# OneRoundSTL: In-Database Seasonal-Trend Decomposition

Zijie Chen
*Tsinghua University*
chenzj23@mails.tsinghua.edu.cn

Shaoxu Song*
*Tsinghua University*
sxsong@tsinghua.edu.cn

Jianmin Wang
*Tsinghua University*
jimwang@tsinghua.edu.cn

***Abstract*—Seasonal-trend decomposition has been widely used in time series analysis, e.g., time series forecasting and anomaly detection. Existing seasonal-trend decomposition methods, such as STL and its variations, assume that the time series is complete and sorted by timestamp. However, popular time series databases usually adopt LSM-Tree based storage, which stores data in pages not necessarily in time order. Moreover, time series stored in databases often suffer from missing values due to sensor failures, further compromising their integrity. A straightforward idea is to first merge and sort the data of different pages, and then decompose them. It obviously leads to heavy online computation, repeated calculations for multiple queries, and still cannot deal with the remaining missing data. In this paper, we propose OneRoundSTL, which pre-calculates offline some results in each individual page and concatenates the pre-calculated results online at query time to obtain the decomposition outcome. OneRoundSTL has been deployed and included as a function in an open source time series database, Apache IoTDB. Experiments on synthetic and real-world datasets in the system show that our OneRoundSTL exhibits high efficiency, far exceeding the state-of-the-art methods, while keeping decomposition effect.**

***Index Terms*—Time Series Database, STL, LSM-Tree**

## I. Introduction

Seasonal-trend decomposition (STD) plays a key role in analyzing time series data, increasingly sourced from the Internet of Things (IoT). It uncovers the trend patterns $\tau$ and seasonal changes $s$ of time series $x$, as illustrated in Figure 1. STD is found useful in applications like anomaly detection [1], [2], forecasting [3], [4], etc.

### A. Challenge

Existing decomposition methods such as STL [5] and its variations, OnlineSTL [6] and OneShotSTL [7], require time series to be collected at equal time intervals, not having missing values. Moreover, they need the data to be arranged in time order. This is obviously inconsistent with the real application scenarios. Specifically, time series data are often stored in dedicated databases, such as Apache IoTDB [8] and InfluxDB [9]. These LSM-Tree based time series databases store data in pages, i.e., the timestamps between different pages are not necessarily continuous with out-of-order arrivals [10]. Even worse, real-world data often suffer from missing values, e.g., due to sensor or network failures. For example, in Figure 1(a)(b), the time series $x_1, \ldots, x_{1728}$ is stored in two pages, where page 1 has $x_1, \ldots, x_{123}, x_{131}, \ldots, x_{855}, x_{858}, \ldots, x_{864}$ and page 2 stores $x_{856}, x_{857}, x_{865}, \ldots, x_{1728}$. In page 1, $x_{123}, \ldots, x_{131}$ are missing due to data collection failures. Moreover, $x_{856}, x_{857}$ are out-of-order arrivals, i.e., delayed and stored in page 2, resulting in non-contiguous timestamps between pages 1 and 2.

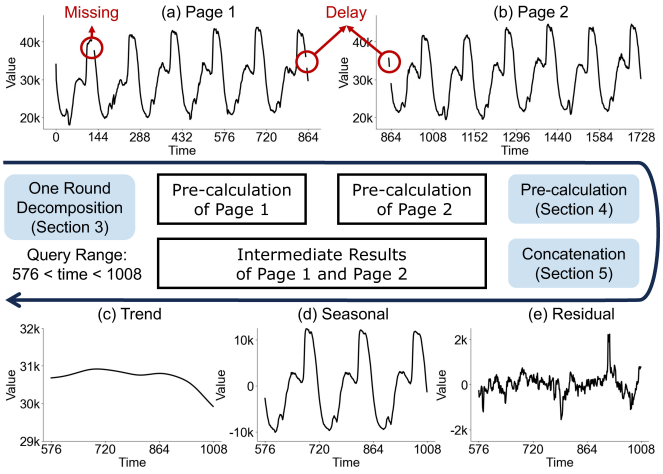


Fig. 1: OneRoundSTL for time series $x$ stored in two pages.

Obviously, existing decomposition methods cannot be directly applied in this scenario. A straightforward method of applying existing decomposition methods to time series databases is to first merge the pages and sort data by timestamps, and then decompose them from scratch. (1) Unfortunately, it requires both merging and decomposing to be performed entirely in the query phase, resulting in heavy online computation. (2) In addition, different decomposition tasks may be conducted on the same part of data, leading to repeated computation. (3) Furthermore, it is still unable to cope with the remaining missing values. Therefore, this simple adaption is neither efficient nor effective in practical use.

### B. Motivation

Note that almost all commodity time series databases employ LSM-tree based storage structures [11] for intensive writes, where the pages are immutable once flushed to disk [12] and updated by compaction [13]. A natural idea is thus to pre-calculate certain results for the immutable data in each page and use them to accelerate various online decomposition queries in the future. Unfortunately, existing decomposition

*Shaoxu Song (https://sxsong.github.io/) is the corresponding author.

2375-026X/25/$31.00 ©2025 IEEE
DOI 10.1109/ICDE65448.2025.00060

methods [5], [6], [7] need multiple iterations of the time series stored in different pages. The computation in a page relies on the results of other pages in the previous iterations, making the pre-calculation solely inside a page useless. It motivates us to devise a new one-round decomposition method that does not require the results of multiple iterations. Moreover, the pre-calculated results of different pages could be concatenated for various decomposition queries.

Owing to the existence of missing and delay issues, data in a page may be incomplete, as shown in Figure 1(a). To enable the pre-calculation of decomposition over incomplete data, we may utilize the values in the same phase of other periods. For instance, for the missing values $x_{123}, \ldots, x_{131}$, the decomposition pre-calculation could consider $x_{267}, \ldots, x_{275}$ in the next period. Of course, some incomplete data such as $x_{856}, x_{857}$ may be stored as delays in other pages, and not accessed in the pre-calculation of the current page 1. Such pre-calculated results thus need re-calculation, when concatenating the results of pages 1 and 2 during the decomposition query processing.

### C. Contribution

In this paper, we devise OneRoundSTL, a seasonal-trend decomposition method tightly integrated with time series database storage. To the best of our knowledge, this is the first work that pre-computes seasonal-trend decomposition results in time series databases. Our main contributions are:

1) We propose the iteration-free OneRoundSTL in Section II to enable pre-calculation. The decomposition problem is transformed to a linear system solving problem, thereby requiring only one round of iteration. Furthermore, we propose pruned $LDL^T$ decomposition to accelerate the linear system solving. Proposition 3 shows that the pruned $LDL^T$ decomposition converges to the standard $LDL^T$ decomposition.
2) We introduce the pre-calculation on each page for OneRoundSTL in Section III. To address the issue of missing values, we propose to explore the seasonal features tailored for different time series positions in Section III-C. We use historical data to handle the pre-calculation for missing values, thereby mitigating their impact on the pre-calculation process. Proposition 4 establishes an error bound for the decomposition results with our handling method.
3) We concatenate the pre-calculated results in multiple pages for OneRoundSTL in Section IV. Proposition 5 indicates that the concatenation results converge to the pre-calculated results, thus significantly enhances the efficiency of concatenation. For different timestamp ranges of two pages, i.e., adjacent, disjoint, and overlapped, we detail specific steps respectively.
4) Remarkably, OneRoundSTL has been realized in a time series database system Apache IoTDB, with details presented in Section V. The document is available on the product website [14] and the code is included in the system repository [15]. Re-calculation on truncated pages is performed for a specific query. Proposition 6 shows that the re-calculated results converge to the pre-calculated results, ensuring consistency of decomposition across different page quantities.

TABLE I: Frequently used notations

| Symbol | Description |
|---|---|
| $x_i$ | data point of time series $x$ at time $i$ |
| $x_i^{(m)}$ | data point of time series $x$ in page $m$ at time $i$ |
| $\tau_i$ | trend component of $x_i$ |
| $s_i$ | seasonal component of $x_i$ |
| $r_i$ | residual component of $x_i$ |
| $v_i$ | $i$-th data point of baseline seasonal component $v$ |
| $z_i$ | $i$-th data point of intermediate results $z$ |
| $a_{ij}$ | $(i, j)$ element of matrix $A$ |
| $b_i$ | $i$-th element of vector $b$ |
| $N$ | time series length |
| $M$ | period length |
| $\alpha^{(m)}$ | starting timestamps in page $m$ |
| $\beta^{(m)}$ | ending timestamps in page $m$ |
| $\Delta$ | first order difference operator |
| $\Delta^2$ | second order difference operator |
| $\Delta_D$ | second order diagonal difference operator |
| $\Delta_T$ | seasonal difference operator |

5) We conduct extensive experimental evaluation in Section VI. Our proposed OneRoundSTL approach achieves much higher efficiency than the state-of-the-art decomposition methods, while keeping the decomposition effect. The experiment code and public data are available in [16] for reproducibility.

Table I lists the frequently used notations.

## II. One Round Decomposition

Existing decomposition methods [17], [18], [7] require multiple scans of time series, which hinders the application of pre-calculation. In this section, we design an iteration-free decomposition method named OneRoundSTL. Section II-A introduces the general seasonal-trend decomposition. Section II-B defines and analyzes the OneRoundSTL problem. Section II-C gives the detailed steps of the decomposition.

### A. Seasonal-Trend Decomposition

For a time series of $N$ observations, $x = [x_1, x_2, \ldots, x_N]^T$, where $x_i$ is the value at time $i$, we can decompose $x$ into

$$x_i = \tau_i + s_i + r_i, i = 1, 2, \ldots, N, \tag{1}$$

where $\tau_i$, $s_i$, $r_i$ denote the trend, seasonal and residual components of $x_i$, respectively. In seasonal-trend decomposition, $\tau_i$ describes the long-term increasing or decreasing trend; $s_i$ describes the cyclic features with a period $M$, which can be estimated by period detection methods [19], [20]; $r_i$ describes the short-term random fluctuations. Figure 1 shows an example of decomposition, where Figures 1 (c), (d), and (e) represent the trend components $\tau_i$, seasonal components $s_i$, and residual components $r_i$, respectively. The decomposition results $\tau$ and $s$ are calculated as $y$ in Formula 16 below by OneRoundSTL, where $y = [\tau_1, s_1, ..., \tau_N, s_N]^T$.

### B. Problem Definition

We first define the challenges of missing data and delays.

**Definition 1** (Missing issues)**.** *Given a time series $x = [x_1, x_2, \ldots, x_N]^T$, there exists $i \in [1, N]$ such that $x_i$ is $NaN$, indicating the absence of a value at time $i$.*

**Definition 2** (Delay issues)**.** *The timestamp ranges of time series across database pages overlap. Specifically, let $x_i^{(m)}$ and $x_j^{(n)}$ denote data points of time series $x$ on pages $m$ and $n$ at times $i$ and $j$, respectively. There exists $i < j$, $i, j \in [1, N]$, with $m > n$.*

In this context, the one-round decomposition problem is inspired by the following three characteristics of time series [21]: (1) Low-noise, where the random fluctuations $(r_i)^2$ are minimized; (2) Seasonality, where the differences in the seasonal components $(\Delta_T s_i)^2$ at the same phase are smaller; and (3) Smoothness, where the changes in the trend $(\Delta^2 \tau_i)^2$ are smoother.

**Definition 3.** *Given a time series $x \in \mathbb{R}^N$, the input seasonal period parameter $M$, the input smoothness parameter $\lambda$ and baseline seasonal component $v = [v_1, v_2 \ldots, v_M]^T$, we define the OneRoundSTL results $\tau_i$ and $s_i$ of $x_i$ as the solution to the following optimization problem.*

$$\begin{aligned} \min_{\tau, s} \quad & \sum_{i=1}^{N} r_i^2 + \sum_{i=1}^{N} (\Delta_T s_i)^2 + \lambda \sum_{i=2}^{N-1} (\Delta^2 \tau_i)^2 \\ \textit{s.t.} \quad & r_i = x_i - \tau_i - s_i \\ & \Delta_T s_i = s_i - v_{i \bmod M} \\ & \Delta^2 \tau_i = \Delta(\Delta \tau_i) = \tau_{i+1} - 2\tau_i + \tau_{i-1} \end{aligned} \tag{2}$$

The baseline seasonal component $v_i$ can be automatically set during the cold start phase as shown in Section II-C2. $\lambda$ is a hyper-parameter to control the smoothness. Indeed, OneRoundSTL can be seen as an extension of the Hodrick-Prescott filter [22], a simple but effective smoothing method that can be solved with a single pass of the data. It motivates us to solve OneRoundSTL by only one round of data scan.

*1) Matrix Reformulation:* The OneRoundSTL problem in Formula 2 can be solved using maximum likelihood estimation [23]. Specifically, Formula 2 is reformulated as a linear system $Ay = b$, where $y$ is a combination of $\tau$ and $s$, as $y = [\tau_1, s_1, ..., \tau_N, s_N]^T$. And the specific forms of parameter $A$ and $b$ are determined as follows. The detailed reformulation process can be found in the supplementary [24].

$$\begin{aligned} A &= A_1^T A_1 + A_2^T A_2 + \lambda A_3^T A_3 \in \mathbb{R}^{2N \times 2N}, \\ b &= A_1^T x + A_2^T u \in \mathbb{R}^{2N}, \\ u &= [v^T, \ldots v^T]^T \in \mathbb{R}^N, \\ A_1 &= \begin{bmatrix} 1 & 1 & & & \\ & & \ddots & & \\ & & & 1 & 1 \end{bmatrix} \in \mathbb{R}^{N \times 2N}, \\ A_2 &= \begin{bmatrix} 0 & 1 & & & \\ & & \ddots & & \\ & & & 0 & 1 \end{bmatrix} \in \mathbb{R}^{N \times 2N}, \\ A_3 &= \begin{bmatrix} 1 & 0 & -2 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & 1 & 0 & -2 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{(N-2) \times 2N}. \end{aligned}$$

*2) Properties of Matrix $A$ and Vector $b$:* Prior to solving the linear system $Ay = b$, we analyze the properties of $A$ and $b$ to accelerate the process. We first focus on matrix $A$. For clarity, we initially present the elements of $A$ as follows:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & \cdots & a_{1,2N} \\ a_{21} & a_{22} & a_{23} & a_{24} & \cdots & a_{2,2N} \\ a_{31} & a_{32} & a_{33} & a_{34} & \cdots & a_{3,2N} \\ a_{41} & a_{42} & a_{43} & a_{44} & \cdots & a_{4,2N} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{2N,1} & a_{2N,2} & a_{2N,3} & a_{2N,4} & \cdots & a_{2N,2N} \end{bmatrix} \in \mathbb{R}^{2N \times 2N}$$

**Proposition 1** (Sparsity of $A$)**.** *$A$ is a band matrix with a bandwidth of 4, i.e.,*

$$a_{ij} = 0, \quad |i - j| > 4, \quad i, j \in [1, 2N].$$

Proposition 1 infers that each row contains no more than nine non-zero elements. This property can reduce the cost of scanning A from O($n^2$) to O($n$), as shown in Algorithm 1.

Furthermore, since the process of extracting seasonal and trend component for each data point is identical, $A$ exhibits a certain level of repetition. For brevity, we define operator $\Delta_D$. For any element $a_{ij}$ of matrix A, $\Delta_D a_{ij} = a_{ij} - a_{i-2,j-2}$.

**Proposition 2** (Repetition of $A$)**.** *Except for the first and last 4 rows, the elements in the remaining rows exhibit a pattern of alternating repetition, i.e.,*

$$\Delta_D a_{ij} = 0, \quad |i - j| \leq 4, \quad i, j \in [7, 2N - 4].$$

Finally, we briefly discuss the construction of vector $b$. It is composed of baseline seasonal component $v$ and original data $x$. Specifically, for $b = [b_1, b_2, \cdots, b_{2N}]$, we have:

$$\begin{aligned} b_{2i-1} &= x_i \quad 1 \leq i \leq N, \\ b_{2i} &= x_i + v_{i \bmod M} \quad 1 \leq i \leq N. \end{aligned} \tag{3}$$

Thus, we can directly construct $b$ without matrix operations.

### C. Problem Solving

We give the detailed solving steps of the linear system $Ay = b$ below. Note that, at this stage, we do not consider missing and delay issues. As a preparation, we perform the $LDL^T$ decomposition of the matrix $A$ in Section II-C1 and construct $b$ in the cold start phase in Section II-C2. The solution contains a forward substitution in Section II-C3 and a backward substitution in Section II-C4. Note that we obtain the results through only one round of bidirectional scanning, i.e., iteration-free.

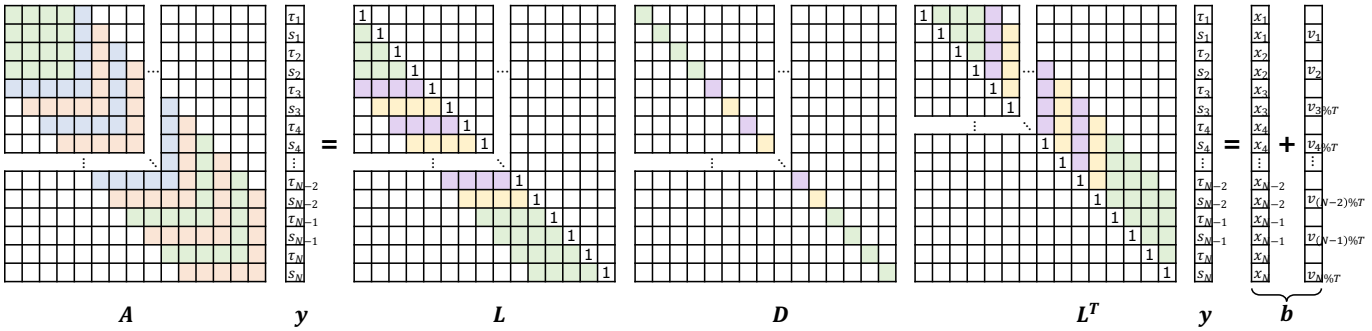


Fig. 2: $LDL^T$ decomposition, where white cells represent 0; the blue and red bands in $A$ are exactly the same; the purple and yellow bands in $L$ and $D$ are similar; green cells represents the independent data of the first and last 4 rows.

*1) $LDL^T$ Decomposition:* Figure 2 shows the process of solving linear system $Ay = b$ using $LDL^T$ decomposition. Specifically, it decomposes matrix $A$ into a lower triangular matrix $L$ and a diagonal matrix $D$, and guarantees that

$$A = LDL^T.$$

After that, we can decompose $Ay = b$ into two linear equations: $Lz = b$ and $DL^Ty = z$.

For clarity, we initially present the elements of decomposition results $L$ and $D$. According to the definition of $LDL^T$ decomposition, $L$ is a lower triangular matrix presented as:

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ l_{21} & 1 & 0 & 0 & \cdots & 0 \\ l_{31} & l_{32} & 1 & 0 & \cdots & 0 \\ l_{41} & l_{42} & l_{43} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{2N,1} & l_{2N,2} & l_{2N,3} & l_{2N,4} & \cdots & 1 \end{bmatrix} \in \mathbb{R}^{2N\times 2N}$$

and $D$ is a diagonal matrix presented as:

$$D = \begin{bmatrix} d_{11} & 0 & 0 & 0 & \cdots & 0 \\ 0 & d_{22} & 0 & 0 & \cdots & 0 \\ 0 & 0 & d_{33} & 0 & \cdots & 0 \\ 0 & 0 & 0 & d_{44} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & d_{2N,2N} \end{bmatrix} \in \mathbb{R}^{2N\times 2N}$$

Due to the properties of $A$ discussed in Section II-B2, $L$ and $D$ also share some properties that can be leveraged to simplify the decomposition process. We first discuss the sparsity of $L$, which is a consequence of the sparsity of $A$ in Proposition 1.

**Corollary 1** (Sparsity of $L$)**.** *$L$ is a lower band matrix with bandwidth 4, i.e.,*

$$l_{ij} = 0, \quad i - j > 4, \quad i, j \in [1, 2N].$$

Unlike $A$, both $L$ and $D$ do not exhibit repetition due to the inconsistency between the initial 4 rows and the subsequent rows of $A$. However, since the subsequent rows of $A$ still present repetition pattern as introduced in Proposition 1, elements of $L$ and $D$ shows a tendency to be equal, i.e., the elements within $L$ and $D$ exhibit convergence.

**Algorithm 1:** Pruned $LDL^T$ Decomposition

**Input:** $A \in \mathbb{R}^{2N\times 2N}$
**Output:** $L \in \mathbb{R}^{2N\times 2N}$, $D \in \mathbb{R}^{2N\times 2N}$

$L = zeros(2N, 2N); D = zeros(2N, 2N)$;
$converge = False$;
**for** $(i = 1; i \le 2N; i++)$ **do**
  $l_{ii} = 1$;
  **if** $converge$ *and* $i \le 2N - 4$ **then**
    $d_{ii} = d_{i-2,i-2}$;
    **for** $(j = i + 1; j \le i + 4; j++)$ **do**
      $l_{ji} = l_{j-2,i-2}$;
  **else**
    $d_{ii} = a_{ii} - \sum_{k=max(1,i-4)}^{i-1} d_{kk} \cdot l_{ik}^2$;
    **for** $(j = i + 1; j \le min(2N, i + 4); j++)$ **do**
      $l_{ji} = (a_{ji} - \sum_{k=max(1,j-4)}^{i-1} l_{jk} \cdot d_{kk} \cdot l_{ik})/d_{ii}$;
    **if** $\sum_{h=i-1}^{i} |\Delta_D d_{hh}| + \sum_{j=h-4}^{h-1} |\Delta_D l_{hj}| < \epsilon$ **then**
      $converge = True$;
**return** $L, D$;

**Proposition 3** (Convergence of $L$ and $D$)**.** *$\{l_{i,j}, l_{i+2,j+2} \cdots\}$ and $\{d_{i,i}, d_{i+2,i+2} \cdots\}$ converge, i.e., for any positive number $\epsilon$, there exist $K \in \mathcal{N}$ such that for all $i > K$ having*

$$|l_{i+2,j+2} - l_{i,j}| < \epsilon \quad \textit{and} \quad |d_{i+2,i+2} - d_{i,i}| < \epsilon$$

Inspired by Corollary 1 and Proposition 3, we present a novel formulation for the pruned $LDL^T$ decomposition, which significantly improves computational efficiency compared to the traditional $LDL^T$ decomposition. Specifically, for $l_{ji}$ of matrix $L$ and $d_{ii}$ of matrix $D$, they can be obtained by:

$$l_{ji} = (a_{ji} - \sum_{k=max(1,j-4)}^{i-1} l_{jk} \cdot d_{kk} \cdot l_{ik})/d_{ii} \tag{4}$$

$$d_{ii} = a_{ii} - \sum_{k=max(1,i-4)}^{i-1} d_{kk} \cdot l_{ik}^2. \tag{5}$$

When the absolute difference between corresponding elements across two rows is less the user-defined threshold $\epsilon$, i.e., $\sum_{h=i-1}^{i}(|\Delta_D d_{hh}| + \sum_{j=h-4}^{h-1}|\Delta_D l_{hj}|) < \epsilon$, we consider L and D have converged and can directly extend the results as:

$$l_{ji} = l_{j-2,i-2} \tag{6}$$

$$d_{ii} = d_{i-2,i-2} \tag{7}$$

Algorithm 1 presents the details of pruned decomposition method. Lines 12 and 10 outline the computation of $L$ and $D$ before convergence, corresponding to Formulas 4 and 5. Similarly, Lines 8 and 6 outline the computation of $L$ and $D$ after convergence, corresponding to Formulas 6 and 7.

As for time complexity, each $l$ and $d$ require at most 4 calculations before convergence. Calculation terminates once converged, where $\phi$ is the minimum number of rows that converge. For each row, $L$ involves 4 off-diagonal computations, as its bandwidth is 5 excluding the diagonal, while $D$ requires only 1 diagonal computation. This results in 5 computations per row, leading to a total time complexity of O($4\times5\phi$). Notably, when $\lambda = 1$, $\phi = 38$, which is significantly smaller than the scale of million-level datasets. Therefore, the algorithm exhibits low computational complexity.

As for space complexity, $L$ and $D$ are stored in system files, enabling decomposition computation for any time series. Considering Corollary 1, both $L$ and $D$ are sparse, and the non-zero data locations are fixed. Therefore, the value 0 does not need to consume space storage. The space complexity of $L$ and $D$ can thus be reduced from O($N^2$) to O($N$). Similarly, considering Proposition 3, the calculation of $L$ and $D$ can converge. After convergence, its values remain unchanged. Therefore, there is no need to store the unchanged values. The space complexity of $L$ can thus be further reduced from O($N$) to O($\phi$), and $D$ from O($N$) to O($\phi$). Note that the space complexity of $L$ and $D$ is independent of the time series length $N$ at this time.

*2) Cold Start:* As indicated in Section II-B, $b$ in the equation $Lz = b$ requires baseline seasonal components $v$. To obtain $v$, we first retain the first $n \times M$ cycles of original data as the cold start phase. Specifically, we use the existing batch seasonal-trend decomposition [21], [18], [17], resulting in the initial seasonal component $\widetilde{s} = s_1, s_2, \ldots, s_{n\times M}$. Subsequently, we derive $v$ by calculating the average of $\widetilde{s}$ with the same phase as:

$$v_i = \frac{1}{n}\sum_{k=0}^{n-1} s_{k\times M+i}. \tag{8}$$

Although we have obtained the decomposition results of the first $n$ periods in the cold start phase, for consistency, OneRoundSTL still starts from the first point and re-calculates.

*Handling Missing Values in Cold Start:* If missing values occur in the cold start phase, we lack any prior knowledge for handling. Furthermore, the existing seasonal-trend decomposition algorithm is unable to handle missing values. Consequently, we resort to the existing methods [25], [26].

**Algorithm 2:** OneRoundSTL

**Input:** $x \in \mathbb{R}^N$, $T$, $\lambda$
**Output:** $\tau \in \mathbb{R}^N$, $s \in \mathbb{R}^N$, $r \in \mathbb{R}^N$

```
calculate L, D ∈ R^{2N×2N} by Algorithm 1;
calculate v ∈ R^M by Formula 8;
calculate z ∈ R^{2N} by Formula 9;
calculate y ∈ R^{2N} by Formula 10;
for (i = 1; i ≤ N; i++) do
    τ_i = y_{2i-1};
    s_i = y_{2i};
    r_i = x_i − τ_i − s_i;
return τ, s, r;
```

*3) Forward Substitution:* Forward substitution is the process of solving the equation $Lz = b$. As shown in Section II-C1, $L$ represents a lower triangular coefficient matrix. Therefore, each $z_i$ can be solved at a time, starting from any chosen row $j$:

$$z_i = \mathcal{F}(i, j, L, b, z) = b_i - \sum_{k=\max(i-3,j)}^{i} l_{i-j,k-j} \cdot z_k \tag{9}$$

for $i = j, \ldots, 2N$, where $i$ denotes the current index and $j$ denotes the starting index of the forward substitution.

*4) Backward Substitution:* Backward substitution is the process of solving the equation $DL^T y = z$. As shown in Section II-C1, $DL^T$ represents an upper triangular coefficient matrix. Therefore, each $y_i$ can be solved at a time, starting from any chosen row $j$:

$$y_i = \mathcal{B}(i, j, L, d, z, y) = \frac{z_i}{d_i} - \sum_{k=i}^{\min(i+3,j)} l_{k-j+2N,i-j+2N} \cdot y_k \tag{10}$$

for $i = 1, \ldots, j$, where $i$ denotes the current index and $j$ denotes the starting index of the backward substitution.

## III. Pre-calculation on Each Page

In this section, we introduce the pre-calculation of OneRoundSTL on each page in LSM-Tree based time series databases. At this stage, we consider the presence of missing and delay issues in the database. Specifically, Section III-A introduces the pages in the database. Section III-B introduces the specific steps of pre-calculation. Section III-C supplements the STD-based missing value handling method.

### A. Pages in LSM-Tree

The Log-Structured Merge-Tree (LSM-Tree) [11] is a popular data storage structure in databases, particularly for facilitating high-throughput writing operations. Therefore, LSM-Tree is commonly incorporated in time series databases designed for IoT scenarios, such as Apache IoTDB [10].

Pages are the fundamental units of storage in LSM-Tree. Each page typically contains a subset of IoT data. Pages help in organizing data, enabling efficient access and retrieval.

When a read operation is performed, the relevant pages are accessed, minimizing the amount of data read from disk. Therefore, we first define the data contained in each page.

**Definition 4.** *The $m$-th page contains a subset of the original data $x$, with timestamp starting from $\alpha^{(m)}$ to $\beta^{(m)}$, i.e., $x_t$ in page $m$, where $\alpha^{(m)} \leq t \leq \beta^{(m)}$.*

### B. Pre-calculation without Missing Values

Since pages in the database are stored in different locations, it is necessary to pre-calculate the data on each page independently. As the calculation deals with only single pages, for simplicity, in this section, we abbreviate the notation in Definition 4 to $\alpha$ and $\beta$. By first constructing $b$ with Formula 3. the pre-calculation can be considered a forward substitution process starting from the starting timestamp $\alpha$, i.e.,

$$z_i = \mathcal{F}(i, \alpha, L, b, z) \tag{11}$$

for $i = \alpha, \ldots, \beta$.

### C. Pre-calculation with Missing Values

For different positions, we adopt different missing values handling methods, including head missing values handling in Section III-C1 and subsequent missing values handling in Section III-C2. We present the complete algorithm and analysis of pre-calculation in the appendix [24].

*1) Handling Head Missing Values:* We first define the range of head values of each page.

**Definition 5.** *The head values of time series $x$ are $x_i$ for $0 < i < \phi$, where $\phi$ is the minimum number of rows that the pruned $LDL^T$ decomposition converges.*

For the head values, we employ existing methods for handling missing values. There are two main reasons: (1) the data on each page is stored in different locations on the hard disk, and thus the head data of each page has less historical information to use; (2) considering Proposition 3, if $LDL^T$ decomposition does not converge, i.e., $0 \leq i \leq \phi$, non-zero values of $L$ and $D$ differ in each row. According to Formulas 4 and 5, the values of the last four rows are derived from the preceding rows, hence the last four rows change with different $i$. In contrast, after convergence, the non-zero values in each row of $L$ and $D$ cease changing, which facilitates the partial decomposition we proposed in Section III-C2. Therefore, we select the range of head values from 1 to $\phi$.

*2) Handling Subsequent Missing Values:* For the data after the head values, we propose another missing value handling strategy. Let $x_i$ be the missing value. At this stage, $z_{2i-1}$ and $z_{2i}$ cannot be derived using Formula 11, hindering the subsequent pre-computation that relies on $z_{2i-1}$ and $z_{2i}$. Inspired by Formula 1 and the stability of time series, we utilize the baseline seasonal component $v_{i \bmod M}$ and $\tau_{i-1}$ to represent $z_{2i-1}$ and $z_{2i}$. Notably, $\tau_{i-1}$ (i.e., $y_{2i-3}$) can be recalculated by using Formula 9 with $x_{i-1}$ as the endpoint, and then decomposing with Formula 10. The specific handling process is as follows:

$$\begin{aligned}
&z_j = \mathcal{F}(j, i-N, L, b, z), \quad 2i-5 \leq j \leq 2i-2 \\
&y_j = \mathcal{B}(j, i-1, L, d, z, y), \quad 2i-3 \leq j \leq 2i-2 \\
&z_{2i-1} = y_{2i-3} + v_{i \bmod M} - \sum_{k=\max(i-3,j)}^{i} l_{i-j,k-j} \cdot z_k \\
&z_{2i} = y_{2i-3} + 2v_{i \bmod M} - \sum_{k=\max(i-3,j)}^{i} l_{i-j,k-j} \cdot z_k
\end{aligned} \tag{12}$$

Hence, we obtain the pre-calculated results $z_{2i-1}$ and $z_{2i}$ for the missing value $x_i$, enabling subsequent pre-calculations to be carried out at $z_j$ for $j > 2i$. Proposition 4 further provides an error bound for $z_{2i-1}$ and $z_{2i}$ obtained using this method.

**Proposition 4** (Error Bound of Missing Value Handling)**.** *For the ground truth of $x_i$, $z_{2i-1}$ and $z_{2i}$, i.e., $\widetilde{x}_i$, $\widetilde{z}_{2i-1}$ and $\widetilde{z}_{2i}$, we have $|\widetilde{z}_{2i-1} - z_{2i}| < |y_{2i-3} + v_{i \bmod M} - \widetilde{x}_i|$ with $z_j = \mathcal{F}(j, i-N, L, b, z)$ for $2i-5 \leq j \leq 2i-2$ and $y_j = \mathcal{B}(j, i-1, L, d, z, y)$ for $2i-3 \leq j \leq 2i-2$, $z_{2i-1} = y_{2i-3} + v_{i \bmod M} - \sum_{k=\max(i-3,j)}^{i} l_{i-j,k-j} \cdot z_k$, $z_{2i} = y_{2i-3} + 2v_{i \bmod M} - \sum_{k=\max(i-3,j)}^{i} l_{i-j,k-j} \cdot z_k$.*

We provide a flowchart along with step-by-step breakdowns of the pre-calculation process to facilitate the reader's understanding. Specifically, the orange section on the left side is completed during database creation, while the blue section on the right is executed when the time series data is input.

Step 1: Problem Definition. Section II-B formulates the mathematical expression of the decomposition problem.

Step 2: Matrix Reformulation. Section II-B1 reformulates the decomposition problem into an equivalent matrix form, i.e., solving the linear equation $Ay = b$, and provides the precise definition of $A$.

Step 3: Pruned $LDL^T$ Decomposition. Section II-C1 decomposes the parameter matrix A into an upper triangular matrix L and a diagonal matrix D, yielding $A = LDL^T$. This splits the equation $Ay = b$ into two parts, $Lz = b$ and $DL^Ty = z$.

Step 4: Cold Start. Section II-C2 performs a cold start, constructing the vector $b$ based on the input time series $x$. However, at this stage, missing values in $b$ remain unhandled.

Step 5: Pre-calculation with missing values handling. Section III presents the pre-calculation process, where $Lz = b$ is solved to obtain the intermediate result $z$. Section III-C then addresses missing values in $b$ by leveraging past pre-calculation results.

**Example 1.** *Figure 5 shows the pre-calculation results of OneRoundSTL. Note that on Page 3, $x_{3n-3}$ is missing. By employing the missing value handling method, we are able to acquire the pre-calculated results $z_{6n-3}$ and $z_{6n-2}$ corresponding to $x_{3n-3}$ with $z_j = \mathcal{F}(j, 3n-3-N, L, b, z)$ for $6n-11 \leq j \leq 6n-8$, $y_j = \mathcal{B}(j, 3n-4, L, d, z, y)$ for $6n-9 \leq j \leq 6n-8$, $z_{2i-1} = y_{2i-3} + v_{i \bmod M} - \sum_{k=\max(i-3,j)}^{i} l_{i-j,k-j} \cdot z_k$, $z_{2i} = y_{2i-3} + 2v_{i \bmod M} - \sum_{k=\max(i-3,j)}^{i} l_{i-j,k-j} \cdot z_k$.*

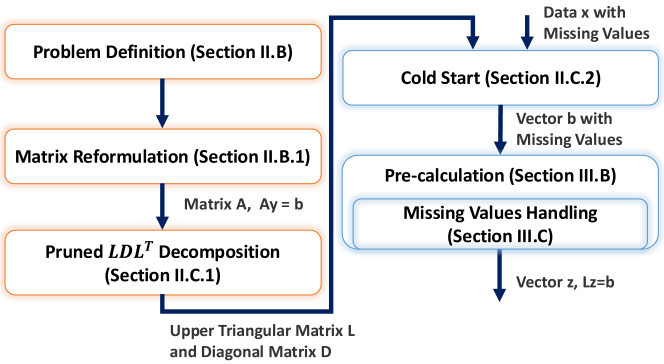


Fig. 3: Flowchart of Pre-calculation

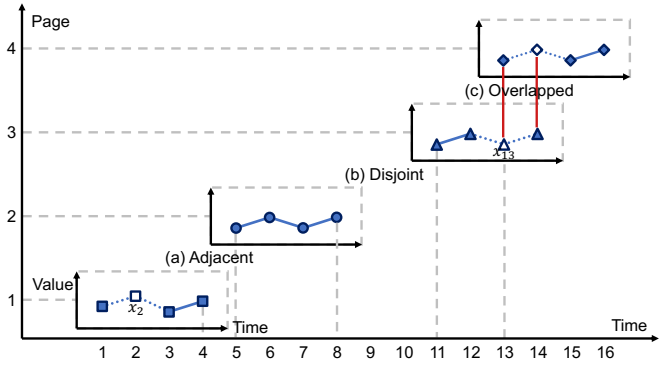


Fig. 4: Pages (with different markers) in LSM-Tree based storage for different parts of a time series. Hollow markers represent missing values, some of which ($x_{13}$) may be filled in the subsequent pages owing to out-of-order arrivals.

## IV. Concatenation of Multiple Pages

In this section, we discuss the operation of concatenation on two pages. It naturally applies to multiple pages as well. We can order all pages based on their start time and then concatenate pages pairwise. We first provide a formal definition of concatenation on two pages.

**Definition 6.** *The concatenation of two pages combines the pre-calculated result $z_j$ for $\alpha^{(m-1)} \le j \le \beta^{(m-1)}$ in the $(m-1)$-th page with $z_k$ for $\alpha^{(m)} \le k \le \beta^{(m)}$ in the $m$-th page, and outputs $z_i$ for $\alpha^{(m-1)} \le i \le \beta^{(m)}$ that is the same as the results calculated directly in the same range.*

As discussed in Section III-B, pre-calculation is performed independently on each page. The concatenation extends the pre-calculated range from a single page to two pages, necessitating the re-calculation of results. We denote the re-calculated results of $z_i$ as $\bar{z}_i$. Referring to Definition 6, we have

$$\bar{z}_i = \mathcal{F}(i, \alpha^{(m-1)}, L, b, z) \tag{13}$$

We first introduce the three types of timestamp range cases for two pages in an LSM-Tree based database in Section IV-A. Different timestamp range cases require different re-calculation process. Correspondingly, we introduce the concatenation method of adjacent pages in Section IV-B. Then, we introduce the concatenation method of disjoint pages in Section IV-C. Finally, we introduce the concatenation method of overlapped pages in Section IV-D.

### A. Page Relationship in LSM-Tree

Figure 4 presents a concise depiction of the storage organization utilized by LSM-Tree. Each page stores a segment of the time series $x$. A larger page number means that its data are newer. Owing to the existence of missing and out-of-order arrivals, two pages can exhibit distinct relationships on timestamps: (a) Adjacent pages, whose timestamps are consecutive, as shown by Pages 1 and 2 in Figure 4. (b) Disjoint pages, whose timestamps have a gap between them, e.g., due to missing or delayed values, as shown by Pages 2 and 3 in Figure 4. (c) Overlapped pages, where the delayed values such as $x_{13}$ in the page having a larger page number fill the incomplete data in the previous page with a smaller number, as shown by Pages 3 and 4 in Figure 4. To assemble the time series $x$, these pages with various relationships should be concatenated, as presented in Section IV.

### B. Concatenating Adjacent Pages

The $(m-1)$-th page and the $m$-th page are adjacent pages if and only if $\beta^{(m-1)} + 1 = \alpha^{(m)}$. Since the two adjacent pages have continuous timestamps, it is only required to re-calculate from the connection point, i.e., $i = \alpha^{(m)}$. Note that it does not require re-calculating the entire $m$-th page due to the convergence property introduced below.

**Proposition 5** (Convergence of Re-calculation in Concatenation)**.** *The re-calculated results $\bar{z}_i = \mathcal{F}(i, \alpha^{(m-1)}, L, b, z)$ converges to $z_i$, i.e., for any positive number $\zeta$, there exists $K \in \mathcal{N}, K \ge 2\alpha^{(m)} - 1$, such that for all $i > K$ having*

$$|\bar{z}_i - z_i| < \zeta.$$

Based on Proposition 5, we can improve the efficiency of re-calculation. Specifically, with a given convergence threshold $\zeta$, we obtain the re-calculated results $\bar{z}_i$ by Formula 13 starting from $i = \alpha^{(m)}$. Then, we concatenate the pages with $z_i = \bar{z}_i$. Once converged, i.e., $|\bar{z}_i - z_i| < \zeta$, we resort to stored pre-calculated results $z_i$ instead of further re-calculating $\bar{z}_i$.

**Example 2.** *Figure 5(a) shows the concatenating process of adjacent pages. Page 1 stores $x_1^{(1)}, x_2^{(1)}, \cdots, x_n^{(1)}$. Page 2 stores $x_{n+1}^{(2)}, x_{n+2}^{(2)}, \cdots, x_{2n}^{(2)}$. Referring to Definition 6, $\beta^{(1)} = n$, $\alpha^{(2)} = n+1$. Since $\alpha^{(2)} = \beta^{(1)} + 1$, Page 1 and 2 are adjacent. Considering $2\alpha^{(2)} - 1 = 2n+1$, the concatenation re-calculates from $z_{2n+1}$ until convergence.*

### C. Concatenating Disjoint Pages

The $(m-1)$-th page and the $m$-th page are disjoint pages if and only if $\beta^{(m-1)} + 1 < \alpha^{(m)}$. Since both $x$ and $z$ are missing between the two pages, i.e., $x_i$ for $\beta^{(m-1)} < i < \alpha^{(m)}$ and $z_i$ for $2\beta^{(m-1)} < i < 2\alpha^{(m)} - 1$ are missing, we need to first handle these missing $x_i$ with Formula 12 and

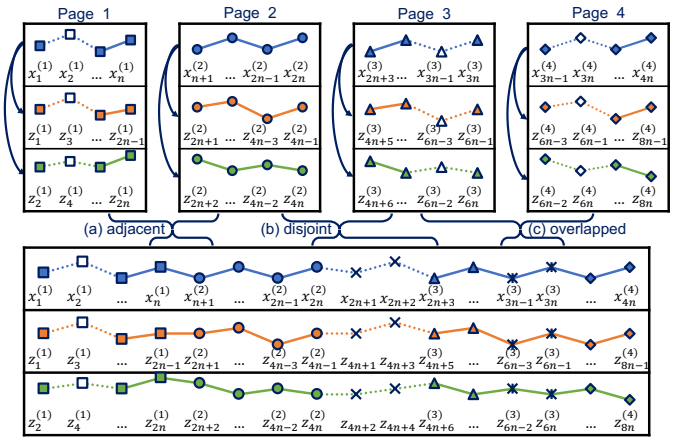


Fig. 5: Concatenation of the pre-calculated results for pages with various relationships.

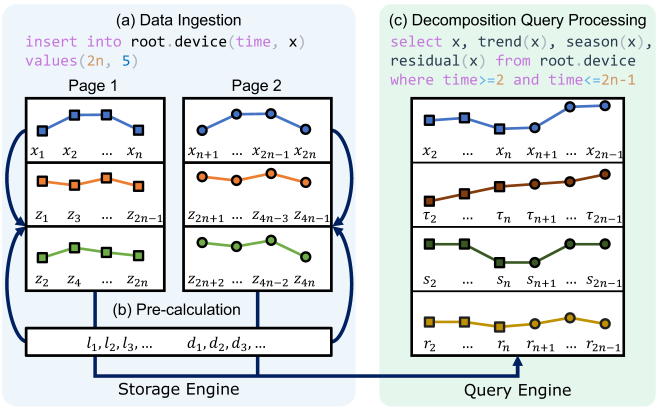


Fig. 6: Deployment of OneRoundSTL in Apache IoTDB.

calculate $z_i$ with Formula 9. Then, referring to Proposition 5, we concatenate the pages by re-calculating $z_i$ from $i = 2\alpha^{(m)} - 1$ until convergence.

**Example 3.** *Figure 5(b) shows the concatenating process of disjoint pages. Page 2 stores $x^{(2)}_{n+1}, x^{(2)}_{n+2}, \cdots, x^{(2)}_{2n}$, while Page 3 stores $x^{(3)}_{2n+3}, x^{(3)}_{2n+4}, \cdots, x^{(3)}_{3n}$. Referring to Definition 6, we have $\beta^{(2)} = 2n$, $\alpha^{(3)} = 2n + 3$. Since $\alpha^{(3)} = \beta^{(2)} + 3 > \beta^{(2)} + 1$, Page 1 and 2 are disjoint. Therefore, we first handling missing value $x_{2n+1}$ and $x_{2n+2}$ with Formula 12. Then we calculate the corresponding $z_{4n+1} \cdots z_{4n+4}$. Finally, considering $2\alpha^{(3)} - 1 = 4n+5$, the concatenation re-calculate from $z_{4n+5}$ until convergence.*

### D. Concatenating Overlapped Pages

The $(m-1)$-th page and the $m$-th page overlap if and only if $\beta^{(m-1)} + 1 > \alpha^{(m)}$. The process of concatenating overlapped pages involves three steps: (1) Select the original $x_i$ for $\alpha^{(m)} \le i \le \beta^{(m-1)}$ in the overlap. If both pages contain points with identical timestamps, retain the points on the page with the larger page number. (2) Recalculate $z_i$ for $2\alpha^{(m)} - 1 \le i \le 2\beta^{(m-1)}$ in the overlap. (3) Recalculate $z_i$ from $i = 2\beta^{(m-1)} + 1$ until convergence.

**Example 4.** *Figure 5(c) shows the concatenating process of disjoint pages. Page 3 stores $x^{(3)}_{2n+3}, x^{(3)}_{2n+4}, \cdots, x^{(3)}_{3n}$, and Page 4 stores $x^{(4)}_{3n-1}, x^{(4)}_{3n}, \cdots, x^{(4)}_{4n}$. Since $x_{3n-1}$ is delayed, it is absent on Page 3 and stored instead on Page 4. Referring to Formula 12, the pre-calculated results $z^{(3)}_{6n-3}$ and $z^{(3)}_{6n-2}$ are obtained, resulting in an overlap of $z$ across two pages. As for concatenation, we first select the original $x_i$ within the overlapped points, i.e., $x_{3n-1} = x^{(4)}_{3n-1}$, and $x_{3n} = x^{(3)}_{3n}$. Then we re-calculate each overlapped $z_i$ as $z_i = \mathcal{F}(i, \alpha^{(m-1)}, L, b, z)$ for $4n + 1 \le i \le 4n + 4$. Finally, the concatenation re-calculates from $z_{6n+1}$ until convergence.*

## V. System Deployment

OneRoundSTL has been included as a function in Apache IoTDB [10], an open-source time series database management system. The document is available on the product website [14] and the code is included in the system repository [15]. Figure 6 presents an overview of the deployment.

### A. Decomposition Query

Since the decomposing process is triggered by query, we first define the decomposition results of specific queries.

**Definition 7** (Decomposition query)**.** *The decomposition query returns trend components $\tau$ and seasonal component $s$ of time series $x$ in the specified range, starting from timestamp $\eta$ to the end $\theta$, i.e., $\tau_i$ and $s_i$, where $\eta \le i \le \theta$.*

The corresponding SQL statement is as follows.

```
select x,trend(x),season(x),residual(x)
from root.device
where time>=2 and time<=7
```

At this point, the seasonal components, trend components, and residual components of time series $x$ with timestamps in the range of [2, 7] are returned.

### B. Re-calculation on Truncated Pages

Considering that the pre-calculation has already performed forward substitution, we only need to perform backward substitution at this time. However, the pre-calculated results are the forward substitution results of the entire data page, ranging from $\alpha$ to $\beta$, rather than the query range $\eta$ to $\theta$. Therefore, we need to do head re-calculation, which adjust the pre-calculated to start from $\eta$, and tail re-calculation, which adjust the pre-calculated to end with $\theta$.

*1) Head Re-calculation:* The re-calculated results $\bar{z}_i$ is computed as

$$\bar{z}_i = \mathcal{F}(i, \eta, L, b, z)$$

Similar to the re-calculation during concatenation, head re-calculation also shows the convergence property.

**Proposition 6** (Convergence of Head Re-calculation)**.** *The re-calculated results $\bar{z}_i = \mathcal{F}(i, \eta, L, b, z)$ converges to $z_i$, i.e., for*

*any positive number $\zeta$, there exists $K \in \mathcal{N}, K \geq 2\eta - 1$, such that for all $i > K$ having*

$$|\bar{z}_i - z_i| < \zeta.$$

Based on Proposition 6, we re-calculate the head $z_i$ as

$$z_i = \mathcal{F}(i, \eta, L, b, z) \tag{14}$$

for $i = 2\eta - 1$ until converge. As a result, the initial pre-calculation timestamp for $z$ shifts from $\alpha$ to $\eta$.

**Example 5.** *Figure 6 illustrates the query processing of OneRoundSTL. As indicated, $x_1, \ldots, x_n$ are stored in Page 1, and $x_{n+1}, \ldots, x_{2n}$ are in Page 2. The SQL statement specifies the query of $x_i$, $\tau_i$, $s_i$, $r_i$ for $i = 2, 3, \ldots, 2n - 1$. Hence, we employ head re-calculation to adjust the starting point of pre-calculation from 1 to 2. Specifically, we compute $z_i = \mathcal{F}(i, 2, L, b, z)$ for $i = 3 \ldots$ until convergence.*

*2) Tail Re-calculation:* In order to adjust $\theta$ as the ending timestamp for pre-calculated results, we need to re-calculate $z_i$. Since the end of the matrix $L$ and $D$ only affect the last four rows as introduced in Proposition 2, we only need to re-calculate the last four $z_i$, i.e., $i = 2\theta - 3, \ldots, 2\theta$. Since $L$ and $D$ have $N$ rows, we designate $\theta - N$ as the starting point for re-calculation, i.e.,

$$z_i = \mathcal{F}(i, \theta - N, L, b, z) \tag{15}$$

for $2\theta - 3 \leq i \leq 2\theta$.

**Example 6** (Example 5 continued)**.** *Note that the query range in Figure 6 spans from 2 to $n-1$. We employ tail re-calculation to adjust the ending point of the pre-calculated results to $n-1$. Specifically, we compute $z_i = \mathcal{F}(i, n - 1 - N, L, b, z)$ for $2n - 5 \leq i \leq 2n - 2$.*

After the re-calculation is completed, we can use backward substitution referring to Formula 10 as

$$y_i = \mathcal{B}(i, \theta, L, d, z, y) \tag{16}$$

for $2\eta - 1 \leq i \leq 2\theta$ to obtain the decomposition results $y$, as mentioned in Section II-C4. Finally, we extract the seasonal and residual components from $z$ as $\tau_i = y_{2i-1}$ and $s_i = y_{2i}$.

### C. Query Processing

Algorithm 3 shows the details of OneRoundSTL in the data query phase. For each pair of consecutive pages in the query range, Line 2 concatenate them. The results of concatenation are $x$ and $z$ of all pages involved in the query. As we only query the decomposition results of some of the data, Line 3 performs head re-calculation to adapt to the query start point, and Line 4 performs tail re-calculation to adapt to the query end point. Finally, Line 5 obtains the decomposition result.

The time complexity of the concatenation is O($\phi$), where $\phi$ is the minimum number of re-calculations to converge. The head re-calculation is also related to the convergence, with a time complexity of O($\phi$). The tail re-calculation is only for the last 4 rows, with a time complexity of O(1). The time complexity of each $z_i$ backward substitution in decomposing is O(1), so the total time complexity is O($2Q$), where $Q = \theta - \eta$ is the size of the data being queried. In summary, the total time complexity is O($2Q + \phi$).

**Algorithm 3:** OneRoundSTL Query

**Input:** $x \in \mathbb{R}^N, t \in \mathbb{R}^N, z \in \mathbb{R}^{2N}, v \in \mathbb{R}^T, L \in \mathbb{R}^{2N \times 2N}, D \in \mathbb{R}^{2N \times 2N}, \eta \in \mathbb{N}^+, \theta \in \mathbb{N}^+$
**Output:** $\tau \in \mathbb{R}^N, s \in \mathbb{R}^N$

1 **foreach** *pair of consecutive pages $(m-1)$-th page and $(m)$-th page where $\beta^{(m-1)} \geq \eta$ and $\alpha^{(m)} \leq \theta$* **do**
2  concatenate by Formula 13;
3  head re-calculate by Formula 14;
4  tail re-calculate by Formula 15;
5  decomposing by Formula 16;
6 **return** $\tau, s, r$;

TABLE II: Comparison on online complexity

| Algorithm | Online Complexity |
|---|---|
| OnlineSTL | O(M) |
| OneShotSTL | O(I) |
| BacktrackSTL | O(W) |
| OneRoundSTL | **O(1)** |

We provide Table II to compare the online complexity of OneRoundSTL with the competitors. Note that we only compare online methods, as existing batch methods, such as RobustSTL, have significantly higher time costs than online methods. Specifically, OnlineSTL's complexity depends on the period length $M$, OneShotSTL's complexity is related to the user-defined maximum number of iterations $I$, and BacktrackSTL is influenced by the user-defined window size $W$. In contrast, OneRoundSTL requires only 4 calculations per data point, resulting in $O(1)$ complexity, independent of any parameters. Therefore, OneRoundSTL clearly outperform existing online methods.

## VI. Experiment

The experiments evaluate (1) decomposition effectiveness and efficiency of OneRoundSTL; (2) performance of OneRoundSTL deployed in time series database Apache IoTDB. We further provide ablation study and parameter study of OneRoundSTL in the appendix [24].

### A. Experimental Setup

*1) Datasets:* We use two synthetic datasets and two real-world datasets, consistent with the setup in [7]. The synthetic datasets have corresponding decomposition ground truth, which can be used to evaluate the effectiveness. And the real-world datasets can be used to evaluate efficiency in the real-world application scenarios. The datasets and their properties are briefly listed in Table III.

- **Triangle** [7] dataset is generated with a triangle wave, and inject an abnormal value into the residual component, to test the robustness of the algorithm.

TABLE III: Features of datasets

| | Type | Time series size $N$ | Period $M$ |
|---|---|---|---|
| Triangle | Synthetic | 1,000,000 | 144 |
| Square | Synthetic | 1,000,000 | 144 |
| Power | Real | 5,241,600 | 144 |
| Voltage | Real | 22,825,440 | 1440 |

• **Square** [7] dataset is generated with a square wave, and add a periodic mutation to the seasonal component, to test the algorithm's ability to deal with periodic mutations.
• **Power** [27] dataset contains records of power consumption from three different distribution networks in Tetouan city, located in northern Morocco.
• **Voltage** [28] dataset contains records of electricity consumption in a house located in Sceaux.

*2) Metrics:* There are three evaluation metrics involved in the experiment: (1) RMSE (Root Mean Square Error) calculates the RMSE between the decomposition result and the ground truth. (2) Space Cost calculates the space occupied by storing the experimental results. The calculation adopts the space cost of 64-bit compiler, that is, each floating-point number occupies 4 bytes. (3) Time Cost calculates the running time of the decomposition process. The results of the $LDL^T$ decomposition, $L$ and $D$, can be computed and stored on disk as prior knowledge. Therefore, the time for $LDL^T$ decomposition is not included.

*3) Methods:* We compare with the state-of-the-art methods, including RobustSTL showing the best decomposition effect, and STL having the highest decomposition efficiency.

• **OneRoundSTL** is our Algorithm 3, which utilizes pre-calculation to improve decomposition performance
• **STL** [5] is a batch STD algorithm using Loess.
• **RobustSTL** [17] is a batch STD algorithm using bilateral filter and non-local seasonal filter.
• **FastRobustSTL** [18] is an extension of RobustSTL, using the ADMM algorithm to improve computational efficiency.
• **OnlineSTL** [6] is an online STD method using tri-cube kernel filter and exponential smoothing.
• **OneShotSTL** [7] is an online STD algorithm using l1-norm filter and solved by IRLS (iterative reweighted least squares).
• **BacktrackSTL** [29] is a non-iterative online STD algorithm that integrates outlier-resilient smoothing, non-local seasonal filtering, and a backtracking technique.

### B. Comparison of Decomposition

The comparative experiment uses synthetic datasets, with known ground truths. It consists of three parts: (1) example decomposition; (2) comparison of decomposition effects; (3) comparison of decomposition efficiency.

*1) Example Decomposition:* We present typical decomposition results on synthetic and real-world datasets, providing a more intuitive visualization of the decomposition effect. As for synthetic dataset, Figure 7 shows that OneRoundSTL can extract trend features well. Although the seasonal component of

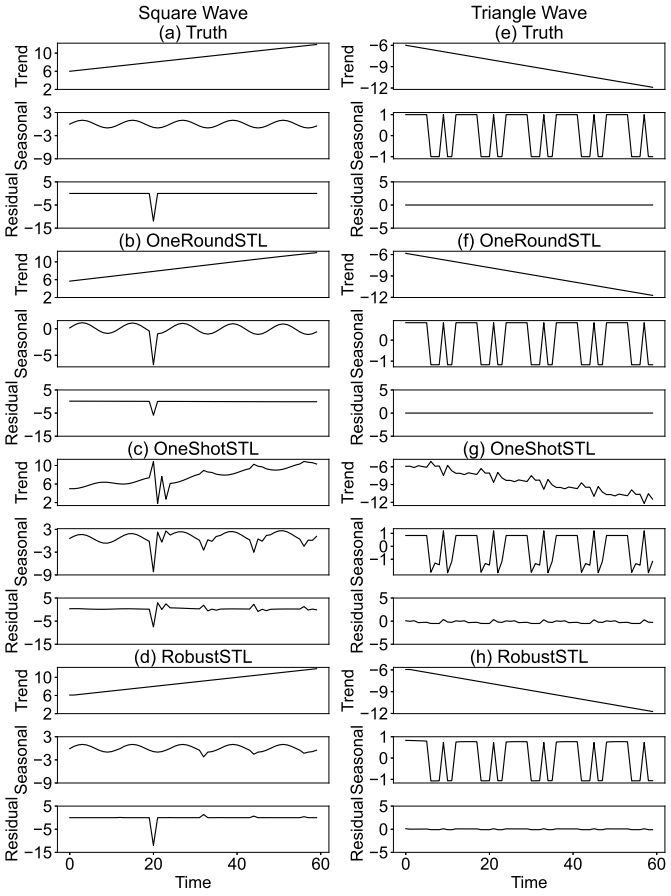


Fig. 7: Decomposition effectiveness on Synthetic Dataset

the outlier is affected, the seasonal features outside the outlier are well preserved. In contrast, OneShotSTL is significantly affected by the outlier around time 20. This is because the IRLS method used by OneShotSTL cannot accurately solve the l1-norm, thereby affecting its decomposition performance. For RobustSTL, the current best-performing method, its results are closest to the ground truth due to its superior filter. Note that RobustSTL is affected by periodic mutations. At this time the residual components in Figure 7(d) exhibit greater fluctuations compared to Figure 7(a), i.e., OneRoundSTL shows superior accuracy relative to RobustSTL. As for real-world dataset, Figures 8 show the decomposition results of OneRoundSTL, OneShotSTL and RobustSTL on Power and Voltage datasets. It is evident that OneRoundSTL yields a smoother trend component compared to OneShotSTL and RobustSTL. In the seasonal component, OneRoundSTL is capable of capturing finer variations, such as the spikes near the local minima of the data, which in real-world scenarios could represent nighttime electricity inspections. Decomposition results from other methods are provided in the appendix [24].

*2) Effectiveness:* Table IV quantitatively shows the effects of various methods. Due to the memory overhead exceeding 64GB for RobustSTL and FastRobustSTL on the full dataset, we first test their effectiveness using a smaller dataset of

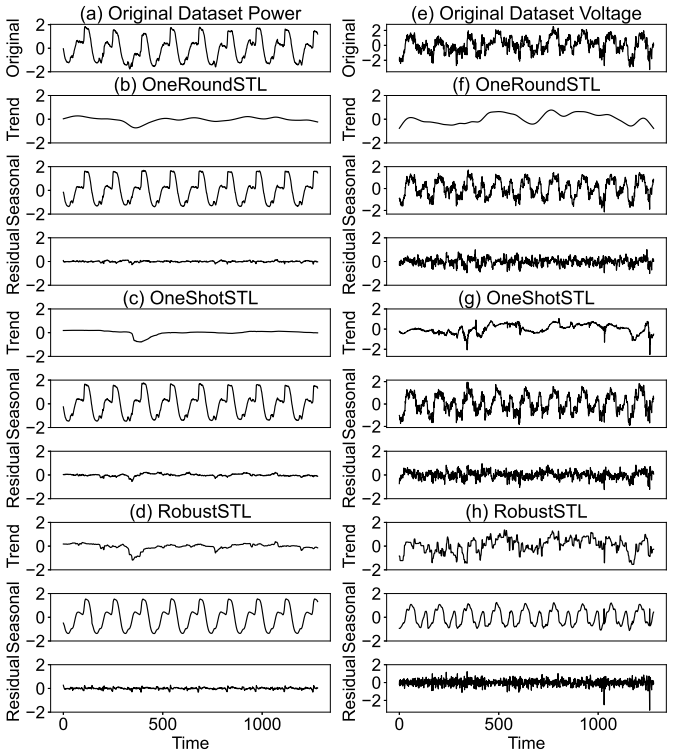


Fig. 8: Decomposition effectiveness on Real-world Dataset

size 1k. As presented, OneRoundSTL present superior RMSE performance over batch method STL and all of the streaming method such as OnlineSTL. It is because the Hodrick-Prescott filter used in OneRoundSTL provides better smoothing effects than filters in other methods, such as the tri-cube filter used by OnlineSTL. Moreover, it offers competitive results even with the state-of-the-art method, RobustSTL, similar to the qualitative analysis in Section VI-B1. It shows that OneRoundSTL handle square waves more effectively than RobustSTL.

*3) Efficiency:* Table IV also shows the time cost of algorithms. OneRoundSTL outperforms all existing methods, with particular advantages over RobustSTL and FastRobustSTL due to the use of pre-calculation. Section V-C shows that OneRoundSTL's online complexity is O(*1*), while the single loop complexity of RobustSTL is O($N^2$), multiple loops are required for convergence. The existing state-of-the-art efficient method, BacktrackSTL, has an online complexity of $O(W)$, where $W$ is the user-defined window size. This $W$ is significantly larger than the computation required for each data point in OneRoundSTL.

### C. Comparison in TSDB

The comparison in TSDB consists of four parts: (1) varying query size; (2) evaluation addressing the missing value issue; We provide experiments on varying page sizes and evaluation addressing the out-of-order issue in the appendix [24].

*1) Query Size:* Due to the low decomposition efficiency of RobustSTL and FastRobustSTL, which limits their practicality for database queries, we present their results only for query

TABLE IV: Comparison on decomposition effectiveness

| Dataset | Size | Algorithm | Trend (RMSE) | Seasonal (RMSE) | Residual (RMSE) | Time Cost (seconds) |
|---|---|---|---|---|---|---|
| Triangle Wave | 1,000 | OneRoundSTL | 0.030 | 0.024 | 0.043 | **0.002** |
| | | STL | 0.013 | 0.453 | 0.453 | 0.018 |
| | | OnlineSTL | 0.054 | 0.056 | 0.031 | 0.088 |
| | | OneShotSTL | 0.771 | 0.741 | 0.057 | 0.108 |
| | | BacktrackSTL | 0.136 | 0.368 | 0.056 | 0.037 |
| | | RobustSTL | 0.017 | 0.006 | 0.018 | 55.690 |
| | | FastRobustSTL | 0.007 | 0.016 | 0.016 | 41.295 |
| | 1,000,000 | OneRoundSTL | 0.024 | 0.020 | 0.035 | **0.078** |
| | | STL | 0.002 | 0.420 | 0.412 | 0.248 |
| | | OnlineSTL | 0.053 | 0.053 | 0.034 | 28.476 |
| | | OneShotSTL | 0.824 | 0.814 | 0.128 | 22.723 |
| | | BacktrackSTL | 0.201 | 0.531 | 0.183 | 1.297 |
| | | RobustSTL | - | - | - | - |
| | | FastRobustSTL | - | - | - | - |
| Square Wave | 1,000 | OneRoundSTL | 0.025 | 0.026 | 0.001 | **0.004** |
| | | STL | 0.106 | 0.107 | 0.003 | 0.038 |
| | | OnlineSTL | 0.050 | 0.051 | 0.001 | 0.097 |
| | | OneShotSTL | 0.232 | 0.153 | 0.131 | 0.433 |
| | | BacktrackSTL | 0.136 | 0.104 | 0.010 | 0.081 |
| | | RobustSTL | 0.069 | 0.017 | 0.015 | 118.910 |
| | | FastRobustSTL | 0.003 | 0.039 | 0.039 | 49.041 |
| | 1,000,000 | OneRoundSTL | 0.028 | 0.028 | 0.002 | **0.084** |
| | | STL | 0.104 | 0.182 | 0.005 | 0.267 |
| | | OnlineSTL | 0.068 | 0.068 | 0.003 | 30.834 |
| | | OneShotSTL | 0.241 | 0.168 | 0.162 | 81.036 |
| | | BacktrackSTL | 0.137 | 0.174 | 0.021 | 1.435 |
| | | RobustSTL | - | - | - | - |
| | | FastRobustSTL | - | - | - | - |

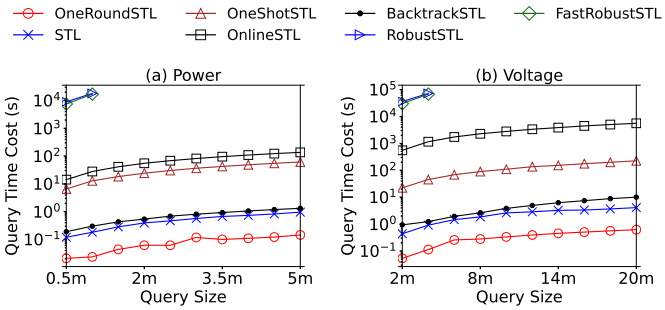


Fig. 9: Varying query size

sizes of 0.5m and 1m. Figure 9 shows that while query times increase with data size for all methods, OneRoundSTL consistently performs over an order of magnitude faster than the other algorithms, demonstrating its superior efficiency. Notably, OnlineSTL's efficiency varies with period length, as seen in Figure 9(b) for the Voltage dataset compared to Figure 9(a) for Power. In contrast, OneRoundSTL maintains stable efficiency regardless of period length.

*2) Missing Value Evaluation:* Figure 10 and 11 shows that OneRoundSTL maintains stable performance with a low missing rate, but its performance significantly degrades with substantial irregularities. Moreover, as the missing rate and length increases, the number of handled data points rises, leading to higher pre-calculation time costs.

To address the issue of poor performance when handling a large number of missing values, we extend the missing value handling approach in Section III-C. Considering the limited historical information, we use all available data to handle missing values. Specifically, we employ LOESS regression models trained on data with the same phase to capture trend and seasonal information. As observed in Figures 10 and 11,

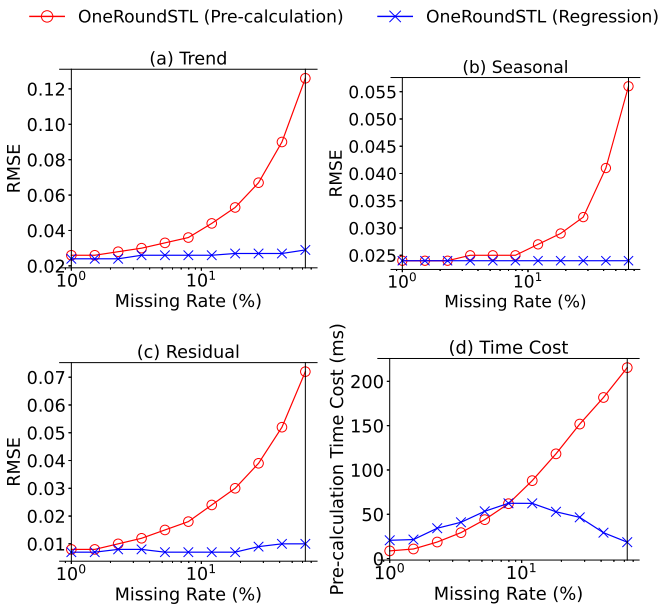


Fig. 10: Varying missing rate

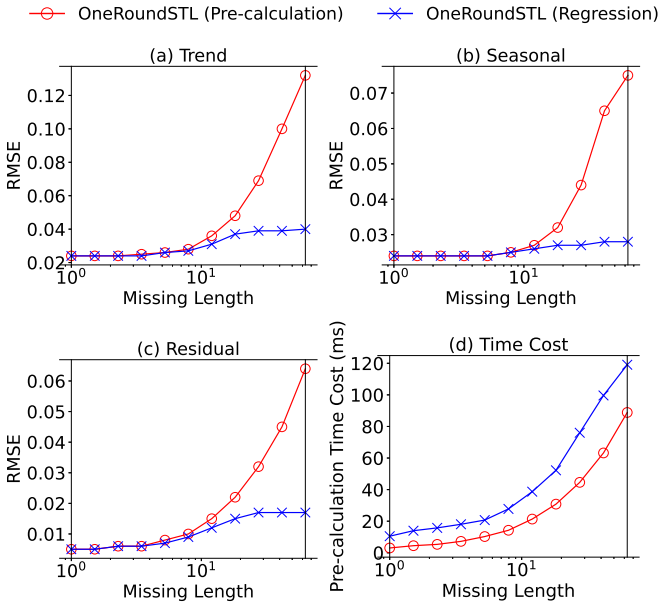


Fig. 11: Varying missing length

OneRoundSTL (Pre-calculation) and OneRoundSTL (Regression) exhibit nearly identical performance when missing rates and lengths are low, with the former demonstrating notably lower runtime. This highlights the proposed OneRoundSTL's advantages under minimal missing conditions. However, as missing rate and length increase, OneRoundSTL (Regression) is less impacted than OneRoundSTL (Pre-calculation). Notably, OneRoundSTL (Regression) shows reduced runtime at high missing rates due to lower the training cost of the regression model.

## VII. Related Work

We discuss the related studies on (1) seasonal-trend decomposition and (2) time series database storage.

### A. Seasonal-Trend Decomposition

Existing seasonal-trend decomposition methods can be divided into batch methods and online methods. (1) Batch methods, such as STL [5], RobustSTL [17] and FastRobustSTL [18], decompose the entire time series. These methods achieve good decomposition effects and are widely used in time series analysis. However, their low decomposition efficiency renders them unsuitable for database scenarios. (2) Online methods, such as OnlineSTL [6] and OneShotSTL [7], can decompose time series in a streaming manner. These methods exhibit high efficiency and suitability for database streaming scenarios. However, they suffer from low effect compared to batch methods.

### B. Time Series Database Storage

Time series databases for IoT widely adopt Log-Structured Merge-Tree (LSM-Tree) [11] for storage, such as InfluxDB [9] and IoTDB [8], since LSM-Tree can effectively deal with write-intensive scenarios. Existing studies have leveraged the structure of LSM-Tree to accelerate the query of data features. LSMAR [30] learns AutoRegression models in LSM-Tree to accelerate time series prediction. LSMAR also utilizes pre-calculation and stores the pre-calculated results in the metadata of each page. Absalyamov et al. [31] learn data synopses in LSM-Tree, such as histograms and wavelets. However, none of these methods consider seasonal-trend decomposition.

## VIII. Conclusion

In this paper, we propose OneRoundSTL an efficient method of seasonal-trend decomposition in time series database. To improve efficiency, our novel one round decomposition design enables pre-calculation in database, which is unlike for the existing methods with multiple rounds of data scan. The implementation of OneRoundSTL in a time series database has three parts, (1) pre-calculation in each page, (2) concatenation the pre-calculated results of multiple queried pages, and (3) online re-calculation of some results for the decomposition query. Owing to the existence of missing and delayed values, we use historical features to handle the data absent from a page in pre-calculation. OneRoundSTL has been deployed as a function of Apache IoTDB, an open source time series database. Experiments on real and synthetic data demonstrate that OneRoundSTL significantly improves the decomposition efficiency while keeping the decomposition effect.

## Acknowledgement

This work is supported in part by the Ministry of Industry and Information Technology, the National Natural Science Foundation of China (62021002, 92267203, 62072265, 62232005), the National Key Research and Development Plan (2021YFB3300500), and Beijing Key Laboratory of Industrial Big Data System and Application. Shaoxu Song (https://sxsong.github.io/) is the corresponding author.

## REFERENCES


[1] T. Kieu, B. Yang, C. Guo, C. S. Jensen, Y. Zhao, F. Huang, and K. Zheng, "Robust and explainable autoencoders for unsupervised time series outlier detection," in *38th IEEE International Conference on Data Engineering, ICDE 2022, Kuala Lumpur, Malaysia, May 9-12, 2022*. IEEE, 2022, pp. 3038–3050. [Online]. Available: https://doi.org/10.1109/ICDE53745.2022.00273

[2] N. Laptev, S. Amizadeh, and I. Flint, "Generic and scalable framework for automated time-series anomaly detection," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, L. Cao, C. Zhang, T. Joachims, G. I. Webb, D. D. Margineantu, and G. Williams, Eds. ACM, 2015, pp. 1939–1947. [Online]. Available: https://doi.org/10.1145/2783258.2788611

[3] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021, pp. 22 419–22 430.

[4] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, "Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting," in *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 2022, pp. 27 268–27 286. [Online]. Available: https://proceedings.mlr.press/v162/zhou22g.html

[5] R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning, "Stl: A seasonal-trend decomposition," *J. Off. Stat*, vol. 6, no. 1, pp. 3–73, 1990.

[6] A. Mishra, R. Sriharsha, and S. Zhong, "Onlinestl: Scaling time series decomposition by 100x," *Proc. VLDB Endow.*, vol. 15, no. 7, pp. 1417–1425, 2022. [Online]. Available: https://www.vldb.org/pvldb/vol15/p1417-mishra.pdf

[7] X. He, Y. Li, J. Tan, B. Wu, and F. Li, "Oneshotstl: One-shot seasonal-trend decomposition for online time series anomaly detection and forecasting," *Proc. VLDB Endow.*, vol. 16, no. 6, pp. 1399–1412, 2023. [Online]. Available: https://www.vldb.org/pvldb/vol16/p1399-he.pdf

[8] "https://iotdb.apache.org."

[9] "https://github.com/influxdata/influxdb."

[10] C. Wang, J. Qiao, X. Huang, S. Song, H. Hou, T. Jiang, L. Rui, J. Wang, and J. Sun, "Apache iotdb: A time series database for iot applications," *Proc. ACM Manag. Data*, vol. 1, no. 2, pp. 195:1–195:27, 2023. [Online]. Available: https://doi.org/10.1145/3589775

[11] P. E. O'Neil, E. Cheng, D. Gawlick, and E. J. O'Neil, "The log-structured merge-tree (lsm-tree)," *Acta Informatica*, vol. 33, no. 4, pp. 351–385, 1996. [Online]. Available: https://doi.org/10.1007/s002360050048

[12] Y. Kang, X. Huang, S. Song, L. Zhang, J. Qiao, C. Wang, J. Wang, and J. Feinauer, "Separation or not: On handing out-of-order time-series data in leveled lsm-tree," in *38th IEEE International Conference on Data Engineering, ICDE 2022, Kuala Lumpur, Malaysia, May 9-12, 2022*. IEEE, 2022, pp. 3340–3352. [Online]. Available: https://doi.org/10.1109/ICDE53745.2022.00315

[13] C. Fang, Z. Chen, S. Song, X. Huang, C. Wang, and J. Wang, "On reducing space amplification with multi-column compaction in apache iotdb," *Proc. VLDB Endow.*, vol. 17, no. 11, pp. 2974–2986, 2024.

[14] "https://iotdb.apache.org/UserGuide/latest/SQL-Manual/UDF-Libraries_apache.html."

[15] "https://github.com/apache/iotdb/tree/research/OneRoundSTL."

[16] "https://github.com/czjthss/OneRoundSTL."

[17] Q. Wen, J. Gao, X. Song, L. Sun, H. Xu, and S. Zhu, "Robuststl: A robust seasonal-trend decomposition algorithm for long time series," in *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 2019, pp. 5409–5416. [Online]. Available: https://doi.org/10.1609/aaai.v33i01.33015409

[18] Q. Wen, Z. Zhang, Y. Li, and L. Sun, "Fast robuststl: Efficient and robust seasonal-trend decomposition for time series with complex patterns," in *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, R. Gupta, Y. Liu, J. Tang, and B. A. Prakash, Eds. ACM, 2020, pp. 2203–2213. [Online]. Available: https://doi.org/10.1145/3394486.3403271

[19] Q. Wen, K. He, L. Sun, Y. Zhang, M. Ke, and H. Xu, "Robustperiod: Robust time-frequency mining for multiple periodicity detection," in *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, G. Li, Z. Li, S. Idreos, and D. Srivastava, Eds. ACM, 2021, pp. 2328–2337. [Online]. Available: https://doi.org/10.1145/3448016.3452779

[20] X. Song, Q. Wen, Y. Li, and L. Sun, "Robust time series dissimilarity measure for outlier detection and periodicity detection," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022*, M. A. Hasan and L. Xiong, Eds. ACM, 2022, pp. 4510–4514. [Online]. Available: https://doi.org/10.1145/3511808.3557686

[21] A. Dokumentov and R. J. Hyndman, "Str: Seasonal-trend decomposition using regression," *arXiv preprint arXiv:2009.05894*, 2020.

[22] A. Moghtaderi, P. Borgnat, and P. Flandrin, "Trend filtering: Empirical mode decompositions versus l1 and hodrick-prescott," *Adv. Data Sci. Adapt. Anal.*, vol. 3, no. 1-2, pp. 41–61, 2011. [Online]. Available: https://doi.org/10.1142/S1793536911000751

[23] J. Fu and S. Levine, "Offline model-based optimization via normalized maximum likelihood estimation," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. [Online]. Available: https://openreview.net/forum?id=FmMKSO4e8JK

[24] "https://github.com/czjthss/OneRoundSTL/blob/main/supplementary.pdf."

[25] S. Song, A. Zhang, J. Wang, and P. S. Yu, "SCREEN: stream data cleaning under speed constraints," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, T. K. Sellis, S. B. Davidson, and Z. G. Ives, Eds. ACM, 2015, pp. 827–841. [Online]. Available: https://doi.org/10.1145/2723372.2723730

[26] A. Zhang, S. Song, and J. Wang, "Sequential data cleaning: A statistical approach," in *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, F. Özcan, G. Koutrika, and S. Madden, Eds. ACM, 2016, pp. 909–924. [Online]. Available: https://doi.org/10.1145/2882903.2915233

[27] A. Salam and A. El Hibaoui, "Comparison of machine learning algorithms for the power consumption prediction:-case study of tetouan city–," in *2018 6th International Renewable and Sustainable Energy Conference (IRSEC)*. IEEE, 2018, pp. 1–5.

[28] B. N. E. Houda, L. Lakhdar, and A. Meraoumia, "Time series analysis of household electric consumption with xgboost model," in *4th International Conference on Pattern Analysis and Intelligent Systems, PAIS 2022, Oum El Bouaghi, Algeria, October 12-13, 2022*. IEEE, 2022, pp. 1–6. [Online]. Available: https://doi.org/10.1109/PAIS56586.2022.9946913

[29] H. Wang, H. Guo, Z. Zhu, Y. Zhang, Y. Zhou, and X. Zheng, "Backtrackstl: Ultra-fast online seasonal-trend decomposition with backtrack technique," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*, R. Baeza-Yates and F. Bonchi, Eds. ACM, 2024, pp. 5848–5859. [Online]. Available: https://doi.org/10.1145/3637528.3671510

[30] Y. Su, W. Ma, and S. Song, "Learning autoregressive model in lsm-tree based store," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, A. K. Singh, Y. Sun, L. Akoglu, D. Gunopulos, X. Yan, R. Kumar, F. Ozcan, and J. Ye, Eds. ACM, 2023, pp. 2061–2071. [Online]. Available: https://doi.org/10.1145/3580305.3599405

[31] I. Absalyamov, M. J. Carey, and V. J. Tsotras, "Lightweight cardinality estimation in lsm-based systems," in *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, G. Das, C. M. Jermaine, and P. A. Bernstein, Eds. ACM, 2018, pp. 841–855. [Online]. Available: https://doi.org/10.1145/3183713.3183761