University of Amsterdam

Masters Thesis

Recurring Payments Detection in Transactional Time Series

Author:
Filippos Dimopoulos

Supervisor
Fabian Jansen

A thesis submitted in partial fulfilment of the requirements for the degree of Master of Science in Computational Science

in the

Computational Science Lab
Informatics Institute

February 2025

Declaration of Authorship

I, Filippos Dimopoulos, declare that this thesis, entitled 'Recurring Payments Detection in Transactional Time Series' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at the University of Amsterdam.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. Except such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date: 20 January 2024

"What I cannot create, I do not understand."

Richard P. Feynman

UNIVERSITY OF AMSTERDAM

Abstract

Faculty of Science

Informatics Institute

Master of Science in Computational Science

Recurring Payments Detection in Transactional Time Series

by Filippos Dimopoulos

The optimisation of recurring payment patterns in wholesale banking is a challenging task due to the irregular, large-volume transactions. Accurately identifying payment structures and thus the relationship between two businesses is crucial for ING. To address this, we present a novel unsupervised approach to detect and optimise recurring payment patterns by employing a windowed optimisation process, to first align the patterns as well as possible and then filter out the noise.

Our approach recognizes individual payments as part of specific patterns, even when those payments exhibit slight variations in timing or amount. The proposed algorithm, through a series of operations -moving, swapping, removing and a combination of those- first optimises the timeline by aligning payments, in an attempt to form an initial recurring payment structure. Then, in a windowed manner, the payments that impose the highest penalty on the objective are removed to refine the signal. This process works originally on smaller windows to detect patterns that do not extend to the end of the timeline and then expands to gradually cover the whole timeline. Additionally, the approach prioritises lower periodicities before moving to the higher ones, ensuring the accurate detection of nested patterns.

Through a series of experiments, we evaluate the algorithm's ability to handle multiple levels of noise, different periodicities, and complex payment structures, while also investigating its performance in detecting incomplete patterns and handling gaps in payment timelines. The results highlight the algorithm's robustness in maintaining pattern integrity under varying conditions, though some limitations exist. Patterns were accurately retrieved even when the noise was more than 60% of the data and the amount fluctuation was about 20%. However, as noise levels reached the 70% mark, the algorithm encounters challenges, especially when the pattern payments exhibit high variation.

Moreover, the expanding window approach is found to have a significant impact on the algorithm's capacity to optimise patterns efficiently. Future research could explore more sophisticated noise-filtering techniques and combinations of optimisation operations to enhance performance further. This study lays the groundwork for more advanced recurring pattern recognition methods, with applications that are not limited to wholesale banking.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Fabian, whose guidance, support, and unwavering patience have been invaluable throughout this journey. Fabian, your ability to stay calm and composed, even when I was bombarding you with questions (many of which I thought were quite silly), has not only been a source of comfort but also a great learning experience. The amount of help you have provided me and the guidance you offered at all times helped me grow.

To my family, thank you for your endless support in all my pursuits. No matter where I go or what I aim to achieve, you've always been there to cheer me on.

My heartfelt thanks also go to my friends, who have been by my side from day one. Your support, and quite often your suggestions, have helped me to push through this challenge.

Finally, I extend my appreciation to everyone I've had the pleasure to work with in the Master's program in Computational Science. A special thanks to Nienke, who has been a tremendous help with all things administrative and has offered support through challenging moments. Your kindness and assistance have made a huge difference in my journey.

To all those who have supported, encouraged, and inspired me along the way—thank you.

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

CSL	**C**omputational **S**cience **L**ab
UvA	**U**niversiteit **v**an **A**msterdam
DTW	**D**ynamic **T**ime **W**arping
MP	**M**atrix **P**rofile
B2B	**B**usiness **T**o **B**usiness
SAX	**S**ymbolic **A**ggregate appro**X**imation
PCA	**P**rincipal **C**omponent **A**nalysis
RS	**R**amanujan **S**ums
ESPRIT	**E**stimation of Signal **P**arameters via **R**otational **I**nvariance **T**echniques
PAST	**P**rojection **A**prroximation **S**ubspace **T**racking
MUSIC	**MU**ltiple **SI**gnal **C**lassification
DFT	**D**iscrete **F**ourier **T**ransform
KDE	**K**ernel **D**ensity **E**stimation
ING	**I**nternationale **N**ederlanden **G**roep

Chapter 1

Introduction

1.1 Motivation

In the rapidly evolving financial world, banks are constantly striving to optimise their operations to improve efficiency, reduce costs, and ensure compliance. This is especially important in wholesale banking, where large transactions between financial institutions and corporations are the norm. Managing and optimising these payment flows is critical to ensure smooth operations, reduce risks, and maintain the stability of the financial system.

For banks, the ability to identify patterns in business-to-business transactions is a powerful tool in this optimisation process. Recognising these patterns can improve risk management, boost operational efficiency, and detect anomalies that might indicate fraud [1]. Furthermore, it allows banks to anticipate customer needs, tailor their services more effectively, and make better strategic decisions.

Over the past decade, electronic payments have become the standard, with a significant increase in their usage. For example, De Nederlandsche Bank (DNB) reports an 80% rise in electronic payments since 2013 [2]. This shift has given banks access to more data than ever before, enabling them to not only record transactions but also predict future behaviour. However, identifying patterns in B2B transactions is far more challenging than in consumer payments, as they are subject to greater variability and complexity.

1.2 Challenges in Business-to-Business(B2B) Transactions

Business-to-business (B2B) transactions refer to financial exchanges between companies, typically involving high-value payments. These transactions are significantly more complex than consumer transactions, as they are driven by various factors, such as contractual agreements, seasonality and other business or market specific factors. In contrast to consumer transactions, which are often predictable and repetitive (e.g., regular utility bills or subscription services), B2B transactions are characterised by variability in amounts, frequencies, and timings. Moreover, payments are rarely labelled or structured, which adds to the difficulty of categorising them into patterns.

One of the main challenges in wholesale banking is characterising relationships between businesses based on their payment flows. Companies often engage in repeated transactions, forming intricate financial networks. If recurring transactions can be detected, it indicates an ongoing business relationship, providing banks like ING with valuable insights. By identifying such patterns, banks can anticipate future payments, improve forecasts, and mitigate potential risks. However, these recurring transactions are often obscured by noise, ie. irregular, sporadic payments that do not follow a predictable pattern. Noise can arise from one-off transactions, for example business events, or irregular cash flow changes, complicating the detection of stable recurring patterns.

Unlike consumer transactions, which tend to be more predictable, B2B transactions are highly variable and less structured. The sheer volume of data in the wholesale banking sector, combined with a lack of labelling, limits the use of supervised learning methods to detect patterns. Consequently, there is a need for unsupervised approaches that can effectively filter out noise and identify recurring payment flows.

1.3 Challenges in Pattern Recognition

Finding recurring payment patterns in wholesale banking data is a tricky task. Unlike retail banking, where transactions can often be categorised based on who the recipient is, wholesale payments do not come with this context. For instance, in retail banking, a payment to a known supermarket can be easily labelled as "groceries". But in wholesale banking, the only information might be the payment amount and the date, making it much harder to categorize. This makes grouping payments into meaningful patterns more challenging.

Another significant challenge is that recurring payments may not always occur on the exact same day as expected. For example, a payment that is supposed to be made every

Wednesday might sometimes be shifted to a Tuesday, Thursday, or even a more distant day. These misalignments can disrupt the periodic structure, complicating pattern recognition.

To address these challenges, it is necessary to adjust the payments to form a clear pattern. In cases where a payment appears on Wednesday but is not of the expected amount, and another payment matching the expected amount occurs on Tuesday, the algorithm developed for this thesis will identify this discrepancy. It will then swap the payments to create a recurrence with the expected pattern and remove any payments that do not conform to it. Essentially, the algorithm repositions payments to match a "perfect" signal, one where payments occur at regular intervals based on their periodicity and the day of the initial occurrence.

The timeline is cluttered with noise and misaligned transactions that deviate from their expected positions. The core objective of the algorithm is twofold: to remove noisy payments and to realign payments that are "out of position" so that they conform to a perfect recurring pattern. Payments outside the expected periodic intervals are filtered out, leaving a cleaned and optimised signal that represents the recurring patterns. This process ensures that the resulting signal is as close as possible to the ideal pattern.

1.4 Objective

To summarise, this research aims to implement various optimisation operations, such as moving and removing payments as well as a combination of these, to refine the signal and identify repeating patterns in a time series in an unsupervised manner. The significance of this approach stems from the irregular nature of transactions in the wholesale banking sector. Unlike consumer payments, wholesale transactions do not follow predictable schedules, making traditional techniques that search for exact payment matches ineffective. By allowing flexible adjustments within defined ranges, this work seeks to uncover patterns that might otherwise remain hidden due to variability in payment behaviour.

The core research question addressed in this project is: Can we identify recurring payment patterns in a cluttered timeline of transactions based on previously known periodicities, and differentiate them from noise? For instance, consider a timeline of 365 days where random payments occur, alongside payments forming weekly or monthly patterns. The objective is to identify which payments belong to each recurring pattern and flag the payments that are considered noise—those that are one-off or cannot be associated with any identifiable periodic pattern. This process aims to optimise the detection of

meaningful payment structures while filtering out irregular transactions that do not fit into a consistent pattern.

1.5 Outline

The following chapters are structured as follows. In Chapter 2 we review related spectral, supervised and unsupervised methods, on the topic of recurring pattern detection. Then, we lay the foundation by explaining the necessary theoretical background to understand the next steps. We review the Matrix Profile, the state-of-the-art unsupervised method to detect patterns currently and its limitations. In Chapter 3, we describe the proposed solution to detect recurring patterns. We delve into the objective of this thesis, mathematically and conceptually, and the operations that can be performed to reach it. In Chapter 4, we investigate if the proposed solution manages to succeed in revealing the pattern in various situations. In Chapter 5, we draw conclusions from the results gathered in the last chapter and discuss the limitations and whether the work of this thesis addresses the research question. Finally, in Chapter 6 we conclude with the most important contributions of this thesis, and suggest future steps in this research direction.

Chapter 2

Literature review

Detecting meaningful patterns in wholesale banking data is a complex and multifaceted challenge. Unlike consumer banking, where transactions often follow predictable trends, wholesale banking transactions are characterized by unlabelled, high-frequency data with overlapping periodicities. Payments may occur weekly, monthly, or even quarterly, and these varying cycles can create fragmented patterns that are difficult to isolate. Further complicating the problem are irregularities such as misaligned transactions, missing payments, and significant noise, which obscure the underlying structure of the data.

While existing methods provide valuable tools for identifying patterns or anomalies, they often fail to address these unique challenges comprehensively. This chapter reviews key methodologies, from anomaly detection and motif discovery to signal processing approaches, highlighting their strengths and limitations. It also underscores the specific gaps—such as quasi-periodicity handling and dynamic misalignments—that motivate the development of the algorithm proposed in this thesis.

2.1 Anomaly Detection

Anomaly detection and repeated pattern recognition, while distinct tasks, share a strong conceptual connection. Both techniques aim to simplify complex datasets by identifying deviations from expected behaviour. Identifying recurring patterns, as stated in the first chapter, allows banks to forecast if, potentially, a business is or will be operational. Anomalies, on the other hand, are often defined as breaks or deviations from these patterns, such as missing or misaligned payments in an otherwise periodic timeline. Conversely, identifying a pattern often requires first isolating and removing anomalies to reveal the underlying regularities.

Triepels et al. [3] employ autoencoders to model normal payment behaviour and detect anomalies. By learning a compressed representation of the data, the autoencoder reconstructs payments and flags high reconstruction errors as anomalies. This unsupervised method is valuable for identifying deviations in payment timelines. However, this approach lacks the ability to classify payments into specific recurring patterns or periodicities.

Similarly, Leon et al. [4] apply principal component analysis (PCA) to reduce the dimensionality of payment data and utilize clustering algorithms to group payments based on their similarities. Deviations from the norms of these clusters are flagged as anomalies. While this approach effectively identifies irregularities in payment behaviours, it does not explicitly identify or classify payments into distinct recurring patterns or periodicities, limiting its applicability for recurring payment detection. Additionally, it focuses solely on identifying deviations rather than classifying or aligning payments within multiple periodicities.

In another approach, Leon et al. [5] develop a supervised machine learning methodology to detect anomalous payments. Using labelled data, the model classifies payments as either normal or anomalous based on historical patterns. Although this method demonstrates robust performance in anomaly detection, it is heavily reliant on labelled datasets, making it less suitable for the unsupervised nature of wholesale banking datasets. Additionally, like the previous methods, it does not assign payments to specific periodicities, focusing solely on the identification of anomalies.

2.2 Motif Discovery

Motif discovery refers to the task of identifying recurring subsequences, also known as motifs, within time series data that exhibit similar patterns or behaviours. These motifs may not necessarily be consecutive and can occur sporadically throughout the timeline. This process is particularly relevant for time-series analysis, where the identification of repeating patterns is essential for understanding underlying behaviours, forecasting, or detecting anomalies. In the context of wholesale banking, motif discovery aligns with the goal of isolating meaningful repetitions from a cluttered timeline.

Chiu et al. [6] introduce a method that employs Symbolic Aggregate approXimation (SAX) [7] to convert time series data into symbolic sequences. SAX discretizes the time series into symbolic subsequences, making the data easier to manage and compare. Then, a random projection technique is used, which involves randomly selecting points on two subsequences and comparing them. This approach is effective in capturing subsequences

that exhibit similar behaviours without the computational burden of exhaustive pairwise comparisons. This approach reduces computational complexity while identifying motifs but lacks the ability to handle multiple periodicities or classify payments into distinct periodic cycles.

The standard method for detecting repeating patterns in time series involves calculating a matrix of pairwise distances between all subsequences and performing an all-pair similarity search to identify the closest and most similar patterns. However, storing such a matrix is computationally expensive due to its quadratic complexity. To address this inefficiency, Yeh et al. [8] propose the Matrix Profile, the basis of the current state-of-the-art group of methods for motif discovery that reduces computational complexity by storing the minimum Euclidean distance between each subsequence and its nearest neighbour. Matrix Profile is powerful for identifying repeating patterns and anomalies in time-series data. However, while it can reveal motifs efficiently, it does not offer the functionality to classify payments into periodic patterns or differentiate between multiple overlapping periodicities.

Building on the first paper of the Matrix Profile group of papers, Matrix Profile IV [9] explores the integration of weakly labelled data to predict future outcomes, enhancing the utility of Matrix Profile in predictive modelling. Matrix Profile XX [10] introduces techniques for detecting motifs of varying lengths, addressing the limitation of fixed subsequence lengths in earlier works. The "MERLIN" [11] algorithm, discussed in Matrix Profile XXI, extends Matrix Profile's capabilities to anomaly detection without predefined parameters, making it more adaptive.

One more instance from the Matrix Profile group [12] presents a method for discovering time series motifs using Dynamic Time Warping (DTW) [13]. DTW is a distance measure that allows for the comparison of time series by stretching or shrinking segments to align similar patterns, even when they are "sped up" or "slowed down." This approach can accurately identify recurring motifs in time series data, accounting for non-linear variations and ensuring that even sequences that are not aligned in the traditional sense, due to being warped in time rather than shifted, can be compared effectively. The proposed algorithm, SWAMP, uses the principles of DTW and Matrix Profile techniques to automatically find an optimal trade-off between computational efficiency and motif discovery quality. It can efficiently handle large-scale time series data while ensuring exactness in motif discovery, even when the patterns exhibit time distortions or are not perfectly aligned.

De Paepe et al. [14] focus on time series analysis and extend the Matrix Profile framework with the Radius Profile technique. Their approach includes the Anytime Ostinato algorithm, which efficiently identifies consensus motifs across datasets by estimating the

radius profile for all subsequences. While this method is scalable and accurate in detecting recurring motifs, it is limited in handling quasi-periodic behaviour within a single series, particularly when patterns deviate from strict periodicity.

These studies attempt to make Matrix Profile a holistic, jack-of-all-trades approach in identifying patterns and detecting anomalies in time series data. While these methods succeed in identifying patterns in an unsupervised manner, they lack forming them based on a temporal component, which is the aim of the algorithm proposed in this thesis.

Van der Vlist et al. [15] and Silva et al. [16] extend the application of DTW in time-series analysis by introducing a framework specifically for tracking recurring patterns. This approach effectively identifies patterns that are not perfectly aligned in time, making it particularly useful for datasets with diverse periodicities. However, like SWAMP, it does not classify payments into distinct periodic categories or address overlapping patterns.

However, the Matrix Profile group of papers face several limitations that reduce its effectiveness in complex scenarios. For instance, they struggle to identify consecutive repeating patterns and are highly sensitive to the choice of the subsequence length m, which usually leads to either missed patterns or excessive noise. Additionally, while the Matrix Profile captures pairwise similarities successfully, it cannot handle dynamic patterns where the frequency or shape of the repeating activity changes over time [17].

Maurizio Sluijmers [18] explores methods to detect regularities and repetitive sequences to forecast future bank transactions in historical transaction data from retail banking. The approach uses the transaction amounts first and the transactions' labels to form hypotheses if a payment is part of a pattern. Then, they predict the next payment, if there is one, and match it with the recurring pattern. However, while the approach is effective for rigid, recurring patterns, it struggles with quasi-periodic behaviour and noise, both of which are common in transactional datasets.

Wang et al. [19] address quasi-periodic repetition in time series with their Recurrence Point Signed Area Persistence (RPSAP) algorithm. This method extends existing repetition detection techniques by exploiting time-warping invariance of path signatures. It is particularly effective in handling variable period lengths, making it robust against quasi-periodic patterns. However, the algorithm's computational complexity and parameter sensitivity make it challenging to scale to large datasets.

2.3 Signal Processing Approaches

Signal processing approaches treat time series as signals, transforming them into alternative domains, such as frequency or subspace domains, to uncover hidden periodic components. These methods are particularly valuable when patterns are obscured by noise, irregularities, or overlapping structures.

Muresan and Parks [20] consider time series as signals and decompose them into orthogonal periodic subspaces to detect and estimate multiperiodic signals. Their Exactly Periodic Subspace Decomposition (EPSD) approach excels in noise environments and isolates periodic subsignals effectively. However, this method struggles with irregular or quasi-periodic patterns, which are common in wholesale banking data.

Sethares and Staley [21] treat time series as signals and use Periodicity Transforms (PTs) to project data onto periodic subspaces for rhythmic pattern detection. Unlike Fourier or wavelet transforms, PTs are linear-in-period, allowing for more flexible detection of periodic patterns. However, they fall short in handling quasi-periodic or irregular data, limiting their adaptability in complex datasets.

Deepa and Manju [22] utilize the Ramanujan subspace projections to identify periodicity within signals. By leveraging the Ramanujan sums [23], their method efficiently detects hidden periodic components in signals with mixtures of periodic behaviours. This approach is computationally lightweight and effective in periodicity detection but struggles with irregular patterns and heavily noisy environments.

Deng and Han [24] introduce the Conjugate Subspace Matching Pursuit (CSMP) algorithm, a signal processing approach for periodic decomposition. By representing signals in conjugate subspaces, this method identifies hidden periods with high accuracy. Its ability to decompose signals into periodic components makes it effective for signal approximation and hidden period identification, but it requires exact period alignment, making it less suitable for noisy or quasi-periodic scenarios.

2.4 Theoretic Background

2.4.1 Time Series and Payments

A time series is a sequence of observations recorded over time, typically at regular intervals. Mathematically, we define a time series as an ordered set of values $T = \{t_1, t_2, \ldots, t_n\}$, where n is the length of the time series and t_i is simply a timestamp,

and the corresponding value at each t may represent a transaction, or, generally, any sequential data.

In the context of this thesis, each each element of T represents a transactional event. In ING's wholesale banking, the only consistent information on the transactions of a single client are the amount and the date the transaction occured. Thus, we can also redefine the time series T as a set of payments $P = \{(t_1, A_1), (t_2, A_2), \ldots, (t_n, A_n)\}$, where t_i is the day the payment occurs and A_i is the transaction amount.

An important attribute of a sequential time series is whether it is periodic, ie. certain values (or patterns) repeat at regular or semi regular intervals. In our case, this translates to payments occurring on a periodic basis, such as weekly, monthly, or quarterly. Mathematically, it a pattern can be written as

$$t_i = t_1 + k \cdot T + devD_i, \quad k = 0, 1, \ldots, N-1,$$

where T is the periodicity, meaning the frequency the payments of the pattern occur, and $devD_i$ is a small potential deviation of these payments from the expected date. These shifts can occur for various reasons, such as banking holidays, weekends or simply ING's client and the recipient of the payment have such an agreement.

While payments may follow a perfect periodic structure in terms of frequency, their amounts may vary. This variation can be expressed as

$$A_i = A + devA_i,$$

where A is the expected amount $devA_i$ is the potential deviation from A.

The primary issue in identifying recurring patterns arises from noise. In this context, noise can be one-time payments or transactions that do not conform to any pattern. However, due to the lack of labels and contextual information, noise may also include payments that are unexpectedly delayed, advanced, or exhibit unexpected fluctuations in amount.

This thesis aims to investigate the existence of structured, recurring payment patterns within a time series of transactions. Specifically, we seek to identify a subset of payments that exhibit periodicity in both time and amount while distinguishing these structured transactions from irregular noise.

$$t_i = t_1 + kT, \quad k = 0, 1, \ldots, N-1$$

2.4.2 Fourier Theory

Fourier analysis allows us to decompose a signal into a combination of sinusoidal components, providing a connection between the time and frequency domains. In the frequency domain, periodic behaviours appear as distinct peaks, making them easier to detect compared to the noisy and irregular time domain. In the context of this thesis, we treat a time series as a signal, enabling the use of Fourier methods to analyse patterns and noise.

For a periodic function $f(x)$ with period T, the Fourier series expresses $f(x)$ as an infinite sum of sinusoidal components

$$f(x) = a_0 + \sum_{n=1}^{\infty} \left(a_n \cos\left(\frac{2\pi n x}{T}\right) + b_n \sin\left(\frac{2\pi n x}{T}\right) \right), \tag{2.1}$$

where a_0 represents the average value of the signal, and a_n and b_n denote the amplitudes of the cosine and sine components for each harmonic n, respectively.

The coefficients a_0, a_n, and b_n are calculated as follows. The constant term a_0, which is the mean value of the function over one period, is

$$a_0 = \frac{1}{T} \int_0^T f(x)\, dx.$$

The amplitudes of the cosine and sine components are given by the equations

$$a_n = \frac{2}{T} \int_0^T f(x) \cos\left(\frac{2\pi n x}{T}\right) dx$$

and

$$b_n = \frac{2}{T} \int_0^T f(x) \sin\left(\frac{2\pi n x}{T}\right) dx.$$

These coefficients quantify the contribution of each harmonic frequency to the overall signal, enabling the reconstruction of $f(x)$ as a sum of its frequency components.

When dealing with non-periodic functions, the Fourier series is generalized to the Fourier transform of a function $f(t)$

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t}\, dt,$$

where ω is the angular frequency. It represents a signal as a continuous sum of sinusoidal components. The inverse Fourier transform reconstructs the original time-domain

function

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{i\omega t} \, d\omega.$$

2.4.2.1 Discrete Fourier Transform

The Discrete Fourier Transformation (DFT) allows for the analysis of discrete data, such as time-series signals. For a discrete sequence $x[n]$ with N samples, the DFT is

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-i2\pi kn/N}, \quad k = 0, 1, \ldots, N-1,$$

where $X[k]$ represents the frequency-domain coefficients. The original sequence can be reconstructed using the Inverse Discrete Fourier Transform (IDFT) as

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{i2\pi kn/N}, \quad k = 0, 1, \ldots, N-1.$$

2.4.2.2 Power and Parseval's Theorem

Parseval's theorem connects the time and frequency domains by stating that the total energy or power of a signal in the time domain is equal to the sum of the squares of its frequency components [25]. For a continuous function $f(t)$, this is expressed as

$$\int_{-\infty}^{\infty} |f(t)|^2 \, dt = \int_{-\infty}^{\infty} |F(\omega)|^2 \, d\omega.$$

For discrete signals, this becomes

$$P = \sum_{n=0}^{N-1} |x[n]|^2 = \sum_{k=0}^{N-1} |X[k]|^2,$$

where P represents the total power, and $X[k]$ are the DFT coefficients.

2.4.2.3 Motivation for Fourier Analysis

The motivation for using Fourier methods in this thesis lies in their ability to simplify the analysis of complex time-domain signals by transforming them into the frequency domain. In the time domain, patterns may be masked by noise or irregular instances of transactions. However, in the frequency domain, periodic signals appear as distinct

peaks, making them easier to identify and analyse. By working in the frequency domain, we gain insight into the structure of the data.

Chapter 3

Methods

3.1 Motivation

The analysis of payment behaviours in wholesale banking reveals that transactions often follow specific periodicities. Figure 3.1 shows the histogram of transaction intervals, highlighting dominant periodicities such as weekly (7 days), bi-weekly (14 days), and monthly (28 days). These regularities are common in financial operations, both in wholesale and personal banking, reflecting random or recurring payments.

However, the presence of noise, irregularities, and overlapping periodicities makes it challenging to detect and align such patterns. The histogram motivates the need for an algorithm that can identify and align periodic structures, provided that there is the knowledge of the periodic cycle to be tracked, even in noisy datasets. This chapter introduces a methodology designed to address this challenge by leveraging these observed frequencies to optimise payment patterns.

3.2 Power and Resonant Power

To understand the reasoning behind the objective function that follows in the next sections, we first need to introduce the concept of power in the context of periodic patterns. Power represents the strength of a signal, which in this case corresponds to the distribution of payment amounts across different periods and days. By analysing how power is distributed, we can differentiate between patterns that align with expected periodic behaviours and those that deviate.

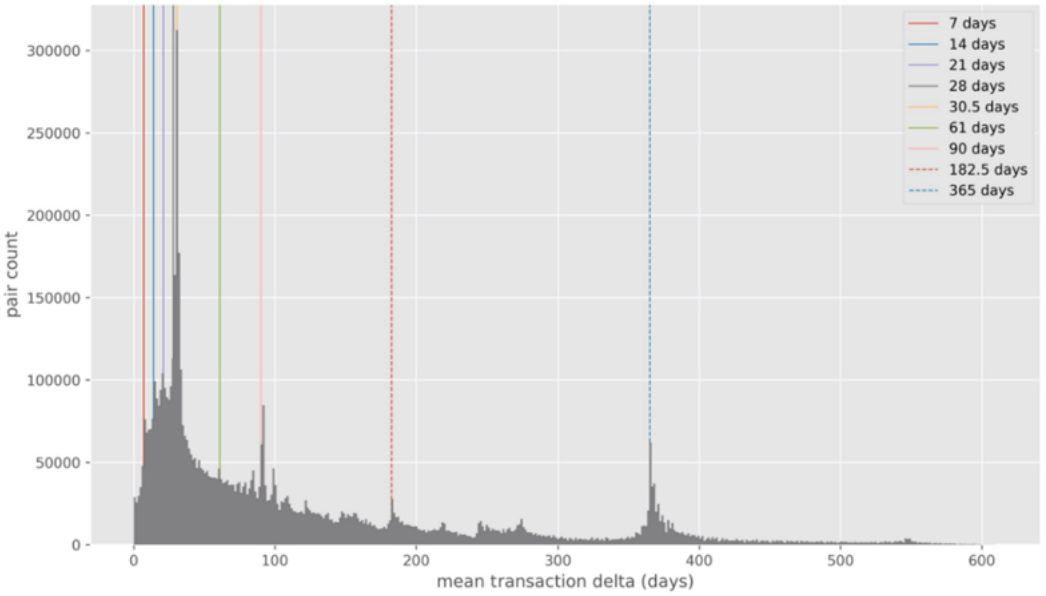

FIGURE 3.1: Distribution of transaction intervals, with dominant periodicities.

3.2.1 Power

In signal processing, Parseval's theorem (2.4.2.2) states that the total power of a signal is equivalent in both the time and frequency domains. For a discrete signal, the power in the frequency domain is expressed as:

$$P = \sum_{k=0}^{N-1} |X[k]|^2, \tag{3.1}$$

where $X[k]$ is the Discrete Fourier Transform (DFT) of the signal $x[n]$, and N represents the total number of samples. Parseval's theorem states that this power is also equal to the sum of the squared magnitudes of the signal values in the time domain

$$P = \sum_{n=0}^{N-1} |x[n]|^2. \tag{3.2}$$

In our case, the signal represents payment amounts distributed across T days in a period and F periods, such that the total number of samples is $N = T \cdot F$. Each time index n can be expressed in terms of its period index i and day index j, as $n = i \cdot T + j$, where $j \in \{0, 1, \ldots, T-1\}$ and $i \in \{0, 1, \ldots, F-1\}$.

The payment amount at time n corresponds to a_{ij}, the payment on day j of period i. Substituting this structure into the equation for power 3.2, we have

$$P = \sum_{n=0}^{N-1} |x[n]|^2 = \sum_{i=0}^{F-1} \sum_{j=0}^{T-1} |a_{ij}|^2. \tag{3.3}$$

Since payments are real-valued, $|a_{ij}|^2 = a_{ij}^2$. To account for normalization and maintain consistency with Parseval's theorem, we divide the power by the total number of samples N. This results in the normalized expression of 3.3,

$$P = \frac{1}{T \cdot F} \sum_{j=0}^{T-1} \sum_{i=0}^{F-1} a_{ij}^2. \tag{3.4}$$

This measure reflects the overall "strength" or magnitude of the payments, capturing both periodic and non-periodic components.

3.2.2 Resonance Power

The central concept introduced in this thesis is resonant power. In a perfectly periodic pattern, payments align consistently with the resonant frequency, representing the fundamental periodic behaviour of the system. Resonant power aims to mathematically represent the recurring component of the signal.

In a perfectly periodic scenario, payments occur consistently on the same relative days across all periods. For example, if a payment is expected every Monday, all Mondays across the timeline would have the same payment value in the absence of noise or deviations. Resonance power captures this aligned, repeating structure in the signal.

Mathematically, resonance power is defined by first averaging the payment amounts for each day j across all periods i. This average isolates the systematic behaviour for each day:

$$\text{Daily average for day } j : \bar{a}_j = \frac{1}{F} \sum_{i=0}^{F-1} a_{ij}, \tag{3.5}$$

where $\bar{a}_j$ represents the average payment on day j across all periods.

To compute the total resonance power, we square these daily averages to account for their contribution to the signal strength and sum over all days j within a single period. Since we normalize the power by the period length T, the resonance power is expressed

as:

$$P_R = \frac{1}{T} \sum_{j=0}^{T-1} \bar{a}_j^2.$$

Substituting the expression for $\bar{a}_j$, the resonance power becomes

$$\begin{aligned} P_R &= \frac{1}{T} \sum_{j=0}^{T-1} \left(\frac{1}{F} \sum_{i=0}^{F-1} a_{ij} \right)^2 \\ &= \frac{1}{T \cdot F^2} \sum_{j=0}^{T-1} \sum_{i=0}^{F-1} a_{ij}^2. \end{aligned} \tag{3.6}$$

Resonance power reflects how much of the total power in the system is due to systematic, periodic behaviour. In an ideal case, where there is no noise or irregularity, payments are perfectly aligned with their expected periodic days, and there is no variation in the daily amounts, thus $\bar{a}_j$ stands $\forall i, j$. Also, in that case, the resonance power P_R equals the total power P, because then $\frac{1}{F} \sum_i a_{ij}^2 = \bar{a}_j^2 = (\frac{1}{F} \sum_i a_{ij})^2$.

However, in real-world data, deviations from the periodic pattern (e.g., noise or irregular payments) reduce P_R, creating a gap between the total power and the resonance power. This gap is what we aim to minimise in our optimisation process. By maximizing P_R, we effectively align payments with their periodic structure, emphasizing recurring patterns and reducing noise.

3.3 Objective Function

To evaluate how well payments align with their periodic patterns, we aim to minimise the deviations from the ideal periodic structure. This is achieved by minimizing the Non-Resonant Power (NRP), which quantifies the irregular, non-periodic components in the signal.

NRP represents the "noise" or irregularities in the data, measuring the payment activity on days other than the recurring periodic pattern. For instance, in a weekly pattern where Monday is the recurring day, NRP measures the power of payments occurring on all other days of the week (Tuesday through Sunday). By minimizing NRP, ideally reducing it to zero, we align payments with their recurring periodic structure, ensuring a consistent pattern. Mathematically, the objective is expressed as

NRP does not only capture payments that occur on incorrect days but also deviations in amounts within the correct periodic structure. For instance, consider a case where

payments are expected every Monday with an amount of 1000. If all Mondays have identical amounts, the resonance power is maximised, and NRP is minimised. However, if one Monday has a significantly higher or lower amount, this introduces non-resonant power,even though the payment still occurs on the correct day.

$$\text{Objective} = NRP = P - P_R. \tag{3.7}$$

3.3.1 Derivation

The derivation of the objective function begins by considering the variance of daily amounts across all periods. For a given day j, the average payment across all periods is

$$\bar{a}_j = \frac{1}{F} \sum_{i=0}^{F-1} a_{ij}, \tag{3.8}$$

where a_{ij} represents the payment amount on day j of period i, and F is the total number of periods.

The deviation of the payment a_{ij} from the daily average $\bar{a}_j$ is

$$a_{ij} - \bar{a}_j = a_{ij} - \frac{1}{F} \sum_{k=0}^{F-1} a_{kj}. \tag{3.9}$$

Squaring this deviation and summing across all periods gives the variance for day j

$$\text{Variance for day } j = \frac{1}{F-1} \sum_{i=0}^{F-1} \left(a_{ij} - \frac{1}{F} \sum_{k=0}^{F-1} a_{kj} \right)^2. \tag{3.10}$$

Note: In equation (3.10), we divide by the term $F-1$ instead of F because it represents the unbiased estimator of variance to correct for the bias introduced by estimating the mean from the same data.

To calculate the total variance across all T days, we sum this expression over all days j and normalize by T, we end up with the expression

$$Var = \frac{1}{T} \sum_{j=0}^{T-1} \frac{1}{F-1} \sum_{i=0}^{F-1} \left(a_{ij} - \frac{1}{F} \sum_{k=0}^{F-1} a_{kj} \right)^2. \tag{3.11}$$

Since variance is measured in squared units, we take the square root to obtain the standard deviation, which is our objective function

$$Obj = \sqrt{\frac{1}{T}\sum_{j=0}^{T-1}\frac{1}{F-1}\sum_{i=0}^{F-1}\left(a_{ij} - \frac{1}{F}\sum_{k=0}^{F-1}a_{kj}\right)^2}. \tag{3.12}$$

Expanding the squared term, substituting and simplifying, we obtain the final equation for the objective

$$Obj = \sqrt{\frac{1}{T}\sum_{j=0}^{T-1}\left(\frac{1}{F-1}\sum_{i=0}^{F-1}a_{ij}^2 - \frac{1}{F(F-1)}\left(\sum_{i=0}^{F-1}a_{ij}\right)^2\right)}. \tag{3.13}$$

The objective function measures the variance of payments across all days, capturing deviations from periodic patterns. It can be broken into two components:

- $\frac{1}{F-1}\sum_{i=0}^{F-1}a_{ij}^2$: This term represents the total power of payments on day j across all periods.
- $\frac{1}{F(F-1)}\left(\sum_{i=0}^{F-1}a_{ij}\right)^2$: This term accounts for the systematic periodic component by subtracting the resonance power.

By minimizing the objective function, we minimise the non-resonant power, reducing noise and ensuring payments are consistent with their expected periodic patterns. This alignment highlights the recurring structure within the data while reducing deviations caused by irregular payments.

3.3.2 Objective in Terms of Individual Payments

The above formulation of the objective is expressed in terms of daily amounts a_{ij}, where a_{ij} is the total amount on day j of period i. However, multiple payments can contribute to a_{ij}, and we need to express the objective function in terms of individual payments. This allows for more granular manipulation of payments during optimisation.

The daily amount a_{ij} can be expressed as the sum of individual payments:

$$a_{ij} = \sum_{p} A_p S_p \delta(I_p = i)\delta(J_p = j), \tag{3.14}$$

where:

- A_p: Amount of payment p,
- S_p: Status of payment p (1 if active, 0 otherwise),
- I_p: Period index of payment p,
- J_p: Day index of payment p,
- $\delta(\cdot)$: Kronecker delta (1 if the condition is true, 0 otherwise).

Substituting a_{ij} into the objective equation 3.12,

$$Obj = \sqrt{\frac{1}{T}\sum_{j=0}^{T-1}\left(\frac{1}{F-1}\sum_{i=0}^{F-1}\left(\sum_{p} A_p S_p \delta(I_p = i)\delta(J_p = j) - \frac{1}{F}\sum_{p} A_p S_p \delta(I_p \in I)\delta(J_p = j)\right)^2\right)} \tag{3.15}$$

Here, we have rewritten a_{ij} as a sum over payments p, where each payment is weighted by its amount A_p, its status S_p, and Kronecker delta functions $\delta(I_p = i)$ and $\delta(J_p = j)$, showing that the payment belongs to period i and occurs on day j. Since the second sum already considers all periods, we have removed the unnecessary $\delta(I_p \in I)$.

Expanding the square term,

$$Obj = \sqrt{\frac{1}{T}\sum_{j=0}^{T-1}\left(\frac{1}{F-1}\sum_{i=0}^{F-1}\left(\sum_{p} A_p S_p \delta(I_p = i)\delta(J_p = j)\right)^2 - \frac{2}{F}\sum_{p,q} A_p S_p A_q S_q \delta(I_p = i)\delta(J_p = j)\delta(J_q = j) + \frac{1}{F^2}\sum_{p,q} A_p S_p A_q S_q \delta(J_p = j)\delta(J_q = j)\right).} \tag{3.16}$$

In this step, we expanded the squared difference into three terms. The first term represents the sum of squared payments within a single period, the second term accounts for interactions between payments across different periods, and the third term represents the squared sum over all periods.

Now, summing over all periods, we get:

$$Obj = \sqrt{\begin{aligned} &= \frac{1}{T}\frac{1}{F-1}\left(\sum_{p,q} A_p S_p A_q S_q \delta(J_p = J_q)\right. \\ &\left. -\frac{1}{F}\sum_{p,q} A_p S_p A_q S_q \delta(J_p = J_q)\right). \end{aligned}} \tag{3.17}$$

The granular objective function can be explained as follows. The first term accounts for payments that fall on the same day across different periods, while the second term represents the squared sum over all periods, normalized by F.

Now that we have explained how the objective function works, both in simple terms and in detail, we can see how each payment contributes to the calculation. This understanding helps us identify the payments that have the most impact on the objective. By focusing on these payments, we can adjust their effect by moving them or turning them off (S_p), making it easier to align payments with their ideal periodic patterns and retain only the ones that belong to the periodic structure we are searching for.

3.3.3 Objective Pseudocode

As mentioned before, the objective function aims to compute the standard deviation of payment amounts over periods, ensuring that payments align with an expected recurring pattern. To achieve this, the algorithm first filters out payments outside the specified start and end days, based on the start day of the period, the anchor day. For each valid payment, we calculate the indices that correspond to its period and day within the overall timeline. These indices are then used to construct a matrix mtx, where each entry aggregates the total payment amounts occurring on the same day within a period. Finally, the algorithm calculates the variance of these amounts across periods and then computes its square root to obtain the standard deviation. The goal is to minimise this standard deviation, thereby aligning payments to a consistent pattern.

3.4 Periodicity Prioritization

To ensure accurate classification of periodic patterns, we prioritize processing the lowest periodicities first. For instance, consider payments of every Monday on a weekly basis and payments of every other Wednesday on a biweekly basis. If we start by analysing biweekly periodicities, both the weekly Monday payments and the biweekly Wednesday payments may be incorrectly grouped as biweekly, causing a loss of information. By resolving the weekly pattern first, we isolate these finer-grained patterns and prevent

Algorithm 1: Objective Function Calculation

Input: Period length T, Payments p, Amounts a, Status s, Start day *start_day*, End day *end_day*, Anchor day *anchor_day*

Output: Objective value of the timeline

Compute total number of periods F in the timeline:

$$F = \frac{(end\_day - 1 - anchor\_day)}{T} + 1 - \frac{(start\_day - anchor\_day)}{T}$$

foreach *payment* p_i *in* p **do**
  if p_i *occurs outside start_day and end_day* **then**
    Remove p_i from consideration;

For each valid payment p_i, find the period and day indices.

Construct matrix mtx of shape (F, T) to aggregate payment amounts.

$$mtx[i, j] = \sum a_{ij} \quad \text{for all payments on day } j \text{ of period } i$$

Calculate the variance and square root it to get the objective.

$$Objective = \sqrt{\frac{1}{T \cdot (F-1)} \left(\sum_{i=0}^{F-1} \sum_{j=0}^{T-1} a_{ij}^2 - \frac{1}{F} \sum_{j=0}^{T-1} \left(\sum_{i=0}^{F-1} a_{ij} \right)^2 \right)}.$$

return Objective value

misclassification when moving to higher periodicities. Once payments are classified as part of a lower periodicity, such as weekly, they are removed from the data to ensure they are not incorrectly classified again under higher periodicities. This step is critical to preserving the integrity of the detected patterns and avoiding overlap between periodicities.

3.5 Payment Manipulation Operations

The operations used in this optimisation approach are based on a greedy algorithm, meaning that at each step, the algorithm selects the locally optimal solution in hopes of finding a global optimum. This strategy works by making small, incremental changes to the payment schedule, always accepting the change that improves the objective function the most at the current step. As a result, a "sophisticated" combination of operations, even if they might be logical, will not be favoured if there is any operation that, at that step, can be more effective in reducing the objective.

In the context of optimising a set of payments over time, it is crucial to have a set of well-defined, controlled operations that allow for the adjustment of individual payments while attempting to minimise an objective function. The operations considered in this work

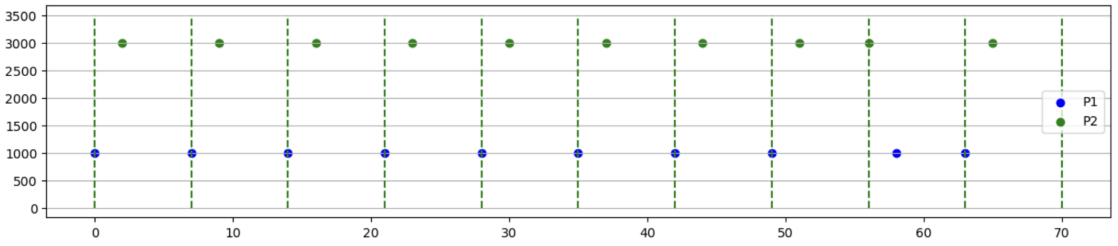

FIGURE 3.2: Swap Scenario

are designed to either adjust individual payments through moving, eliminate payments through removing, swap payments between days, or, finally, a combination of swap and remove in one step. However, all operations but the removal must respect a critical constraint: each payment can only be moved or swapped within a pre-specified range of days from its original position. This limitation ensures that adjustments maintain the temporal consistency of the payments and prevent drastic shifts that could distort the original pattern. By enforcing these constraints, the optimisation remains controlled and reflective of realistic scenarios where payments are expected to occur within a defined range, ensuring the integrity of the resulting schedule.

To optimise the objective function and align payments with their expected periodic patterns, four key operations are employed: Move, Remove, Swap, and SwapRemove. These operations iteratively adjust the payment schedule, recalculating the objective function to identify improvements at each step.

The Move operation shifts payments to a nearby day within a predefined range. This adjustment is useful for payments that are slightly misaligned with the expected pattern. For each payment, potential new days are evaluated by temporarily modifying the payment schedule, computing the objective for each case, and selecting the day that minimises the objective.

The Remove operation evaluates whether eliminating a payment improves the objective function. This is achieved by iterating over payments, temporarily deactivating each one, and calculating the resulting objective. Payments that contribute noise or do not meaningfully align with the pattern are removed if this action reduces the objective.

A Swap operation is introduced because two consecutive move operations may not always work if the individual moves do not improve the objective function. In some cases, the combined effect of two moves is necessary to reduce variance, but since each individual step does not improve the objective on its own, the algorithm will not execute them separately.

Consider a simple scenario where only two regular weekly patterns, P_1 and P_2, occur, as in Figure 3.2. In this scenario, the payments of the two patterns occur reversely.

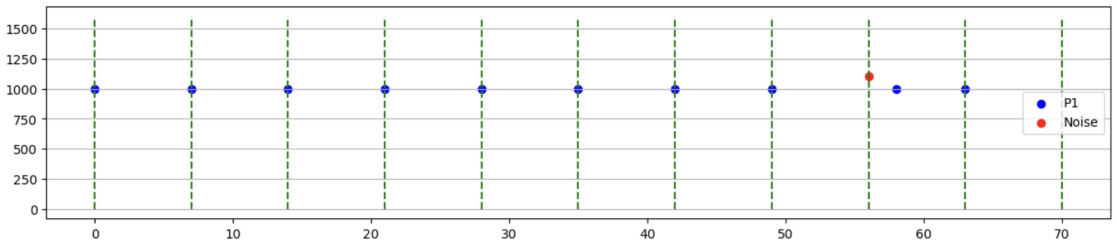

FIGURE 3.3: SwapRemove Scenario

In the ideal case, where swap has not been introduced yet, the rational steps would be to move one by one to the correct location. However, each of these operations, when performed independently, would initially worsen the objective function. To address this issue, we introduce the swap operation. The pattern payments will swap, minimising the objective in one step.

In certain scenarios, the operations above are insufficient to deal with misalignments. The SwapRemove operation is introduced to simultaneously swap two payments and remove the hopefully unwanted payment that minimises the objective the most, in a single step.

Figure 3.3 shows a noise payment, marked with red, and a sequence of payments happening in regular intervals, marked with blue. However, we see that one pattern payment is shifted to day 58, while the noise payment occurs on the expected day of the pattern, day 56. Swapping these two payments would minimise the variance on the expected pattern day, day 0. However, it would raise the variance even further on day 2, which would worsen the objective. Removing one payment would have the same outcome. As a result, we introduce SwapRemove, will swap the two payments and remove the noisy one in one step. This would minimise the objective the most (effectively the objective value would be 0, since there will be no deviations in neither the day nor the amount).

This begs the question of the limit of combinations of operations, since we saw that they offer flexibility and, simply, one extra tool to solve this complex task. However, extending to combinations of three or more actions introduces the significant challenge of the computational cost, as it increases exponentially as more combinations are evaluated. At each step, the algorithm assesses the impact of various possible actions on a given transaction, and with each additional action, the number of possible outcomes multiplies. Lastly, as the number of combined actions grows, the interpretability of the optimisation process worsens, making it difficult to justify or understand the sequence of operations applied. Therefore, the limit is set by a trade-off between computational feasibility, the complexity of implementation, and the practical need for interpretability and effectiveness in real-world scenarios.

3.6 Windowed Optimisation

The optimisation process we employ is designed to adjust payments over a timeline in such a way that patterns and consistency are improved, with a focus on minimizing variance and aligning payments with expected periodicities. The algorithm works by greedily manipulating the payment schedule using the operations introduced in the last section, but the optimisation is performed in windows rather than across the entire timeline at once. This windowed optimisation strategy brings several advantages in terms of improved outcomes when dealing with noisy data and partial patterns.

3.6.1 Overview of the Windowed Optimisation Process

The optimisation function operates by adjusting the payments in a stepwise manner, where the timeline is divided into smaller windows, expanding at each step until the window size reaches the timeline length. In each window, only a subset of the payments is considered for optimisation, and after each window is processed, the global objective is recalculated. The overall process is designed as follows:

1. **Initial Move and Swap Only on the Whole Timeline.** The algorithm starts by performing a limited optimisation over the entire timeline, using only the move and swap operations. This first run is intended to adjust payments into better positions without prematurely removing any payments. Its purpose is to restructure the payment timeline to reposition payments as close as possible to their expected final locations before finer adjustments are made.

2. **Windowed optimisation with All Operations.** After the initial optimisation, the windowing process starts, where all operations (move, swap, remove, swapremove) are allowed. The first window begins at the start day and extends up to the defined window size. The optimisation is carried out within the current window only, allowing early identification and refinement of recurring payment patterns. When there are no more possible optimisations in the window, the algorithm calculates the objective value, the power and other metrics from the starting day up to the end day of the specific window.

   Note that, if a payment is turned off at a point, it can potentially turn on again in a future window.

3. **Window Expansion.** The window expands along the timeline by a step size, overlapping with the previous window, while always maintaining the starting day at 0. The process is repeated until the end of the timeline.

In the windowed optimisation process, several design choices had to be considered. One of the simplest yet most impactful decisions was whether to use a sliding window or an expanding window for the better and clearer optimisation of the payments' locations.

In the sliding window approach, the optimisation would focus on a segment of the timeline, moving forward by the step size. While computationally efficient, this method has a significant drawback: it creates an ideal scenario in the first window and then attempts to replicate it in subsequent windows. This approach risks missing patterns that are either imperfect or absent in the initial window, as the algorithm becomes constrained by the decisions made early in the process. A partial solution to that would be the next window to overlap with the former one. However, when we tried to find partial patterns, or the quality of a pattern was low in the beginning, the sliding window did not perform well under any setting.

To address this limitation, we opted for the expanding window approach. Here, the optimisation considers the entire timeline up to the current window's end at each step, keeping the starting point the same. This allows the algorithm to reconsider earlier decisions and adapt to new information as the window expands. While computationally more intensive, the expanding window method provides a more robust and adaptive solution for optimising payment schedules.

3.6.2 Algorithm Inputs and Outputs

The windowed optimisation algorithm takes several key inputs that guide the optimisation process. The primary input is the data dictionary, which contains a list of payment patterns, each with attributes such as the payment days, amounts, and associated IDs. In addition to these patterns, the algorithm accepts parameters such as the period length, start and end days for the optimisation, and the anchor day that serves as a reference point for the periodicity. Other inputs include a status array that indicates whether each payment is active or not, and an array, `original_new_days`, which defines the allowed range for moving payments. Here we opted for a small number of allowed deviations as the realistic problem we aimed to address was delays due to weekends and days banks were not operating fully.

In this thesis, we evaluate the algorithm using simulated datasets, where we generate structured payment patterns alongside additional noise.The use of synthetic data allows us to create controlled scenarios where the true periodic payments are known in advance, enabling us to assess the algorithm's ability to extract them. However, the challenge remains that noisy payments are often similar in amount to the pattern payments,

and pattern payments themselves may exhibit slight deviations in timing and amounts, increasing the difficulty of distinguishing between the two.

Thus, one of the essential aspects to note, which will be apparent in the experiment section, is that the algorithm does not distinguish between pattern payments and noise. The payments and noise are initialised together in the input dictionary, and the algorithm treats all payments equally during the optimisation. As a result, without any filtering or post-processing, the algorithm may move, swap, or remove any payment, even if it belongs to a noise pattern. This behaviour highlights the importance of pre-processing steps, such as filtering low-amount payments, to ensure that noise does not affect the optimisation outcome.

The output of the algorithm consists of the optimised payment schedule, including the days, amounts, and status of each payment. Additionally, the algorithm returns several diagnostic metrics, such as the objective values for each window, global objective values, power calculations, and the list of payments within each window. These outputs allow for evaluating the optimisation's effectiveness and monitoring how the objective and power change as the optimisation progresses.

3.6.3 Filtering

A key challenge in the optimisation process is that the algorithm tends to stack small payments on top of larger ones to get closer to the expected average amount for a given day. This happens because the objective function sums all payments on the same day, and the optimisation prioritises minimising deviations from the expected pattern amount. Thus, small payments, which are ubiquitous, tend to be retained as they cancel out the deviations of the pattern payments around the expected average.

Suppose a scenario where the expected pattern amount A is X with a variance of Y. As mentioned in chapter 3.6.1, the first step of the algorithm is to minimise the objective by moving or swapping payments. This often leads to payments being stacked on the day of the pattern's occurrence. In such cases, if there is a payment p_1 with amount $a_1 = (X - Y)$, and another payment p_2 with amount a_2 in the range of $(0, Y)$, the algorithm will consider the sum of the two as a better combination of payments for the pattern. This is because the sum $a_1 + a_2$ is closer to the expected pattern amount A, even though p_1 alone might be a more valid part of the pattern.

Due to this issue, we implemented a filtering mechanism to exclude payments falling within the lowest 5^{th} percentile of amounts. This filtering ensures that payments with extreme, low amounts -likely to be noise- are removed before the optimisation. By

filtering out these low amounts, we prevent them from being incorrectly combined with valid payments during the move or swap phases, which ultimately improves the accuracy of pattern detection and ensures that the results are not skewed by insignificant noise.

3.6.4 Advantages of Windowed optimisation

Running the optimisation in windows offers several advantages over performing a full optimisation on the entire timeline at once:

Firstly, one of the key challenges in this optimisation process is dealing with noise in the timeline. If the remove or swapremove operation were allowed from the beginning, especially in the presence of noise, the optimisation could be overly aggressive and remove too many payments, including valid ones that contribute to the overall pattern. Running the optimisation (without remove and swapremove) in windows allows the algorithm to detect patterns more effectively and avoid excessive deletions.

Secondly, optimising the entire timeline in one run can be less effective due to the complexity of interactions between payments over a long period. Windowed optimisation reduces the computational cost by breaking the problem into smaller, more manageable parts. This leads to faster convergence.

Chapter 4

Experiments and results

In this chapter, we present a series of experiments designed to evaluate the proposed algorithm across various scenarios. These experiments assessed how well the algorithm identifies patterns, manages noise, and optimises the objective function in different configurations of patterns and payments. Starting from simple cases with minimal variations and noise, we progressively move to more complex scenarios with multiple patterns, higher levels of noise, and broader payment variations, such as patterns of various periodicities.

4.1 Experimental Setup

Each experiment consists of at least one pattern hidden in a noisy environment. The goal is to optimise payment configurations to align with the underlying patterns while removing or minimizing the influence of noise. The algorithm was evaluated under different configurations, such as variations in the number of patterns, the size of the payment deviations, the noise percentage, and the allowed operations (move, swap, remove, and swap-remove).

To generate the synthetic data, we define a function that creates regular payment patterns based on key parameters such as periodic intervals, start days, and noise levels. The data generation process uses the known periodic intervals presented in 3.1 to simulate realistic payment behaviour while incorporating randomness to represent variability.

In the next sections, we will present a different scenario, each time more complex than the one before. The goal of these experiments is to test the ability of the algorithm to succeed in realistic scenarios, but also to show its weak points. Each section starts by explaining what we test and what is expected of the algorithm. Then, we set the input

settings for each experiment. Then, we present a set of plots that illustrate the payment patterns after the optimisation, a plot that shows the alignment of the payments after the optimisation, and the performance of the algorithm in terms of resonant power. The key parameters for each experiment are:

- Periodicity: The frequency of payments
- Start_Day: The initial day of the payment cycle.
- dDay: Maximum allowable deviation in days from the expected periodicity.
- Amounts: Base amount of payments in the pattern.
- DAmount: Variability in payment amounts.
- pi: It determines how many days within the total timeline will contain noise payments. If pi ¡ 1, Each day of the timeline has an independent probability pi to contain a noisy payment. If pi $\geq$ 1, exactly pi number of random days are selected to contain a noisy payment.
- End_Day: Length of the timeline, set to 364 days unless specified otherwise.

The results are discussed, focusing on how well the algorithm identified the underlying patterns and maximised the resonant power.

4.1.1 Single pattern with low noise

The first test we will present is a basic one. Assume one pattern hidden in low noise. The pattern in this case will not have any variation in the amount and will have slight variation in day, as presented in the table below. The timeline is restricted to 70 days (10 periods) to clearly showcase the algorithm's behaviour.

Pattern 1				
Periodicity	Start Day	dDay	Amount	dAmount
Weekly	0	1	1000	0
Noise				
Type	Scale	pi	Noise Length	Data Length
Exponential	3000	0.1	7	17

TABLE 4.1: Experiment 1 Configuration. A single weekly pattern with low noise and no variation in payment amounts. Noise and data length can be read as how many noisy payments are in the dataset. In this case 7 out of the 17 payments are noisy.

In Figure 4.1, the top plot illustrates the resulting payment schedule (blue), while the bottom plot shows removed noise (red). Notice in the first period that there is also

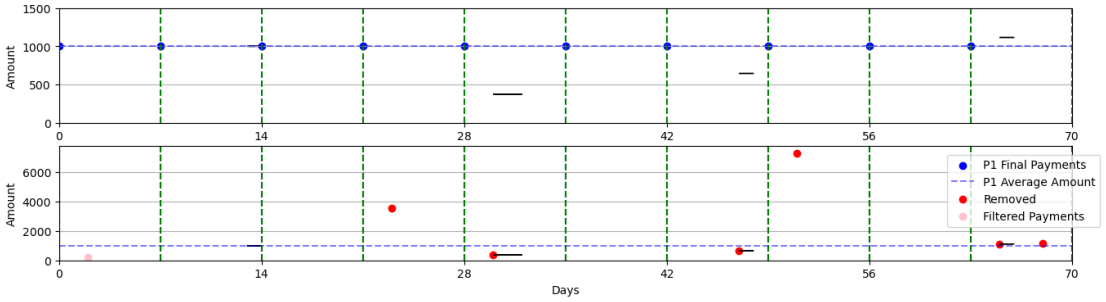

FIGURE 4.1: Low noise has been filtered out resulting in a clear pattern.

a payment in light red, which was filtered by the percentile filter discussed in section 3.6.3. It is important to note that the vertical axes of the top and bottom plots, both in this case and the future ones, are scaled differently: the top plot reflects the amounts of the highest active payments, while the bottom plot includes all payments, including outliers. As a result, a payment with a much higher amount than the active ones may appear prominently in the bottom plot but remain absent from the top plot, where only active payments are displayed.

The vertical green dashed lines indicate the boundaries between periods. Notably, since the recurring payments are scheduled on day 0 of each period, they align precisely with these boundaries. The horizontal black lines indicate the movement of the payments from their original positions in search of the optimal position to maximise the objective. As illustrated, several noisy payments were moved during the phase when only moves and swaps are allowed to explore their alignment with the recurring pattern. However, in the second phase, when removal and swap-remove operations were allowed, these payments were ultimately deemed harmful to the objective and removed. An interesting observation is the active payment on day 14, which was adjusted to better align with the expected pattern, as it can be seen in the next plot.

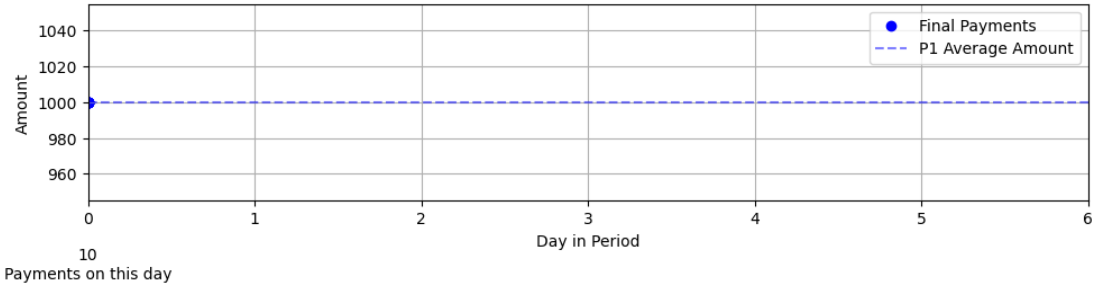

FIGURE 4.2: Given that there are 10 weeks within a 70-day timeline, the pattern correctly retains 10 payments, matching the expected count. We can confirm from this plot that all the payments are aligned on day 0.

From a power perspective, power P is calculated based on the final active payments using the equation defined earlier. Since there are active payments only on one day

(day 0) within each period, this day alone contributes to the power equation. Similarly, the resonance power P_R is calculated only for this recurring day, as the sum a_{ij} in this case is simply the sum of the single payment in each period. As a result, P and P_R are expected to be equal, leading to an objective value of $P - P_R = 0$.

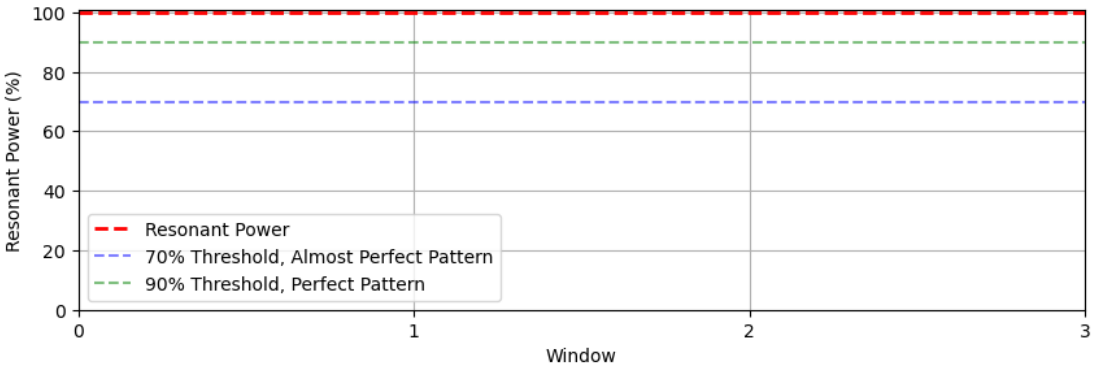

FIGURE 4.3: The resonance power is calculated in the second phase of the windowed optimisation, after the best possible objective has been achieved by manipulating the payments of the window. The x-axis of this plot shows the iteration of the windowed optimisation.

In this plot, we can see that the resonant power is set to 100%, throughout the windowed optimisation. This is only logical, since all the active payments are of the same amount and occur on the same day without any variations in the amount. In subsequent experiments, we will observe how this behaviour may change when additional complexity is introduced, revealing differences in pattern alignment and optimisation outcomes.

4.1.2 Single pattern with low noise and variation in amount

Now in this second example we will see only a slight change in the settings of the experiment. We will introduce a slight variation in the settings by adding deviations in the payment amounts. Something that we should expect to change though, is the resonant power plot. Since this pattern is the only one expected to remain active after optimisation, it will solely influence the resonance power. However, even a 10% variation in the payment amounts is significant enough to be reflected in the results.

Pattern 1				
Periodicity	Start Day	dDay	Amount	dAmount
Weekly	0	1	1000	100
Noise				
Type	Scale	pi	Noise Length	Data Length
Exponential	3000	0.1	7	17

TABLE 4.2: Experiment 2 Configuration. Introducing variation in payment amounts with low noise

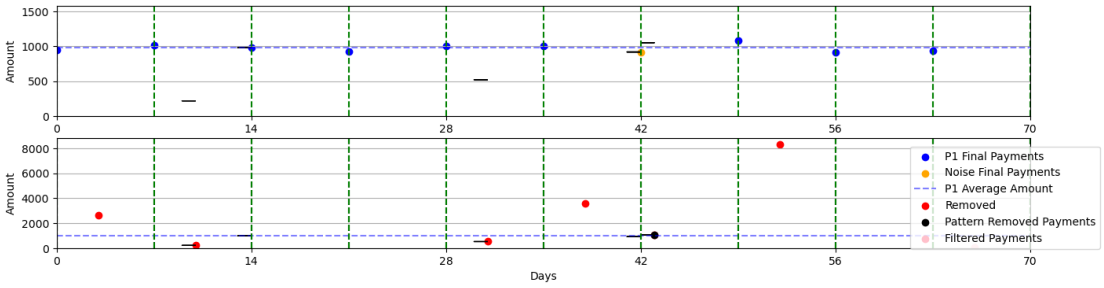

FIGURE 4.4: The figure shows the pattern P_1 in blue with a noisy payment (orange) preferred over the expected pattern payment, which is removed (black in the bottom plot).

As shown in Figure 4.4, most outcomes align with expectations. However, on day 42 of the final payment schedule, a noisy payment is retained, while the expected pattern payment is removed. This behaviour is not an error; rather, it reflects the algorithm's design. As discussed in 3.6.2, the algorithm evaluates payments based solely on their contribution to the objective function, without prior knowledge of which payments should remain active. Thus, it prioritizes payments that maximize the objective, even if they were designed to be a noisy payment.

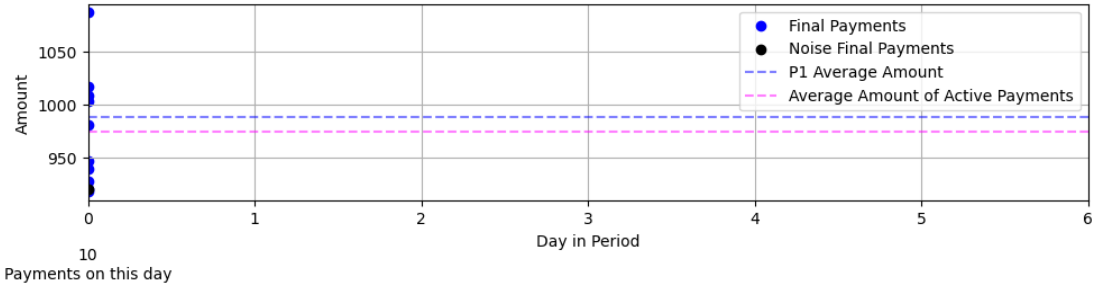

FIGURE 4.5: Once again, the payments aligned on the day they were supposed to. Also, due to the variation in the amount we generated the pattern data, we can see that the average amount the pattern was generated with is lower than 1000. But, this swap of the two payments has affected the average of the final payments' schedule.

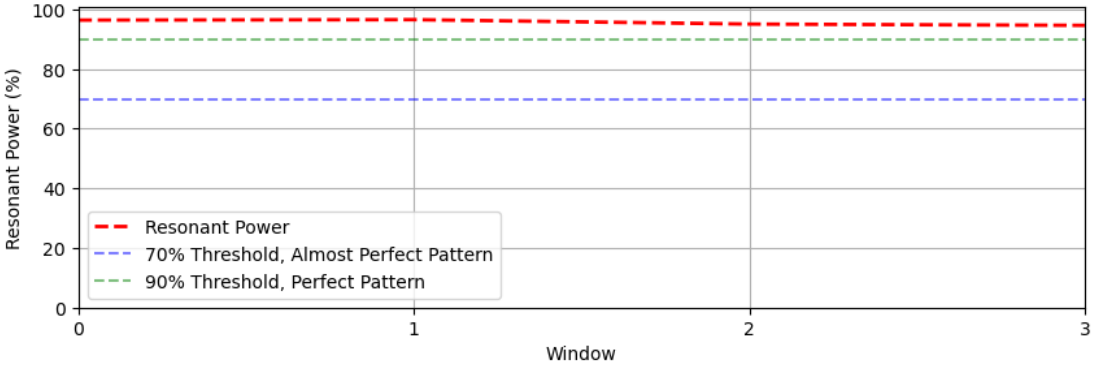

FIGURE 4.6: The resonant power is indeed affected by the variation in the amounts.

As seen in Figure 4.6, the variation in payment amounts impacts the resonance power. While still high, the resonance power is not reaching the 100% level as before due to payments farther from the average. This demonstrates the algorithm's ability to maintain alignment despite moderate variations but highlights its sensitivity to deviations that influence the power metric. Subsequent experiments will explore how the algorithm performs under increasing noise and complexity.

4.1.3 Single pattern with higher noise and variation

Now, we aim for a realistic scenario. Most businesses have periodic expenses, that have some variation on either the occurrence or the amount. The intent is to evaluate how the algorithm handles these deviations in amounts while maintaining alignment with the pattern. The resonance power plot is expected to reflect this variability, as the presence of a 10% variation in the payment amounts is not negligible and will have an impact on the alignment metric.

Pattern 1				
Periodicity	Start Day	dDay	Amount	dAmount
Weekly	0	2	1000	100
Noise				
Type	Scale	pi	Noise Length	Data Length
Exponential	3000	0.3	102	154

TABLE 4.3: Experiment 3 Configuration: A single weekly pattern with low noise and 10% variation in payment amounts.

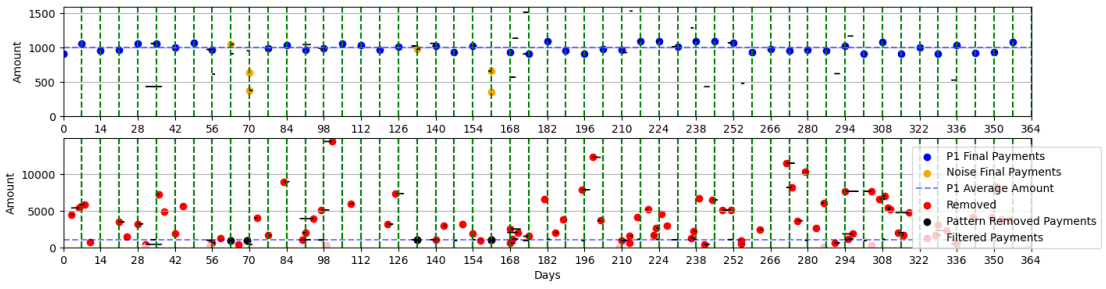

FIGURE 4.7: This time the algorithm considers, other than the two one-to-one swaps, two sum of two payments to be more beneficial to the objective.

In this case, there are two things to notice. One, the algorithm prefers noisy payments to one-to-one replace the ones generated to be part of the pattern P_1. However, as we discussed in the last section, this is expected and the algorithm is designed to behave this way. The new piece of information we gain is that the combination of two or more payments can be more beneficial to the algorithm than a single payment.

We observe that, on days 70 and 161, two noisy payments are retained instead of the single payment designed to be part of the pattern. The combined amounts of these pairs are closer to the expected value of 1000, which explains the algorithm's decision. However, this behaviour highlights a key limitation of the algorithm: it can prioritize combinations of noisy payments over the intended pattern payment if they contribute more favourably to the objective function.

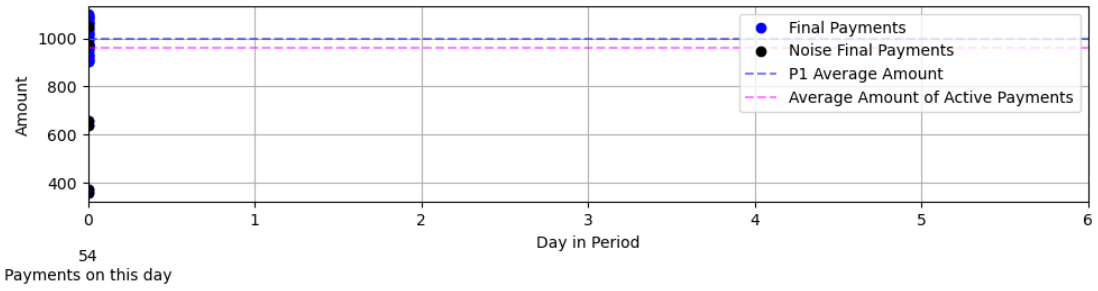

FIGURE 4.8: As expected, the payments align on the day they are supposed to. We can see a cluster of payments around the expected amount of 1000, but the 4 outliers were misclassified as part of the pattern.

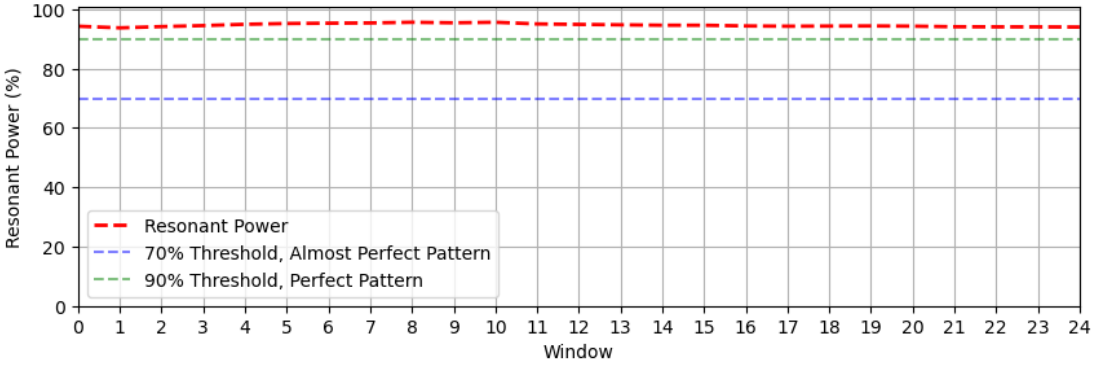

FIGURE 4.9: In this plot, even if there is a misclassification, the resonance power remains high.

In Figure 4.9, we can see that the resonance power is high enough to strongly suggest a perfect pattern. In the discussion section, we will address this issue, as it arises from the inherent property of the algorithm to calculate the sum of payments per day.

4.1.4 Two patterns with the same periodicity in a noisy scenario

In this experiment, we introduce a second pattern alongside the first, both sharing the same periodic behaviour. The second pattern has a higher average amount and occurs on different days than the first pattern. The goal is to observe how the addition of a second pattern impacts the optimisation and how the algorithm manages to align the payments for both patterns while eliminating noise and inconsistencies.

Pattern 1				
Periodicity	Start Day	dDay	Amount	dAmount
Weekly	0	2	1000	100
Pattern 2				
Periodicity	Start Day	dDay	Amount	dAmount
Weekly	4	2	3000	100
Noise				
Type	Scale	pi	Noise Length	Data Length
Exponential	3000	0.3	130	234

TABLE 4.4: Experiment 4 Configuration. Introducing two patterns with the same periodicity but different average values to evaluate the algorithm's performance.

In this configuration, both patterns have a weekly periodicity but differ in their average amounts and start days. P_1 has an average amount of 1000 and starts on day 0, while P_2 has an average amount of 3000 and starts on day 4. Noise is added to the timeline to test the robustness of the algorithm.

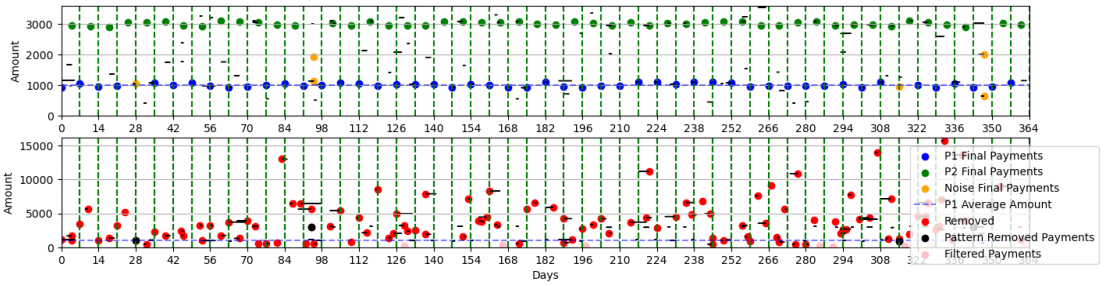

FIGURE 4.10: The figure displays patterns P_1 (blue) and P_2 (green). Some noisy payments (orange) are retained, and expected payments are removed (black in the bottom plot).

Figure 4.10 shows the results of the optimisation process. While the algorithm successfully identifies and aligns many of the payments belonging to P_1 and P_2, it also retains several noisy payments (orange) and removes expected pattern payments (black in the bottom plot). This misclassification primarily occurs when noisy payments better satisfy the objective function than the intended pattern payments.

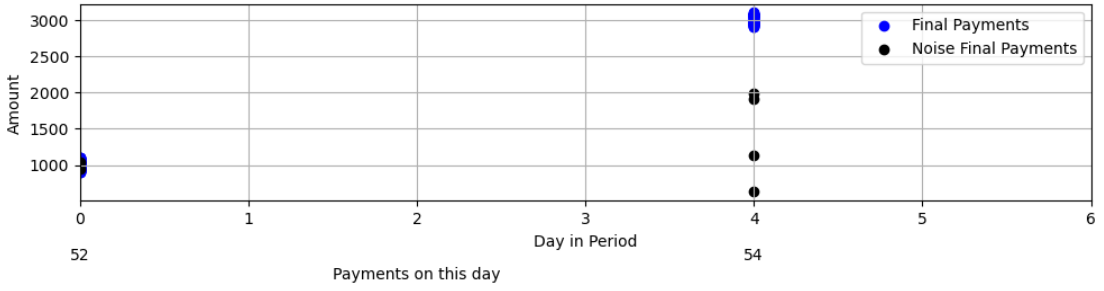

FIGURE 4.11: As expected from the figure above, the lower pattern is represented as a cluster on day 0. The orange pairs of payments are now the black points on day 4.

Figure 4.11 displays the alignment of payments within the weekly periodicity. The blue dots represent the final active payments aligned with the recurring pattern, while the black dots indicate noisy payments that were not removed during the optimisation process. In this case, multiple noisy payments remain active on day 4, contributing to a misalignment. This highlights again the limitation in the optimisation where one or more payments can replace the pattern payment.

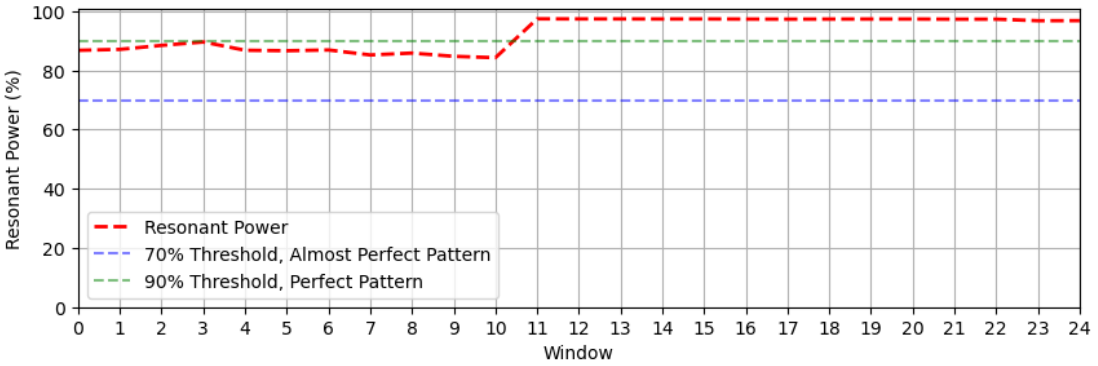

FIGURE 4.12: In this figure, the resonance power is not as high in the beginning as in the earlier cases. It occurs because the optimisation starts with noisy and misaligned payments that negatively affect the alignment of patterns.

In Figure 4.12, the resonance power is lower than expected. This can be attributed to a few different reasons. To start with, due to the strong presence of noise, in the first few windows noise payments that do not belong to any periodic structure may still be present. Furthermore, it can also be caused from the initial step of the windowed process. The first step of the process is to move or swap payments to find a possible optimal scenario. Essentially, this means that the algorithm initially stacks payments on certain days to minimise objective variance. As a result, in the early iterations of the windowed optimisation, the algorithm has not distributed the payments the way we see in the final output in Figure 4.10. Finally, a simple explanation is that the lower the already optimised windows, the less the information available and, thus, the more difficult it is for the algorithm to identify clear patterns.

However, the algorithm gradually aligns payments to their respective patterns, filtering out noise and improving the alignment of recurring payments. This process increases the resonance power as the payments are optimised to better match the expected periodic behaviour of the patterns. The steep increase seen after several windows reflects the transition from a noisy, misaligned state to a well-aligned, optimised payment structure.

4.1.5 Two patterns with different periodicities

In this experiment, we aim to test the algorithm's ability to handle two patterns with differing periodicities, a weekly P_1 and a monthly one P_2. The primary goal is to ensure that the algorithm can distinguish between the two periodicities, correctly align payments, and filter out noise, even when the patterns overlap.

In our approach, we prioritize discovering patterns with the lowest periodicity before considering higher ones. This ensures we correctly identify the underlying structure of the data.

Consider a scenario without noise, involving two payment patterns: a weekly pattern P_1 and a biweekly pattern P_2, both with similar amounts. If we begin by testing the higher periodicity (14 days), every payment would appear to align with a biweekly pattern. For example:

- The 1st weekly payment aligns with the 3rd, 5th, 7th, etc.
- The 2nd weekly payment aligns with the 4th, 6th, etc.
- The actual P_2 payments align perfectly with their biweekly intervals.

Since every weekly pattern is inherently also a biweekly pattern with two equal payments in each period, this overlap would misclassify all payments, including the weekly ones, as part of the biweekly pattern. As a result, the true weekly pattern would be lost. By starting with the lowest periodicity, we avoid this misclassification and preserve the structure of shorter patterns.

4.1.5.1 Sequential Approach

By first aligning payments according to the lowest periodicity, such as weekly payments in this case, we ensure that those payments are correctly categorized and grouped. Once the weekly pattern P_1 is identified and separated, the algorithm can then move on to the higher periodicity (biweekly in this case) and align those payments appropriately. This sequential approach prevents the misclassification of patterns with shorter periodicities, ensuring that each periodicity is properly identified and categorized.

Thus, smaller periodic cycles are first optimised and removed from consideration for the longer ones. Hence, payments belonging to P_1 are correctly classified as weekly, and only the remaining payments are considered for the biweekly pattern P_2. This ensures that each pattern is appropriately represented and aligned within its expected periodicity.

Pattern 1				
Periodicity	Start Day	dDay	Amount	dAmount
Weekly	0	2	1000	100
Pattern 2				
Periodicity	Start Day	dDay	Amount	dAmount
Monthly	4	2	3000	100
Noise				
Type	Scale	pi	Noise Length	Data Length
Exponential	3000	0.3	110	175

TABLE 4.5: Experiment 5 Configuration. After introducing two patterns with two different periodicities, we are testing whether the algorithm can identify the payment structure of two different periodicities.

This experiment aims to test the algorithm's performance when dealing with two patterns of differing periodicities. Specifically, we have one weekly pattern P_1 aligning with day 0 and, for clarity, one monthly pattern P_2 aligning on day 4. The goal is to observe how well the algorithm aligns the payments of each periodicity and filters out noise under these more complex conditions.

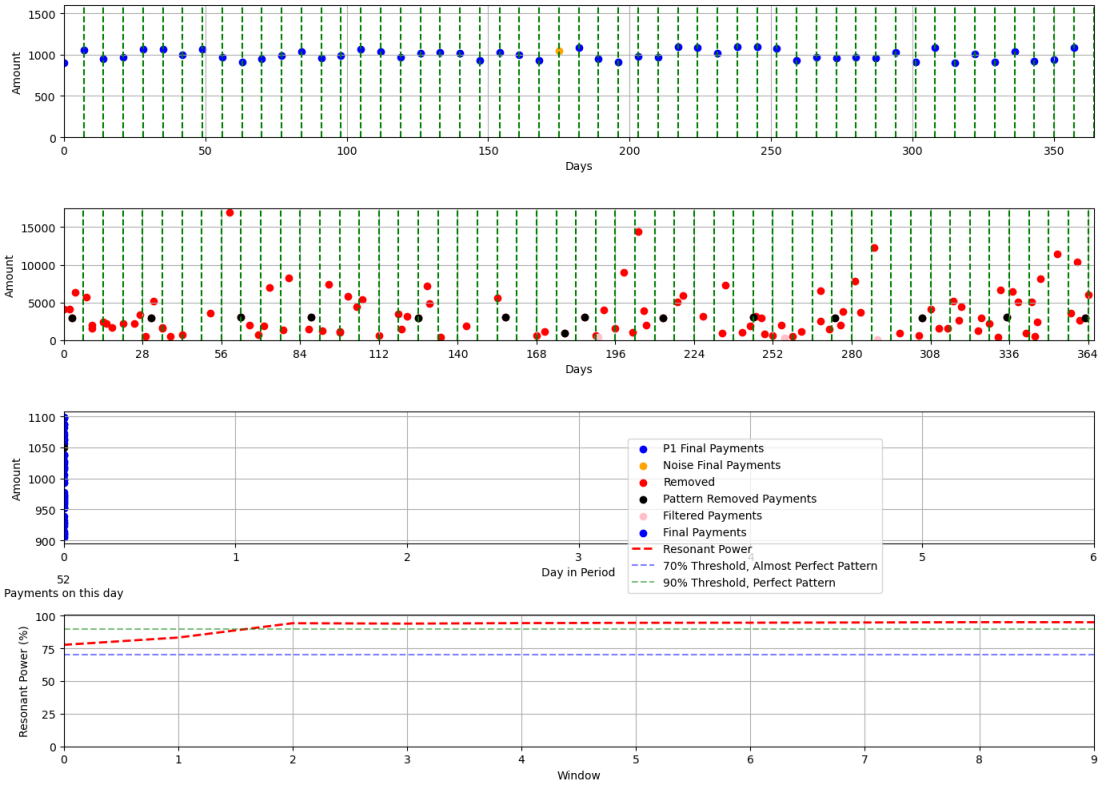

FIGURE 4.13: The experiment begins by detecting a weekly pattern P_1 (blue) amidst noise and a monthly pattern P_2, which in this case should be considered noise. The weekly payments are optimised first, removing noise and aligning the data. Following this, P_1 is subtracted from the dataset, and the algorithm is rerun to identify the monthly pattern P_2 (green). Resonance power confirms a strong alignment for both patterns after optimisation.

In this experiment, two periodic patterns are processed: a weekly pattern P_1 and a monthly pattern P_2, both affected by noise. The optimisation was performed in two stages. In the first stage, the algorithm focused on identifying weekly periodic patterns, successfully identifying and aligning P_1's payments while filtering out noise, except the noisy payment on day 175. Due to the high deviation from the average and, possibly, a difficulty of the algorithm to remove the correct payments initially, we observe an initially lower level of resonance power. However, as the optimisation progresses, the resonance power reaches an acceptable level.

Once P_1 was optimised, all its contributions were removed from further processing, resetting the dataset to its original state, and the algorithm was rerun with the second dataset to detect the monthly pattern P_2.

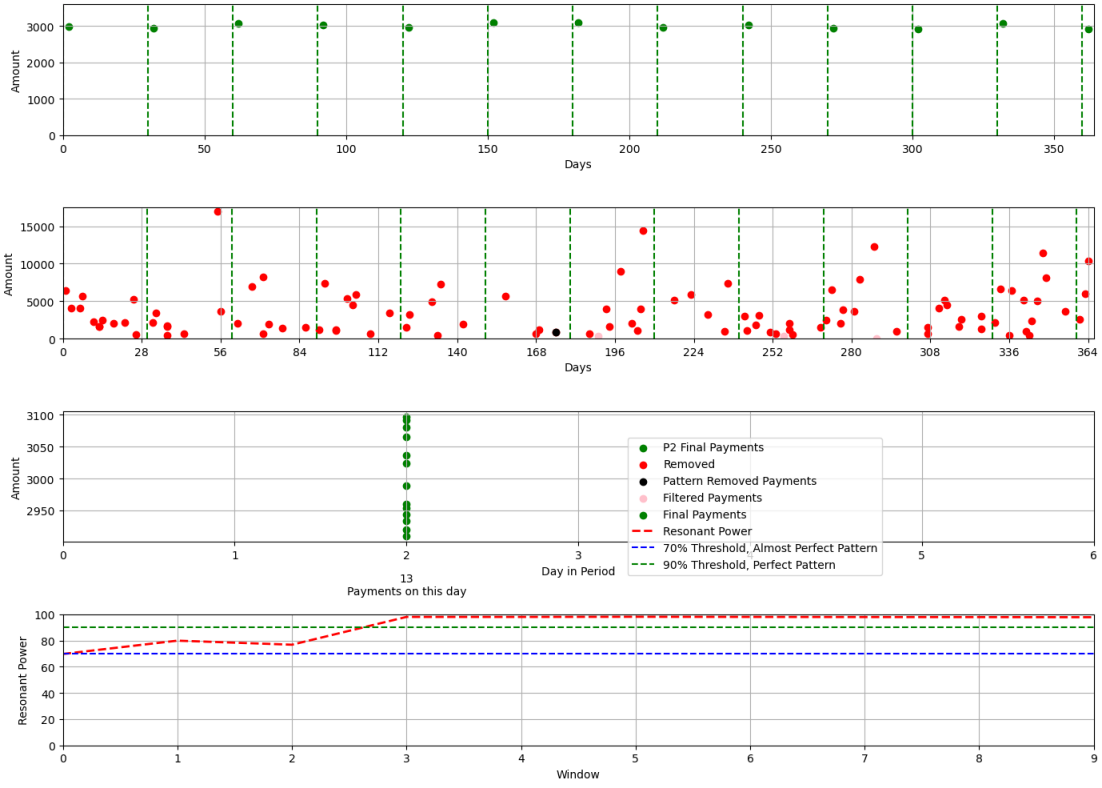

FIGURE 4.14: The algorithm initially detects and optimises a weekly pattern (P_1, blue) and subsequently removes it to focus on the monthly pattern (P_2, green). The resonance power shows high alignment with both patterns after optimisation.

The monthly pattern aligns well after optimisation, as shown with the green markers. The resonance power plot highlights the strong alignment with both patterns, with values consistently approaching the perfect threshold after the optimisation steps. The clear separation of periodicities demonstrates the robustness of the method in handling multi-pattern scenarios.

4.1.6 Two Patterns that one does not span the full timeline

One key challenge in analysing payment patterns is handling scenarios where a pattern does not span the entire timeline. For example, a business may have recurring payments that only occur for a limited period of the year. To address this, we conducted an experiment where we introduced two patterns: one that runs for the entire year, representing a "full" pattern with a payment amount of 3000, and another pattern that only runs for 280 days, with a payment amount of 8000. The purpose of this experiment is to observe how well the algorithm identifies partial patterns and flags irregularities.

Pattern 1 - End Day = 364				
Periodicity	Start Day	dDay	Amount	dAmount
Weekly	0	2	3000	100
Pattern 2 - End Day = 280				
Periodicity	Start Day	dDay	Amount	dAmount
Weekly	5	2	8000	100
Noise				
Type	Scale	pi	Noise Length	Data Length
Exponential	3000	0.3	131	223

TABLE 4.6: Experiment 6 Configuration. Two patterns of the same periodicity are tested in this case, but one of them is cut short.

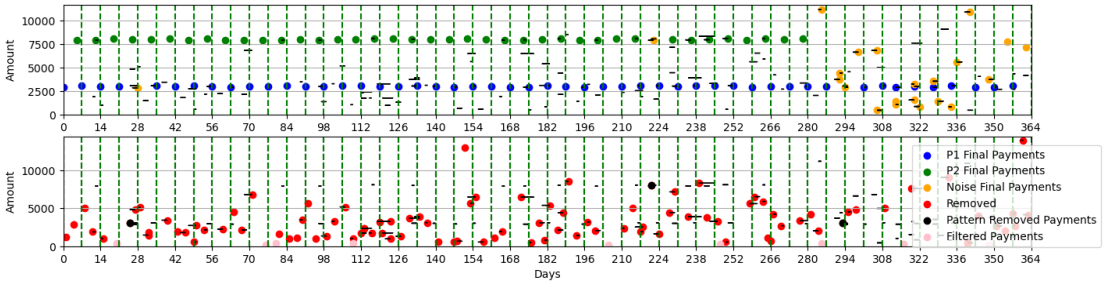

FIGURE 4.15: The figure shows the final payments of two patterns, the partial pattern P_2 with a higher amount, active for the first 280 days, and the full-year pattern P_1 with a lower amount. Noise payments (orange) are being flagged as part of the patterns to cover in place of the P_2. The absence of P_2 after day 280 causes irregularities, reflected in the misalignment and classification of payments.

Figure 4.15 highlights the complexities that arise when dealing with partial patterns, especially when a pattern with a higher payment amount does not span the entire timeline. The absence of payments from the higher-value pattern significantly impacts the objective, which result to the orange payments being retained.

This behaviour reveals a key limitation of the optimisation approach: it does not inherently recognize partial patterns that span only a portion of the timeline. This highlights the need for additional constraints or logic to account for patterns with limited durations to preserve the integrity of all patterns in the dataset.

To address this, we introduce a flagging mechanism to identify problematic sections of the timeline. A region is flagged if it meets the following criteria:

- The resonant power of window i+1 is lower than that of window i, indicating a possible start of a worsening in alignment.
- The resonant power of window $i + 2$ is at least 5% lower than the one of window i. This information shows us that the decrease in the resonance power over the two windows is more than the 1-3% we have empirically seen due to be caused by deviations of the retained payments.
- The resonant power of window $i + 2$ is lower than 90%. That is the threshold we have observed that the payments either exhibit high deviations but still form a pattern, or that high portions of noise are retained, forming unexpected pattern.
- The last window's resonant power is lower than that of window i that is the beginning of the flagging area. This provides us with the information that the algorithm did not manage to deal with this obstacle that appeared on window i and decreases the resonant power significantly.

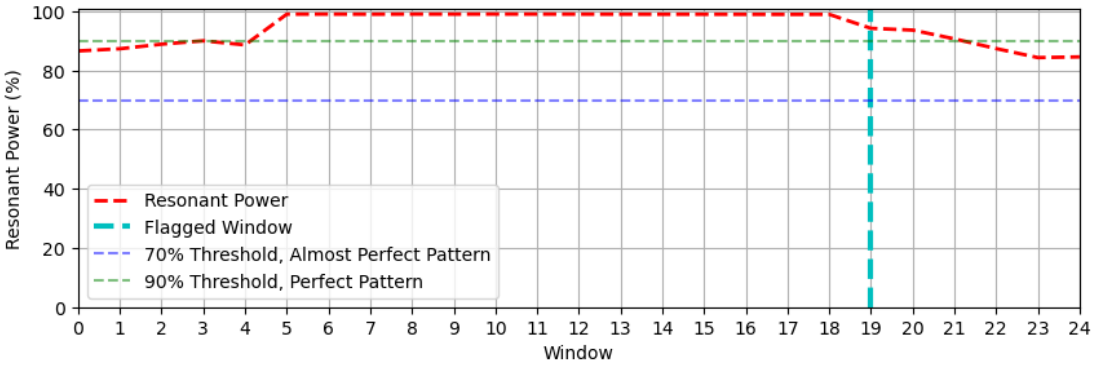

FIGURE 4.16: In this figure we observe the resonance power as the windows progress. Around window 19, however, a significant drop in resonant power occurs. This drop corresponds to the timeline where the higher-value pattern ends, leaving gaps in the payment schedule. This window is flagged according to the criteria cited above to express the uncertainty in the payments that comprise the patterns from that window onwards.

To sum up, this flagged region demonstrates the algorithm's sensitivity to partial patterns. The sudden drop in resonant power reflects the difficulty of the algorithm in identifying patterns correctly when the higher-value pattern P_2 ends prematurely. The flagged area highlights the sections where additional adjustments or supplementary data would be required to improve accuracy.

Additionally, the flagged area effectively captures periods where the absence of expected payments from P_2 leads to the misclassification of P_1 payments. For example, in the

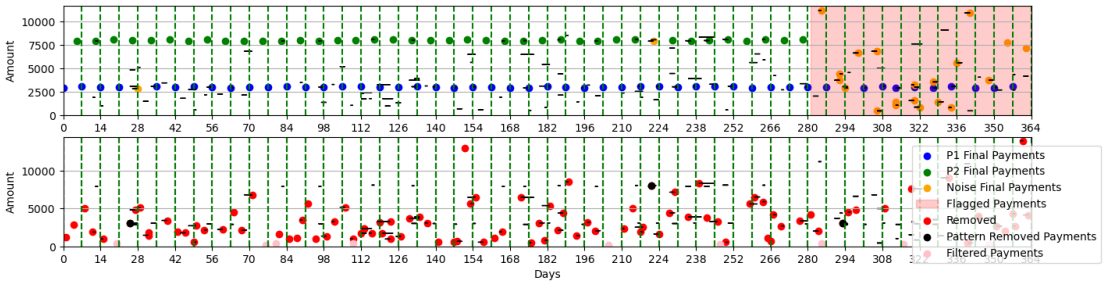

FIGURE 4.17: This flagged region, which shows the payments when the payments of the window of the flagged window and onwards, corresponds to the part of the timeline where the higher-value partial pattern is missing. Its absence results in alignment issues, leading to misclassifications and irregularities in the remaining full-year pattern too.

non-flagged area, the P_1 payments are consistently aligned with day 0, as expected. However, in the flagged area, we observe that the algorithm moves P_1 payments to day 5 in four instances to compensate for the gaps left by the absent P_2 payments.

This behaviour is mathematically expected. Recall that the objective function minimises the variance of payments on each day across periods. When P_2 is absent in some periods, the average amount for those days is calculated as $a_{ij} = \frac{\text{total payments on day } j}{\text{number of active periods}}$. With no payment on day 5 from P_2, $a_{ij} = 0$ for that day in those periods. This creates a significant deviation between the expected average ($a \approx 8000$ for P_2) and the actual amount ($a_{ij} = 0$), leading to a high penalty in the objective function.

To mitigate this penalty, the algorithm attempts to "fill the gap" by moving one or a combination of payments, possibly from P_1, or noisy ones, to day 5, as long as the payments are allowed to move to day 5. This reduces the variance across periods for day 5, even though it misclassifies the P_1 payment. Mathematically, this behaviour arises because the objective function does not account for the fact that P_2 spans only part of the timeline. Instead, it assumes that every day in the timeline should have payments aligning with the active patterns, resulting in a misalignment of P_1 payments to compensate for P_2's absence.

In conclusion, the flagged area highlights this misclassification, underscoring the limitations of the algorithm when dealing with partial patterns. This insight can guide improvements to the algorithm, such as introducing constraints or penalties that account for patterns spanning only part of the timeline.

4.1.7 Single pattern in a very high noise scenario

In this experiment, we stress test the algorithm's performance in identifying a single recurring pattern under a high level of noise. The goal is to evaluate the algorithm's

ability to distinguish between legitimate payments and noise, particularly when noise is both frequent and potentially similar in amount to the pattern.

Pattern 1				
Periodicity	Start Day	dDay	Amount	dAmount
Weekly	0	2	3000	100
Noise				
Type	Scale	pi	Noise Length	Data Length
Exponential	3000	0.5	190	242

TABLE 4.7: Experiment 7 Configuration. Now, we are testing the algorithm's performance in case we raise the noise to 50%.

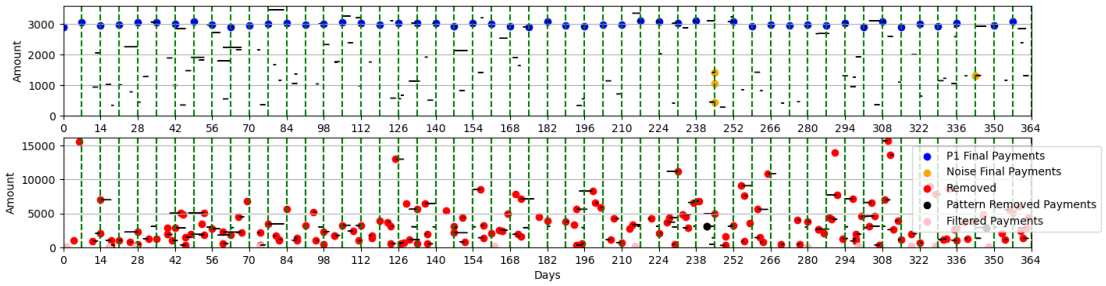

FIGURE 4.18: The figure shows the algorithm's ability to distinguish the pattern in a very noisy scenario. However, we also see that combination of payments are preferred in a few occasions over the pattern payments.

Figure 1 4.18 displays the final payment configuration and the noise removal process. The upper plot shows the optimised payments in blue, aligned with the recurring weekly pattern, while the bottom plot highlights the removed payments in red. Notably, due to the high noise level, several noisy payments were retained as part of the pattern. For the first time we also see that three payments are retained in place of one. This outcome aligns with the inherent trade-offs in the algorithm's decision-making process, where payments with amounts close to the average pattern amount and reasonably aligned within the periodic structure are more likely to be retained.

In Figure 4.19 we observe that in earlier windows, where noise significantly impacts the payment alignment, the resonance power is lower, reflecting the algorithm's limitations under high noise conditions. However, the algorithm performs very well in the end, showing its robustness towards excessive noise.

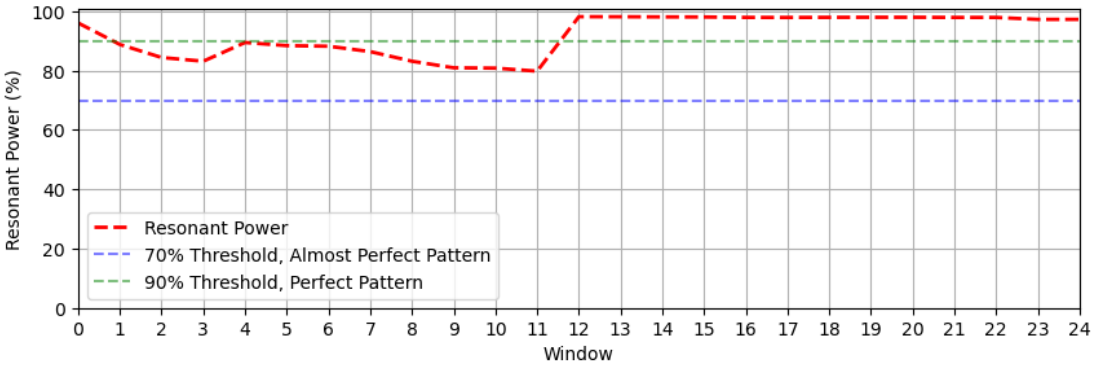

FIGURE 4.19: Initially, the resonance power fluctuates due to misaligned and noisy payments, but as the optimisation progresses, the algorithm effectively aligns the payments.

Chapter 5

Discussion

The experiments presented in this work provided a thorough examination of the capabilities of the algorithm in detecting recurring payment patterns under a variety of conditions. This chapter delves into the robustness and performance of the algorithm, as well as the implications of design choices, the challenges and limitations of the algorithm.

5.1 Performance and Robustness of the Algorithm

5.1.1 Noise Handling and Deviations

A key objective of the algorithm is to filter out noise while preserving recurring payment patterns. Across varying levels of noise, the algorithm demonstrated both strengths and areas requiring improvement. At lower noise levels, the algorithm effectively identified patterns and aligned payments to their intended periodicities, showcasing robust noise-handling capabilities. However, as noise levels increased, distinguishing genuine pattern payments from noise became challenging. When the amount of noisy payments rose, combined with deviations in the amounts of the pattern, the likelihood of sums of noisy payments being closer to the average than the respective pattern payment increased. Evidently, when the noise scale factor was set at 50%, the algorithm retained more noisy payments resembling the pattern. Conversely, legitimate pattern payments that slightly deviated from expectations were sometimes misclassified as noise.

The more complex experiments, involving multiple periodicities and/or incomplete patterns, showed how the algorithm handles conflicting patterns and gaps in the timeline. The results indicated that, while the algorithm is capable of correctly identifying multiple patterns, its tendency to fill gaps in the timeline with any available payment can

lead to misclassifications, especially when the payments of different patterns are close in value.

This trade-off underscores the need for an approach that can simultaneously tackle noise and deviations more effectively. While the algorithm's mechanism to approximate the average pattern amount per period and align to the periodic structure, incorporating additional constraints can improve its ability to handle both noise and deviations without sacrificing the integrity of the pattern.

5.1.2 Trade Offs in Optimisation

The algorithm attempts to balance two competing goals: retaining the payments that correctly form the pattern while removing the noise. A form of penalty mechanism is determining the optimisation, one that penalises deviations from the expected pattern. A stricter penalty could encourage adherence to the pattern and reducing false positives, but it would potentially remove pattern payments that show slight irregularities. While this penalty term can act as a fine tuning mechanism, we opted against introducing additional parameters. The rationale behind this decision was to avoid complexities in tuning and to maintain a free-parameter algorithm, as it is often difficult to find a single setting that performs well in different scenarios.

The results showed that in many cases, the algorithm effectively retained valid payments and filtered out noise. Nevertheless, there were examples where the amount deviation of specific noisy payments was close enough to the pattern's average, leading to their inclusion in the final pattern. On the other hand, valid pattern payments that strayed slightly from their expected behaviour were occasionally misclassified as noise.

5.1.3 Windowed optimisation

The use of windowed optimisation enabled more efficient, context-based optimisation. By focussing on a section of the timeline at each step, the system was able to make small changes to payments, progressively increasing alignment and noise handling. The window and step sizes are important considerations in determining the efficacy of this strategy. A smaller window size allows for finer changes and captures more localised data, but it runs the risk of overfitting to slight differences in payment behaviour. A bigger window size, on the other hand, captures a broader context, aligning payments over a longer timeframe while possibly disregarding minor irregularities.

In this setting, the default window size and step size were set to four and two periods, respectively. This decision was deliberate, as four periods are the smallest window size

that we could expect to be able to demonstrate periodic behaviour. Such a window size gives sufficient context to detect patterns without being excessively restrictive or prone to wrongly interpreting fluctuations as meaningful behaviour. This ensures that the algorithm has enough data within each window to recognise patterns, allowing it to effectively align and optimise payments based on their frequency.

Furthermore, the step size of two periods enables regulated expansion between windows. This overlap guarantees that any patterns observed in one window are taken into account in the next, improving payment alignment as the timeline progresses. The combination of a 4-period window and a 2-period step size provides a balanced approach, allowing the algorithm to capture long-term patterns while being adaptable to changes over shorter time periods.

Overall, the windowed optimisation method was effective in balancing computation time and pattern identification accuracy. However, in certain circumstances with complex payment structures or overlapping periodicities, the windowed component of the optimisation made it difficult to capture the complete recurrent pattern, especially if it spanned numerous windows.

5.2 Challenges in Pattern Recognition

5.2.1 Definition

One of the most significant challenges in pattern recognition, as evidenced by the experiments, is the ambiguity in defining what constitutes a pattern. In the example where P_1 spanned the full timeline while P_2 was cut off midway, the question if we observed two patterns or one pattern with two instances per period is raised. Is the purpose of pattern recognition to identify payments of the same sum per period or to identify the exact same payments over these periods? Unfortunately, the answer is not definite and choices have to be made.

5.2.2 Multiple Periodicities

Overall, the windowed optimisation strategy was effective in balancing computation time and pattern identification accuracy. However, in certain circumstances with complex payment structures or overlapping periodicities, the windowed aspect of the optimisation made it difficult to capture the complete recurrent pattern, especially if it spanned numerous windows.

This strategy did not, however, come without drawbacks. The algorithm sometimes had trouble distinguishing separate patterns when payments from several periodicities were very similar to one another. For instance, if the amount or time of a biweekly payment series is similar, it could be mistaken for a weekly pattern. This overlap may cause payments to be misclassified, incorrectly falling under a different periodicity or being written off as noise.

In other experiments with incomplete patterns (e.g., P_2 running only for 280 out of 364 days), the algorithm was able to flag segments of the timeline where the absence of payments created irregularities. However, it often filled these gaps with payments from either noise or the other pattern, leading to misclassification.

5.3 Limitations and Time Complexity

In this project, the timeline is set to one year. But in the algorithm's settings, numerically, the length of the timeline was set to 364 days instead of the conventional 365. This choice was made due to its impact on the behaviour of the algorithm. For example, with a weekly periodicity, a window size of 4 periods, and a step size of 2 periods, the final window tested in the case of an end day set to 364 would span from 0 to 364.

However, if the timeline was extended to 365 days, an additional window from 0 to 378 would be required. However, the second week of the extra window would not contain any payments, introducing a significant penalty to the objective. This penalty would disproportionately lower the resonance power and affect the optimisation results. This is an inherent limitation of the windowed optimisation, which computes the objective even when there are no payments to align, thus raising the objective. To address this, the timeline is truncated to the last complete step of the optimisation, ensuring that the last window still contain payments and unnecessary penalties are avoided.

The current version of the algorithm operates in two primary phases. First, it runs a preliminary optimisation with only "move" and "swap" operations on the entire timeline, followed by a windowed optimisation phase where all operations (move, swap, remove, and swapremove) are considered. While this stepwise approach is effective in filtering out noise and aligning payments to their expected patterns, it also introduces certain trade-offs.

The "move" and "swap" operations allow for efficient adjustments of payments, thereby optimising the objective function by realigning payments to their ideal positions in the timeline. However, restricting these operations to only move and swap in the initial phase introduces a risk of misclassifying some payments, as later changes may require

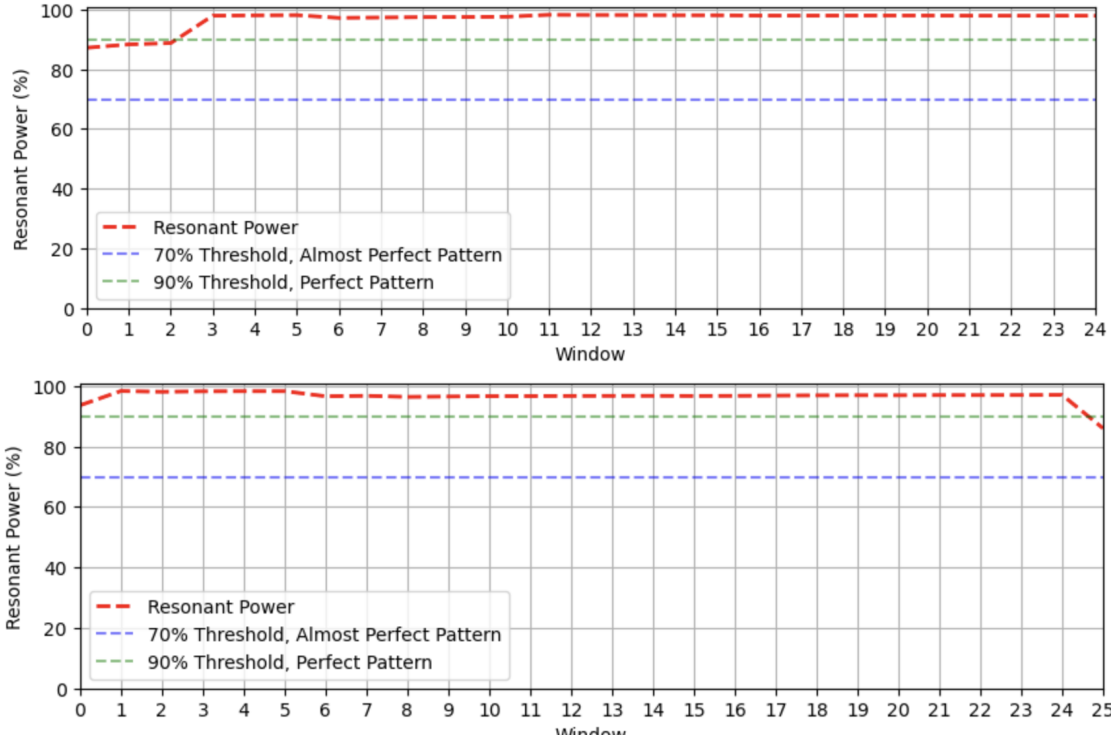

FIGURE 5.1: The figure shows the algorithm's limitation to the length of the timeline. We see in the bottom plot, where the timeline length is 365 days, there is a penalty on the last window. This is caused by the lack of payments in the last period.

multiple operations. For example, a payment might already be in the position that it is expected to be, however multiple noisy payments might be moved to its location. Thus, in the second phase, it is more beneficial to remove the pattern payment and make the objective better in one step.

The primary limitation of the current algorithm is its speed, particularly as the length of the timeline increases. The algorithm's computational cost grows exponentially with longer timelines, which is why the experiments were conducted over 210 days to maintain a manageable runtime. For real-world applications involving longer timelines, optimising the algorithm's efficiency is critical.

The time complexity of the algorithm is primarily driven by the windowed optimisation process. Given a timeline of length N, a period T, and window size W, the complexity can be approximated as $O(N \cdot T)$. This complexity arises because each window requires iterating over the payments and performing operations to optimise their alignment. The performance is further influenced by the number of operations allowed (move, swap, remove), each of which has its own computational cost. Therefore, while the algorithm performs well on shorter timelines, its scalability to longer timelines remains an area for improvement.

Chapter 6

Conclusion and future work

6.1 Conclusion

This thesis explored a new method for detecting and optimising recurring payment patterns in noisy datasets. The goal was to identify these patterns accurately while handling variations and noise effectively. The proposed algorithm, which uses windowed optimisation and basic operations, performs well in challenging scenarios but struggles with high noise levels and overlapping patterns. However, certain limitations remain, particularly in handling high levels of noise and overlapping patterns.

The use of windowed optimisation, where data is processed sequentially in smaller sections, was a key part of the algorithm's success. This approach allowed for step-by-step improvements in pattern recognition, leading to better alignment of payments and an optimised result by the end of the process.

A key strength of the method is its flexibility. The algorithm uses moves, swaps, removals, and swap-removals to handle irregularities and align payments simultaneously. The objective of this algorithm is determined by the deviations of the payments, which can also be considered a penalty term. However, to maintain simplicity, no additional hyperparameters were introduced, ensuring the method remains straightforward and generalizable.

The objective of the algorithm is closely connected with the notion of resonance power that we introduced. It measures the alignment of payments within their expected periodic cycles and reflects the performance of the algorithm across all optimisation windows. It provides a quantifiable way to track how well payments are aligned to their patterns as the optimisation progresses.

In this thesis, we tested the ability of the algorithm to handle high levels of noise and manage payment patterns in various scenarios. It showed promising results in dealing with complex patterns and periodicities. However, it is worth noting that the use of synthetic data limits the exploration and understanding of the algorithm's robustness, when confronted with unpredictable data.

To summarise, the method presented in this thesis provides a unique approach to detecting recurring payment patterns within noisy timelines. By initially aligning payments and, then, optimising them using an expanding window technique, the algorithm attempts to reveal hidden patterns. However, the algorithm is not scalable to real-world data yet, as it lacks the computational efficiency to deal with vast datasets. Furthermore, the need to find a way to deal with patterns that deviate from their original amount is apparent. Scenarios where the average amount changes over time, possibly due to inflation for example, cannot be considered in the algorithm's current state. Further work is needed to explore different optimisation strategies that can enhance its performance and scalability

6.2 Future Work

The experiments in this thesis showed much room for improvement to handle noise, complex patterns and computational demands. Potential future steps of the algorithm follow below.

The current noise-filtering method relies on a simple percentile approach, which can struggle in high-noise scenarios. A more robust alternative is to use Kernel Density Estimation (KDE) methods. KDE can help identify regions with higher probabilities of noise versus genuine patterns.

For example, DBScan [26, 27] groups points based on density, identifying clusters while labelling outliers as noise. However, it is difficult to tune the KDE for noise reduction, as under- or overestimating the noise can remove legitimate payments from the dataset.

Currently, the algorithm relies on a set of basic operations, prioritizing simplicity, clarity, and explainability. However, additional operations could address specific issues, such as combining payments. For instance, a double removal paired with a swap involving inactive payments could potentially resolve some misclassifications. Yet, as seen in the experiment with a 50% noise factor, the algorithm replaced a single expected payment with three noisy ones in a period. In such cases, a double removal and swap approach would not solve the problem.

This raises the challenge of handling deviations more effectively. One potential solution is to calculate the average number of payments per period required to achieve an optimal pattern. By introducing a penalty term, the algorithm could be forced to maintain this average, ensuring it retains the number of payments that lower the objective function the most.

The algorithm's current objective function does not account for gaps in the timeline effectively, leading to penalties when expected payments are absent. If a payment within the expected range in amount and day is absent, the algorithm could have an option to either find an alternative payment to cover the gap or skip counting that period altogether. This would prevent gaps from negatively impacting both the objective and the resonant power calculations. Essentially, if a period's objective value is too low, the algorithm could opt to exclude it from both the objective and power calculations, improving the alignment without distorting the assessment of the overall pattern.

Another potential improvement involves refining the flagging mechanism to adapt dynamically to different datasets and noise conditions. The current approach relies on fixed thresholds, such as a 5% drop in resonant power, which may not be generalised across all scenarios. An adaptive flagging mechanism could adjust these thresholds based on statistical properties of the dataset, such as the variance in resonant power over multiple windows, equipping the algorithm with the ability to dynamically determine when to flag problematic regions.

Finally, vectorising the algorithm is another key area for future work. The current approach was designed to be more intuitive for the sake of clarity and understanding. However, a vectorised version could significantly enhance computational efficiency. While this might involve a more complex implementation, the speed-up from such a change would make the algorithm more scalable and applicable to larger datasets.

Bibliography

[1] Effective transaction pattern analysis in banking systems. https://insightfulbanking.com/transaction-pattern-analysis/, 2024. Accessed: 2024-10-22.

[2] Betalen aan de kassa 2023, 2023. URL https://www.dnb.nl/media/3nmfkclc/78259_dnb_betalen_aan_de_kassa_2023_v4_web.pdf.

[3] Ron Triepels, Hennie Daniels, and Ronald Heijmans. Anomaly detection in real-time gross settlement systems. In *Proceedings of the 19th International Conference on Enterprise Information Systems*. SCITEPRESS - Science and Technology Publications, 2017. ISBN 9789897582479.

[4] Carlos León. Detecting anomalous payments networks: A dimensionality-reduction approach. *Latin American Journal of Central Banking*, 1(1-4):100001, 2020.

[5] Carlos León, Paolo Barucca, Oscar Acero, Gerardo Gage, and Fabio Ortega. Pattern recognition of financial institutions' payment behavior. *Latin American Journal of Central Banking*, 1(1–4):100011, 2020. ISSN 2666-1438. doi: 10.1016/j.latcb.2020.100011. URL http://dx.doi.org/10.1016/j.latcb.2020.100011.

[6] Bill Chiu, Eamonn Keogh, and Stefano Lonardi. Probabilistic discovery of time series motifs, 2003. URL https://www.cs.ucr.edu/~eamonn/SIGKDD_Motif.pdf.

[7] Jessica Lin. Experiencing sax: A novel symbolic representation of time series, 2007. URL https://cs.gmu.edu/~jessica/SAX_DAMI_preprint.pdf.

[8] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. Matrix profile i: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, page 1317–1322. IEEE, 2016. ISBN 9781509054732.

[9] Chin-Chia Michael Yeh, Nickolas Kavantzas, and Eamonn Keogh. Matrix profile iv: Using weakly labeled time series to predict outcomes. *Proceedings of the VLDB*

Endowment International Conference on Very Large Data Bases, 10(12):1802–1812, 2017. ISSN 2150-8097. doi: 10.14778/3137765.3137784. URL http://dx.doi.org/10.14778/3137765.3137784.

[10] Frank Madrid, Shima Imani, Ryan Mercer, Zachary Zimmerman, Nader Shakibay, and Eamonn Keogh. Matrix profile xx: Finding and visualizing time series motifs of all lengths using the matrix profile. In *2019 IEEE International Conference on Big Knowledge (ICBK)*. IEEE, 2019. ISBN 9781728146072.

[11] Takaaki Nakamura, Makoto Imamura, Ryan Mercer, and Eamonn J. Keogh. Merlin: Parameter-free discovery of arbitrary length anomalies in massive time series archives. *Industrial Conference on Data Mining*, page 1190–1195, 2020. doi: 10.1109/ICDM50108.2020.00147. URL https://www.cs.ucr.edu/~eamonn/MERLIN_Long_version_for_website.pdf.

[12] Sara Alaee, Kaveh Kamgar, and Eamonn Keogh. Matrix profile xxii: Exact discovery of time series motifs under dtw. In *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2020. ISBN 9781728183169.

[13] Meinard Muller. Dynamic time warping. *Information retrieval for music and motion*, pages 69–84, 2007.

[14] Dieter De Paepe and Sofie Van Hoecke. Mining recurring patterns in real-valued time series using the radius profile. In *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2020. ISBN 9781728183169.

[15] Rik van der Vlist, Cees Taal, and Richard Heusdens. Tracking recurring patterns in time series using dynamic time warping. In *2019 27th European Signal Processing Conference (EUSIPCO)*. IEEE, 2019. ISBN 9789082797039.

[16] Diego Furtado Silva and Gustavo E. A. P. A. Batista. Elastic time series motifs and discords. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2018. ISBN 9781538668054.

[17] Mahtab Mirmomeni. *Detection of Repeating Activities Recorded by Sensors in the Absence of Labeled Data.* PhD thesis, University of Melbourne, Parkville, Victoria, Australia, 2022.

[18] Maurizio J.A. Sluijmers. Predicting future incoming bank transactions by detecting recurring transaction sequences. Master's thesis, Technische Universiteit Eindhoven, 2022.

[19] Chenyang Wang, Ling Luo, and Uwe Aickelin. *Quasi-periodicity detection via repetition invariance of path signatures*, page 301–313. Springer Nature Switzerland, 2023. ISBN 9783031333828.

[20] D.D. Muresan and T.W. Parks. Orthogonal, exactly periodic subspace decomposition. *IEEE Transactions on Signal Processing*, 51(9):2270–2279, 2003. doi: 10.1109/TSP.2003.815381.

[21] W. A. Sethares and T. W. Staley. Periodicity transforms. *IEEE transactions on signal processing: a publication of the IEEE Signal Processing Society*, 47(11): 2953–2964, 1999. ISSN 1053-587X. doi: 10.1109/78.796431. URL http://dx.doi.org/10.1109/78.796431.

[22] Deepa Abraham and Manju Manuel. Signal periodicity detection using ramanujan subspace projection. *Journal of Electrical Engineering*, 71(5):326–332, 2020. ISSN 1335-3632. doi: 10.2478/jee-2020-0044. URL http://dx.doi.org/10.2478/jee-2020-0044.

[23] P. P. Vaidyanathan. Ramanujan sums in the context of signal processing—part i: Fundamentals. *IEEE transactions on signal processing: a publication of the IEEE Signal Processing Society*, 62(16):4145–4157, 2014. ISSN 1053-587X. doi: 10.1109/tsp.2014.2331617. URL http://dx.doi.org/10.1109/tsp.2014.2331617.

[24] Shi-wen Deng and Ji-qing Han. Signal periodic decomposition with conjugate subspaces. *IEEE Transactions on Signal Processing*, 64(22):5981–5992, 2016. doi: 10.1109/TSP.2016.2600509.

[25] Ali Grami. Chapter 3 - signals, systems, and spectral analysis. In Ali Grami, editor, *Introduction to Digital Communications*, pages 41–150. Academic Press, Boston, 2016. ISBN 978-0-12-407682-2. doi: https://doi.org/10.1016/B978-0-12-407682-2.00003-X. URL https://www.sciencedirect.com/science/article/pii/B978012407682200003X.

[26] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Knowledge Discovery and Data Mining*, 1996. URL https://api.semanticscholar.org/CorpusID:355163.

[27] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. Dbscan revisited, revisited: Why and how you should (still) use dbscan. *ACM Trans. Database Syst.*, 42(3), jul 2017. ISSN 0362-5915. doi: 10.1145/3068335. URL https://doi.org/10.1145/3068335.