})$  such that this equation could still hold for a given  $M$ .

### A.6 Proof of the Triangle Inequality for ( $M', D_{\text{NAd}}$ )

The triangle inequality is demonstrable when  $M$  is restricted to  $M' = M_{\Delta t_{\text{min}}}$ , defined as  $\{\mathcal{T} \subset \mathbb{R} : \mathcal{T} = \{t_i : t_i \in \mathbb{R} \wedge t_i - t_{i-1} > \Delta t_{\text{min}} \forall i = 1, 2, \dots, n\} \subset \mathcal{P}(\mathbb{R})$ , a set with strictly ascending order and a minimum distance of  $\Delta t_{\text{min}} > 0$  between elements.

We will show that

$$\begin{aligned} \forall \Delta t_{\text{min}} : \exists d'_{\text{max}} : (D_{\text{NAd}}(d_{\text{max}}), M_{\Delta t_{\text{min}}}) &\text{ is a metric space} \\ \forall d_{\text{max}} \leq d'_{\text{max}} &= \Delta t_{\text{min}}/2 \end{aligned}$$

**Lemma 1** *In order to prove the triangle inequality, we first need to show that for any pair of added elements:*

$$\forall \Delta t_{\min} : \exists d'_{\max} : \\ D_{\text{NAd}}(\mathcal{X}_{n+1}, \mathcal{Y}_{n+1}) = \frac{1}{n+1} (n \cdot D_{\text{NAd}}(\mathcal{X}_n, \mathcal{Y}_n) + |y_{n+1} - x_{n+1}|)$$

**Proof of Lemma 1** This is the case if every  $y_{n+1}$  is only matched with  $x_{n+1}$ . As this should hold for newly added elements regardless of their order in the respective sets, we notate these elements by  $x_j$  and  $y_i$ . W.l.o.g., we prove Lemma 1 by showing  $x_j - y_i \leq y_{i+1} - x_j \quad \forall y_i \leq x_j < y_{i+1}$  by constraining the differences of two measurements over their true event timestamps:

$$(y_{i+1} - x_j) - (x_j - y_i) = (y_{i+1} - t_{l+1} + \Delta t_{\min} + t_l - x_j) \\ - (x_j - t_l + t_l - y_i)$$

We use again the distance function  $d(x_i, y_i) := \min(x_i - y_i, d_{\max}) \geq 0$  (which is not symmetric).

$$(y_{i+1} - x_j) - (x_j - y_i) \\ = d(y_{i+1}, t_{l+1}) + \Delta t_{\min} + d(t_l, x_j) - d(x_j, t_l) - d(t_l, y_i) \\ \geq \Delta t_{\min} - d(x_j, t_l) - d(t_l, y_i)$$

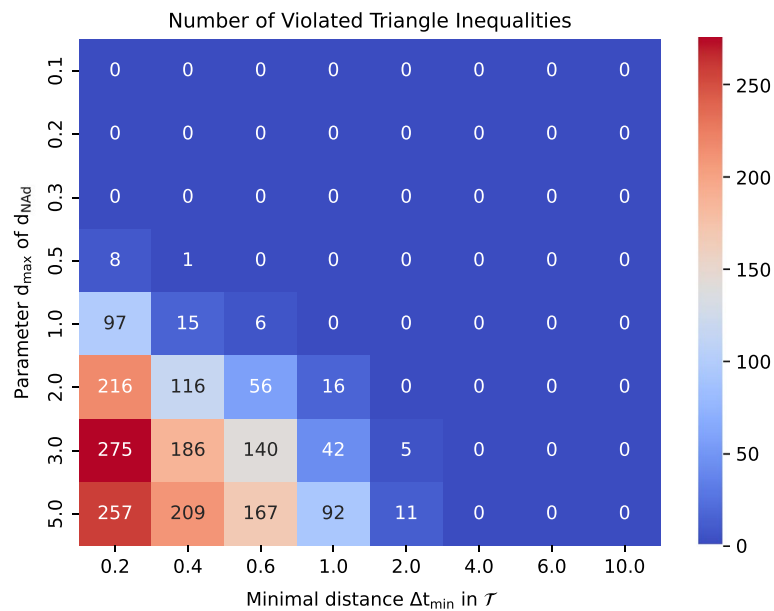
This term can be limited by either constraining  $d_{\max}$  or the difference between  $|x_j - t_l|$  resp.  $|y_i - t_l|$  with an upper threshold  $\varepsilon + |\phi|$  where  $\varepsilon$  is the maximal error in measuring the true timestamp and  $\phi$  the time delay. Then two limits follow:

$$(y_{i+1} - x_j) - (x_j - y_i) \geq \Delta t_{\min} - 2d_{\max} \geq 0 \\ \forall d_{\max} \leq \Delta t_{\min}/2 \\ (y_{i+1} - x_j) - (x_j - y_i) \geq \Delta t_{\min} - 2(\varepsilon + |\phi|) \geq 0 \\ \forall 2(\varepsilon + |\phi|) \leq \Delta t_{\min}/2$$

Therefore, Lemma 1 holds  $\forall d_{\max} \leq \Delta t_{\min}/2$ . The second limit is of theoretical interest as it shows that the triangle inequality holds for any  $\Delta t_{\min}$  if an arbitrarily high data quality and with already corrected time delay  $\phi$  can be achieved.

*Proof of the Triangle Inequality using Lemma 1*

$$D_{\text{NAd}}(\mathcal{X}_{n+1}, \mathcal{Y}_{n+1}) + D_{\text{NAd}}(\mathcal{Y}_{n+1}, \mathcal{Z}_{n+1}) \\ = \frac{1}{n+1} (n \cdot D_{\text{NAd}}(\mathcal{X}_n, \mathcal{Y}_n) + n \cdot D_{\text{NAd}}(\mathcal{Y}_n, \mathcal{Z}_n) \\ + |y_{n+1} - x_{n+1}| + |z_{n+1} - y_{n+1}|) \\ \geq \frac{1}{n+1} (n \cdot D_{\text{NAd}}(\mathcal{X}_n, \mathcal{Z}_n) + |z_{n+1} - x_{n+1}|) \\ = D_{\text{NAd}}(\mathcal{X}_{n+1}, \mathcal{Z}_{n+1})$$



**Fig. 8** Impact of the parameters  $d_{\max}$  and  $\Delta t_{\min}$  on the number of violated triangle inequalities from  $10^5$  runs each

From now on, it is trivial to show that this also holds for the symmetric Nearest Advocate.

#### A.7 Conclusion of the shown statements

The properties of positive-definiteness, symmetry, and triangle inequality have been established, confirming that  $(D_{d_{\max}}, M_{\Delta t_{\min}})$  is a metric.  $\square$

#### A.8 Empirical Simulation

As the metric space  $(D_{d_{\max}}, M_{\Delta t_{\min}})$  depends on its parameters  $d_{\max}$  and  $\Delta t_{\min}$ , it is of practical interest to empirically show their influence of them on the fraction of violated triangle inequalities. In Fig. 8 the number of violated triangle inequalities for  $10^5$  runs of each different parameter configuration is shown as a heatmap. The arrays have equal lengths for between two to 40 randomly sampled elements and randomly sampled  $\varepsilon$  and  $\phi$ . The diagonal represents the constraint  $d_{\max} = \Delta t_{\min}/2$ , thus all entries in and above the diagonal must be zero.

### Appendix B: Event-weighted nearest advocate

This pseudocode depicts the changes in teal necessary to weight individual events of the test array with a given weight.

**Algorithm 2** Weighted Nearest Advocate, inner part

---

**Require:**  $a_r$  array of sorted reference timestamps  
**Require:**  $a_s$  array of sorted test timestamps, shifted by a time offset to evaluate  
**Require:**  $w_s$  weights of the test array's elements with  $|w_s| = |a_s|$   
**Require:**  $d_{\max} > 0$  maximal accepted distance between advocate events

```

1:  $i_r \leftarrow 0$  // index for the reference array
2:  $i_s \leftarrow 0$  // index for the test array
3:  $d_{\text{cum}} \leftarrow 0$  // cumulative distance of advocate events
4: // Phase 1: Skip indices in reference array until first matching
5: while  $i_r + 1 < |a_r| \wedge a_r[i_r + 1] \leq a_s[i_s]$  do
6:    $i_r = i_r + 1$ 
7: end while
8: if  $i_r + 1 < |a_r|$  then
9:   // Phase 2: Match leading test events
10:  while  $i_s < |a_s| \wedge a_s[i_s] < a_r[i_r]$  do
11:    // Calculate distance to the nearest (advocate) event:
12:     $d_{\text{cum}} \leftarrow d_{\text{cum}} + w_s[i_s] \min(a_r[i_r - a_s[i_s]], d_{\max})$ 
13:     $i_s = i_s + 1$ 
14:  end while
15:  // Phase 3: Regular matching case
16:  while  $i_s < |a_s|$  do
17:    if  $a_s[i_s] < \text{last}(a_r)$  then
18:      while  $a_r[i_r + 1] \leq a_s[i_s]$  &  $i_r < |a_r| - 1$  do
19:         $i_r \leftarrow i_r + 1$  // forward reference until invariant holds
20:      end while
21:      if  $i_r < |a_r|$  then
22:        // Invariant:  $a_r[i_r] \leq a_s[i_s] < a_r[i_r + 1]$ 
23:        // Calculate distance to the nearest (advocate) event:
24:         $\text{distance} \leftarrow w_s[i_s] \min(a_s[i_s] - a_r[i_r], a_r[i_r + 1] - a_s[i_s], d_{\max})$ 
25:         $d_{\text{cum}} \leftarrow d_{\text{cum}} + \text{distance}$ 
26:      end if
27:    end if
28:     $i_s \leftarrow i_s + 1$  // forward test array
29:  end while
30: end if
31: // Phase 4: Match trailing test events
32: while  $i_s < |a_s|$  do
33:  // Calculate distance to the nearest (advocate) event:
34:   $d_{\text{cum}} \leftarrow d_{\text{cum}} + w_s[i_s] \min(a_s[i_s] - \text{last}(a_r), d_{\max})$ 
35:   $i_s = i_s + 1$ 
36: end while
37: return  $d_{\text{cum}} / |a_s| / \sum_{i_s} w_s[i_s]$  // mean distance of advocate events

```

---

**Acknowledgements**

The authors thank Elisabeth Haeusler for supporting the development of this algorithm and paper as well as Rade Kutil and Mathias Tonis-Schmoigl for their feedback on state-of-the-art methods and valuable feedback on the algorithm.

**Author contributions**

CS conceptualized and implemented the algorithm, designed and performed experiments, and analytical validation. SM and SB conducted the literature review, contributed to algorithm validation, algorithm applications, and assisted in manuscript writing and proofreading. CH validated the algorithm, its analytical proof, and also aided in manuscript writing and proofreading.

**Funding**

This work was thankfully supported by the Austrian Federal Ministry for Climate Action, Environment, Energy, Mobility, Innovation and Technology under Contract No. 2021-0.641.557.

**Availability of data and materials**

The authors ensure the reproducibility of all experiments by providing all code and data on Github under <https://github.com/iot-salzburg/nearest-advocate>. The experiments were run in Python 3.9 on a single core of an AMD Ryzen Threadripper PRO 5975WX processor inside a container environment using the Docker orchestration software with the image 'cschranz/gpu-jupyter' and tag 'v1.4\_cuda-11.0\_ubuntu-20.04', available on Dockerhub.

**Declarations****Competing interests**

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.



Received: 8 December 2023 Accepted: 20 March 2024

Published online: 05 April 2024

## References

1. G. Carter, Coherence and time delay estimation. *Proc. IEEE* **75**, 236–255 (1987)
2. F. Viola, W. Walker, A spline-based algorithm for continuous time-delay estimation using sampled data. *IEEE Trans. Ultrason. Ferroelectr. Freq. Control* **52**, 80–93 (2005)
3. M. Guggenberger, M. Lux, L. Böszörményi, An analysis of time drift in hand-held recording devices. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **8935**, 203–213 (2015)
4. M. Meier, C. Holz, BMAR: barometric and motion-based alignment and refinement for offline signal synchronization across devices. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* **7**(2), 259 (2023). <https://doi.org/10.1145/3596268>
5. D. Bannach, O. Amft, P. Lukowicz, Automatic event-based synchronization of multimodal data streams from wearable and ambient sensors, in *Smart Sensing and Context: 4th European Conference, EuroSSC, Guildford, UK, September 16–18, 2009. Proceedings 4*, Springer **2009**, pp. 135–148 (2009)
6. S. Shabani, A. K. Bourke, A. Muaremi, J. Praestgaard, K. O’Keeffe, R. Argent, M. Brom, C. Scotti, B. Caulfield, L.C. Walsh, An automatic foot and shank IMU synchronization algorithm: proof-of-concept, in *2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pp. 4210–4213 (IEEE, 2022)
7. F. Tirado-Andrés, A. Araujo, Performance of clock sources and their influence on time synchronization in wireless sensor networks. *Int. J. Distrib. Sens. Netw.* **15**(9), 1550147719879372 (2019)
8. J. Cheong, Four ways to quantify synchrony between time series data. <https://towardsdatascience.com/four-ways-to-quantify-synchrony-between-time-series-data-b99136c4a9c9> (2019)
9. M. Ferreira, M. Rodriguez, Exploring time series correlation. <https://www.researchgate.net/publication/369009724> (2023)
10. C.H. Knapp, G.C. Carter, The generalized correlation method for estimation of time delay. *IEEE Trans. Acoust. Speech Signal Process.* **24**, 320–327 (1976)
11. D. Maskell, G. Woods, The estimation of subsample time delay of arrival in the discrete-time measurement of phase delay. *IEEE Trans. Instrum. Meas.* **48**, 1227–1231 (2000)
12. R. Vio, W. Wamsteker, Limits of the cross-correlation function in the analysis of short time series. *Publ. Astron. Soc. Pac.* **113**(779), 86 (2001)
13. J. P. Lewis, Fast normalized cross-correlation (2010). <https://api.semanticscholar.org/CorpusID:2397291>
14. D. Lyon, The discrete Fourier transform, part 6: cross-correlation. *J. Object Technol.* **9**(2), 17 (2010)
15. R.K. Pearson, *Mining Imperfect Data*. Society for Industrial and Applied Mathematics (2005). <http://epubs.siam.org/doi/book/10.1137/1.9780898717884>
16. H. Park, S. Nam, Time-delay estimation using m-band wavelet transform and projection cross-correlation. *Electron. Lett.* **38**(9), 1 (2002)
17. C. Wang, L. Zhang, L. Xie, J. Yuan, Kernel cross-correlator. *Proc. AAAI Conf. Artif. Intell.* **32**, 4179–4186 (2018)
18. E. Keogh, S. Lonardi, C.A. Ratanamahatana, Towards parameter-free data mining, in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 206–215 (ACM, Seattle, 2004). <https://dl.acm.org/doi/10.1145/1014052.1014077>
19. T. Giorgino, Computing and visualizing dynamic time warping alignments in R: the dtw package. *J. Stat. Softw.* **31**(7), 259 (2009)
20. C. Schranz, S. Mayr, Ein neuer algorithmus zur zeitsynchronisierung von ereignis- basierten zeitreihendaten als alternative zur kreuzkorrelation, 9 (2022). <https://zenodo.org/record/7370958>
21. D. Mills, Network time protocol (version 3) specification, implementation and analysis. Technical report (1992)
22. C. Schranz, P. Michael Jeremias, Deterministic time-series joins for asynchronous high-throughput data streams, in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, pp. 1031–1034 (2020)
23. T. Wescott, Sampling: what nyquist didn’t say, and what to do about it (2016). [https://neuron.eng.wayne.edu/auth/ece4330/practical\\_sampling.pdf](https://neuron.eng.wayne.edu/auth/ece4330/practical_sampling.pdf)
24. S. Kullback, R.A. Leibler, On information and sufficiency. *Ann. Math. Stat.* **22**(1), 79–86 (1951). <https://doi.org/10.1214/aoms/1177729694>
25. S.K. Lam, A. Pitrou, S. Seibert, Numba: a LLVM-based python JIT compiler, in *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, ser. LLVM ’15 (Association for Computing Machinery, New York, 2015). <https://doi.org/10.1145/2833157.2833162>
26. S. Kranzinger, S. Baron, C. Kranzinger, D. Heib, C. Borgelt, Generalisability of sleep stage classification based on inter-beat intervals: validating three machine learning approaches on self-recorded test data. *Behaviormetrika* **85**, 1–18 (2023)
27. S. Bernhart, E. Harbour, S. Kranzinger, U. Jensen, T. Finkenzerler, Wearable chest sensor for stride and respiration detection during running. *Springer Nature Sports Engineering* (2023)
28. R. Wilcoxon, A note on the Theil–Sen regression estimator when the regressor is random and the error term is heteroscedastic. *Biom. J.* **40**(3), 261–268 (1998)
29. V. Nair, G.E. Hinton, Rectified linear units improve restricted Boltzmann machines, in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML’10 (Omnipress, Madison, 2010), pp. 807–814
30. K. Hornik, Approximation capabilities of multilayer feedforward networks. *Neural Netw.* **4**, 251–257 (1991)

31. M.A. Daley, D.M. Bramble, D.R. Carrier, Impact loading and locomotor-respiratory coordination significantly influence breathing dynamics in running humans. *PLoS ONE* **8**(8), 1–10 (2013). <https://doi.org/10.1371/journal.pone.0070752>
32. R.B. Banzett, J. Mead, M.B. Reid, G.P. Topulos, Locomotion in men has no appreciable mechanical effect on breathing. *J. Appl. Physiol.* **72**(5), 1922–1926 (1992)
33. D.M. Bramble, D.R. Carrier, Running and breathing in mammals. *Science* **219**(4582), 251–256 (1983)
34. S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D.S. Seljebotn, K. Smith, Cython: the best of both worlds. *Comput. Sci. Eng.* **13**(2), 31–39 (2011)

### **Publisher's Note**

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.