Plato: Approximate Analytics over Compressed Time Series with Tight Deterministic Error Guarantees

Etienne Boursier
ENS Paris-Saclay
eboursie@ens-paris-saclay.fr

Jaqueline J. Brito
University of São Paulo
jjbrito@icmc.usp.br

Chunbin Lin
University of California, San Diego
chunbinlin@cs.ucsd.edu

Yannis Papakonstantinou
University of California, San Diego
yannis@cs.ucsd.edu

ABSTRACT

Plato provides *sound and tight deterministic error guarantees* for approximate analytics over *compressed* time series. Plato supports expressions that are compositions of the (commonly used in time series analytics) linear algebra operators over vectors, along with arithmetic operators. Such analytics can express common statistics (such as correlation and cross-correlation) that may combine multiple time series. The time series are segmented either by fixed-length segmentation or by (more effective) variable-length segmentation. Each segment (i) is compressed by an estimation function that approximates the actual values and is coming from a user-chosen estimation function family, and (ii) is associated with one to three (depending on the case) precomputed error measures. Then Plato is able to provide tight deterministic error guarantees for the analytics over the compressed time series.

This work identifies two broad estimation function family groups. The *Vector Space (VS)* family and the presently defined *Linear Scalable Family (LSF)* lead to theoretically and practically high-quality guarantees, even for queries that combine multiple time series that have been independently compressed. Well-known function families (e.g., the polynomial function family) belong to LSF. The theoretical aspect of "high quality" is crisply captured by the *Amplitude Independence (AI)* property: An AI guarantee does not depend on the amplitude of the involved time series, even when we combine multiple time series. The experiments on four real-life datasets validated the importance of the Amplitude Independent (AI) error guarantees: When the novel AI guarantees were applicable, the guarantees could ensure that the approximate query results were very close (typically 1%) to the true results.

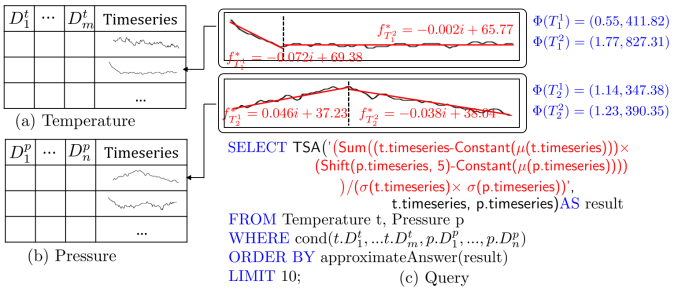


Figure 1: Example of SQL query using the TSA UDF.

1 INTRODUCTION

Attention to time series analytics is bound to increase in the IoT era as cheap sensors can now deliver vast volumes of many types of measurements. The size of the data is also bound to increase. E.g., an IoT-ready oil drilling rig produces about 8 TB of operational data in one day. [1] One way to solve this problem is to increase the expense in computing and storage in order to catch up. However, in many domains, the data size increase is expected to outpace the increase of computing abilities, thus making this approach unattractive [5, 13]. Another solution is *approximate analytics* over compressed time series.

Approximate analytics enables fast computation over historical time series data. For example, consider the database in Figure 1, which has a Temperature table and a Pressure table. Each table contains (i) one Timeseries column containing time series data, as a UDT [11] and (ii) several other "dimension" attributes D, such as geographic locations and other properties of the sensors that delivered the time series. The Plato SQL query in Figure 1(c) "returns the top-10 temperature/pressure 5-second cross-correlation scores among all

[1] https://wasabi.com/storage-solutions/internet-of-things/

Conference'17, July 2017, Washington, DC, USA
2018. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

	function family	error guarantees on aligned time series	AI	Tight	error guarantees on misaligned time series	AI	Tight
$Sum(\boldsymbol{T_1 \times T_2})$	ANY\VS	$\sum_{i=1}^{k}\left(\|\varepsilon_{T_1^i}\|_2 \times \|\varepsilon_{T_2^i}\|_2\right) + \sum_{i=1}^{k}\left(\|\varepsilon_{T_1^i}\|_2 \times \|f_{T_2^i}\|_2\right) + \sum_{i=1}^{k}\left(\|\varepsilon_{T_2^i}\|_2 \times \|f_{T_1^i}\|_2\right)$	✗	✓	$\sum_{i=1}^{k_1}\left(\|\varepsilon_{T_1^i}\|_2 \times (\sum_{j\in\Pi_{T_2,[a_1^i,b_1^i]}}\|f_{T_2^j}\|_2^2)^{\frac{1}{2}}\right) + \sum_{i=1}^{k_2}\left(\|\varepsilon_{T_2^i}\|_2 \times (\sum_{j\in\Pi_{T_1,[a_2^i,b_2^i]}}\|f_{T_1^j}\|_2^2)^{\frac{1}{2}}\right) + \sum_{[a,b]\in OPT(L_{T_1},L_{T_2})}\left(\left(\sum_{i\in\Pi_{T_1,[a,b]}}\|\varepsilon_{T_1^i}\|_2^2\right)^{\frac{1}{2}} \times\left(\sum_{i\in\Pi_{T_2,[a,b]}}\|\varepsilon_{T_2^i}\|_2^2\right)^{\frac{1}{2}}\right)$	✗	✓
	VS\LSF	$\sum_{i=1}^{k}\left(\|\varepsilon_{T_1^i}\|_2 \times \|f_{T_2^i}\|_2\right)$	✓	✓			
	LSF				$\sum_{i=1}^{k_1}\left(\|\varepsilon_{T_1^i}\|_2 \times \|\boldsymbol{f_{T_2}}|_{[\boldsymbol{a_1^i},\boldsymbol{b_1^i}]} - \boldsymbol{f^*_{T_1}}^i\|_2\right) + \sum_{i=1}^{k_2}\left(\|\varepsilon_{T_2^i}\|_2 \times \|\boldsymbol{f_{T_1}}|_{[\boldsymbol{a_2^i},\boldsymbol{b_2^i}]} - \boldsymbol{f^*_{T_2}}^i\|_2\right) + \sum_{[a,b]\in OPT(L_{T_1},L_{T_2})}\left(\left(\sum_{i\in\Pi_{T_1,[a,b]}}\|\varepsilon_{T_1^i}\|_2^2\right)^{\frac{1}{2}} \times\left(\sum_{i\in\Pi_{T_2,[a,b]}}\|\varepsilon_{T_2^i}\|_2^2\right)^{\frac{1}{2}}\right)$	✓	✓
$Sum(\boldsymbol{T_1 + T_2})$	ANY	$\sum_{i=1}^{k}(\gamma_{T_1^i} + \gamma_{T_2^i})$	✓	✓	$\sum_{i=1}^{k_1}\gamma_{T_1^i} + \sum_{j=1}^{k_2}\gamma_{T_2^j}$	✓	✓
$Sum(\boldsymbol{T_1 - T_2})$							

Table 1: Error guarantees for the time series analytic (TSA) $Sum(T_1 \diamond T_2)$ **where** $\diamond \in \{\times, +, -\}$ **on both aligned and misaligned time series compressed by estimation functions in different families. We assume** T_1 **and** T_2 **have** k_1 **and** k_2 **segments respectively. In the aligned case, we have** $k_1 = k_2 = k$**.** $OPT(L_{T_1}, L_{T_2})$ **is the optimal segment combination returned by the algorithm OS in Section 4.2.1**

the (temperature, pressure) pairs satisfying a (not detailed in the example) condition over the dimension attributes". Notice, the first argument of the TSA UDF is a *time series analytic expression* (in red italics). We could write simply 'CCorr(t.timeseries, p.timeseries, 5)', as there is a built-in cross-correlation expression CCorr but, instead, the example writes the equivalent expression that uses more basic functions (such as the average μ, the standard deviation σ and the time Shifting) to exhibit the ability of Plato to process expressions that are compositions of well-known arithmetic operators, vector operators, aggregation and time shifting. Either way, computing the accurate cross-correlations would cost more than 10 minutes. However, Plato reduces the running time to within one second by computing the approximate correlations. It also delivers deterministic error guarantees. (In SQL, the result is a string concatenation of the approximate answer and the error guarantee. The functions approximateAnswer and guarantee extract the respective pieces.)

The success of approximate querying on IoT time series data is based on an important beneficial property of time series data: the points in the sequence of values normally *depend* on the previous points and exhibit *continuity*. For example, a temperature sensor is very unlikely to report a 100 degrees increase within a second. Therefore, in the signal processing and data mining communities [3, 12, 19, 22], time series data is usually modeled and compressed by continuous functions in order to reduce its size. For instance, the Piecewise Aggregate Approximation (PAA) [22] and the Piecewise Linear Representation (PLR) [19] adopt polynomial functions (0-degree in PAA and 1-degree in PLR) to compress the time series; [36] uses Gaussian functions; [45] applies natural logarithmic functions and natural exponential functions to compress time series. Plato is open to any existing time series compression techniques. Notice that there is no one-size-fits-all function family that can best model all kinds of time series data. For example, polynomials and ARMA models are better at modeling data from physical processes such as temperature [8, 33], while Gaussian functions are better for modeling relatively randomized data [25] such as stock prices. How to choose the best function family has been widely studied in prior work [10, 27, 39, 48] and recent efforts even attempt to automate the process [28]. We assume that the Plato users make a proper selection of how to model/compress the time series data and we do not further discuss this issue.

Architecture. Figure 2 shows the high-level architecture. During insertion time, the provided time series is compressed. In particular, a compression *function family* (e.g., 2nd-degree

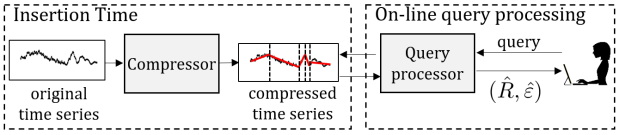


Figure 2: Plato's Approximate Querying

polynomials) is chosen by the user. Internally, in a simple version, each time series is segmented (partitioned) first in equal lengths. Then, for each segment the system finds the best *estimation function*, which is the member of the function family that best approximates the values in this segment. The most common definition of "best" is the minimization of the *reconstruction error*, i.e., the minimization of the Euclidean distance between the original and the estimated values. This is also the definition that Plato assumes. The compressed database stores the parameters of the estimation function for each segment, which take much less space than the original time series data. In the more sophisticated version, segmentation and estimation are mingled together [20, 26] to achieve better compression. The result is that the time series is partitioned into variable-length segments.

Consequently, given a query q with TSA UDF calls, [2] the database computes quickly an approximate answer for each TSA call by using the compressed data. Note, the TSAs may combine multiple time series; e.g., a correlation or a cross-correlation.

Example 1. *Consider a room temperature time series T_1 and an air pressure time series T_2 in Figure 1 and consider the TSA('Ccorr(T_1, T_2, 60)', T_1, T_2) where 'Ccorr(T_1, T_2, 60)' refers to the 60-seconds cross-correlation of T_1 and T_2 (see definition in Table 4). Both T_1 and T_2 have* 600 *data points at 1-second resolution and are segmented by variable length segmentation methods and compressed by PLR (1-degree polynomial functions). The precise answer is* 0.303. *But instead of accessing the* 1200 *(*600 × 2*) original data points, Plato produces the approximate answer* 0.300 *(error is* 0.003*) by accessing just the function parameters (*−0.072, 69.38*), (*−0.002, 65.77*) for T_1 and (*−0.046, 37.23*), (*−0.038, 38.04*) for T_2 in the compressed database.*[3]

The well-known downside of approximate querying is that errors are introduced. When the example's user receives the approximate answer 0.300 she cannot tell how far this answer is from the *true answer*, i.e., the precise answer. The novelty of Plato is the provision of *tight (i.e., lower bound) deterministic error guarantees* for the answers, even when the time series expressions combine multiple series. In the Example 1, Plato guarantees that the true answer is within ±0.0032 of the approximate answer 0.300 with 100% confidence. (Indeed, 0.303 is within ±0.0032 of 0.300.) It produces these guarantees by utilizing *error measures* associated with each segment.

Error measures	Comments
$\|\|\varepsilon_T\|\|_2 = \sqrt{\sum_{i=a}^{b}(T[i] - f_T^*(i))^2}$	L_2-norm of the estimation errors
$\|\|f_T\|\|_2 = \sqrt{\sum_{i=a}^{b}(f_T^*(i))^2}$	L_2-norm of the estimated values
$\gamma_T = \|\sum_{i=a}^{b} T[i] - \sum_{i=a}^{b} f_T^*(i)\|$	Absolute reconstruction error

Table 2: Error measures stored for a time series segment T running from a to b and approximated with the estimation function f_T^*.

Scope of Queries and Error Guarantees. Plato supports the time series analytic expressions formally defined in Table 3 (Section 2).They are composed of vector operators (+, −, ×, Shift), arithmetic operators, the aggregation operator Sum that turns its input vector into a scalar, and the Constant operator that turns its input scalar into a vector. As such, Plato queries can express not only statistics that involve one time series (eg, average, variance, and n-th moment) but also statistics that involve multiple time series, such as correlation and cross-correlation.

The error guarantee framework is also general. It allows efficient error guarantee computation for all possible estimation function families, as long as the error measures of Table 2 are computed in advance. [4] Figure 1 shows the error measures Φ (in blue) for each segment of the example. With the help of the error measures, no matter whether a time series is compressed by trigonometric functions or polynomial functions or some other family, Plato is able to give tight deterministic error guarantees for queries involving the compressed time series.

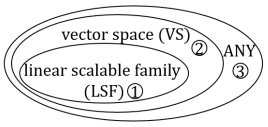
Function Family Groups Producing Practical Error Guarantees. Plato produces tight error guarantees, for *any* function family that may have been used in the compression. In addition, our theoretical and experimental analysis identifies which families lead to high quality guarantees.

The formulas of Table 1 provide error guarantees for characteristic, simple expressions and exhibit the difference in guarantee quality. Any other expression, e.g., the statistics of Table 4, are also given error guarantees by composing the error measures and guarantees of their subexpressions (as shown in the paper) and the same quality characterizations apply to them inductively.

[2] We focus on aggregation queries whose results are single scalar values, so the approximate answers are also scalar values.

[3] Due to reasons relating to computation efficiency, as explained in Section 4.2.2, Plato does not actually store the parameters (−0.072, 69.38), (−0.002, 65.77) and (−0.046, 37.23), (−0.038, 38.04) in their standard basis but rather it stores coefficients in an orthonormal basis.

[4] We will show that in certain cases one or two measures suffice.

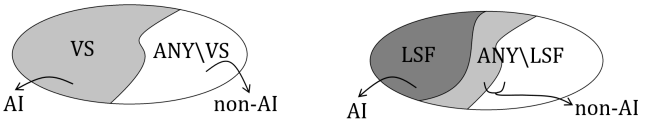


(a) Function family groups

Family Group	Example Function Family
LSF①	$\{ax + b \mid a, b \in R\}$
VS\LSF②	$\{a\ln(1+x) + b \mid a, b \in R\}$
ANY\VS③	$\{\frac{L}{1+\exp(ax+b)} + c \mid L, a, b, c \in R\}$ $\{a\exp(\frac{-(x-b)^2}{2c^2}) + d \mid a, b, c, d \in R\}$

(b) Example function families

Figure 3: Function family groups and examples.



(a) Function family dichotomy for error guarantees of queries on aligned time series

(b) Function family dichotomy for error guarantees of queries on misaligned time series

Figure 4: Function family groups and resulting guarantees

This is how to interpret the results of Table 1: Three function family groups have been identified: (1) The Linear Scalable Family group (LSF), (2) the Vector Space (VS), which includes the LSF and (3) ANY, which, according to its name, includes everything. Given the function family $\mathbb{F}$ used in the compression, we first categorize $\mathbb{F}$ in one of LSF or VS/LSF (i.e., VS excluding LSF) or ANY/VS. For example, if $\mathbb{F}$ is the 2-degree polynomials, then $\mathbb{F}$ belongs to LSF. See Figure 3 for other examples. Next, we consider whether the segments of the involved compressed time series are aligned or misaligned and finally we look at the error guarantee formula for the expression.

The specifics of interpreting the table's results and the specifics of their efficient computation require the detailed discussion of the paper. (Eg, the summation index *OPT* corresponds to the optimal segment combination (Section 4.2.1.) Nevertheless, a clear and general high level lesson about the practicality of the error guarantees emerges from the table's summary: *Some function families allow for much higher quality error guarantees than other function families.* The typical characteristic of "higher quality" is *Amplitude Independence (AI).* If an error guarantee is AI, then it is not influenced by the $\|f_T\|_2$ measure, i.e., it is not affected by the amplitude of the values of the estimation functions and, thus, it is not affected from the amplitude of the original data. An AI error guarantee is only affected by the reconstruction errors caused by the estimation functions, which intuitively implies that AI error guarantees are close to the actual error.

These guarantees are *tight* in the following sense. Given (a) the function family categorization into LSF, VS/LSF or ANY/VS and (b) segments with the error measures of Table 2, the formula provided by Table 1 produces an error guarantee that is as small as possible. That is, for this superfamily and for the given error measures, any attempt to create a better (i.e., smaller) error guarantee will fail because there are provably time series and at least one time series analytics expression where the true error is exactly as large as the error guarantee.

The experimental results, where we tried data sets with different characteristics and different compression methods, verified the above intuition: AI error guarantees were *order(s) of magnitude smaller* than their amplitude dependent counterparts. Indeed, AI ones over variable-length compressions were invariably small enough to be practically meaningful, while non-AI guarantees were too large to be practically useful.

Particularly interesting are the analytics that combine multiple vectors, such as correlation and cross-correlation, by vector multiplication. Then the amplitude independence of the error guarantees does not apply generally. Rather the dichotomy illustrated in Figure 4 emerges: (i) for compressions with aligned time series segments, the error guarantee is AI when the used function family forms a *Vector Space (VS)* in the conventional sense [16]; and (ii) for compressions with misaligned time series segments, which are the more common case, choosing a VS family is not enough for AI guarantees. The family must be a *Linear Scalable Family (LSF)*, which is a property that we define in this paper (Section 3.1).

The contributions are summarized as follows.

- We deliver tight deterministic error guarantees for a wide class of analytics over compressed time series. The key challenge is analytics (e.g., correlation and cross-correlation) that combine multiple time series but it is not known in advance which time series may be combined. Thus, each time series has been compressed individually, much before a query arrives. The reconstruction errors of the individual time series' compressions cannot provide, by themselves, decent guarantees for queries that multiply time series. To make the problem harder, time series segmentations are generally misaligned.[5]
- The provided guarantees apply regardless of the specifics of the segmentation and estimation function family used during the compression, thus making the provided deterministic error guarantees applicable to any prior work on segment-based compression (eg, variable-sized histograms etc). The only requirement is the common assumption that the estimation function minimizes the Euclidean distance between the actual values and the estimates.

[5]Misalignment happens because the most effective compressions use variable length segmentations. But even if the segmentations were fixed length, queries such as cross-corellation and cross-autocorellation time shift one of their time series, thus producing misalignment with the second time series.

Time Series Analytic (TSA)	
Q → Ar	
Arithmetic Expression (Ar)	
Ar → literal value in R	
| Ar ⊗ Ar	where $\otimes \in \{+, -, \times, \div, \sqrt{}\}$
| Agg	
Aggregation Expression (Agg)	
Agg → Sum$(\boldsymbol{T}, a', b')$	$\sum_{i=a'}^{b'} \boldsymbol{T}[i]$, where $[a', b'] \subseteq [a, b]$
Time Series Expression (TSE)	
$\boldsymbol{T}$ → input time series	
| Constant(v, a, b)	$(a, b, [\underbrace{v, v, ..., v}_{b-a+1}])$
| Shift$(\boldsymbol{T}, k)$	$(a + k, b + k, [\boldsymbol{T}[a], ..., \boldsymbol{T}[b]])$
| $\boldsymbol{T_1 + T_2}$	$(a, b, [\boldsymbol{T_1}[a] + \boldsymbol{T_2}[a], ..., \boldsymbol{T_1}[b] + \boldsymbol{T_2}[b]])$
| $\boldsymbol{T_1 - T_2}$	$(a, b, [\boldsymbol{T_1}[a] - \boldsymbol{T_2}[a], ..., \boldsymbol{T_1}[b] - \boldsymbol{T_2}[b]])$
| $\boldsymbol{T_1 \times T_2}$	$(a, b, [\boldsymbol{T_1}[a] \times \boldsymbol{T_2}[a], ..., \boldsymbol{T_1}[b] \times \boldsymbol{T_2}[b]])$

Table 3: Grammar of time series analytic (TSA). Let $T_1 = (a_1, b_1, [T_1[a_1], ..., T_1[b_1]])$ and $T_2 = (a_2, b_2, [T_2[a_2], ..., T_2[b_2]])$ be the input time series in the time series expressions, $a = \max(a_1, a_2)$ and $b = \min(b_1, b_2)$.

- We identify broad estimation function family groups (namely, the already defined Vector Space family and the presently defined Linear Scalable Family) that lead to theoretically and practically high quality guarantees. The theoretical aspect of high quality is crisply captured by the Amplitude Independence (AI) property. Furthermore, the error guarantees are computed very efficiently, in time proportional to the number of segments.
- The results broadly apply to analytics involving composition of the typical operators, which is powerful enough to express common statistics, such as variance, correlation, cross-correlation and other in any time range.
- We conduct an extensive empirical evaluation on four real-life datasets to evaluate the error guarantees provided by Plato and the importance of the VS and LSF properties on error estimation. The results show that the AI error guarantees are very narrow - thus, practical. Furthermore, we compare to sampling-based approximation and show experimentally that Plato delivers deterministic (100% confidence) error guarantees using fewer data than it takes to produce probabilistic error guarantees with 95% and 99% confidence via sampling.

2 TIME SERIES AND EXPRESSIONS

Time Series A time series $\boldsymbol{T} = (a, b, [\boldsymbol{T}[a], \boldsymbol{T}[a+1], ..., \boldsymbol{T}[b]])$, $a \in N$, $b \in N$, is a sequence of data points $[\boldsymbol{T}[a], \boldsymbol{T}[a+1], ..., \boldsymbol{T}[b]]$ observed from start time a to end time b ($a, b \in N$). Following the assumptions in [6, 34, 46] we assume that time is discrete and the resolution of any two time series is the same. Equivalently, we say $\boldsymbol{T}$ is fully defined in the integer time domain $[a, b]$. We assume a domain $[1, n]$ is the global domain meaning that all the time series are defined within subsets of this domain. When the domain of a time series $\boldsymbol{T}$ is implied by the context, then $\boldsymbol{T}$ can be simplified as $\boldsymbol{T} = [\boldsymbol{T}[a], \boldsymbol{T}[a+1], ..., \boldsymbol{T}[b]]$.

Example 2. *Assume the global domain is* $[1, 100]$. *Consider two time series* $\boldsymbol{T_1} = (1, 5, [61.52, 59.54, 58.64, 59.36, 60.44])$ *and* $\boldsymbol{T_2} = (3, 6, [1.02, 1.03, 1.02, 1.02])$. *Then* $\boldsymbol{T_1}$ *and* $\boldsymbol{T_2}$ *are fully defined in domains [1,5] and [3,6] respectively.* $\boldsymbol{T_2}[4] = 1.03$ *refers to the* 2^{nd} *data point of* $\boldsymbol{T_2}$ *at the* 4*-th position in the global domain.*

Time Series Analytic (TSA) Expressions Table 3 shows the formal definition of the *time series analytic* (called *TSA*). The TSAs supported are expressions composed of linear algebra operators and arithmetic operators. Typically, the TSA has subexpressions that compose one or more linear algebra operators over multiple time series vectors as defined below.

- Given a numeric value v and two integers a and b, Constant$(v, a, b) = (a, b, [v, ..., v])$. For example, Constant$(1.6, 3, 5)$ produces $(3, 5, [1.6, 1.6, 1.6])$.
- Given a time series $\boldsymbol{T} = (a, b, [\boldsymbol{T}[a], ..., \boldsymbol{T}[b]])$ and an integer value k, Shift$(\boldsymbol{T}, k)$=$(a+k, b+k, [\boldsymbol{T}[a], ..., \boldsymbol{T}[b]])$. Notice Shift$(\boldsymbol{T}, k)[i + k] = \boldsymbol{T}[i]$ for all $a \le i \le b$. Figure 5(a) visualizes the Shift operator. Consider the time series $\boldsymbol{T} = (1, 3, [1.8, 1.6, 1.6])$, then Shift$(\boldsymbol{T}, 6)$ is $(7, 9, [1.8, 1.6, 1.6])$.
- Given two time series $\boldsymbol{T_1} = (a_1, b_1, [\boldsymbol{T_1}[a_1], ..., \boldsymbol{T_1}[b_1]])$ and $\boldsymbol{T_2} = (a_2, b_2, [\boldsymbol{T_2}[a_2], ..., \boldsymbol{T_2}[b_2]])$, $\boldsymbol{T_1 \times T_2} = (a, b, [\boldsymbol{T_1}[a] \times \boldsymbol{T_2}[a], ..., \boldsymbol{T_1}[b] \times \boldsymbol{T_2}[b]])$ where $a = max(a_1, a_2)$ and $b = min(b_1, b_2)$. [6] For example, given $\boldsymbol{T_1} = (1, 2, [3.3, 3.5])$ and $\boldsymbol{T_2} = (1, 2, [1.0, 1.2])$ then $\boldsymbol{T_1 \times T_2} = (1, 2, [3.3, 4.2])$. Similarly, we define $\boldsymbol{T_1 + T_2}$ and $\boldsymbol{T_1 - T_2}$.

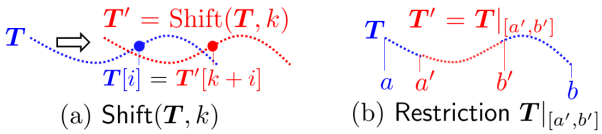
A time series analytic (TSA) is an arithmetic expression of the form $Arr_1 \otimes Arr_2 \otimes \ldots Arr_n$, where $\otimes$ are the standard arithmetic operators $(+, -, \times, \div, \sqrt{})$ and Arr_i is either an arithmetic literal or an aggregation over a time series expression. An aggregation expression Sum$(\boldsymbol{T}, a', b')$ computes the summation of the data points of T in the domain $[a', b']$, i.e., Sum$(\boldsymbol{T}, a', b')$=$\sum_{i=a'}^{b'} \boldsymbol{T}[i]$ where $\boldsymbol{T}$ can be an input time series or a derived time series computed by time series expressions (TSEs). [7] When the bounds of a' and b'

[6] Setting $a = max(a_1, a_2)$ and $b = min(b_1, b_2)$ ensures all the data points in $\boldsymbol{T_1 \times T_2}$ are defined.

[7] Note that, when the time series expressions involve time shifting, we assume that the aggregation will only operator in the valid data points, that is the data points in the defined range.

TSA Expression	Definition	Equivalent TSA Expression	Usage of error measures
Average μ_{T_1} '$\mu(T_1)$'	$\frac{1}{b_1-a_1+1}(\sum_{i=a_1}^{b_1} T_1[i])$	$\frac{1}{b_1-a_1+1}(\text{Sum}(\boldsymbol{T_1}))$	γ_{T_1}
Standard Deviation σ_{T_1} '$\sigma(T_1)$'	$\sqrt{\frac{1}{b_1-a_1+1}\left(\sum_{i=a_1}^{b_1}(T_1[i]-\mu_{T_1})^2\right)}$	$\sqrt{\frac{1}{b_1-a_1+1}\times Sum(\boldsymbol{T_1}-\text{Constant}(\mu_{T_1}))}$	γ_{T_1}
Correlation $r_{(T_1,T_2)}$ 'Corr(T_1,T_2)'	$\frac{\sum_{i=max(a_1,a_2)}^{min(b_1,b_2)}\left((T_1[i]-\mu_{T_1})(T_2[i]-\mu_{T_2})\right)}{\sigma_{T_1}\times\sigma_{T_2}}$	$\frac{Sum\left((\boldsymbol{T_1}-\text{Constant}(\mu_{T_1}))\times(\boldsymbol{T_2}-\text{Constant}(\mu_{T_2}))\right)}{\sigma_{T_1}\times\sigma_{T_2}}$	$\|\varepsilon_{T_1}\|_2,\|f_{T_1}\|_2,\gamma_{T_1}$, $\|\varepsilon_{T_2}\|_2,\|f_{T_2}\|_2,\gamma_{T_2}$
Cross-correlation $r_{(T_1,T_2,m)}$ 'CCorr(T_1,T_2,m)'	$\frac{\sum_{i=max(a_1,a_2+m)}^{min(b_1,b_2+m)}\left((T_1[i]-\mu_{T_1})(T_2[i+m]-\mu_{T_2})\right)}{\sigma_{T_1}\times\sigma_{T_2}}$	$\frac{Sum\left((\boldsymbol{T_1}-\text{Constant}(\mu_{T_1}))\times(Shift(\boldsymbol{T_2},m)-\text{Constant}(\mu_{T_2}))\right)}{\sigma_{T_1}\times\sigma_{T_2}}$	$\|\varepsilon_{T_1}\|_2,\|f_{T_1}\|_2,\gamma_{T_1}$, $\|\varepsilon_{T_2}\|_2,\|f_{T_2}\|_2,\gamma_{T_2}$
Auto-correlation $r_{(T_1,m)}$ 'ACorr(T_1,m)'	$\frac{\sum_{i=a_1+m}^{b_1}\left((T_1[i]-\mu_{T_1})(T_1[i+m]-\mu_{T_1})\right)}{\sigma_{T_1}^2}$	$\frac{Sum\left((\boldsymbol{T_1}-\text{Constant}(\mu_{T_1}))\times(Shift(\boldsymbol{T_1},m)-\text{Constant}(\mu_{T_1}))\right)}{\sigma_{T_1}\times\sigma_{T_1}}$	$\|\varepsilon_{T_1}\|_2,\|f_{T_1}\|_2,\gamma_{T_1}$

Table 4: Example TSA's for common statistics. Let $T_1 = (a_1, b_1, [\cdots])$ and $T_2 = (a_2, b_2, [\cdots])$ be the input time series in the time series analytic.



(a) Shift($\boldsymbol{T}, k$) (b) Restriction $\boldsymbol{T}|_{[a',b']}$

Figure 5: Time series *Shift* and *Restriction* operators.

are implied from the context, we simplify $Sum(\boldsymbol{T}, a', b')$ to $Sum(\boldsymbol{T})$.

3 INTERNAL, COMPRESSED TIME SERIES REPRESENTATION

When a user inserts a time series into the database, Plato physically stores the *compressed time series representation* instead of the raw time series. More precisely, the user provides (i) a time series T, (ii) the identifier of a segmentation algorithm, which is chosen from a list provided by Plato, and (iii) the identifier of a function family, which is selected from a list provided by Plato. Internally, Plato uses the chosen segmentation algorithm and the chosen compression function family to partition T into a list of disjoint segments $T^1, ..., T^n$. For each segment $T^i = (a, b, [T^i[a], ..., T^i[b]])$, instead of storing its original data points $[T^i[a], ..., T^i[b]]$, Plato stores a *compressed segment representation* $\tilde{T^i} = (a, b, \tilde{f_T^*}, \Phi(T))$, where a is the start position, b is the end position, $\tilde{f_T^*}$ is the function representation of f_T^*, where f_T^* is the estimation function chosen from the identified function family and $\Phi(T)$ is a set of (two to three depending on the function family) *error measures*.

Overall, for a time series T, Plato physically stores (i) the list L_T=($\tilde{T^1}$, ..., $\tilde{T^n}$), and (ii) one token (which can simply be an integer) as the function family identifier. [8]

We comment on the prior state-of-the-art segmentation / compression algorithms that Plato uses in Appendix A. Next, we introduce the selection of the estimation function and the computation of error measures.

3.1 Estimation Function Selection

Choosing an estimation function for a time series segment has two steps: (i) user identifies the function family, and (ii) Plato selects the best function in the family, i.e., the function that minimizes the Euclidean distance between the original values and the estimated values produced by the function.

Step 1: Function family selection. Table 7 gives example function family identifiers, which the user may select, and the corresponding function expressions. For example, τ="p_2" means that the chosen function family is the "second-degree polynomial function family" and the corresponding function family expression is $\{ax^2 + bx + c | a, b, c \in R\}$.

Step 2: Estimation function selection. Any function f in the chosen function family $\mathbb{F}$ is a *candidate estimation function*. Following the prior work [2, 30], Plato selects the candidate estimation function that minimizes the Euclidean distance between the original values and the estimated values produced by the function to be the final estimation function.

[8] It is not necessary for Plato to physically store a token for the segmentation algorithm identifier as the time series stored in Plato has been partitioned already.

More precisely,

$$f_T^* = \arg\min_{f \in \mathbb{F}} \Big(\sum_{i=a}^{b} (\boldsymbol{T}[i] - f(i))^2 \Big)^{1/2} \quad (1)$$

Example 3. *Given a time series* $\boldsymbol{T} = (1, 5, [0.2, 0.4, 0.4, 0.5, 0.6])$*, assume the function family identifier is "*p_1*" (i.e., "first-degree polynomial function family"). Functions* $f_1 = 0.05 \times i + 0.3$ *and* $f_2 = 0.09 \times i + 0.15$ *are two candidate estimation functions. Finally, Plato selects* $f_2 = 0.09 \times i + 0.15$ *as the estimation function since it produces the minimal Euclidean error, i.e.,* 0.0837*.*

Function Representation (Physical) vs. Function (Logical). Once an estimation function f_T^* is selected, Plato stores the corresponding *function representation* $\tilde{f}_T^*$, which includes (i) the coefficients of the function f_T^*, and (ii) the function family identifier τ. [9] For example, the function representation of the estimation function in Example 3 is $\tilde{f}_T^* = ((0.09, 0.15), p_1)$ where p_1 is a function family identifier indicating that the function family is "1-degree polynomial function family".

When we talk about the function itself logically, it can be regarded as a vector that maps time series: given a domain $[a, b]$, the vector $[f(a), f(a+1), \ldots, f(b)]$ maps a value to each position in the domain $[a, b]$. For example, consider the estimation function $f_T^* = 0.09 \times i + 0.15$ in Example 3. Then $\boldsymbol{T} - f_T^* = [0.2 - f_T^*(1), 0.4 - f_T^*(2), 0.4 - f_T^*(3), 0.5 - f_T^*(4), 0.6 - f_T^*(5)] = [0.2 - 0.24, 0.4 - 0.33, 0.4 - 0.42, 0.5 - 0.51, 0.6 - 0.6] = [0.04, 0.07, -0.02, -0.01, 0]$.

3.2 Error Measures

In addition to the estimation function, Plato stores extra *error measures* $\Phi(T) = \{\|\varepsilon_T\|_2, \|f_T\|_2, \gamma_T\}$ for each time series segment T (defined in domain $[a, b]$) where $\|\varepsilon_T\|_2$, $\|f_T\|_2$, and γ_T are defined in Table 2.

Example 4. *Consider the time series* $\boldsymbol{T} = (1, 5, [0.2, 0.4, 0.4, 0.5, 0.6])$ *in Example 3 again.* $f_T^* = 0.09 \times i + 0.15$ *is the estimation function. Thus* $\|\varepsilon_T\|_2 = \sqrt{\sum_{i=1}^{5} (T[i] - f_T^*(i))^2} = 0.0837$*,* $\|f_T\|_2 = \sqrt{\sum_{i=1}^{5} (f_T^*(i))^2} = 0.9813$*, and* $\gamma_T = |\sum_{i=1}^{5} T[i] - \sum_{i=1}^{5} f_T^*(i)| = 2.1 - 2.1 = 0$*.*

Elimination of γ_T. We will see in Lemma 1 (Section 4.1.1) that if the selected function family forms a vector space, then γ_T is guaranteed to be 0. Then we can avoid storing it.

4 ERROR GUARANTEE COMPUTATION

Error Guarantee Definition. Given a TSA q involving time series $T_1, .., T_n$, let R be the accurate answer of q by executing q directly on the original data points of $T_1, .., T_n$. Let $\hat{R}$ be the approximate answer of q by executing q on the compressed time series representations. Then $\varepsilon = |\hat{R} - R|$ is the *true error* of q. Notice that ε is unknown since R is unknown. An upper bound $\hat{\varepsilon}$ ($\hat{\varepsilon} \geq \varepsilon$) of the true error is called a *deterministic error guarantee* of q. With the help of $\hat{\varepsilon}$, we know that the accurate answer R is within the range $[\hat{R} - \hat{\varepsilon}, \hat{R} + \hat{\varepsilon}]$ with 100% confidence. Plato provides *tight* deterministic error guarantees for time series expressions defined in Table 3 (Section 2).

Error Guarantee Decomposition. Recall that the time series analytic q defined in Table 3 (Section 2) combines one or more time series aggregation operations via arithmetic operators, i.e., $q = Agg_1 \otimes Agg_2 \otimes \cdots \otimes Agg_n$ where $\otimes \in \{+, -, \times, \div, \sqrt{}\}$. In order to provide the deterministic error guarantee $\hat{\varepsilon}$ of the time series analytic q, the key step is to calculate the deterministic error guarantee $\hat{\varepsilon}_{Agg_i}$ of each aggregation operation Agg_i. Once we have $\hat{\varepsilon}_{Agg_i}$ for each aggregate expression, it is not hard to combine them to get the final error guarantee (see Appendix B).

Given a TSA $Agg = \mathsf{Sum}(\boldsymbol{T})$ and the compressed time series representation $L_T = \{\tilde{T^1}, ..., \tilde{T^k}\}$. When calculating $\hat{\varepsilon}_{Agg}$, there are two cases depending on whether $\boldsymbol{T}$ is an input time series or not. [10]

- Case 1. $\boldsymbol{T}$ is an input time series, then $\hat{\varepsilon}_{Agg} = \sum_{i=1}^{k} \gamma_{T^i}$ where γ_{T^i} is the reconstruction error in the error measures of T^i. [11]
- Case 2. $\boldsymbol{T}$ is a derived time series by applying the time series operators (recursively), $\mathsf{Constant}(v, a, b)$, $\mathsf{Shift}(\boldsymbol{T}, k)$, $\boldsymbol{T_1 + T_2}$, $\boldsymbol{T_1 - T_2}$ and $\boldsymbol{T_1 \times T_2}$. In this case, the aggregation operator $Agg = Sum(\boldsymbol{T})$ can be depicted as a tree. Figure 6 shows an example tree of the aggregation operator in the "correlation TSA". In order to compute $\hat{\varepsilon}_{Agg}$, we first calculate the error measures $\Phi(\boldsymbol{T}) = (\|\varepsilon_T\|_2, \|f_T\|_2, \gamma_T)$ for the root time series in the tree by propagating the error measures from the bottom time series to the root. Then we return the γ_T in the $\Phi(\boldsymbol{T})$ as the final error guarantee.

Next, we focus on computing the error measures for derived time series. We first explain the simpler case where each time series is a single segment. Table 8 shows the formulas for computing error measures for derived time series in this case. For the general scenario where multiple segments are involved in each input time series in the expression, there are two cases depending on whether the segments are aligned or not: If the i-th segment in T_1 has the same domain with

[9] All the segments in the same time series share one token τ.

[10] If a time series is generated by applying some time series operators, then it is not a base time series. For example, $\boldsymbol{T} = \boldsymbol{T_1} \times \boldsymbol{T_2}$, then $\boldsymbol{T}$ is not a base time series.

[11] Here we assume the aggregation operator aggregates the whole time series.

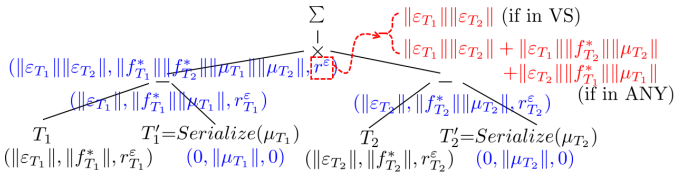
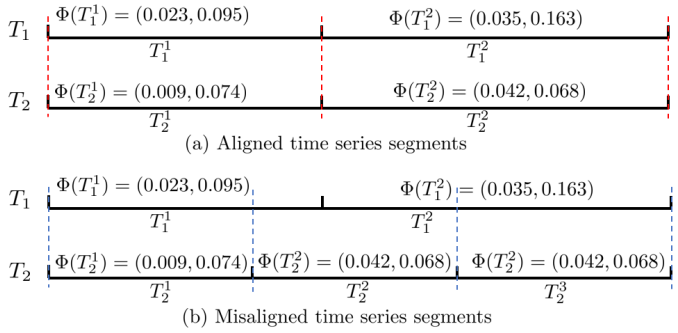


Figure 6: Example of error measures propagation. Error measures in black color are precomputed offline during insertion time, while error measures in blue color are computed during the TSA processing time. The final error guarantees are in red color.



(a) Aligned time series segments

(b) Misaligned time series segments

Figure 7: Example of aligned segments and misaligned segments.

the i-th segment in T_2 for all i, then T_1 and T_2 are *aligned*, otherwise, they are *misaligned*.

In the following, we will show how to compute the most challenging error guarantee $\hat{\varepsilon}_{Sum(T_1 \times T_2)}$ in both aligned and misaligned cases in Section 4.1 and Section 4.2 respectively. The computation of error guarantees of other expressions (i.e., Constant(v, a, b), Shift($\boldsymbol{T}, k$), $\boldsymbol{T_1 + T_2}$ and $\boldsymbol{T_1 - T_2}$) is presented in Appendix C.

4.1 Error Guarantee on Aligned Segments

Notations. Given a time series $\boldsymbol{T} = (\boldsymbol{T}[a], ..., \boldsymbol{T}[b])$ and the estimation function f^*_T of T, $\boldsymbol{\varepsilon}_T = \boldsymbol{T} - f^*_T = (\boldsymbol{T}[a] - f^*_T(a), ..., \boldsymbol{T}[b] - f^*_T(b))$ is the *vector of errors* produced by the estimation function. In the following, $\boldsymbol{T}$, f^*_T and $\boldsymbol{\varepsilon}$ are all regarded as vectors. $\langle f_1, f_2 \rangle = \sum_{i=a}^{b} f_1(i) f_2(i)$ is the inner product of f_1 and f_2. $V|_{[a,b]}$ is a *restriction* operation, which restricts a vector V to the domain [a,b]. Recall a time series segment is a subsequence of a time series. Thus, a segment is the *restriction* of a time series $\boldsymbol{T}$ from a bigger domain $[a, b]$ into a smaller domain $[a', b'] \subseteq [a, b]$, denoted as $\boldsymbol{T}|_{[a',b']}$. Figure 5(b) visualizes the restriction operator. For example, consider a time series $\boldsymbol{T} = (1, 4, [1.2, 1.3, 1.3, 1.2])$, then $\boldsymbol{T}|_{[2,3]} = (2, 3, [1.3, 1.3])$ is a restriction of $\boldsymbol{T}$. Note that $\boldsymbol{T}|_{[a',b']}[i] = T[i]$ for all $i \in [a', b']$.

Given two compressed time series representation $L_{T_1} = (\tilde{T}_1^1, ..., \tilde{T}_1^k)$ and $L_{T_1} = (\tilde{T}_2^1, ..., \tilde{T}_2^k)$ for the *aligned* time series $T_1 = (T_1^1, ..., T_1^k)$ and $T_2 = (T_2^i, ..., T_2^k)$ where $T_1^i = T_1|_{[a_i,b_i]}$ and $T_2^i = T_2|_{[a_i,b_i]}$. Notice T_1^i and T_2^i have the same domain, i.e., $[a_i, b_i]$, for all $i \in [1, k]$. For any estimation function family, the error guarantee of Sum($\boldsymbol{T_1 \times T_2}$) on aligned time series is:

$$\varepsilon \leq \sum_{i=1}^{k} \left(\|\varepsilon_{T_1^i}\|_2 \|\varepsilon_{T_2^i}\|_2 + \|\varepsilon_{T_1^i}\|_2 \|f_{T_2^i}\|_2 + \|f_{T_1^i}\|_2 \|\varepsilon_{T_2^i}\|_2 \right) \quad (2)$$

The details are shown in Appendix D.

Example 5. *Consider the two aligned time series in Figure 7(a). Both T_1 and T_2 are partitioned into two segments in this case, i.e., (T_1^1, T_1^2) and (T_2^1, T_2^2). Plato stores the error measures $\Phi(T_i^j)$ for each segment T_i^j. For instance, $\Phi(T_1^1) = (\|\varepsilon_{T_1^1}\|_2, \|f_{T_1^1}\|_2, \gamma_{T_1^1}) = (0.023, 0.95, 0)$. Then the error guarantee of Sum($\boldsymbol{T_1 \times T_2}$) on T_1 and T_2 is computed as* $(\|\varepsilon_{T_1^1}\|_2\|\varepsilon_{T_2^1}\|_2 + \|\varepsilon_{T_1^1}\|_2\|f_{T_2^1}\|_2 + \|f_{T_1^1}\|_2\|\varepsilon_{T_2^1}\|_2) + (\|\varepsilon_{T_1^2}\|_2\|\varepsilon_{T_2^2}\|_2 + \|\varepsilon_{T_1^2}\|_2\|f_{T_2^2}\|_2 + \|f_{T_1^2}\|_2\|\varepsilon_{T_2^2}\|_2) = (0.023 \times 0.009 + 0.023 \times 0.074 + 0.095 \times 0.009) + (0.035 \times 0.042 + 0.035 \times 0.068 + 0.163 \times 0.042) = 0.01346$.

4.1.1 *Orthogonal projection optimization.*

If the estimation function family forms a vector space (VS), [12] then we can apply the *orthogonal projection property* in VS to significantly reduce the error guarantee of $sum(T_1 \times T_2)$ from Formula 2 to Formula 3.

$$\varepsilon = \left| \sum_{i=1}^{k} \left(\underbrace{\langle \boldsymbol{\varepsilon}_{T_1^i}, f^*_{T_2^i} \rangle}_{=0 \text{ in } VS} + \underbrace{\langle \boldsymbol{\varepsilon}_{T_2^i}, f^*_{T_1^i} \rangle}_{=0 \text{ in } VS} + \langle \boldsymbol{\varepsilon}_{T_1^i}, \boldsymbol{\varepsilon}_{T_2^i} \rangle \right) \right|$$

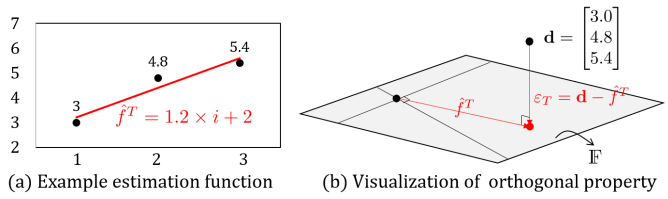
$$\leq \sum_{i=1}^{k} \left(\|\varepsilon_{T_1^i}\|_2 \|\varepsilon_{T_2^i}\|_2 \right) \quad (3)$$

Example 6. *Consider the two aligned time series in Figure 7(a) again. The estimation function family is polynomial function family, it is VS. Based on Formula 3, the error guarantee for Sum($\boldsymbol{T_1 \times T_2}$) is* $\|\varepsilon_{T_1^1}\|_2 \times \|\varepsilon_{T_2^1}\|_2 + \|\varepsilon_{T_1^2}\|_2 \times \|\varepsilon_{T_2^2}\|_2 = 0.023 \times 0.009 + 0.035 \times 0.042 = 0.001677$. *This error guarantee is about 8× smaller than that in Example 5 (i.e., 0.01346), where we did not take into account that the function family is VS.*

Orthogonal projection property. Example 6 indicates the power of the orthogonal projection optimization. Lemma 1 is a proof of Formula 3.

Lemma 1. *(Orthogonal Projection Property) Let $\mathbb{F}$ be a function family forms a vector space VS and $f^*_T \in \mathbb{F}$ be the estimation function of time series T. Then f^*_T is the orthogonal projection of* $\mathbf{T}$ *onto $\mathbb{F}$ [35].*

[12] A vector space is a set that is closed under finite vector addition and scalar multiplication. http://mathworld.wolfram.com/VectorSpace.html.



(a) Example estimation function (b) Visualization of orthogonal property

Figure 8: (a) shows the estimation function for three data points. (b) visualizes the orthogonal projection of the three data points onto the 2-dimensional plane $\mathbb{F}$.

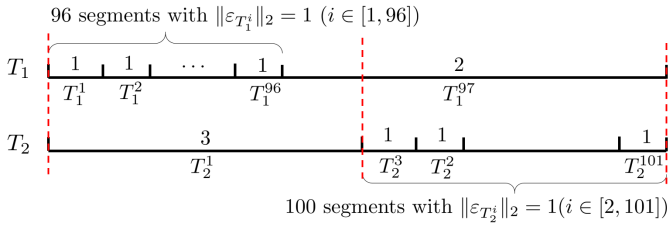


Figure 9: Example of segment combination selection.

Lemma 1 implies that $\boldsymbol{\varepsilon}_T$ is orthogonal to any function $f_T \in \mathbb{F}$, which means $\langle \boldsymbol{\varepsilon}_T, \boldsymbol{f}_T \rangle = 0$. Therefore, given any two aligned segments T_1^i and T_2^i, as both $f^*_{T_1^i}$ and $f^*_{T_2^i}$ are in VS, thus $\langle \boldsymbol{\varepsilon}_{T_1^i}, f^*_{T_2^i} \rangle = 0$ and $\langle \boldsymbol{\varepsilon}_{T_2^i}, f^*_{T_1^i} \rangle = 0$.

For visualization purposes, consider a time series with three data points $T = (1, 3, [3.0, 4.8, 5.4])$ and let $\mathbb{F}$ be the 1-degree polynomial function family (i.e., 2-dimensional). The estimation function that minimizes the error to the original data is $f^*_T = 1.2{\times}i{+}2$ (Figure 8(a)). As shown in Figure 8(b), f^*_T is the orthogonal projection of $\boldsymbol{T}$ onto $\mathbb{F}$. The error vector is $\boldsymbol{\varepsilon}_T = (-0.2, 0.4, -0.2)$. Based on Lemma 1, for any candidate estimation function $f = \alpha \times i + \beta$ $(\alpha, \beta \in R)$, we have $\langle \boldsymbol{\varepsilon}_T, \boldsymbol{f} \rangle = 0.8\alpha - 0.8\alpha + 0.4\beta - 0.4\beta = 0$.

Elimination of γ_T. We can get an extra benefit from the orthogonal projection property in saving space, i.e., the error measure γ_T can be avoided as it is guaranteed to be 0. This is because $\gamma_T = \langle T - f^*_T, 1 \rangle$ and 1 is a constant function in the function family in VS. According to Lemma 1, we know $\langle T - f^*_T, 1 \rangle = 0$. Therefore, we have $\gamma_T = 0$.

Amplitude-independent (AI). The orthogonal projection optimization can significantly reduce the error guarantees. It allows the error guarantees to get rid of the *amplitudes* of the original time series values (referring to $\|f_T\|_2$) by only consider the reconstruction error (referring to $\|\varepsilon_T\|_2$) of each time series. The error guarantees provided by Plato in VS are called *amplitude-independent (AI)* error guarantees.

4.2 Error Guarantee on Misaligned Segments

Given two compressed time series representation $L_{T_1} = (\tilde{T}_1^1, ..., \tilde{T}_1^{k_1})$ and $L_{T_1} = (\tilde{T}_2^1, ..., \tilde{T}_2^{k_2})$ for the misaligned time series $T_1 = (T_1^1, ..., T_1^{k_1})$ and $T_2 = (T_2^1, ..., T_2^{k_2})$ where the domains of T_1^i and T_2^i are $[a_i^1, b_i^1]$ and $[a_i^2, b_i^2]$ respectively. The major challenge in the misaligned case is that for a domain $[a_1^i, b_1^i]$, the error measures of the segment $T_1|_{[a_1^i, b_1^i]}$ are precomputed, however, the error measures of the segment $T_2|_{[a_1^i, b_1^i]}$ may be unknown as $T_2|_{[a_1^i, b_1^i]}$ in general is not one of the segments $T_2^1, ..., T_2^{k_2}$.

Let $\Pi_{T,[a,b]}$ be the set of segments in T covering the domain $[a, b]$. For example, consider the two misaligned time series T_1 and T_2 in Figure 7(b), $\Pi_{T_2,[a_1^1, b_1^1]} = \{T_2^1, T_2^2\}$ as the segments T_2^1 and T_2^2 in T_2 cover the domain $[a_1^1, b_1^1]$. [13] If any kinds of function families are allowed, i.e., in ANY, the error guarantee $\hat{\varepsilon}$ of $\mathrm{Sum}(\boldsymbol{T_1} \times \boldsymbol{T_2})$ on misaligned time series is:

$$
\begin{aligned}
\hat{\varepsilon} &= \left| \sum_{i=a}^{b} \boldsymbol{T}_1[i]\boldsymbol{T}_2[i] - \sum_{i=a}^{b} f^*_{T_1}(i) f^*_{T_2}(i) \right| \\
&\le \left| \langle \boldsymbol{\varepsilon}_{\boldsymbol{T_1}}, f^*_{T_2} \rangle \right| + \left| \langle \boldsymbol{\varepsilon}_{\boldsymbol{T_2}}, f^*_{T_1} \rangle \right| + \left| \langle \boldsymbol{\varepsilon}_{\boldsymbol{T_1}}, \boldsymbol{\varepsilon}_{\boldsymbol{T_2}} \rangle \right| \\
&= \left| \sum_{i=1}^{k_1} \langle \boldsymbol{\varepsilon}_{\boldsymbol{T_1^i}}, f^*_{T_2}|_{[a_1^i, b_1^i]} \rangle \right| + \left| \sum_{i=1}^{k_2} \langle \boldsymbol{\varepsilon}_{\boldsymbol{T_2^i}}, f^*_{T_1}|_{[a_2^i, b_2^i]} \rangle \right| + \left| \langle \boldsymbol{\varepsilon}_{\boldsymbol{T_1}}, \boldsymbol{\varepsilon}_{\boldsymbol{T_2}} \rangle \right| \\
&\le \sum_{i=1}^{k_1} \left(\|\varepsilon_{T_1^i}\|_2 \Big(\sum_{j \in \Pi_{T_2,[a_1^i, b_1^i]}} \|f_{T_2^j}\|_2^2 \Big)^{\frac{1}{2}} \right) + \sum_{i=1}^{k_2} \left(\|\varepsilon_{T_2^i}\|_2 \Big(\sum_{j \in \Pi_{T_1,[a_2^i, b_2^i]}} \|f_{T_1^j}\|_2^2 \Big)^{\frac{1}{2}} \right) \quad (1) \\
&\quad + \left| \langle \boldsymbol{\varepsilon}_{\boldsymbol{T_1}}, \boldsymbol{\varepsilon}_{\boldsymbol{T_2}} \rangle \right| \quad (2) \qquad (4)
\end{aligned}
$$

Formula 4 is a stepping stone towards producing the final formula as the computation of $|\langle \boldsymbol{\varepsilon}_{\boldsymbol{T_1}}, \boldsymbol{\varepsilon}_{\boldsymbol{T_2}} \rangle|$ (Formula 4②) has not been given yet. It will be discussed in Section 4.2.1. Section 4.2.2 discusses how to apply the orthogonal property optimization to improve Formula 4①.

4.2.1 Segment combination selection. To compute $|\langle \boldsymbol{\varepsilon}_{\boldsymbol{T_1}}, \boldsymbol{\varepsilon}_{\boldsymbol{T_2}} \rangle|$, one straightforward method (called IS) is to use the domains of segments in T_1 and T_2 independently, then choose the one with minimal value. Let's first see how to compute $|\langle \boldsymbol{\varepsilon}_{\boldsymbol{T_1}}, \boldsymbol{\varepsilon}_{\boldsymbol{T_2}} \rangle|$ with the domains of segments in T_1.

$$
\begin{aligned}
|\langle \boldsymbol{\varepsilon}_{\boldsymbol{T_1}}, \boldsymbol{\varepsilon}_{\boldsymbol{T_2}} \rangle| &\le \sum_{i=1}^{k_1} \left| \langle \boldsymbol{\varepsilon}_{\boldsymbol{T_1}}|_{[\boldsymbol{a_1^i}, \boldsymbol{b_1^i}]}, \boldsymbol{\varepsilon}_{\boldsymbol{T_2}}|_{[\boldsymbol{a_1^i}, \boldsymbol{b_1^i}]} \rangle \right| = \sum_{i=1}^{k_1} \left| \langle \boldsymbol{\varepsilon}_{\boldsymbol{T_1^i}}, \boldsymbol{\varepsilon}_{\boldsymbol{T_2}}|_{[\boldsymbol{a_1^i}, \boldsymbol{b_1^i}]} \rangle \right| \\
&\le \sum_{i=1}^{k_1} \left(\|\varepsilon_{T_1^i}\|_2 \Big(\sum_{j \in \Pi_{T_2,[a_1^i, b_1^i]}} \|\varepsilon_{T_2^j}\|_2^2 \Big)^{\frac{1}{2}} \right)
\end{aligned}
$$

[13] If time series T_1 and T_2 are aligned, then $\Pi_{T_2,[a_1^i, b_1^i]}$ always returns one single segment.

In the last step of the above Formula, $T_2|_{[a_1^i, b_1^i]}$ is not a segment that Plato precomputed in T_2. Thus, we need to use all the segments in T_2 covering $[a_1^i, b_1^i]$, i.e., $\Pi_{T_2,[a_1^i,b_1^i]}$. Similarly, we can compute $|\langle \boldsymbol{\varepsilon}_{T_1}, \boldsymbol{\varepsilon}_{T_2} \rangle|$ according to the domains of segments in T_2. Finally, IS chooses the minimal one between them. However, IS does not produce tight guarantees, Plato does not use it. Next, we show the tight computation called *OS*, which is used by Plato.

Optimal strategy (OS) OS (Algorithm 1) first computes an error distribution array E_{T_1} (resp. E_{T_2}) for T_1 (resp. T_2) (line 2) according to the domains of the segments as follows:

$$E_{T_1} = \left\{ \|\varepsilon_{T_1^i}\|_2 \times \left(\sum_{j \in \Pi_{T_2,[a_1^i,b_1^i]}} \|\varepsilon_{T_2^i}\|_2^2 \right)^{\frac{1}{2}} \middle| 1 \le i \le k_1 \right\}$$

$$E_{T_2} = \left\{ \|\varepsilon_{T_2^i}\|_2 \times \left(\sum_{j \in \Pi_{T_1,[a_2^i,b_2^i]}} \|\varepsilon_{T_1^i}\|_2^2 \right)^{\frac{1}{2}} \middle| 1 \le i \le k_2 \right\}$$

Then OS increases ε_1 (resp. ε_2) by adding the values from E_{T_1} (resp. E_{T_2}) (lines 4-7) and checks whether the current domain achieves the minimal errors (lines 8-17). If yes, OS adds the current domain (either $[start, b_1^{i_1}]$ or $[start, b_2^{i_2}]$) to the final segment combination list. After that, OS starts from a new domain and repeats the previous steps until all the segments are processed. The time complexity of OS is $O(k_1 + k_2)$.

Let $OPT(L_{T_1}, L_{T_2})$ be the segment combination returned by OS. Then $|\langle \boldsymbol{\varepsilon}_{T_1}, \boldsymbol{\varepsilon}_{T_2} \rangle|$ is computed as follows:

$$|\langle \boldsymbol{\varepsilon}_{T_1}, \boldsymbol{\varepsilon}_{T_2} \rangle| \le \sum_{[a,b] \in OPT(L_{T_1}, L_{T_2})} \left| \langle \boldsymbol{\varepsilon}_{T_1}|_{[a,b]}, \boldsymbol{\varepsilon}_{T_2}|_{[a,b]} \rangle \right|$$

$$\le \sum_{[a,b] \in OPT(L_{T_1}, L_{T_2})} \left(\left(\sum_{i \in \Pi_{T_1,[a,b]}} \|\boldsymbol{\varepsilon}_{T_1^i}\|_2^2 \right)^{\frac{1}{2}} \left(\sum_{i \in \Pi_{T_2,[a,b]}} \|\boldsymbol{\varepsilon}_{T_2^i}\|_2^2 \right)^{\frac{1}{2}} \right)$$

OS provides the optimal segment combination that produces the minimum $|\langle \boldsymbol{\varepsilon}_{T_1}, \boldsymbol{\varepsilon}_{T_2} \rangle|$. The tightness proof is presented in Appendix E.

Example 7. *Consider the two misaligned time series in Figure 9. The value of $\|\varepsilon_{T_i^j}\|_2$ for each segment T_i^j is labeled there. OS produces the segment combination $S = \{[a_1^1, b_1^1], [b_2^1, b_2^{101}]\}$ as visualized by the red lines. Then $|\langle \boldsymbol{\varepsilon}_{T_1}, \boldsymbol{\varepsilon}_{T_2} \rangle| = (3 \times (96 \times 1^2 + 2^2)^{\frac{1}{2}}) + (2 \times (100 \times 1^2)^{\frac{1}{2}}) = 3 \times 10 + 2 \times 10 = 50$. However, IS outputs $|\langle \boldsymbol{\varepsilon}_{T_1}, \boldsymbol{\varepsilon}_{T_2} \rangle| = min((3 \times 96 + 2 \times \sqrt{100 + 9}), (3 \times \sqrt{96 + 2^2 + 100 \times 2})) = \min(308.88, 230) = 230$, which is 4.6× larger than the result returned by OS.*

4.2.2 Orthogonal projection optimization. In this part, we present how to apply orthogonal property optimization to improve Formula 4①. Recall that in the aligned case (if the function family is in VS) we can apply the orthogonal property optimization to guarantee $\langle \varepsilon_{T_1^i}, f^*_{T_2}|_{[a_1^i, b_1^i]} \rangle = 0$. This is

Algorithm 1: Optimal segment combination (OS)

Input: Compressed segment representations L_{T_1}, L_{T_2}
Output: A segment combination *OPT*

1 $\varepsilon_1 = 0, \varepsilon_2 = 0, i_i = 0, i_2 = 0, start = 0, OPT = \emptyset, current = \emptyset$;
2 Compute E_{T_1} and E_{T_2};
3 **while** $i_1 < k_1$ *or* $i_2 < k_2$ **do**
4 **if** $b_1^{i_1} \le b_2^{i_2}$ **then**
5 $\varepsilon_1 += E_{T_1}[i_1 ++]$;
6 **else**
7 $\varepsilon_2 += E_{T_2}[i_2 ++]$;
8 **if** $\varepsilon_1 \le \varepsilon_2$ *AND* $b_1^{i_1} \ge b_2^{i_2}$ **then**
9 $current = [start, b_1^{i_1}]$;
10 $OPT \leftarrow OPT \cup \{current\}$;
11 $start = b_1^{i_1} + 1$;
12 $\varepsilon_2 \leftarrow \varepsilon_1$;
13 **if** $\varepsilon_2 \le \varepsilon_1$ *AND* $b_2^{i_2} \ge b_1^{i_1}$ **then**
14 $current = [start, b_2^{i_2}]$;
15 $OPT \leftarrow OPT \cup \{current\}$;
16 $start = b_2^{i_2} + 1$;
17 $\varepsilon_1 \leftarrow \varepsilon_2$;
18 Return *OPT*;

because $f^*_{T_2}|_{[a_1^i, b_1^i]} = f^*_{T_2^i}$, which is a function in the family. However, in misaligned case $\langle \varepsilon_{T_1^i}, f^*_{T_2}|_{[a_1^i, b_1^i]} \rangle$ cannot be guaranteed to be 0 since $f^*_{T_2}|_{[a_1^i, b_1^i]}$ may not be a function in the family. For example, in Figure 9 $T_2|_{[a_2^1, b_2^1]}$ is not a precomputed segment in T_2, it is just a subsegment. The restriction of the estimation function $f^*_{T_2^1}$ to this sub-domain $f^*_{T_2}|_{[a_1^1, b_1^1]}$ may not be a function in the family anymore.

To guarantee the restriction of the function from a bigger domain to a smaller domain is still in the same function family, we identify a function family group called linear scalable function family (LSF), which is subset of VS but superset of the polynomial function family.

Linear Scalable Function Family (LSF). Informally, a linear scalable family is a function family such that for any function f in that family and any translation $a - a'$, there is a function f' in that family such that $f'(x + a - a') = f(x)$ for all x in the domain. Definition 1 gives the formal definition.

Definition 1 (Linear scalable family (LSF)). *Let $\mathbb{F}$ be a function family defined in domain $[a, b]$, $\mathbb{F}$ is a linear scalable family if for any function $f \in \mathbb{F}$ and any range $[a', b'] \subseteq [a, b]$, there exists a function $f' \in \mathbb{F}$ such that Shift$(f|_{[a',b']}, a - a') = f'|_{[a, a+b'-a']}$.*

Lemma 2. *The polynomial family belongs to the linear scalable family.*

The proof of Lemma 2 is shown in Appendix F.

Recall that, in this paper, we study three different function family groups, i.e., ANY, VS, and LSF. Figure 3 shows the relation of the three function family groups and also provides example function families for each group.

In the following, we present how to use the orthogonal projection optimization in the misaligned case to improve Formula 4①. Let f_{T_1} (resp. f_{T_2}) be the function created from the concatenation of the individual estimation functions on the segments T_1^i ($i \in [1, k_1)$ (resp. T_2^j ($j \in [1, k_2]$)). That is $f_{T_1}|_{[a_1^i, b_1^i]} = f^*_{T_1^i}$ for all $i \in [1, k_1]$ and $f_{T_2}|_{[a_2^i, b_2^i]} = f^*_{T_2^i}$ for all $i \in [1, k_2]$. Then the Equation 4① in the misaligned environment can be reduced as follows. We highlight the parts that would disappear if the segments were aligned.

$$\sum_{i=1}^{k_1} \left(\|\varepsilon_{T_1^i}\|_2 \times \overbrace{\|f_{T_2}|_{[a_1^i, b_1^i]} - f^*_{T_1^i}\|_2}^{=0 \ if \ aligned} \right) + \sum_{i=1}^{k_2} \left(\|\varepsilon_{T_2^i}\|_2 \times \overbrace{\|f_{T_1}|_{[a_2^i, b_2^i]} - f^*_{T_2^i}\|_2}^{=0 \ if \ aligned} \right) \quad (5)$$

The proof of the tightness is in Appendix G.

Efficient Computation of the Error Guarantee Notice that both $\|f_{T_2}|_{[a_1^i, b_1^i]} - f^*_{T_1^i}\|_2$ and $\|f_{T_1}|_{[a_2^i, b_2^i]} - f^*_{T_2^i}\|_2$ can only be computed during query processing time, since only then the pairs of intersecting but misaligned segments become known. A brute force $O(n)$ method, where n is the size of the domain of the segment, would be to literally create the series of n data points predicted by the estimation functions and then perform the straightforward calculation/aggregation described by the formulas. Of course, such brute force approach would require CPU cycles that are proportional to conventional (non-approximate) query processing. We show that these formulas can be computed in $O(dim(\mathbb{F})^3)$ where $dim(\mathbb{F})$ is the dimension of the estimation function family. Obviously, the dimension is much smaller than the number of data points in a segment - that is why we employ compression in the first place. For example, for a 1-degree polynomial function family, $dim(\mathbb{F}) = 2$. The key intuition is to store the estimation function's coefficients in an orthonormal basis. The distance between two functions can be efficiently computed using the $dim(\mathbb{F})$ coefficients (in the orthonormal basis). Importantly, the orthonormal basis also allows us to compute the coefficients of the restriction of an estimation function in $O(dim(\mathbb{F})^3)$. The detailed algorithms and proofs complexity appear in the Appendix H.

Elimination of $\|f_T\|_2$. If T is compressed by a function in LSF [14], then $\|f_T\|_2$ can be safely eliminated. This is because the error guarantees provided by LSF can get rid of $\|f_T\|_2$ while those given by ANY or VS rely on $\|f_T\|_2$.

[14] And we know that it many only be combined with other segments compressed by a function in LSF.

	avg # of data points in each time series	# of time series	resolution
HF	126, 059, 817	15	millisecond
HI	2, 676, 311	14	second
HB	1, 669, 835	16	minute
HA	1, 587, 258	11	minute

Table 5: Data Characteristics

	# of coefficients	# of error measures
Polynomial	2	1
Gaussian	4	3

Table 6: Number of coefficients and error measures

5 EXPERIMENTS

5.1 Environment and Setting

All experiments were conducted on a computer with a 4^{th} Intel *i*7-4770 processor (3.6 GHz), 16 GB RAM, running Ubuntu 14.04.1. The algorithms were implemented in *C++* and were compiled with *g++* 4.8.4.

Dataset. We evaluated all the error guarantee methods on four real-life datasets: Historical Forex Data (HF), Historical IoT Data (HI), Historical Bitcoin Exchanges Data (HB), and Historical Air Quality Data (HA). Table 5 summarizes the data characteristics. The detailed description of each dataset is presented in Appendix I.

Segmentation algorithms. We adopt the fixed-length segmentation (FL) and the sliding window algorithm (SW). The segments produced by the FL have equal lengths, and will be utilized in our aligned experiments, while the segments created by the SW have variable lengths and are used in our misaligned experiments.

Estimation function families. Following the prior work lessons [19, 36], we choose the 1-degree polynomial function family ($\{ax + b | a, b \in R\}$) and the Gaussian function family ($\{a \exp\left(\frac{-(x-b)^2}{2c^2}\right) + d | a, b, c, d \in R\}$) as representatives to compress the time series. Notice that the Gaussian function family is in ANY, while the polynomial function family is in LSF (also in VS). Table 6 summarizes the number of coefficients and error measures stored for each segment compressed by the corresponding estimation functions.

Queries We evaluate the correlation TSA over all the time series pairs in each dataset. The corresponding SQL queries are shown in Appendix I. All the error guarantees and true errors reported in the following are the average values (including the standard variances) across all correlations in a dataset.

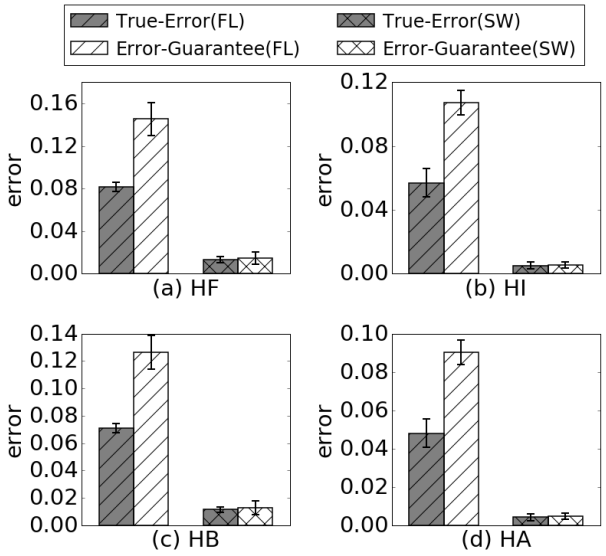


Figure 10: True errors and error guarantees in aligned (FL) and misaligned cases (SW). The True-Error(SW) are 0.0132 and 0.00508 in (a) and (b).

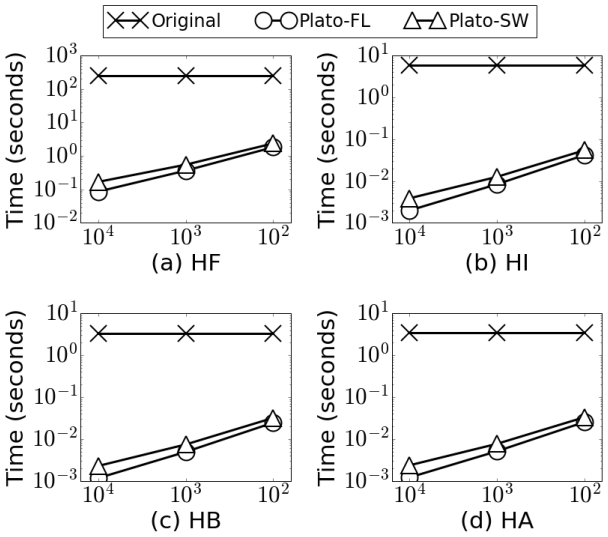


Figure 11: Running time of TSAs in aligned and misaligned cases.

5.2 Experimental Results

We evaluate the error guarantees for TSAs over aligned, fixed-length time series segmentations and misaligned, variable-length time series segmentations. In order to provide a fair comparison, we fix the space cost for both cases, i.e., they have the same compression ratios.

Error Guarantees Quality Figure 10 reports the absolute true errors and the error guarantees of the correlation TSAs in the aligned/fixed-length (FL) and misaligned/variable-length (SW) cases using the polynomial function family. Since the TSAs are correlations, the approximate results may range between 1 (perfect correlation) and -1 (perfect reverse correlation), with 0 meaning no correlation at all.

Under the same compression ratio [15] the variable-length error guarantees are much smaller than the fixed-length error guarantees. In Figure 10, the misaligned Error-Guarantee (SW) is 10× ∼ 20× smaller than the aligned Error-Guarantee (FL) on the average (ranging the compression ratio from 10, 000 to 100). This is mainly because, as it has already been known, variable-length allows for much better estimation. Indeed, notice the misaligned true errors are also much smaller than the aligned true errors. For example, In Figure 10, True-Error(SW) is 6× ∼ 11× smaller than True-Error(FL) on the average.

Importantly, the error guarantees are close to the true errors, especially for the misaligned error guarantees, which matter most practically. In particular, Error-Guarantee(SW) is only 1.08× ∼ 1.11× larger than the True-Error(SW) in HF and HI respectively (on the average). Furthermore, they are very small in absolute terms. This indicates the high quality and practicality of AI (Amplitude-independent) error guarantees.

Run time performance Figure 11 reports the total running time of the correlation TSAs over (i) the original time series (Original), (ii) the time series segmented into a fixed length, aligned segments (Plato-FL) and (iii) time series segmented into misaligned, variable-length segments by SW (Plato-SW). The estimation function family is the polynomial family. The x-axis is the compression ratio (from 10000 to 100).

Both Plato-FL and Plato-SW outperform vastly the Original in all the datasets. For example, when the compression ratio is 1000, Plato-FL and Plato-SW are about three orders of magnitude faster than Original.

Plato-SW is about 1.8× slower than Plato-FL due to the intricacy of the segment combination selection algorithm. However, a mere 80% penalty is a minor price to pay for the orders-of-magnitude superior error guarantees delivered by misaligned/variable-length segmentations.

Comparison with sampling In this part, we compare (i) the space cost and (ii) the runtime performance of Plato with the sampling methods when providing similar error guarantees. We use a uniform random sampling scheme with a global seed in order to create a samples database. We also assume knowledge of minimums and maximums. That is, let $X_1, ..., X_n$ be the random variables such that $d_{min} \leq X_i \leq d_{max}$ for all i where $X_i = d_i^{T_1} \times d_i^{T_2}$, $d_{min} = \min\{d_i^{T_1}\} \times \min\{d_i^{T_2}\}$, and $d_{max} = \max\{d_i^{T_1}\} \times \max\{d_i^{T_2}\}$. Let $R = \sum_{i=1}^{n} X_i$ and ε be the error guarantee. Using the Chernoff bounds [15],

[15]Compression ratio is the size of the original data over the size of the compressed data.

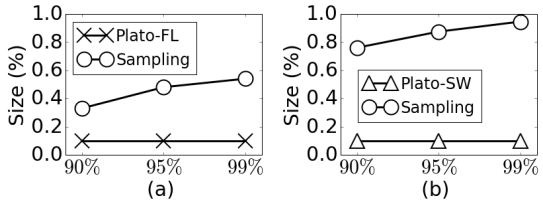


Figure 12: Space cost of sampling and Plato when providing the same error guarantees.

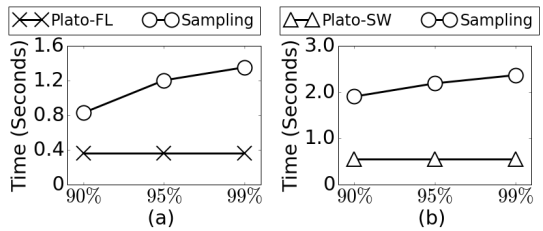


Figure 13: Running time of sampling and Plato when providing the same error guarantees.

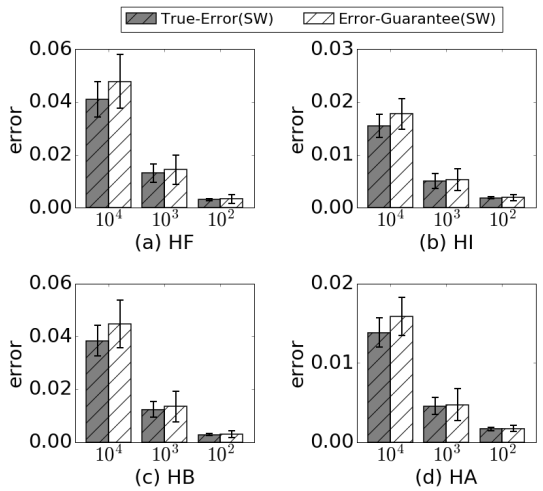


Figure 14: Effect of compression ratios.

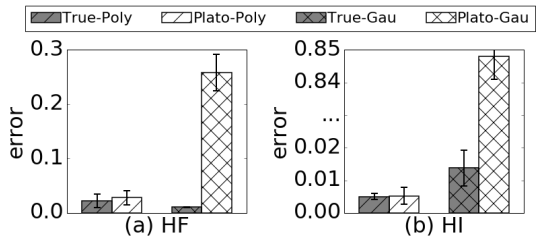


Figure 15: Effect of estimation function families.

we can obtain the minimal sample size needed in order to achieve the desired error guarantee with certain confidence.

Figure 12 reports the sizes (as percentage to the original data size) of sampled data points in order to provide similar error guarantees with the Plato-FL (the error guarantee of TSAs over aligned, fixed-length time series produced by FL) and Plato-SW (the error guarantee of TSAs over misaligned time series produced by SW) with 1000 compression ratio in HF respectively. Figure 13 shows the corresponding runtime cost. To achieve similar error guarantees, sampling needs more space and more time than Plato. We define “similar" to mean 90%, or 95% or 99% confidence - in contrast to Plato’s deterministic, 100% confidence guarantees.

5.2.1 Effects of Individual Factors. In this part, we study the effects of (i) compression ratios, (ii) estimation function families, (iii) orthogonal optimizations, and (iv) segment combination selection strategies.

Compression ratios In order to isolate the effect of the compression ratios, [16] we fix the estimation function family to be polynomials and fix the segment list building algorithm to be SW. In Figure 14, we change the compression ratios from 10, 000 to 100 by controlling the error threshold values and report the corresponding true errors (True-Error(SW)) and the error guarantees (Error-Guarantee(SW)).

Naturally, higher compression ratios lead to smaller true errors and error guarantees. For example, in Figure 14(a), the true error and error guarantee with 100 compression ratio are 13.32× and 15.58× smaller than those with 10, 000 compression ratio on the average. Importantly, the error guarantees provided by Plato are close to the true error in all the datasets and are generally small in absolute terms (with the relative exception of 10, 000 compression on HF). Again, this indicates the high quality of the error guarantees provided by Plato.

Estimation function families In order to isolate the effect of the estimation function families, we fix the segment list building algorithm to be SW and fix the compression ratio to 1000. Figure 15 presents the true errors and the error guarantees for TSAs over time series compressed by polynomial functions (True-Error(Poly), Error-Guarantee(Poly)) and Gaussian functions (True-Error(Gau), Error-Guarantee(Gau)) respectively.

The error guarantees with estimation functions from LSF (polynomials) are significantly smaller than those with estimation functions in ANY (Gaussians). In Figure 15(a), Error-Guarantee(Poly) (in LSF and VS) is about 10× smaller than Error-Guarantee(Gau) (in ANY) on the average and in Figure 15(b), Error-Guarantee(Poly) (in LSF and VS) is about 160× smaller than Error-Guarantee(Gau) (in ANY) on the average. Notice that the error guarantees provided by Plato-Poly is AI, while those of Plato-Gau are not. So the results show that AI error guarantees are practical while non-AI error guarantees are not. Interestingly, True-Error(Gau) is smaller than True-Error(Poly) in the HF dataset, which indicates that Gaussian functions model HF data better than the

[16] Compression ratio is the size of the original data over the size of the compressed data.

polynomial functions - not surprising given the more random movements of financial data. The guarantees produced by the polynomials are far better thanks to AI.

Effect of Orthogonal Optimization and LSF To measure the effect on error guarantees of the orthogonal optimization (and its extension to misaligned segmentations, enabled by LSF) we fix the estimation function family to the polynomials, which are LSF and, trivially, are also in ANY. We use both the general error guarantees of ANY (Error-Guarantee(ANY)) and the specialized error guarantees of LSF (Error-Guarantee(LSF)) for TSAs over misaligned segments compressed by polynomial functions (using variable-length segmentations with the SW algorithm). We fix the compression ratio to 1000. As shown in Figure 16, the error guarantee for LSF certifies that the true result is just within ±0.0137 in HF and within ±0.0052 in HI.

Segment combination selection strategies To isolate the quality effect of employing the optimal segment combination selection strategy (OS) we compare it with IS strategy (the straightforwad method mentioned in Section 4.2.1) on a case of variable-length compression with an LSF function family (polynomials). Figure 17 shows that Plato-OS is about 5× smaller than Plato-MS on the average. In addition, the running time of Plato-IS and Plato-OS are close. For example, the running time of Plato-IS and Plato-OS are 0.536 and 0.548 seconds in HF respectively.

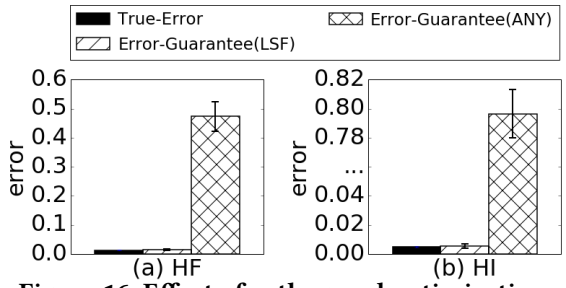


Figure 16: Effect of orthogonal optimization.

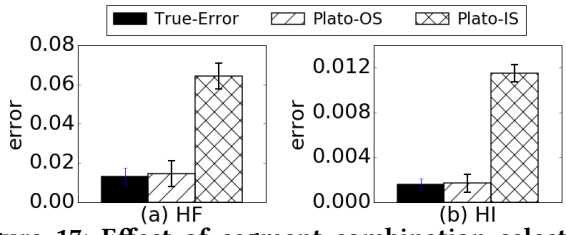


Figure 17: Effect of segment combination selection strategies.

6 RELATED WORK

Approximate query processing (AQP) and data compression have been widely studied. whose most relevant aspects are summarized next.

AQP with probabilistic error guarantees. Approximate query processing using *sampling* [1, 4, 37, 44] computes approximate answers by appropriately evaluating the queries on small samples of the data, e.g., STRAT [4], SciBORQ [44], and BlinkDB [1]. Such approaches typically leverage statistical inequalities and the central limit theorem to compute the confidence interval (or variance) of the computed approximate answer. As a result, their error guarantees are probabilistic - as opposed to this work's deterministic (100% confidence) ones. Note however that, unlike sampling, our compression-based techniques are tuned for time series and continuous data.

AQP with deterministic error guarantees. Approximately answering queries while providing deterministic error guarantees has been successfully applied in many applications [9, 14, 29, 32, 41, 42]. However, existing work in the area has focused on simple aggregation queries that involve only a single time series (or table) and aggregates such as SUM, COUNT, MIN, MAX and AVG. Our work extends the prior work, as it addresses analytics over multiple compressed time series such as correlation, cross-correlation. In addition, this work is the first one to categorize compression function families based on their suitability for error guarantees.

Data summarizations and compressions The database community has mostly focused on creating summarizations (also referred to as synopses or sketches) that can be used to answer specific queries. These include among others histograms [18, 40, 43, 47] (e.g., EquiWidth and EquiDepth histograms [40], V-Optimal histograms [18], and Hierarchical Model Fitting (HMF) histograms [47]), used among other for cardinality estimation [18] and selectivity estimation [41]. The signal processing community produced a variety of methods that can be used to compress time series data and thus are more relevant to the present work, as they provide the underlying compressions. These include among others the Piecewise Aggregate Approximation (PAA) [22], and the Piecewise Linear Representation (PLR) [19]. Plato is orthogonal to those data summarization and compression techniques.

7 SUMMARY AND FUTURE DIRECTION

This work indicates that deterministic error guarantees are feasible and practical, given the appropriate combination of error measures and estimation function family. Future work may develop such combinations for other important families also. Note that the tightness results of this paper do not preclude the future development of practical and theoretically-sound deterministic error guarantees for families are currently outside the LSF (or outside the VS in the case of aligned series). Researchers may come up with other interesting properties of function families outside LSF (or VS) and deliver good error guarantees, based on such properties.

REFERENCES

[1] Sameer Agarwal, Barzan Mozafari, Aurojit Panda, Henry Milner, Samuel Madden, and Ion Stoica. 2013. BlinkDB: queries with bounded errors and bounded response times on very large data. In *EuroSys*. 29–42.

[2] Saeed Reza Aghabozorgi, Ali Seyed Shirkhorshidi, and Ying Wah Teh. 2015. Time-series clustering - A decade review. *Inf. Syst.* 53 (2015), 16–38.

[3] Kin-pong Chan and Ada Wai-Chee Fu. 1999. Efficient Time Series Matching by Wavelets. In *ICDE*. 126–133.

[4] Surajit Chaudhuri, Gautam Das, and Vivek Narasayya. 2007. Optimized stratified sampling for approximate query processing. *TODS* 32, 2 (2007), 9.

[5] Surajit Chaudhuri, Bolin Ding, and Srikanth Kandula. 2017. Approximate query processing: no silver bullet. In *Sigmod*. ACM, 511–519.

[6] Lei Chen and Raymond T. Ng. 2004. On The Marriage of Lp-norms and Edit Distance. In *VLDB*. 792–803.

[7] Ward Cheney and David Kincaid. 2009. Linear algebra: Theory and applications. *The Australian Mathematical Society* 110 (2009).

[8] ByoungSeon Choi. 2012. *ARMA model identification*. Springer Science & Business Media.

[9] Graham Cormode, Flip Korn, S. Muthukrishnan, and Divesh Srivastava. 2005. Effective Computation of Biased Quantiles over Data Streams. In *ICDE*. 20–31.

[10] DGT Denison, BK Mallick, and AFM Smith. 1998. Automatic Bayesian curve fitting. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 60, 2 (1998), 333–350.

[11] Andrew Eisenberg and Jim Melton. 2002. SQL/XML is making good progress. *ACM Sigmod Record* 31, 2 (2002), 101–108.

[12] Christos Faloutsos, M. Ranganathan, and Yannis Manolopoulos. 1994. Fast Subsequence Matching in Time-Series Databases. In *SIGMOD*. 419–429.

[13] Alex Galakatos, Andrew Crotty, Emanuel Zgraggen, Carsten Binnig, and Tim Kraska. 2017. Revisiting reuse for approximate query processing. *Proceedings of the VLDB Endowment* 10, 10 (2017), 1142–1153.

[14] Michael Greenwald and Sanjeev Khanna. 2001. Space-Efficient Online Computation of Quantile Summaries. In *SIGMOD*. 58–66.

[15] Torben Hagerup and Christine Rüb. 1990. A guided tour of Chernoff bounds. *Information processing letters* 33, 6 (1990), 305–308.

[16] Paul Richard Halmos. 2012. *Finite-dimensional vector spaces*. Springer Science & Business Media.

[17] Walter Hoffmann. 1989. Iterative algorithms for Gram-Schmidt orthogonalization. *Computing* 41, 4 (1989), 335–348.

[18] Yannis E. Ioannidis and Viswanath Poosala. 1995. Balancing Histogram Optimality and Practicality for Query Result Size Estimation. In *SIGMOD*. 233–244.

[19] Eamonn Keogh. 1997. Fast similarity search in the presence of longitudinal scaling in time series databases. In *ICTAI*. 578–584.

[20] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. 2001. An online algorithm for segmenting time series. In *ICDM*. 289–296.

[21] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. 2004. Segmenting time series: A survey and novel approach. In *Data mining in time series databases*. World Scientific, 1–21.

[22] Eamonn J. Keogh, Kaushik Chakrabarti, Michael J. Pazzani, and Sharad Mehrotra. 2001. Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. *KAIS* 3, 3 (2001), 263–286.

[23] Eamonn J. Keogh and Michael J. Pazzani. 1998. An Enhanced Representation of Time Series Which Allows Fast and Accurate Classification, Clustering and Relevance Feedback. In *KDD*. 239–243.

[24] Eamonn J Keogh and Michael J Pazzani. 1999. Relevance feedback retrieval of time series data. In *SIGIR*. 183–190.

[25] Kyoung-jae Kim. 2003. Financial time series forecasting using support vector machines. *Neurocomputing* 55, 1-2 (2003), 307–319.

[26] Antti Koski, Martti Juhola, and Merik Meriste. 1995. Syntactic recognition of ECG signals by attributed finite automata. *Pattern Recognition* 28, 12 (1995), 1927–1940.

[27] Geza Kovács, Shay Zucker, and Tsevi Mazeh. 2002. A box-fitting algorithm in the search for periodic transits. *Astronomy & Astrophysics* 391, 1 (2002), 369–377.

[28] Arun Kumar, Robert McCann, Jeffrey F. Naughton, and Jignesh M. Patel. 2015. Model Selection Management Systems: The Next Frontier of Advanced Analytics. *SIGMOD Record* 44, 4 (2015), 17–22.

[29] Iosif Lazaridis and Sharad Mehrotra. 2001. Progressive Approximate Aggregate Queries with a Multi-Resolution Tree Structure. In *SIGMOD*. 401–412.

[30] Iosif Lazaridis and Sharad Mehrotra. 2003. Capturing Sensor-Generated Time Series with Quality Guarantees. In *Proceedings of the 19th International Conference on Data Engineering, March 5-8, 2003, Bangalore, India*. 429–440.

[31] Chung-Sheng Li, Philip S. Yu, and Vittorio Castelli. 1998. MALM: A Framework for Mining Sequence Database at Multiple Abstraction Levels. In *CIKM*. 267–272.

[32] Gurmeet Singh Manku, Sridhar Rajagopalan, and Bruce G. Lindsay. 1998. Approximate Medians and other Quantiles in One Pass and with Limited Memory. In *SIGMOD*. 426–435.

[33] Jonathan Mei and José M. F. Moura. 2017. Signal Processing on Graphs: Causal Modeling of Unstructured Data. *IEEE Trans. Signal Processing* 65, 8 (2017), 2077–2092.

[34] Michael D. Morse and Jignesh M. Patel. 2007. An efficient and accurate method for evaluating time series similarity. In *SIGMOD*. 569–580.

[35] Edward Nelson. 1973. Probability theory and Euclidean field theory. In *Constructive quantum field theory*. Springer, 94–124.

[36] Zhuokun Pan, Yueming Hu, and Bin Cao. 2017. Construction of smooth daily remote sensing time series data: a higher spatiotemporal resolution perspective. *Open Geospatial Data, Software and Standards* 2, 1 (2017), 25.

[37] Niketan Pansare, Vinayak R. Borkar, Chris Jermaine, and Tyson Condie. 2011. Online Aggregation for Large MapReduce Jobs. *PVLDB* 4, 11 (2011), 1135–1145.

[38] Sanghyun Park, Dongwon Lee, and Wesley W Chu. 1999. Fast retrieval of similar subsequences in long sequence databases. In *KDEX*. 60–67.

[39] John S Philo. 1997. An improved function for fitting sedimentation velocity data for low-molecular-weight solutes. *Biophysical Journal* 72, 1 (1997), 435–444.

[40] Gregory Piatetsky-Shapiro and Charles Connell. 1984. Accurate Estimation of the Number of Tuples Satisfying a Condition. In *SIGMOD*. 256–276.

[41] Viswanath Poosala, Yannis E. Ioannidis, Peter J. Haas, and Eugene J. Shekita. 1996. Improved Histograms for Selectivity Estimation of Range Predicates. In *SIGMOD*. 294–305.

[42] Navneet Potti and Jignesh M. Patel. 2015. DAQ: A New Paradigm for Approximate Query Processing. *PVLDB* 8, 9 (2015), 898–909.

[43] Frederick Reiss, Minos N. Garofalakis, and Joseph M. Hellerstein. 2006. Compact Histograms for Hierarchical Identifiers. In *VLDB*. 870–881.

[44] Lefteris Sidirourgos, Martin L. Kersten, and Peter A. Boncz. 2011. SciBORQ: Scientific data management with Bounds On Runtime and Quality. In *CIDR*. 296–301.

[45] Mikio Tobita. 2016. Combined logarithmic and exponential function model for fitting postseismic GNSS time series after 2011 Tohoku-Oki earthquake. *Earth, Planets and Space* 68, 1 (2016), 41.

[46] Michail Vlachos, Dimitrios Gunopulos, and George Kollios. 2002. Discovering Similar Multidimensional Trajectories. In *ICDE*. 673–684.

[47] Hai Wang and Kenneth C. Sevcik. 2008. Histograms based on the minimum description length principle. *VLDB J.* 17, 3 (2008), 419–442.

[48] WJ Wiscombe and JW Evans. 1977. Exponential-sum fitting of radiative transmission functions. *J. Comput. Phys.* 24, 4 (1977), 416–444.

A SEGMENTATION ALGORITHM

We summarize the state-of-the-art time series segmentation algorithms, which can be classified into two categories: (i) Fix-length segmentation (FL), which partitions a time series based on fixed time windows. The segments produced by the FL have equal lengths, and will be utilized in our aligned-segments experiments; and (ii) Variable-length segmentation. There are three groups of algorithms produce variable-length segmentations: the Top-down methods [31, 38], the Bottom-up approaches [23, 24] and the Sliding-window techniques [20, 26]. Among them, the Sliding-window (SW) has been proven to be more efficient than the Top-down and the Bottom-up methods [20, 21]. Thus, we choose the Sliding-window (SW) as the representative variable length segmentation algorithm in our experiments. The segments created by the SW have variable lengths [21] and are used in our misaligned-segments experiments. Figure 1 adopts the SW method, which produces variable-length segments.

B PROPAGATING ERROR GUARANTEES IN ARITHMETIC OPERATORS

For arithmetic operator $Ar_1 \otimes Ar_2$ where $\otimes\{+, -, \times, \div\}$. If both Ar_1 and Ar_2 are scalar values, the Plato gives accurate answers. Then we discuss in the following two cases: (i) Ar_1 or Ar_2 is an aggregation result produced by Plato, and (ii) both Ar_1 and Ar_2 are aggregation results produced by Plato.

Case 1. Without loss of generality, we assume Ar_1 is an aggregation operator and Ar_2 is a scalar value. Let $\hat{R}$ be the approximate answer provided by Plato for Ar_1 and $\hat{\varepsilon}$ is the corresponding error guarantee. The approximate answer and the error guarantee of $Ar_1 \otimes Ar_2$ is summarized in Table 18.

Case 2. Both Ar_1 and Ar_2 are aggregation operators. Let $\hat{R}_1$ (resp. $\hat{R}_2$) and $\hat{\varepsilon}_1$ (resp. $\hat{\varepsilon}_2$) be the approximate answer and error guarantee provided by Plato for Ar_1 and Ar_2 respectively. The approximate answer and the error guarantee of $Ar_1 \otimes Ar_2$ is summarized in Table 19.

C ERROR GUARANTEES OF OTHER EXPRESSIONS

In this part, we present the error guarantees for the other core expressions, i.e., (i) Sum(Constant(v, a, b)), (ii) Sum(Shift(T, k)), (iii) Sum($T_1 + T_2$), and (iv) Sum($T_1 - T_2$).

Error guarantee of Sum(Constant(v, a, b)). For the time series T=Constant(v, a, b), the estimation function is $f_T^* = v$ [17], then the error measures stored by Plato are ($\|\varepsilon_T\|_2 = 0$, $\|f_T\|_2 = v\sqrt{b-a+1}$, $\gamma_T = 0$). The error guarantee of Sum(Constant(v, a, b)) is $\gamma_T = 0$.

Error guarantee of Sum(Shift(T, k)). For the time series T=Shift(T, k), we need to use the error measures ($\|\varepsilon_T\|_2$, $\|f_T\|_2$, γ_T) defined in domain $[a+k, b+k]$. Then the error guarantee of Sum(Shift(T, k)) is γ_T.

Error guarantees of Sum(T_1+T_2) and Sum(T_1-T_2). Given two time series $T_1 = (T_1^1, ..., T_1^{k_1})$ and $T_2 = (T_2^1, ..., T_2^{k_2})$. Then the error measures of $T = T_1+T_2$ are ($\|\varepsilon_T\|_2$, $\|f_T\|_2$, γ_T) where $\|\varepsilon_T\|_2 = \sum_i^{k_1} \|\varepsilon_{T_1^i}\|_2 + \sum_i^{k_2} \|\varepsilon_{T_2^i}\|_2$, $\|f_T\|_2 = \sum_i^{k_1} \|f_{T_1^i}\|_2 + \sum_i^{k_2} \|f_{T_2^i}\|_2$, and $\gamma_T = \sum_i^{k_1} \gamma_{T_1^i} + \sum_i^{k_2} \gamma_{T_2^i}$. And the error guarantees of Sum($T_1 + T_2$) is $\gamma_T = \sum_i^{k_1} \gamma_{T_1^i} + \sum_i^{k_2} \gamma_{T_2^i}$. The error measures of $T_1 - T_2$ are the same with those of $T_1 + T_2$.

Operator	approximate answer	error guarantee
$Ar_1 + Ar_2$	$\hat{R} + Ar_2$	$\hat{\varepsilon}$
$Ar_1 - Ar_2$	$\hat{R} - Ar_2$	$\hat{\varepsilon}$
$Ar_1 \times Ar_2$	$\hat{R} \times Ar_2$	$\hat{\varepsilon} \times Ar_2$
$Ar_1 \div Ar_2$	$\hat{R} \div Ar_2$	$\hat{\varepsilon} \div Ar_2$

Figure 18: Error guarantee propagation in case 1.

Operator	approximate answer	error guarantee
$Ar_1 + Ar_2$	$\hat{R}_1 + \hat{R}_2$	$\hat{\varepsilon}_1 + \hat{\varepsilon}_2$
$Ar_1 - Ar_2$	$\hat{R}_1 - \hat{R}_2$	$\hat{\varepsilon}_1 + \hat{\varepsilon}_2$
$Ar_1 \times Ar_2$	$\hat{R}_1 \times \hat{R}_2$	$\hat{\varepsilon}_1\hat{R}_2 + \hat{\varepsilon}_2\hat{R}_1 + \hat{R}_1\hat{R}_2$
$Ar_1 \div Ar_2$	$\hat{R}_1 \div \hat{R}_2$	$\frac{(\hat{\varepsilon}_1\hat{R}_2 + \hat{\varepsilon}_2\hat{R}_1)}{(\hat{R}_2 - \hat{\varepsilon}_2)\hat{R}_2}$

Figure 19: Error guarantee propagation in case 2.

D COMPUTATION OF FORMULA 2

$$
\begin{aligned}
\varepsilon &= \left| \sum_{i=a}^{b} \boldsymbol{T}_1[i]\boldsymbol{T}_2[i] - \sum_{i=a}^{b} f_{T_1}^*(i) f_{T_2}^*(i) \right| \\
&= \left| \sum_{i=1}^{k} \left(\sum_{j=a_i}^{b_i} \boldsymbol{T}_1[i]\boldsymbol{T}_2[i] - \sum_{j=a_i}^{b_i} f_{T_1}^*(i) f_{T_2}^*(i) \right) \right| \\
&= \left| \sum_{i=1}^{k} \left(\langle \boldsymbol{\varepsilon}_{T_1^i}, f_{T_2^i}^* \rangle + \langle \boldsymbol{\varepsilon}_{T_2^i}, f_{T_1^i}^* \rangle + \langle \boldsymbol{\varepsilon}_{T_1^i}, \boldsymbol{\varepsilon}_{T_2^i} \rangle \right) \right| \\
&\le \left| \sum_{i=1}^{k} \langle \boldsymbol{\varepsilon}_{T_1^i}, f_{T_2^i}^* \rangle \right| + \left| \sum_{i=1}^{k} \langle \boldsymbol{\varepsilon}_{T_2^i}, f_{T_1^i}^* \rangle \right| + \left| \sum_{i=1}^{k} \langle \boldsymbol{\varepsilon}_{T_1^i}, \boldsymbol{\varepsilon}_{T_2^i} \rangle \right| \\
&\le \sum_{i=1}^{k} \left(\|\varepsilon_{T_1^i}\|_2 \|\varepsilon_{T_2^i}\|_2 + \|\varepsilon_{T_1^i}\|_2 \|f_{T_2^i}\|_2 + \|f_{T_1^i}\|_2 \|\varepsilon_{T_2^i}\|_2 \right)
\end{aligned}
$$

[17] Under the reasonable assumption that any practical family will also include the constant function.

τ	Expression	Comment
p_i	$\{\sum_{i=0}^{i} a_i x^i \mid a_i \in R\}$	i-degree Polynomial
g	$\{a\exp(\frac{-(x-b)^2}{2c^2}) + d \mid a,b,c,d \in R\}$	Gaussian
l	$\{\frac{L}{1+\exp(ax+b)} + c \mid L,a,b \in R\}$	Logistic

Table 7: Example function family identifiers

	Generated Error Measures		
	$\|\varepsilon_T\|_2$	$\|f_T\|_2$	γ_T
$T = T_1 + T_2$	$\|\varepsilon_{T_1}\|_2 + \|\varepsilon_{T_2}\|_2$	$\|f_{T_1}\|_2 + \|f_{T_2}\|_2$	$\gamma_{T_1} + \gamma_{T_2}$
$T = T_1 - T_2$	$\|\varepsilon_{T_1}\|_2 + \|\varepsilon_{T_2}\|_2$	$\|f_{T_1}\|_2 + \|f_{T_2}\|_2$	$\gamma_{T_1} + \gamma_{T_2}$
$T = T_1 \times T_2$	$\|\varepsilon_{T_1}\|_2\|\varepsilon_{T_2}\|_2$	$\|f_{T_1}\|_2\|f_{T_2}\|_2$	$\|\varepsilon_{T_1}\|_2\|\varepsilon_{T_2}\|_2$
	$+\|\varepsilon_{T_1}\|_2\|f_{T_2}\|_2$		$\|\varepsilon_{T_1}\|_2\|\varepsilon_{T_2}\|_2$
	$+\|\varepsilon_{T_2}\|_2\|f_{T_2}\|_2$		$+\|\varepsilon_{T_1}\|_2\|f_{T_2}\|_2$
			$+\|\varepsilon_{T_2}\|_2\|f_{T_2}\|_2$

Table 8: Error measures propagation. $\gamma_{T=T_1\times T_2}$ has two possible computation methods. If the estimation function family forms a vector space, then we use the one in the grey background.

The last inequality is obtained by Applying the Hölder inequality [7].

E PROOF OF THE OPTIMALITY OF OS

Proof. We use a proof by induction to show that the error guarantee produced by the segment combination returned by OS (Algorithm 1) is optimal.

Let $OPT(\tilde{T_1}, \tilde{T_2}) = \{[a_i, b_i] | i \in [1, m]\}$ be the segment combination returned by OS. First, let's see the base case where $OPT(\tilde{T_1}, \tilde{T_2}) = \{[a_1, b_1]\}$ has only one domain. There are two cases depending on $b_1 = b_1^1$ or $b_1 = b_2^t$ where $\Pi_{T_2,[a_1,b_1]} = \{T_2^1, ..., T_2^t\}$.

Case 1: $b_1 = b_1^1$. Since OS chooses $[a_1, b_1^1]$ as the domain, then $b_2^1 \le b_1^1$. Otherwise, OS does not choose $[a_1, b_1^1]$. This is because, (i) if $E_{T_1}[0] \ge E_{T_2}[0]$ then OS will choose $[a_1, b_2^1]$ instead; or (ii) if $E_{T_1}[0] < E_{T_2}[0]$, then OS can not enter the loop in lines 8 - 17. Since $b_2^1 \le b_1^1$, then we know $E_{T_1}[0] \le E_{T_2}[0]$, so the error guarantee is $\|\varepsilon_{T_1^1}\|_2\|\varepsilon_{T_2^1}\|_2$, which is the minimal error guarantee in domain $[a_1, b_1]$. Assume we split the domain $[a_1, b_1]$ into p ($p \ge 2$) sub-domains $[a_1, c_1], [c_1, c_2], ..., [c_{p-1}, b_1]$, then the error guarantee is $p\|\varepsilon_{T_1^1}\|_2\|\varepsilon_{T_2^1}\|_2$, therefore, domain $[a_1, b_1] = [a_1^1, b_1^1]$ produces the minimal error guarantee.

Case 2: $b_1 = b_2^t$. Since OS chooses $[a_1, b_2^t]$ as the domain, we know that the error guarantee is

$$(\sum_{i\in\Pi_{T_2,[a_2^1,b_2^t]}} \|\varepsilon_{T_2^1}\|_2^2)^{\frac{1}{2}} (\sum_{i\in\Pi_{T_1,[a_2^1,b_2^t]}} \|\varepsilon_{T_1^1}\|_2^2)^{\frac{1}{2}}$$

which is less than $\|\varepsilon_{T_1^1}\|_2(\sum_{i\in\Pi_{T_2,[a_1^1,b_1^1]}} \|\varepsilon_{T_2^1}\|_2^2)^{\frac{1}{2}}$. If we split $[a_2^1, b_2^t]$ into several sub-domains, the error guarantee is greater than $\|\varepsilon_{T_1^1}\|_2(\sum_{i\in\Pi_{T_2,[a_1^1,b_1^1]}} \|\varepsilon_{T_2^1}\|_2^2)^{\frac{1}{2}}$. Thus, $[a_1, b_1] = [a_2^1, b_2^t]$ produces the minimal error guarantee.

Suppose $OPT(\tilde{T_1}, \tilde{T_2}) = \{[a_i, b_i] | i \in [1, m-1]\}$ produces the minimal error guarantee, then for the case $OPT(\tilde{T_1}, \tilde{T_2}) = \{[a_i, b_i] | i \in [1, m]\}$, we only need to prove the last domain $[a_m, b_m]$ produces the minimal error guarantee, which is the same to the base case. □

F PROOF OF LEMMA 1

Proof. Let $\mathbb{F} = \{\sum_i \alpha_i t^i \mid \alpha_i \in R\}$ be a polynomial function family defined on $[a, b]$. The restriction of $f \in \mathbb{F}$ on $[a', b'] \subseteq [a, b]$ is $f|_{[a',b']} = (a', b', [\sum_i \alpha_i (a')^i, ..., \sum_i \alpha_i (b')^i])$. The shift of $f|_{[a',b']}$ to $a - a'$ steps is $\text{Shift}(f|_{[a',b']}, a - a')$ $= (a, a + b' - a', [\sum_i \alpha_i (a')^i, ..., \sum_i \alpha_i (b')^i])$. $[\sum_i \alpha_i (a')^i, ..., \sum_i \alpha_i (b')^i]$ can be transformed into $[\sum_i \beta_i (a)^i, ..., \sum_i \beta_i (a + b' - a')^i]$ such that $\beta_i = \frac{\alpha_i (a'+k)^i}{(a+k)^k}$ for all $i \in [a, a + b' - a']$. Let $f' = \sum_i \beta_i t^i$ be a function in $\mathbb{F}$. Thus $f'|_{[a,a+b'-a']} = [\sum_i \beta_i (a)^i, ..., \sum_i \beta_i (a+b'-a')^i] = Shift(f|_{[a',b']}, a-a')$. □

G PROOF THE CORRECTNESS AND TIGHTNESS OF EQUATION 5

Proof. Let $\varepsilon_{Sum(T_1\times T_2)}$ be the true error of $Sum(T_1 \times T_2)$.

$$\begin{aligned}\varepsilon_{Sum(T_1\times T_2)} &= |\langle \boldsymbol{\varepsilon}_{T_1}, \boldsymbol{f}_{T_2}\rangle + \langle \boldsymbol{\varepsilon}_{T_2}, \boldsymbol{f}_{T_1}\rangle + \langle \boldsymbol{\varepsilon}_{T_1}, \boldsymbol{\varepsilon}_{T_2}\rangle| \\ &\le |\langle \boldsymbol{\varepsilon}_{T_1}, \boldsymbol{f}_{T_2}\rangle| + |\langle \boldsymbol{\varepsilon}_{T_2}, \boldsymbol{f}_{T_1}\rangle| + |\langle \boldsymbol{\varepsilon}_{T_1}, \boldsymbol{\varepsilon}_{T_2}\rangle|\end{aligned}$$

The first term $|\langle \boldsymbol{\varepsilon}_{T_1}, \boldsymbol{f}_{T_2}\rangle|$ can be rewritten as

$$\begin{aligned}|\langle \boldsymbol{\varepsilon}_{T_1}, \boldsymbol{f}_{T_2}\rangle| &= \Big|\sum_{i=1}^{k_1} \langle \boldsymbol{\varepsilon}_{T_1}|_{[a_i,b_i]}, \boldsymbol{f}_{T_2}|_{[a_i,b_i]}\rangle\Big| \\ &= \Big|\sum_{i=1}^{k_1} \Big(\overbrace{\langle \boldsymbol{\varepsilon}_{T_1}|_{[a_1^i,b_1^i]}, f^*_{T_1^i}\rangle}^{=0} + \langle \boldsymbol{\varepsilon}_{T_1}|_{[a_1^i,b_1^i]}, \boldsymbol{f}_{T_2}|_{[a_1^i,b_1^i]} - f^*_{T_1^i}\rangle\Big)\Big| \\ &\le \sum_{i=1}^{k_1} \Big|\Big(\langle \boldsymbol{\varepsilon}_{T_1}|_{[a_1^i,b_1^i]}, \boldsymbol{f}_{T_2}|_{[a_1^i,b_1^i]} - f^*_{T_1^i}\rangle\Big)\Big| \\ &\le \sum_{i=1}^{k_1} \Big\|\boldsymbol{\varepsilon}_{T_1}|_{[a_1^i,b_1^i]}\Big\|_2 \Big\|\boldsymbol{f}_{T_2}|_{[a_1^i,b_1^i]} - f^*_{T_1^i}\Big\|_2 \\ &= \sum_{i=1}^{k_1} \|\varepsilon_{T_1^i}\|_2 \|\boldsymbol{f}_{T_2}|_{[a_1^i,b_1^i]} - f^*_{T_1^i}\|_2\end{aligned}$$

Similarly, we have:

$$|\langle \boldsymbol{\varepsilon}_{T_2}, \boldsymbol{f}_{T_1}\rangle| \le \sum_{i=1}^{k_2} \Big(\|\varepsilon_{T_2^i}\|_2 \times \|\boldsymbol{f}_{T_1}|_{[a_2^i,b_2^i]} - f^*_{T_2^i}\|_2\Big)$$

Recall that the computation of $|\langle \boldsymbol{\varepsilon}_{T_1}, \boldsymbol{\varepsilon}_{T_2}\rangle|$ is presented in Section 4.2.1. Combining the results of $|\langle \boldsymbol{\varepsilon}_{T_1}, \boldsymbol{\varepsilon}_{T_2}\rangle|$, $|\langle \boldsymbol{\varepsilon}_{T_1}, \boldsymbol{f}_{T_2}\rangle|$, and $|\langle \boldsymbol{f}_{T_1}, \boldsymbol{\varepsilon}_{T_2}\rangle|$ completes the proof. □

SELECT TSA('(Sum((t1.timeseries-Constant(μ(t1.timeseries)))×
t2.timeseries-Constant(μ(t2.timeseries))))
)/(σ(t1.timeseries)× σ(t2.timeseries))',
t1.timeseries, t2.timeseries)AS result
FROM HF t_1, HF t_2;

Figure 20: SQL query computing correlation TSA for all the time series pairs in HF.

H COMPUTATION OF FORMULA 5

Here we present how to compute $\|f_{T_2}|_{[a_1^i,b_1^i]} - f^*_{T_1^i}\|_2$ and $\|f_{T_1}|_{[a_2^i,b_2^i]} - f^*_{T_2^i}\|_2$ in $O(dim(\mathbb{F})^3)$.

Let's first look into $\|f_{T_2}|_{[a_1^i,b_1^i]} - f^*_{T_1^i}\|_2$.

$$\|f_{T_2}|_{[a_1^i,b_1^i]} - f^*_{T_1^i}\|_2 =$$

$$\Big(\sum_{j\in\Pi_{T_2,[a_1^i,b_1^i]}} \|f_{T_2}|_{[a_1^i,b_1^i]\cap[a_2^j,b_2^j]} - f^*_{T_1^i}|_{[a_1^i,b_1^i]\cap[a_2^j,b_2^j]}\|_2^2\Big)^{\frac{1}{2}}$$

$$= \Big(\sum_{j\in\Pi_{T_2,[a_i,b_i]}} \|\Psi([a_2^i,b_2^i],[a_1^i,b_1^i]\cap[a_2^j,b_2^j])f^*_{T_2^j}$$

$$- \Psi([a_1^i,b_1^i],[a_1^i,b_1^i]\cap[a_2^j,b_2^j])f^*_{T_1^i}\|_2^2\Big)^{\frac{1}{2}}$$

where Ψ is an orthonormal basis transformation matrix, which can be computed in $O(dim(\mathbb{F})^3)$. $\Psi([a_2^i,b_2^i],[a_1^i,b_1^i]\cap[a_2^j,b_2^j])$ transforms the orthonormal basis from the domain $[a_2^i,b_2^i]$ to the sub-domain $[a_1^i,b_1^i]\cap[a_2^j,b_2^j]$. In the following, we will show the details of computing Ψ.

Given a function family $\mathbb{F}$, let $(\varphi_i^{[a,b]})_{1\le i\le dim(\mathbb{F})}$ be an orthonormal basis of $\mathbb{F}$ on the domain $[a,b]$ for the scalar product $\langle f_1,f_2\rangle = \sum_{i=a}^{b}(f_1(i)\times f_2(i))$ where $f_1,f_2\in\mathbb{F}$. Such orthonormal basis can be obtained by using the Gram–Schmidt process [17]. Given a domain $[a,b]$ and one sub-domain $[a',b']\subset[a,b]$, let $\Psi([a,b],[a',b'])$ be the basis transform matrix such that

$$\Psi([a,b],[a',b'])_{i,j} = \langle\varphi_i^{[a,b]}|_{[a',b']},\varphi_j^{[a',b']}\rangle$$

That is using Ψ, we can directly obtain the orthonormal basis for any sub-domain. The size of Ψ is $dim(\mathbb{F})^2$ and the computation of each $\Psi([a,b],[a',b'])_{i,j}$ is $O(dim(\mathbb{F}))$. Therefore, the overall cost of computing Ψ is $O(dim(\mathbb{F})^3)$.

I EXPERIMENT SETTING DETAILS

Datasets. We evaluated all the error guarantee methods on four real-life datasets.

- Historical Forex Data (HF) are tick-by-tick market data for 15 Forex (foreign exchange) data pairs, e.g., AUD/JPY (Australian Dollar vs. Japanese Yen) from May 2009 to November 2016. Each Forex pair is considered a time series with ∼ 126 million data points (3 per second).
- Historical IoT Data (HI) were provided by Teradata and measure the internal oil pressure and the oil temperature every second from 8/19/2015 to 11/17/2015, as reported by seven engines in mining trucks in Chile.
- Historical Bitcoin Exchanges Data (HB)[18] contains 16 cryptocurrency exchange prices per minute from January 2012 to January 2018. Each cryptocurrency is considered as a time series.
- Historical Air Quality Data (HA)[19] present 11 different air quality measurements such as air pressure, air temperature and relative humidity from 09/10/2011 to 09/10/2014 in San Diego, at 1-minute resolution.

The HF and HB are financial market data, which are considered hard-to-model, while HI and HA are climate data following certain patterns. For example, the temperature in afternoon is usually higher than that at night, etc. Not surprisingly, the HB experiments behaved very similarly to the HF experiments, while the HA experiments behaved similarly to the HI ones.

Queries. The SQL query computing the correlation TSA for all the time series pairs in HF is shown in Figure 20. The SQL queries on the other three datasets are similar by change the table HF to HI, HB and HA respectively.

[18]https://www.kaggle.com/mczielinski/bitcoin-historical-data/data
[19]https://www.kaggle.com/ktochylin