Research on Time Series Decomposition for Real-time Analysis of Dynamic and Stable Behaviors in Streaming Data

A Dissertation submitted in partial fulfilment
of the requirements for the degree of

Doctor of Philosophy

by
Thanapol Phungtua-eng

Department of Information Science and Technology

Graduate School of Science and Technology

Shizuoka University

June 2024

Acknowledgments

I would like to take this opportunity to express my deep gratitude to my adviser, Dr. Yoshitaka Yamamoto, for guiding me into research on time series data mining. His invaluable guidance, mentorship, and generous support have been instrumental throughout my Ph.D. studies. I am extremely fortunate and grateful to have received his guidance in completing my research and Ph.D.

I express my gratitude to the colleagues I met during my time at the Laboratory of Big Data Processing at Shizuoka University for their invaluable support and friendship. I would also like to thank my colleagues at Shizuoka University: K. Oliva, B. Yao, C. Xiao, as well as C. Chokchai and C. Amornbunchornvej, who provided guidance, encouragement, and camaraderie throughout my Ph.D. journey.

I am grateful to M. Aizawa, K. Kashiyama, and N. Arima for their useful suggestions and for providing the knowledge and dataset used for evaluation in Chapter 3. Specifically, I would like to thank Dr. Shigeyuki Sako at the University of Tokyo for his invaluable opportunities to research in astroinformatics, as well as for his useful suggestions and for providing the light curve data for the synthetic dataset.

I would also like to express my heartfelt gratitude to the Royal Thai Government's scholarships and the Thai Embassy for their financial support during my Ph.D. studies, and for their assistance in enabling my travel to Japan during the COVID-19 pandemic in 2021.

Finally, I would like to thank my wife, Sasipha Yoonuch, for being my partner in life and my greatest supporter.

To my wife, my partner in life and my greatest supporter, Sasipha Yoonuch

ABSTRACT

The past decade has seen a surge in research using time series data to analyze and understand both known and unknown phenomena in the natural sciences. Time series data inherently contain a mix of *dynamic* and *stable* behaviors: dynamic behaviors reflect sudden changes, and stable behaviors reveal persistent patterns. These behaviors appear in the three primary *time series components*: trend, seasonal, and residual components. The trend component represents the long-term or movement, the seasonal component represents the repeating cycles at their specific intervals (e.g., weekly, monthly, yearly), and the residual component represents the noise remaining after extracting trend and seasonal components.

The intertwining of these components and behaviors often complicates the analysis to distinguish and interpret underlying insights. One technique that tackles the tangle within time series data is *time series decomposition*. It is a crucial technique for separating original data into its trend, seasonal, and residual components. This separation helps analysts clearly understand dynamic and stable behaviors in those components. Such clarity not only enhances analytical capabilities but also supports more informed decision-making. However, traditional decomposition methods face significant challenges. Firstly, those methods are not suitable for streaming data, where behaviors continuously change over time. Secondly, their effectiveness heavily depends on specific input parameters (e.g., segmentation size and buffer size).

In this thesis, we propose three novel time series decomposition methods in response to these challenges: Elastic Data Binning (EBinning), Online Season Length Estimation (OnlineSLE), and Adaptive Seasonal-Trend Decomposition (ASTD). Each method

is designed to extract insights from dynamic and stable behaviors in time series components with enhanced flexibility and minimal reliance on predefined parameters. EBinning is a novel, parameter-free method that effectively captures dynamic behaviors in the trend component, which is crucial for analyzing phenomena in the natural sciences. This method dynamically adjusts segmentation sizes based on observed data behaviors, allowing for more accurate capture of significant changes without manual adjustment. Additionally, EBinning is specifically designed to capture phenomena that are both unknown and short-lived. Its adaptability makes it especially useful in environments where characteristics of the trend component continuously change over time.

OnlineSLE method rapidly and accurately identifies season lengths in time series data, referring to the intervals of cycles that characterize the seasonal component. Unlike EBinning, which focuses on trend components, this method interprets the stable and unstable behaviors of the seasonal component in real-time by estimating season lengths. Constant season lengths indicate stable behaviors, while changing season lengths indicate unstable behaviors. This precision in estimation ensures that the season lengths are accurately aligned with the observed behaviors in the time series, thereby enhancing the reliability of the real-time decomposition.

Finally, ASTD is a parameter-free method for the real-time decomposition of seasonal and trend components. This method integrates the Seasonal-Trend Decomposition technique with OnlineSLE, enabling it to utilize the optimal season length for decomposition. This integration allows ASTD's decomposition results to accurately reflect stable and unstable behaviors in the seasonal component. However, a remaining challenge for ASTD

is effectively capturing dynamic behaviors in the trend component, which could potentially be addressed by integrating EBinning with ASTD in future enhancements.

We conducted extensive evaluations of these methods using synthetic datasets. Additionally, we showcased their versatility across various fields, including astronomy, meteorology, and cardiology, emphasizing their robustness and broad applicability. The results demonstrate their superiority over traditional methods such as latency time, accuracy, decomposition quality and more. Those results highlight significant advancements in three methods to distinguish and interpret dynamic and stable behaviors for each time series component in streaming data. Our proposed methods provide valuable tools for scientists to enhance their knowledge and gain insights into phenomena in the natural sciences. We intend to present these methods as tools for distinguishing and interpreting behavior in time series, but also as a foundation for further exploration in the broader domain of time series data mining. They hold potential for applications in prediction, forecasting, anomaly detection, and discord discovery.

Contents

List of Figures **xi**

List of Tables **xv**

1 Introduction **1**
1.1 Our Contributions 5
1.2 List of Publications 6
1.3 Organization of the Thesis 8

2 Time Series Components and Decomposition **9**
2.1 Time Series 10
2.1.1 Four Kinds of Behaviors in Time Series 15
2.1.2 Definition of the Residual Component 19
2.1.3 Subsequence of Time Series 21
2.2 Time Series Decomposition 24
2.2.1 Dynamic Behavior Capturing using Data Binning 26
2.2.2 Season Length Estimation (SLE) 29
2.2.3 Seasonal-Trend Decomposition (STD) 32
2.3 Discussion and Conclusion 35

3 Elastic Data Binning **37**
3.1 Background 38
3.1.1 Light Curves 38
3.1.2 Dynamic Behavior Captured in TDAA 39
3.1.3 Data Binning in TDAA 43
3.2 Related Work 45
3.2.1 Data Sketching and Binning 45
3.2.2 Time Series Analysis for TDAA 46
3.3 Elastic Data Binning 50
3.3.1 Mean-EBinning (M-EBinning) 52
3.3.2 Linear-EBinning (L-EBinning) 56
3.4 Light Curve Datasets 59

3.4.1 Synthetic datasets . . . 61
3.4.2 Real world datasets . . . 63
3.5 Experimental Setting . . . 64
3.5.1 Window Size . . . 64
3.5.2 Symbol Size for SAX ,1D-SAX . . . 65
3.5.3 Subsequence Length for MP . . . 65
3.5.4 Z-normalized and Non-normalized Euclidean Distance for MP . . . 66
3.6 Experimental Results . . . 66
3.6.1 Experimental Results with Synthetic Datasets . . . 66
3.6.2 Experimental Results with Real-World Datasets . . . 76
3.7 Discussion and Conclusions . . . 80

4 Online Season Length Estimation **82**
4.1 Background . . . 83
4.2 Related Work . . . 84
4.2.1 Problem statement . . . 84
4.2.2 Periodgram and Auto-correlation Function . . . 85
4.2.3 Existing SLE methods . . . 90
4.3 Online Season Length Estimation . . . 92
4.3.1 Sliding Discrete Fourier Transform (SDFT) . . . 93
4.3.2 Spectral Peak Location Estimator . . . 97
4.3.3 OnlineSLE method . . . 101
4.4 Datasets . . . 103
4.4.1 Synthetic Datasets . . . 104
4.4.2 Real-world Datasets . . . 105
4.5 Experimental Setting . . . 106
4.5.1 Comparison Methods . . . 106
4.5.2 Evaluation Metrics . . . 107
4.6 Experimental Results . . . 110
4.6.1 Time Performance Evaluation . . . 110
4.6.2 Error in Numerical Computation of SDFT Evaluation . . . 112
4.6.3 Accuracy by Spectral Peak Location Estimator of OnlineSLE . . . 113
4.6.4 Accuracy Rate Evaluation with Synthetic Datasets . . . 115
4.6.5 Accuracy Rate Evaluation with Real-world Datasets . . . 117
4.7 Discussion and Conclusions . . . 118

5 Adaptive Seasonal-trend Decomposition **119**
5.1 Background . . . 120
5.2 Related Work . . . 122
5.2.1 Single vs. Multiple seasonal component . . . 122
5.2.2 Existing STD methods . . . 123
5.3 Adaptive Seasonal-Trend Decomposition . . . 127
5.3.1 Algorithm . . . 127
5.3.2 Trend Filter . . . 130
5.3.3 Seasonal Filter . . . 131

5.3.4 OnlineSLE . . . 132
5.4 Datasets . . . 133
5.4.1 Synthetic datasets . . . 133
5.4.2 Real-world Datasets with Seasonality Tansitions or Fluctuations . . 134
5.4.3 Real-world datasets from Comprehensive R Archive Network (CRAN) 137
5.5 Experimental Setting . . . 138
5.5.1 Metrics for decomposition quality with synthetic datasets . . . 138
5.5.2 Metrics for Decomposition Quality with Real-World Datasets . . . 138
5.6 Experimental Results . . . 142
5.6.1 Experimental Results with Synthetic Datasets . . . 142
5.6.2 Experimental Results with Real-world Datasets . . . 149
5.7 Discussion and Conclusions . . . 154

6 Discussions and Future Challenges **156**
6.1 Elastic Data Binning (EBinning) . . . 156
6.1.1 Contributions and Problem-Solving Capabilities of EBinning . . . 156
6.1.2 Remaining Challenges and Limitations of EBinning . . . 158
6.2 Online Season Length Estimation (OnlineSLE) . . . 160
6.2.1 Contributions and Problem-Solving Capabilities of OnlineSLE . . . 160
6.2.2 Remaining Challenges and Limitations of OnlineSLE . . . 161
6.3 Adaptive Seasonal-trend Decomposition (ASTD) . . . 163
6.3.1 Contributions and Problem-Solving Capabilities of ASTD . . . 163
6.3.2 Remaining Challenges and Limitations of ASTD . . . 165
6.4 Conclusion . . . 167

7 Conclusions **168**
7.1 Summary . . . 168
7.2 Future Work . . . 170

Bibliography **172**

A Supplementary Materials **183**
A.1 Analyzing Flares with M-EBinning . . . 183
A.2 Datasets and Source Codes . . . 184

List of Figures

1.1 Time series data example in various fields. (Top) Light intensity measurements of a celestial object, illustrating the astronomical time series data. (Bottom) Daily mean sea surface temperature (SST) data, showcasing an example of time series data used in meteorological monitoring. 2
1.2 Data decomposition results for Figure 1.1. (Top) Light curve sketch with red lines showing mean values of segments from data binning and green lines marking Kepler flare artifacts. (Bottom) SST data cycles from seasonal-trend decomposition. 3

2.1 A simple time series and its components. (Top-left) Original time series. (Top-right) Trend component of this time series. (Bottom-left) Seasonal component of this time series. (Bottom-right) Residual component of this time series. 12
2.2 Examples of time series data illustrating the combination of different components: (Top-left) Light curve without astronomical phenomena, (Top-right) Light curve with stellar flare, (Middle-left) Electrocardiogram from healthy volunteers, (Middle-right) Time series from a rotation sensor GyrY during walking and joggjng, (Bottom-left) Monthly carbon dioxide concentration in the global atmosphere, (Bottom-right) Number of employees in US retail. . 13
2.3 Examples of noise and unwanted outliers in time series data. (Top) Electricity transformer temperature including noise from various factors, (Bottom) Light curve with heavy noise from cloud turbulence and outliers from satellite flares. 20
2.4 Series of subsequences segmented by 100 instances, with segmentation points marked by green lines on the original time series. The bottom plot shows the mean values of each subsequence, indicated by a red line. 23
2.5 Series of subsequences segmented by 200 instances corresponding to the length of each cycle. The top figure shows segmentation points marked with green lines on original time series, and the bottom figure displays plots for each subsequence. 24
2.6 Data binning process in time series analysis: (Top) Segmentation of the time series with bin size of 20 instances (green lines); (Bottom) Compressed bins showing mean values (red lines). 27

2.7 Influence of varying the bin size in data binning. (Top) Sample representation of data binning with a large bin size; (Bottom) Sample representation of data binning with a small bin size. 28
2.8 Three time series data, arranged from top to bottom as follows: daily SST, monthly mean sunspot number, and weekly influenza cases. 30
2.9 Seasonal-trend decomposition results of monthly CO2 from Figure 2.2 (Bottom-right), performed using STL with the season length of 12. 33
2.10 Seasonal-trend decomposition results of monthly CO2 from Figure 2.2 (Bottom-right), performed using STL with the season length of 7. 34

3.1 Comparison between the video for each frame and the LC that is derived from this video. 39
3.2 Comparison of an unwanted outlier (Left) and a transient pattern (Right) in LCs. The insets in the top-right corners show detailed cut-outs of the images captured at the moments the signals appear, highlighting the sources of the variations. 42
3.3 Example of an artificial flare, indicated by the red line, overlaid on the normalized flux of a celestial object depicted by the blue line. 42
3.4 Comparative results between PAA, SAX, and 1D-SAX. The original time series is depicted in blue, while the results from each method are shown in red. 46
3.5 Two boundaries in the same distribution. 54
3.6 Four scenarios in the LC. 61
3.7 Comparison between the smallest and the largest transient pattern of Square- and Triangle-LCs. 62
3.8 Comparison between the smallest and the largest Kepler transient pattern. 63
3.9 Comparison between classical and complex flares in real-world dataset. . . . 64
3.10 Comparison of varying IOU results. 67
3.11 Sample results of MP-Z where the red lines indicate the start and end points of the transient pattern. The blue highlight marks the subsequence containing the transient pattern, while the green highlight shows the subsequence most similar to the blue subsequence as identified by MP-Z. 70
3.12 Comparison of IOU results for Kepler pattern. The solid red line represents the result of EBinning, the dashed red line represents the peak of the Kepler pattern, and the green line represents all phases of the Kepler pattern. . . . 71
3.13 Comparison of Euclidean distance by various parameters. 73
3.14 Comparison of ROC curve by various methods. 75
3.15 Sketching results showing transient patterns from stellar flares [3], where the black lines represent the original LCs and the blue lines represent the results from M-EBinning. 77
3.16 Example of LC including nearby star 79
3.17 LC from a trajectory of the artificial object: (Left) Full-length LC with the trajectory of the artificial object occurrence highlighted in red; (Right) A zoomed-in view of the period highlighted in red. 79
3.18 Video from a trajectory of the artificial object. 80

4.1 Two classical time series. (Top) Monthly totals of international airline passengers [11].; (Bottom) Monthly numbers of sunspots[112] 84
4.2 Comparison between the time-domain and frequency-domain expressions the periodogram is an alternative representation of the original data. 86
4.3 Illustration of frequency resolution issues in the periodogram due to influencer of data points in the DFT. 87
4.4 Comparison between the original time series and the ACF. 88
4.5 ACF analysis on a subset of the time series. 89
4.6 ACF analysis on a the time series with Gaussian noise injection. 90
4.7 Comparison of FFT results between timestamps 4 and 5 with $k = 1$, demonstrating the SDFT update process. 94
4.8 Example of DFT coefficients resolution issue. 97
4.9 Visualization of the synthetic datasets Syn1 and Syn2, demonstrating the different seasonality patterns and noise levels. 105
4.10 The real-world datasets, where the red line indicating the point of rapid rotation of the tilt table affecting the volunteers. (Top) ECG dataset; (Bottom) ABP dataset. 106
4.11 First 1600 timestamps of the real-world datasets from Fig. 4.10. (Left) ECG dataset; (Right) ABP dataset. 106
4.12 Comparison of time performance for different SLE methods. 110
4.13 Numerical error computation between FFT and SDFT by various datasets. (Left) Syn1; (Right) Syn2. 113
4.14 Comparison of accuracy results with synthetic datasets by various estimators and sliding window size. (Left) the accuracy results with Syn1 (Ground truth = 100); (Middle) the accuracy results with the first phase of Syn2 (Ground truth = 50); (Right) the accuracy results with the second phase of Syn2 (Ground truth = 80). 114

5.1 STD results on a time series with seasonality changes marked by a red line at timestamp 300, highlighting the challenges posed by fixed seasonal lengths. 121
5.2 Samples of seasonality transitions and fluctuations: (Left) At the 3800 timestamps, a seasonality transition leads to a significantly shorter season length in subsequent cycles. (Right) there was a fluctuation in seasonality between 1860 and 1870, characterized by a season length that was shorter than a decade. 122
5.3 Procedure of the initialization phase 127
5.4 Comparison of trend results. Note that red line overlaps with the black line. 131
5.5 Syn1 and its components. 134
5.6 Syn2 and its components. 135
5.7 All datasets of Real1, arranged from top to bottom as follows: WalkJogRun1, WalkJogRun2, CO2, Canadian Lynx, SOI, and Sunspots. 136
5.8 Comparison of three time series based on their Kruskal-Wallis p-values, from highest (left) to lowest (right), indicating increasing seasonality consistency. 141
5.9 Comparison of Syn1's trend component by various STD methods. 143
5.10 Comparison of Syn2's trend component by various STD methods. 143
5.11 Comparison of Syn1's seasonal component by various STD methods. 145

5.12 Comparison of Syn2's seasonal component by various STD methods. 145
5.13 Comparison of Syn1's residual component by various STD methods 146
5.14 Comparison of Syn2's residual component by various STD methods 146
5.15 Comparison of MSE by various season lengths and methods. Color legends: Our proposed method, ASTD (■), FastRobustSTL (■), OneShotSTL (■), OnlineSTL (■), STR (■), STL (■), and SlidingSTL (■). 148
5.16 Comparison of Syn1's trend component by various STD methods ($m = 129$). 150
5.17 Comparison of Syn1's seasonal component by various STD methods ($m = 129$).150
5.18 Comparison of Syn1's residual component by various STD methods ($m = 129$).151
5.19 Comparison of decomposition quality on **Real1** by various season length and methods. Color legends: Our proposed method, ASTD (■), FastRobustSTL (■), OneShotSTL (■), OnlineSTL (■), STR (■), STL (■), and SlidingSTL (■). 152

6.1 Comparison between the original LC with the artifact Kepler flare and the results from M-EBinning. The green highlight indicates the Kepler flare, with the peak of this flare occurring at timestamp 200. 158
6.2 Comparison of EBinning results with different initial bin sizes. 159
6.3 Half-hourly electricity demand for Victoria, a multiple seasonal component time series displaying daily and weekly seasonal components [5]. (Top) seven cycles corresponding to the daily seasonal component, (Bottom) cycles corresponding to the weekly seasonal component. 163

A.1 All flare periods representation results obtained using M-EBinning. 184
A.2 All flare periods representation results obtained using M-EBinning (continued).185

List of Tables

2.1 Comparison of four behaviors in time series 17

3.1 Comparison IOU of Square-LCs representation by varying methods 67
3.2 Comparison IOU of Triangle-LCs representation by varying methods 68
3.3 Comparison IOU of Kepler-LCs representation by varying methods 68
3.4 Detection results with real flare by various methods 78

4.1 Computational cost of each method 112
4.2 Comparison of accuracy rate results with **Syn1** by various methods. 116
4.3 Comparison of accuracy rate results with **Syn2** by various methods. The bolded and underlined results represent the highest accuracy rate across SLE methods. 116
4.4 Experimental results with real-world datasets by various methods. The bolded and underlined results represent the highest accuracy rate across SLE methods. 117

5.1 STD Methods Comparison. 126
5.2 Evaluation of Decomposition Quality on **Real2** across different season lengths. The bolded and underlined results indicate the lowest values among online TSD methods. 152

A.1 List of datasets. 186
A.2 List of methods in Chapter 3. 186
A.3 List of methods in Chapter 4. 187
A.4 List of methods in Chapter 5. 187

Chapter 1

Introduction

Time series data are especially valuable in the natural sciences, where they help scientists explain known phenomena and discover new ones. Beyond their role in observation, time series data are an integral part of scientific inquiry, facilitating the study of temporal dynamics. They serve crucial analytical roles in various fields such as tracking stock price movements in finance, detecting temporal phenomena in astronomy, assessing climate phenomena levels in meteorology, monitoring network reliability in information technology, and more. We showcase the examples of time series data from various fields in Figure 1.1.

Figure 1.1 (Top) shows a time series from the Tomo-e Gozen system[1]. This time series is a measurement of the light intensity of a celestial object and contains noise from turbulence. It is known as a light curve[2]. Figure 1.1 (Bottom) shows a time series from the National Centers for Environmental Information (NOAA), representing daily mean

[1]The Tomo-e Gozen is an optical wide-field video observation system composed of a mosaic CMOS camera mounted on the 1.05 m Kiso Schmidt telescope. https://tomoe.mtk.ioa.s.u-tokyo.ac.jp/

[2]Light curve is described in detail in Chapter 3.

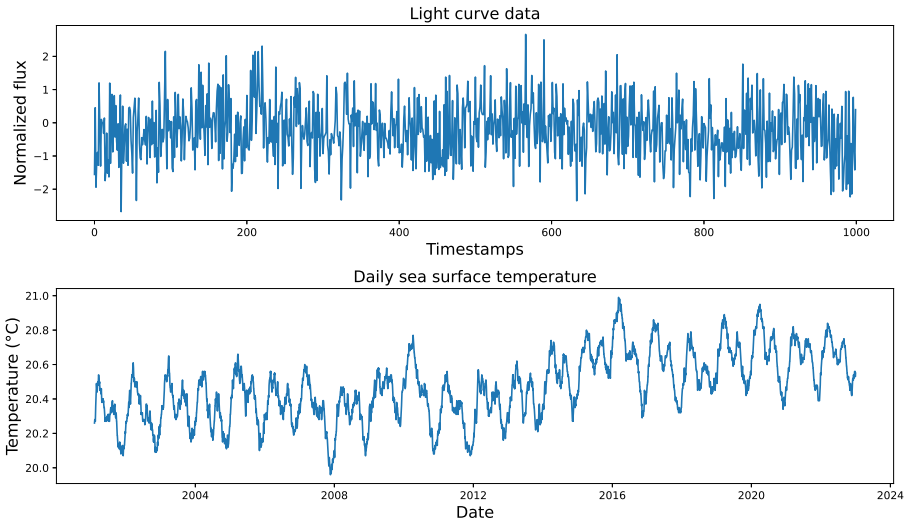


Figure 1.1: Time series data example in various fields. (Top) Light intensity measurements of a celestial object, illustrating the astronomical time series data. (Bottom) Daily mean sea surface temperature (SST) data, showcasing an example of time series data used in meteorological monitoring.

sea surface temperature (SST) [75]. Although these datasets are invaluable for scientific analysis, they are inherently complex and characterized by static, dynamic, stable, and unstable behaviors.

Dynamic and unstable behaviors, or changes in a time series, include sudden shifts or spikes in data points that can disrupt the overall stability of the series. These behaviors may also involve changes in cycle lengths or patterns, adding to the complexity. In contrast, stable and static behaviors, such as the predictable seasonal cycles observed in meteorological data or steady economic trends, are foundational for forecasting and long-term planning. Moreover, these behaviors also include unvarying trends where data points remain constant over time. Understanding the interplay between these behaviors is crucial for analysis, as

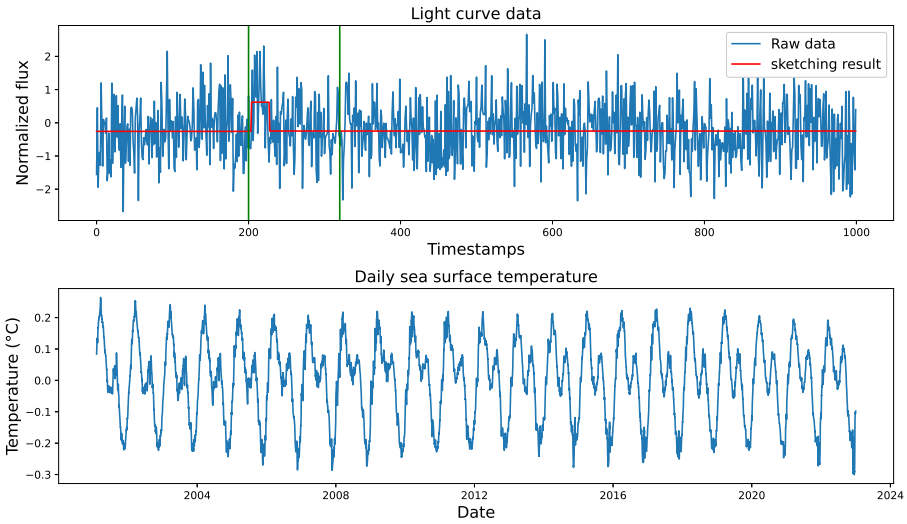


Figure 1.2: Data decomposition results for Figure 1.1. (Top) Light curve sketch with red lines showing mean values of segments from data binning and green lines marking Kepler flare artifacts. (Bottom) SST data cycles from seasonal-trend decomposition.

it involves distinguishing meaningful signals from noise and understanding how both can influence or distort the perceived phenomena. This characterization presents a significant challenge for scientists aiming to interpret the data meaningfully.

To give the reader a better understanding of why time series data are challenging to interpret meaningfully, we pose two questions related to Figure 1.1: 'Can we identify the transient changes in the light curve?' and 'Are the seasonal cycles of SST stable?' Characterizing the behaviors of these series requires careful monitoring. A thorough understanding of the data is essential, as simple analysis may not be sufficient. Here, we demonstrate an alternative way to visualize these time series data, showing how *time series decomposition* can improve interpretation, as shown in Figure 1.2.

In Figure 1.2 (Top), we utilize *data binning*[3], which decomposes the time series into segments and calculates the mean for each segment. The results are visually represented by the red line, which indicates the mean value of each segment. By examining this line, we can distinctly detect a sudden change at timestamp 200. This change is a phenomenon called a Kepler flare, which is characterized by a rapid rise followed by a slow decay [23, 34]. This pattern is typical of stellar flares in the universe.

Moving on to Figure 1.2 (Bottom), we utilize *seasonal-trend decomposition*[4] to extract trends, cyclical properties, and noise within the time series. To address the earlier question regarding the stability of cycles within the SST data, we plot these cycles from the decomposition results. These temperature cycles are influenced by seasonal variations, this fact was reported in the literature [98]. This visualization allows readers to quickly discern the cyclical properties within the time series. This highlights the importance of using time series decomposition techniques like data binning and seasonal-trend decomposition to extract and refine meaningful insights hidden within the time series data.

Time series decomposition is a preprocessing technique that breaks down complex data into more manageable and interpretable components, analogous to untangling a knotted rope to examine each individual strand [8, 45, 130]. This technique helps clarify the data, improving our understanding and facilitating processing for data analysis. The outcome of time series decomposition is important for various applications in data mining tasks, such as anomaly detection, time series forecasting, and classification.

In this thesis, we propose methods for time series decomposition with a particular

[3]Data binning is discussed in Section 2.2.1.

[4]Seasonal-trend decomposition is discussed in Section 2.2.3.

focus on applications in the natural sciences. Time series data in the natural sciences often exhibit specific features such as noise with large amplitude, cyclical variations, and trend variations. These characteristics result in a high signal-to-noise ratio, posing unique challenges for analysis. As shown in Figure 1.1, we observe that the impact of noise affects our ability to understand or recognize the underlying patterns and phenomena. Therefore, we liken our approach to carefully untangling a knotted rope, where meticulously separating each component or segment substantially enhances our ability to analyze and comprehend the natural phenomena being studied.

1.1 Our Contributions

In this thesis, we propose several novel methods in the field of time series decomposition, with a special emphasis on their applications within the natural sciences. The contributions detailed below not only enhance traditional techniques but also introduce new capabilities for real-time analysis. Our contributions include:

- We propose *Elastic Data Binning*, a novel method that dynamically adjusts free parameters based on specific periods and data characteristics. This method significantly enhances the adaptability and accuracy of capturing unstable behaviors in trend, that reflect to temporal phenomena such as stellar flares. Moreover, it is particularly effective in reducing noise in time series, ensuring more reliable analysis.

- We propose *Online Season Length Estimation* for dynamic analysis of streaming time series data. This method accurately estimates the duration of cycles within a dataset,

which is crucial for understanding dynamic behavior. Online Season Length Estimation offers fast and precise estimations, pivotal for real-time data analysis.

- We propose *Adaptive Seasonal-Trend Decomposition*, a novel method for real-time decomposition of streaming time series data. This method is parameter-free, eliminating the need for user-defined assumptions that can lead to errors in decomposition.

- We demonstrate that our proposed methods exhibit minimal or no influence from specified input parameters, reducing the risk of error or bias. This ensures more robust results, as evidenced by key metrics such as accuracy rate and decomposition quality.

- We demonstrate the effectiveness of our proposed methods across real-world datasets in various fields such as astronomy, meteorology, and cardiology. Moreover, we evaluate our proposed method through key metrics including decomposition quality, accuracy rates, and computation speed. These results highlight that our methods consistently outperform traditional methods in terms of both efficiency and accuracy.

1.2 List of Publications

The referred publications are as follows:

- T. Phungtua-Eng, Y. Yamamoto. Adaptive seasonal-trend decomposition for streaming time series data with transitions and fluctuations in seasonality. In *Proceedings of the 2024 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, **(To appear)** [86].

- T. Phungtua-Eng, Y. Yamamoto. A fast season length estimation using sliding discrete Fourier transform for time series streaming data. In *Proceedings of the 16th International Congress on Advanced Applied Informatics*, pages 482–487. [84].

- T. Phungtua-eng, S. Sako, Y. Nishikawa, and Y. Yamamoto. Elastic data binning: Time-series sketching for time-domain astrophysics analysis. *SIGAPP Appl. Comput. Rev.*, 23(2):5–22, 2023 [83].

- T. Phungtua-Eng, Y. Yamamoto, and S. Sako. Elastic data binning for transient pattern analysis in time-domain astrophysics. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing (SAC'23)*, pages 342–349, 2023. [91].

- T. Phungtua-Eng, Y. Yamamoto, and S. Sako. Dynamic binning for the unknown transient patterns analysis in astronomical time series. In *Proceedings of the 2021 IEEE International Conference on Big Data (BigData)*, pages 5988–5990, 2021 [89].

- T. Phungtua-Eng, Y. Yamamoto, and S. Sako. Detection for transient patterns with unpredictable duration using chebyshev inequality and dynamic binning. In *Proceedings of the 9th International Symposium on Computing and Networking Workshops*, pages 454–458, 2021 [88].

- T. Phungtua-eng, Y. Yamamoto, and S. Sako. Transient pattern detection from streaming nature data. In *Proceedings of the 8th International Symposium on Computing and Networking Workshops*, pages 435–439, 2020 [87].

1.3 Organization of the Thesis

The remainder of this thesis is organized as follows. Chapter 2 provides a background on time series data and time series decomposition. Chapter 3 describes the Elastic Data Binning method, a time series decomposition technique to capture transient changes for time-domain astrophysics analysis. Chapter 4 describes the Online Season Length Estimation method, used for analyzing cyclical properties within streaming time series data. Subsequently, Chapter 5 describes the Adaptive Seasonal-Trend Decomposition method, which integrates online season length estimation into the seasonal-trend decomposition technique for decomposing streaming time series data. After detailing our three proposed methods, Chapter 6 summarizes how these methods can be applied to solve problems in the application domain and discusses the remaining challenges. Finally, we conclude and discuss directions for future work in Chapter 7. The appendix includes URLs for further information on our methods, existing methods, and datasets used in this thesis, intended for those wishing to explore these resources and reproduce the results.

Chapter 2

Time Series Components and Decomposition

Time series decomposition involves breaking down time series into components such as long-term trends, cyclical behaviors, and irregularities [8, 45, 107]. This process helps to analyze and interpret the various underlying behaviors within the data, improving the precision of subsequent analyses. It is useful for predicting future trends, detecting anomalies, and understanding cyclical behaviors. The goal of time series decomposition is to extract relevant characteristics from the original time series by separating it into its constituent components. Our proposed methods align with this goal. Before delving into the details of these methods, it is crucial to have an understanding of the characteristics of time series and the mechanism of time series decomposition.

This chapter is the foundation for the understanding of the preliminary concepts used throughout the rest of this thesis. Section 2.1 begins with an overview of time series

data, explaining its characteristics, importance, and applications in various domains. Following this introduction, Section 2.2 delves into the state-of-the-art methods for analyzing time series, with a focus on time series decomposition. This section aims to showcase the advancements in the field and how these methods are applied to extract meaningful insights from complex time series data. By providing this background, the chapter sets the stage for the subsequent exploration of innovative time series decomposition methods developed in this thesis. Finally, we offer conclusions in Section 2.3.

2.1 Time Series

The time series data are utilized in this thesis are one-dimensional time series data. As previously mentioned, these consist of sequential measurements over time, where the x-axis represents the timestamp and the y-axis represents the measurement value at each timestamp. We begin by defining of a time series:

Definition 1 A *time series* Y is a sequence of measurements taken over the time, denoted as $Y = (Y_1, Y_2, \ldots, Y_t)$, where t represents the last time index and indicates that the series contains t observations.

The time series comprises three components: trend, seasonal, and residual [8, 19, 45, 107, 130]. These components are collectively represented as:

Definition 2 The time series Y consists of three components, formally expressed as: $Y = T + S + R$, where Y denotes the original time series, T denotes the trend component, S denotes the seasonal component, and R denotes the residual component. Each component itself is a time series.

Here, the details of each component are described below:

- *Trend component* (T) represents the long-term direction or movement in the time series [8, 19, 45, 54, 97, 125]. It captures the underlying pattern of gradual change, typically observed as a consistent increase or decrease over time. For instance, in a company's annual revenue data, a steady rise over several years would indicate a positive trend. This component helps identify whether the data is moving in a particular direction and is crucial for understanding overall progressions or declines within the dataset. The trend can change over time due to various factors, reflecting the temporal change of the underlying phenomenon. It may exhibit slow increases, sudden drops, and returns to slow increases, demonstrating responsiveness to external influences. Importantly, the trend component does not show recurrent patterns at specific intervals.

- *Seasonal component* (S) represents repeating cycles at specific intervals, such as daily, weekly, or yearly [7, 8, 12, 45, 132, 136]. Unlike the trend component, the seasonal component exhibits periodic or cyclical behavior. It allows analysts to understand and predict fluctuations in seasonality, which is particularly useful in planning and forecasting where seasonality plays a significant role. For instance, retail sales often peak during the holiday season each year, demonstrating a clear seasonal pattern. Recognizing these patterns can help in optimizing inventory and staffing levels during peak times.

- *Residual component* (R) represents the irregularity or noise that remain once the trend and seasonal components have been extracted [8, 45, 59, 115, 139]. These are the un-

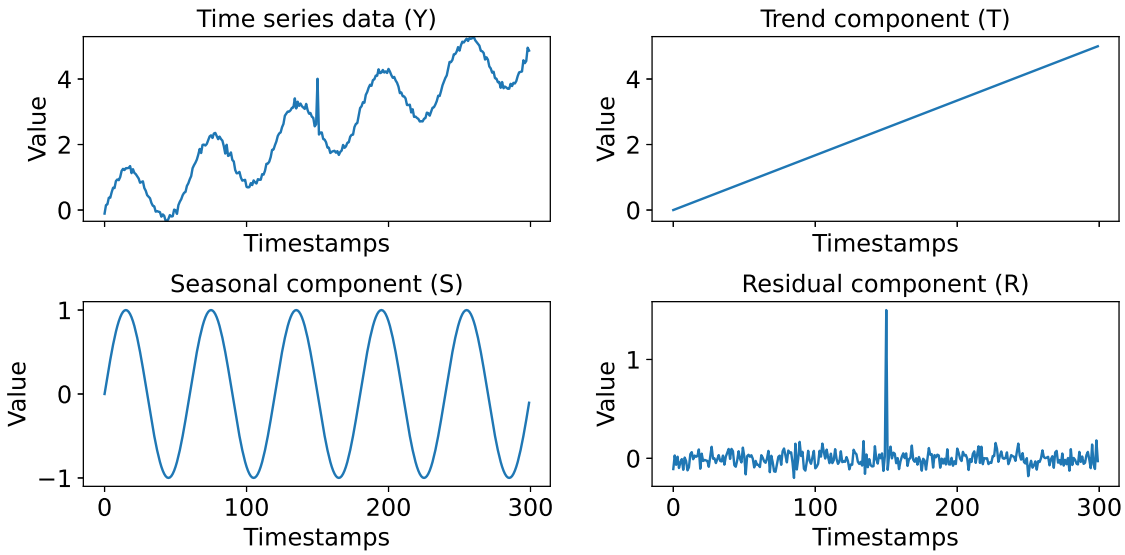


Figure 2.1: A simple time series and its components. (Top-left) Original time series. (Top-right) Trend component of this time series. (Bottom-left) Seasonal component of this time series. (Bottom-right) Residual component of this time series.

predictable or random fluctuations that cannot be attributed to the trend or seasonal factors. The residual component is critical for detecting outliers, or unexpected events within the time series data.

To help the reader better understand, we begin with a visual comparison between the original time series and its decomposed components, as shown in Figure 2.1. From the time series in Figure 2.1 (Top-left), we quickly discern three main characteristics: an increasing long-term trend, cyclical behaviors spanning five cycles, and a high spike at timestamp 150. These characteristics correspond to the trend, seasonal, and residual components of the time series.

To further illustrate the practical application of these components, we next show-

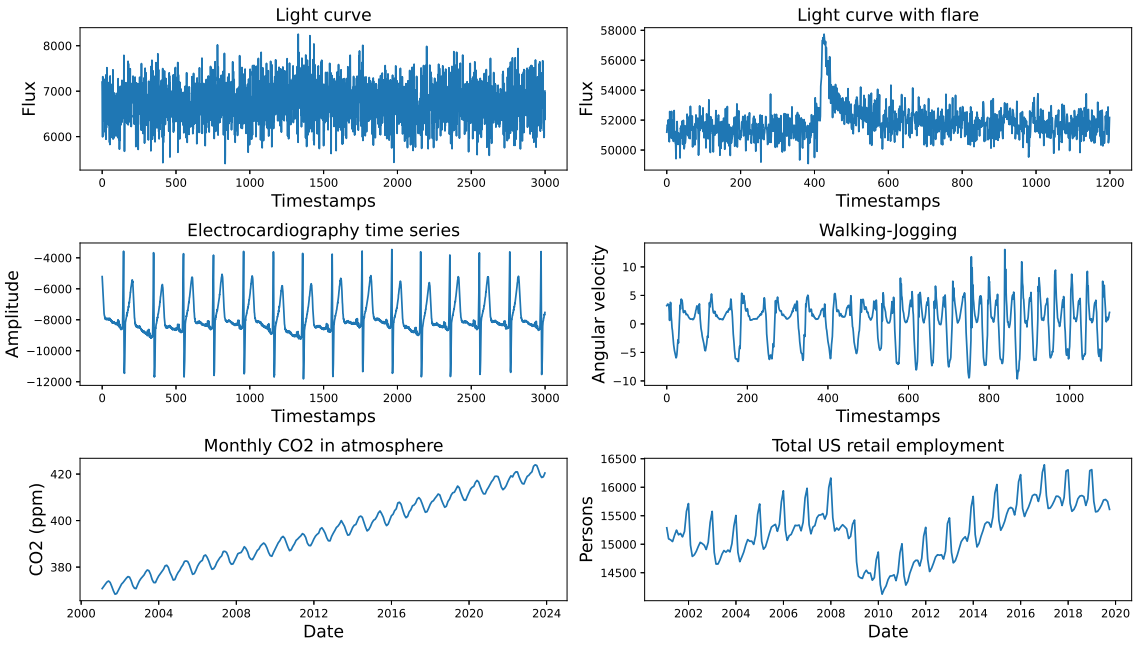


Figure 2.2: Examples of time series data illustrating the combination of different components: (Top-left) Light curve without astronomical phenomena, (Top-right) Light curve with stellar flare, (Middle-left) Electrocardiogram from healthy volunteers, (Middle-right) Time series from a rotation sensor GyrY during walking and joggjng, (Bottom-left) Monthly carbon dioxide concentration in the global atmosphere, (Bottom-right) Number of employees in US retail.

case how they combine in real-world data. Figure 2.2 shows various examples, each demonstrating a unique combination of the trend, seasonal, and residual components identified in the time series. A brief description of each component is provided below:

- **Figure 2.2 (Top-left):** This light curve measures the light intensity of a celestial object and includes noise. Astronomers have confirmed that this time series does not exhibit significant astronomical phenomena [3]. It shows no trend changes and lacks

cyclical behaviors associated with seasonal components.

- **Figure 2.2 (Top-right):** The light curve includes a stellar flare, verified by astronomers [3]. In this figure, we observe a sudden rise and a slow decay between 400 and 600. Additionally, it lacks cyclical behaviors associated with seasonal components.

- **Figure 2.2 (Middle-left):** An electrocardiogram from healthy volunteers measures the electrical activity of the heart [32, 37]. This time series shows the heart's beating cycles, indicative of the physiological state of the individual. The dataset shows stable cycles of the seasonal component, with no long-term trend, as the healthy volunteer remained in a stable condition on a tilt table with foot support.

- **Figure 2.2 (Middle-right):** This time series contains the angular velocity of the rotation sensor GyrY on the left forearm during walking and jogging [6, 32]. The activity is transited from walking to jogging around timestamp 800.

- **Figure 2.2 (Bottom-left):** Monthly carbon dioxide concentration in the global atmosphere shows a strong long-term increasing trend and stable cycles of the seasonal component due to monthly variations [57]. These data reflect the steady increase in carbon dioxide levels influenced by environmental and anthropogenic factors.

- **Figure 2.2 (Bottom-right):** Monthly employment figures in US retail, as depicted in the dataset from [45], show stable seasonal cycles but exhibit a sudden drop around 2009. This decline corresponds to the impact of the subprime mortgage crisis, which is detailed in [100].

From this figure, we categorize the data into three combinations: the trend compo-

nent including ($Y = T + R$) at the top row, the seasonal component including ($Y = S + R$) in the middle row, and both trend and seasonal components including ($Y = T + S + R$) at the bottom row of Figure 2.2. We observe that time series characteristics depend on the behavior of the phenomena or situations they represent. Typically, a time series exhibits four main behaviors: static, dynamic, stable, and unstable. Understanding these behaviors is crucial for scientists to gain insights into the phenomena or situations being studied.

2.1.1 Four Kinds of Behaviors in Time Series

To develop an understanding, we begin with the details of the four main behaviors: static and dynamic behaviors of the trend component, and stable and unstable behaviors of the seasonal component. The details of each behavior are as follows:

- *Static behavior* of the trend component represents a completely unvarying trend, implying no change in the data points over time. This behavior is characterized by a flat line with no upward or downward movement.

- *Dynamic behavior* of the trend component involves continuous or temporal changes compared to previous observations. This can be seen as a prolonged, slow increase or decrease over time, indicating long-term environmental or evolutionary trends.

- *Stable behavior* of the seasonal component represents stability and predictability in cycles. These stable behaviors are reflected by the regularity of cycles that recur on a daily, weekly, or annual basis, facilitating accurate predictions about seasonal effects on the data.

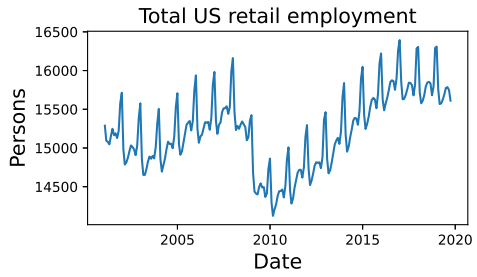
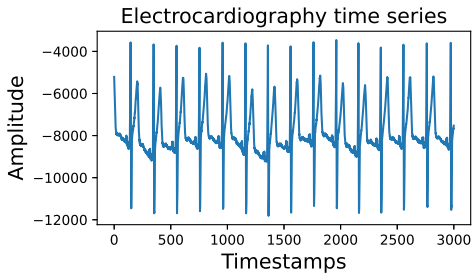
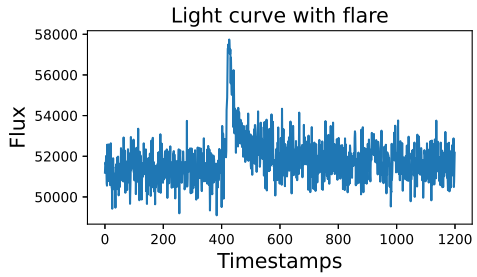
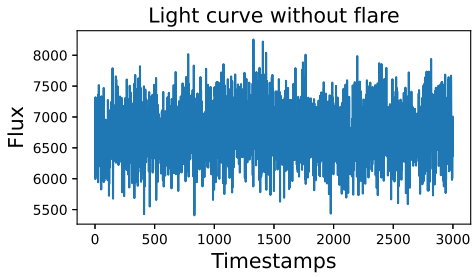
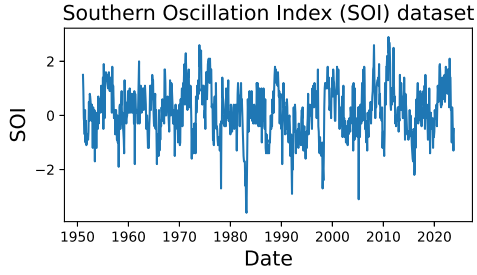
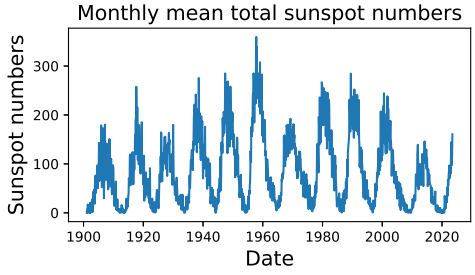
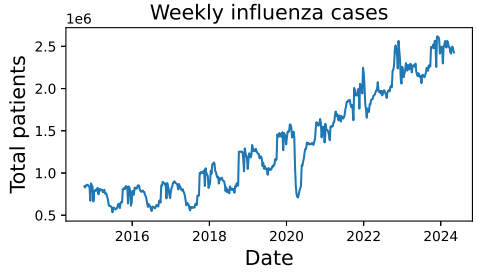
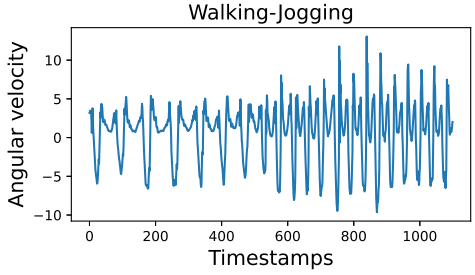
- *Unstable behavior* of the seasonal component responds to changes from external factors or natural phenomena. It manifests as alterations in the shapes of cycles or as changes in cycle lengths, transitioning from long to short durations. These adaptations in the seasonal components underscore the sensitivity of time series data to environmental changes and natural dynamics.

Notably, stable behavior may transition to unstable behavior when changes occur over time, and unstable behavior can also become stable under certain conditions. Here, we illustrate the comparison of these behaviors by grouping real-world datasets, as shown in Table 2.1. The details of each cell are as follows:

- **Top-left**: These figures include dynamic behavior in the trend component due to the subprime crisis for the US retail employment time series [100], and the rapid increase and slow decay between timestamps 400 to 600 in the light curve (LC) reflecting the transient occurrence of a stellar flare [3]. For the seasonal component, these figures show stable behavior. All cycles of the US retail employment time series are stable, and the LC has no seasonal component, confirmed by astronomers, indicating static behavior with no change in data points over time.

- **Top-right**: These figures exhibit static behavior in the trend component and stable behavior in all cycles of the seasonal component. The trend component shows a steady trend with no changes, similar to a constant in both figures. For the stable behavior of the seasonal component, the electrocardiography data are recorded from a healthy volunteer, showing no evidence of unstable heart rate cycles. The LC also indicates

Table 2.1: Comparison of four behaviors in time series

Seasonal component	Trend component: Dynamic behavior	Trend component: Static behavior
Stable behavior		
		
Unstable behavior		
		

stable behavior with no seasonal component.

- **Bottom-left**: These figures exhibit dynamic behavior in the trend component and unstable behavior in the seasonal component. The time series data in this scenario are influenced by real phenomena changes. These changes affect both the trend and seasonal components. For example, the Southern Oscillation Index (SOI) data records the monthly sea level pressure differences between the western and eastern tropical Pacific. This dataset is linked to climate oscillations and occurrences of extreme events such as El Niño and La Niña phenomena. Many external factors, such as volcanic eruptions [24], can contribute to the observed unstable and dynamic behaviors. Another example is the number of influenza cases in the US. A noticeable change occurred during 2020 due to the COVID-19 pandemic. People changed their behaviors, such as practicing social distancing and wearing masks, which impacted the decrease in influenza cases.

- **Bottom-right**: This figure exhibits unstable behavior in the seasonal component while lacking a trend component, indicating static behavior in the trend. The number of sunspots, which naturally follows an 11-year cycle, shows the Sun shifting from relatively calm to stormy periods. This phenomenon, known as solar maximum and solar minimum, involves the Sun's magnetic poles reversing [18]. This contributes to the dynamic behavior in the seasonal component within the time series. Additionally, as mentioned in Figure 2.2 (Middle-right), this time series records the angular velocity on the left forearm during walking and jogging. The cycles may change over time due to human activity, indicating unstable behavior in the seasonal component.

Following this subsection, we discuss the noise and outliers that typically appear in of the residual component after trend and seasonal decomposition. These noise and outliers are commonly exhibited in time series data of the natural sciences.

2.1.2 Definition of the Residual Component

As mentioned earlier, the residual component is the part remaining after extracting the trend and seasonal components. It typically includes noise, erroneous data, or unwanted information. This component may contain dynamic behaviors that are not of interest for analysis, referred to as 'unwanted outliers' [2]. An unwanted outlier in the residual component is a *data point* that behaves unusually at a specific time instant compared to neighboring points [2, 9, 59].

However, time series data of the natural sciences typically contain heavy noise and outliers from external factors that should be removed or cleaned to improve data quality. These unwanted outliers should generally be ignored during analysis to ensure more accurate results. We demonstrate two time series datasets, one from electricity transformer temperature and one from astronomy, to illustrate unwanted outliers caused by external factors, as shown in Figure 2.3.

Figures 2.3 (Top) illustrate examples of noise encountered while measuring the electricity transformer temperature (ETT) [138]. The ETT dataset is a crucial indicator for long-term electric power deployment. However, predicting ETT demand for a specific area is challenging because it varies with weekdays, holidays, seasons, and weather temperatures. These factors contribute to noise in the dataset. For example, the noise can result from

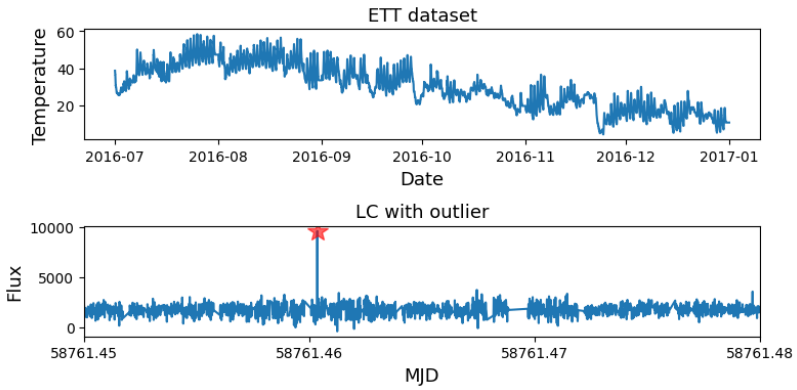


Figure 2.3: Examples of noise and unwanted outliers in time series data. (Top) Electricity transformer temperature including noise from various factors, (Bottom) Light curve with heavy noise from cloud turbulence and outliers from satellite flares.

weather conditions on a particular day or decreased electricity demand during extended holidays when companies operate outside regular working hours. This noise complicates efforts to predict and manage electricity demand.

Figures 2.3 (Bottom) provide examples of unwanted outliers from measurement errors caused by satellite flares, which commonly occur during sky surveys. Satellite flares are sudden, bright reflections of sunlight off satellite surfaces that can create spurious data points in astronomical observations. These unwanted outliers were verified by domain experts as being uninteresting and irrelevant for sky surveys [3].

Scientists are particularly interested in phenomena that exhibit dynamic behavior in the trend component and unstable behaviors in the seasonal component. These behaviors typically appear as *subsequences* of time series that collectively exhibit unusual patterns over time [9]. Such behaviors are important as they often reflect changes in real-world phenomena and are frequently referred to as 'anomalies' [2]. These anomalies are important because

they can indicate significant phenomena. For example, detecting and analyzing stellar flares can provide crucial insights into stellar dynamics and properties, as shown in Figures 2.2 (Top-right). Therefore, these anomalies are of high interest in natural sciences studies, contrasting sharply with unwanted outliers. In this thesis, we aim to characterize four behaviors, which are identified as *subsequences* of interest.

2.1.3 Subsequence of Time Series

To characterize behaviors within the trend and seasonal components, we analyze specific subsequences of a time series that exhibit distinct properties. This analysis is particularly useful in identifying anomalies, as detailed in previous sections. Moreover, analyzing these subsequences helps us understand localized behaviors that might not be apparent when examining the entire time series.

Definition 3 A *subsequence* $Y_{(i,\,m)}$ within time series Y is defined as a segment starting at index i and extending for m elements. Formally, $Y_{(i,\,m)} = (Y_i, Y_{i+1}, \ldots, Y_{i+m-1})$, where i denotes the starting point and m denotes the length of the subsequence, with the condition $1 \leq i \leq t - m + 1$.

Importantly, all subsequences defined in this thesis are non-overlapping, with each starting immediately after the previous one ends. These are collectively referred to as a *series of subsequences*.

Definition 4 A *series of subsequences* of Y is the collection of non-overlapping subsequences, formally expressed as $Y = (Y_{(1,\,m_1)}, Y_{(m_1+1,\,m_2)}, \ldots, Y_{(m_{q-1}+1,\,m_q)})$, where each subsequence follows consecutively without overlap from the end of the previous subsequence,

and the sum of the lengths of all subsequences equals to the total length of the original series: $\sum_{j=1}^{q} m_j = t$.

The series of subsequences serves as an alternative representation for characterizing the time series before decomposition. It enables the detailed examination of dynamic, static, stable, and unstable behaviors within the time series, thereby facilitating more precise decomposition. To provide the reader with a more concrete understanding of how a series of subsequences facilitates the decomposition process, consider two examples: the time series from a light curve containing a stellar flare, as illustrated in Figure 2.2 (Top-right), and the electrocardiogram data, as shown in Figure 2.2 (Middle-left). The decomposition results for each of these time series, segmented into subsequences, are shown in Figures 2.4 and 2.5, respectively.

We segment the time series into subsequences, each containing 100 instances. The series of subsequences can be expressed as $Y = (Y_{(1,\,100)}, Y_{(101,\,100)}, \ldots, Y_{(1101,\,100)})$. By segmenting the time series into subsequences, we can decompose the localized behaviors of each segment through averaging, and plot the mean values with a red line (as shown in the bottom part of Figure 2.4). we can discern significant deviations such as the pronounced change at $Y_{(401,\,100)}$ compared to the preceding subsequence $Y_{(301,\,100)}$. Without specific domain knowledge, we use this segmentation to identify dynamic behaviors in the trend. A notable example is at $Y_{(401,\,100)}$, which reflects a stellar flare. This flare represents an anomaly.

Similarly, the electrocardiogram data includes a seasonal component. We seg-

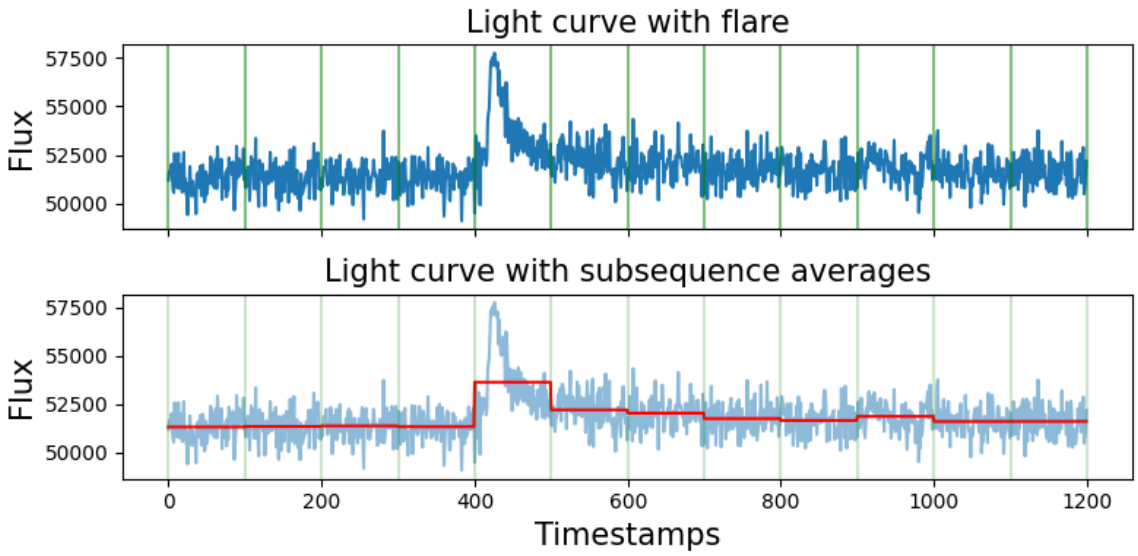


Figure 2.4: Series of subsequences segmented by 100 instances, with segmentation points marked by green lines on the original time series. The bottom plot shows the mean values of each subsequence, indicated by a red line.

mented this series into subsequences, each containing 200 instances, based on ground truth information. Each segment corresponds to one cycle of the seasonal component. We then plotted each subsequence on the same graph, as illustrated in Figure 2.5 (Bottom). This visualization clearly demonstrates stable behavior, showing that all cycles follow the same pattern. In this case, the subsequence length of 200 instances defining each cycle is termed the 'season length'. This parameter is crucial for decomposing time series that include a seasonal component, similar to how season length is used to decompose localized behaviors.

From these two examples, we observe that season length is a crucial parameter for segmenting time series into a series of subsequences. The next section will provide a brief overview of time series decomposition. It will focus on how these decomposition

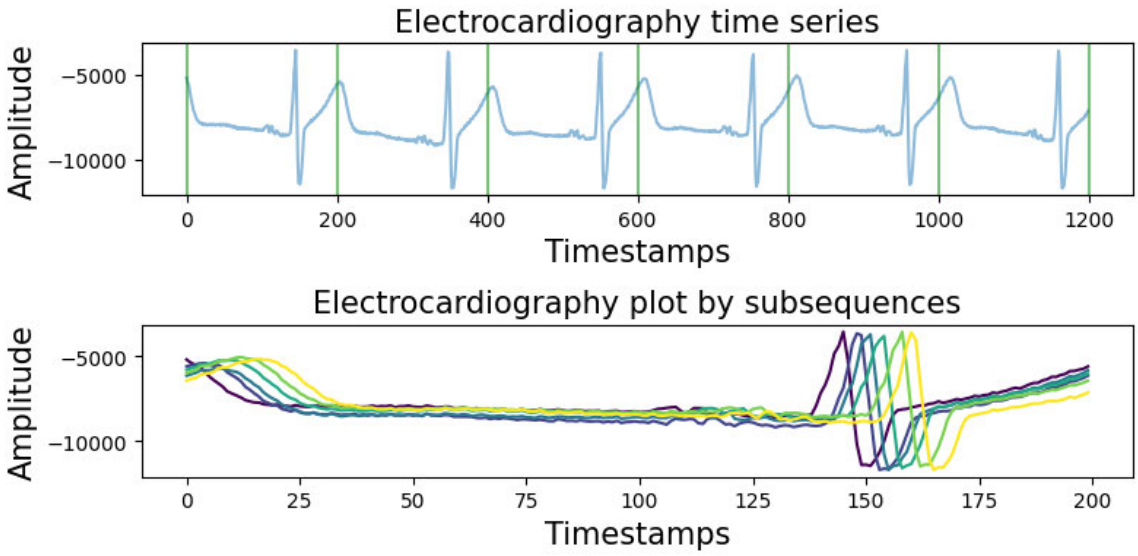


Figure 2.5: Series of subsequences segmented by 200 instances corresponding to the length of each cycle. The top figure shows segmentation points marked with green lines on original time series, and the bottom figure displays plots for each subsequence.

methods interact with different components of time series data, emphasizing the critical role of subsequence length for time series decomposition.

2.2 Time Series Decomposition

Numerous studies have been conducted on time series decomposition, using different techniques to break down complex data into more understandable and analyzable components. Some of the well-known techniques include data sketching [22], wavelet decomposition [108], seasonal trend decomposition [19], autoregressive integrated moving average (ARIMA) models [12], signal extraction [13], trend or seasonality filters [7, 54, 125], and more. Each of these methods provides a distinct approach for isolating and analyzing the

underlying components within time series data, catering to various types of analysis and application requirements. Our thesis has focused on three main techniques based on time series decomposition, as follows:

1. A *dynamic behavior capturing* technique is employed in this thesis to analyze time series data. It involves segmenting the time series into series of subsequences. Then, characteristics of each subsequence are compressed into statistical summary tuples, such as mean, variance, and slope, to capture specific details [22, 50]. Data binning, the traditional method used for this purpose, is utilized for filtering unwanted outliers and enhancing the quality of time series data. This method is well-established for analysis in astrophysics, as seen in [3, 29, 69]. Specifically, this thesis focuses on capturing dynamic behaviors in the trend component of astrophysical time series, as shown in Figures 2.2 (Top row).

2. A *season length estimation* technique focuses on extracting the main feature of the seasonal component, namely the *season length*. This length represents the periodicity of the repeating cycles within the time series data and is crucial for understanding periodic behaviors [8, 118, 120, 121]. This technique is particularly useful for analyzing time series data that include the seasonal component. Examples of such time series data are shown in Figures 2.2 (Middle row).

3. A *seasonal-trend decomposition* technique breaks down the time series into three components: trend, seasonal, and residual components [19, 45]. This technique is utilized for preprocessing time series that contain these three components, as shown in Figure 2.2 (Bottom row).

The following subsections explore the background, advantages, and ongoing challenges of three key techniques for time series decomposition: dynamic behavior capturing, season length estimation, and seasonal-trend decomposition. Each technique provides distinct data analysis advantages but presents particular challenges, for which we propose novel solutions in this thesis.

2.2.1 Dynamic Behavior Capturing using Data Binning

As discussed previously, we specifically focus on capturing dynamic behaviors in the trend component of time series by utilizing data binning.

Data binning reduces noise and removes unwanted outliers from the residual component of the time series. This process consists of two steps: segmenting the time series into a series of subsequences and compressing each subsequence into statistical summary tuples, commonly referred to as *bins*. These bins reflect distinct aspects of the data, capturing its essential characteristics. Figure 2.6 provides an illustrative example of each step.

In Figure 2.6 (Top), the time series is segmented into subsequences, each of which is defined by a subsequence length termed 'bin size' in data binning. Notably, all bin sizes are fixed at 20 instances, highlighted with green lines. Each subsequence is then compressed into a statistical summary tuple, specifically the mean value of each subsequence, which is visually represented by a red line in Figure 2.6 (Bottom). As a result at the red line, the 20 bins represent characteristics of the 400 instances from the original time series data. This compression effectively distills the characteristics of the original 400 instances into just 20 bins, succinctly capturing the essence of the data. This result, also known as the sketching

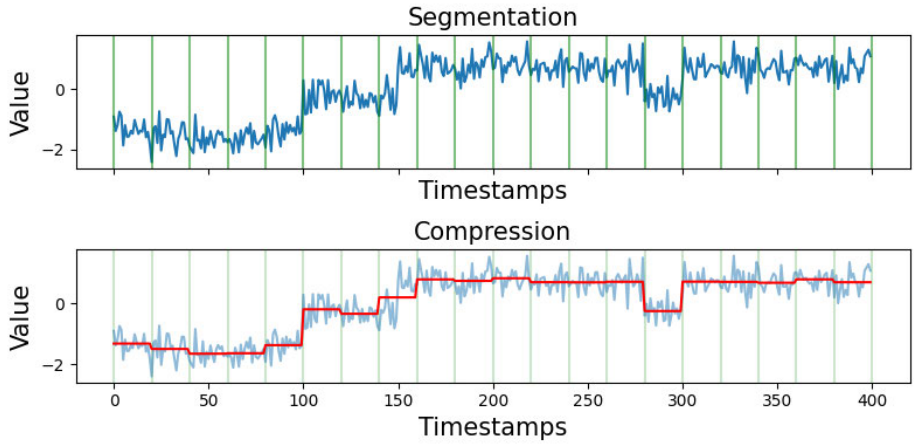


Figure 2.6: Data binning process in time series analysis: (Top) Segmentation of the time series with bin size of 20 instances (green lines); (Bottom) Compressed bins showing mean values (red lines).

result, efficiently summarizes the original time series data [21, 101].

Figure 2.6 (Bottom) shows how time series data can be simplified through data binning and presents the sketching result. This result demonstrates the characteristics of the original time series, including sudden changes in trends, effectively capturing its essential features.

Data binning is a valuable method in scenarios where overall trends are more important than external factors, such as noise or unwanted outliers. This is because random noise inherently has a mean value of zero, so the cumulative value of the noise tends to be zero. By averaging the original data within each bin, the signal-to-noise ratio is increased. Moreover, data binning not only filters noise but also minimizes memory usage and computational demands. As shown in Figure 2.6 (Bottom), it transforms 400 instances into 20 mean values from bins, enhancing memory usage efficiency.

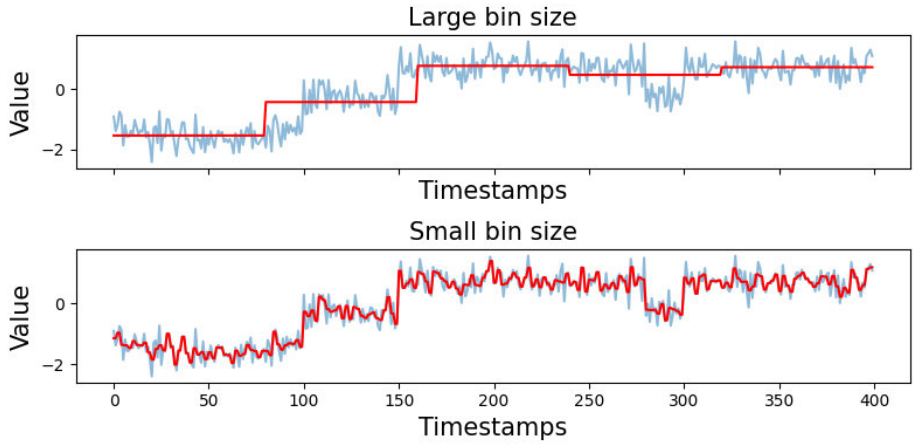


Figure 2.7: Influence of varying the bin size in data binning. (Top) Sample representation of data binning with a large bin size; (Bottom) Sample representation of data binning with a small bin size.

However, the quality of the sketching results is the most important aspect of utilizing data binning, with the bin size being a critical factor [85, 110]. This size significantly influences the sketching result: if too large, it may cause the loss of essential data characteristics, such as the disappearance of sudden changes observed between timestamps 270 to 300 in Figure 2.7 (Top). Conversely, if too small, the result may retain excessive noise, thus degrading its quality. Figure 2.7 illustrates how different bin sizes impact the sketching results, with the two red lines showcasing the variance caused by different bin sizes.

Scientists and analysts who use data binning must carefully define the bin size. If the bin size is incorrect, it can lead to erroneous analysis or decisions. To address bin size issues that affect the tradeoff between sketching quality and quality-degrading noise, we propose *Elastic Data Binning* (EBinning). This method dynamically adjusts the bin size in a nonparametric manner, compressing periods without significant signal through

zero-suppression while effectively capturing characteristic periods with bins. A detailed explanation of our proposed method is provided in Chapter 3 of this thesis.

2.2.2 Season Length Estimation (SLE)

The season length is a crucial feature of the season component for scientists and analysts to gain valuable insights into various phenomena. For example, season lengths can be analyzed in contexts such as sunspot cycle analysis, electrocardiogram signal interpretation, climate change monitoring, and industrial productivity cycles, thereby enhancing the understanding of these complex systems and their periodic behaviors [60, 103, 117, 127]. Moreover, many time series data mining algorithms require the season length as input parameter, such as ARIMA [12], seasonal-trend decomposition [19], and clustering techniques [122].

Accurately determining the season length is a complex task that requires a robust analytical approach due to the intricacy of temporal patterns in data. This is particularly important in cases that the cyclic behavior does not conform to standard intervals (e.g., daily, weekly, monthly), as mere observation and manual estimation may not suffice. For example, in the electrocardiogram dataset presented in Figure 2.5 (Top), we might estimate the season length to be 200 for each cycle based on perception. However, it is uncertain whether the actual season length varies within a $\pm 5\%$ range.

However, some time series data have timestamps in standard formats (e.g., calendar dates, months, years), but the season length may not align with standard intervals. To give the reader a better understanding, we showcase three time series datasets from various fields that highlight how season lengths may not align with standard intervals. These

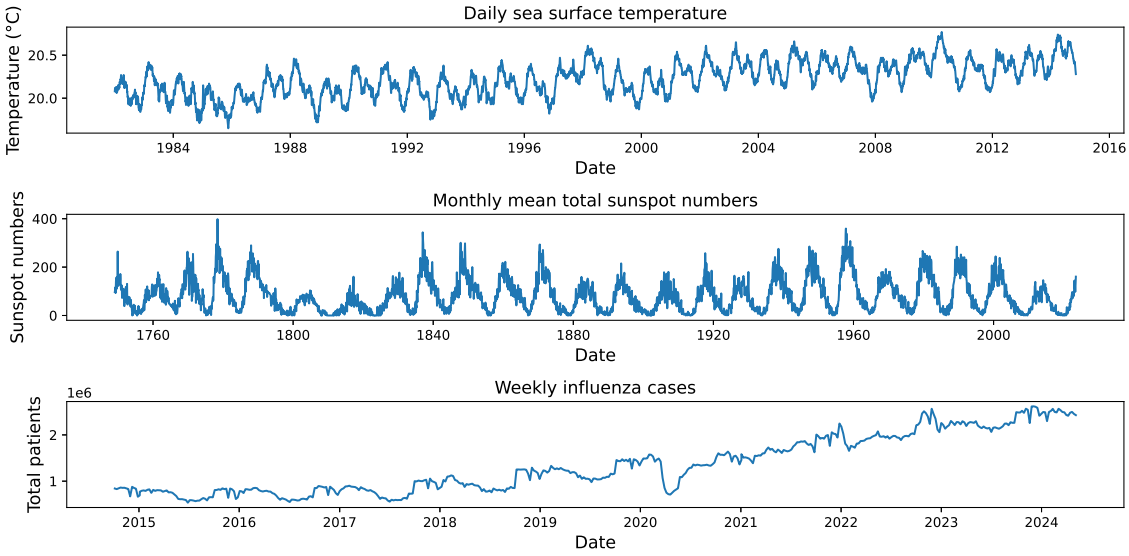


Figure 2.8: Three time series data, arranged from top to bottom as follows: daily SST, monthly mean sunspot number, and weekly influenza cases.

datasets include data from daily sea surface temperature (SST) [75], monthly mean sunspot numbers [112], and weekly influenza cases in the USA [46]. These data are shown in Figure 2.8.

From Figure 2.8 (Top), we can observe that each cycle of the SST data is unstable, especially during 1996-2000. This instability may bias users, but the ground truth in this case is that the season length equals 12, corresponding to seasonal variations over each year [98], as shown in Figure 1.2 (Bottom). Note that this instability is affected by dynamic behaviors from the trend component.

In the case of sunspot numbers in Figure 2.8 (Middle), we can clearly observe stable cycles. However, these cycles do not correspond to seasonal variations over each year, leading users to rely on human perception to determine the season length. The actual

season length for sunspot behavior is 11 years [112].

For weekly influenza cases in Figure 2.8 (Bottom), we observe a cyclical behavior before 2020. However, the cyclic behavior of each cycle after this year differs, which may affect users' ability to approximate the season length by human perception. The change in behavior around 2020 corresponds to the COVID-19 pandemic, during which the number of influenza cases decreased due to mitigation measures such as wearing face masks, staying home, and physical distancing, as reported by the National Center for Immunization and Respiratory Diseases [46].

From these examples, we observe that dynamic behavior in the trend component and unstable in the season component can bias human perception in identifying season lengths. Therefore, relying solely on human perception is not recommended, as it can be misleading and insufficient, especially in domains requiring specialized knowledge. Manual estimation may seem feasible, but it is prone to inaccuracies and biases, particularly when the data does not follow standard intervals like daily or monthly periods. It is preferable to use a technique that provides the precise season length, rather than relying on human perception. This task is accomplished by season length estimation (SLE).

SLE is an important technique that involves identifying the season length of repeating cycles within a time series [27, 121]. This process is important when dealing with datasets that exhibit irregular or unconventional cyclic patterns. It requires a nuanced and objective technique to uncover the actual underlying periodicity.

However, many traditional SLE methods operate offline mode, requiring access to the entire time series data beforehand. Examples of such offline SLE methods are presented

in [92, 118, 120, 127]. This condition limits their applicability to online mode for real-time analysis. To overcome this limitation, we propose a method called *Online Season Length Estimation* (OnlineSLE) for estimating season length in streaming time series data. OnlineSLE is a fast and accurate method, particularly suitable for online mode. Chapter 4 of this thesis provides a detailed explanation of our proposed method.

2.2.3 Seasonal-Trend Decomposition (STD)

STD breaks down the original time series into trend, seasonal, and residual components [8, 19, 45]. It is a useful technique widely utilized to support or preprocess for time series data mining tasks, such as anomaly detection, forecasting, real-time monitoring, and more [15, 39, 73, 74, 119, 115, 139].

The first method for STD, introduced by R. B. Cleveland *et al.*, is known as Seasonal-Trend decomposition using LOESS (STL)[1] [19]. LOESS stands for locally estimated scatterplot smoothing. STL is straightforward to implement and has become popular in various applications, including volcanology, climatology, and economics [15, 119, 133, 139, 137]. Here, we demonstrate the STL method with a monthly CO2 dataset from Figure 2.2 (bottom-right). Note that STL requires the season length as an input parameter before decomposition. We set the season length to 12, which corresponds to the annual cycle of CO2 levels. This reflects the seasonal variations over each year as reported in [57, 78]. This length captures the cyclic behavior observed in CO2 concentrations throughout the months of the year. The results of STL are shown in Figure 2.9.

[1]Details of STL are provided in Chapter 5.

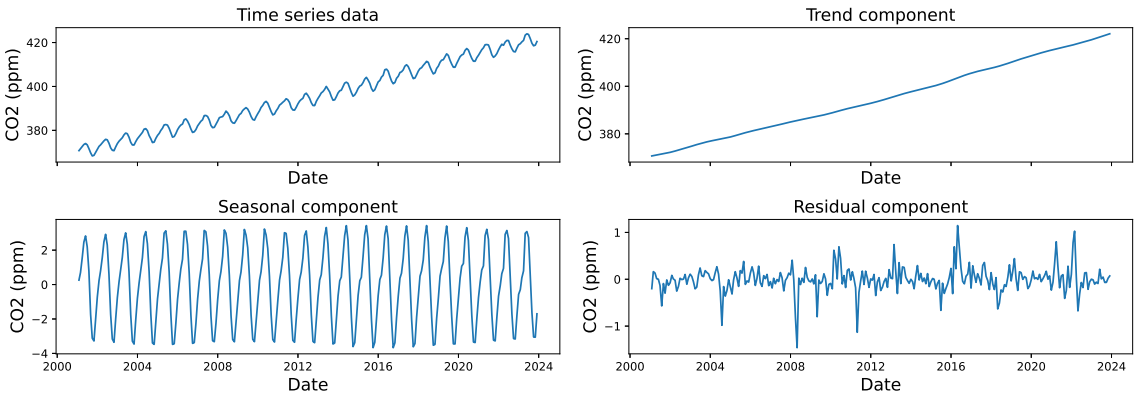


Figure 2.9: Seasonal-trend decomposition results of monthly CO2 from Figure 2.2 (Bottom-right), performed using STL with the season length of 12.

As a result, two key aspects can be quickly discerned: a slowly increasing long-term trend within the trend component and the stability of cyclic behavior within the seasonal component. These decomposition results can elucidate aspects of climate change driven by global warming, demonstrating that monthly variations do not significantly alter cyclic behavior [57, 78]. As previously mentioned, STD is a useful technique for analyzing time series in various applications. One important aspect of this technique is that it requires the season length as a specific input parameter. This parameter must be provided by users in advance and relies on domain-specific knowledge. To demonstrate why this parameter is important for the STD technique, we decompose the monthly CO2 data with STL using an improper season length of 7, which does not align with the annual cycle of CO2 levels. The results of STL is shown as Figure 2.10.

As shown in Figure 2.10, the decomposition results reveal a non-smooth trend and an anomalous cycle in the seasonal component during 2023. Based on this observation, one

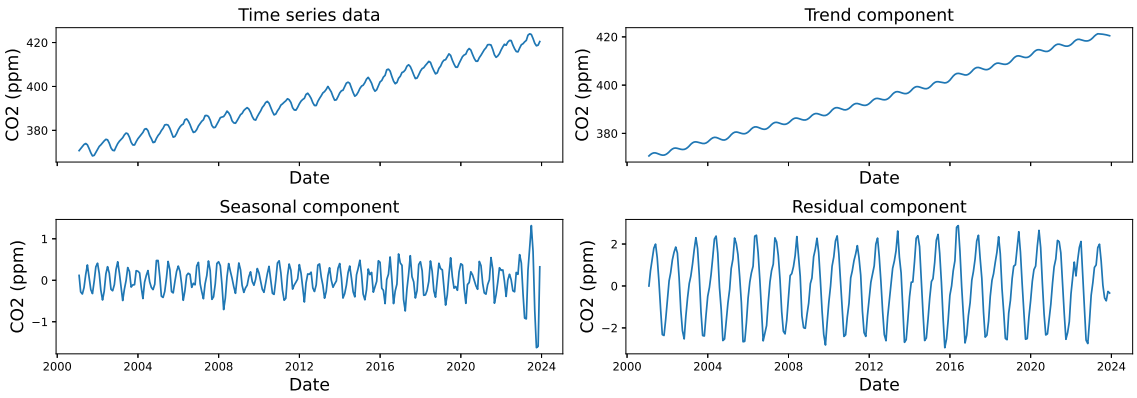


Figure 2.10: Seasonal-trend decomposition results of monthly CO2 from Figure 2.2 (Bottom-right), performed using STL with the season length of 7.

might mistakenly conclude that an anomaly occurred in the cycle during 2023. However, the actual behavior of CO2 indicates that there is no anomaly in the cyclic behavior, and the trend consistently shows a slow increase, as shown in Figure 2.9. These results align with those from NOAA [57]. Moreover, many real-world datasets may not align with standard intervals or contain dynamic behavior, making it difficult to identify season lengths solely by human perception, as discussed in Section 2.2.2. Consequently, using an improper season length can lead to incorrect analyses or decisions. Many STD methods, such as those cited in [5, 19, 35, 73, 126], require this length as an input and are influenced by the season length issue. This is one of the primary remaining challenges in STD.

The second challenge is that, similar to many SLE methods, most STD methods traditionally operate in offline mode, including [5, 19, 25, 126]. This limitation restricts their application to real-time decomposition or real-time analysis. Currently, only two methods, as cited in [35, 73], operate in online mode. However, they still require the season length as an input.

To address the influence of season length issues and to enable operation in online mode, we propose an *Adaptive Seasonal-Trend Decomposition* (ASTD) method that integrates our OnlineSLE into the decomposition process. A detailed explanation of our proposed method is provided in Chapter 5 of this thesis.

2.3 Discussion and Conclusion

This chapter has provided a detailed exploration of time series data and its decomposition into fundamental components: trend, seasonal, and residual. These components are often intertwined, adding complexity to the data and making it challenging to understand.

We introduced three key techniques for time series decomposition: data binning, season length estimation, and seasonal-trend decomposition. The utility and application of each technique were thoroughly examined, highlighting their importance in enhancing the analysis of complex time series data. These methods break down intricate data into more manageable and interpretable components, facilitating a deeper understanding of underlying trends and cycles.

As detailed in Section 2.2, these techniques still face numerous challenges that require further development and refinement. A key issue these techniques have in common is the influence of specified input parameters. These parameters, such as window size, season length, and threshold values, significantly affect the performance and accuracy of the methods. Determining these parameters typically requires prior knowledge or extensive trial and error, which is impractical in real-time analysis scenarios. In such scenarios, it is essential to employ methods capable of dynamic adaptation without requiring manual

parameter tuning. The inability to automatically adjust parameters to reflect the features of time series limits the applicability of these techniques for real-time analysis in streaming data, where behavior may change over time.

To address these issues, subsequent chapters will delve into our innovative methods, including EBinning in Chapter 3, OnlineSLE in Chapter 4, and ASTD in Chapter 5. Each method offers potential solutions to overcome the existing limitations of traditional decomposition techniques, promising more accurate and flexible analysis capabilities.

Chapter 3

Elastic Data Binning

This chapter explores the Elastic Binning (EBinning) method for capturing astronomical phenomena in the time-domain of astrophysics. EBinning includes two modes: Linear-EBinning and Mean-EBinning. Mean-EBinning utilizes Hoeffding's inequality to assess the distribution characteristics of data. On the other hand, Linear-EBinning utilizes the Student's t-test to assess linear relationships.

This chapter is organized into five sections for clarity and depth. Section 3.1 introduces the necessary background in time-domain astrophysics analysis (TDAA). Section 3.2 reviews related work, highlighting contributions and identifying gaps in the literature. Section 3.3 describes the EBinning methodology. Sections 3.4 and 3.5 detail the datasets and experimental settings, respectively. Section 3.6 presents our experimental results. Section 3.7 concludes with a discussion of findings and potential future research directions.

3.1 Background

TDAA is the study of astronomical objects and phenomena that exhibit variability over time, aiming to uncover new discoveries or deepen understanding of the universe [102]. Researchers in TDAA use time series data derived from various sources, including electromagnetic radiation and gravitational waves. Our focus is on analyzing *light curves* (LCs), which are one-dimensional time series representing the light intensity of celestial objects over time.

3.1.1 Light Curves

LCs graphically depict the brightness of a celestial object or area over time [123]. To obtain LCs from celestial objects or specific regions, astronomers produce LCs using the aperture photometry technique, which quantifies light intensity within a specified radius at each frame of a video sequence [56]. Figure 3.1 compares the video sequence with the derived LC. Notably, in frames 8-10, there is a marked increase and decrease in intensity, indicative of the brightness variations of an artificial celestial object[1].

LCs are alternative data to original video data as they provide detailed insights into the brightness and properties of astronomical objects. These LCs allow astronomers to identify features such as periodic behaviors in binary systems, trends in light intensity, and stellar pulsation. LCs also provide accurate, quantitative measurements of brightness over time, which allows for a more detailed analysis than what is possible with video data alone. The utility of LCs in TDAA is widely recognized. These LCs are extensively used to

[1]This celestial object is generated using Gaussian point spread function model from Astropy library.

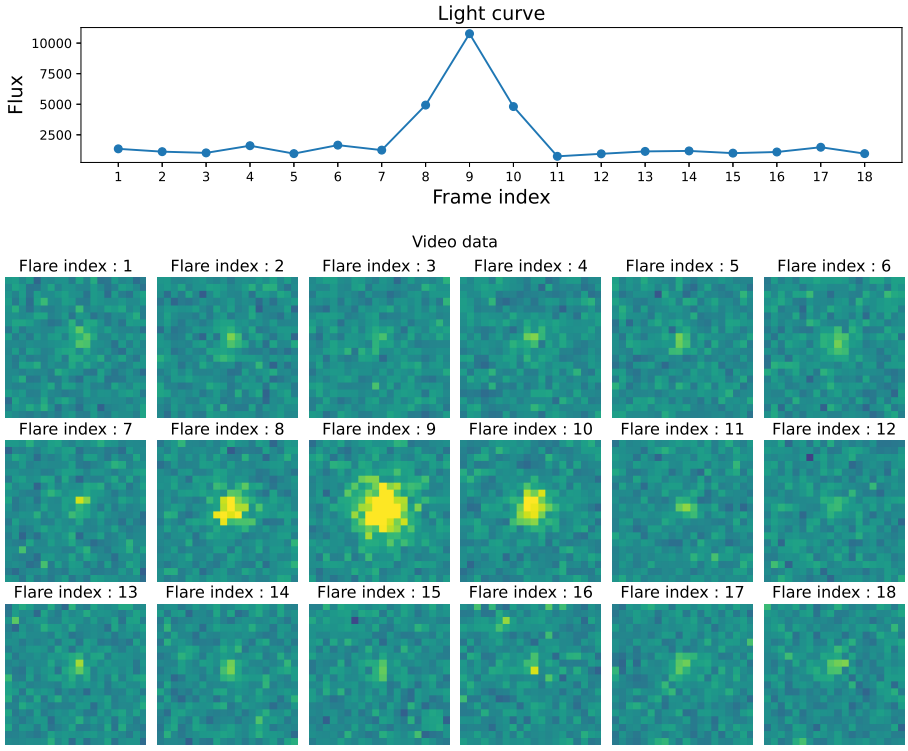


Figure 3.1: Comparison between the video for each frame and the LC that is derived from this video.

investigate transient phenomena in celestial objects, unfolding rapidly over periods ranging from seconds to minutes [3, 23, 72, 77, 81]. The following subsection will explore the specific astronomical phenomena that this thesis aims to capture and analyze.

3.1.2 Dynamic Behavior Captured in TDAA

As detailed in Section 2.2.1, our primary goal is to capture the dynamic behaviors within the trend components of astronomical data. We specifically focus on transient changes in LCs, which may correspond to stellar flares in real-world scenarios. We refer to these transient changes as *transient patterns*. Understanding these patterns is essential for

advancing knowledge and uncovering new insights into celestial dynamics.

Transient pattern analysis is a crucial task for TDAA, focusing on detecting and interpreting short-lived phenomena. These phenomena range from sudden rises to rapid decay in brightness, each providing critical insights into the dynamic nature of stellar objects. To effectively capture these fast transient phenomena, highly sensitive optical telescopes are utilized in observatories.

Although these telescopes are capable to capture such phenomena, they also accidentally capture unwanted outliers such as atmospheric turbulence, hardware measurement errors, or faint stars [2, 38]. These outliers can impact certain pixels in the video file, potentially leading to inaccuracies in the LCs. While generally uninteresting to astronomers, these outliers can cause erroneous interpretations if not properly identified and excluded. Consequently, we distinguish between unwanted outliers and transient patterns.

Unwanted Outliers

Unwanted Outliers in a LC typically manifest as single observation points that significantly deviate from the other data points [9, 135]. These outliers can impact the study of transient patterns either by pointing to real astronomical events or misleading researchers due to errors in data collection. Unwanted outliers in LCs may arise from various sources, including hardware measurement errors, satellite flares, and cosmic rays. To systematically identify and manage these outliers, we establish a formal definition as follows:

Definition 5 An unwanted outlier is an observation point Y_i at timestamp i in a LC,

which is a form of time series $Y = (Y_1, Y_2, \ldots, Y_t)$. An observation point Y_i is considered an outlier if it deviates from its expected value $E[Y_i]$ by more than a predetermined threshold θ. Formally, an outlier satisfies the condition $|Y_i - E[Y_i]| > \theta$.

Transient Patterns

Transient patterns in astronomical data are subsequences that exhibit significant, abrupt changes in brightness, identifying them as important phenomena for astrophysical studies [9]. These patterns frequently reflect dynamic events in the universe and are critical for understanding the physical processes underlying various astronomical phenomena. Examples of such transient patterns in astronomy include gamma-ray bursts, flares, and supernovae. Below, we provide the formal definition of the transient patterns that this thesis aims to capture:

Definition 6 A transient pattern is defined as a subsequence of points $Y_{(i,\,m)}$ from a LC, of subsequence length m, starting from timestamp i ($Y_{(i,\,m)} = (Y_i, Y_{i+1}, \ldots, Y_{i+m-1})$). This subsequence is considered transient pattern if it deviates from its expected behavior ($E[Y_{(i,\,m)}]$) by more than a predetermined threshold θ. Formally, a transient pattern satisfies the condition $d(Y_{(i,\,m)}, E[Y_{(i,\,m)}]) > \theta$, where d denotes the dissimilarity metric between two subsequences.

For better understanding, Figure 3.2 shows two LCs for comparison of transient patterns and unwanted outliers. The y-axis represents the brightness of celestial objects, and the x-axis represents the timestamp in modified Julian dates (MJD) units. Figure 3.2 (Left) depicts a sample unwanted outlier, identified by a single data point that sharply

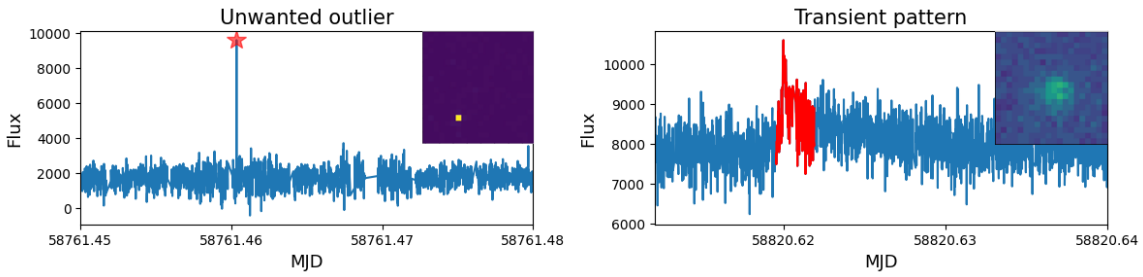


Figure 3.2: Comparison of an unwanted outlier (Left) and a transient pattern (Right) in LCs. The insets in the top-right corners show detailed cut-outs of the images captured at the moments the signals appear, highlighting the sources of the variations.

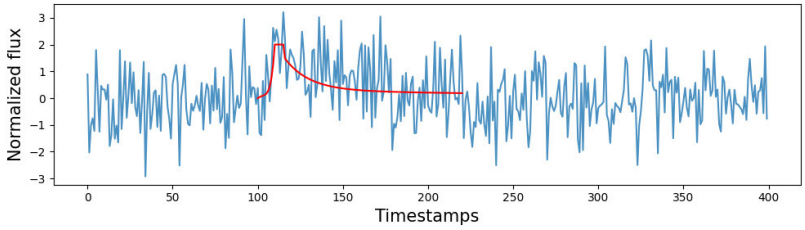


Figure 3.3: Example of an artificial flare, indicated by the red line, overlaid on the normalized flux of a celestial object depicted by the blue line.

deviates from the surrounding values. Conversely, Figure 3.2 (Right) depicts a sample transient pattern that is referred to as a fast optical flare from a star [3].

In this thesis, we focus on transient patterns that are characterized by three distinct phases: a sudden and intense occurrence, an observational peak, and a decay back to normal conditions. These patterns are commonly referred to as *flares* [3, 23]. Figure 3.3 shows an artificial flare, which is based on a model proposed by M. Aizawa *et al.* [3].

To address the challenge of distinguishing transient patterns from the unwanted outliers, astronomers often utilize a data binning method. This method is a critical prepro-

cessing step to filter out unwanted outliers and enhance LCs for accurate transient pattern analysis [29].

3.1.3 Data Binning in TDAA

Building upon the fundamental concept of data binning introduced in Section 2.2.1, this section delves into detailed methods specifically for TDAA. Data binning is crucial to enhance LCs for better analysis of the transient patterns. By segmenting a time series into subsequences and summarizing each with representative statistics, we can reduce the impact of noise and extract the underlying transient patterns for capturing phenomena. To facilitate a clear understanding of the methodologies discussed, it is essential to define the specific notation and terms used throughout this section.

Notations

An LC ($Y = (Y_1, Y_2, \ldots, Y_t)$) is a time series representing light intensity, which our LCs do not include a seasonal component confirmed by astronomer. We segment Y into a series of subsequences $Y = (Y_{(1,m_1)}, Y_{(m_1+1,m_2)}, \ldots, Y_{(m_{q-1}+1,m_q)})$ (as defined in Definition 4 in Section 2.1.3). Each subsequence is then compressed into a *bin*.

Definition 7 A *bin* is a statistical summary represented by a tuple derived from a subsequence $Y_{(i,m)}$ of an LC. This subsequence starts at timestamp i and has a length m, known as the bin size. The tuple is denoted as $\{m, f, st\}$, where f contains statistical properties of the subsequence, such as mean, variance, and slope, and st includes auxiliary variables necessary for updating the bin. The composition of f and st may vary depending on the

data binning method used, with detailed definitions provided in subsequent sections. This tuple summarizes the key characteristics of each bin.

The series of subsequences of an LC is alternatively described by a series of bins as $(bin_1, \ldots, bin_q)$. This series is maintained in a *window* ($\mathcal{W}$).

Definition 8 A *window* ($\mathcal{W}$), denoted as $\mathcal{W} = (bin_1, \ldots, bin_q)$, where q $(1 \leq q \leq t)$ denotes the window size. The value of q is determined by memory resource limitations or user-defined.

As discussed in Section 2.2.1, an important parameter in data binning is the bin size. Setting this parameter too small may lead to results that include excessive noise, while setting it too large may distort the main features of the LC, as demonstrated in Figure 2.7. Using a fixed bin size for all bins often proves inadequate for capturing transient patterns. One potential solution is to allow users to define the bin size for each bin manually. However, this approach can be difficult and time-consuming for excluding irrelevant information and capturing only significant features.

To tackle these challenges, we propose the Elastic Data Binning method (EBinning). This method dynamically adjusts the bin size for each bin based on the data characteristics, effectively reducing the inclusion of irrelevant information and capturing only significant features. This adaptive approach helps achieve an optimal balance between reducing unwanted outliers and preserving the essential details of the LC. Following this introduction, a description of the related work is given before the details of EBinning are provided.

3.2 Related Work

3.2.1 Data Sketching and Binning

Data sketching is a technique that has gained significant attention in time series analysis due to its ability to approximate and reduce the dimensionality of time series data. This method transforms the original time series, which has a length of t, into a lower-dimensional representation of length q, where q is much smaller than t ($q << t$). This transformation ensures that the essential characteristics are retained while reducing the dataset to a lower-dimensional form. Numerous studies have explored data sketching [21, 28, 63, 64, 101].

Data binning is related to data sketching and can be considered a form of sketching that summaries data into bins[29, 50]. Each bin represents a mean value of the data points within that bin, which f of each bin is the mean value. This technique reduces the noise and variability in the data, enhancing the visibility of trends and patterns in the time series. Note that data binning can also be referred to *piecewise aggregate approximation* (PAA) [50].

To reduce dimensionality, one can utilize the *symbolic aggregate approximation* (SAX) [62, 63]. SAX is an extension of the PAA method by transforming the mean values of each bin into a string of symbols. This method employs a quantile-based approach to symbolizing the mean values of bins. By classifying the mean values into quantiles, each bin is assigned a symbol that reflects its statistical significance within the overall distribution of the dataset. Recently, an enhancement known as *1D-SAX* [71] has been introduced, which incorporates slope values into the symbolic representation for more detailed analysis.

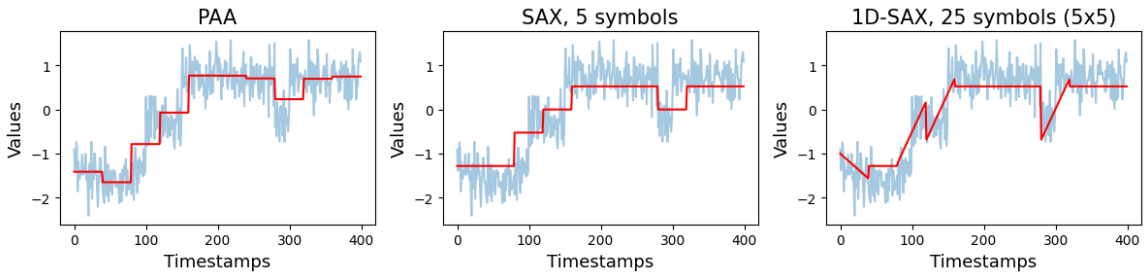


Figure 3.4: Comparative results between PAA, SAX, and 1D-SAX. The original time series is depicted in blue, while the results from each method are shown in red.

Several studies have refined SAX, resulting in variants such as ESAX [68], SAX-TD [111], and SAX-BD [36], each improving different aspects of the original method. The comparison of sketching results using various methods, including data binning, SAX, and 1D-SAX, show as Figure 3.4. However, both SAX and 1D-SAX require additional parameters that can impact the quality of the sketching results. Therefore, our EBinning approach does not quantize results into symbols.

3.2.2 Time Series Analysis for TDAA

Time series analysis for TDAA includes various applications and methods designed to specific needs and phenomena. This subsection discusses the foundational role of data binning and other methods used in TDAA. Data binning is a widely adopted technique commonly used by astronomers. It is available in the Astropy library, a useful Python library for astronomy [29]. There are many research that enhance data binning [69, 88, 89, 106, 110].

As previously discussed, the choice of bin size significantly impacts the quality of sketching results, balancing between reducing noise and preserving the main features

of the original data. Manually selecting an optimal bin size can be challenging and time-consuming. To address this, R. Sulo, T. Berger-Wolf, and R. Grossman proposed an automated framework known as *temporal window in network* (TWIN), which aims to achieve this balance [110]. This framework provides a straightforward method to determine the optimal bin size. However, it relies on an iterative search process to find this size, which can be time-consuming, as reported in our technical report [85].

One notable example is the application of data binning for Lunar Laser Ranging data as proposed by Shevlyakov and Kan in the *SK-method* [106]. This method identifies sudden change points in LCs using the Chebyshev inequality. However, it employs a traditional approach where bin sizes are equal and fixed in advance. Setting improper bin sizes in this method can lead to incorrect results.

Turning to the concept of dynamic bin sizing in data binning, *Adaptive-binning* for TDAA with LCs was proposed by B. Lott *et al.* [69]. This method was specifically used to explore blazar phenomena with LCs from the Fermi Large Area Telescope. *Adaptive-binning* adjusts the bin size dynamically, using the correlation between the source photon spectral index and the gamma-ray flux to determine the appropriate bin size for each segment. This method is designed for specific applications for exploring blazar phenomena. In contrast, our approach is designed to handle unknown phenomena without relying on predefined assumptions, thereby differing fundamentally in objective and application from Adaptive-binning.

Building on our previously established Dynamic Data Binning (DyBin) framework [87, 88, 89], we have enhanced our approach with an improved 'bin-merge' technique. This

strategy methodically merges neighboring bins when their characteristics are deemed similar, as determined by a measured similarity measure grounded in the t-test score comparison of their mean values. While DyBin showed promise for detecting transient patterns, it was primarily designed for detection and did not focus on extracting or capturing the nuanced characteristics of these patterns. EBinning, as a refined iteration of this concept, extends beyond mere detection. It aims to precisely delineate and capture the transient phenomena, thereby enhancing our ability to understand and analyze these fleeting patterns.

In the realm of TDAA, researchers have explored a multitude of analytical techniques that diverge from the data binning technique. For example, U. Rebbapragada *et al.* introduced PCAD, an unsupervised anomaly detection method specifically designed for LCs that exhibit three types of transient patterns [96]. This method adapts the k-means clustering algorithm for specialized use in time-domain astrophysics, with the primary goal of categorizing different transient patterns within LCs to enhance understanding through pattern clustering. In contrast, our research aims to refine the data binning process itself, focusing on the extraction and sketching of LCs to serve as a foundation for detecting transient patterns from these reduced representations.

The application of ARIMA models for TDAA was explored by E.D. Feigelson, G.J. Babu, and G.A. Caceres, who focused on analyzing the periodic behavior of celestial objects through LCs [30]. Their research specifically addressed LCs that contain seasonal components. Unlike their study, which assumes the presence of the seasonal component in LCs, our examined LCs have been verified by astronomers to lack the seasonal component. Therefore, our objectives differ from their study. Notably, various methods are available for

analyzing periodic behavior in LCs for TDAA, such as the frequency domain analysis and other techniques found in the RobPer library [116].

The discord discovery was introduced by E. Keogh *et al.* [51]. It is employed to identify the most dissimilar (or unusual) subsequence within a given time series, compared to other subsequences. Note that the opposite of discord discovery is motif discovery, which focuses on finding the most similar subsequences. In their study [51], E. Keogh *et al.* proposed Heuristically Ordered Time Series using Symbolic Aggregate Approximation (HOT-SAX), which integrates SAX with discord discovery. Despite HOT-SAX's applications in various fields, its application to LCs has not been extensively explored. We have conducted an evaluation of HOT-SAX specifically for transient pattern analysis in LCs and compared its effectiveness with our EBinning method, aiming to provide a detailed comparative analysis and demonstrate the adaptability of HOT-SAX to this specific domain.

Another technique that is highly regarded in the field of time-series data mining is the Matrix Profile (MP). It has been employed in various applications such as motif and discord discovery, anomaly detection, and data segmentation [134, 140, 141]. It efficiently encapsulates subsequences in a data structure that comprises a distance profile and an index, requiring only the subsequence length as a user-defined parameter. Recently, Mplots have emerged as a novel method of MP for exoplanet detection within LCs through motif discovery [82, 105]. Our method distinguishes itself by focusing on the identification of previously unknown astronomical phenomena, thereby highlighting a unique aspect of the utility of Mplots method.

To summarize, these studies in the literature highlight the research direction and

the remaining challenges. We will now delve into our EBinning method in the following section.

3.3 Elastic Data Binning

We introduce a baesline method of EBinning that automatically determines the appropriate bin size (m) for each bin. The key concept involves measuring the potential for merging two neighboring bins within a sequence in a window ($\mathcal{W}$). This potential is quantified using a metric known as the *mergeability score* (k). We design this metric such that a low value indicates similar features between two neighboring bins, allowing them to be merged into a new bin. Conversely, a high value suggests that the neighboring bins have distinct features, indicating potential changes in behavior during the corresponding period. This foundational concept is the baseline EBinning method, detailed in Algorithm 1.

The input parameters for EBinning include the time series Y, the initial bin size m, and the window size q. Here, Y corresponds to an input LC. In this thesis, we set m_{ini} to 8 instances to ensure that each bin initially contains data from a single distribution, which is achieved by keeping m_{ini} small size. Based on experimental results reported in [85], we adjust q to accommodate 20 bins within the window $\mathcal{W}$.

Initially, EBinning creates three empty arrays: a buffer $\mathcal{B}$, a window $\mathcal{W}$, and score profile $\mathcal{S}$. $\mathcal{B}$ stores subsequences for initializing bin creation. $\mathcal{W}$ stores a series of bins (as defined in Definition 8). $\mathcal{S}$ stores the mergeability scores for each pair of neighboring bins within $\mathcal{W}$.

It fills $\mathcal{B}$ with incoming data Y_i until $\mathcal{B}$ is full. Once $\mathcal{B}$ is full, it creates the initial

Algorithm 1: Baseline of EBinning

Input: Time series $Y = (Y_1, Y_2, \ldots, Y_t)$, initial bin size m_{ini}, window size q

Output: Window $\mathcal{W} = \{bin_1, \ldots, bin_p, \ldots, bin_q\}$)

```
1  W ← ∅, S ← ∅, B ← ∅
2  for i ← 1 to t do
3      B.append(Y_i)
4      if B is full then
5          W.append(Initialize(B))
6          S.append(ComputeScore(bin_q, bin_{q-1}))
7          if W is full then
8              p ← argmin_p(S[2 :])
9              bin_new ← Merge(bin_p, bin_{p-1})
10             Remove bin_p from W
11             Replace bin_{p-1} in W with bin_new
12             Update(S)
```

bin using the `initialize` function and append this bin into $\mathcal{W}$ (Line 5). After initialization, we compute the mergeability score k between two neighboring bins, bin_q and bin_{q-1}, using the `ComputeScore` function and append it to $\mathcal{S}$ (Line 6). Note that if $\mathcal{W}$ contains only one bin, k must be zero.

When $\mathcal{W}$ is full, it finds the index where the bin has the minimum k value using the `argmin` function (Line 8). Note that we add one to the index because the first index of $\mathcal{S}$ is zero, as bin_1 does not have a prior bin. After this step, it merges two neighboring bins into a new bin and update $\mathcal{W}$. When two neighboring bins are merged, only the two k values of the neighboring bins for the merged bin in $\mathcal{S}$ must be updated. From this based line, we proposed EBinning with two methods: Mean- and Linear-EBinning.

3.3.1 Mean-EBinning (M-EBinning)

We assume that the bin corresponding to $Y_{(i,\,m)}$ is a normal situation without any occurrence of events, where each value within $Y_{(i,\,m)}$ is random and independent. In contrast, bins that contain transient patterns exhibit distributions that differ from those of neighboring bins, which are assumed to be in a normal situation. Consequently, the mergeability score focuses on the similarity of distributions between two neighboring bins. If two bins are derived from the same distribution, their means are expected to fall within a bounded range. The method that utilizes the mergeability score based on this concept is called *Mean-EBinning* (M-EBinning).

Initialization function

In M-EBinning, the initialization function processes the subsequences $Y_{(i,\,m)}$ within $\mathcal{B}$ into an initial bin, formatted as $\{m, f, st\}$. Here, f is defined as the tuple $\{min, max, \mu\}$, where min and max denote the minimum and maximum values of the subsequence, respectively, and μ denotes the mean corresponding to the original subsequence. The st component is left empty because auxiliary variables are not stored.

Then, it utilizes the values in f to compute the boundary ε based on Hoeffding's inequality [31, 40]. ε can be expressed in terms of δ as follows:

$$\varepsilon = \sqrt{\frac{c^2}{2m} log \frac{2}{\delta}} \tag{3.1}$$

where c denotes the difference between the minimum and maximum values (max-min), and m is the bin size of bin, δ is the user-defined error probability. This boundary ε is used to define a range around μ given by $[\mu-\varepsilon, \mu+\varepsilon]$. It is used to determine if neighboring bins should be merged based on the similarity of their intervals, thereby assessing whether they deviate from this boundary.

Score Computation Function

We assume that bin_p and bin_{p-1} belong to the same distribution and can be merged without distorting the original features. The two boundaries of these neighboring bins are defined by $[\mu_p - \epsilon_p, \mu_p + \epsilon_p]$ and $[\mu_{p-1} - \epsilon_{p-1}, \mu_{p-1} + \epsilon_{p-1}]$, respectively. Both boundaries fall within the error probability $delta$, ensuring that their merge maintains the integrity of the data distribution (See Figure 3.5).

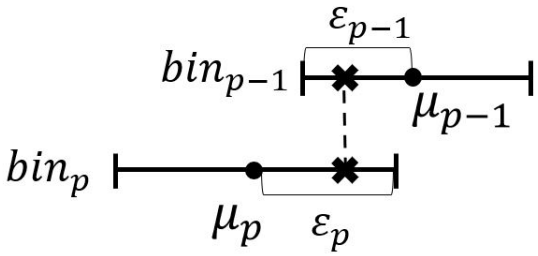


Figure 3.5: Two boundaries in the same distribution.

Based on this observation, we determine the mergeability of bin_p and bin_{p-1} using the following formula:

$$k = \frac{|\mu_p - \mu_{p-1}|}{\texttt{min}(\varepsilon_p, \varepsilon_{p-1})} \tag{3.2}$$

where function `min` selects the smaller value between ε_p and ε_{p-1}. When k is closest to zero, it indicates that the neighboring bins can be merged, as their distributions are similar. Conversely, when k is farthest from zero, it suggests that the neighboring bins should not be merged, due to dissimilar distributions. Notably, a limitation of the M-EBinning method arises from Eq. 3.2 in that ε must not be zero to avoid a divide-by-zero issue. This condition occurs when the minimum and maximum values within a bin are the same, resulting in a zero range and consequently, a zero error bound, ε. In such cases, the distribution cannot be assessed.

Proof. This proof elucidates the concept of the mergeability score as detailed in Eq. (3.2) for checking if two bounds overlap or not. Given two neighboring bins indexed p and $p-1$ in the window, we consider k to measure the potential for merging between two neighboring bins. The boundaries for for bin_p and bin_{p-1} denote as as $[\mu_p - \epsilon_p, \mu_p + \epsilon_p]$ and

$[\mu_{p-1} - \epsilon_{p-1}, \mu_{p-1} + \epsilon_{p-1}]$, respectively.

We give the overlap condition that there must be some common range between them. Mathematically, this condition can be checked by ensuring that the distance between their means is less than or equal to the sum of their error bounds. This can be expressed as:

$$|\mu_p - \mu_{p-1}| \leq \varepsilon_p + \varepsilon_{p-1} \tag{3.3}$$

Alternatively, the intervals overlap if:

$$|\mu_p - \mu_{p-1}| \leq \min(\varepsilon_p, \varepsilon_{p-1}) \tag{3.4}$$

This condition is derived from considering that ε_p and ε_{p-1} represent bounds, and in the worst case, one bound will dominate. Therefore, we use the minimum to be conservative in our overlap check.

By normalizing the distance between the means by the minimum of the error bounds, we get:

$$k = \frac{|\mu_p - \mu_{p-1}|}{\min(\varepsilon_p, \varepsilon_{p-1})} \tag{3.5}$$

If k is closest to zero, it implies that the distance between the means is within one error bound (the tighter bound of the two), suggesting that the two bounds belong to the same distribution and can be merged.

Merging Function

When bin_{p-1} and bin_p are merge, they form bin_{new}. bin_{p-1} and bin_p correspond to original subsequences of LC $Y_{(i_{p-1},m_{p-1})}$ and $Y_{(m_{p-1}+1,m_p)}$, respectively. The values of bin_{new} are calculated as follows:

$$m_{new} = m_p + m_{p-1} \tag{3.6a}$$

$$\mu_{new} = (m_p\mu_p + m_{p-1}\mu_{p-1})/(m_p + m_{p-1}) \tag{3.6b}$$

$$min_{new} = \texttt{min}(min_p, min_{p-1}) \tag{3.6c}$$

$$max_{new} = \texttt{max}(max_p, max_{p-1}) \tag{3.6d}$$

3.3.2 Linear-EBinning (L-EBinning)

The characterization of transient patterns includes a rapid rise and a slow decay. Therefore, we propose an alternative method of EBinning that focuses on the similarity of slope values between two neighboring bins. The slope value for each bin is computed using linear regression, enabling the analysis to address the dynamic behavior characteristic of transient patterns specifically. The second method of EBinning is called *Linear-EBinning* (L-EBinning).

Initialization Function

The initialization function summarizes the subsequences ($Y_{(i,m)}$) within $\mathcal{B}$ into an initial bin, represented in the form $\{m, f, st\}$. Here, f is defined as the tuple $\{\alpha, \beta\}$, where α is the intercept and β is the slope, both derived from linear regression analysis. To

facilitate fast computation during the merging of two neighboring bins, st is defined as the tuple $\{\mu, \sigma^2, \sum_{n=i}^{i+m-1} n, \sum_{n=i}^{i+m-1} n^2, \sum_{n=i}^{i+m-1} nY_n\}$, which includes the mean μ, variance σ^2, and sums of time indices and their products with data points, essential for quick updates. α and β that based on linear regression are defined as:

In the regression analysis, $\overline{n}$ represents the mean time index for $Y_{(i,m)}$, calculated as $(2i + m - 1)/2$, which is the midpoint of the time indices. The values of α and β are then defined by the following linear regression formulas:

$$\text{Intercept } \alpha = \mu - \beta\overline{n}, \tag{3.7a}$$

$$\text{Slope } \beta = \frac{\sum_{n=i}^{i+m-1}(n - \overline{n})(Y_n - \mu)}{\sum_{n=i}^{i+m-1}(n - \overline{n})^2}. \tag{3.7b}$$

Score Computation Function

For L-EBinning, the mergeability score k is based on the Student's t-test for testing the equality of slopes between two regression lines from neighboring bins [4]. k is calculated in two ways: by testing whether the variances of two neighboring bins suggest that they are from the same distribution or not. This is because we need to avoid the prone to introduce some statistical bias as discussed in [4, 124]. Therefore, k between two neighboring bins bin_p and bin_{p-1} is defined as follows:

$$k = \begin{cases} \frac{|\beta_p - \beta_{p-1}|}{\sqrt{(\sigma_p^2/m_p) + (\sigma_{p-1}^2/m_{p-1})}} & F_{cal} \leq F_{\theta/2;m_p-1;m_{p-1}-1} \\ \frac{|\beta_p - \beta_{p-1}|}{\sqrt{[(\sigma_p^2+\sigma_{p-1}^2)/(m_p+m_{p-1}-2)][(1/m_p)+(1/m_{p-1})]}} & F_{cal} > F_{\theta/2;m_p-1;m_{p-1}-1} \end{cases} \tag{3.8}$$

In Equation 3.8, F_{cal} denote the calculated F-statistic to compare the variances of the two regression slopes and is defined as:

$$F_{cal} = \frac{\sigma^2_{large}}{\sigma^2_{small}} \tag{3.9}$$

where σ^2_{large} (*resp.* σ^2_{small}) is the larger (*resp.* smaller) variance of either bin_{p-1} or bin_p. Here, $F_{\theta/2;m_p-1;m_{p-1}-1}$ is the critical value of the F-distribution corresponding to a significance level of θ, and degrees of freedom $m_{p-1}-1$ and m_p-1. Notably, significance level for F-test is set a 5-percent significance level that regarded as a convention for F-test as discussed in [124].

Algorithm 2: `ComputeScore` function of Linear-EBinning

Input: bin_p, bin_{p-1}, θ

Output: k

```
F ← calculate the F-test by Eq. 3.9
if F ≤ F_{θ/2;m_p−1;m_{p−1}−1} then
    k ← calculate score by Eq. 3.8 (Upper)
else
    k ← calculate score by Eq. 3.8 (Lower)
return k
```

Merging Function

When bin_{p-1} and bin_p are merge, they form bin_{new}. bin_{p-1} and bin_pcorrespond to original subsequences of LC $Y_{(i_{p-1},m_{p-1})}$ and $Y_{(m_{p-1}+1,m_p)}$, respectively. The values of bin_{new} are calculated as follows:

$$m_{new} = m_p + m_{p-1} \tag{3.10a}$$

$$\beta_{new} = \sum_{n=i_{p-1}}^{n_{end}} (n - \bar{n})(Y_n - \mu_{new}) / \sum_{n=i_{p-1}}^{n_{end}} (n - \bar{n})^2 \tag{3.10b}$$

$$\alpha_{new} = \mu_{new} - \bar{t}\beta_{new} \tag{3.10c}$$

where $n_{end} = i_{p-1} + m_{new} - 1$ represents the endpoint of the summation interval for the new bin. $\overline{n}$ is the median timestamp, defined as $(2i_{p-1} + m_{new} - 1)/2$, and μ_{new} is detailed in Eq. (3.6d). Access to the values of the original subsequences is not possible once the subsequences are compressed into bins. Therefore, EBinning utilizes auxiliary variables *st* to compute the new values for bin_{new}. The methodologies for computing these values using *st* are detailed in the original supplementary document [90] of EBinning publication [91].

3.4 Light Curve Datasets

The experiments are conducted using LCs derived from video observations captured by the Tomo-e Gozen, a wide-field complementary metal oxide-semiconductor (CMOS) camera mounted on the 105cm Schmidt telescope at Kiso Observatory in Nagano, Japan [99]. This camera system is specifically designed to survey and detect short timescale transient patterns, such as those lasting only one second.

We have a total of 108 LCs confirmed by astronomers that no transient patterns are included (hereafter referred to as Normal LCs). Normal LCs exhibit several essential characteristics, which can be categorized into four distinct scenarios encountered during

the sky survey. These scenarios are illustrated in Figure 3.6, with a comparative analysis provided below. Each scenario is described in detail in the subsequent sections:

- Stable behavior: The light intensity remains constant within the expected noise range throughout the period, as illustrated in Figure 3.6 (Top-left).
- Unstable behavior: The light intensity experiences sudden changes during the period. This behavior may be caused by variations in atmospheric opacity or contamination from the brightness of nearby stars or faint stars, as illustrated in Figure 3.6 (Top-right).
- Outliers: This represents a single event that may be caused by instrumental noise or cosmic ray events, as shown in Figure 3.6 (Bottom-left).
- Gradual change: This scenario may exhibit a transition to a new normal state, as shown in Figure 3.6 (Bottom-right). This scenario is not considered a target phenomenon for our study because it lacks the distinct peak and decay phases typical of the transient pattern (as explain in Section 3.1.2).

In this study, two distinct datasets were employed: synthetic and real-world datasets. The synthetic dataset was created by injected artifact transient patterns into normal LCs. Conversely, the real-world dataset includes LCs that contain real flares, which are astronomical transient patterns.

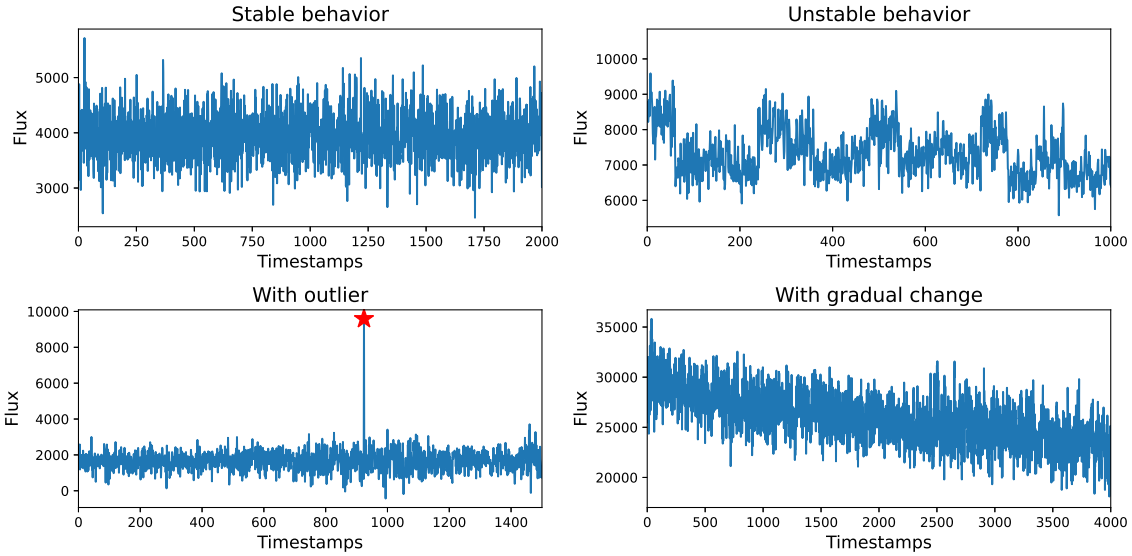


Figure 3.6: Four scenarios in the LC.

3.4.1 Synthetic datasets

We considered three types of transient patterns for our synthetic datasets: square, triangle, and Kepler patterns. For square and triangle patterns, we varied the durations and heights with combinations of 60, 100, 200, and 500 instances for durations and 1σ and 3σ for heights, where σ denotes the standard deviation of the original data for each Normal LC. This approach yielded 16 unique combinations. Consequently, we generated a total of 7,616 LCs, each containing a single transient pattern, either square or triangle. These LCs are subsequently referred to as Square-LCs and Triangle-LCs, respectively, according to the shape of the transient pattern they contain. The smallest and largest transient patterns are illustrated in Figure 3.7.

However, square and triangle transient patterns may not be realistic for astronomical phenomena. Therefore, we generated more transient patterns following the Kepler flare

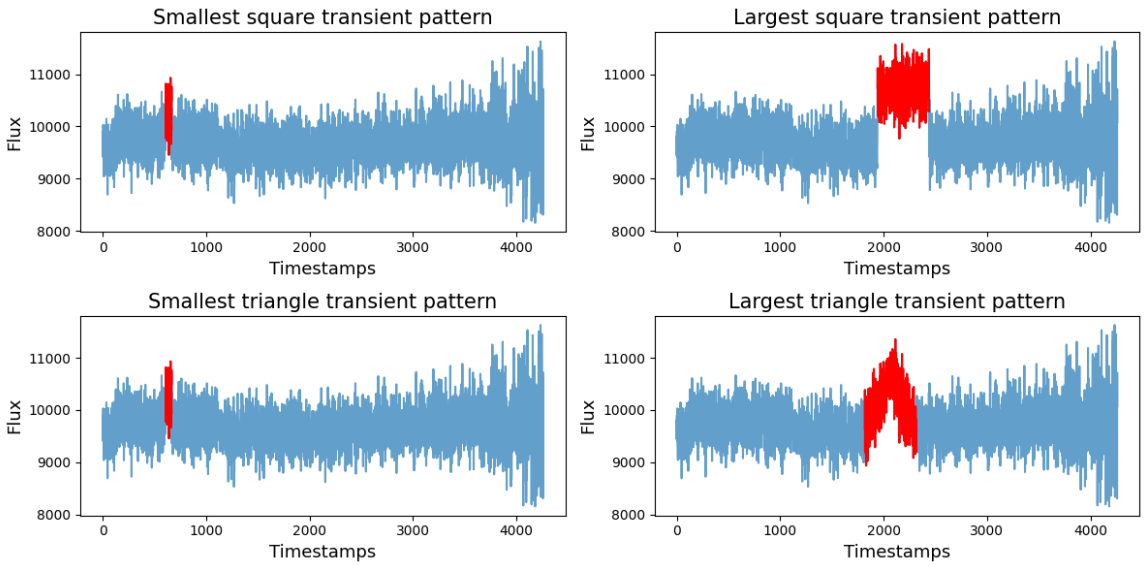


Figure 3.7: Comparison between the smallest and the largest transient pattern of Square- and Triangle-LCs.

model described in the literature [23]. The Kepler flare model is separated into two phases: the rising phase and the decay phase. The mathematical representation of each phase is as follows:

$$f_{\text{Kepler}}(t) = \begin{cases} 1 + 1.941t - 0.175t^2 - 2.246t^3 - 1.125t^4 & (-1 < t \leq 0) \\ 0.689\exp(-1.6t) + 0.303\exp(-0.2783t) & (0 < t < 6) \end{cases} \tag{3.11}$$

where t denotes the timestamp of the model. The time range is divided into 120 frames, with t spanning from -1 to 6. Each frame represents a discrete time step within this range.

These artificial flares were then injected into Normal LCs, with their intensities modulated to reflect variations at 1σ, 2σ, and 3σ heights. This process allowed us to

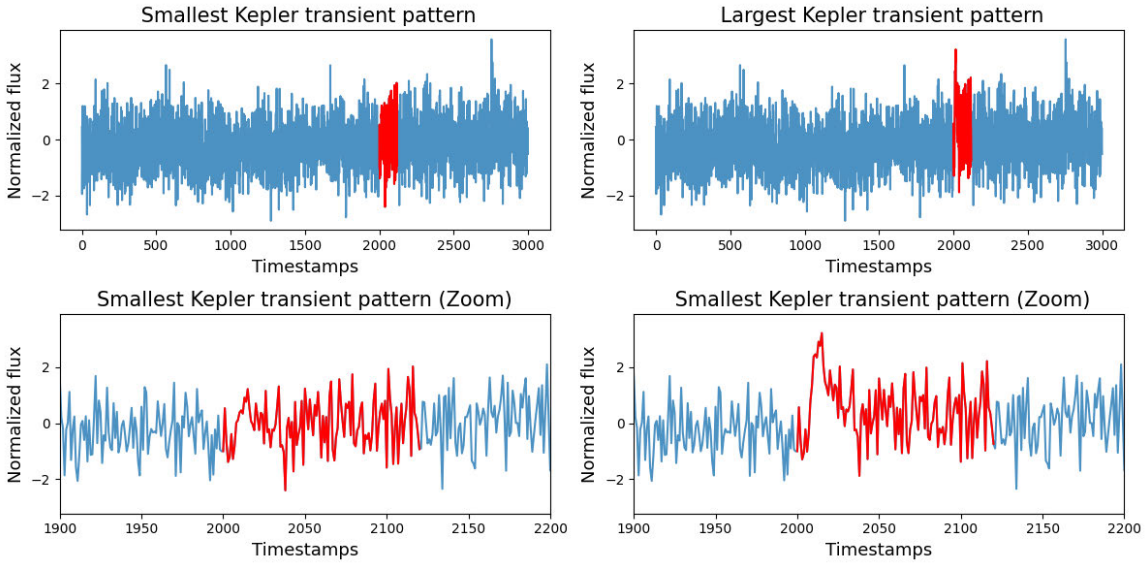
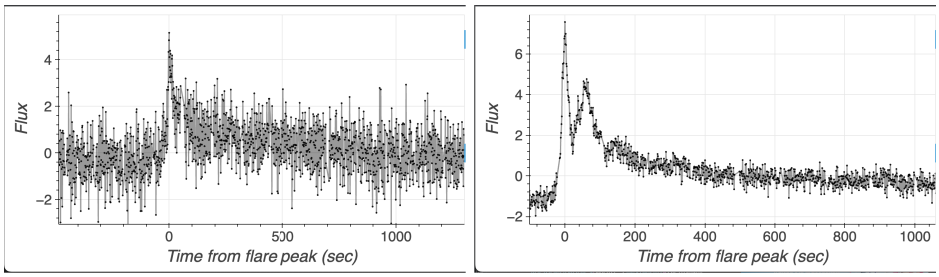


Figure 3.8: Comparison between the smallest and the largest Kepler transient pattern.

create a controlled environment to test the efficacy of our detection methods against known, quantifiable events. Hereafter, LCs including Kepler flare referred as Kepler-LCs. The smallest and largest transient patterns are illustrated in Figure 3.8.

3.4.2 Real world datasets

The real-world dataset used in this study originates from the research conducted by M. Aizawa *et al.* [3]. To confirm the presence of astronomical transient patterns, each LC was visually inspected against relevant image frames with list of astronomical catalogues. The LCs of nearby stars were also examined to evaluate potential influences from contamination and atmospheric turbulence. They successfully identified 18 stellar flares within the LCs from this meticulous process. These detected flares were categorized into two types based on their characteristics: classical and complex flares, as shown in



(a) Classical flare. (b) Complex flare.

Figure 3.9: Comparison between classical and complex flares in real-world dataset.

Figure 3.9. Classical flares are characterized by a single-peak profile with clear rising and decaying phases, while complex flares feature multiple peaks during the flare event [3, 23].

3.5 Experimental Setting

Here, we describe how to set parameters for various methods in this experimental and list of methods that were utilized as follows:

3.5.1 Window Size

The window size (q) is a critical parameter in time-series analysis. As explained in Section 3.3, the value of q was set to 20 based on the findings reported in [85]. This standard window size is adopted for uniform evaluation across all methods.

For methods like PAA, SAX, and 1D-SAX that require a predetermined bin size, we set the bin size (m) at 213 when $q = 20$. The total length of each LC analyzed was 4260 instances, with w determined by the formula $t = q \times m$, where t is the length of the LC. Additionally, for a detailed evaluation, we set the bin size that aligned with transient

patterns with a duration of 60 for $q = 71$.

In the case of MP, we employed Fast Low-cost Unipotent Semantic Segmentation (FLUSS), an extended version of MP for time series segmentation [32]. FLUSS segments the time series into q segments using the distance profile generated by MP, allowing us to set q at 20 based on the TWIN results.

3.5.2 Symbol Size for SAX ,1D-SAX

Both SAX and 1D-SAX require the specification of symbol sizes to quantize the statistical features within each bin. For SAX, we quantized the means of each bin into N symbol. For example, if we quantize their values into four different symbols, it is referred to as SAX $N = 4$. For 1D-SAX, a more detailed quantization strategy is employed, where both the mean and slope values are divided into $N1 \times N2$ symbols, with $N1$ and $N2$ representing the number of quantization levels for mean and slope, respectively. As an example, we quantized mean values into four symbols and slope values into ten symbols, denoted as 1D-SAX $N = 4 \times 10$.

3.5.3 Subsequence Length for MP

MP requires the specification of a subsequence length (L) to compute the distance profiles used in FLUSS. The length of each subsequence significantly affects how MP assesses the distance to the most dissimilar subsequence within the whole LC. MP is based on comparing an observed subsequence to all others in the LC using Euclidean distance. We hypothesized that subsequences containing a complete transient pattern would have a higher Euclidean distance. Therefore, we set L to match the durations of known transient patterns,

specifically 60 and 200. This ensured that the subsequence represented a complete transient pattern, thus facilitating accurate pattern detection.

3.5.4 Z-normalized and Non-normalized Euclidean Distance for MP

For Euclidean distance computation of MP, it supports two modes: Z-normalized and non-normalized Euclidean distances [52, 58]. We conducted experiments in both settings. 'MP-Z' refers to the use of Z-normalized Euclidean distance, while 'MP-non' refers to non-normalized Euclidean distance.

3.6 Experimental Results

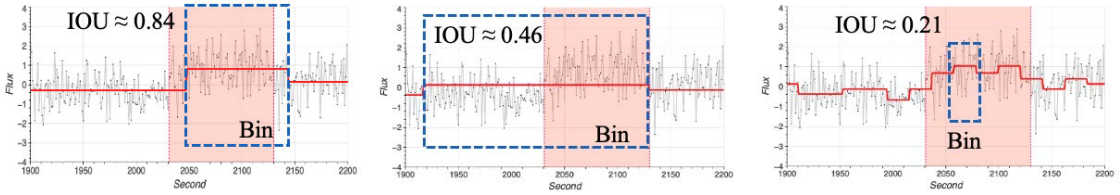
3.6.1 Experimental Results with Synthetic Datasets

Quality of Capturing Transient Patterns

The objective of this experiment is to evaluate the effectiveness of capturing transient patterns within bins. Ideally, a single bin should capture the complete transient pattern. To evaluate the quality of capturing, we utilized the Intersection over Union (IOU) as an indicator. Let $Y_{(i,m)}$ be a subsequence where all data points within this subsequence are a transient pattern, and $Y_{(i_p,m_p)}$ be a subsequence corresponding to bin_p. The IOU is expressed as:

$$IOU = \frac{|Y_{(i,m)} \cap Y_{(i_p,m_p)}|}{|Y_{(i,m)} \cup Y_{(i_p,m_p)}|} \tag{3.12}$$

A higher IOU indicates better transient pattern capture, while a lower IOU suggests lower quality. Figure 3.10 compares varying IOU results. However, in cases with



(a) High IOU (b) Low IOU (c) IOU with multiple bins

Figure 3.10: Comparison of varying IOU results.

Table 3.1: Comparison IOU of Square-LCs representation by varying methods

Method	q	Height = 1σ				Height = 3σ			
		Duration 60	Duration 100	Duration 200	Duration 500	Duration 60	Duration 100	Duration 200	Duration 500
M-Ebinning	20	0.659	0.752	0.792	0.712	0.914	0.931	0.907	0.777
L-Ebinning	20	0.223	0.286	0.386	0.499	0.317	0.414	0.473	0.558
PAA	20	0.260	0.401	0.604	0.426	0.262	0.391	0.609	0.426
SAX N=4	20	0.130	0.239	0.546	0.676	0.229	0.379	0.618	0.706
SAX N=20	20	0.241	0.388	0.605	0.544	0.259	0.391	0.614	0.641
1D-SAX 4x10	20	0.236	0.380	0.596	0.510	0.259	0.390	0.609	0.513
1D-SAX 5x5	20	0.217	0.387	0.595	0.564	0.261	0.390	0.609	0.591
1D-SAX 10x4	20	0.240	0.392	0.600	0.515	0.261	0.391	0.609	0.544
MP-Z L=60	20	0.130	0.156	0.188	0.292	0.146	0.159	0.181	0.298
MP-Z L=200	20	0.179	0.233	0.317	0.346	0.177	0.286	0.409	0.367
MP-non L=60	20	0.118	0.121	0.158	0.306	0.125	0.156	0.201	0.338
MP-non L=200	20	0.257	0.353	0.375	0.414	0.268	0.394	0.428	0.400
PAA	71	0.613	0.577	0.300	0.120	0.626	0.576	0.300	0.120
SAX N=4	71	0.537	0.625	0.702	0.623	0.617	0.713	0.832	0.933
SAX N=20	71	0.607	0.594	0.464	0.300	0.632	0.684	0.814	0.770
1D-SAX 4x10	71	0.605	0.577	0.391	0.241	0.626	0.576	0.431	0.309

multiple bins associated with the transient pattern (Figure 3.10c), we consider only the bin with the highest IOU.

Tables 3.1 - 3.3 show the IOU results for each method and dataset. Note that the IOU close to one in these tables indicates that a bin effectively captures the complete transient pattern.

Table 3.2: Comparison IOU of Triangle-LCs representation by varying methods

Method	q	Height = 1σ				Height = 3σ			
		Duration 60	Duration 100	Duration 200	Duration 500	Duration 60	Duration 100	Duration 200	Duration 500
M-Ebinning	20	0.390	0.479	0.515	0.456	0.515	0.462	0.392	0.302
L-Ebinning	20	0.189	0.248	0.361	0.466	0.319	0.364	0.417	0.539
PAA	20	0.259	0.408	0.589	0.426	0.255	0.404	0.623	0.426
SAX N=4	20	0.105	0.200	0.375	0.545	0.148	0.336	0.615	0.513
SAX N=20	20	0.213	0.376	0.583	0.471	0.246	0.399	0.625	0.463
1D-SAX 4x10	20	0.209	0.353	0.557	0.438	0.240	0.396	0.623	0.427
1D-SAX 5x5	20	0.161	0.308	0.554	0.455	0.235	0.399	0.623	0.426
1D-SAX 10x4	20	0.220	0.375	0.579	0.431	0.247	0.400	0.623	0.426
MP-Z L=60	20	0.142	0.156	0.195	0.287	0.135	0.173	0.199	0.288
MP-Z L=200	20	0.191	0.248	0.325	0.325	0.179	0.244	0.383	0.384
MP-non L=60	20	0.103	0.102	0.124	0.228	0.121	0.128	0.165	0.272
MP-non L=200	20	0.239	0.329	0.361	0.347	0.253	0.342	0.451	0.355
PAA	71	0.640	0.574	0.300	0.120	0.625	0.578	0.300	0.120
SAX N=4	71	0.446	0.534	0.475	0.427	0.614	0.659	0.699	0.670
SAX N=20	71	0.509	0.590	0.355	0.411	0.625	0.594	0.396	0.612
1D-SAX 4x10	71	0.626	0.570	0.323	0.223	0.623	0.577	0.318	0.233

Table 3.3: Comparison IOU of Kepler-LCs representation by varying methods

Methods	q	Kepler (All phases)			Kepler (Rising phase only)		
		1σ	2σ	3σ	1σ	2σ	3σ
M-Ebinning	20	0.259	0.342	0.354	0.107	0.260	0.317
L-Ebinning	20	0.271	0.264	0.255	0.060	0.067	0.070
PAA	20	0.464	0.448	0.466	0.135	0.137	0.137
SAX N=4	20	0.181	0.233	0.248	0.050	0.063	0.066
SAX N=20	20	0.338	0.411	0.448	0.094	0.122	0.128
1D-SAX 4x10	20	0.330	0.389	0.425	0.095	0.118	0.127
1D-SAX 5x5	20	0.189	0.284	0.412	0.055	0.091	0.122
1D-SAX 10x4	20	0.380	0.412	0.457	0.107	0.121	0.133
MP-Z L=60	20	0.236	0.264	0.227	0.112	0.108	0.089
MP-Z L=200	20	0.243	0.267	0.229	0.066	0.070	0.062
MP-non L=60	20	0.158	0.183	0.185	0.073	0.091	0.087
MP-non L=200	20	0.356	0.371	0.347	0.102	0.109	0.103
PAA	71	0.492	0.492	0.492	0.432	0.441	0.425
SAX N=4	71	0.438	0.493	0.496	0.205	0.224	0.333
SAX N=20	71	0.511	0.506	0.494	0.379	0.413	0.406
1D-SAX 4x10	71	0.515	0.507	0.491	0.404	0.436	0.423

As summarized in Table 3.1, M-EBinning achieves the highest IOU with $q = 20$ for capturing square patterns. Moreover, it also achieves the highest IOU for capturing the triangle patterns with short-duration (see Table 3.2). Although, L-EBinning achieves the highest IOU for capturing large triangle patterns, as shown in Table 3.2. However, the IOU of L-EBinning is lower when capturing short-duration or low-height triangle patterns. This reduced performance is attributed to the difficulty in distinguishing slopes of short-duration triangle-wave transient patterns (see Figure 3.7). Additionally, L-EBinning does not achieve high IOU with Kepler-LCs because the duration of the rise phases of Kepler patterns is very short (10 instances) as shown IOU results in Table 3.3.

As demonstrated in Tables 3.1 and 3.2, the IOUs for PAA, SAX, and 1D-SAX methods were significantly high when the window size was set to 71. This effectiveness is attributed to the bin size approximately matching a transient pattern duration of 60. Knowing the duration of transient patterns in advance enables the optimal configuration of bin sizes when applying methods like PAA, SAX, or 1D-SAX. Additionally, SAX configured with $q = 71$ also improved IOU for capturing long-duration transient patterns (duration of 100 - 500). This is because we integrated a merging function into SAX, similar to the strategy used in EBinning. Without this merging function, SAX's performance mirrors that of PAA.

The IOU of MP-Z was the lowest because MP-Z compares the similarity between the observational subsequence and all subsequences in LCs using the Z-normalized Euclidean distance. The limitation is that Z-normalized Euclidean distance is not suitable for LCs that include heavy noise. This is because MP-Z rescales each subsequence by Z-normalization,

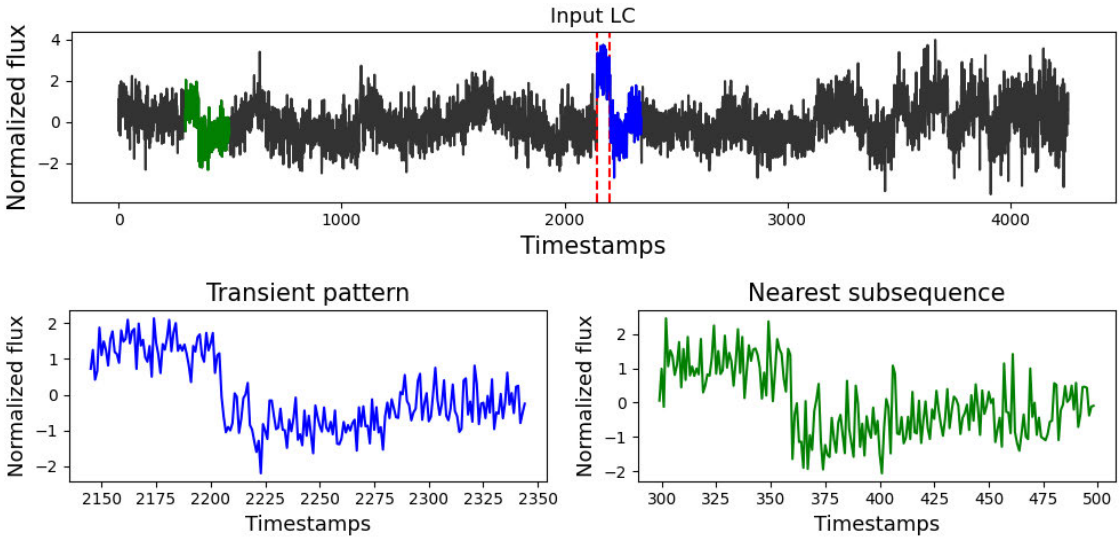


Figure 3.11: Sample results of MP-Z where the red lines indicate the start and end points of the transient pattern. The blue highlight marks the subsequence containing the transient pattern, while the green highlight shows the subsequence most similar to the blue subsequence as identified by MP-Z.

which may amplify the noise or transient pattern within those subsequences. Figure 3.11 demonstrates how Z-normalization can amplify noise and transient pattern. As illustrated in this figure, the amplified noise and transient pattern significantly distort the inherent patterns in the data, leading to poor similarity assessments. Consequently, MP-Z yields lower IOU scores.

We utilized MP-non, a mode that does not employ Z-normalization for each subsequence. MP-non achieved better IOU scores than MP-Z because the peaks of the transient patterns and noise were not rescaled by Z-normalization. However, MP-non struggles to capture small transient patterns. Notably, our evaluation of MP is based on publications

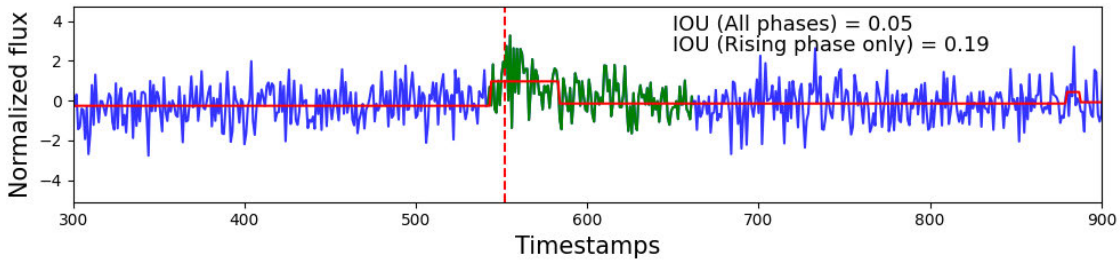


Figure 3.12: Comparison of IOU results for Kepler pattern. The solid red line represents the result of EBinning, the dashed red line represents the peak of the Kepler pattern, and the green line represents all phases of the Kepler pattern.

[32, 134, 141]. Recently, they proposed an enhanced version named Mplot, which is used to identify exoplanets with LCs. Investigating and evaluating Mplot remains a task for future work.

Turning to the IOU results of Kepler-LCs by varying methods, we observed that PAA achieved the highest IOU across all phases of the Kepler pattern. In contrast, EBinning captures only the rising phase but fails to accurately captures all phases of the pattern. This failure is partly because EBinning might misclassify the decay phase as similar to normal conditions, as illustrated in Figure 3.12. This impacts the IOU calculation, which relies on the intersection interval between the bin containing the transient pattern and the transient pattern itself. Consequently, we recalculated the IOU, focusing only on the rising phase of the Kepler pattern. As shown in Table 3.3, MEBinning achieved the highest IOU, confirming its effectiveness in capturing the rising phase.

However, EBinning did not achieve the highest IOU for Kepler patterns at 1σ height, as these patterns are challenging to distinguish, as depicted in Figure 3.8. We also

observed that methods using a window size of 71, which employ smaller bin sizes, attained higher IOU values compared to those with a window size of 20. This improvement suggests that smaller bin sizes can enhance IOU.

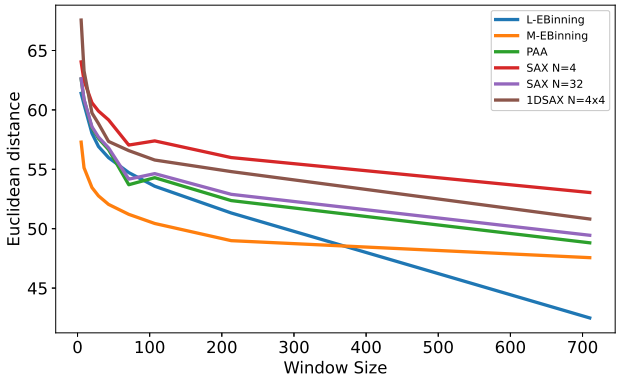
As previously mentioned, a larger window size with small bin sizes may not eliminate noise from unwanted outliers. This is because small bin sizes are less reliable and analogous to the concept where small sample sizes fail to reflect the true mean based on specific features of subsequences, resulting in each bin ultimately failing to reduce the noise. Next, we will provide results on sketching quality in the following subsection.

Sketching Quality

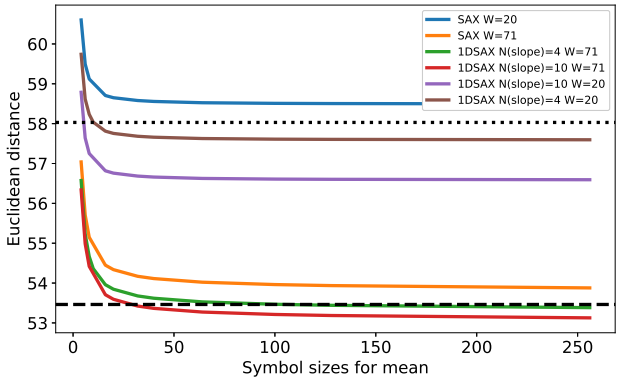
We assessed the representation quality by measuring the approximation error. We calculated as the Euclidean distance between the LCs and the sketching results for each method.

Figure 3.13 shows the comparison results of the Euclidean distance for various parameters. As previously discussed regarding window size, these results demonstrate that both of our proposed methods achieved lower Euclidean distances compared to PAA, SAX, and 1D-SAX (see Figure 3.13a). This indicates that our methods provide higher quality in terms of capturing and sketching. Moreover, increasing the window size in our proposed methods resulted in the lowest Euclidean distances among all methods tested.

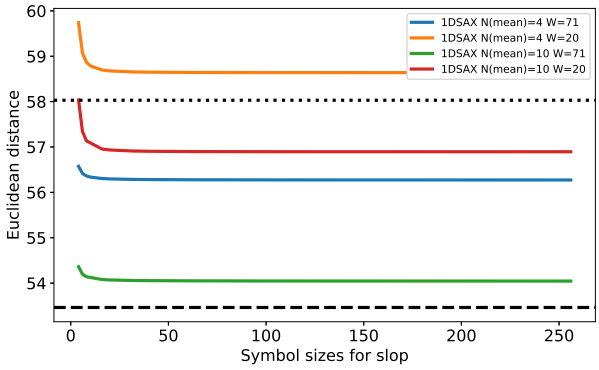
By adding more parameters concerning the number of symbols, we observed that the symbol count significantly impacts sketching quality, as depicted in Figures 3.13b and 3.13c. Note that we established two thresholds; the dashed and dotted lines represent the



(a) Various window size.



(b) Various symbols for means.



(c) Various symbols for slopes.

Figure 3.13: Comparison of Euclidean distance by various parameters.

Euclidean distances for M-EBinning and L-EBinning, respectively, with $q = 20$. These results exhibit behavior similar to that shown in Figure 3.13a. The findings suggest that while increasing the window size and the number of symbols can reduce the Euclidean distance, it may not necessarily yield higher IOU scores and may not achieve lower distances than those offered by our proposed methods. Additionally, defining these parameters manually for other methods that yield good results can be time-consuming.

Accuracy for Detecting the Transient Pattern

We conducted an evaluation based on the accuracy rate for detecting transient patterns in Square- and Triangle-LCs using each method. The detection procedure includes the following three steps:

1. We computed sketching results for each LC by varying the method with the window size set to $q = 20$.

2. We listed the top-k bin boundaries for the LC based on the metrics of each method.

3. If a bin boundary falls within the period when the transient occurs, we consider the transient pattern detection to have succeeded in that LC.

For our proposed methods, we enhanced transient pattern detection by adapting the mergeability scores (Eqs. 3.2 and 3.8). After computing EBinning, we ordered the mergeability scores in $\mathcal{S}$ to list the top-k, where the highest mergeability scores indicate the inclusion of a transient pattern within the bin. We also applied this approach with the SK-method and DyBin.

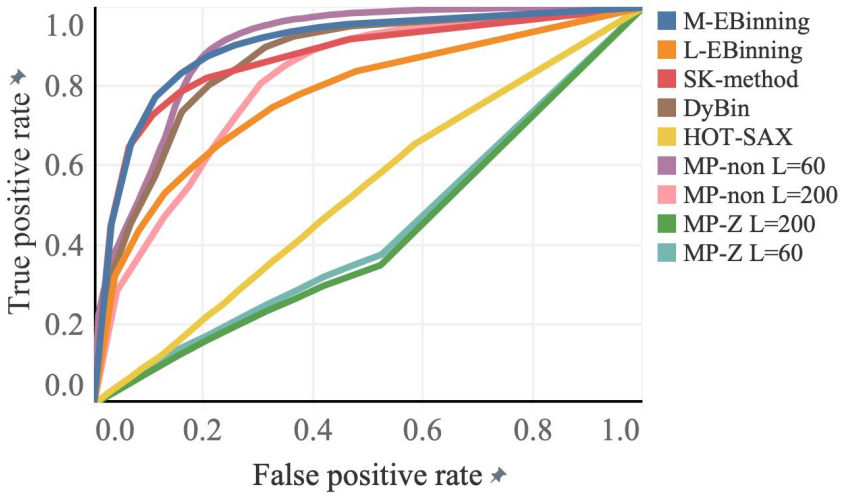


Figure 3.14: Comparison of ROC curve by various methods.

For MP and HOT-SAX, which measure the dissimilarity between the observational subsequence and every other subsequence in the LCs, they select the top-k subsequences based on their most dissimilar distances. The highest dissimilar distances suggest that these subsequences are distinctly different from the others, especially since we injected one transient pattern into each LC.

Figure 3.14 shows the ROC curve of this experiment for various methods. We concluded the experimental results as follows:

M-EBinning achieved satisfactory results in detecting transient patterns in both Square- and Triangle-LCs. However, the performance of L-EBinning significantly dropped compared to M-EBinning. This decrease is attributed to L-EBinning's lower IOU, suggesting that it may fail to capture some transient patterns.

For MP, we observed that MP-non performed better than MP-Z. Those results are similar to the evaluations in terms of the quality of capturing transient patterns, with

MP-non achieving a higher IOU than MP-Z. However, MP-Z did not provide satisfactory results compared to M-EBinning, as the false negative rate in MP-non was higher than in M-EBinning at a true positive rate around 0.7 in Figure 3.14.

HOT-SAX, a discord discovery method, did not provide satisfactory results because its ROC curve was close to a linear line. This issue arises because HOT-SAX utilizes SAX, where the number of symbols impacts the quality of capturing transient patterns. Therefore, optimizing the parameter for the number of symbols may improve results, but it can be difficult to assume the optimal settings for each LC, which may exhibit different behaviors.

In summary, M-EBinning exhibited superior performance in terms of the quality of capturing transient patterns, sketching results, and accuracy rate for detecting transient patterns in synthetic datasets. However, other methods may provide high performance with optimally provided input parameters. Our proposed method is a better choice that is flexible for users without any assumptions regarding input parameters.

3.6.2 Experimental Results with Real-World Datasets

In this subsection, we present detailed experimental results using LCs containing stellar flares[2]. These experimental results demonstrate the success of using M-EBinning and highlight some limitations that require further research for improvement.

[2]LC dataset used in these experiments was provided by M. Aizawa and K. Kashiyama. Interested readers can refer to [3] for more details.

Stellar Flare Analysis Using M-EBinning

We utilized M-EBinning to LCs containing stellar flares and visually analyzed the results. The sample outcomes are illustrated in Figure 3.15, where we observed that M-EBinning can capture a stellar flare within a single bin. Note that comprehensive results covering all flares are presented in the Appendix A.1.

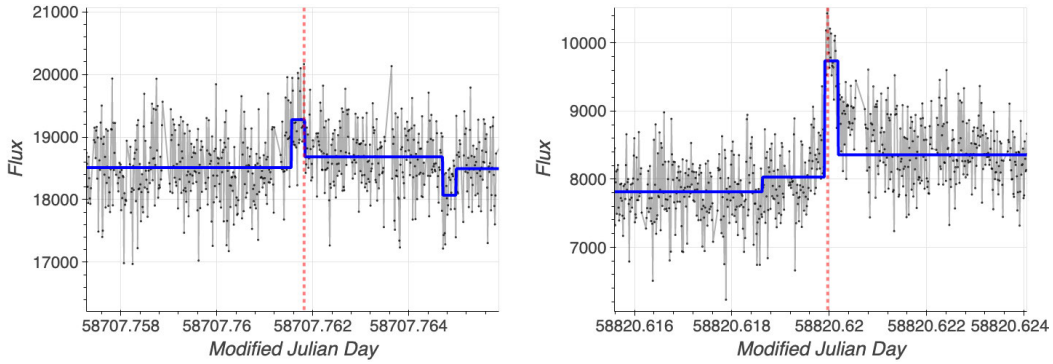


Figure 3.15: Sketching results showing transient patterns from stellar flares [3], where the black lines represent the original LCs and the blue lines represent the results from M-EBinning.

Stellar Flare Detection

We evaluated flare detection using M-EBinning, HOT-SAX, and MP-non. Similar to our approach with synthetic datasets, we adapted mergeability scores for M-EBinning and dissimilar distances for MP-non and HOT-SAX. However, we selected only the top-1 result, where the bin had the highest mergeability scores or dissimilar distances. To calculate the true positive rate (TPR), we utilized 18 LCs, each containing only one bin with the peak flare (the peak is referred to as one point). A bin that included the peak

Table 3.4: Detection results with real flare by various methods

Method	TPR	FPR
M-EBinning	0.778	0.011
HOT-SAX	0.722	0.015
MP-Non	0.611	0.021

flare and had the highest mergeability scores (or dissimilar distances) was considered a true positive. Conversely, a bin that had the highest scores but did not include the peak flare was considered a false positive. The results are presented in Table 3.4. Overall, M-EBinning, HOT-SAX, and MP-non prove effective for detecting stellar flares in LC data. However, M-EBinning exhibits slightly superior performance compared to the other methods.

Remaining Challenges in Analysis Using M-EBinning

Here, we showcase examples of false alarms provided by M-EBinning, highlighting ongoing challenges in the analysis. We conducted the evaluation using LCs that did not include stellar flares verified by astronomers. The first scenario is related to LCs generated from videos that capture the main star at the center and also include nearby stars. The light intensity from nearby stars can destabilize the LCs. In real-world analysis, astronomers mitigate these effects by analyzing the video data directly or reducing the aperture size when generating LCs through aperture photometry. Figure 3.16 shows this scenario. Note that the main star as shown in this figure appears in the center of the video but the nearby star is high-intensity.

The second scenario is related to outliers that persist for more than two seconds. Typically, unwanted outliers are single samples caused by measurement errors from hard-

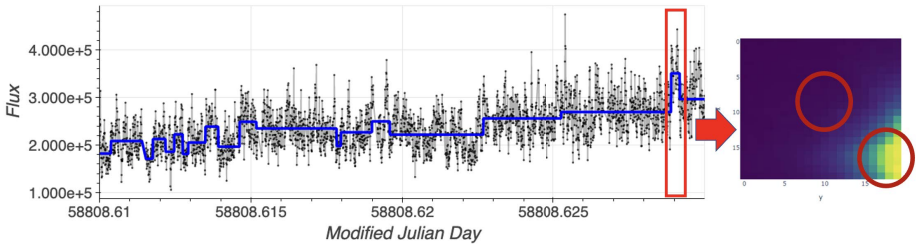


Figure 3.16: Example of LC including nearby star

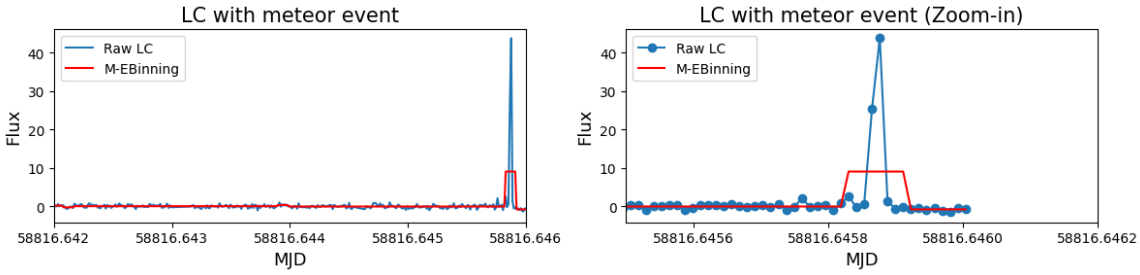
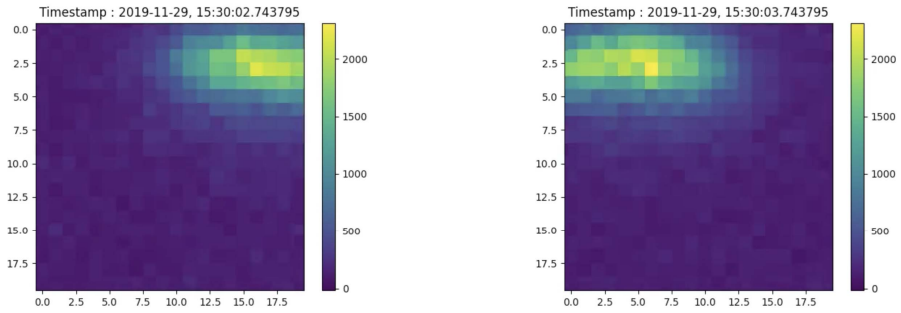


Figure 3.17: LC from a trajectory of the artificial object: (Left) Full-length LC with the trajectory of the artificial object occurrence highlighted in red; (Right) A zoomed-in view of the period highlighted in red.

ware or cosmic rays. However, the outliers in these LCs do not occur as single samples and the subsequences containing these outliers do not resemble stellar flares, as characterized by the Kepler or Aizawa models [3, 23]. Upon reviewing the video data related to this period of the LC, we observed an anomalous object moving from the top right to the top left of the video with high-intensity. After further analysis, astronomers speculated that it might be a trajectory of artificial object. Although M-EBinning captured this phenomenon, it was considered a false positive because it did not correspond to a stellar flare. This LC and the related video are shown in Figures 3.17 and 3.18.



(a) First frame (b) Second frame

Figure 3.18: Video from a trajectory of the artificial object.

From these examples, it is evident that analyzing only LCs has limitations in distinguishing different phenomena. Typically, astronomers analyze both videos and LCs to determine whether they are observing targeted phenomena. Therefore, there remains a challenge in analyzing videos using data binning techniques or in reducing false alarms caused by other astronomical phenomena in analyzing LCs.

3.7 Discussion and Conclusions

This chapter details EBinning, a novel method that automatically adjusts bin sizes based on a mergeability score. This score indicates whether two neighboring bins can be merged without distorting the original features. EBinning is available in two modes: M-EBinning, which compares distributions between two neighboring bins, and L-EBinning, which compares trends between two neighboring bins.

M-EBinning has proven effective in terms of high quality in capturing and sketching transient patterns, and accuracy in detection. In contrast, the performance of other methods

often depends on input parameters such as bin size, the number of symbols for quantization, or subsequence length. Therefore, EBinning offers valuable tools for astronomers studying transient phenomena, eliminating the need for assumptions about input parameters.

In our future work, efforts will focus on reducing false positives that occur due to nearby stars or other astronomical phenomena. We aim to explore the potential of applying EBinning to detect other astronomical phenomena such as fast radio bursts and meteors. These enhancements are expected to broaden the applicability of EBinning and improve its accuracy in diverse astronomical settings. Additionally, we find opportunities to analyze LCs that include a seasonal component, such as those from the periodic behavior of variable stars [33] or the behaviors of γ-ray emissions [80].

Chapter 4

Online Season Length Estimation

This chapter explores the Online Season Length Estimation (OnlineSLE) method, a crucial part of our Adaptive Seasonal-Trend Decomposition method. The primary goal of OnlineSLE is to estimate the season length, which indicates the duration of each complete cycle in the time series data. We specifically address environments where data is continuously updated in a streaming manner.

We organize this chapter into seven sections to enhance clarity and depth. Section 4.1 provides the necessary background on the seasonal components in time series. Section 4.2 reviews related work and identifies remaining gaps in existing methods. Section 4.3 explains the methodology behind OnlineSLE. Subsequently, Sections 4.4 and 4.5 detail the datasets and experimental settings used. Section 4.6 presents the results of our experiments. Finally, Section 4.7 offers the remaining challenge, conclusions, and outline directions for future research.

4.1 Background

Building on the foundational knowledge presented in Section 2.2.2, this chapter explores the complexities and methodologies of estimating season lengths in real-time environments. As discussed earlier, accurate estimation of season lengths is pivotal across various domains, including economic forecasting, climate monitoring, and health diagnostics, where existing methods did not design for handling streaming data. In this thesis, we address two previously mentioned challenges: fast computation and accurate estimation. Here, we further detail these challenges in the following paragraphs.

Typically, season length aligns with standard intervals such as daily, weekly, or monthly periods. However, some time series do not depend on these intervals, and a lack of knowledge about the season length can be problematic. To enhance understanding, we provide two classical time series in Figure 4.1 [11, 112]. Figure 4.1 (Top) shows the monthly totals of international airline passengers from 1949 to 1960 after removing trends. We observe that passenger numbers strongly depend on the month, peaking in summer and declining in winter. Through visual inspection, we conclude a season length of 12 months. Turning to Figure 4.1 (Bottom), it displays the monthly numbers of sunspots, we again identify cyclic behavior that does not align with monthly intervals. We approximate the season length at about 11 years. However, one might question whether it could be nine years. It might be preferable to estimate season length without relying on visual inspection, as the human eye can be misleading and requires specific knowledge. We address these issues using the season length estimation (SLE) technique, with numerous studies supporting this method [92, 118, 120, 127].

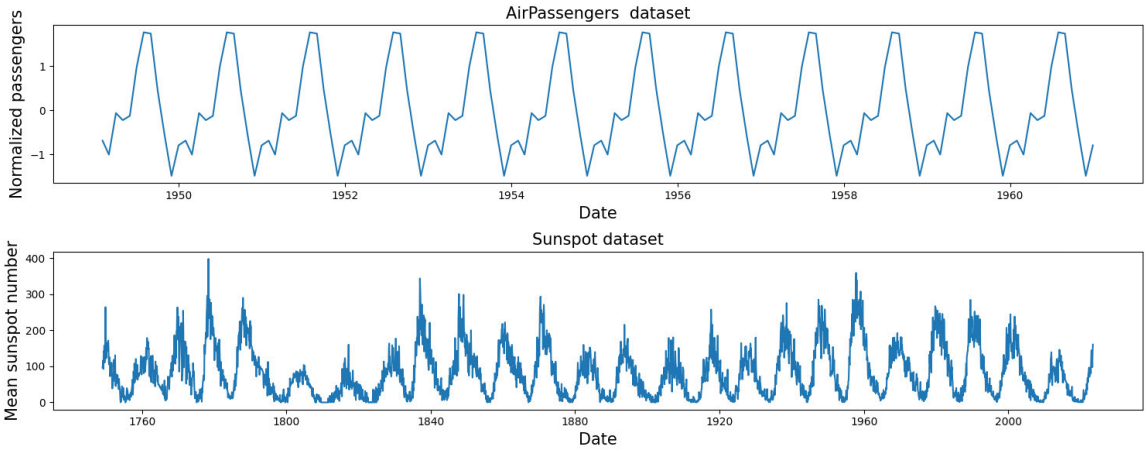


Figure 4.1: Two classical time series. (Top) Monthly totals of international airline passengers [11].; (Bottom) Monthly numbers of sunspots[112]

To address these challenges, our work introduces the concept of *Online Season Length Estimation* (OnlineSLE). OnlineSLE represents an innovative approach designed to work in streaming settings, enabling analysts to estimate season lengths accurately and swiftly without waiting for complete data cycles.

4.2 Related Work

In this section, we first begin with the problem statement. Then, we will discuss the two baseline methods that are traditionally utilized for SLE: the *periodogram* and the *auto-correlation function* (ACF). We will also explore related works in the field of SLE.

4.2.1 Problem statement

Informally, season length can be estimated by identifying the number of observations between observable peaks or cyclic behaviors within a dataset, typically by visual

inspection. However, for computational analysis, this task must be formalized mathematically. The SLE problem involves determining the periodicity of a time series by quantifying the length of these cyclic behaviors.

To formalize this, let us define a time series Y as $Y = T + S + R$ (as in Definition 2). Here, Y_t, T_t, S_t, and R_t denote the observed value of the time series, trend component, seasonal component, and residual component at timestamp t, respectively. The goal of SLE is to determine the season length m, where $S_t \approx S_{t-m}$ holds for every t, indicating that the season length m is consistent throughout the S.

4.2.2 Periodgram and Auto-correlation Function

The periodogram and ACF analyze data in different domains: the periodogram operates within the *frequency domain*, while ACF within the *time domain*.

Periodogram

The periodogram is a graph that displays the power spectral density of a time series within the frequency domain. It can indicate which frequencies are predominant in the given time series by transforming a time series from the time domain into the frequency domain through the Discrete Fourier Transform (DFT).

Consider a time series without a trend component $Y = (Y_1, \ldots, Y_N)$, where N is the length of the series and $n = 1, \ldots, N$ serves as the index for each element of Y. The periodogram ($\mathcal{P}(k)$) is defined by the following equations:

$$\mathcal{P}(k) = \frac{1}{N} \left\| \sum_{n=1}^{N} Y_n e^{-j2\pi(n-1)k/N} \right\|^2 \qquad k = 0, 1, \ldots N-1 \tag{4.1a}$$

$$\mathcal{P}(k) = \frac{1}{N} \left\| \mathcal{F}(k) \right\|^2 \tag{4.1b}$$

In these equations, $e^{-j2\pi k/N}$ denotes the twiddle factor, which facilitates the transformation of the time series into the frequency domain. The normalized bin index k/N corresponds to a specific frequency f, expressed as $f = k/N$. The term $\mathcal{F}(k)$ in Equation 4.1b denotes the DFT of Y.

In the periodogram, large values indicate that the time series exhibits cyclical properties of a seasonal component at specific frequencies. For example, consider a time series consisting of five sine waves with a season length of 20 timestamps. When calculating the periodogram of this entire time series, we observe that the peak value occurs at a frequency of 0.05, as illustrated in Figure 4.2.

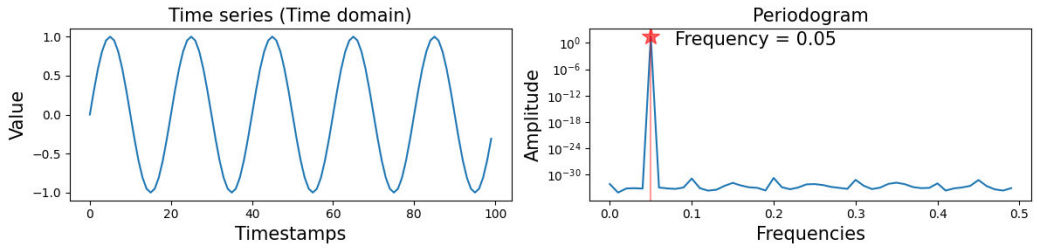


Figure 4.2: Comparison between the time-domain and frequency-domain expressions the periodogram is an alternative representation of the original data.

As shown in Figure 4.2, we can map a frequency f to the time domain in the context of the periodogram by taking the reciprocal of this frequency. This transformation reveals that the highest peak at a frequency of 0.05 corresponds to a season length of 20

timestamps in the time domain. This correlation between frequency and season length underscores the utility of the periodogram in identifying the length of seasonal component within time series.

The primary limitation of employing DFT is its frequency resolution, which depends on the total number of points N used in the DFT [47, 66]. This limitation can lead to inaccurate results in SLE. To illustrate this, we analyze a subset of the time series shown in Figure 4.2, using only the first 83 timestamps ($N = 83$). The computed periodogram for this time series is shown in Figure 4.3.

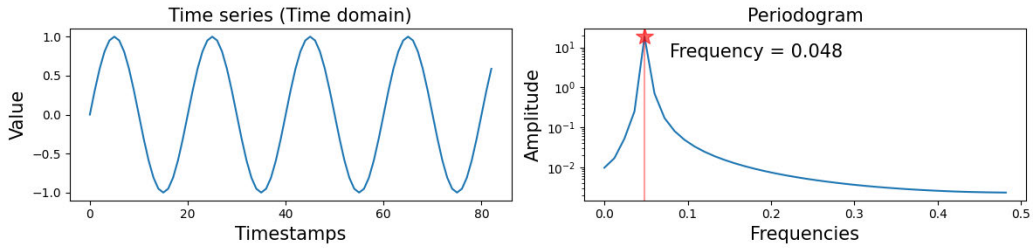


Figure 4.3: Illustration of frequency resolution issues in the periodogram due to influencer of data points in the DFT.

The peak of periodogram approximately appears at a frequency of 0.048, where $k/N = 4/83$. As k must be an integer in DFT, taking the reciprocal of this frequency suggests a season length of approximately 20.75, which deviates from the ground truth of 20 timestamps. Adjusting k by ± 1 to 3 or 5 yields reciprocal season lengths of 27.67 and 16.6 in the time domain, respectively, neither of which match the ground truth. Adjusting k by ± 1 to 3 or 5 results in reciprocal season lengths of 27.67 and 16.6 in the time domain. This further illustrates the mismatch from the ground truth season length. In contrast, the

ACF can address this issue by analyzing the data directly in the time domain, providing a more reliable estimate of season length where frequency resolution is less critical.

Auto-correlation Function

ACF operates within the time domain and measures the self-similarity of a signal over varying delay times, referred to as lag. This function is particularly useful for identifying cyclical properties within a time series, thereby assisting in determining the season lengths of such series. Consider a time series without a trend component $Y = (Y_1, \ldots, Y_N)$. The ACF for a lag k is defined by the following equations:

$$\hat{\rho}_k = \frac{\sum_{n=1}^{N-k}(Y_n - \bar{Y})(Y_{n+k} - \bar{Y})}{\sum_{n=1}^{N}(Y_n - \bar{Y})^2} \tag{4.2}$$

where Y_n denotes the data points in the time series, $\bar{Y}$ denotes the mean of the time series. The ACF helps to identify the degree to which current values in the time series are influenced by past values up to a certain lag k, thereby revealing the seasonal component at this length. To illustrate this, we compare the time series (using similar input as shown in Figure 4.2) with the results of the ACF applied to this series, as shown in Figure 4.4.

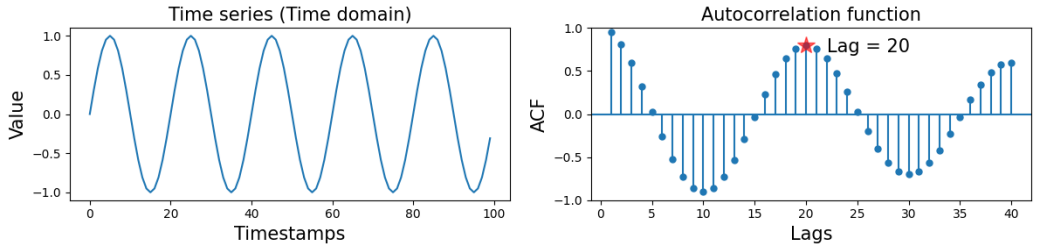


Figure 4.4: Comparison between the original time series and the ACF.

As shown in Figure 4.4, the highest peak of the ACF at lag k indicates that this k corresponds to the seasonal component of length k within the time series. From this example, the highest peak occurring at lag 20 suggests that the cycle length within the time series is 20 timestamps. This analysis helps to demonstrate the reliability of ACF in identifying true seasonal lengths without the frequency resolution limitations observed with the periodogram. Additionally, we have shown the ACF results by providing a subset of the time series using only the first 83 timestamps, as shown in Figure 4.5.

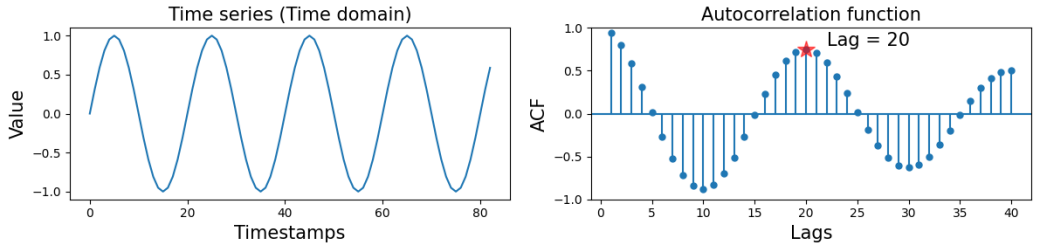


Figure 4.5: ACF analysis on a subset of the time series.

We found that the influence of the length of input data does not significantly impact the ACF, with the highest peak still occurring at lag 20. However, analyzing in the time domain can be challenging when estimating data with noise. To illustrate this, we analyze a time series shown in Figure 4.2 but injected with Gaussian noise. The new time series and its computed ACF are shown as Figure 4.6 (Left) and (Right), respectively.

From these results, we observe that the highest peak occurs at lag 21, suggesting that the cycle length within the time series is 21 timestamps. It deviates from the ground truth of 20 timestamps. This indicates that ACF analysis may produce incorrect results when dealing with heavily noisy data. In such scenarios, the periodogram, which analyzes

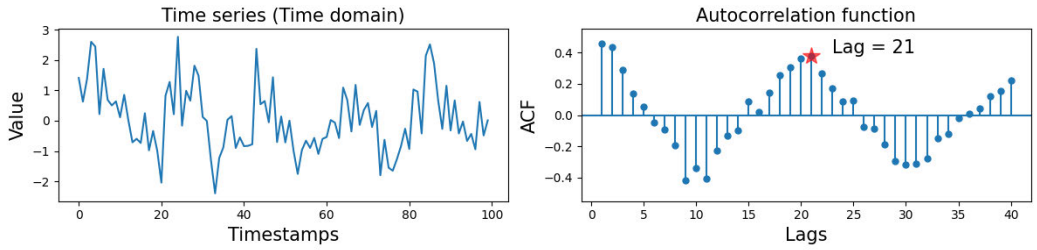


Figure 4.6: ACF analysis on a the time series with Gaussian noise injection.

data in the frequency domain, may be a better choice.

Both the periodogram and ACF are utilized to estimate the season length within different domains. Each has its advantages and disadvantages. The following subsection will provide a detailed of existing SLE methods based on these two baseline methods.

4.2.3 Existing SLE methods

The first baseline method that combines the periodogram and ACF for SLE was introduced by M. Vlachos, P. Yu, and V. Castelli [120], known as the AutoPeriod method. This method consists of two primary steps: generating a list of candidate season lengths using the periodogram and then verifying this list using the ACF. They proposed an efficient computation approach for both the periodogram and ACF through the use of fast Fourier transform (FFT) and inverse fast Fourier transform (iFFT), respectively. As a result, AutoPeriod achieves a time complexity of $O(N \log N)$, where N is the length of the time series, primarily due to the utilization of FFT and iFFT.

The improved version of AutoPeriod is the Clustered Filtered Detrended AutoPeriod (CFD-Autoperiod) by T. Puech *et al.* They introduced additional steps such as density clustering, lowpass filtering, and linear detrending of the ACF. These enhancements aim to

increase the accuracy and robustness of the season length estimates. However, these improvements also result in a higher computational demand for CFD-Autoperiod compared to AutoPeriod.

Another improvement based on the periodogram and ACF is RobustPeriod [127]. This method enhances robustness by integrating the Huber loss [41] into the periodogram and ACF calculations. Huber loss is a hybrid error metric that combines the advantages of squared error loss and absolute error loss, which it is less sensitive to outliers within time series. Their study showed improved outcomes, although at a higher computational cost than AutoPeriod. Notably, the recent version of RobustPeriod, known as Robust Dominant Periodicity, addresses time series with missing data [128]. The foundational approach of this method combines the periodogram and ACF, adopting Huber loss to enhance its robustness and reliability.

M. Toller, T. Santos, and R. Kern introduced an enhanced baseline method that combines the periodogram and ACF, known as SAZED, which stands for Spectral and Average Autocorrelation ZEro Distance density [118]. This method is an ensemble approach that incorporates three components: the traditional baseline techniques of the periodogram and ACF, along with the average of time series zero distances. Although, they introduced an additional baseline component, the overall computational cost remains $\mathrm{O}(N \log N)$ in terms of time complexity by relying on FFT and iFFT.

An alternative method for SLE that utilizes only the ACF is FindLength [79]. This method identifies the highest spike from the ACF results, as shown in Figure 4.4. While FindLength benefits from faster computation due to its reliance solely on ACF, its

accuracy for SLE may decrease when dealing with time series data containing heavy noise, as discussed in Section 4.2.2.

The SLE methods discussed in this subsection were originally designed for batch processing in offline mode, requiring access to the entire time series data for estimation. To adapt these methods for online operation, a sliding window concept can be employed. Although this adaptation enables online functionality, the computational cost remains a concern in streaming environments. Each sliding window update requires $O(N \log N)$ in time complexity, largely due to the reliance on FFT and iFFT. Addressing this challenge is crucial for effective streaming data analysis.

Our proposal for an online SLE method utilizes the Sliding Discrete Fourier Transform (SDFT), which iteratively updates Fourier transform results by adding new data and subtracting the oldest, mirroring the sliding window concept. This concept significantly reduces the computational time to $O(N)$, which is faster than traditional FFT-based methods. Additionally, due to the frequency resolution issues inherent in the DFT concept, methods that rely on the ACF typically require $O(N \log N)$ time complexity. To overcome this, we propose an alternative solution that uses a spectral peak location estimator. This method achieves a computational cost of $O(N)$, thereby enabling faster computation compared to traditional SLE methods.

4.3 Online Season Length Estimation

This section details the concept for reducing the computational cost of the periodogram using the SDFT, discussed in Section 4.3.1. It then addresses the limitations of

the periodogram, explaining why existing methods utilize the ACF and how our proposed solution circumvents these limitations without relying on ACF, by employing the spectral peak location estimator detailed in Section 4.3.2. Finally, the process of implementing our OnlineSLE method is described in Section 4.3.3.

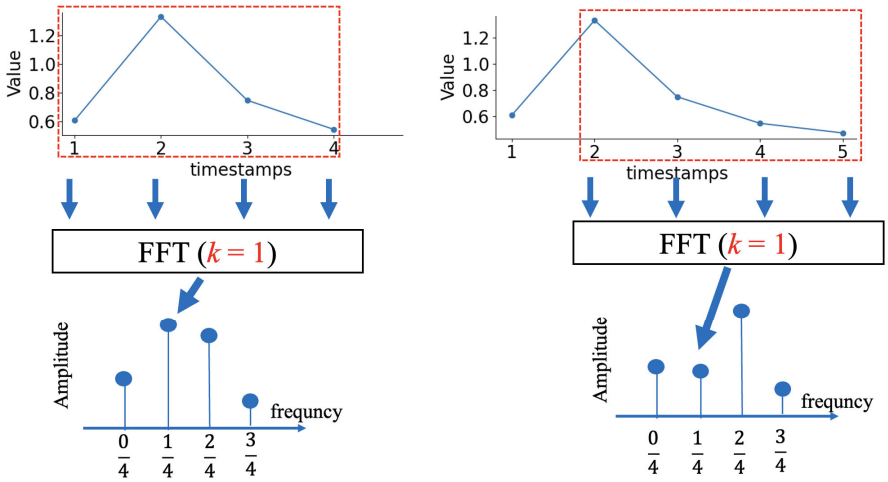
4.3.1 Sliding Discrete Fourier Transform (SDFT)

SDFT operates by leveraging the previously calculated DFT results to update the transform incrementally. This avoids the computational expense of recalculating the DFT from scratch for every incoming data in a sliding window, which is particularly advantageous for streaming data.

Consider a sliding window containing the time series $(Y_{t-N+1}, \ldots, Y_t)$, and compare it to the time series from the previous window $(Y_{t-N}, \ldots, Y_{t-1})$. We denote the DFT results for these two windows as $\mathcal{F}_t(k)$ and $\mathcal{F}_{t-1}(k)$, respectively. The key difference between $\mathcal{F}_t(k)$ and $\mathcal{F}_{t-1}(k)$ lies in the addition of the incoming data Y_t and the subtraction of the oldest data Y_{t-N} from every point in the frequency domain. This is mathematically represented by the following recursive update formula:

$$\mathcal{F}_t(k) = e^{j2\pi k/N} \left[\mathcal{F}_{t-1}(k) + Y_t - Y_{t-N}\right] \quad k = 0, 1, \ldots, N-1 \tag{4.3}$$

In this equation, the operations include adding Y_t, subtracting Y_{t-N}, and multiplying by the twiddle factor $(e^{j2\pi k/N})$ for each value of k from 0 to $N-1$. SDFT is executed once per frequency component in the DFT, across all N points of the sliding window. As such, the total computational cost for updating the SDFT involves N constant-time oper-



(a) $\mathcal{F}_4(k)$ (b) $\mathcal{F}_5(k)$

Figure 4.7: Comparison of FFT results between timestamps 4 and 5 with $k = 1$, demonstrating the SDFT update process.

ations, resulting in an overall complexity of $\mathrm{O}(N)$.

To illustrate the practical of SDFT, consider the example of transitioning from a time series window containing time series $(Y_1, \ldots, Y_4)$ to a new window as $(Y_2, \ldots, Y_5)$. We denote the DFT results for these windows as $\mathcal{F}_4(k)$ and $\mathcal{F}_5(k)$, respectively. Figure 4.7 shows how the SDFT is computed between timestamps 4 and 5 for $k = 1$. The difference between $\mathcal{F}_4$ and $\mathcal{F}_5$ highlights the addition of Y_5 and the subtraction of Y_1 from every point in the frequency domain, aligning with the SDFT's update formula.

Proof. This proof elucidates the recursive nature of the SDFT as detailed in Eq. (4.3), demonstrating how each new data point incrementally updates the DFT results, thus underscoring its computational efficiency. Given an input sequence with a length of at least $(N + q + 1)$, where q denotes the starting index of the DFT window, we consider a DFT of

length N for the window $(Y_q, Y_{q+1}, \ldots, Y_{q+N-1})$:

$$X_q = \sum_{n=0}^{N-1} Y_{n+q} e^{-j2\pi nk/N} \tag{4.4}$$

Then, sliding to the next window with the starting point at the $(q+1)$-th position, we compute the DFT of length N for this new window $(Y_{q+1}, Y_{q+2}, \ldots, Y_{q+N})$, dynamically tracking changes in the frequency domain as the window advances:

$$X_{q+1} = \sum_{n=0}^{N-1} Y_{n+q+1} e^{-j2\pi nk/N} \tag{4.5}$$

Substituting $p = n + 1$ for the range 1 to N, we have:

$$X_{q+1} = \sum_{p=1}^{N} Y_{p+q} e^{-j2\pi (p-1)k/N} \tag{4.6}$$

Adjusting for the N-th term by subtracting and adding the $p = 0$ case:

$$X_{q+1} = \sum_{p=0}^{N-1} Y_{p+q} e^{-j2\pi (p-1)k/N} + Y_{q+N} e^{-j2\pi (N-1)k/N} - Y_q e^{j2\pi k/N} \tag{4.7}$$

The exponential terms can be factored as follows:

$$X_{q+1} = e^{j2\pi k/N} \left[\sum_{p=0}^{N-1} Y_{p+q} e^{-j2\pi pk/N} + Y_{q+N} e^{-j2\pi Nk/N} - Y_q \right] \tag{4.8}$$

The $e^{-j2\pi Nk/N}$ term simplifies to $1 + j0$ for k is always integer values, since

$e^{-j2\pi Nk/N} = 1$, leading to:

$$X_{q+1} = e^{j2\pi k/N} \left[\sum_{p=0}^{N-1} Y_{p+q} e^{-j2\pi pk/N} + Y_{q+N} - Y_q \right] \tag{4.9a}$$

$$= e^{j2\pi k/N} \left[\sum_{n=0}^{N-1} Y_{n+q} e^{-j2\pi nk/N} + Y_{q+N} - Y_q \right] \tag{4.9b}$$

$$= e^{j2\pi k/N} [X_q + Y_{q+N} - Y_q] \tag{4.9c}$$

Note that the summation enclosed in square brackets in Eq. (6a) represents the DFT calculated for the kth component, using p as the indexing variable rather than n. For the latest timestamp t, the DFT results from the current sliding window $(Y_{t-N+1}, \ldots, Y_t)$ and the previous sliding window $(Y_{t-N}, \ldots, Y_{t-1})$ are denoted as $\mathcal{F}_t(k)$ and $\mathcal{F}_{t-1}(k)$, respectively. This notation allows us to succinctly express the DFT update formula, transitioning from $\mathcal{F}_{t-1}(k)$ to $\mathcal{F}_t(k)$ as follows:

$$\mathcal{F}_t(k) = e^{j2\pi k/N} [\mathcal{F}_{t-1}(k) + Y_t - Y_{t-N}] \tag{4.10}$$

By replacing FFT with SDFT for periodogram computation, we significantly reduce the computational cost from $\mathrm{O}(N \log N)$ to $\mathrm{O}(N)$. However, the challenge with frequency resolution remains. As previously discussed, existing SLE methods address this issue by utilizing ACF, which also has a computational cost of $\mathrm{O}(N \log N)$. The following subsection will introduce spectral peak location estimator, an alternative technique that maintains computational efficiency, ensuring the cost does not exceed $\mathrm{O}(N)$.

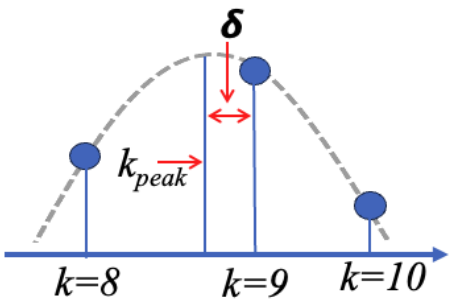


Figure 4.8: Example of DFT coefficients resolution issue.

4.3.2 Spectral Peak Location Estimator

As previously mentioned in Section 4.2.2, a primary limitation of employing DFT in a periodogram is its frequency resolution, which depends on the total number of points N used in the DFT. To illustrate this issue, we plot the periodogram in Figure 4.8.

Figure. 4.8 shows a plotting periodogram that is computed by DFT. In concept DFT, the index k must be an integer ($0 \leq k \leq N-1$). The $k = 9$ is closest to k_{peak}, where actual index yields the actual highest power in the periodogram. Adjusting k to 8 does not correspond to the actual highest power. This example highlights the potential for error due to frequency resolution limitations. As discussed in Section 4.2.2, numerous studies have utilized ACF to address this issue by analyzing both the frequency domain using the periodogram and the time domain using ACF. However, this combination incurs a computational cost of $\mathrm{O}(N \log N)$ due to ACF relies on iFFT. To reduce the computational cost, we adopted an alternative approach that utilizes *spectral peak location estimation* to interpolate k_{peak} with a computational cost less than $\mathrm{O}(N)$.

Spectral peak location estimator interpolates the index k_{peak}, which corresponds to

the highest power in the Fourier transform result without an increase in N [47, 66, 93, 104]. The k_{peak} is determined by $k_{peak} = \hat{k} + \delta$, where $\hat{k}$ denotes the index of the highest power location from periodogram (or DFT), and δ denotes the residual frequency, which can be positive or negative, as shown in Figure. 4.8. Spectral peak location estimator is based on the cuvre-fiting technique. This estimator is divided into two groups: non-iterative and iterative estimators [1, 104].

Non-iterative estimator

Non-iterative estimator is the method that computes k_{peak} by utilizing curve-fitting with the neighborhood DFT results of $\hat{k} \pm 1$. The interpolatation of this group generally relies on the peak DFT coefficient and its neighbors. For example, in Figure. 4.8, $\hat{k}$ corresponds to 9 and $\hat{k} \pm 1$ corresponds to 8 and 10. However, the interpolatation must be done in one time without reiteration for recheck the results.

To demonstrate the non-iterative estimator, we utilized the Quinn estimator for explanation, which is a classical estimator [66, 93]. The procedure of the Quinn estimator is as follows:

$$\alpha_1 = \Re(X_{\hat{k}-1}/X_{\hat{k}}) \tag{4.11a}$$

$$\alpha_2 = \Re(X_{\hat{k}+1}/X_{\hat{k}}) \tag{4.11b}$$

$$\delta_1 = \alpha_1/(1-\alpha_1) \tag{4.11c}$$

$$\delta_2 = -\alpha_2/(1-\alpha_2) \tag{4.11d}$$

where $\Re\{\cdot\}$ denotes the real part, and X_k denotes the DFT coefficients for positive

integer k values. The final δ estimation can be expressed as:

$$\delta = \begin{cases} \delta_2 & \delta_1 > 0 \text{ and } \delta_2 > 0 \\ \delta_1 & \text{otherwise} \end{cases} \tag{4.12}$$

The Quinn estimator requires adjusting $\hat{k}$ with δ. However, it has a limitation when $X_{\hat{k}}$ corresponds to the highest power provided by integer k from the DFT. This situation arises because the estimator must add the δ to $\hat{k}$, as discussed in [66, 104]. Therefore, we utilized an iterative estimator method to estimate k_{peak} instead of the non-iterative estimator.

Iterative estimator

The iterative estimator is an interpolator that, unlike non-iterative estimators, provides robust results through iterative verification and re-interpolation [1, 104]. However, iterative estimator requires the high computational cost than non-iterative estimators. In this thesis, we utilized Q-Shift Estimator (QSE), that the iterative estimator is proposed by A. Serbes [104]. This estimator is call Q-Shift Estimator (QSE). The process of this algorithm show in Algorithm 3.

Here, $X_k = \sum_{n=0}^{N-1} Y_n \exp(-j2\pi nk/N)$ allows k to extend to non-integer values by modifying it as $\hat{k} + \delta_{i-1} \pm q$. The function $c(q)$ is defined as $c(q) = \frac{1-\pi q \cot(\pi q)}{q \cos^2(\pi q)}$, with q appropriately selecting the value of the shift. The number of iterations for QSE is suggested to be 3, based on evaluations in the original QSE publication [104]. The computational cost of QSE does not exceed O($3N$), which is attributed to the transformation from the time domain to the frequency domain using the twiddle factor in Line 3 of Algorithm 3. Moreover,

Algorithm 3: Q-shift Estimator (QSE)

Input: Sliding window $\mathcal{W}$, $\hat{k}$, number of iterations Q

Output: Residual frequency δ

$\delta_0 \leftarrow 0$

for $i \leftarrow 1$ **to** Q **do**

$\quad \delta_i = \frac{1}{c(q)} \left(\Re \left\{ \frac{X_{\hat{k}+\delta_{i-1}+q} - X_{\hat{k}+\delta_{i-1}-q}}{X_{\hat{k}+\delta_{i-1}+q} + X_{\hat{k}+\delta_{i-1}-q}} \right\} \right) + \delta_{i-1}$

$\delta \leftarrow \delta_Q$

return δ

Algorithm 4: HAQSE

Input: Sliding window $\mathcal{W}$, $\hat{k}$

Output: Residual frequency δ

$q \leftarrow \frac{1}{\sqrt[3]{N}}$

$\delta_\alpha = \frac{N}{2\pi} \arcsin \left(\sin \left(\frac{\pi}{N} \right) \Re \left\{ \frac{X_{\hat{k}+0.5} + X_{\hat{k}-0.5}}{X_{\hat{k}+0.5} - X_{\hat{k}-0.5}} \right\} \right)$

$\delta = \frac{1}{c(q)} \left(\Re \left\{ \frac{X_{\hat{k}+\delta_\alpha+q} - X_{\hat{k}+\delta_\alpha-q}}{X_{\hat{k}+\delta_\alpha+q} + X_{\hat{k}+\delta_\alpha-q}} \right\} \right) + \delta_\alpha$

return δ

the value of q was suggested to be in the range between -0.5 and 0.5 from the peak DFT magnitude, as reported by Y. Liu *et al.* [65], and E. Aboutanios and B. Mulgrew [1]. Given this, A. Serbes proposed a second version called the Hybrid Aboutanios and Mulgrew and q-shift estimator (HAQSE) in the QSE publication [104]. The algorithm HAQSE is shown as Algorithm 4.

One of the key features of HAQSE is its transformation from the iterative ap-

proach of QSE to a non-iterative approach, significantly enhancing computational speed at Algorithm 4 Line 2. This transformation was proposed by J. Liao and S. Lo in [61]. By eliminating the need for repeated iterations, it speeds up the computation process. Consequently, HAQSE offers faster computations compared to QSE.

Note that there are many studies related to spectral peak location estimators, such as the Improved Quinn [66], Jacobsen estimator [47], among others. In this thesis, we utilize both the Quinn estimator and HAQSE to compare the performance between non-iterative and iterative estimators in terms of latency time and stability in estimating the season length.

4.3.3 OnlineSLE method

Here, we provide detail of the OnlineSLE method and include the pseudocode shown in Algorithms 5 and 6. The OnlineSLE is divided into two phases: the initial phase and the online phase. User must be set a sliding window size (N) and provide the time series with length N to collect in the sliding window for the initial phase. Conversely, user provides incoming data (Y_t) as input parameter for each timestamp.

During the initial setup, the user must provide a time series to fill the sliding window. If sliding window is full, OnlineSLE then calculates the periodogram ($\mathcal{P}(k)$) for the initial sliding window using FFT, which incurs a computational cost of $\mathrm{O}(N \log N)$. After computing the periodogram, the peak index $\hat{k}$ is identified using the `argmax` function. In Line 7, OnlineSLE employs a spectral peak location estimator function (`SPLE`) to pinpoint the actual frequency where it has the highest value in the periodogram. Note that both

Algorithm 5: OnlineSLE: offline phase

Input: Sliding window size (N), $Y = (Y_1, Y_2, \ldots, Y_N)$

Output: Initial season length (m)

1 **begin**

2 $\quad \mathcal{W} \leftarrow Y$

3 $\quad \mathcal{F}(k) \leftarrow \texttt{FFT}(\mathcal{W})$

4 $\quad \mathcal{P}(k) \leftarrow \|\mathcal{F}(k)\|^2$

5 $\quad \hat{k} \leftarrow \texttt{argmax}_k(\mathcal{P}(k))$

6 $\quad f_{peak} \leftarrow \texttt{SPLE}(\hat{k})$

7 $\quad m \leftarrow (1/f_{peak})$

8 $\quad$ **return** m

OnlineSLE and ASTD[1] utilize the HAQSE method for the `SPLE` function, although it is possible to substitute this with the Quinn estimator or to omit the `SPLE` function. Following this, the initial season length (m) is estimated by taking the reciprocal of this frequency.

In the online phase, OnlineSLE produces m_t for each incoming data point by processing data within the current sliding window. This phase is similar to the initial phase, with the primary difference being the computation of the periodogram using SDFT instead of FFT (Line 15). In this phase, OnlineSLE recalculates the periodogram by adding the incoming data point (Y_t) and subtracting the oldest data point Y_{oldest} from the previous periodogram results. Following this update, OnlineSLE continues with processes similar to those in the initial phase, including the use of the `SPLE` function and calculating the season

[1]ASTD is our proposed method for online seasonal-trend decomposition, which integrates with OnlineSLE. Details of this method are provided in Chapter 5.

Algorithm 6: OnlineSLE: online phase

Input: Incoming data$(Y_t), t \geq N$

Output: season length at t (m_t)

begin

- $Y_{oldest} \leftarrow \mathcal{W}[1]$
- $\mathcal{W} \leftarrow \mathcal{W}[2:N]$
- $\texttt{append}(\mathcal{W}, Y_t)$
- $\mathcal{F}(k) \leftarrow \texttt{SDFT}(\mathcal{F}(k), Y_t, Y_{oldest})$
- $\mathcal{P}(k) \leftarrow \|\mathcal{F}(k)\|^2$
- $\hat{k} \leftarrow \texttt{argmax}_k(\mathcal{P}(k))$
- $f_{peak} \leftarrow \texttt{SPLE}(\hat{k})$
- $m_t \leftarrow (1/k_{peak})$
- **return** m_t

length (m_t) by taking the reciprocal of the frequency result from the spectral peak location estimator.

4.4 Datasets

In our experiments, we evaluated the performance of our OnlineSLE and existing SLE methods using both synthetic and real-world datasets. To ensure consistency and comparability across different datasets, we applied Z-normalization to standardize the data. Additionally, we removed any linear trends using the detrend function from the SciPy

library[2], applied to the entire time series. This preprocessing step helps to minimize the influence of non-stationary elements in the data, thereby enhancing the accuracy of SLE methods under comparison.

4.4.1 Synthetic Datasets

We employed two synthetic datasets to evaluate the performance. These datasets were generated different patterns for seasonal components and various noise conditions.

- **Syn1**: This dataset features a consistent square wave pattern where each cycle spans a length of 100 timestamps, with the season length remaining constant throughout the dataset.

- **Syn2**: This dataset comprises three distinct seasonality phases, each modeled with a sine wave pattern. The season lengths for these phases are set at 50, 80, and 50 timestamps, respectively, with transitions occurring at 1,800 and 3,600 timestamps.

To further evaluate the resilience of SLE methods, both datasets were subjected to varying levels of Gaussian noise. Noise was introduced at four different standard deviation levels: 0.05σ, 0.10σ, 0.50σ, and 0.75σ. Here, σ represents the standard deviation of the original time series before noise injection. This procedure generated a total of eight synthetic datasets, that are shown in Figure 4.9.

[2]https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.detrend.html

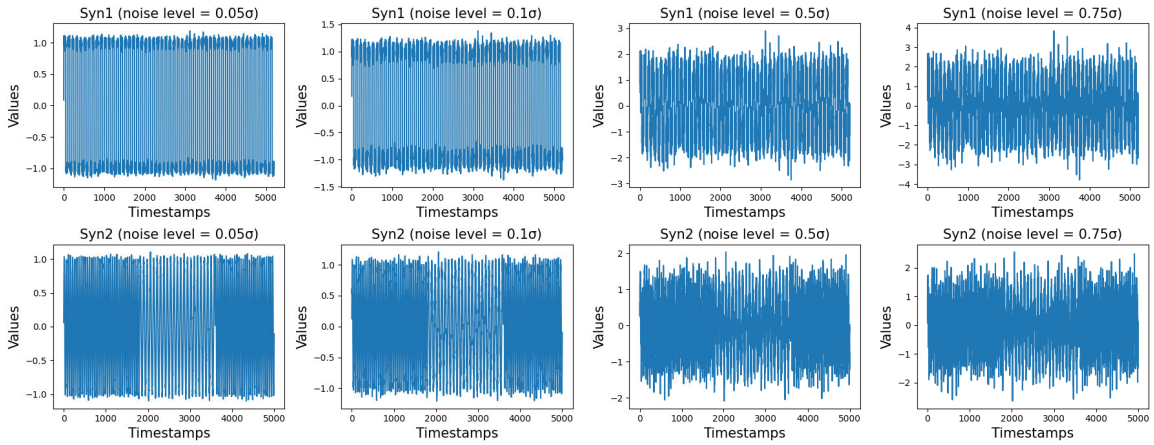


Figure 4.9: Visualization of the synthetic datasets Syn1 and Syn2, demonstrating the different seasonality patterns and noise levels.

4.4.2 Real-world Datasets

In our evaluations, we utilized three real-world datasets: the Sunspots, Electrocardiograms (ECG), and Arterial Blood Pressure (ABP) datasets. The Sunspots dataset records the monthly average number of sunspots from 1794 to 2023, with the average cycle length being approximately 11 years, as shown in Figure 4.1 (Bottom) [112]. The ECG and ABP datasets comprise recordings from electrocardiograms and arterial blood pressure, collected from a healthy volunteer who was positioned on a tilt table with foot support [32, 37]. Illustrations of ECG and ABP datasets are shown in Figures 4.10 and 4.11. That these data were subjected to Z-normalization and detrending across the entire time series. However, it is possible that portions of the trend may still be contained within the time series.

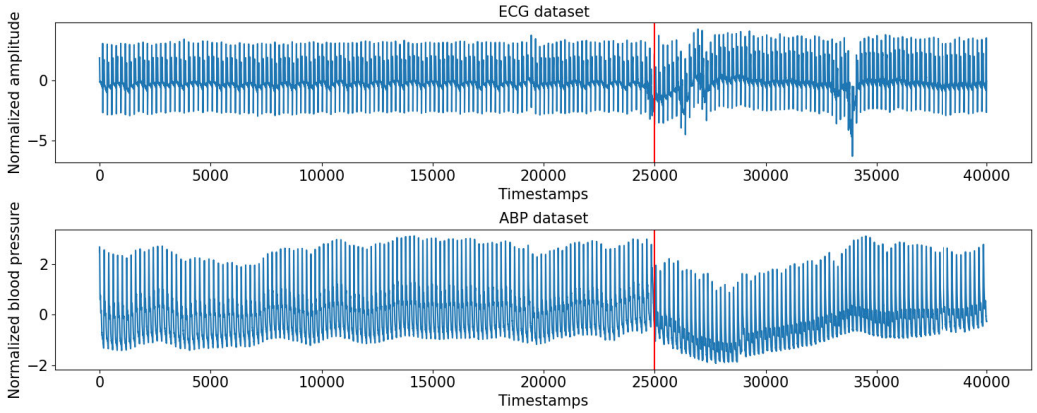


Figure 4.10: The real-world datasets, where the red line indicating the point of rapid rotation of the tilt table affecting the volunteers. (Top) ECG dataset; (Bottom) ABP dataset.

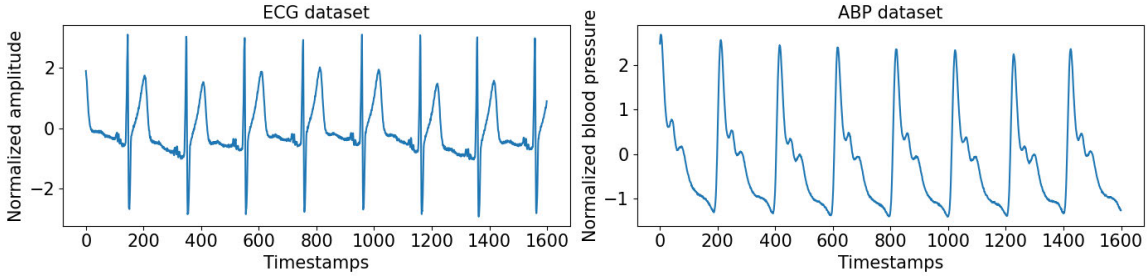


Figure 4.11: First 1600 timestamps of the real-world datasets from Fig. 4.10. (Left) ECG dataset; (Right) ABP dataset.

4.5 Experimental Setting

4.5.1 Comparison Methods

In our study, we initially focused on comparing periodogram implementations using FFT and SDFT. For FFT computations, we utilized the implementation available in the NumPy[3] and SciPy[4] library.

[3]https://numpy.org/doc/stable/reference/generated/numpy.fft.fft.html
[4]https://docs.scipy.org/doc/scipy/reference/generated/scipy.fft.fft.html

Shifting our focus to SLE methods, we compared OnlineSLE with existing SLE methods to assess performance. These methods were implemented using Python 3.9.2 and R 4.3.0. To ensure a fair comparison, all methods, including AutoPERIOD [120], SAZED [118], and FindLength [79], were adapted to operate in an online mode using a sliding window approach.

For the spectral peak location estimators, we re-implemented the Quinn [93], QSE [104] , and HAQSE [104] estimators for OnlineSLE in Python, as the original source codes were available only in Matlab. Hereafter, we refer to OnlineSLE-none, OnlineSLE-Quinn, OnlineSLE-QSE, and OnlineSLE-HAQSE as OnlineSLE without an estimator, OnlineSLE with the Quinn estimator, OnlineSLE with the QSE, and OnlineSLE with the HAQSE, respectively.

4.5.2 Evaluation Metrics

As previously discussed, the primary objectives of OnlineSLE are fast and accuracy in SLE. To quantitatively evaluate performance, we considered two metrics: latency time and accuracy.

Latency Time Metric

To measure the latency time for each sliding window, we recorded the processing latency from the moment the incoming data was received until the method provided the season length for each sliding window. The pseudocode for recording latency time is shown in Algorithm 7.

Algorithm 7: Latency time of SLE method during the online phase for each timestamp

Input: Incoming data$(Y_t), t \geq N$

Output: latency time at t

begin

$Y_{oldest} \leftarrow \mathcal{W}[1]$

$\mathcal{W} \leftarrow \mathcal{W}[2:N]$

`append`$(\mathcal{W}, Y_t)$

start_time $\leftarrow$ `get_time`() `// Record the start time`

$m_t \leftarrow$ `SLEmethod`$(\mathcal{W})$

end_time $\leftarrow$ `get_time`() `// Record the end time`

latency $\leftarrow$ end_time $-$ start_time

return latency

From this pseudocode, we recorded the latency time for each sliding window and then computed the average. A lower average latency indicates enhanced computational efficiency, reflecting a faster method that is more suitable for real-time applications.

Accuracy Rate Metric

For the accuracy evaluations, we considered three phases: numerical accuracy evaluation between FFT and SDFT, accuracy evaluation by various spectral peak location estimator of OnlineSLE, and accuracy evaluation by various SLE methods.

Firstly, we conducted an evaluation of the numerical computation error between

results using FFT and SDFT for each sliding window. This evaluation, based on the two studies cited in [26, 53], evaluates the potential deviations in the numerical error of the SDFT from the FFT. In this phase, we measured the mean absolute error (MAE) between the FFT and SDFT results across each sliding window. A lower MAE indicates negligible differences between FFT and SDFT results.

The second phase evaluates the accuracy of different estimators used in OnlineSLE. We utilized the estimation results to determine whether they exactly match the ground truth season length, with variations in the sliding window size serving as an input parameter.

In the third phase, our focus shifted to a broader comparison across various SLE methods. We tallied the number of sliding windows where the estimated season length matched the ground truth for each dataset. The accuracy ratio was then calculated by dividing the number of correct estimations by the total number of sliding windows evaluated for each dataset. To ensure a comprehensive evaluation of the accuracy rate in third phase, we considered two criterion settings:

- $\pm 0\%$ (Exact Match): Under this criterion, an estimation is deemed accurate only if it exactly match with the ground truth season length.

- $\pm 20\%$: This more lenient setting allows for an error bound, deeming an estimation correct if it falls within a bound of $\pm 20\%$ of the ground truth season length. Note that we set the error bound at $\pm 20\%$ based on the evaluation in [118].

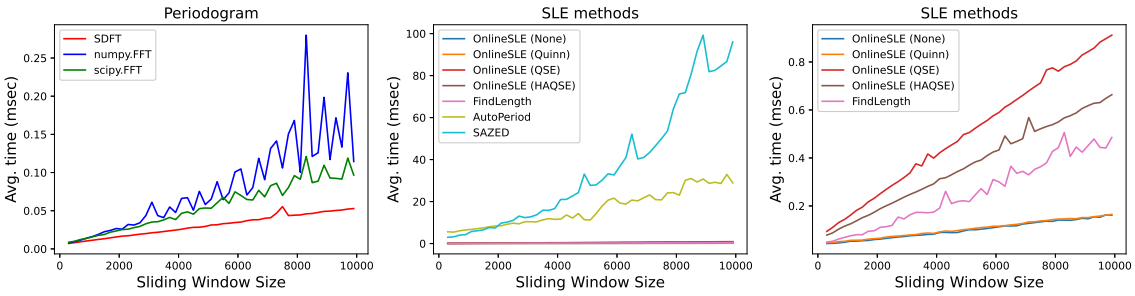


Figure 4.12: Comparison of time performance for different SLE methods.

4.6 Experimental Results

4.6.1 Time Performance Evaluation

Time performance evaluation was conducted on a MacBook Air (2022) with 8 CPU cores (Apple M2) and 16 GB of memory. The key parameter for all SLE methods is the sliding window size, which influences the time complexity. For example, the time complexity of OnlineSLE is $O(N)$, while AutoPeriod is $O(N \log N)$, where N denotes the sliding window size. Although the sliding window sizes varied, we set the total number of sliding windows to 10,000 for each scenario to ensure a fair comparison.

The results of latency times for various methods and sliding window sizes are shown in Figure 4.12. Overall, all methods exhibit the same trend: as the sliding window size increases, the computational cost also increases.

The periodogram computations using FFT and SDFT were particularly noteworthy, as shown in Figure 4.12 (Left). FFT from SciPy is faster than NumPy because SciPy uses Bluestein's algorithm, which ensures computation is never worse than $O(N \log N)$ [10]. Comparing SDFT and FFT, the results demonstrate that periodogram computations using

SDFT are faster than those using FFT from both NumPy and SciPy. This suggests that SDFT achieves a significant reduction in time complexity. Note that FFT is most efficient when N is a power of two due to symmetries in the calculated terms of Cooley-Tukey algorithm [20].

Analysis indicated that SAZED and AutoPeriod required more computation time compared to OnlineSLE. AutoPeriod relies on both the periodogram and ACF, which depend on FFT and iFFT. Similarly, SAZED relies on these components, in addition to the average of time series zero distances. Consequently, both AutoPeriod and SAZED operate at $\mathrm{O}(N \log N)$, which is slower than OnlineSLE with the $\mathrm{O}(N)$ computational cost.

OnlineSLE-None and FindLength demonstrated better efficiency in computation time compared to the other methods evaluated. This efficiency is particularly evident when processing data in sliding windows.

Turning to various spectral peak location estimator, OnlineSLE-None achieved faster computation compared to versions that incorporate such estimators. This is primarily because adding an estimator generally increases the computational cost. Among the estimators, the Quinn estimator, with a computational cost of $\mathrm{O}(1)$, showed better efficiency in computation time compared to QSE, which has a computational cost of $\mathrm{O}(3N)$ due to its iterative approach. HAQSE provides faster computation than QSE, which is HAQSE's hybrid version. Notably, with these variations in estimators, OnlineSLE performs faster than both SAZED and AutoPeriod. We summarized the computational cost for each method in Table 4.1.

Table 4.1: Computational cost of each method

	DFT	ACF	SPLE
AutoPeriod	O($N \log N$)	O($N \log N$)	-
SAZED	O($N \log N$)	$O(N \log N)$	-
FindLength	-	O($N \log N$)	-
OnlineSLE (None)	O(N)	-	-
OnlineSLE (Quinn)	O(N)	-	O(1)
OnlineSLE (QSE)	O(N)	-	O($3N$)
OnlineSLE (HAQSE)	O(N)	-	O(N)

4.6.2 Error in Numerical Computation of SDFT Evaluation

This subsection introduces a numerical computation error of the SDFT, specifically concerning the approximation of twiddle factors in its recursive implementation. This concerning was highlighted by J. Kim and T. Chang [53].

Unlike the FFT, which computes each transformation by applying the twiddle factor to the original time series data only once in the time domain, the SDFT multiplies twiddle factors with previous DFT results at each timestamp. Consequently, in SDFT, results are multiplied many times with twiddle factors through a recursive approach. It can introduce deviations from the actual DFT results, potentially leading to inaccuracies. To investigate these deviations, we conducted a comparative analysis of the numerical computations between FFT and SDFT using our synthetic datasets with noise level 0.05σ. Figure 4.13 shows an error graph that quantifies the numerical errors for various datasets.

The graph shows that the numerical error for SDFT is consistently on the order of 10^{-14}, with an average computation error of 0.6864×10^{-14}. Significantly, the analysis shows that these errors do not increase, even with the repetitive recalculations inherent in the sliding window strategy employed by SDFT. Thus, these findings support the use of

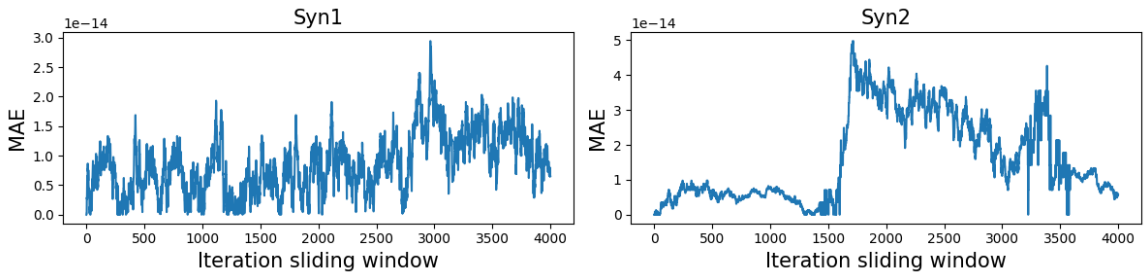


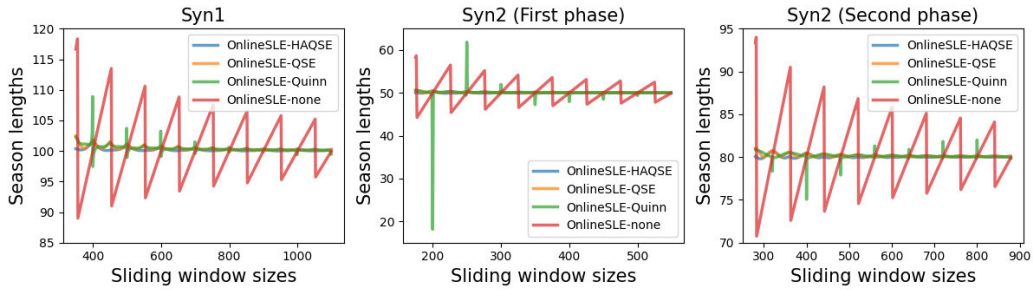
Figure 4.13: Numerical error computation between FFT and SDFT by various datasets. (Left) Syn1; (Right) Syn2.

SDFT over FFT, due to its minimal numerical error and fast computation capabilities.

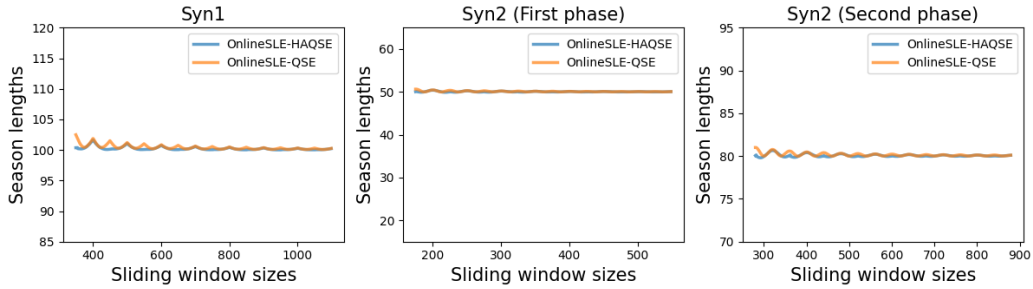
4.6.3 Accuracy by Spectral Peak Location Estimator of OnlineSLE

We conducted an evaluation of the accuracy of OnlineSLE using various spectral peak location estimators and different sliding window sizes. The goal of this evaluation is to demonstrate how the size of the sliding window influences frequency resolution. We utilized the Syn1 and Syn2 datasets with noise level 0.05σ for this analysis. Note that Syn2 were sepeated each phase for evaluation in this subsection. The results are shown as Figure 4.14, where the x-axis represents the window sizes and the y-axis represents the season length results provided by the estimator.

The window size is considered to be the optimal window size for accurately determining k_{peak} if the window size divided by a positive integer k_{peak} equals the ground truth. For example, consider Syn1 with a ground truth season length of 100. If the window size is 400 and k_{peak} is 4, the frequency of this k_{peak} is 0.01, and taking the reciprocal of this frequency suggests a season length of 100, which matches the ground truth. However, if the



(a) All estimators



(b) QSE vs HAQSE

Figure 4.14: Comparison of accuracy results with synthetic datasets by various estimators and sliding window size. (Left) the accuracy results with Syn1 (Ground truth = 100); (Middle) the accuracy results with the first phase of Syn2 (Ground truth = 50); (Right) the accuracy results with the second phase of Syn2 (Ground truth = 80).

window size is 399, no positive integer for k_{peak} can map to the ground truth, as explained in Section 4.2.2.

The window size is considered to be the optimal window size for accurately determining k_{peak} if window size divided by a positive integer k_{peak} equals ground truth. For example, consider Syn1 (Ground truth = 100), the window size is 400 and k_{peak} is 4 the frequncy of this k_{peak} is 0.01 and taking the reciprocal of this frequency suggests a season length 100 that match with ground truth. However, if the window size is 399 any positive

integer k_{peak} cannot map to the ground truth, as explained in Section 4.2.2.

The season length determined by OnlineSLE-none is unstable and often deviates significantly from the ground truth. However, using the optimal window size can yield the correct season length that exactly matches the ground truth. The results from OnlineSLE-Quinn were more stable than those from the non-estimator version, but they still failed to provide the correct season length even with the optimal window size. This limitation is due to the nature of the Quinn estimator, a non-iterative estimator, which does not verify whether the k_{peak} is the actual peak, this limitation also discussed in [66]. Both QSE and HAQSE exhibited stable results, independent of window size. However, HAQSE provided results that were closer to the ground truth compared to QSE, as shown in Figure 4.14b. Therefore, we utilized HAQSE to mitigate the influence of window size on season length estimation.

4.6.4 Accuracy Rate Evaluation with Synthetic Datasets

We conducted an evaluation to compare the accuracy rates of various SLE methods using synthetic datasets. The optimal window sizes for all algorithms were set based on preliminary tests: 500 instances for Syn1 and 400 instances for Syn2. Note that in subsequent discussions, OnlineSLE will be referred to as including the HAQSE estimator to mitigate the influence of window size on DFT results. The outcomes of these evaluations are presented in Tables 4.2 and 4.3.

All methods successfully estimated the season length for the Syn1 dataset, as the season length was kept constant. However, the accuracy rate of FindLength was signif-

Table 4.2: Comparison of accuracy rate results with **Syn1** by various methods.

Methods	Accuracy rate							
	±0% setting				±20% setting			
	0.05σ	0.10σ	0.50σ	0.75σ	0.05σ	0.10σ	0.50σ	0.75σ
AutoPeriod	1.000	1.000	0.923	0.629	1.000	1.000	1.000	1.000
FindLength	1.000	1.000	0.912	0.465	1.000	1.000	0.989	0.698
SAZED	1.000	1.000	0.000	0.000	1.000	1.000	1.000	1.000
OnlineSLE	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Table 4.3: Comparison of accuracy rate results with **Syn2** by various methods. The bolded and underlined results represent the highest accuracy rate across SLE methods.

Methods	Accuracy Rate							
	±0% setting				±20% setting			
	0.05σ	0.10σ	0.50σ	0.75σ	0.05σ	0.10σ	0.50σ	0.75σ
AutoPeriod	0.696	0.694	0.369	0.310	**0.911**	0.911	0.906	0.897
FindLength	0.683	0.681	0.342	0.212	0.875	0.873	0.837	0.594
SAZED	0.826	0.647	0.546	0.506	0.908	0.908	0.909	0.909
OnlineSLE	**0.910**	**0.913**	**0.920**	**0.920**	0.910	**0.913**	**0.920**	**0.920**

icantly impacted by increased noise levels, as shown in Table 4.2. However, FindLength exhibited lower accuracy in estimating season lengths when faced with heavy noise. This issue is discussed in more detail in Section 4.2.3, which it relies on only the ACF to analyze data in the time domain. These results clearly demonstrate OnlineSLE's advantage over FindLength, as OnlineSLE maintains a higher accuracy rate while providing equally fast computation.

Referring to Table 4.3, it is evident that SAZED and AutoPeriod often achieved second or third-highest accuracy rates. However, the computational cost for these methods are substantially greater than for OnlineSLE. As for OnlineSLE, it consistently surpassed other SLE methods in terms of both accuracy and computational efficiency.

Table 4.4: Experimental results with real-world datasets by various methods. The bolded and underlined results represent the highest accuracy rate across SLE methods.

Method	Accuacy rate					
	Sunspots dataset		ECG dataset		ABP dataset	
	±0%	±20%	±0%	±20%	±0%	±20%
AutoPeriod	0.126	0.946	0.058	**0.963**	0.018	0.591
FindLength	0.126	0.946	0.053	0.928	0.018	**1.000**
SAZED	0.000	**0.965**	0.000	0.958	0.016	0.986
OnlineSLE	**0.824**	0.921	**0.867**	0.867	**0.741**	0.811

4.6.5 Accuracy Rate Evaluation with Real-world Datasets

Similar to Section 4.6.3, we evaluated the performance of various SLE methods in terms of accuracy rates. For this evaluation, we used the Sunspots, ECG, and ABP datasets with ground truth season lengths of 132 (for 11 years), 200 and 210, respectively. The results of these evaluations are presented in Tables 4.4.

Overall, OnlineSLE achieved a high accuracy rate in the ±0% setting. This performance is largely due to its handling of residual trends remaining after trend filtering, which significantly affect ACF calculations. In contrast to existing SLE methods, OnlineSLE analyzes data exclusively in the frequency domain, thereby avoiding the impact of residual trends. While other SLE methods may demonstrate superior performance in the ±20% setting, the accuracy in the ±0% setting is particularly advantageous for actual analysis.

It should be noted that the accuracy rates of existing methods evaluated in this study may drop significantly in a streaming environment. This decline is due in part to the preprocessing approach employed, which involved the application of trend filtering to the entire dataset. This approach is not practical for streaming data.

4.7 Discussion and Conclusions

In conclusion, we have introduced OnlineSLE, a fast and accurate method for online season length estimation. This method operates by analyzing the time series exclusively in the frequency domain. Additionally, we have introduced a novel solution to address frequency resolution issues that incorporate a spectral peak location estimator with the periodogram. This solution achieved fast computation and higher accurate than existing SLE methods, which rely on ACF.

Future work will focus on two directions: numerical error computation in SDFT and reducing the computational cost of OnlineSLE. The first direction is related to an approximation of twiddle factors in recursive SDFT as explained in Section 4.6.2. Further insights into SDFT structures are provided by A. Chayhan and K.M. Singh, who reviewed various adaptations of SDFT, including modulated-SDFT, Douglas&Soh-SDFT, and observer-based-SDFT [16]. Their review highlights the advantages and disadvantages of each structure, discussing aspects such as numerical error computation, noise performance, and stability analysis. Moreover, R. Lyons[5] and C. Howard have proposed enhancements to stabilize SDFT in recently [70], suggesting that further investigation into different SDFT structures could enhance our OnlineSLE. These findings highlight existing gaps and potential improvements for enhancing the robustness of our OnlineSLE using different SDFT structures.

[5]R. Lyons is a co-author of the original publication on SDFT [48].

Chapter 5

Adaptive Seasonal-trend Decomposition

This chapter describes the Adaptive Seasonal-trend Decomposition (ASTD), a method for real-time time series decomposition into trend, seasonal, and residual components. The primary goal is to adaptively handle transitions and fluctuations in seasonality that may continuously change over time.

We organize this chapter into seven sections to enhance clarity and depth. Section 5.1 provides the necessary background on the seasonal-trend decomposition (STD). Section 5.2 reviews related work and identifies remaining gaps in existing methods. Section 5.3 explains the methodology behind ASTD. Subsequently, Sections 5.4 and 5.5 detail the datasets and experimental settings used. Section 5.6 presents the results of our experiments. Finally, Section 5.7 offers the remaining challenge, conclusions, and outline directions for future research.

5.1 Background

Building on the foundational knowledge from Section 2.2.3, the season length is a critical parameter for STD. The existing STD methods request users predefine this parameter before decomposition in offline and online modes, such as STL[19], RobustSTL[126], OnlineSTL[73], and more. By predefining this parameter, it remains the two challenges.

The first challenge involves the predefined season length set by users, which must closely align with the actual season length to ensure accurate decomposition results. The accuracy of the decomposition heavily depends on selecting the correct season length. Using an incorrect season length can lead to inaccurate results, as demonstrated in Figures 2.9 and 2.10. This inaccuracy has significant implications, potentially causing scientists to make erroneous analyses of the phenomena under study, which could impact research conclusions and applications.

The second challenge arises from using a fixed season length as an input parameter to decompose trend and seasonal components. In streaming data, where seasonality may dynamically change over time due to unstable behavior, a fixed parameter might not effectively capture these variations. This is particularly problematic when the actual season length deviates from the predefined length set by the user, potentially leading to inaccuracies in capturing the seasonal component.

To provide a more concrete understanding of this issue, consider the time series shown in Figure 5.1 (Top-left). We applied the STL method [19], setting the season length to match the first phase of seasonality, specifically a length of 50 from timestamps 1 to 300. In the subsequent subfigure in Figure 5.1, we observe that while STL can decompose the trend

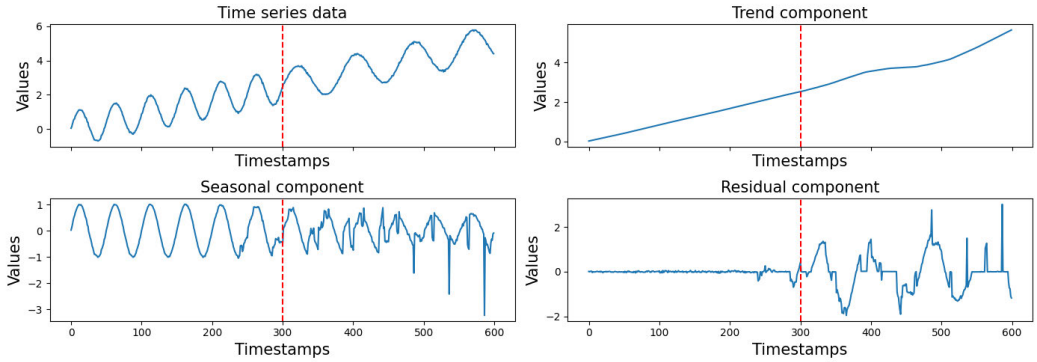


Figure 5.1: STD results on a time series with seasonality changes marked by a red line at timestamp 300, highlighting the challenges posed by fixed seasonal lengths.

and seasonal components with initial stability, it fails to maintain this stability throughout the series. This instability arises because the predefined season length corresponds only to the initial phase, failing to accommodate dynamic changes due to unstable behavior in the seasonal component that occur later in the series.

The unstable behaviors in the seasonal component can be divided into two types: seasonality transitions and fluctuations.

Seasonality transitions refer to changes in the durations of the repeating cycles within time series, which often happen with changes in activity or phenomena. Figure 5.2 (Left) illustrates a dataset with transition, where the red line indicates human activity changing from walking to jogging [32].

Seasonality fluctuations involve variations in the durations of the repeating cycles due to temporal factors. For example, Figure 5.2 (Right) shows the population of Canadian lynx, where the season length deviates by decade in accordance with food availability [14, 44].

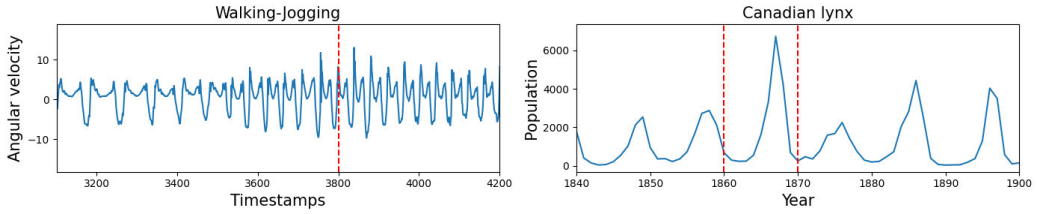


Figure 5.2: Samples of seasonality transitions and fluctuations: (Left) At the 3800 timestamps, a seasonality transition leads to a significantly shorter season length in subsequent cycles. (Right) there was a fluctuation in seasonality between 1860 and 1870, characterized by a season length that was shorter than a decade.

These transitions and fluctuations can be observed in various time-series data. The next section delves into related works in STD to discuss the advantages and disadvantages of each method. Additionally, it outlines the remaining challenges for STD.

5.2 Related Work

5.2.1 Single vs. Multiple seasonal component

In this thesis, STD breaks down time series into trend, seasonal, and residual components. However, numerous studies have explored multiple seasonal-trend decomposition (MSTD) [5, 25, 73, 129]. This subsection discusses the differences between time series models that use single STD and those that use MSTD.

The primary focus of this thesis is the time series with a single seasonal component. This series can be expressed as $Y = T + S + R$, where where Y denotes the time series, T denotes the trend component, S denotes the seasonal component, and R denotes the

residual component, as defined in Definition 2 of Section 2.1.

For MSTD, the time series can be expressed as $Y = T + \sum_{p=1}^{q} S_p + R$, where S_p denotes the p-th seasonal components, $\sum_{p=1}^{q} S_p$ is the sum of multiple seasonal component. In cases involving a single seasonal component, q must be one. Note that in this chapter, we only consider to decompose the time series with a single seasonal component.

Moreover, numerous studies have modified time series components to accommodate domain-specific needs, such as incorporating a holiday effects component, as described in the paper [114], or separating the trend component into long-term and short-term trends, as described in the paper [67].

5.2.2 Existing STD methods

STD algorithms are categorized into offline and online modes. In offline mode, the entire time series is available from the outset, allowing the method to comprehensively process and decompose it into trend, seasonal, and residual components. Conversely, online mode is required to process each incoming data with observed data within the sliding window or buffer. The online mode is suitable for real-time decomposition.

Offline STD methods

The classical STD method, Seasonal-Trend decomposition using LOESS (STL), is straightforward to implement and does not require training data [19]. However, STL faces challenges with long season lengths and high noise levels due to its inherent inflexibility. An enhanced version designed to decompose multiple seasonal components is the Multiple Seasonal-Trend decomposition using LOESS (MSTL) [5]. Despite its enhancements, MSTL

still shares some of the core limitations of STL, particularly in terms of flexibility and handling noise effectively.

An alternative for STD that does not use LOESS is based on regularized optimization, which aims to minimize error values to better fit the model to the time series. A. Dokumentov and R.J. Hyndman proposed the Seasonal-Trend Decomposition using Regression (STR) [25], which employs ridge regression for this purpose [55]. This method offers increased flexibility in handling fluctuations in seasonality. However, while STR enhances the adaptability to seasonality changes, it does not comprehensively address the challenges posed by transitions in season length. This is because it requires a predefined season length as an input parameter beforehand. Additionally, the computational cost associated with STR is relatively high.

The robust STD method for offline mode is RobustSTL [126], which is based on regularized optimization using an ℓ_1-norm regularizer for trend filtering [54]. This filtering approach is specifically designed to effectively handle both abrupt and gradual trend changes in time series data, where abrupt changes refer to sudden and significant directional shifts. Additionally, RobustSTL can adeptly manage fluctuations in seasonality. An enhanced version is FastRobustSTL [129]. It significantly reduces the time complexity for regularized optimization and is capable of decomposing multiple seasonal components. While these methods offer robust results, they require the specification of more parameters compared to STL and STR. These include a neighborhood interval for input parameter to monitor seasonal fluctuations and parameters to adjust the roughness and smoothness of the trend—'rough' to address abrupt changes and 'smooth' for gradual changes. Properly

configuring these parameters is crucial for high-quality decomposition but can be time-consuming.

Online STD methods

The first online STD method, known as OnlineSTL, was introduced by A. Mishra, R. Sriharsha, and S. Zhong [73]. This method employs a kernel trend filter based on LOESS and an exponential smoothing filter [45] for real-time decomposition. These filters apply a weighting function that decreases for past observations, thus smoothing the trend and seasonal components while emphasizing more recent observations. OnlineSTL achieves computational efficiency with a complexity of $O(N)$ for each time decomposition, where N denotes the length of time series data within the sliding window. However, it fails to effectively handle seasonality transitions and fluctuations.

More recently, X. He *et al.* proposed a method for Online STD that reduces the computational cost to O(1) per decomposition, named OneShotSTL [35]. This method offers higher-quality decomposition results by incorporating an adaptive ℓ_1-norm regularizer, improving upon OnlineSTL. However, like FastRobustSTL, OneShotSTL's ℓ_1-norm regularizer also requires specific input parameters, which can be a limitation. The following describes the main limitation of those methods that correspond to season length as an input parameter.

Influence of the season length for STD

The methods previously proposed typically require a predefined season length as an input and often lack the adaptive capabilities necessary for handling transitions and

Table 5.1: STD Methods Comparison.

Methods	Online	MSTD	Seasonality transitions	Seasonality fluctuations	Season length free
STL	No	No	No	No	No
MSTL	No	Yes	No	No	No
STR	No	Yes	No	Yes	No
RobustSTL	No	No	No	Yes	No
FastRobustSTL	No	Yes	No	Yes	No
OnlineSTL	Yes	Yes	No	No	No
OneShotSTL	Yes	No	No	Yes	No
ASTD	**Yes**	**No**	**Yes**	**Yes**	**Yes**

fluctuations in seasonality. As discussed in Section 2.2.3, the choice of season length significantly impacts the quality of decomposition results. An improper season length can lead to low-quality decomposition, where residual trends may appear in the seasonal component, or residuals of repeating cycles may be misclassified as trend or residual components. Such inaccuracies can lead to erroneous analyses by scientists, compromising the reliability of scientific studies.

Our objective for a novel STD method includes eliminating the need for a predefined season length as an input parameter and enabling online STD to support real-time monitoring. To address this challenge, we integrate our OnlineSLE method with online STD, allowing for the dynamic adjustment of the season length during decomposition. The proposed method improves the accuracy of online STD without requiring manual entry of the season length. We summarize the comparisons of STD methods in Table 5.1.

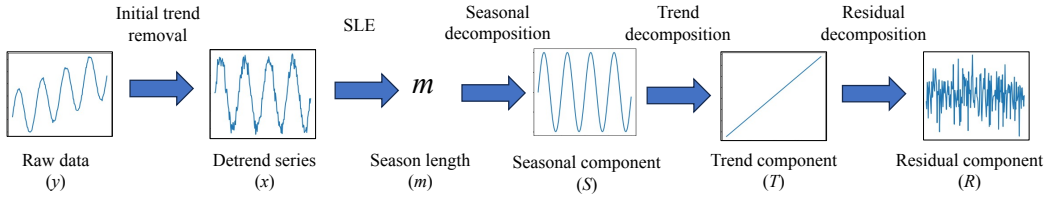


Figure 5.3: Procedure of the initialization phase

5.3 Adaptive Seasonal-Trend Decomposition

We divided the ASTD into two phases: the initialization and online phases, which are similar to the existing online STD method [35, 73]. The initialization phase operates in offline mode, decomposing the data within $\mathcal{W}$ to prepare all necessary arrays for maintaining online decomposition. The online phase involves iteratively decomposing each latest data point, Y_t. The procedures for both phases are illustrated in Figure 5.3.

5.3.1 Algorithm

Initialization phase

In the initialization phase, we collect the time series Y with N elements in the sliding window $\mathcal{W}$, where N denotes the sliding window size. ASTD then decomposes $\mathcal{W}$ into its components in a manner similar to the procedure depicted in Figure 5.3. The details are as follows:

1. **Initial trend removal:** Use the trend filter to decompose the initial trend from each element in $\mathcal{W}$, collecting the results into $\mathcal{T}1$. Then, calculate the difference between $\mathcal{W}$ and $\mathcal{T}1$ to obtain the detrended series, $\mathcal{DT}$.

2. **SLE:** Estimate m using the OnlineSLE method, which is explained in the pseudocode in Section 4.3. Note that OnlineSLE utilizes FFT to compute the periodogram in offline mode.

3. **Seasonal decomposition:** Use the seasonal filter to decompose the seasonal component from each element in $\mathcal{DT}$, using the determined season length m, and collect the results into S.

4. **Trend decomposition:** Calculate the difference between $\mathcal{W}$ and S to obtain the deseasonalized series $\mathcal{DS}$. Then, use the trend filter to each element in $\mathcal{DS}$ and collect the trend component into T.

5. **Residual decomposition:** Decompose the residual component as $R = \mathcal{W} - T - S$.

Online phase

For the online phase, the procedure is detailed in Algorithm 8. Note that the function '`update`(Array, newdata)' removes the oldest data of the array and appends new data into the array, functioning similar to a first-in, first-out process. The steps for decomposing Y_t into three components at timestamp t are as follows:

1. **Initial trend removal:** Update $\mathcal{W}$ with Y_t, then use a trend filter to decompose the initial trend from the data within $\mathcal{W}$, obtaining $\mathcal{T}1_t$. Calculate the difference between Y_t and $\mathcal{T}1_t$ to obtain the detrended data at t ($\mathcal{DT}_t$).

2. **SLE:** Estimate m using the OnlineSLE method by provided $\mathcal{DT}_t$. Note that OnlineSLE utilizes SDFT to compute the periodogram in online mode.

Algorithm 8: ASTD: Online Phase

Input: Time series Y_t, where $t > N$

Output: T_t, S_t, R_t

begin

$\mathcal{W} \leftarrow \texttt{update}(\mathcal{W}, Y_t)$

$\mathcal{T}1_t \leftarrow \texttt{TF}(W)$ `// Decompose initialization trend`

$\mathcal{DT}_{oldest} \leftarrow \mathcal{DT}[1]$

$\mathcal{DT}_t \leftarrow Y_t - \mathcal{T}1_t$

$\mathcal{DT} \leftarrow \texttt{update}(\mathcal{DT}, \mathcal{DT}_t)$

$m \leftarrow \texttt{OnlineSLE}(\mathcal{DT}_t)$ `// OnlineSLE (online phase)`

$S_t \leftarrow \gamma \mathcal{DT}_t + (1-\gamma) S_{t-m},$ `// Decompose seasonal component`

$S \leftarrow \texttt{update}(S, S_t)$

$\mathcal{DS} \leftarrow \texttt{update}(\mathcal{DS}, Y_t - S_t)$

$T_t \leftarrow \texttt{TF}(\mathcal{DS})$ `// Decompose trend component`

$R_t \leftarrow Y_t - T_t - S_t$ `// Decompose residual component`

return T_t, S_T, R_t

3. **Seasonal decomposition:** Apply the seasonal filter to $\mathcal{DT}_t$ using the determined m to decompose the seasonal component at t (S_t), then update S with S_t.

4. **Trend decomposition:** Update $\mathcal{DS}$ with $Y_t - S_t$, and then use a trend filter to decompose the trend component at timestamp t, collecting the result into T_t.

5. **Residual decomposition:** Decompose the residual component at timestamp t as $R_t = Y_t - T_t - S_t$.

5.3.2 Trend Filter

The robust filter used for decomposing the trend component often utilized the ℓ_1-norm regularizer for trend filtering, as seen in [35, 125, 126, 129]. However, this method relies heavily on the specification of input parameters, which can limit its adaptability, as detailed in Section 5.2.2. To enhance flexibility, we utilize a trend filter based on tricube kernel smoothing, similar to the approach used in STL and OnlineSTL [19, 73]. This filter decomposes the trend component of the time series using tricube kernel smoothing. The tricube weight function, crucial for this smoothing process, is defined as follows:

$$w(u) = \begin{cases} (1-u^3)^3 & 0 \leq u < 1 \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

In our setting, we define that the latest element of $\mathcal{W}$ receives the highest weight, whereas the oldest element of $\mathcal{W}$ receives the lowest weight. This weighting is based on the assumption that incoming data (Y_t) are more indicative of the current trend and therefore should have a more substantial influence on the estimation of the trend component. We assign a weight to each element in the window $\mathcal{W}$ as $w\left(\left|\frac{i-t}{N}\right|\right)$, where N denotes the length

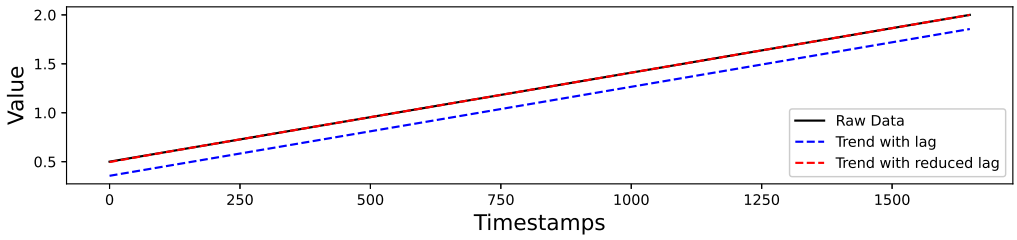


Figure 5.4: Comparison of trend results. Note that red line overlaps with the black line.

of $\mathcal{W}$, and i denotes the index in $\mathcal{W}$ for $t - N + 1 \leq i \leq t$. The trend filter via this weight function is expressed as:

$$\texttt{TF}_1(\mathcal{W}) = \frac{\sum_{i=t-N+1}^{t} w\left(\left|\frac{i-t}{N}\right|\right) \cdot Y_i}{\sum_{i=t-N+1}^{t} \left|w\left(\left|\frac{i-t}{N}\right|\right)\right|} \tag{5.2}$$

where Y_i denotes the i^{th} element in W. However, the trend filtered using tricube kernel smoothing lags the actual trend because it cannot fully eliminate the influence of older data points within $\mathcal{W}$ [73, 95]. To reduce the lag in trend, the final trend calculation is adjusted to $T_t = \texttt{TF}(\mathcal{W}) = \texttt{TF}_1(\mathcal{W}) + \texttt{TF}_1(\mathcal{W} - \texttt{TF}_1(\mathcal{W}))$ [73]. For better understanding, we plot results of trend filter with and without lagging reduction as shown in Figure 5.4. From this figure, we observe that the blue line lags behind the actual trend. By applying the lag reduction, the trend filter can provide a trend that matches the actual trend, as shown by the red line overlapping with the black line.

5.3.3 Seasonal Filter

Seasonal decomposition in time series analysis utilizes various filters, such as moving averages [19], double seasonal exponential smoothing [113], and non-local seasonal filters

[126]. However, these filters often fail to accurately decompose the seasonal component when faced with transitions or fluctuations in season lengths. This issue arises because these filters typically utilize data from more than three points within a sliding window to estimate the seasonal component. For instance, the seasonal component at S_{t-2m} may not align with S_{t-m} due to variations in season lengths. To address this, we assume that S_t and S_{t-m} align over consistent intervals. Therefore, we utilize single exponential smoothing for the seasonal filter [45], which calculates the current seasonal component S_t by balancing the current detrended data point, $\mathcal{DT}_t$, with the seasonal component from the previous cycle, S_{t-m}. This filter dynamically estimates the seasonal component and adapts to potential transitions or fluctuations in season lengths.

$$S_t = \gamma \mathcal{DT}_t + (1 - \gamma) S_{t-m}, \tag{5.3}$$

where S_t denotes the seasonal component at time t, γ is the smoothing factor, and $\mathcal{DT}_t$ is the detrended data at time t. Other filters can be substituted for our trend and seasonal filters to better suit different analytical needs, but our designed ASTD that maintain a computational cost not exceeding $\mathrm{O}(N)$ for each timestamp.

5.3.4 OnlineSLE

The OnlineSLE method is capable of estimating the season length in an online mode both quickly and accurately, as detailed in Chapter 4. As discussed previously, the season length is a critical parameter for achieving high-quality decomposition results. Utilizing this method supports enhanced decomposition techniques. To address the challenges

of online STD in time series that include seasonality transitions and fluctuations, we have integrated our OnlineSLE into our ASTD to automatically estimate the season length. Notably, to circumvent the frequency resolution issues influenced by the sliding window size, OnlineSLE employs the HAQSE algorithm for the `SPLE` function (as explained in Section 4.3.2).

5.4 Datasets

In our experiments, we evaluated the performance of our OnlineSLE and existing SLE methods using both synthetic and real-world datasets. We divided those datasets into three groups: synthetic dataset, real-world datasets with seasonality transitions or fluctuations, and real-world datasets from the Comprehensive R Archive Network (CRAN)[1].

5.4.1 Synthetic datasets

To evaluate the performance of our proposed STD methods, we generated two synthetic datasets, detailed below and shown in Figures 5.5 and 5.6.

- **Syn1:** This dataset is designed to simulate seasonality transitions and trend changes. It comprises three distinct seasonality phases: the season length of first and third phases are 50 timestamps, and the season length of second phase is 80 timestamps. These phases transition at points 1800 and 3600, respectively. Additionally, the trend component is divided into three distinct phases: an initial phase of increasing trend, a subsequent phase of stable trend, and a final phase of decreasing trend.

[1] https://cran.r-project.org/

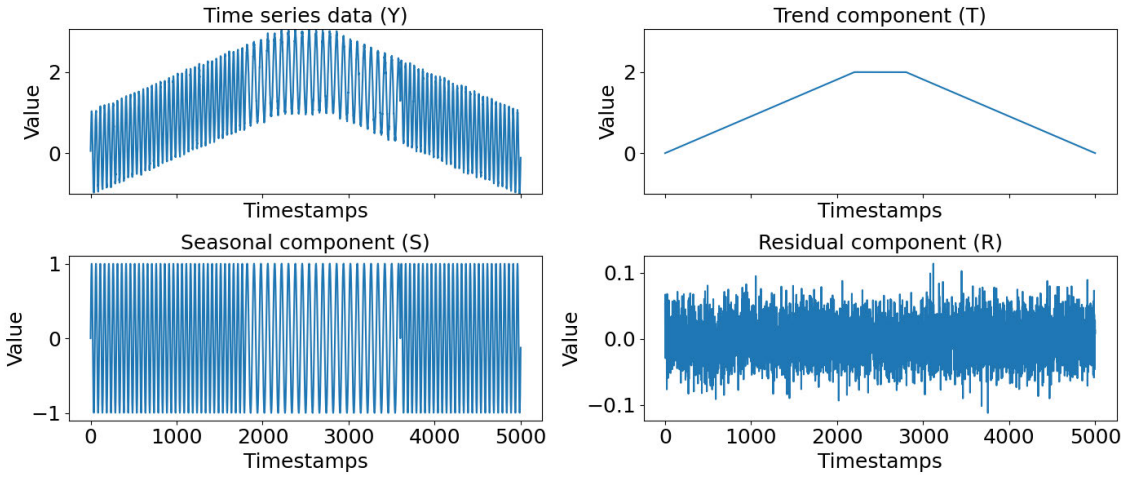


Figure 5.5: Syn1 and its components.

- **Syn2:** This dataset includes a single seasonal component with a primary season length of 80. To test the robustness of decomposition methods against irregular fluctuations, it incorporates 10 randomly selected seasonality fluctuations. Additionally, the trend component has four abrupt trend change.

5.4.2 Real-world Datasets with Seasonality Tansitions or Fluctuations

We selected six real-world datasets, those referred to as **Real1**. To evaluate the effectiveness of various decomposition methods in handling seasonality transitions or fluctuations. Figure 5.7 exhibits full length of those datasets and below is a detailed description of each dataset:

- **WalkJogRun1** [32]: Records human movement activities (walking, jogging, running) using a rotation sensor GyrY on the left lower arm. It features seasonality transitions corresponding to changes in activity, with red and green lines indicating transitions

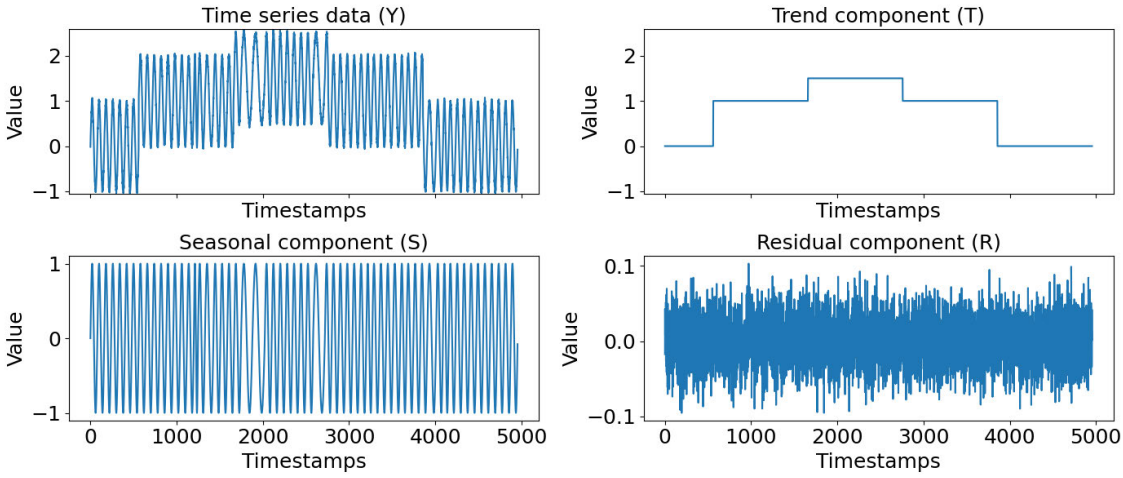


Figure 5.6: Syn2 and its components.

from walking to jogging and jogging to running, respectively.

- **WalkJogRun2** [32]: Similar to WalkJogRun1 but with the sensor positioned on the left calf. The activity transitions are the same timestamps of WalkJogRun1. A notable aspect of this dataset is the inclusion of an anomaly pattern related to stumbling, observed around the 1400 to 1600 timestamps.

- **Co2** [57]: Captures the global monthly average atmospheric CO2 levels from March 1958 to November 2023. This dataset is a classical dataset from STL publication [19].

- **Canadian lynx** [14, 44]: Counts the annual number of lynx trapped in Northwest Canada's McKenzie River district from 1821 to 1934. The population dynamics of lynx are influenced by food availability.

- **SOI** [76]: Observes the monthly sea level pressure differences between the western and eastern tropical Pacific from January 1951 to October 2023. Known as the Southern Oscillation Index (SOI), this standardized index reflects the El Niño and La Niña

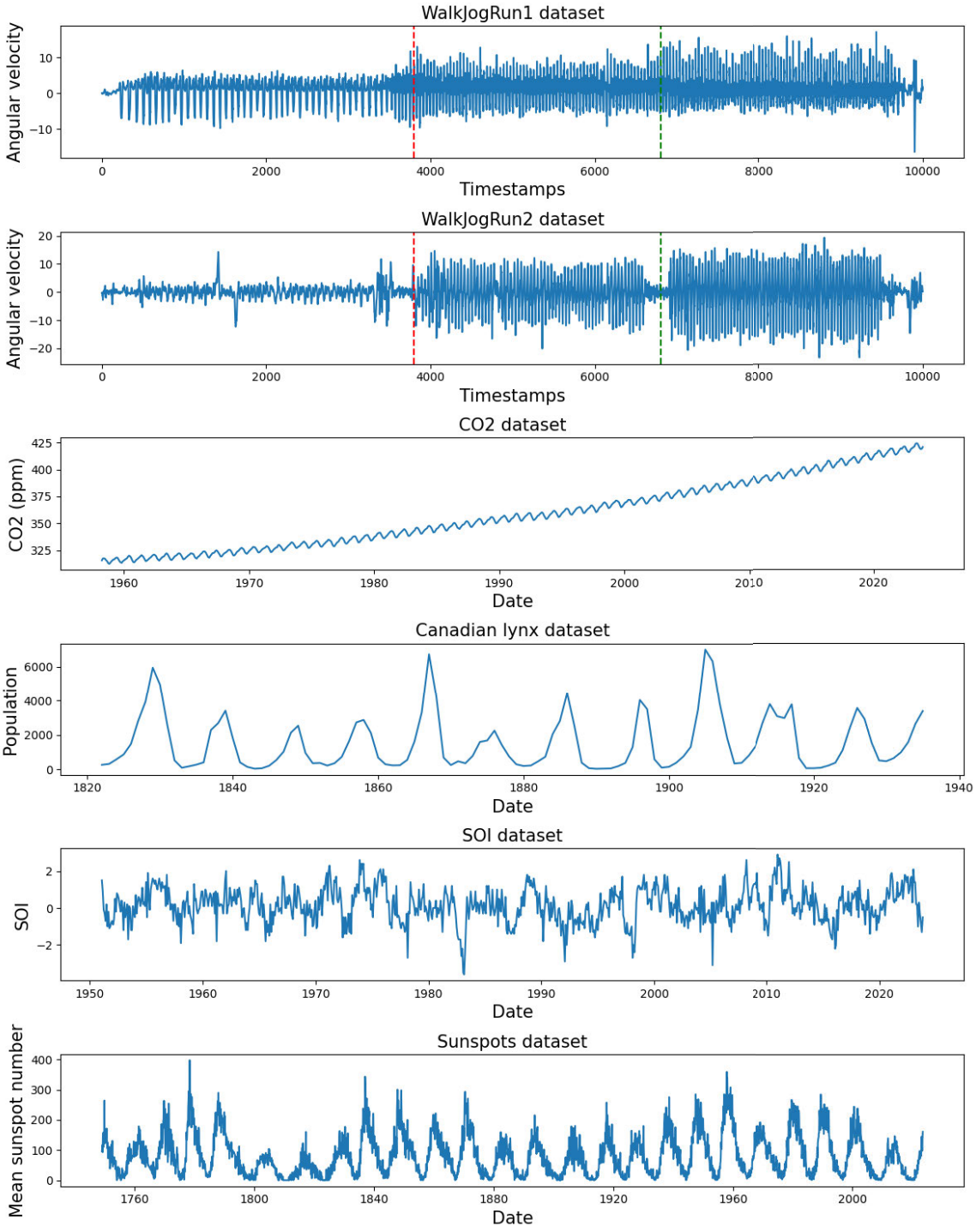


Figure 5.7: All datasets of Real1, arranged from top to bottom as follows: WalkJogRun1, WalkJogRun2, CO2, Canadian Lynx, SOI, and Sunspots.

phenomena. Season length for each cycle typically aligns with a five-year cycle but may show variability due to external factors like volcanic eruptions [24].

- **Sunspots** [112]: Records the average monthly number of sunspots from January 1749 to October 2023. The typical cycle of 11 years makes this dataset pivotal for understanding solar activity behavior.

5.4.3 Real-world datasets from Comprehensive R Archive Network (CRAN)

To demonstrate the comprehensiveness of our ASTD method relative to existing STD methods, we expanded our evaluation to include a broader range of real-world datasets. These datasets are open-source time series data available in packages from the CRAN. For this analysis, we specifically selected datasets from various domains such as the economy, meteorology, and retail sales. These datasets were sourced from four notable CRAN packages: *astsa*, *expsmooth*, *fma*, and *fpp2*, collectively referred to hereafter as **Real2**.

In the preprocessing for Real2, we manually check all datasets from four packages to ensure the quality and relevance of the datasets for our analysis. Initially, we excluded any datasets that lacked a defined ground truth for the season length, as this is crucial for validating our decomposition methods. We further eliminated datasets where the ground truth season length did not align with identifiable cycles, which are essential for seasonal-trend analysis. Moreover, we focused on datasets that included data spanning more than five complete cycles, providing a robust basis for assessing trend and seasonal components. After applying these criteria, a total of 55 datasets were retained for inclusion in our analysis.

5.5 Experimental Setting

Our implementation utilized Python 3.9.2 and R 4.3.0. The methods assessed for offline STD included STL [19], STR [25], and FastRobustSTL [129]. Online STD methods were also evaluated, including as OnlineSTL [73] and OneShotSTL [35]. Furthermore, we developed our own online STD methods, STL adapted to a sliding window approach (hereafter referred to as SlidingSTL), and our proposed ASTD. In this evaluation, we divided into two phase: decomposition quality on synthetic and real-world datasets by various STD methods.

5.5.1 Metrics for decomposition quality with synthetic datasets

The evaluation of synthetic datasets was designed to demonstrate that our proposed method achieves a decomposition quality that closely approximates the ground truth. We measured this quality by calculating the Mean Square Error (MSE) between the ground truth and the results provided by various STD methods. Lower MSE values indicate a closer approximation to the ground truth, suggesting higher accuracy, while higher MSE values signify a greater deviation.

5.5.2 Metrics for Decomposition Quality with Real-World Datasets

For the real-world datasets, the ground truth for each component was unknown. Therefore, we utilized alternative metrics to evaluate decomposition quality: the smoothness of the trend component, the presence of seasonality in the seasonal component, and the randomness of the residual component.

Trend Smoothness

We measured the smoothness of the trend component via the standard deviation of its first-order difference [73]. Given the trend component denoted by $T = (T_1, T_2, \ldots, T_t)$, the first-order difference of the trend component (ΔT_i), is calculated as:

$$\Delta T_i = T_{i+1} - T_i \quad \text{for} \quad i = 1, 2, \ldots, t-1, \tag{5.4}$$

where t denotes the latest timestamp, indicating the total length of the trend component data. The trend smoothness ($\sigma_{\Delta T}$) is then the standard deviation of these first-order differences:

$$\sigma_{\Delta T} = \sqrt{\frac{1}{t-1}\sum_{i=1}^{t-1}(\Delta T_i - \mu_{\Delta T})^2} \tag{5.5}$$

where $\mu_{\Delta T}$ is the mean of the first-order differences of trend component. Lower values of $\sigma_{\Delta T}$ indicate smoother trends, suggesting higher decomposition quality.

Presence of Seasonality

We measured the presence of seasonality by applying the Kruskal–Wallis test to the seasonal component [7]. Given the seasonal component denoted by $S = (S_1, S_2, \ldots, S_t)$, the Kruskal–Wallis test statistic is calculated as:

$$W = \frac{12}{N(N+1)}\sum_{j=1}^{g}\frac{U_j^2}{n_j} - 3(N+1) \tag{5.6}$$

where N denotes the length of S, g denotes the number of groups, n_j denotes the number of observations in the j-th group, and U_j denotes the sum of ranks in the j-th

group. To determine the number of groups (g), it is set equal to the season length m, which reflects the position within the cycle [7]. For example, if we have monthly data spanning one year (with a season length of 12), we group the data into 12 groups, with each group corresponding to one month within the cycle. Therefore, we group the observations by month, starting with January as the first group, February as the second, and so on until December, which is the twelfth group. This aligns with the season length m.

To illustrate the calculation of the sum of ranks (U_j) within each group for the Kruskal-Wallis test, consider an example with three groups, resulting in each group having its unique set of data points:

- Group 1 ($j = 1$): 5, 3, 8
- Group 2 ($j = 2$): 7, 6, 2
- Group 3 ($j = 3$): 4, 9, 1

Ranks are assigned to the original observations within each group, and U_j, the sum of ranks in the j-th group, is calculated:

- U_1 for Group 1: $5 + 3 + 8 = 16$
- U_2 for Group 2: $7 + 6 + 2 = 15$
- U_3 for Group 3: $4 + 9 + 1 = 14$

Thus, U_j denotes the sum of ranks within each group. The values are $U_1 = 16$, $U_2 = 15$, and $U_3 = 14$. After calculating the Kruskal-Wallis test statistic W, it is compared against a chi-square distribution with $g - 1$ degrees of freedom. The resulting p-value is

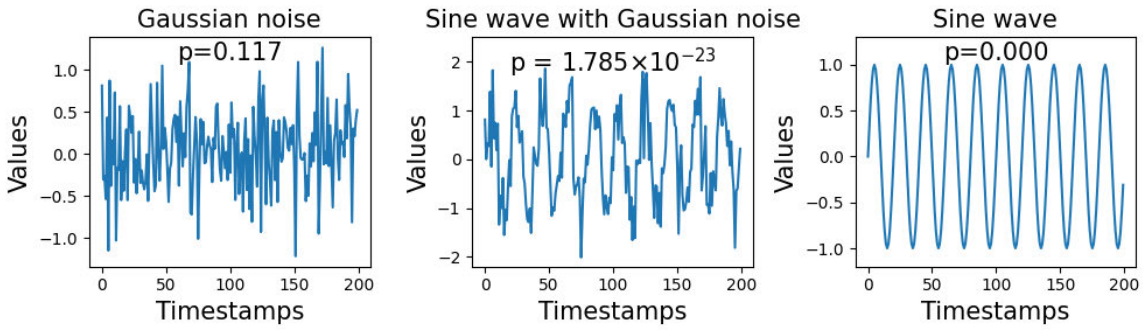


Figure 5.8: Comparison of three time series based on their Kruskal-Wallis p-values, from highest (left) to lowest (right), indicating increasing seasonality consistency.

used to determine the statistical significance of the observed test statistic. Lower values suggest stable repeating cycles, indicating consistent seasonality, while higher values may suggest inconsistencies in the seasonal component. Figure 5.8 illustrates this by showing three time series with the highest and lowest p-values. From left to right, the p-values are 0.117, 1.785×10^{-23}, and 0.

Randomness

We measured randomness in the residual component by applying the Ljung-Box test to the residual component [7, 45]. Given the residual component denoted by $R = (R_1, R_2, \ldots, R_t)$, the Ljung-Box test statistic is calculated as:

$$Q = N(N+2)\sum_{k=1}^{h}\frac{\hat{\rho}_k^2}{N-k} \tag{5.7}$$

where N denotes the length of R, h is the number of lags being tested, and $\hat{\rho}_k$ is the autocorrelation at lag k. $\hat{\rho}_k$ is calculated as:

$$\hat{\rho}_k = \frac{\sum_{i=1}^{N-k}(R_i - \mu_R)(R_{i+k} - \mu_R)}{\sum_{i=0}^{N}(R_i - \mu_R)^2} \tag{5.8}$$

where μ_R denotes the the mean of the residual component (R). To determine h, it is set to $\min(2m, N/5)$, where m is the season length [45]. Lower values suggest that the residual component originates from independent and identically distributed (iid) data, indicating the successful extraction of the seasonal component. To illustrate this point, we utilized the Ljung-Box test to measure the randomness in three time series depicted in Figure 5.8. From left to right, the randomness values are 50, 2042, and 3616, respectively.

5.6 Experimental Results

5.6.1 Experimental Results with Synthetic Datasets

In this subsection, we organize the discussion based on a structured approach to evaluate the performance of various decomposition methods. We will first discuss the results for each component (trend, seasonal, and residual) across different methods. Following this analysis, we will present how the results vary with changes in season length for each component, providing insights into the adaptability and efficiency of each method under different seasonal conditions.

Comparison by visualization

We set the season length at 80 for all methods. This corresponds to the length of the second phase of seasonality in Syn1 and the main seasonality in Syn2. Figures 5.9 and

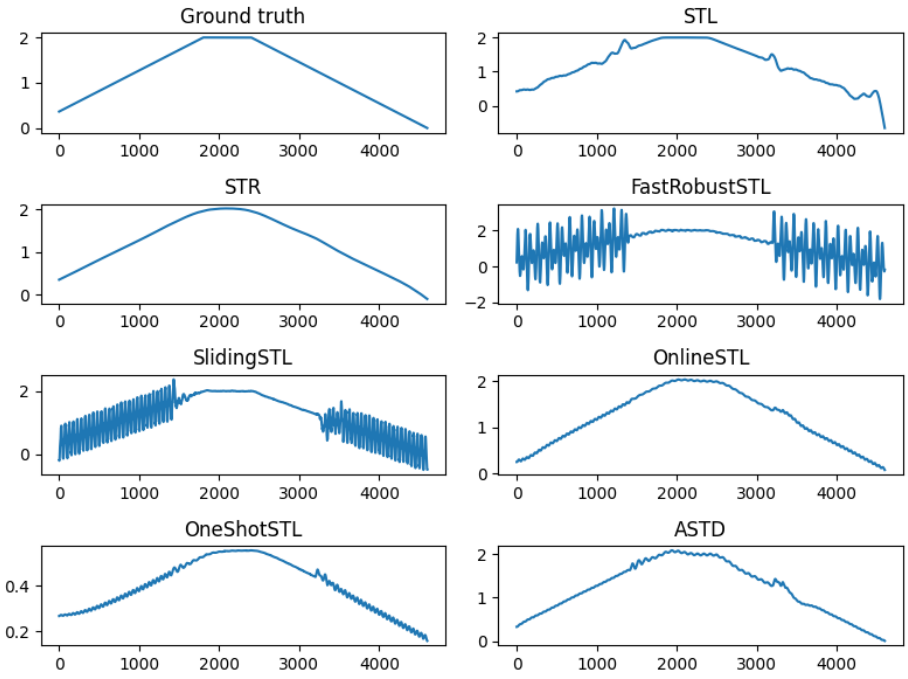


Figure 5.9: Comparison of Syn1's trend component by various STD methods.

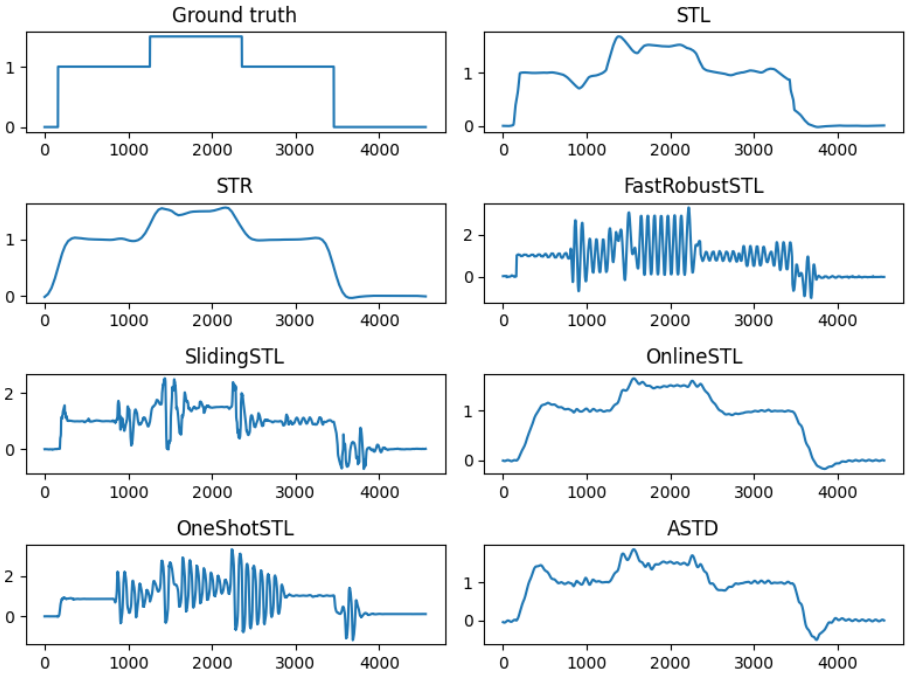


Figure 5.10: Comparison of Syn2's trend component by various STD methods.

5.10 show the trend component from decomposition results by various methods.

In the case of offline STD methods, STR achieved robust results in the trend component for both datasets. In contrast, while STL and SlidingSTL are both based on the STL method, the results of STL were closer to the ground truth because this method utilizes the entire time series for decomposition. Utilizing the entire time series allows STL to better adjust its smoothing parameters to fit the broader context of the data, which is not possible with SlidingSTL. SlidingSTL failed to decompose the trend, particularly where seasonality occurs in the first and third phases with a season length of 50. The suitable season length impacted to decomposition results.

For online STD methods, including OnlineSTL, OneShotSTL, and ASTD, the results were robust and close to the ground truth. However, OneShotSTL only approximated the shape for Syn1, with trend values ranging from 0.1 to 0.6, which deviated from the ground truth values (0 to 2). Moreover, the trend component of OneShotSTL contained the portion seasonal component.

Turning to the highlights of this evaluation, the decomposition results for the seasonal component are shown in Figures 5.11 and 5.12. For Syn1, both FastRobustSTL and ASTD provided robust results for the seasonal component. However, ASTD also provided robust results for the trend component and operates in online mode. Additionally, ASTD achieved these results without needing a predefined season length. While STR did provide robust results for the trend component, it failed to do so for the seasonal component. Consequently, ASTD outperforms STR in this regard.

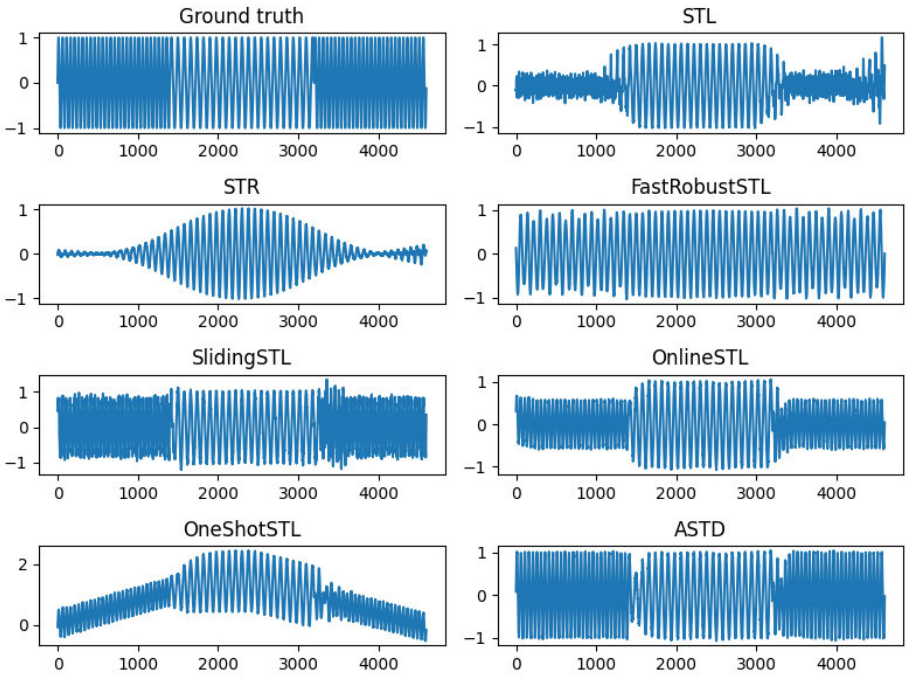


Figure 5.11: Comparison of Syn1's seasonal component by various STD methods.

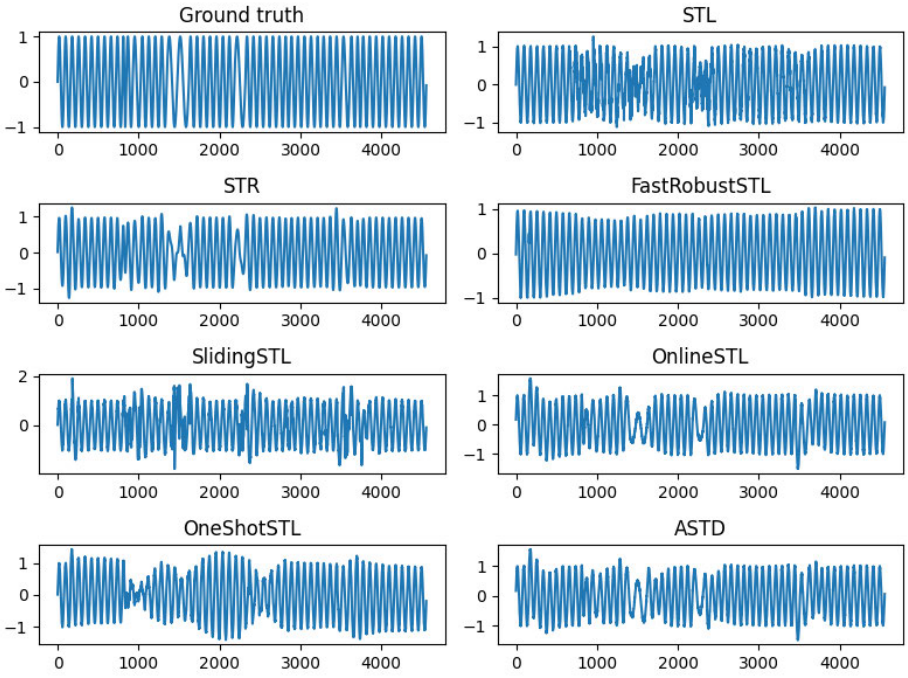


Figure 5.12: Comparison of Syn2's seasonal component by various STD methods.

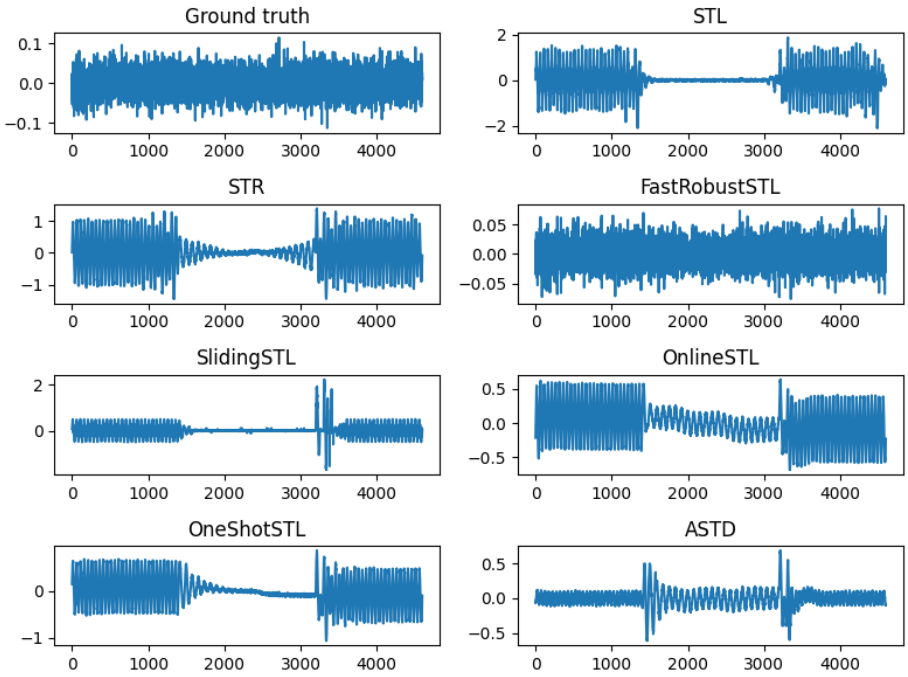


Figure 5.13: Comparison of Syn1's residual component by various STD methods

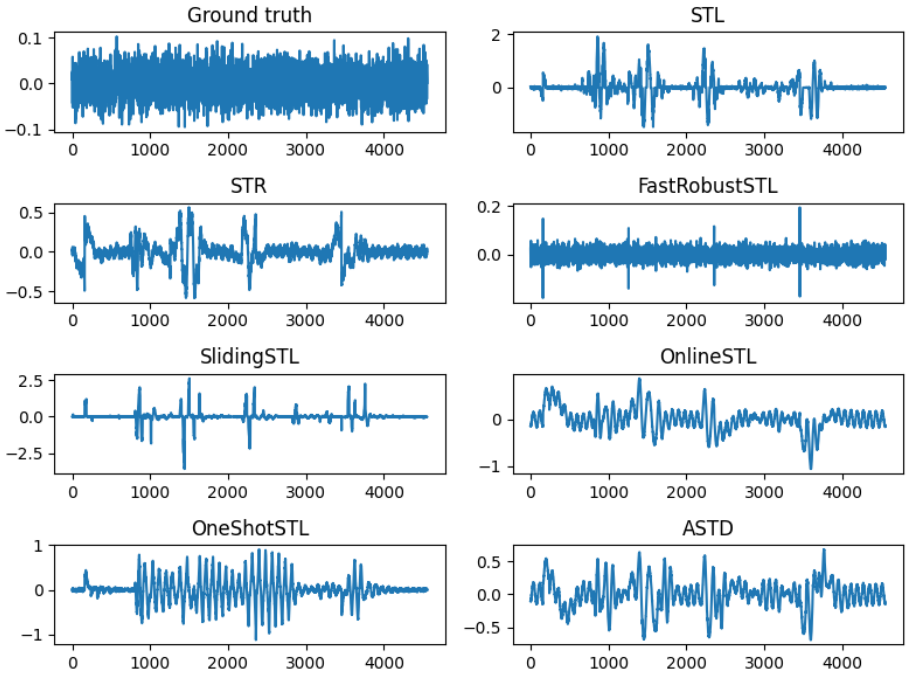


Figure 5.14: Comparison of Syn2's residual component by various STD methods

Finally, Figures 5.13 and 5.14 show the residual components from the decomposition results by various methods. FastRobustSTL achieved results that were closest to the ground truth, primarily because it is the only STD method in this evaluation that incorporates noise filtering before decomposition. This preprocessing step is why FastRobustSTL yields high-quality results for the residual component. However, FastRobustSTL's performance is influenced by the season length, as observed in the trend component decomposition results, and it operates in an offline mode.

Comparison by Various Season Lengths

To illustrate the influence of season length on decomposition quality, we plotted the MSE for each component and dataset by various STD methods. Note that an MSE value close to zero indicates a result that closely matches the ground truth. The results are shown as Figure 5.15.

For offline STD methods, both STL and STR achieved lower MSE values for the trend component by utilizing the entire time series for decomposition. However, these MSE values for the seasonal component were influenced by the season length due to mismatches with the actual season length. FastRobustSTL provided the lowest MSE for the residual component, as previously explained. Optimal season lengths, such as 80 and 100, can reduce MSE in both the trend and seasonal components, though the MSE values remain higher than those for STL and STR.

Notably, the MSE of FastRobustSTL could be improved with prior knowledge of the time series characteristics, specifically whether they exhibit gradual or abrupt changes.

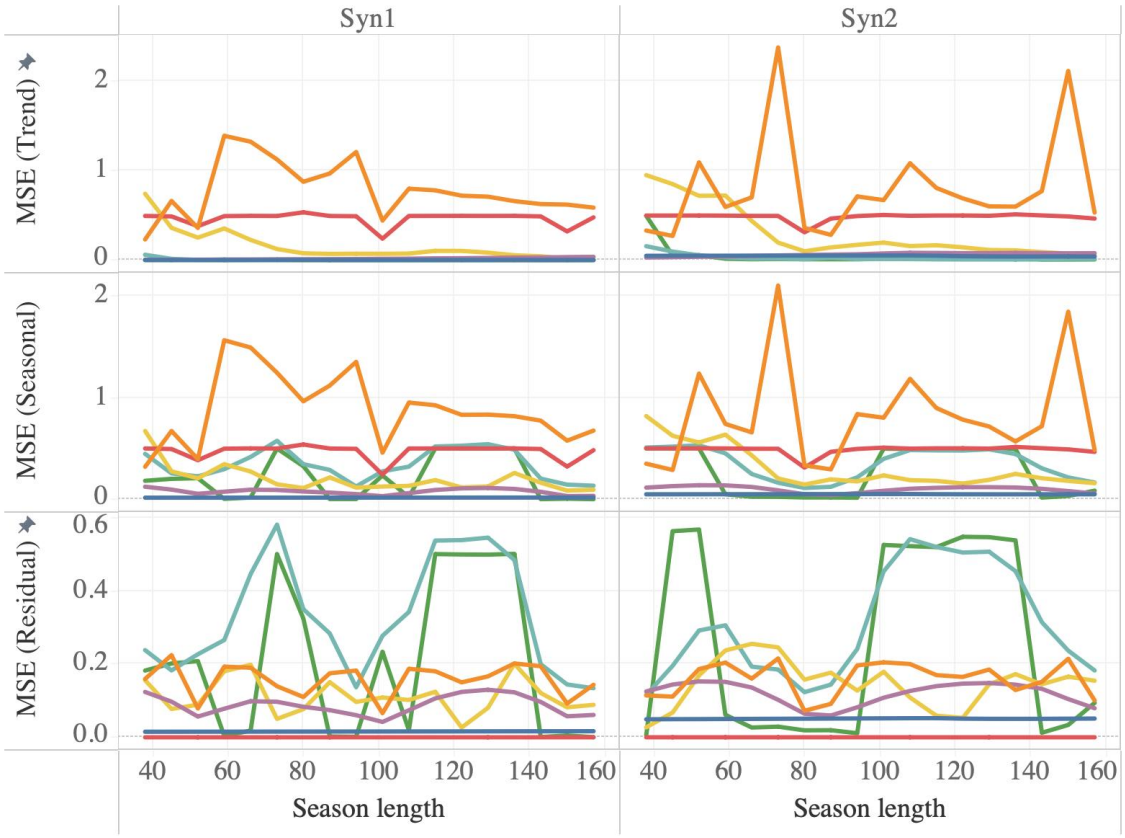


Figure 5.15: Comparison of MSE by various season lengths and methods. Color legends: Our proposed method, ASTD (■), FastRobustSTL (■), OneShotSTL (■), OnlineSTL (■), STR (■), STL (■), and SlidingSTL (■).

This knowledge enables optimal adjustment of the ℓ_1 filtering parameters, as explained in Section 5.2.2.

Turning to online STD methods, the MSE values for the trend component in SlidingSTL are higher than those in STL, similar to previous evaluations with optimal season lengths. OneShotSTL has similar limitations to FastRobustSTL, and its performance trends are similar to FastRobustSTL. When the optimal season length is provided, the MSE

values for OneShotSTL decrease.

ASTD and OnlineSTL utilized the same filters for decomposing the trend and seasonal components, which resulted in similar MSE values for these components. By leveraging OnlineSLE to adjust the season length, ASTD achieved a lower MSE value for the residual component. This indicates that ASTD effectively decomposes the seasonal component at an optimal length through automatic adjustments provided by OnlineSLE. The residual component did not contain the portion of seasonal component. This outcomes highlight the significant impact of season length on decomposition quality.

In summary, ASTD's automatic adjustment of season length during decomposition leads to lower MSE values and high-quality results. To illustrate, we present the decomposition results for Syn1 using various methods with a season length of 129 in Figures 5.16 - 5.18. Note that 129 does not match the ground truth.

From these figures, we observe how the season length affects the results of the existing STD methods. However, if the optimal season length is known in advance, it can be utilized with other STD methods.

5.6.2 Experimental Results with Real-world Datasets

We compared ASTD with different STD methods. Note that the ground truth for season lengths varies across datasets. We adjust the season length input parameters for the existing TSD methods based on an error margin from the ground truth. For example, if the ground truth of season length is 100 and the error margin is 0.9 times, the input season

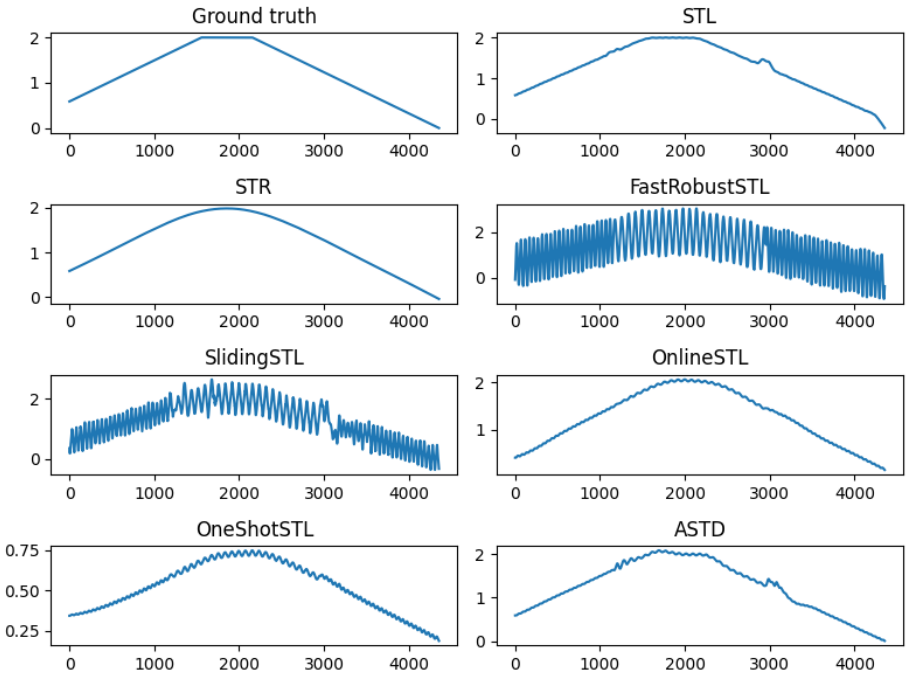


Figure 5.16: Comparison of Syn1's trend component by various STD methods ($m = 129$).

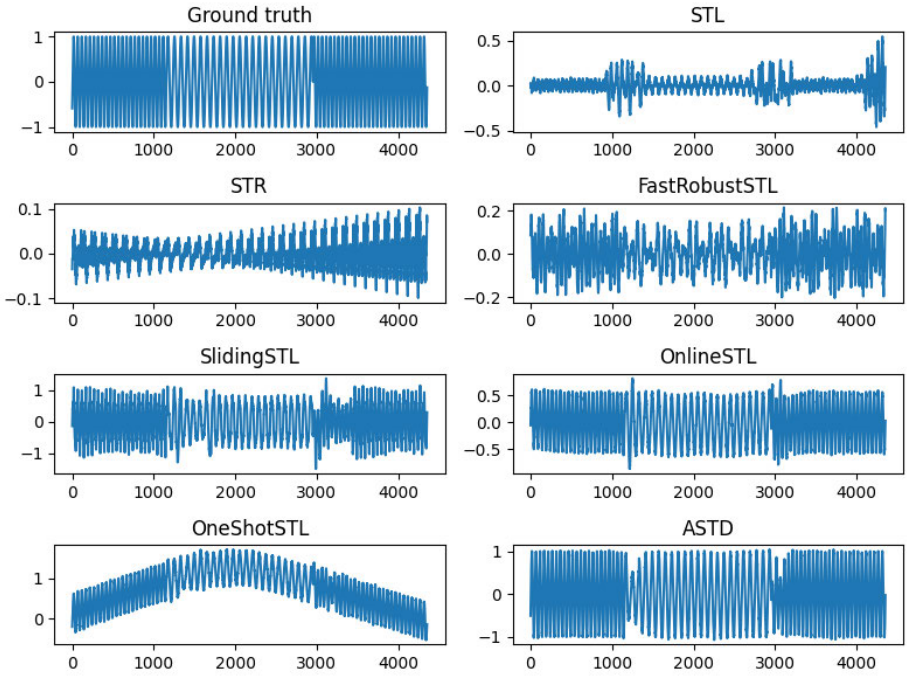


Figure 5.17: Comparison of Syn1's seasonal component by various STD methods ($m = 129$).

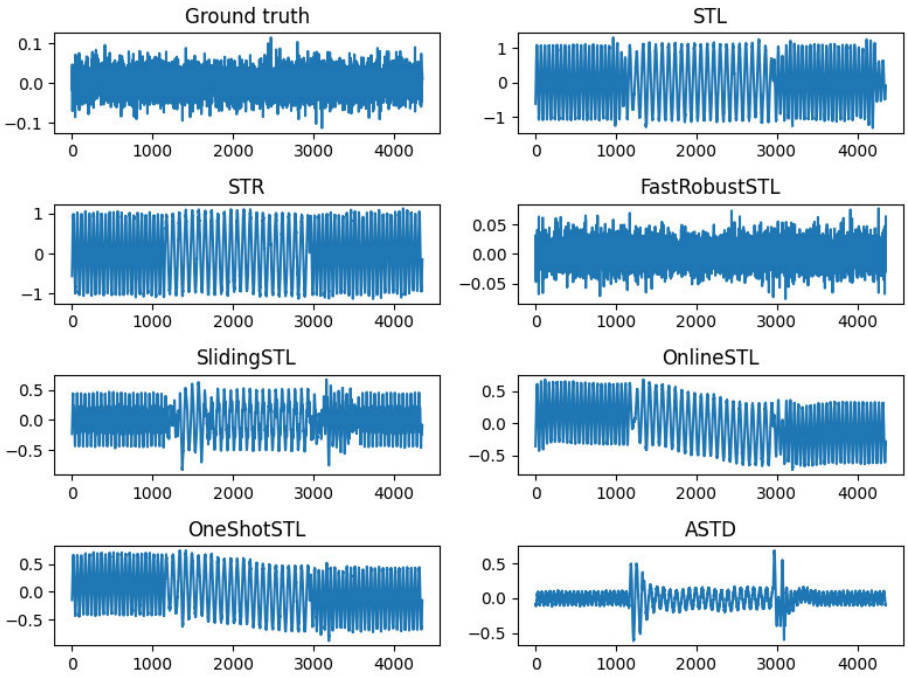


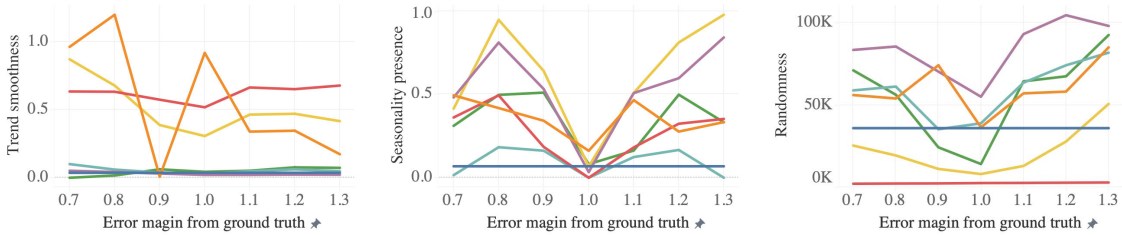
Figure 5.18: Comparison of Syn1's residual component by various STD methods ($m = 129$).

length for existing STD methods would be 90. This setting facilitates comparison across different datasets.

Comparison of Decomposition Quality on Real1

We evaluated various STD methods on the Real1 dataset, with results presented in Figure 5.19 in terms of trend smoothness, presence of seasonality, and randomness.

Overall, these results are consistent with those obtained from the synthetic datasets, underscoring the importance of the optimal season length for achieving high quality decomposition. Notably, the evaluation of presence of seasonality demonstrated the capture of seasonal components when the optimal season length was provided. Although the presence of seasonality in ASTD was slightly higher than in some other STD methods, ASTD offers



(a) Trend smoothness (b) presence of seasonality (c) Randomness

Figure 5.19: Comparison of decomposition quality on **Real1** by various season length and methods. Color legends: Our proposed method, ASTD (■), FastRobustSTL (■), OneShotSTL (■), OnlineSTL (■), STR (■), STL (■), and SlidingSTL (■).

Table 5.2: Evaluation of Decomposition Quality on **Real2** across different season lengths. The bolded and underlined results indicate the lowest values among online TSD methods.

Methods		0.8 times ground truth				Optimal season length				1.2 times ground truth			
		TS.	PS.	Ran.	Avg. time	TS.	PS.	Ran.	Avg. time	TS	PS.	Ran.	Avg. time
Offline	STL	0.039	0.077	1138	0.019	0.027	0.001	392	0.023	0.028	0.061	1470	0.028
	STR	0.026	0.728	705	0.946	0.033	0.005	366	1.087	0.028	0.685	1101	1.705
	FastRobustSTL	0.045	0.830	446	2.422	0.041	0.031	582	2.621	0.051	0.711	611	2.473
Online	SlidingSTL	0.192	0.757	**558**	1.898	0.084	**0.042**	**112**	2.607	0.142	0.702	**829**	3.468
	OnlineSTL	**0.024**	0.755	2053	0.016	**0.016**	0.043	3171	0.017	**0.015**	0.771	2806	0.018
	OneShotSTL	0.100	0.735	1419	**0.009**	0.183	0.189	1271	**0.008**	0.179	0.721	1538	**0.008**
	ASTD	0.029	**0.241**	1173	0.048	0.029	0.241	1173	0.048	0.029	**0.241**	1173	0.048

a significant advantage by eliminating the need for a predefined season length as an input. This approach provides greater flexibility compared to traditional STD methods, which rely on users to specify the season length, thus affecting the decomposition results.

Comparison of Decomposition Quality on Real2

The evaluation results of various STD methods on the Real2 dataset are presented in Table 5.2. In this table, 'TS.', 'PS.', 'Ran.', and 'Avg. time' represent trend smoothness,

presence of seasonality, randomness, and average computation time per dataset (in seconds), respectively. Notably, the datasets in Real2 are typically short, averaging 376 instances, and lack seasonality transitions or fluctuations. This characteristic makes the Real2 datasets distinct from others.

Offline STD methods typically achieve lower values for trend smoothness, presence of seasonality, and randomness, as they utilize optimal season lengths for decomposition. STR and FastRobustSTL, in particular, incur higher computational times due to their optimization processes, which aim to find the best fit for the trend and seasonal components with the original time series. Conversely, STL achieved the lowest average computation time in offline mode. However, when STL is adapted to online mode (as SlidingSTL), the computation cost increases significantly because of its iterative decomposition at each timestamp. Consequently, these offline STD methods, when applied using a sliding window strategy, are not suitable for online STD.

In the evaluation of online STD methods, ASTD stood out by achieving the lowest scores for the presence of seasonality when the optimal season length was not specified for the other methods. Generally, it ranked second in the remaining metrics. Conversely, OnlineSTL recorded the best trend smoothness results but was not consistently better across other metrics. Moreover, it was faster than ASTD, as the latter utilized OnlineSLE to estimate the season length for each timestamp. OneShotSTL was notable for its minimal computational computation costing O(1) per timestamp, but its performance depended on specific parameters impacting decomposition quality. SlidingSTL performed best in presence of seasonality and randomness when the optimal season length was provided. As

explained in the previous paragraph, the computation cost of SlidingSTL is higher than that of other online STD methods, thereby making it unsuitable for streaming environments.

This evaluation highlights the trade-offs associated with each STD method. Firstly, the choice of operational mode depends on whether the entire input time series can be accessed. Secondly, the applicability of each method hinges on prior knowledge of the season length or specific dataset characteristics. If this information is known, the other STD methods may be choice that utilized for decomposition. Conversely, if such details are unknown, our ASTD offers an effective alternative solution, capable of handling seasonality transitions or fluctuations. Therefore, ASTD offers flexibility in decomposing time series without the predefined season length as an input, which this length impacts decomposition results in other STD methods

5.7 Discussion and Conclusions

ASTD is a novel method that incorporates OnlineSLE to dynamically estimate the season length. This capability allows ASTD to offer the flexibility needed to decompose time series in a streaming environment, where data continuously changes over time due to seasonality transitions or fluctuations. The method is meticulously designed to maintain a computational cost of $O(N)$. Our evaluations demonstrate that ASTD can adaptively decompose comprehensive time series across various datasets effectively. Unlike other STD methods, which require predefined specific dataset characteristics for effective decomposition, ASTD provides a robust alternative when such information is unavailable. For future work, we plan to pursue two main directions: 1) Explore the integration of ℓ_1 or ℓ_2 trend

filtering [54, 97] within our ASTD framework. 2) Work on reducing the computational cost of OnlineSLE and all filtering processes in ASTD to O(1), aiming for efficiency comparable to that of OneShotSTL.

Chapter 6

Discussions and Future Challenges

In this chapter, we discuss the practical applications and future challenges of our three proposed methods. Our aim is to highlight the potential of these methods for the research community and encourage further exploration and utilization. We summarize the practical applications and remaining challenges that have the potential to enhance or complement our proposed methods for each method—EBinning, OnlineSLE, and ASTD—in Sections 6.1 to 6.3, respectively. Following this, Section 6.4 concludes with a discussion of three methods.

6.1 Elastic Data Binning (EBinning)

6.1.1 Contributions and Problem-Solving Capabilities of EBinning

We propose EBinning, a novel method to capture transient patterns of the dynamic behavior in the trend component. Our specified domain for EBinning in this thesis is capturing transient patterns in the time-domain of astrophysics. EBinning consists of two

methods: Mean-EBinning, based on Hoeffding's inequality, and Linear-EBinning, based on Student's t-test. This method achieves high-quality capturing results without the influence of input parameters, compared to Data binning (PAA) [50], SAX [62], MP [134], and others. We conducted evaluations of EBinning using both synthetic and real-world datasets. Below are the contributions of EBinning and the problems it can solve:

High quality for transient pattern capturing in time series with heavy noise: As demonstrated in Section 3.4, our LCs dataset contains heavy noise, making it difficult to identify small transient patterns through visual inspection. Figure 6.1 illustrates the results of EBinning (red line) capturing the small Kepler pattern. By visually inspecting this result, we rapidly identified the sudden changes at timestamp 200 that correspond to the ground truth of the Kepler flare. Additionally, EBinning provides high-quality transient pattern capturing in terms of IOU, sketching quality, and accuracy for detecting transient patterns, as shown in Section 3.6.

Elimination of specific input parameter requirements: A key contribution of EBinning is its ability to operate without requiring users to specify input parameters, which can often introduce error or bias. As shown in Section 3.6, our comprehensive evaluation with existing methods shows that results often depend on specific input parameters. EBinning offers a flexible method that operates without any prior knowledge, dynamically adjusting subsequence sizes based on specific features for capturing transient patterns. However, if users know the essential features of the time series or transient patterns in advance, existing methods may be preferable for high-quality transient pattern capturing. This is because users can provide optimal parameters for capturing transient patterns.

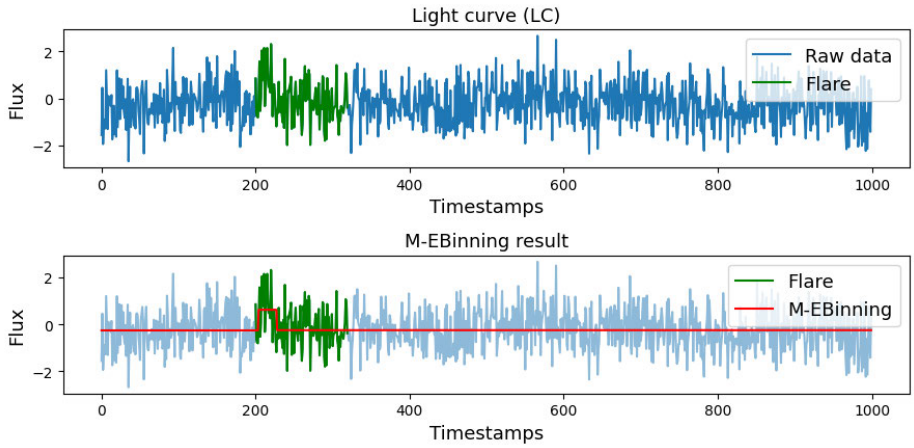


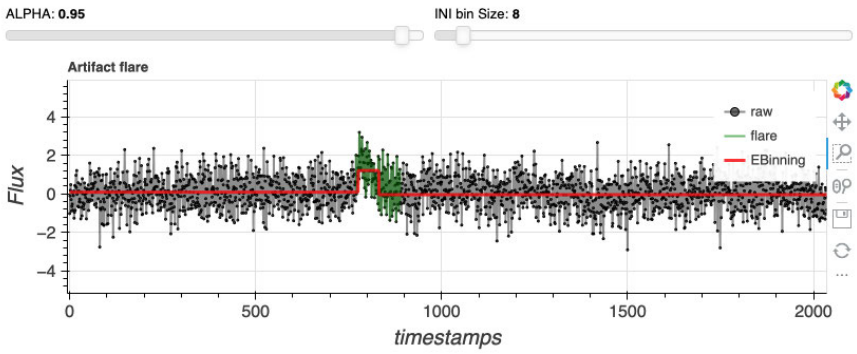
Figure 6.1: Comparison between the original LC with the artifact Kepler flare and the results from M-EBinning. The green highlight indicates the Kepler flare, with the peak of this flare occurring at timestamp 200.

To further enhance user experience, we created a new prototype version of EBinning using JavaScript for interactive use. This version allows users to adjust the initial bin size in a user-friendly manner using a sliding bar, making it easier to analyze transient patterns in LCs. The EBinning prototype using JavaScript is shown in Figure 6.2. Note that while the initial bin size is an important input parameter for EBinning, it does not impact the sketching results (see Figure 6.2).

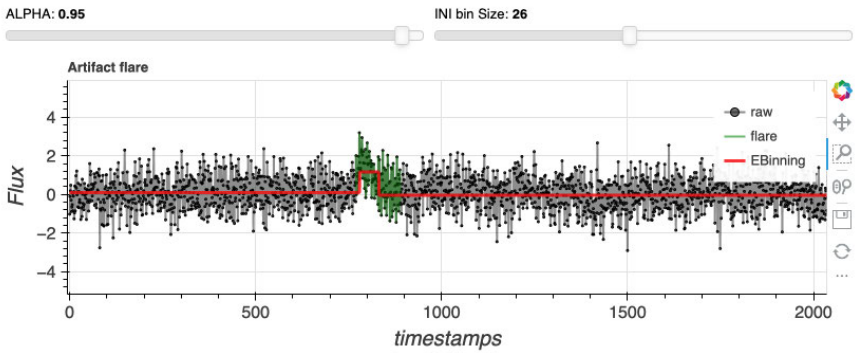
6.1.2 Remaining Challenges and Limitations of EBinning

This subsection highlights the remaining challenges and limitations associated with EBinning as observed in our research.

Extension for anomaly detection: Although EBinning is effective for capturing transient patterns, extending its capabilities to anomaly detection could further enhance



(a) Initial bin size = 8 (default setting based on evaluation in Section 3.6)



(b) Initial bin size = 26

Figure 6.2: Comparison of EBinning results with different initial bin sizes.

its utility. The mergeability score in EBinning measures the potential for merging between two neighboring bins. In this thesis, we used a top-k approach based on the mergeability score within the score profile. A high mergeability score indicates that the bin contains a transient pattern. However, there are cases where the highest mergeability score may be close to zero, suggesting that two neighboring bins do not contain a transient pattern. This presents a challenge for anomaly detection, as the method may not identify anomalies accurately. Addressing this limitation by refining the mergeability score or incorporating additional criteria could improve EBinning's performance in anomaly detection.

Potential for using other metrics: In this thesis, the mergeability score is based on Hoeffding's inequality and Student's t-test. However, there is potential for using other metrics to enhance the accuracy and robustness of EBinning. These metrics should be based on statistical features for each bin, which depend on the specific domain or area of interest. Examples of other metrics for the mergeability score include Chebyshev's inequality, the Ljung-Box test, and ACF. Exploring and utilizing these alternative statistical measures could provide more reliable detection of transient patterns and anomalies. Thereby, it expands the applicability of the EBinning.

Application in other domains: Currently, the focus of EBinning is on astrophysics, but the method has potential applications in various other fields such as finance, healthcare, and environmental science. Future research will explore how EBinning can be adapted and applied to these different domains to capture transient patterns and enhance data analysis.

6.2 Online Season Length Estimation (OnlineSLE)

6.2.1 Contributions and Problem-Solving Capabilities of OnlineSLE

We propose OnlineSLE, a novel framework for SLE by incorporating periodogram and HAQSE. This framework differs from traditional frameworks that integrate periodogram and ACF, such as AutoPeriod [120], SAZED [118], and RobustPeriod [127]. OnlineSLE achieves fast computation and high accuracy in estimating the season length in time series, as demonstrated by our evaluation results on both synthetic and real-world datasets. The contributions of this method, and the problems that OnlineSLE can solve, are as follows:

Faster computation: OnlineSLE achieves faster computation than existing methods, which typically have a time complexity of $O(N \log N)$ for periodogram and ACF computation. By utilizing SDFT with a time complexity of $O(N)$ for the periodogram, OnlineSLE outperforms FFT both theoretically and in practical viewpoints, as shown in Section 4.6.1. Additionally, OnlineSLE utilizes HAQSE with a time complexity of less than $O(3N)$, which is faster than ACF. HAQSE addresses the influence of sliding window size on accuracy without relying on ACF.

High accuracy rate: OnlineSLE achieves a high accuracy rate in determining the season length compared to existing methods in both synthetic and real-world datasets. This success is due to the incorporation of periodogram and HAQSE, which are robust against heavy noise and dynamic behavior in the trend component. Additionally, the influence of sliding window size does not impact the results of OnlineSLE, allowing users to set the sliding window without specific assumptions. This reduces the time spent on defining optimal input parameters

6.2.2 Remaining Challenges and Limitations of OnlineSLE

Here, we provide the remaining challenges and limitations of our OnlineSLE that we found during our research.

SDFT structure improvement: As explained in Section 4.7, this direction addresses the numerical error computation from the approximation of twiddle factors in recursive SDFT. Recently, R. Lyons and C. Howard have proposed enhancements to stabilize SDFT [70]. However, these improvements may increase the computational cost of our OnlineSLE. Therefore, we will investigate potential improvements to our OnlineSLE to

enhance the robustness of our results while maintaining low computational cost using other SDFT structures.

Multiple seasonal components: Our time series model, defined in Definition 1 in Section 2.1, is denoted as $Y = T + S + R$. However, some datasets exhibit multiple seasonal components, which are out of scope of this thesis. The time series that contains multiple seasonal components denotes as $Y = T + \sum_{p=1}^{q} S_p + R$, where S_p denotes the p-th seasonal components, $\sum_{p=1}^{q} S_p$ is the sum of multiple seasonal component. Note that details on multiple seasonal components are mentioned in Section 5.2.1. We demonstrate the multiple seasonal component time series, as shown in Figure 6.3 [5]. The top plot displays seven cycles corresponding to the daily seasonal component, which increases during daylight hours and decreases at night. Conversely, the bottom plot shows cycles corresponding to the weekly seasonal component, with demand decreasing during weekends due to reduced work activity at companies. Multiple seasonality affects OnlineSLE because OnlineSLE is designed to determine the season length of time series with a single seasonal component. OnlineSLE must provide only one season length for each timestamp. Therefore, it cannot provide all season lengths.

Sliding window size is set to small: We found a worst-case scenario that can occur in SLE methods, including OnlineSLE. This scenario arises when the sliding window size is too small to contain two or more cycles of the seasonal component. If the sliding window contains only one cycle, SLE methods may incorrectly assume that the time series has no seasonal component due to the lack of periodic behavior within the window. As mentioned in Chapter 4, the season length may change over time due to unstable behavior in

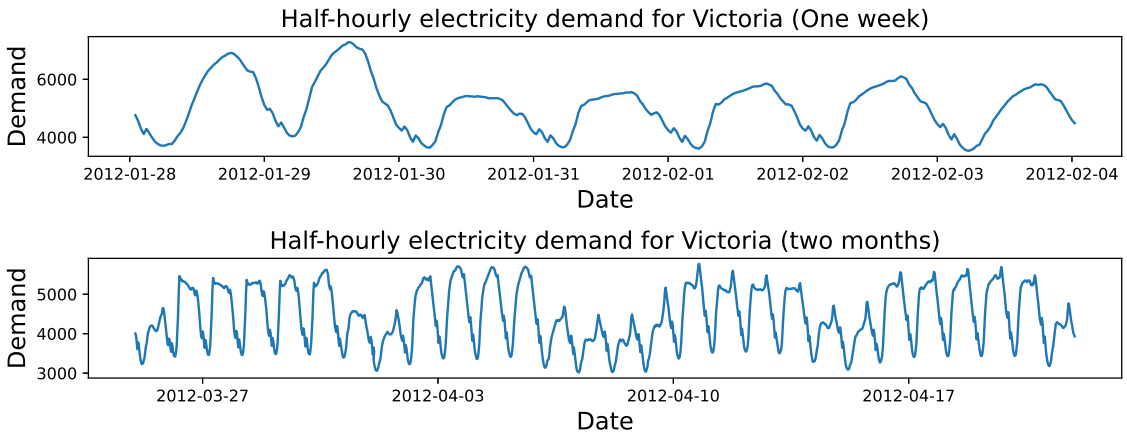


Figure 6.3: Half-hourly electricity demand for Victoria, a multiple seasonal component time series displaying daily and weekly seasonal components [5]. (Top) seven cycles corresponding to the daily seasonal component, (Bottom) cycles corresponding to the weekly seasonal component.

the seasonal component. The current setting of the sliding window length may not capture two or more cycles consistently. Developing a mechanism to avoid this issue remains a challenge for our OnlineSLE.

6.3 Adaptive Seasonal-trend Decomposition (ASTD)

6.3.1 Contributions and Problem-Solving Capabilities of ASTD

We propose the ASTD method, which does not require specifying the season length by incorporating OnlineSLE. ASTD can decompose time series with unstable behavior in the seasonal component and dynamic behavior in the trend component. ASTD eliminates the need for user assumptions about season length, making it more flexible and adaptable

to various time series data. Therefore, ASTD provides robust decomposition even in the presence of significant fluctuations and transitions in seasonality due to unstable behavior in the seasonal component. The contributions of this method, and the problems that OnlineSLE can solve, are as follows:

Eliminating season length assumptions from users: Our ASTD, incorporating OnlineSLE into STD, eliminates the need for user assumptions about season length. As mentioned in Section 5.1, determining the optimal season length is crucial for effective seasonal-trend decomposition. By automatically providing the season length, ASTD ensures accurate and efficient decomposition without relying on user-defined parameters. Additionally, the automatically determined season length aligns with the periodicity of the seasonal component, ensuring that the decomposition accurately reflects the cycles in the seasonal component.

High quality decomposition with fluctuations and transitions in seasonality: Our ASTD provides high-quality decomposition for time series with fluctuations and transitions in seasonality. We have demonstrated that ASTD outperforms existing methods across various quality decomposition metrics, including MSE, trend smoothness, seasonality presence, and randomness. This is achieved by using the optimal season length provided by OnlineSLE as input for the seasonal-trend decomposition. Utilizing the optimal season length from OnlineSLE ensures that the decomposition results reflect the behavior of the seasonal component. Note that if the optimal season length is known in advance, users can utilize existing methods that may provide higher quality decomposition than ASTD.

6.3.2 Remaining Challenges and Limitations of ASTD

Overall, the limitations of ASTD are similar to those of OnlineSLE since OnlineSLE is incorporated into ASTD. The remaining challenges related to ASTD are as follows:

Time computation cost reduction: Overall, our ASTD is designed with a computation cost not exceeding O(N). However, the fastest method for online STD is OneShotSTL, with a computational cost of O(1) [35]. Additionally, OnlineSTL has a computational cost of O(N) but does not utilize any SLE methods [73]. This means that OnlineSTL is faster than ASTD because OnlineSLE within ASTD requires a computational cost of O(N). The computational cost comparison results are shown in Table 5.2, where we are slower than both OnlineSTL and OneShotSTL in theory and practice. Therefore, reducing the time computation cost remains a challenge that needs to be addressed.

Multiple seasonal components decomposition: This limitation is similar to the OnlineSLE method, as we aim to decompose a single seasonal component. However, X. He *et al.* have explained a solution to extend OneShotSTL for multiple seasonal components decomposition [35]. Extending ASTD for multiple-seasonal components decomposition will be a future direction of our research.

Influence of sliding window size: This limitation is similar to the OnlineSLE method when users set the sliding window size too small. The sliding window refers to the observational or historical data considered for decomposing the time series into trend, seasonal, and residual components. A small sliding window may not capture complete cycles, causing the season length used for decomposition to inaccurately reflect the behavior

of the seasonal component. Consequently, this leads to inaccurate decomposition of both seasonal and trend components. We will investigate how to handle this scenario in future research.

Time series without seasonal component: Our seasonal filter uses single exponential smoothing. It is based on the value between incoming data after initial detrending and the previous observation at $t - m$. Here, m denotes the season length at timestamp t and t denotes the latest timestamp (See as Eq. 5.3).

In the worst-case scenario, ASTD cannot decompose the seasonal component and sets S_{t-m} to zero because OnlineSLE determines there is no seasonal component at $t - m$. Therefore, the output of the seasonal filter at timestamp t will depend solely on the initial detrending and will not utilize previous cycles to consider the seasonal component at timestamp t. This is a limitation of our ASTD. One possible solution is to reconstruct the three components at timestamp $t - m$ into Y_{t-m} and then decompose the seasonal component at timestamp t. The seasonal filter is expressed as:

$$S_t = \gamma \mathcal{DT}_t + (1 - \gamma) Y_{t-m}, \tag{6.1}$$

where S_t denotes the seasonal component at time t, γ is the smoothing factor, and $\mathcal{DT}_t$ is the detrended data at time t. Implementing this solution may help address the issue of missing seasonal components in the ASTD method.

6.4 Conclusion

In summary, our proposed methods—EBinning, OnlineSLE, and ASTD—offer significant advancements in capturing transient patterns, estimating season lengths, and decomposing time series with fluctuating and transitioning seasonal components. Each method addresses specific challenges and contributes to the broader field of time series analysis. However, challenges remain, such as improving computation costs, handling outliers, extending capabilities to anomaly detection, and applying these methods to various domains. Future research will focus on overcoming these limitations and exploring the full potential of these methods. Moreover, we provide the details for reproduction of our proposed method at appendix of this thesis.

Chapter 7

Conclusions

7.1 Summary

In this thesis, we have presented three novel methods designed to enhance the decomposition for the analysis and understanding of four behaviors in time series data: static, dynamic, stable, and unstable behaviors. These methods—Elastic Data Binning (EBinning), Online Season Length Estimation (OnlineSLE), and Adaptive Seasonal-Trend Decomposition (ASTD)—specifically address the unique challenges of decomposing streaming data. Developed to dissect and interpret the intertwined trend, seasonal, and residual components, these techniques offer significant advancements in the field of time series decomposition. In concluding the performance and utility of each proposed method, we observe the following:

- **EBinning:** This method enhances traditional data binning by dynamically adjusting bin sizes based on the characteristics of the original time series. We proposed two

methods: M-EBinning, which compares distributions, and L-EBinning, which focuses on trend differentiation. Our evaluations with synthetic and real-world datasets in time-domain astrophysics show that M-EBinning is highly effective in capturing and sketching transient patterns with high accuracy. However, L-EBinning, the second mode of EBinning, cannot provide high-quality capture and sketching of transient patterns due to the short-lived rise phase of the target transient patterns, which cannot distinguish the trend. Notably, we exhibit M-EBinning for stellar flare capturing with LCs containing flares discovered by M. Aizawa [3].

- **OnlineSLE:** This method is a fast and accurate method for online season length estimation. OnlineSLE facilitates real-time analysis by estimating the season length of the seasonal component within time series. This feature is especially valuable in environments with continuously changing the seasonal component. We utilized the SDFT and spectral peak location estimator, differing from traditional SLE methods that use the FFT and ACF. Our combination yields faster computation and higher accuracy than traditional SLE methods. These results suggest that SLE can be performed relying solely on frequency domain analysis without utilizing ACF. Moreover, it is a crucial part of our ASTD method.

- **ASTD:** This method is a novel method for online STD by utilizing the optimal season length provided by OnlineSLE. ASTD provides a robust framework for decomposing streaming time series data into trend, seasonal, and residual components. This method effectively handles transitions and fluctuations in seasonality, providing detailed insights into dynamic behavior within time series. We conducted extensive

evaluations with synthetic and real-world datasets, specifically from scenarios with short and long data series, and including or excluding transitions and fluctuations in seasonality. Moreover, the real-world datasets are open and have been utilized in numerous publications across various fields such as climatology, ecology, and finance. Our ASTD method yields high-quality decomposition results and effectively manages transitions and fluctuations in seasonality. Additionally, we highlight the trade-offs between our proposed ASTD and existing STD methods. In conclusion, ASTD offers flexibility in decomposing time series without requiring a predefined season length as input.

Overall, these three methods offer flexibility and minimal reliance on predefined parameters, such as bin size, sliding window size, and season length. They provide alternative solutions for addressing dynamic and stable behaviors in time series. While traditional methods can yield better results with optimal parameter selection, they often require time-consuming processes and specific domain knowledge. Our aim is to present these alternative solutions to support scientists in analyzing time series. These methods may serve as a foundation for time series data mining tasks, including prediction, forecasting, and more.

7.2 Future Work

While the developed methods are robust, the exploration of further enhancements and broader applications continues. Here, future work will concentrate on several key areas:

- **Extension for time Series data mining tasks:** We aim to apply the three proposed methods to a variety of data mining tasks, including seasonal adjustment,

anomaly detection, and forecasting. These applications will test the adaptability and effectiveness of the methods in different scenarios not covered in this thesis.

- **Integration of EBinning with ASTD:** This thesis has successfully integrated OnlineSLE with seasonal-trend decomposition to create ASTD, which provides high-quality decomposition results when time series data contains seasonality transitions and fluctuations. There is potential to further integrate EBinning with ASTD. We hypothesize that EBinning, which captures dynamic behavior in the trend component, can offer a more robust solution than the current version of ASTD, which integrates OnlineSLE. Therefore, fully integrating EBinning remains a challenge for further investigation.

- **Reducing computational costs:** In the future, the need for faster computation in streaming data applications will grow. For example, the emergence of large language models requires significant computational hardware. Therefore, efforts will be directed at reducing the computational costs to improve the efficiency of our methods, aiming to achieve a computational complexity of O(1). These improvements will not only refine the methods but also expand their applicability to large language models and enhance performance in real-world scenarios.

It is important to note the numerous opportunities for future research. We expect that scientists across various fields and communities will build upon our proposed methods to further enrich knowledge in the natural sciences. We hope these advancements will lead to deeper insights and a greater understanding within the scientific community.

Bibliography

[1] E. Aboutanios and B. Mulgrew. Iterative frequency estimation by interpolation on fourier coefficients. *IEEE Transactions on Signal Processing*, 53(4):1237–1242, 2005.

[2] C. C. Aggarwal. *An Introduction to Outlier Analysis*. Springer International Publishing, 2017.

[3] M. Aizawa, K. Kawana, K. Kashiyama, R. Ohsawa, H. Kawahara, F. Naokawa, T. Tajiri, et al. Fast optical flares from M dwarfs detected by a one-second-cadence survey with Tomo-e Gozen. *Publications of the Astronomical Society of Japan (PASJ)*, 74(5):1069–1094, 2022.

[4] J. Andrade and M. Estévez-Pérez. Statistical comparison of the slopes of two regression lines: A tutorial. *Analytica Chimica Acta*, 838:1–12, 2014.

[5] K. Bandara, R. J. Hyndman, and C. Bergmeir. MSTL: A seasonal-trend decomposition algorithm for time series with multiple seasonal patterns. *arXiv preprint arXiv:2107.13462*, 2021.

[6] O. Banos, M. A. Toth, M. Damas, H. Pomares, and I. Rojas. Dealing with the effects of sensor displacement in wearable activity recognition. *Sensors*, 14(6):9995–10023, 2014.

[7] E. Bee Dagum and S. Bianconcini. *Linear Filters Seasonal Adjustment Methods: Census Method II and Its Variants*, pages 79–114. Springer International Publishing, 2016.

[8] E. Bee Dagum and S. Bianconcini. *Time Series Components*, pages 29–57. Springer International Publishing, 2016.

[9] A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano. A review on outlier/anomaly detection in time series data. *ACM Comput. Surv.*, 54(3):1–33, 2021.

[10] L. Bluestein. A linear filtering approach to the computation of discrete fourier transform. *IEEE Transactions on Audio and Electroacoustics*, 18(4):451–455, 1970.

[11] G. Box and G. Jenkins. *Time Series Analysis: Forecasting and Control*. Holden-Day, 1976.

[12] G. E. P. Box and D. A. Pierce. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *Journal of the American Statistical Association*, 65(332):1509–1526, 1970.

[13] J. P. Burman. Seasonal adjustment by signal extraction. *Journal of the Royal Statistical Society. Series A (General)*, 143(3):321–337, 1980.

[14] M. J. Campbell and A. M. Walker. A survey of statistical work on the Mackenzie River series of annual Canadian lynx trappings for the years 1821-1934 and a new analysis. *Journal of the Royal Statistical Society*, 140(4):411–431, 1977.

[15] H.-P. Chan, K. I. Konstantinou, and M. Blackett. Spatio-temporal surface temperature variations detected by satellite thermal infrared images at Merapi volcano, Indonesia. *Journal of Volcanology and Geothermal Research*, 420:107405, 2021.

[16] A. Chauhan and K. M. Singh. Recursive sliding DFT algorithms: A review. *Digital Signal Processing*, 127:103560, 2022.

[17] G. Chiarot and C. Silvestri. Time series compression survey. *ACM Comput. Surv.*, 55(10):1–32, 2023.

[18] F. Clette, L. Svalgaard, J. M. Vaquero, and E. W. Cliver. Revisiting the sunspot number. *Space Science Reviews*, 186(1):35–103, Dec 2014.

[19] R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning. STL: A seasonal-trend decomposition. *Journal of Official Statistics*, 6(1):3–73, 1990.

[20] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.

[21] G. Cormode. Current trends in data summaries. *SIGMOD Rec.*, 50(4):6–15, 2022.

[22] G. Cormode and K. Yi. *Small Summaries for Big Data*. Cambridge University Press, 2020.

[23] J. R. A. Davenport, S. L. Hawley, L. Hebb, J. P. Wisniewski, A. F. Kowalski, E. C. Johnson, M. A. Malatesta, et al. Kepler flares. II. the temporal morphology of white-light flares on GJ 1243. *The Astrophysical Journal*, 797(2):122, 2014.

[24] M. M. Dogar, L. Hermanson, A. A. Scaife, D. Visioni, M. Zhao, I. Hoteit, H.-F. Graf, M. A. Dogar, M. Almazroui, and M. Fujiwara. A review of El Niño southern oscillation linkage to strong volcanic eruptions and post-volcanic winter warming. *Earth Systems and Environment*, 7(1):15–42, 2023.

[25] A. Dokumentov and R. J. Hyndman. STR: Seasonal-trend decomposition using regression. *INFORMS Journal on Data Science*, 1(1):50–62, 2021.

[26] K. Duda. Accurate, guaranteed stable, sliding discrete Fourier transform. *IEEE Signal Processing Magazine*, 27(6):124–127, 2010.

[27] M. Elfeky, W. Aref, and A. Elmagarmid. Periodicity detection in time series databases. *IEEE Transactions on Knowledge and Data Engineering*, 17(7):875–887, 2005.

[28] P. Esling and C. Agon. Time-series data mining. *ACM Comput. Surv.*, 45(1):1–34, 2012.

[29] A. C. et al. The astropy project: Building an open-science project and status of the v2.0 core package. *The Astronomical Journal*, 156(3):123, 2018.

[30] E. D. Feigelson, G. J. Babu, and G. A. Caceres. Autoregressive times series methods for time domain astronomy. *Frontiers in Physics*, 6:1–13, 2018.

[31] I. Frías-Blanco, J. d. Campo-Ávila, G. Ramos-Jiménez, R. Morales-Bueno, A. Ortiz-Díaz, and Y. Caballero-Mota. Online and non-parametric drift detection methods based on hoeffding's bounds. *IEEE Transactions on Knowledge and Data Engineering*, 27(3):810–823, 2015.

[32] S. Gharghabi, Y. Ding, C.-C. M. Yeh, K. Kamgar, L. Ulanova, and E. Keogh. Matrix profile VIII: Domain agnostic online semantic segmentation at superhuman performance levels. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 117–126, 2017.

[33] M. J. Graham, A. J. Drake, S. G. Djorgovski, A. A. Mahabal, C. Donalek, V. Duan, and A. Maker. A comparison of period finding algorithms. *Monthly Notices of the Royal Astronomical Society*, 434(4):3423–3444, 08 2013.

[34] S. L. Hawley, J. R. A. Davenport, A. F. Kowalski, J. P. Wisniewski, L. Hebb, R. Deitrick, and E. J. Hilton. Kepler flares. I. active and inactive M dwarfs. *The Astrophysical Journal*, 797(2):121, 2014.

[35] 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. *VLDB*, 16(6):1399–1412, 2023.

[36] Z. He, S. Long, X. Ma, and H. Zhao. A boundary distance-based symbolic aggregate approximation method for time series data. *Algorithms*, 13(11):284, 2020.

[37] T. Heldt, M. Oefinger, M. Hoshiyama, and R. Mark. Circulatory response to passive and active changes in posture. In *Proceedings of the Computers in Cardiology*, pages 263–266, 2003.

[38] G. Helou and C. A. Beichman. The confusion limits to the sensitivity of submillimeter telescopes. *ESA, From Ground-Based to Space-Borne Sub-mm Astronomy*, 29:117–123, 1990.

[39] J. Hochenbaum, O. S. Vallis, and A. Kejariwal. Automatic anomaly detection in the cloud via statistical learning. *arXiv preprint arXiv:1704.07706*, 2017.

[40] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[41] P. J. Huber and E. M. Ronchetti. *Robust Statistics*. Wiley, Jan. 2009.

[42] R. Hyndman. *fpp2: Data for "Forecasting: Principles and Practice" (2nd Edition)*, 2023. R package version 2.5.

[43] R. J. Hyndman. *expsmooth: Data Sets from "Forecasting with Exponential Smoothing"*, 2015. R package version 2.3.

[44] R. J. Hyndman. *fma: Data sets from "Forecasting: methods and applications" by Makridakis, Wheelwright & Hyndman*, 2023. R package version 2.5.

[45] R. J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. OTexts, 3rd edition, 2021.

[46] Influenza Division at Centers for Disease Control and Prevention. Weekly influenza patients. https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html.

[47] E. Jacobsen and P. Kootsookos. Fast, accurate frequency estimators. *IEEE Signal Processing Magazine*, 24(3):123–125, 2007.

[48] E. Jacobsen and R. Lyons. The sliding DFT. *IEEE Signal Processing Magazine*, 20(2):74–80, 2003.

[49] E. Jacobsen and R. Lyons. An update to the sliding DFT. *IEEE Signal Processing Magazine*, 21(1):110–111, 2004.

[50] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, 3(3):263–286, 2001.

[51] E. Keogh, J. Lin, and A. Fu. HOT SAX: efficiently finding the most unusual time series subsequence. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 226 – 233, 2005.

[52] E. Keogh and A. A. Mueen. Time series data mining using the matrix profile: A unifying view of motif discovery, anomaly detection, segmentation, classification, clustering and similarity joins. Tutorials of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017.

[53] J.-H. Kim and T.-G. Chang. Analytic derivation of the finite wordlength effect of the twiddle factors in recursive implementation of the sliding-DFT. *IEEE Transactions on Signal Processing*, 48(5):1485–1488, 2000.

[54] S.-J. Kim, K. Koh, S. Boyd, and D. Gorinevsky. ℓ_1 trend filtering. *SIAM Review*, 51(2):339–360, 2009.

[55] R. Koenker. *Quantile Regression*. Econometric Society Monographs. Cambridge University Press, 2005.

[56] R. R. Laher, V. Gorjian, L. M. Rebull, F. J. Masci, J. W. Fowler, G. Helou, S. R. Kulkarni, and N. M. Law. Aperture photometry tool. *Publications of the Astronomical Society of the Pacific*, 124(917):737, 2012.

[57] X. Lan, P. Tans, and K. Thoning. Trends in globally-averaged CO2 determined from NOAA Global Monitoring Laboratory measurements, 2023. https://gml.noaa.gov/ccgg/trends/gl_data.html.

[58] S. M. Law. STUMPY: A Powerful and Scalable Python Library for Time Series Data Mining. *The Journal of Open Source Software*, 4(39):1504, 2019.

[59] K. Lee, Y. Jeong, S. Joo, Y. S. Yoon, S. Han, and H. Baik. Outliers in financial time series data: Outliers, margin debt, and economic recession. *Machine Learning with Applications*, 10:100420, 2022.

[60] L. Li and F. Zhang. Hourly and sub-hourly rainfall under synoptic patterns during the anomalous Meiyu season 2020. *Atmosphere*, 14(4):727, 2023.

[61] J.-R. Liao and S. Lo. Analytical solutions for frequency estimators by interpolation of dft coefficients. *Signal Processing*, 100:93–100, 2014.

[62] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery (DMKD)*, pages 2–11, 2003.

[63] J. Lin, E. Keogh, L. Wei, and S. Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Mining and Knowledge Discovery*, 15(2):107–144, 2007.

[64] X. Lin, H. Lu, J. Xu, and J. Yu. Continuously maintaining quantile summaries of the most recent n elements over a data stream. In *Proceedings of the 20th International Conference on Data Engineering*, pages 362–373, 2004.

[65] Y. Liu, Z. Nie, Z. Zhao, and Q. H. Liu. Generalization of iterative fourier interpolation algorithms for single frequency estimation. *Digital Signal Processing*, 21(1):141–149, 2011.

[66] Y. Liu, Z. Xu, H. Zhou, and Y. Zhao. An efficient algorithm for frequency estimation of sinusoid signal based on improved Quinn. *Journal of Physics: Conference Series*, 1952(4):042094, 2021.

[67] A. M. D. Livera, R. J. Hyndman, and R. D. Snyder. Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American Statistical Association*, 106(496):1513–1527, 2011.

[68] B. Lkhagva, Y. Suzuki, and K. Kawagoe. New time series data representation ESAX for financial applications. In *Proceedings of the 22nd International Conference on Data Engineering Workshops (ICDEW)*, pages 17–22, 2006.

[69] B. Lott, L. Escande, S. Larsson, and J. Ballet. An adaptive-binning method for generating constant-uncertainty/constant-significance light curves with fermi-lat data. *Astronomy & Astrophysics (A&A)*, 544:A6, 2012.

[70] R. Lyons and C. Howard. Improvements to the sliding discrete Fourier transform algorithm. *IEEE Signal Processing Magazine*, 38(4):119–127, 2021.

[71] S. Malinowski, T. Guyet, R. Quiniou, and R. Tavenard. 1d-SAX: A novel symbolic representation for time series. In *Proceedings of the Advances in Intelligent Data Analysis XII*, pages 273–284, 2013.

[72] K. Mandel and E. Agol. Analytic light curves for planetary transit searches. *The Astrophysical Journal*, 580(2):L171, 2002.

[73] A. Mishra, R. Sriharsha, and S. Zhong. OnlineSTL: Scaling time series decomposition by 100x. *VLDB*, 15(7):1417–1425, 2022.

[74] Y. Nam, P. Trirat, T. Kim, Y. Lee, and J.-G. Lee. Context-aware deep time-series decomposition for anomaly detection in businesses. In *Proceedings of the European Conference on Machine Learning and Data Mining (ECML-PKDD)*, pages 330–345, 2023.

[75] National Centers for Enviromental Information. Daily sea surface temperature (SST). https://climatereanalyzer.org/clim/sst_daily/.

[76] National Centers for Enviromental Information. Southern oscillation index (SOI). https://www.ncei.noaa.gov/access/monitoring/enso/soi.

[77] J. Nordin, V. Brinnel, J. van Santen, M. Bulla, U. Feindt, A. Franckowiak, C. Fremling, et al. Transient processing and analysis using AMPEL: alert management, photometry, and evaluation of light curves. *Astronomy & Astrophysics (A&A)*, 631:A147, 2019.

[78] J. G. Olivier, K. Schure, J. Peters, et al. Trends in global CO2 and total greenhouse gas emissions. *PBL Netherlands Environmental Assessment Agency*, 5:1–11, 2017.

[79] J. Paparrizos, Y. Kang, P. Boniol, R. S. Tsay, T. Palpanas, and M. J. Franklin. TSB-UAD: an end-to-end benchmark suite for univariate time-series anomaly detection. *VLDB*, 15(8):1697–1711, 2022.

[80] P. Peñil, A. Domínguez, S. Buson, M. Ajello, J. Otero-Santos, J. A. Barrio, R. Nemmen, S. Cutini, B. Rani, A. Franckowiak, and E. Cavazzuti. Systematic search for γ-ray periodicity in active galactic nuclei detected by the Fermi large area telescope. *The Astrophysical Journal*, 896(2):134, 2020.

[81] A. Petralia and G. Micela. Principal component analysis to correct data systematics. case study: K2 light curves. *Experimental Astronomy*, 49(3):97–114, 2020.

[82] R. A. Phillipson. Complex Long-Term Variability of X-ray Binaries and Active Galaxies Revealed by Novel Methods. *Bulletin of the AAS*, 52(3), 2020.

[83] T. Phungtua-eng, S. Sako, Y. Nishikawa, and Y. Yamamoto. Elastic data binning: Time-series sketching for time-domain astrophysics analysis. *SIGAPP Appl. Comput. Rev.*, 23(2):5–22, 2023.

[84] T. Phungtua-Eng and Y. Yamamoto. A fast season length estimation using sliding discrete fourier transform for time series streaming data. In *Proceedings of the 16th International Congress on Advanced Applied Informatics*, pages 482–487.

[85] T. Phungtua-Eng and Y. Yamamoto. A novel framework of non-parametric for adjusting the window size. Technical Report 5, Department of Informatics, Shizuoka University, Department of Informatics, Shizuoka University, 2022.

[86] T. Phungtua-Eng and Y. Yamamoto. Adaptive seasonal-trend decomposition for streaming time series data with transitions and fluctuations in seasonality. In *Proceedings of the 2024 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, 2024. (To appear).

[87] T. Phungtua-eng, Y. Yamamoto, and S. Sako. Transient pattern detection from streaming nature data. In *Proceedings of the 8th International Symposium on Computing and Networking Workshops*, pages 435–439, 2020.

[88] T. Phungtua-Eng, Y. Yamamoto, and S. Sako. Detection for transient patterns with unpredictable duration using chebyshev inequality and dynamic binning. In *Proceedings of the 9th International Symposium on Computing and Networking Workshops*, pages 454–458, 2021.

[89] T. Phungtua-Eng, Y. Yamamoto, and S. Sako. Dynamic binning for the unknown transient patterns analysis in astronomical time series. In *Proceedings of the 2021 IEEE International Conference on Big Data (BigData)*, pages 5988–5990, 2021.

[90] T. Phungtua-Eng, Y. Yamamoto, and S. Sako. Supplementary material for "elastic data binning for transient pattern analysis in time-domain astrophysics", 12 2022. https://sites.google.com/view/elasticdatabinning.

[91] T. Phungtua-Eng, Y. Yamamoto, and S. Sako. Elastic data binning for transient pattern analysis in time-domain astrophysics. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing (SAC)*, pages 342–349, 2023.

[92] T. Puech, M. Boussard, A. D'Amato, and G. Millerand. A fully automated periodicity detection in time series. In *Proceedings of the Advanced Analytics and Learning on Temporal Data*, pages 43–54, 2020.

[93] B. Quinn. Estimation of frequency, amplitude, and phase from the dft of a time series. *IEEE Transactions on Signal Processing*, 45(3):814–817, 1997.

[94] C. A. Ralanamahatana, J. Lin, D. Gunopulos, E. Keogh, M. Vlachos, and G. Das. *Mining Time Series Data*, pages 1069–1103. 2005.

[95] A. Raudys, V. Lenčiauskas, and E. Malčius. Moving averages for financial data smoothing. In *Proceedings of the Information and Software Technologies*, pages 34–45, 2013.

[96] U. Rebbapragada, P. Protopapas, C. E. Brodley, and C. Alcock. Finding anomalous periodic time series. *Machine Learning*, 74(3):281–313, 2009.

[97] A. K. Roonizi. ℓ_2 and ℓ_1 trend filtering: A Kalman filter approach. *IEEE Signal Processing Magazine*, 38(6):137–145, 2021.

[98] R. Ruela, M. Sousa, M. deCastro, and J. Dias. Global and regional evolution of sea surface temperature under climate change. *Global and Planetary Change*, 190:103190, 2020.

[99] S. Sako, R. Ohsawa, H. Takahashi, Y. Kojima, M. Doi, N. Kobayashi, T. Aoki, et al. The Tomo-e Gozen wide field CMOS camera for the Kiso Schmidt telescope. In *Proceedings of the Ground-based and Airborne Instrumentation for Astronomy VII (SPIE)*, page 107020J, 2018.

[100] A. B. Sanders. The Subprime Mortgage Crisis. In *Public Real Estate Markets and Investments*. Oxford University Press, 09 2014.

[101] K. Sayood. *Introduction to Data Compression*. The Morgan Kaufmann Series in Multimedia Information and Systems. Morgan Kaufmann, 3rd edition, 2006.

[102] J. D. Scargle. Studies in astronomical time series analysis. I - modeling random processes in the time domain. *Astrophysical Journal Supplement Series*, 45:1–71, 1981.

[103] A. Schuster. On lunar and solar periodicities of earthquakes. *Proceedings of the Royal Society of London*, 61(369-377):455–465, 1897.

[104] A. Serbes. Fast and efficient sinusoidal frequency estimation by using the DFT coefficients. *IEEE Transactions on Communications*, 67(3):2333–2342, 2019.

[105] M. Shahcheraghi, R. Mercer, J. M. De Almeida Rodrigues, A. Der, H. F. S. Gamboa, Z. Zimmerman, and E. Keogh. Matrix profile XXVI: Mplots: Scaling time series similarity matrices to massive data. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 1179–1184, 2022.

[106] G. Shevlyakov and M. Kan. Stream data preprocessing: Outlier detection based on the Chebyshev inequality with applications. In *Proceedings of the 26th Conference of Open Innovations Association (FRUCT)*, pages 402–407, 2020.

[107] R. H. Shumway and D. S. Stoffer. *Time Series Analysis and Its Applications: With R Examples*. Springer International Publishing, 2017.

[108] S. Soltani. On the use of the wavelet decomposition for time series prediction. *Neurocomputing*, 48(1):267–277, 2002.

[109] D. Stoffer and N. Poison. *astsa: Applied Statistical Time Series Analysis*, 2023. R package version 2.0.

[110] R. Sulo, T. Berger-Wolf, and R. Grossman. Meaningful selection of temporal resolution for dynamic networks. In *Proceedings of the Eighth Workshop on Mining and Learning with Graphs (MLG)*, pages 127–136, 2010.

[111] Y. Sun, J. Li, J. Liu, B. Sun, and C. Chow. An improvement of symbolic aggregate approximation distance measure for time series. *Neurocomputing*, 138:189–198, 2014.

[112] Sunspot Index and Long-term Solar Observations. Sunspot number dataset, 2023. https://www.sidc.be/SILSO/datafiles.

[113] J. W. Taylor. Short-term electricity demand forecasting using double seasonal exponential smoothing. *Journal of the Operational Research Society*, 54(8):799–805, 2003.

[114] S. J. Taylor and B. Letham. Forecasting at scale. *PeerJ Preprints*, pages 1–25, 2017.

[115] M. Theodosiou. Forecasting monthly and quarterly time series using STL decomposition. *International Journal of Forecasting*, 27(4):1178–1195, 2011.

[116] A. M. Thieler, R. Fried, and J. Rathjens. RobPer: An R package to calculate periodograms for light curves based on robust regression. *Journal of Statistical Software*, 69(9):1–37, 2016.

[117] R. Tolas, R. Portase, A. Iosif, and R. Potolea. Periodicity detection algorithm and applications on IoT data. In *Proceedings of the 20th International Symposium on Parallel and Distributed Computing (ISPDC)*, pages 81–88, 2021.

[118] M. Toller, T. Santos, and R. Kern. SAZED: parameter-free domain-agnostic season length estimation in time series data. *Data Mining and Knowledge Discovery*, 33(6):1775–1798, 2019.

[119] A. Tuzcu Kokal, I. Ismailoglu, N. Musaoglu, and A. Tanik. Detection of surface temperature anomaly of the sea of Marmara. *Advances in Space Research*, 71(7):2996–3004, 2023.

[120] M. Vlachos, P. Yu, and V. Castelli. On periodicity detection and structural periodic similarity. In *SIAM International Conference on Data Mining (SDM)*, pages 449–460, 2005.

[121] J. Wang, T. Chen, and B. Huang. Cyclo-period estimation for discrete-time cyclostationary signals. *IEEE Transactions on Signal Processing*, 54(1):83–94, 2006.

[122] X. Wang, K. Smith, and R. Hyndman. Characteristic-based clustering for time series data. *Data Mining and Knowledge Discovery*, 13(3):335–364, Nov 2006.

[123] B. D. Warner. *A Practical Guide to Lightcurve Photometry and Analysis*. Springer Cham, 2nd edition, 2016.

[124] B. L. Welch. The significance of the difference between two means when the population variances are unequal. *Biometrika*, 29(3/4):350–362, 1938.

[125] Q. Wen, J. Gao, X. Song, L. Sun, and J. Tan. RobustTrend: A huber loss with a combined first and second order difference regularization for time series trend filtering. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3856–3862, 2019.

[126] 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 *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 5409–5416, 2019.

[127] Q. Wen, K. He, L. Sun, Y. Zhang, M. Ke, and H. Xu. RobustPeriod: Robust time-frequency mining for multiple periodicity detection. In *Proceedings of th International Conference on Management of Data*, pages 2328–2337, 2021.

[128] Q. Wen, L. Yang, and L. Sun. Robust dominant periodicity detection for time series with missing data. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023.

[129] Q. Wen, Z. Zhang, Y. Li, and L. Sun. Fast RobustSTL: Efficient and robust seasonal-trend decomposition for time series with complex patterns. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, pages 2203–2213, 2020.

[130] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun. Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125*, 2023.

[131] H. Wu, J. Xu, J. Wang, and M. Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 2021.

[132] Z. Wu, N. E. Huang, S. R. Long, and C.-K. Peng. On the trend, detrending, and variability of nonlinear and nonstationary time series. *Proceedings of the National Academy of Sciences*, 104(38):14889–14894, 2007.

[133] D. Yang, V. Sharma, Z. Ye, L. I. Lim, L. Zhao, and A. W. Aryaputera. Forecasting of global horizontal irradiance by exponential smoothing, using decompositions. *Energy*, 81:111–119, 2015.

[134] C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. Keogh. Matrix profile I: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 1317–1322, 2016.

[135] L. Yuan, B. Pfahringer, and J. P. Barddal. Addressing feature drift in data streams using iterative subset selection. *SIGAPP Appl. Comput. Rev.*, 19(1):20–33, 2019.

[136] C. Zhang, T. Zhou, Q. Wen, and L. Sun. TFAD: A decomposition time series anomaly detection architecture with time-frequency analysis. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 2497–2507, 2022.

[137] Y. Zhang, G. Li, B. Muskat, and R. Law. Tourism demand forecasting: A decomposed deep learning approach. *Journal of Travel Research*, 60(5):981–997, 2021.

[138] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Conference*, volume 35, pages 11106–11115, 2021.

[139] X. Zhu and D. Guo. Urban event detection with big data of taxi od trips: A time series decomposition approach. *Transactions in GIS*, 21(3):560–574, 2017.

[140] Y. Zhu, S. Gharghabi, D. F. Silva, H. A. Dau, C.-C. M. Yeh, N. Shakibay Senobari, A. Almaslukh, K. Kamgar, Z. Zimmerman, G. Funning, A. Mueen, and E. Keogh. The swiss army knife of time series data mining: ten useful things you can do with the matrix profile and ten lines of code. *Data Mining and Knowledge Discovery*, 34(4):949–979, 2020.

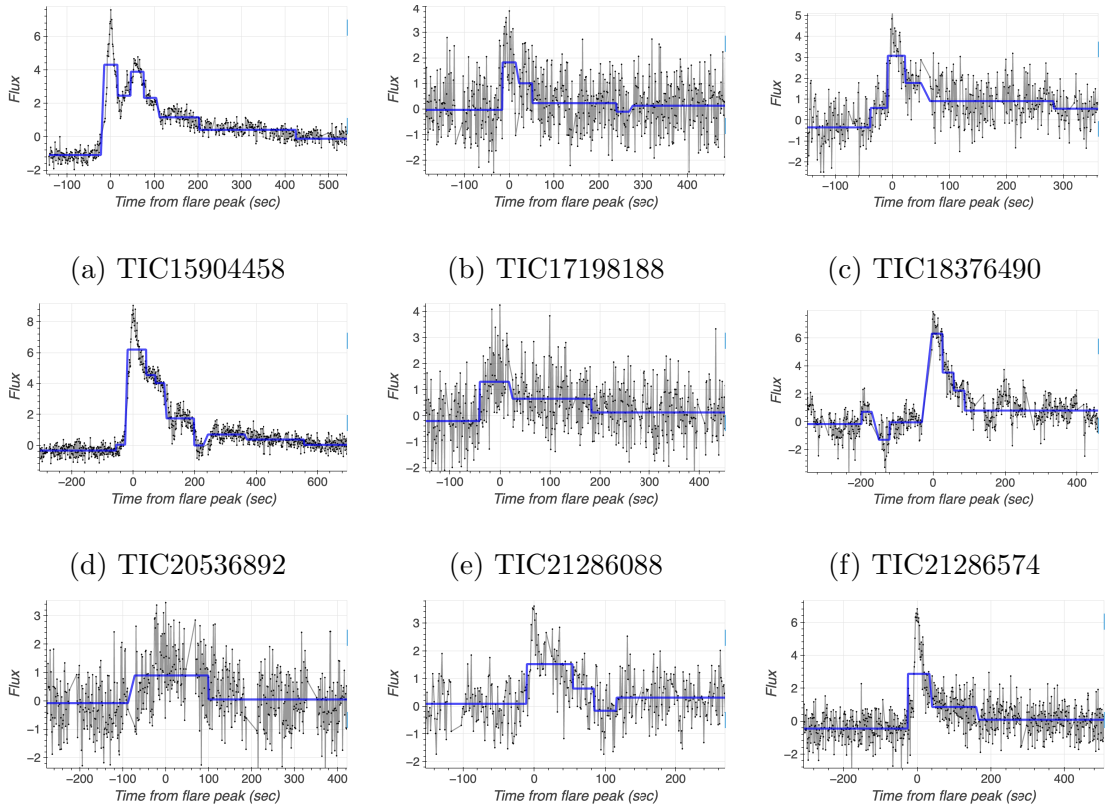
[141] Y. Zhu, Z. Zimmerman, N. S. Senobari, C.-C. M. Yeh, G. Funning, A. Mueen, P. Brisk, and E. Keogh. Matrix profile II: Exploiting a novel algorithm and gpus to break the one hundred million barrier for time series motifs and joins. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 739–748, 2016.

Appendix A

Supplementary Materials

A.1 Analyzing Flares with M-EBinning

As mentioned in Section 3.6.2, this section showcases the analysis of LCs with stellar flares discovered by Aizawa *et al.* [3]. The results are shown as Figures A.1 and A.2. In all figures, the black lines represent the LCs, and the blue lines represent the results of M-EBinning. The x-axes represent time from the flare peaks in seconds. For a detailed comparison, we recommend reviewing these results alongside Appendix 2 of the publication by Aizawa *et al.* [3].



(a) TIC15904458 (b) TIC17198188 (c) TIC18376490

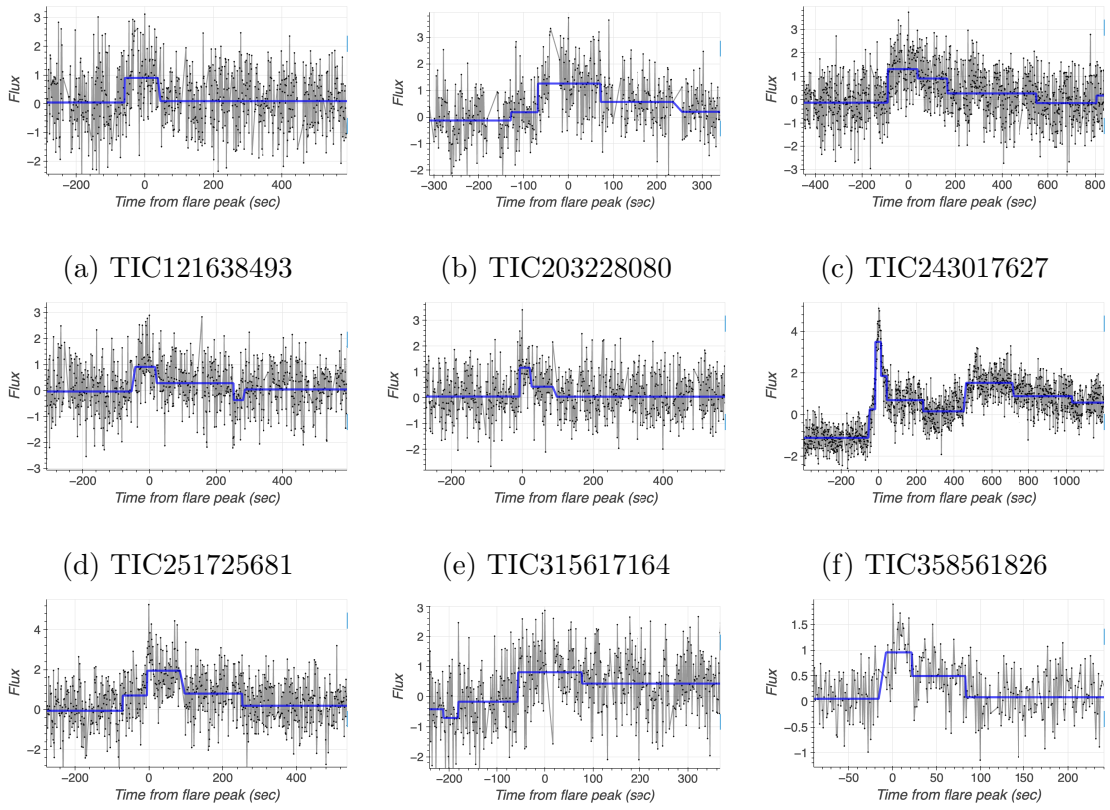
(d) TIC20536892 (e) TIC21286088 (f) TIC21286574

(g) TIC46305452 (h) TIC55288759 (i) TIC119031076

Figure A.1: All flare periods representation results obtained using M-EBinning.

A.2 Datasets and Source Codes

Here, we summarize all the datasets and source codes utilized in this thesis for reproducibility, as detailed in Tables A.1 and A.4.



(a) TIC121638493 (b) TIC203228080 (c) TIC243017627

(d) TIC251725681 (e) TIC315617164 (f) TIC358561826

(g) TIC358573853 (h) TIC366485128 (i) TIC435904068

Figure A.2: All flare periods representation results obtained using M-EBinning (continued).

Table A.1: List of datasets.

Dataset name	Chapter	Source
Sqaure-LCs	3	Supplementary website for EBinning publication [90]
Triangle-LCs	3	Supplementary website for EBinning publication [90]
Kepler-LCs	3	Supplementary website for EBinning publication [90]
Aizawa dataset	3	Provided by M. Aizawa [3]
Sythetic dataset	4	Supplementary website for OnlineSLE publication [84]
ECG	4	Supplementary website of Matrix Profile VIII publication [32]
ABP	4	Supplementary website of Matrix Profile VIII publication [32]
Sunspots	4	"Sunspot Index and Long-term Solar Observation" website [112]
Sythetic dataset	5	Supplementary website for ASTD publication [86]
Real1 dataset	5	Supplementary website for ASTD publication [86]
astsa (Real2)	5	Available on CRAN package [109]
fma (Real2)	5	Available on CRAN package [44]
expsmooth (Real2)	5	Available on CRAN package [43]
fpp2 (Real2)	5	Available on CRAN package [42]

Table A.2: List of methods in Chapter 3.

Abbreviation	Full Name	Implementation Source	Language
M-EBinning	Mean-Elastic Data Binning [90]	Own	Python
L-EBinning	Linear-Elastic Data Binning [90]	Own	Python
DyBin (or PSY)	Dynamic Binning [88]	Own	Python
TWIN	Temporal Window In Networks [110]	Own	Python
PAA	Piecewise Aggregate Approximation (or Classical Data binning) [50]	scikit-learn	Python
SAX	Symbolic Aggregate approXimation [62]	scikit-learn	Python
1D-SAX	1D-Symbolic Aggregate approXimation [71]	scikit-learn	Python
MP	Matrix Profile [134]	Stumpy	Python
FLUSS	Fast Low-cost Unipotent Semantic Segmentation [32]	Stumpy	Python
HOT-SAX	Heuristically Ordered Time Series using Symbolic Aggregate Approximation [51]	jmotif	R

Table A.3: List of methods in Chapter 4.

Abbreviation	Full Name	Implementation Source	Language
OnlineSLE	Online Season Length Esitmation [84]	Own	Python
SDFT	Sliding Discrete Fourier transform [48, 49]	Own	Python, R
Quinn	Quinn Estimator [93]	Own	Python
QSE	Q-shift Estimator [104]	Own	Python
		Original	Matlab
HAQSE	Hybrid Aboutanios and Mulgrew and q-shift estimator [104]	Own	Python
		Original	Matlab
FindLength	FindLength [79]	TSB-UAD	Python
AutoPeriod	AutoPeriod [120]	Periodicity detection	Python
SAZED	Spectral and Average Autocorrelation ZEro Distance density [118]	sazedR	R

Table A.4: List of methods in Chapter 5.

Abbreviation	Full Name	Implementation Source	Language
ASTD	Adaptive Seasonal-Trend Decomposition [86]	Own	Python
SlidingSTL	STL based on the sliding window [86]	Own	Python
OnlineSTL	OnlineSTL [73]	Own	Python
STL	Seasonal-trend Decomposition using LOESS [19]	statsmodels	Python
MSTL	Multiple Seasonal-Trend Decomposition using LOESS [5]	forecast	R
STR	Seasonal-Trend decomposition using Regression [25]	stR	R
RobustSTL	RobustSTL [126]	Original	Python
Fast-RobustSTL	Fast-RobustSTL [129]	Original	Python
OneShotSTL	OneShotSTL [35]	Original	Java