Automatic Spike Sorting and Robust Power Line Interference Cancellation for Neural Signal Processing

Mohammad Reza Keshtkaran

(B.Sc., Shiraz University)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2014

تقدیم به بی همتا انسانی که از برکت وجودش صبر،
درس ایستادگی و ایثار از خود گذشتن را آموخت،
و عشق تعلیم مادری یافت...

To the memory of my mother...

Declaration

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Mohammad Reza Keshtkaran

18 August 2014

Acknowledgements

I would like to take this opportunity to express my sincere appreciation to all those who supported me during my PhD pursuit. Without their help and support this thesis would not have been possible.

I would like to express my gratitude towards my supervisor Dr. Zhi Yang, for his guidance, encouragement and support. I sincerely thank my doctoral committee A/Prof. Chun-Huat Heng, and A/Prof. Cheng Xiang for their insightful feedback on my work and this thesis. I would like to thank Prof. Karim Rastgar and Prof. Mohammad Ali Masnadi-Shirazi, my undergraduate advisors who have been far beyond mentors for me both in my academic and personal life. I am also grateful to Prof. Teng Joon Lim for his generous time and helpful advice.

I would like to thank A/Prof. Shuicheng Yan for helpful technical discussions, and the course on pattern recognition. Some of the ideas presented in this thesis would not have been developed without the insightful course I took with him.

I am grateful to Mojtaba Ranjbar, Amir Tavakkoli K.G., Mahmood Khayatzadeh, Mehdi Jafary-Zadeh, Mehran M. Izad, Narjes Allahrabi, Roya Bazyari, Zahra Kadivar, Sahra Sedigh and many others who have helped me during my PhD journey. I thank my friends Akbar, Ahmad, Atieh, Mahsa, Siavash, Pooya, Mohammad, Amin, Sajjad, Sadegh, Kamran, Mahyar, Mostafa, Navid, Dorsa, Elham, Maryam, Omid, Farshad, Zeinab, Maedeh and my other friends for the great friendship and all the good time we have had together. I would like to thank all my colleagues and friends in Signal Processing and VLSI Design Lab, especially Tong Wu for technical helps.

I am deeply indebted to my father and sisters Shahrzad, Shahrnaz, Parinaz and Parisa, for their eternal love, patience, and unwavering support throughout my life and especially in the last four years. I dedicate this thesis to the memory of my mother. Every bit of success that I have had or will have in my life

undoubtedly arises from her ineffable love, selfless sacrifices, and invaluable support.

Contents

Summary

Recording the electrical activity of the brain has permitted researchers to analyse cognition and study the brain's mechanisms of information processing. Extracellular recording is a method of measuring neuronal activity through inserting microelectrodes into the brain tissue which picks up neural signals from population of neurons i.e. local field potentials (LFPs), action potentials from a few surrounding neurons (neural spikes), and noise.

Recently, there has been an increasing attention to the LFP gamma oscillations (> 30 Hz) due to their correlation with a wide range of cognitive and sensory processes. However, gamma oscillations are usually corrupted by power line interference at 50/60 Hz and harmonic frequencies. It is therefore desired to remove the interference without compromising the actual neural signals at the interference frequency bands. Available real-time methods either fail to work on neural signals or produce excessive distortion in the interference bands. The first objective of this thesis was thus to develop a robust and efficient algorithm to remove power line interference from neural recordings. We present the theory and structure of the algorithm followed by implementation details and practical discussions. While minimally affecting the signal bands of interest, the proposed algorithm consistently yields fast convergence (< 100 ms) and substantial interference rejection (output SNR > 30 dB) in different conditions of interference strengths (input SNR from -30 dB to 30 dB), power line frequencies (45–65 Hz), and phase and amplitude drifts. In addition, the algorithm features a straightforward parameter adjustment since the parameters are independent of the input SNR, input signal power, and the sampling rate. As the next aim of the thesis, the VLSI architecture and ASIC of the proposed algorithm is presented for real-time interference cancellation in multichannel recording. The proposed architecture is scalable with respect to the number of channels and/or harmonics,

where the performance is optimized through pipelining and resource sharing techniques. The ASIC was fabricated in a 65-nm CMOS technology consuming 0.11 mm^2 of silicon area and 77 μW of power.

In addition to LFP, signals from individual neurons (single-unit) are of particular interest in many neuroscience studies and brain machine interface applications. However, implanted microelectrodes record the superimposed spikes from multiple surrounding neurons. Thus it is necessary to identify and cluster (i.e. sort) the spikes from multiple neurons in order to obtain the single-unit activity. A crucial stage in spike sorting is feature extraction which determines the quality of the next stage clustering. Conventional spike feature extraction approaches give a projection subspace which does not necessarily provide the most discriminative subspace for clustering. Hence, the clusters which appear inherently separable in some discriminative subspace may overlap if projected using conventional feature extraction approaches, leading to a poor sorting accuracy especially when the noise level is high. The further objective of this thesis was to develop a noise robust and unsupervised spike sorting approach based on learning discriminative spike features. First, two unsupervised discriminative subspace learning approaches which can handle outliers in data are presented. We further introduce methods for selecting the number of neurons along with these approaches. Based on these methods, we propose two automatic spike sorting algorithms whose comparative simulation results on synthetic and *in-vivo* recordings indicate high sorting accuracy, significantly better separability of clusters, and high level of robustness to noise.

List of Tables

List of Figures

List of Symbols

$\hat{a}_k$	Estimate of a_k
α	General symbol for pole radii
α_0	Initial pole radii of the ANF
α_{st}	Rate of change from α_0 to α_∞
α_∞	Asymptotic pole radii of the ANF
α_{f}	Pole radii of the ANF
f_{s}	Sampling rate (Hz)
γ	Smoothing parameter of the frequency estimator
γ'	Cut-off frequency of the smoothing filter; set at 90 Hz
$\hat{b}_k$	Adaptive coefficient for amplitude/phase estimation
$\hat{c}_k$	Adaptive coefficient for amplitude/phase estimation
$\hat{h}_k(n)$	Estimate of h_k
$\hat{h}_{i,k}$	Estimated k^{th} harmonic for channel i
$\hat{p}(n)$	Estimate of $p(n)$
$\hat{s}(n)$	Estimate of $s(n)$
κ_k	Frequency control parameters for harmonic k
κ_{f}	Adaptive coefficient for frequency estimation
λ	General symbol for forgetting factor
λ_0	Initial forgetting factor of the frequency estimator
λ_{st}	Rate of change from λ_0 to λ_∞
λ_∞	Asymptotic forgetting factor of the frequency estimator

λ_{a} Forgetting factor of the amplitude/phase estimator

λ_{f} Forgetting factor of the frequency estimator

$\mathbf{R}_k$ RLS sample correlation matrix for harmonic k

$\mathbf{U}_k$ RLS input sample vector for harmonic k

$\mathbf{W}_k$ RLS coefficients vector for harmonic k

$\mathbf{j}$ Unit imaginary number

$\mathcal{C}_k$ Cluster k

$\mathcal{I}m\{z\}$ Imaginary part of z

$\mathcal{T}$ Function for a test of unimodality

ϕ_k Phase of k^{th} harmonic

$\phi_{i,k}$ Phase of the k^{th} harmonic in channel i

$\hat{\phi}_k$ Estimate of ϕ_k

ψ_k Initial phase shift of u_k and u'_k

$\boldsymbol{\mu}_k$ Center of cluster k

$\mathbf{x}_i$ i^{th} spike samples

ω_{f} Fundamental frequency of the interference in rad/s

$\hat{\omega}_{\mathrm{f}}$ Estimate of ω_{f}

a_k Amplitude of k^{th} harmonic

$a_{i,k}$ Amplitude of the k^{th} harmonic in channel i

B_0 Initial notch bandwidth of the frequency estimator (Hz)

B_∞ Asymptotic notch bandwidth of the frequency estimator (Hz)

B_{st} Settling time from B_0 to B_∞ (s)

E_k RLS weighted squares error

e_k RLS instantaneous error

$e_{i,k}$ Instantaneous error for harmonic k and channel i

f Output of the all-pole section

f_{I} Fundamental frequency of the synthetic interference

G	Coefficient for gain control
$H(\cdot)$	40–70 Hz 4^{th} order IIR bandpass filter
h_k	k_{th} harmonic of the interference
$h_{i,k}$	k^{th} harmonic of the interference in channel i
I_2	2×2 unity matrix
K	Number of clusters
L	Cluster indicator matrix
M	Matrix if cluster centers
M'	Number of harmonics to remove
$M'_{\max}$	Maximum number of harmonics that can be present in the signal
N	Number of samples
n	Discrete sample number
$p(n)$	Power line interference at time sample n
P_0	Initial settling time of the frequency estimator (s)
$p_i(n)$	Power line interference in channel i
P_∞	Asymptotic settling time of the frequency estimator (s)
P_{st}	Settling time from P_0 to P_∞ (s)
$r_{m,k}$	m^{th} element of $\mathbf{R}_k$ for harmonic k
$s(n)$	Actual signal of interest at time sample n
S_{b}	Between-class scatter matrix
S_{w}	Within-class scatter matrix
$s_i(n)$	Actual signal of interest in channel i
t_{set}	Settling time in seconds
u'_k	Discrete oscillator state variable orthogonal to u_k
u_k	Discrete oscillator state variable
v'_k	Amplitude of u'_k
v_k	Amplitude of u_k

W Projection matrix

$w'_{i,k}$ Adaptive coefficient for harmonic k and channel i

W_{a} Settling time of amplitude/phase estimator (s)

$w_{i,k}$ Adaptive coefficient for harmonic k and channel i

X Matrix of spike samples

$x(n)$ Recorded signal from one electrode at time sample n

$x_{\mathrm{d}}(n)$ Output of the 1^{st}-order differentiator

$x_{\mathrm{f}}(n)$ Output of the bandpass filter

$x_i(n)$ Recorded signal from channel i

Y Spike feature matrix

κ_{t} Adaptive coefficient κ_{f} before smoothing

$\mathrm{SNR}_{\mathrm{in}}$ SNR of the interference canceller input signal

$\mathrm{SNR}_{\mathrm{out}}$ SNR of the interference canceller output signal

List of Acronyms

AC	Alternating Current
ANC	Adaptive Noise Canceller
ANF	Adaptive Notch Filter
ASIC	Application Specific Integrated Circuit
BMI	Brain Machine Interface
DD	Discrete Derivative
DWT	Discrete Wavelet Transform
ECG	Electrocardiography
ECoG	Electrocorticography
EEG	Electroencephalography
EM	Expectation Maximization
EMG	Electromyography
GMM	Gaussian Mixture Model
GUI	Graphical User Interface
IIR	Infinite-Impulse-Response
LDA	Linear Discriminant Analysis
LE	Laplacian Eigenmaps
LFP	Local Field Potential
MSE	Mean Squared Error
PCA	Principal Component Analysis
PLL	Phase-Locked Loop

PSD Power Spectral Density

RLS Recursive Least Squares

SNR Signal-to-Noise Ratio

SPC Superparamagnetic Clustering

VLSI Very-Large-Scale Integration

Chapter 1

Introduction

1.1 Extracellular Neural Recording

Recording the electrical activity of the brain has permitted researchers to analyse cognition and study the brain's mechanisms of information processing. Extracellular recording using micro-electrode arrays provides high fidelity signals of both single- and multi-unit activities and field potentials. Single- and multi-unit activities are spike trains that have a dominant spectrum at 300 Hz–5 kHz, while local field potentials (LFPs) are aggregated from a large number of synchronized synaptic activities with a dominant spectrum in 0.1–200 Hz. While each one possesses unique characteristics which may make it preferred over another depending on the application, both LFP and neural spikes have been widely used for brain signal analysis and information decoding [1, 2].

1.1.1 Local Field Potentials

LFPs have been receiving increasing attention in long-term BMI experiments due to their better tolerance to neural interface degeneration and glial cell encap-

sulation. In addition, different frequency bands of LFP oscillations characterise specific functional responses of population activity, and are useful to study the mechanisms of information processing of the brain.

Due to various recording imperfections and experimental protocols, neural recordings are frequently superimposed with interferences and artefacts, which can cause erroneous data analysis. A more common cause of concern is the power line interference which is mainly due to the capacitive coupling between the subject and nearby electrical appliances and mains wiring [3, 4].

For studying field potentials at lower frequencies (e.g. < 30 Hz), a low-pass filter is sufficient to reject the power line interference. However, there is an increasing attention to the gamma band oscillations (> 30 Hz) due to their correlation with a wide range of cognitive and sensory processes [4–14]. For example, the frequency bands of 80–500 Hz in [7], 40–180 Hz in [9], 76–150 Hz in [10], 0–200 Hz in [15], and 30–200 Hz in [13] have been shown useful for studying cognitive and motor processing. In this case, in addition to the fundamental harmonic at 50 Hz or 60 Hz, high order harmonics of the interference should also be removed before data analysis.

1.1.2 Neural Action Potentials

In addition to LFP which reflects the population activity, neural action potentials, which are also called spikes, provide information at the level of individual neurons which are useful for understanding the underlying mechanisms of neural processing, through for example, analysing the correlation among activities of different neurons, or observing how a neuron responds to a specific stimulus. This is one of the critical components that permits large-scale recording of neural activity [2]. Depending on the proximity of the micro-electrode to the surrounding neurons, the recording may contain several spike waveforms generated by different neurons.

An indispensable step in spike-train analysis is to sort the spikes after detection to assign each spike to its originating neuron. This is the fundamental first step in all multiple spike train data analyses, for example the analysis of spike rate, spike time synchrony, and inter-spike interval [16, 17]. The accuracy of the spike sorting critically affects the accuracy of all subsequent analyses.

1.2 Thesis Motivation

1.2.1 Power Line Interference Cancellation

Power line interference is usually non-stationary, and can vary in frequency, amplitude and phase. An ideal signal processing method should be able to quickly and accurately track these variations and cancel the interference while not compromising the neural signal of interest at the interference frequency bands. Furthermore, it is desired that the algorithm does not impose any modification or additional requirements (such as extra recording channels) on the recording hardware. Along these lines, many methods based on adaptive filtering have been proposed for interference removal from biomedical signals which are mainly proposed for electrocardiography (ECG) signal processing [18–20].

There are a few application related challenges that affect the performance of these methods when applied to neural recording. First, the spectrum of neural data follows $1/f^{x_{(1<x<3)}}$ distribution that violates the assumption of white Gaussian noise made in many algorithms, which may cause algorithm malfunction. Moreover, neural signals are non-stationary and there could be transient or sustained LFP oscillations appearing at the interference frequencies that should remain intact. The algorithms that are tailored for a certain type of biomedical recording (e.g. ECG) rely on specific signal characteristics (e.g. detection of QRS waveform) to operate adequately; this makes them not applicable to neural

recordings. When applied on neural recordings, although some of these methods can track and remove the interference, they leave high level of distortion in the signal, which should be avoided to properly retrieve the underlying LFP signals. In addition to these facts, these methods require careful tuning of their parameters to be able to operate adequately. However, the proper tuning of the parameters is usually difficult in practical applications where the interference and signal power can vary significantly. These limitations call for the design of a power line interference cancellation algorithm that: 1– works well on LFP signals and other modalities of neural recordings. 2– can cancel the interference in real-time, and have low computational complexity. 3– does not compromise the signal of interest and can work reliably under different signal and interference conditions. 4– have a straightforward parameter adjustment.

1.2.2 Clustering of Neural Action Potentials (Spike Sorting)

Common spike sorting methods involve detecting neural spikes, extracting and selecting features from the detected spike waveforms, detecting the number of neurons, and assigning the spikes to their originating neurons [16]. Among these stages feature extraction and detecting the number of neurons are specially challenging and significantly affect the accuracy and reliability of sorting process. A good feature extraction method should retain the most useful information for discriminating different spike shapes in a reasonably low dimension [17]. However, many methods including principal component analysis (PCA), discrete wavelet transform (DWT), waveform derivatives [21], Laplacian eigenmaps (LE) [22], wavelet optimization [23], and Fourier transform [24]) used for spike sorting do not necessarily extract features which provide the most separation between

the clusters. Hence, the clusters which appear inherently separable in some discriminative subspace may overlap if projected using conventional features extraction methods. Such cluster overlaps increases the misclassification, may lead to incorrect detection of the number of the clusters, thus hindering reliable clustering. Therefore, a spike sorting/feature extraction method is desired to seek for features which provide maximum separation between different clusters, and meanwhile be robust to noise and outliers.

1.3 Thesis Objectives

In previous sections two problems including power line interference cancellation and spike sorting were highlighted, and some limitations of current solutions were briefly discussed. The aim of this thesis was to provide methods for improving the quality of neural signals (both LFPs and spike trains) which are widely used in fundamental neuroscience studies, and modern BMIs. Along these lines, the specific objectives of this thesis were to

- Propose a reliable and computationally efficient harmonic estimation algorithm to remove power line interference from neural recordings without compromising the actual neural signals at the interference frequency bands.
- Propose an efficient and scalable VLSI architecture of the aforementioned algorithm that is optimized for removing multiple harmonics from multichannel recordings. And to implement the proposed architecture on a microchip in a 65-nm CMOS technology.
- Propose unsupervised spike sorting algorithms based on discriminative subspace learning to achieve more reliable and accurate identification of neurons and spike waveforms especially in low SNR conditions.

1.4 Overview and Contributions

This section provides an overview of the contributions of this thesis. This thesis contains three chapters of contributions. In the beginning of each chapter, we provided a detailed literature review of the topics discussed in that chapter.

In Chapter 2, we proposed an adaptive algorithm for removing the 50/60 Hz line interference and its harmonics. First, the theory and structure of the algorithm were presented along with the mathematical derivations used to decrease its computational complexity. After that, we provided alternative forms of the algorithm parameters which have intuitive meaning to make the parameter adjustment straightforward, and presented a thorough guide to adjust them. We further discussed the results of extensive simulations that are carried out to quantitatively evaluate the performance of the proposed algorithm under various signal and parameter conditions. The performance of the algorithm was also compared with other popular interference removal methods. A significant portion of this chapter has been presented in [25] and [26].

In Chapter 3, first we extended the interference cancellation algorithm to process multichannel data efficiently. First, brief simulation results on different types of biopotential recording were presented. Further, we proposed an efficient and scalable VLSI architecture of the multichannel version of the algorithm. A microchip implementation in a 65-nm CMOS technology, along with its real-time testing results were also presented. A significant portion of this chapter has been presented in [25] and [27].

In Chapter 4, we introduced two automatic spike sorting algorithms based on discriminative subspace learning. First, we briefly discussed the basics of subspace learning, and presented the formulations. After that, two robust algorithms for spike sorting were presented which can automatically learn the feature space

and the number of the clusters (i.e. neurons). The results of comprehensive simulations using synthetic and real data were presented and compared with several popular spike sorting methods. A significant portion of this chapter has been presented in [28].

Finally, in Chapter 5, we summarized the results of the works presented in this thesis. We also described some of the possible directions of research that are left as open problems for future studies.

In addition to the main chapters, in Appendix A we presented the open source software implementation of the power line interference removal algorithm.

Chapter 2

Power Line Interference Cancellation: Algorithm Design

2.1 Introduction

As briefed in the previous chapter, power line interference may severely corrupt neural recordings at 50/60 Hz and harmonic frequencies. While high signal-to-noise ratio (SNR) (i.e. power of the clean neural signal divided by the power of the interference) is preferred for reliable data analysis, the interference pickup can be severe, degrading the SNR to as low as -20 dB (the interference is 100 times stronger than the signal). This is especially the case in some experiments where the operation of nearby electrical appliances is unavoidable, and the desired recording isolations cannot be obtained [3, 4, 29–31].

For studying field potentials at lower frequencies (e.g. <30 Hz), a low-pass filter is sufficient to reject the power line interference. However, there is an increasing attention to the gamma band oscillations (>30 Hz) due to their correlation with a wide range of cognitive and sensory processes [4–14]. For

example, the frequency bands of 80–500 Hz in [7], 40–180 Hz in [9], 76–150 Hz in [10], 0–200 Hz in [15], and 30–200 Hz in [13] have been shown useful for studying cognitive and motor processing. In this case, in addition to the fundamental harmonic at 50 Hz or 60 Hz, high order harmonics of the interference should also be removed before data analysis.

The interference is usually non-stationary and can vary in frequency, amplitude and phase. The frequency variations are usually small, and mainly originated from the AC power system [32, 33]. Nevertheless, the amplitude and phase variations can be large, which may significantly decrease the SNR of the recorded signal. These variations are mostly due to the subject movements, abrupt changes in nearby AC loads, and changes in capacitive coupling [3, 32, 34]. As a result, automatic cancellation of non-stationary power line interference would be advantageous for reliable data analysis.

A number of solutions are available for reducing the interference pickup. To attenuate the interference at hardware level, biopotential amplifiers are frequently designed to take differential input with large common mode rejection ratio and large isolated-mode rejection ratio. In addition, using active electrodes, shielding electrodes and the subject, and grounding the nearby electrical appliances are useful ways to further reduce the interference [3, 29, 35–37]. Despite these considerations, large residual interference may remain in the signal [3, 4, 20, 31], thus further signal processing is required to completely remove the interference.

Notch filtering has been widely used to attenuate the interference by rejecting its predetermined frequency components (i.e. at 50/60 Hz and harmonic frequencies). To avoid making excessive distortion, the filter should feature narrow notch bandwidth, small phase distortion, and negligible artificial oscillations [18, 34, 38, 39]. However, it is difficult to meet these specifications when the interference frequency is not stable and the filter is to accommodate the frequency

variations. On the one hand, a very narrow notch may lead to an inadequate removal of the interference, especially when its frequency shifts outside the notch bandwidth. On the other hand, a wide notch can attenuated the interference, but it also results in the excessive removal of information-bearing signal components. These reasons have made notch filtering not a good candidate for power line interference removal in neural recording applications [4, 34].

Other techniques based on spectrum estimation have been used for detecting and removing the spectral peaks (thus the interference) [4]. A drawback is that, these methods require buffering a large number of samples, which slows down the signal processing and is not suitable for real-time implementation. Furthermore, they usually lose their effectiveness when the interference is non-stationary [20,39].

Another popular approach is to use adaptive interference cancellation which addresses some of the drawbacks of notch filtering. When an auxiliary reference signal of the interference is available, the well-known adaptive noise canceller (ANC) can be utilized to remove the interference [34, 40, 41]. However, it may become ineffective when the interference contains higher order harmonics. Moreover, a reference signal may not always be available in practice. To address these limitations, several reference-free adaptive methods have been proposed, mainly tailored for ECG signal processing [18–20, 39]. Nevertheless, the performance and reliability of these methods have not been tested on neural recordings. In general, several issues might arise when applying the same algorithms to neural recordings. For example, in some algorithms [18,39], the detection of QRS periods of the ECG signals is necessary to tackle non-stationarity; however, this method is not applicable to neural signals since the on/off period of neural oscillations cannot be easily detected in the presence of the interference. In addition, the power spectral density (PSD) of neural signals follows $1/f^{\alpha}{}_{(1<\alpha<3)}$ distribution [15, 42, 43] which is different from that of the ECG; this might lead to inaccurate

operation of the interference removal algorithms that are specifically tailored for ECG processing.

In this chapter, a robust and computationally efficient algorithm for power line interference cancellation is proposed. It can reliably estimate and remove the 50/60 Hz line interference and its harmonics from neural recordings. The algorithm does not require any reference signal, and can track the variations in the frequency, phase, and amplitude of the interference at both the fundamental and the harmonic frequencies. The algorithm is compared with two adaptive methods of [20] and [18], and simulation results on synthetic and real *in-vivo* data are presented. The algorithm is implemented in software as well as on ASIC. The software implementation is discussed in this chapter, and ASIC implementation is separately detailed in Chapter 3.

The rest of this chapter is organized as follows. Section 2.2 details the proposed algorithm, its pseudocode, and parameter adjustment. Section 2.3 gives the experimental results based on both synthesised and real data, and presents a performance comparison with other methods. Section 2.4 presents the discussion, and Section 4.6 concludes this chapter.

2.2 Proposed Algorithm

A recorded neural signal from one electrode can be represented by

$$x(n) = s(n) + p(n), \quad n \in \mathbb{Z}, \tag{2.1}$$

where $x(n)$ is the measured signal, $s(n)$ is the signal of interest (neural signal + neural noise), and $p(n)$ is the power line interference, all sampled at f_s Hz. $x(n)$ is assumed to be zero-mean, $s(n)$ has a $1/f^{\alpha}{}_{(1<\alpha<3)}$ power spectrum, and $p(n)$ consists of a set of harmonic sinusoidal components with unknown frequencies,

phases and amplitudes as

$$p(n) = \sum_{k=1}^{M} \underbrace{a_k \cos(k\omega_{\mathrm{f}} n + \phi_k)}_{h_k(n)} = \sum_{k=1}^{M} h_k(n). \tag{2.2}$$

Here, ω_{f} is the fundamental frequency in rad/s, a_k and ϕ_k are the amplitude and phase of the k^{th} harmonic, and M is the number of harmonics present in the interference.

An ideal interference cancellation algorithm should eliminate the interference $p(n)$, while perfectly preserving the neural signal $s(n)$. Let $\hat{\omega}_{\mathrm{f}}$, $\hat{a}_k$, $\hat{\phi}_k$, $\hat{h}_k(n)$, and $\hat{p}(n)$ denote the estimate of ω_{f}, a_k, ϕ_k, $h_k(n)$, and $p(n)$, respectively. The clean (i.e. interference-free) signal $\hat{s}(n)$ is obtained as

$$\hat{s}(n) = x(n) - \hat{p}(n), \tag{2.3a}$$

where

$$\hat{p}(n) = \sum_{k=1}^{M'} \hat{h}_k(n). \tag{2.3b}$$

Here, M' represents the desired number of harmonics to be removed from the recorded signal. It is chosen based on the bandwidth of interest, and its maximum value $M'_{\max} = \lfloor \pi/\hat{\omega}_{\mathrm{f}} \rfloor$ can be adopted if it is desired to remove all the harmonics up to the Nyquist frequency.

The following approach is proposed to cancel the interference. First, the interference fundamental frequency ω_{f} is estimated by using a fast and numerically well-behaved frequency estimator. Subsequently, based on the estimated frequency $\hat{\omega}_{\mathrm{f}}$, each harmonic signal $h_k(n)$ is obtained by using discrete-time oscillators and then its amplitude and phase (i.e. $\hat{a}_k$ and $\hat{\phi}_k$, respectively) are estimated by using a simplified recursive least squares (RLS) algorithm. The cascaded stages of frequency and amplitude/phase estimation allow individually adjustable

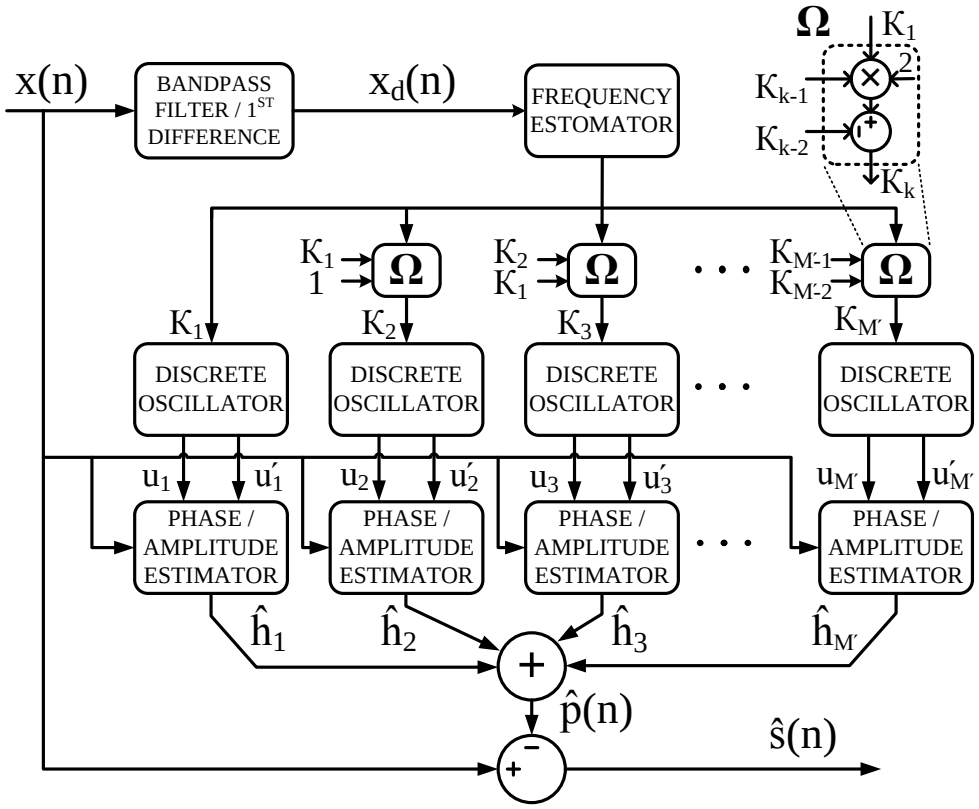

Figure 2.1: Functional block diagram of the proposed algorithm. $x(n)$ is the input signal contaminated by power line interference, $\hat{p}(n)$ is the estimated interference, and $\hat{s}(n)$ is the output interference-free signal.

adaptation rates for each of these estimators, which helps to achieve a fast and reliable estimation of the interference. Finally, the estimated interference $\hat{p}(n)$ is subtracted from the input signal $x(n)$ to obtain the clean signal $\hat{s}(n)$. The structure of the proposed algorithm is shown in Figure 2.1.

2.2.1 Fundamental Frequency Estimation

For robust estimation of the fundamental frequency, first, the signal is preprocessed to enhance the fundamental harmonic of the interference. After that, the enhanced signal is used for frequency estimation. The preprocessing stage is described in Section 2.2.1, followed by the frequency estimation stage in Section 2.2.1.

Initial Band-pass Filtering and Spectrum Shaping

Since the input signal $x(n)$ has a coloured PSD $(1/f)$, the direct application of a typical adaptive frequency estimator would lead to a biased estimation of the frequency [44]. It is also possible that the interference $p(n)$ is more dominant at certain harmonic frequencies than at the fundamental frequency; this might be due to operation of nearby electrical appliances or the amplifier distortion. This may prevent the frequency estimator from converging to a correct frequency estimate. To address these issues and improve the frequency estimation, the input signal $x(n)$ is bandpass filtered with a 4^{th}-order infinite-impulse-response (IIR) filter to enhance the fundamental harmonic of the interference and attenuate higher harmonics. This filtering is also useful for attenuating lower frequency artefacts and signal components which may negatively affect the frequency estimation. The filter passband is by default set to 40–70 Hz to accommodate both 50 Hz and 60 Hz power line frequencies and their worst case variations, but it can be further customized; for example, to 55–65 Hz if the nominal power line frequency is known to be 60 Hz. Let $H(\cdot)$ be the realization of the bandpass filter, the filtered signal $x_{\text{f}}(n)$ is obtained as

$$x_{\text{f}}(n) = H(x(n)). \tag{2.4a}$$

To further reduce the estimation bias, a 1^{st}-order differentiator is utilized which mitigates the effect of the power law spectrum of the input signal:

$$x_{\text{d}}(n) = x_{\text{f}}(n) - x_{\text{f}}(n-1). \tag{2.4b}$$

Here, $x_{\text{d}}(n)$ is the first difference signal fed into the next stage for frequency estimation. The effect of 1^{st}-order differentiation on the overall performance

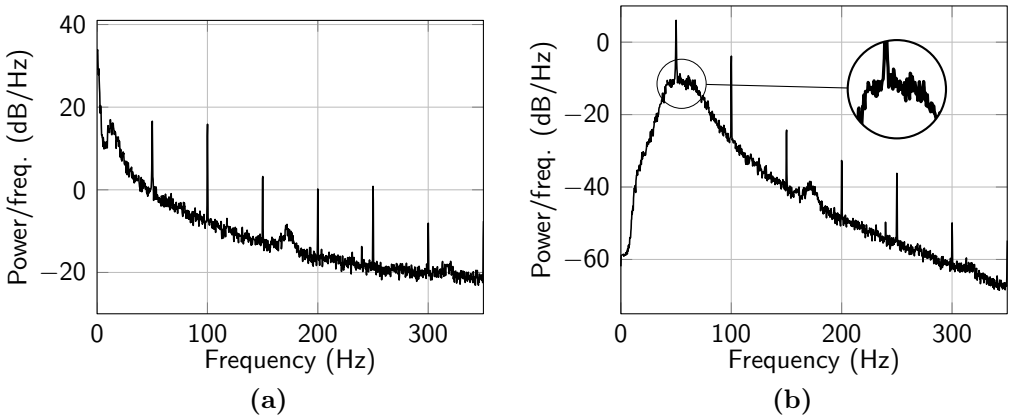

Figure 2.2: The effect of bandpass filtering and spectrum shaping. (a) PSD of a real ECoG signal. (b) PSD after bandpass filtering and spectrum shaping, where the fundamental harmonic is enhanced.

of the algorithm is not very significant; however, in practice, the first order differentiator can be incorporated into the bandpass filter with negligible computational overhead. Figure 2.2 shows the effects of bandpass filtering and spectrum shaping, where the fundamental harmonic of the interference is enhanced. It should be noted that signal x_{d} is only used for frequency estimation, and not for amplitude/phase estimation.

Frequency Estimation

The estimation of the instantaneous frequency of a single sinusoid buried in broadband noise has been largely investigated in the literature. Various well-established methods exist for frequency estimation differing in performance with regard to computational complexity, and estimation bias and variance [45–49]. In this work, a lattice adaptive notch filter (ANF)-based frequency estimator is utilized since it features instantaneous estimation of the frequency, desirable performance, low complexity, and suitability for real-time finite-precision implementation.

It should be noted that the ANF is merely used for frequency estimation and

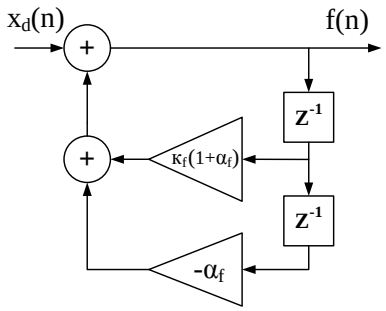

Figure 2.3: Signal flow graph of the all-pole lattice ANF structure. Notch frequency and bandwidth are determined by κ_{f} and α_{f}, respectively.

not for notch filtering, hence the all-zero section is not needed. Figure 2.3 shows the structure of the ANF where $x_d(n)$ is the input signal from the preprocessing stage and $f(n)$ is the output of the all-pole section. The transfer function of the all-pole section is given by

$$T(z) = \frac{1}{1 - \kappa_{\mathrm{f}}(n)(\alpha_{\mathrm{f}} + 1)z^{-1} + \alpha_{\mathrm{f}} z^{-2}}, \tag{2.5}$$

where $\kappa_{\mathrm{f}}(n)$ is the adaptive coefficient at time step n, which gives the frequency estimate $\hat{\omega}_{\mathrm{f}}(n)$ through $\hat{\omega}_{\mathrm{f}}(n) = \cos^{-1}\kappa_{\mathrm{f}}(n)$, and $0 < \alpha_{\mathrm{f}} < 1$ is the pole radii and determines the notch bandwidth. The lattice algorithm of [48] is employed to adjust κ_{f} as follows.

$$c(0) = d(0) = \epsilon > 0,\ f(-1) = f(-2) = 0,\ \kappa_{\mathrm{f}}(0) = 0, \tag{2.6a}$$

$$c(n) = \lambda_{\mathrm{f}} c(n-1) + f(n-1)(f(n) + f(n-2)), \tag{2.6b}$$

$$d(n) = \lambda_{\mathrm{f}} d(n-1) + 2f(n-1)^2, \tag{2.6c}$$

$$\kappa_{\mathrm{t}}(n) = \frac{c(n)}{d(n)}, \tag{2.6d}$$

$$\kappa_{\mathrm{t}}(n) = \begin{cases} \kappa_{\mathrm{t}}(n) & \text{if } -1 < \kappa_{\mathrm{t}}(n) < 1, \\ 1 & \text{if } \kappa_{\mathrm{t}}(n) > 1, \\ -1 & \text{if } \kappa_{\mathrm{t}}(n) < -1, \end{cases} \tag{2.6e}$$

$$\kappa_{\mathrm{f}}(n) = \gamma\kappa_{\mathrm{f}}(n-1) + (1-\gamma)\kappa_{\mathrm{t}}(n), \tag{2.6f}$$

where $0 \ll \lambda_{\mathrm{f}} < 1$ is the forgetting factor, $f(n)$ is the output of the all-pole section, $\kappa_{\mathrm{f}}(n)$ is the estimated parameter ($\hat{\omega}_{\mathrm{f}}(n) = \cos^{-1}\kappa_{\mathrm{f}}(n)$), and γ is the smoothing factor. Equation (2.6a) sets the initial condition, (2.6b)–(2.6d) form the frequency estimator, and (2.6e) is used to limit κ_{t} in the range of $[-1, 1]$ to guarantee stability. (2.6f) is used to further smooth $\kappa_{\mathrm{f}}(n)$. For simplicity in notation, in the rest of this chapter, κ_{f} is short for $\kappa_{\mathrm{f}}(n)$ and $\hat{\omega}_{\mathrm{f}}$ is short for $\hat{\omega}_{\mathrm{f}}(n)$.

The parameters α_{f} and λ_{f} control the speed and accuracy of frequency estimation. It is advantageous to use time-varying values for α_{f} and λ_{f} due to several reasons. In initial convergence, if the notch is too narrow (α_{f} very close to 1), the ANF may not sense the presence of the input sinusoid, which in turn leads to a very slow initial convergence or even not converging to the correct frequency estimate. Similarly, an initial value of λ_{f} very close to 1, significantly slows down the initial adaptation. On the other hand, smaller values of α_{f} and λ_{f} increase the steady-state error. A solution is to start the algorithm with smaller values of α_{f} and λ_{f} to reach a fast convergence, and after that gradually increase their values to obtain more accurate frequency estimation. For this purpose, α_{f} and λ_{f} are updated in each iteration as

$$\alpha_{\mathrm{f}}(n) = \alpha_{\mathrm{st}}\alpha_{\mathrm{f}}(n-1) + (1-\alpha_{\mathrm{st}})\alpha_{\infty}, \tag{2.7a}$$

$$\lambda_{\mathrm{f}}(n) = \lambda_{\mathrm{st}}\lambda_{\mathrm{f}}(n-1) + (1-\lambda_{\mathrm{st}})\lambda_{\infty}, \tag{2.7b}$$

where α_∞ determines the asymptotic notch bandwidth and α_{st} sets the rate of change from the initial value $\alpha_{\text{f}}(0) = \alpha_0$ to the asymptotic value α_∞. Similarly, λ_∞ determines the asymptotic forgetting factor and λ_{st} sets the rate of change from initial value $\lambda_{\text{f}}(0) = \lambda_0$ to the asymptotic value λ_∞. Detailed discussion on choosing proper values for the parameters are presented in Section 2.2.4

2.2.2 Harmonic Estimation

Having estimated κ_{f}, the algorithm proceeds to estimate the harmonic components. Harmonic estimation comprises two stages. First, a series of harmonic sinusoids with fundamental frequency $\hat{\omega}_{\text{f}}$ are generated. Subsequently, the amplitudes and the phases of the generated harmonics are estimated to match their corresponding components in the interference. In this section, a description of harmonic generation followed by amplitude/phase estimation is presented.

Harmonic Signal Generation

The harmonic sinusoids are generated through using discrete-time oscillators, which require less computation compared with the Taylor expansion method [50]. Among different oscillator structures, a digital waveguide oscillator is chosen (Figure 2.4(a)). This structure provides orthogonal outputs, which are exploited to simplify the next stage RLS algorithm. More importantly, the oscillator output frequency can be directly controlled by $\cos k\hat{\omega}_{\text{f}}$, where $k\hat{\omega}_{\text{f}}$ is the oscillation frequency. This enables the output of the frequency estimator κ_{f} to be directly employed for harmonic generation, thus avoiding the calculation of computationally expensive trigonometric functions. To further reduce the complexity, the frequency estimates of higher harmonics are obtained through the recurrence formulation in (2.8), which also avoid trigonometric function calculation. For each harmonic k, the frequency control parameter of the oscillator

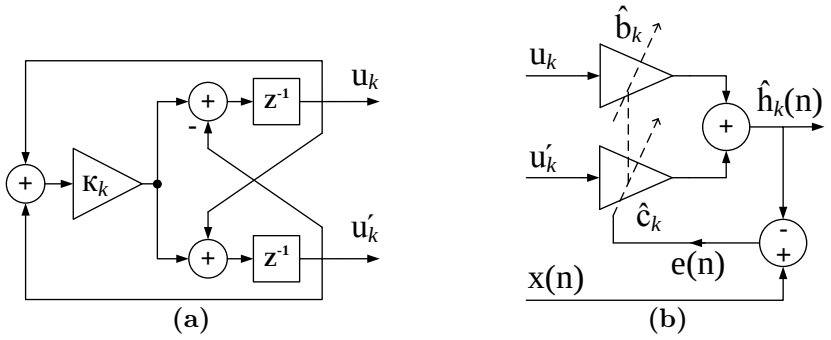

Figure 2.4: Signal flow graph of (a) Discrete-time oscillator. The parameter κ_k adjusts the oscillation frequency. u_k and u'_k represent orthogonal sinusoids at frequency $k\omega_\mathrm{f}$. (b) Adaptive linear combiner, used for amplitude/phase adaptation. The weights a_k and b_k are adapted by the simplified RLS algorithm which minimizes the weighted least square error between $x(n)$ and $\hat{h}_k(n)$.

is denoted as $\kappa_k = \cos k\hat{\omega}_\mathrm{f}$, and is recursively calculated through

$$\kappa_k = 2\kappa_1\kappa_{k-1} - \kappa_{k-2}, \quad \text{for} \quad k = 2, 3, \cdots, M', \tag{2.8a}$$

where

$$\kappa_0 = 1, \kappa_1 = \kappa_\mathrm{f} = \cos\hat{\omega}_\mathrm{f}. \tag{2.8b}$$

The calculated parameter κ_k is used to set the oscillation frequency of the oscillator.

Figure 2.4(a) shows the signal flow graph of the digital waveguide oscillator, whose characteristic function is represented by (2.9a):

$$\begin{bmatrix} u_k(n) \\ u'_k(n) \end{bmatrix} = \begin{bmatrix} \kappa_k & \kappa_k - 1 \\ \kappa_k + 1 & \kappa_k \end{bmatrix} \begin{bmatrix} u_k(n-1) \\ u'_k(n-1) \end{bmatrix}, \tag{2.9a}$$

$$G = 1.5 - \left(u_k(n)^2 - \frac{\kappa_k - 1}{\kappa_k + 1} u'_k(n)^2 \right), \tag{2.9b}$$

$$u_k(n) = Gu_k(n), \quad u'_k(n) = Gu'_k(n). \tag{2.9c}$$

Here, $u(n)$ and $u'(n)$ are state variables serving as sinusoidal outputs. The values of $u_k(0)$ and $u'_k(0)$ determine the initial phase and amplitude, which are arbitrarily chosen. (2.9b) and (2.9c) are used to apply gain control for stabilizing oscillation amplitude in dynamic frequency operation. The output of (2.9) can be generally expressed as

$$u_k(n) = v_k \sin(k\hat{\omega}_{\mathrm{f}} n + \psi_k), \qquad u'_k(n) = v'_k \cos(k\hat{\omega}_{\mathrm{f}} n + \psi_k), \tag{2.10}$$

where v_k and v'_k are the amplitudes of the generated sinusoids, and ψ_k is the initial phase shift. The values of ψ_ks do not influence any further derivations and are neglected for simplicity.

Amplitude and Phase Estimation

The amplitudes and phases (i.e. $\hat{a}_k$ and $\hat{\phi}_k$) of the generated harmonics are not necessarily the same with their corresponding power line interference components in (2.2); thus, an additional step is required to estimate them. The estimate of the k^{th} harmonic, $\hat{h}_k(n)$, can be obtained via (2.2) by substituting a_k and ϕ_k with their estimates that gives

$$\hat{h}_k(n) = \hat{a}_k \sin(k\hat{\omega}_{\mathrm{f}} n + \hat{\phi}_k) \tag{2.11a}$$

$$= \hat{b}'_k \sin(k\hat{\omega}_{\mathrm{f}} n) + \hat{c}'_k \cos(k\hat{\omega}_{\mathrm{f}} n). \tag{2.11b}$$

where

$$\hat{b}'_k = \hat{a}_k \cos\hat{\phi}_k, \quad \text{and} \quad \hat{c}'_k = \hat{a}_k \sin\hat{\phi}_k.$$

Here, instead of directly adapting $\hat{a}_k$ and $\hat{\phi}_k$ in (2.11a) we can equivalently adapt $\hat{b}'_k$ and $\hat{c}'_k$ in (2.11b) to obtain $\hat{h}_k(n)$. This transformation converts the non-convex search space in $\hat{a}_k$-$\hat{\phi}_k$ coordinates into a convex search space in

rectangular coordinates. Using (2.10) and (2.11b), $\hat{h}_k(n)$ can be written as

$$\hat{h}_k(n) = \hat{b}_k u_k(n) + \hat{c}_k u'_k(n). \tag{2.12}$$

Here, $\hat{b}_k$ and $\hat{c}_k$ are defined as $\hat{b}'_k/v_k$ and $\hat{c}'_k/v'_k$, where v_k and v'_k merely scale the adaptive coefficients and do not affect the estimation performance. For each harmonic k, $\hat{b}_k$ and $\hat{c}_k$ are adapted by minimizing the exponentially weighted squared error between $\hat{h}_k(n)$ and $x(n)$. This is done by applying the simplified RLS algorithm, where $u_k(n)$ and $u'_k(n)$ serve as the input to an adaptive linear combiner (Figure 2.4(b)). A detailed description of the simplified RLS algorithm is followed in the next section. The resultant update equations to adapt $\hat{b}_k$ and $\hat{c}_k$ are summarized as

$$\begin{aligned}
r_{1,k}(-1) = r_{1,k}(-1) = \hat{b}_k(-1) = \hat{c}_k(-1) = 0,\\
r_{1,k}(n) = \lambda_{\mathrm{a}} r_{1,k}(n-1) + u_k(n)^2,\\
r_{4,k}(n) = \lambda_{\mathrm{a}} r_{4,k}(n-1) + u'_k(n)^2,\\
\hat{b}_k(n) = \hat{b}_k(n-1) + u_k(n)e_k(n)/r_{1,k}(n),\\
\hat{c}_k(n) = \hat{c}_k(n-1) + u'_k(n)e_k(n)/r_{4,k}(n),
\end{aligned} \tag{2.13}$$

where $e_k(n) = x(n) - \hat{h}_k(n)$ is the instantaneous error (Figure 2.4(b)), and $0 \ll \lambda_{\mathrm{a}} < 1$ is the forgetting factor.

In each iteration, the most recent estimates $\hat{b}_k(n)$ and $\hat{c}_k(n)$ are used to obtain $\hat{h}_k(n)$ through (2.12). The interference-free neural signal is then obtained by

$$\hat{s}(n) = x(n) - \sum_{k=1}^{M'} \hat{h}_k(n). \tag{2.14}$$

RLS algorithm

The RLS algorithm is used to adapt the weights of the adaptive linear combiner in Figure 2.4(b). The weighted least squares cost function is defined as

$$E_k = \sum_{i=0}^{n} \lambda_{\mathrm{a}}^{n-i} e_k^2(i), \tag{2.15a}$$

where

$$e_k(i) = x(i) - \hat{h}_k(i), \tag{2.15b}$$

is the instantaneous error and $0 < \lambda_{\mathrm{a}} \ll 1$ is the forgetting factor. The RLS algorithm is described as follows. Let $\mathbf{U}_k(n) = [u_k(n)\ u'_k(n)]^{\mathrm{T}}$ be the input sample vector and $\mathbf{W}_k(n) = [\hat{b}_k(n)\ \hat{c}_k(n)]^{\mathrm{T}}$ be the parameter vector. The input sample correlation matrix is defined as

$$\mathbf{R}_k(n) = \sum_{i=0}^{n} \lambda_{\mathrm{a}}^{n-i} \mathbf{U}_k(n) \mathbf{U}_k{}^{T}(n) = \begin{bmatrix} r_{1,k}(n) & r_{2,k}(n) \\ r_{3,k}(n) & r_{4,k}(n) \end{bmatrix}. \tag{2.16a}$$

$\mathbf{R}_k(n)$ can be recursively calculated through

$$\begin{aligned} &\mathbf{R}_k(0) = \epsilon I_2, \epsilon > 0 \quad \text{for } k = 1, \cdots, M', \\ &\mathbf{R}_k(n) = \lambda_{\mathrm{a}} \mathbf{R}_k(n-1) + \mathbf{U}_k(n) \mathbf{U}_k^{\mathrm{T}}(n). \end{aligned} \tag{2.16b}$$

Now, $\mathbf{W}_k$ can be adapted through the following RLS update equation [51].

$$\mathbf{W}_k(n+1) = \mathbf{W}_k(n) - \mathbf{R}_k^{-1} \mathbf{U}_k(n) e_k(n). \tag{2.16c}$$

In the standard RLS method, matrix inversion lemma is used to obtain $\mathbf{R}_k^{-1}$ in order to avoid inverse matrix calculation. In this work, we suggest a simplification on $\mathbf{R}_k$ which leads to a less computational parameter adaptation. It is assumed that the forgetting factor parameter λ_{a} is selected close to the recommended values (i.e $0.5 < W < 5$). In this case, it is shown in the following section that $r_{2,k}$ and $r_{3,k}$ would become very small compared with $r_{1,k}$ and $r_{4,k}$, thus they can be neglected and $\mathbf{R}_k$ becomes a diagonal matrix. This leads to simplified update equations as

$$\begin{aligned} r_{1,k}(-1) &= r_{1,k}(-1) = \hat{b}_k(-1) = \hat{c}_k(-1) = 0, \\ r_{1,k}(n) &= \lambda_{\mathrm{a}} r_{1,k}(n-1) + u_k(n)^2, \\ r_{4,k}(n) &= \lambda_{\mathrm{a}} r_{4,k}(n-1) + u'_k(n)^2, \\ \hat{b}_k(n) &= \hat{b}_k(n-1) + u_k(n)e_k(n)/r_{1,k}(n), \\ \hat{c}_k(n) &= \hat{c}_k(n-1) + u'_k(n)e_k(n)/r_{4,k}(n). \end{aligned} \tag{2.17}$$

These update equations are much simpler than that of (2.16c) with regards to the number of arithmetic operations. We also investigated the effect of this simplification on the performance of the algorithm by comparing the results with the case that the standard RLS algorithm was used, and no significant difference in performance was observed, which validates the proposed approximation.

Simplification of the RLS algorithm

We propose a simplification on the RLS algorithm used in the phase and amplitude estimation stage of the algorithm. This simplification is based on approximating the RLS sample correlation matrix $\mathbf{R}_k(n)$ in (2.16a) with a diagonal matrix. The

elements of $\mathbf{R}_k(n)$ can be expanded as

$$r_{1,k} = \sum_{i=0}^{n} \lambda_{\mathrm{a}}^{n-i} v^2 \sin^2(k\omega_{\mathrm{f}} i), \tag{2.18a}$$

$$r_{4,k} = \sum_{i=0}^{n} \lambda_{\mathrm{a}}^{n-i} v'^2 \cos^2(k\omega_{\mathrm{f}} i), \tag{2.18b}$$

$$r_{2,k} = r_{3,k} = \sum_{i=0}^{n} \lambda_{\mathrm{a}}^{n-i} v v' \sin(k\omega_{\mathrm{f}} i)\cos(k\omega_{\mathrm{f}} i). \tag{2.18c}$$

If it can be shown that for typical parameters values and sampling rates the coefficients $|r_{2,k}|$ and $|r_{3,k}|$ are much less than $|r_{1,k}|$ and $|r_{4,k}|$, then the sample correlation matrix $R_k(n)$ can be well-approximated by a diagonal matrix (i.e. $r_{2,k}$= $r_{3,k}$=0). This subsequently leads to much less computational RLS update equations. In the following derivations, we first show that $|r_{2,k}|, |r_{3,k}| \ll r_{1,k}$; the inequality $|r_{2,k}|, |r_{3,k}| \ll r_{4,k}$ can be similarly derived and is not presented here. We would like to show that the following inequality holds for typical parameter values:

$$\left| \sum_{i=0}^{n} \lambda_{\mathrm{a}}^{n-i} v v' \sin(k\omega_{\mathrm{f}} i)\cos(k\omega_{\mathrm{f}} i) \right| \ll \sum_{i=0}^{n} \lambda_{\mathrm{a}}^{n-i} v^2 \sin^2(k\omega_{\mathrm{f}} i), \tag{2.19a}$$

or equivalently

$$\frac{v v'}{v^2} \frac{\left| \sum_{i=0}^{n} \lambda_{\mathrm{a}}^{n-i} \sin(k\omega_{\mathrm{f}} i)\cos(k\omega_{\mathrm{f}} i) \right|}{\sum_{i=0}^{n} \lambda_{\mathrm{a}}^{n-i} v^2 \sin^2(k\omega_{\mathrm{f}} i)} \ll 1. \tag{2.19b}$$

Note that the right-hand side of (2.19a) is always positive and equal to its absolute value. The magnitudes v and v' are not initially included for the following derivation and will be considered in the last stage. After a little

manipulation of (2.19a) using trigonometric identities we get

$$\frac{\sqrt{2}}{2}\left|\sum_{i=0}^{n}\lambda_{\mathrm{a}}^{n-i}\sin(2k\omega_{\mathrm{f}}i+\frac{\pi}{4})\right| \ll \frac{1}{2}\sum_{i=0}^{n}\lambda_{\mathrm{a}}^{n-i} \tag{2.20}$$

which can be expressed in the complex domain as

$$\sqrt{2}\cdot\left|\mathcal{I}m\left\{e^{\mathbf{j}\frac{\pi}{4}}\frac{\lambda_{\mathrm{a}}^{n+1}-e^{\mathbf{j}2k\omega_{\mathrm{f}}(n+1)}}{\lambda_{\mathrm{a}}-e^{\mathbf{j}2k\omega_{\mathrm{f}}}}\right\}\right| \ll \frac{\lambda_{\mathrm{a}}^{n+1}-1}{\lambda_{\mathrm{a}}-1}. \tag{2.21}$$

Using the property $|\mathcal{I}m\{z\}| \leq |z|$, where z is a complex number, we can alternatively show that

$$\sqrt{2}\cdot\left|e^{\mathbf{j}\frac{\pi}{4}}\frac{\lambda_{\mathrm{a}}^{n+1}-e^{\mathbf{j}2k\omega_{\mathrm{f}}(n+1)}}{\lambda_{\mathrm{a}}-e^{\mathbf{j}2k\omega_{\mathrm{f}}}}\right| =$$

$$\left[2\frac{1+\lambda_{\mathrm{a}}^{2(n+1)}-2\lambda_{\mathrm{a}}^{n+1}\cos(2k\omega_{\mathrm{f}}(n+1))}{1+\lambda_{\mathrm{a}}^{2}-2\lambda_{\mathrm{a}}\cos(2k\omega_{\mathrm{f}})}\right]^{\frac{1}{2}} \ll \frac{\lambda_{\mathrm{a}}^{n+1}-1}{\lambda_{\mathrm{a}}-1}. \tag{2.22}$$

We define

$$\mathcal{A} = \frac{\left[2\frac{1+\lambda_{\mathrm{a}}^{2(n+1)}-2\lambda_{\mathrm{a}}^{n+1}\cos(2k\omega_{\mathrm{f}}(n+1))}{1+\lambda_{\mathrm{a}}^{2}-2\lambda_{\mathrm{a}}\cos(2\omega_{\mathrm{f}})}\right]^{\frac{1}{2}}}{\frac{\lambda_{\mathrm{a}}^{n+1}-1}{\lambda_{\mathrm{a}}-1}}, \tag{2.23}$$

$$\mathcal{B} = \max\{\frac{v}{v'},\frac{v'}{v}\}, \tag{2.24}$$

$$\mathcal{C} = \max_{k,n}\mathcal{A}\cdot\mathcal{B}, \tag{2.25}$$

where v/v' is given by (see [50])

$$\frac{v}{v'} = \sqrt{\frac{1+\cos(k\omega_{\mathrm{f}})}{1-\cos(k\omega_{\mathrm{f}})}}. \tag{2.26}$$

If we can show that $\mathcal{C} \ll 1$, then (2.19b) holds. To show this, numerical

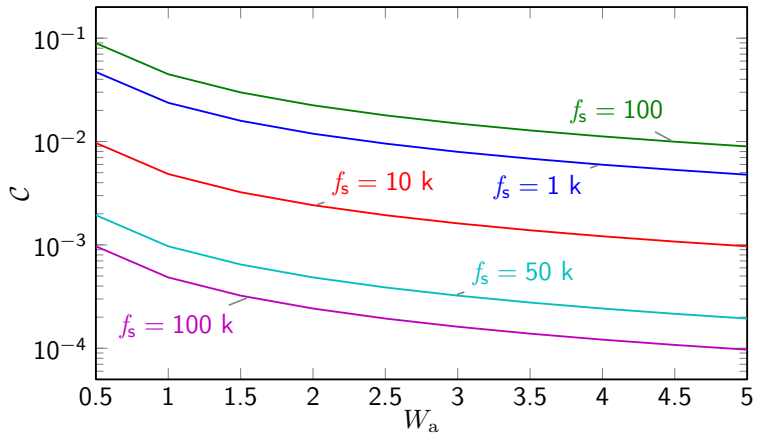

Figure 2.5: The values of $\mathcal{C}$ at different sampling rates. It can be seen that at the minimum sampling rate $f_s = 100$, the maximum value of $\mathcal{C}$ is less than $0.1 \ll 1$. The value of $\mathcal{C}$ monotonously decreases with increasing f_s and W_a

simulation is used to obtain the upper bound values on all the harmonics and in the operating condition where $n \gg 1$. For this purpose, the values of λ_a are obtained from W_a through (2.29a), where W_a is swept in the range of 0.5–5 which covers the recommended range. Figure 2.5 displays the values of $\mathcal{C}$ in different sampling rates. As can be seen, in all the conditions, $\mathcal{C} < 0.1 \ll 1$ indicating that the inequalities $|r_{2,k}|, |r_{3,k}| \ll r_{1,k}$ and $|r_{2,k}|, |r_{3,k}| \ll r_{4,k}$ hold.

2.2.3 Algorithm Implementation

The algorithm is implemented in software and hardware. The pseudocode of the algorithm is presented in algorithm 2.1 with the MATLAB source code available online at [52]. The proper parameter values can be obtained through the guidelines in Section 2.2.4. Hardware implementation and testing results are thoroughly discussed in Chapter 3.

2.2.4 Parameter Setting

The performance of the algorithm is mainly controlled by three basic parameters including notch filter pole radii (α_f), frequency estimator's forgetting factor

Algorithm 2.1: Proposed Algorithm

Input: x
Output: $\hat{s}$
Constants:
f_s, M', N, α_0, α_st, α_∞, λ_0, λ_st, λ_∞, λ_a, γ
$H(\cdot) \leftarrow$ 40–70 Hz IIR filter
Initialization:
$\kappa_0 \leftarrow 1$, $\kappa_\mathrm{f} \leftarrow 0$
$f_{-2} \leftarrow f_{-1} \leftarrow 0$
$c, d > 0$
$u_k, u'_k > 0$
$r_{1,k}, r_{4,k} > 0$
$\hat{b}_k \leftarrow \hat{c}_k \leftarrow 0$
$\alpha_\mathrm{f} \leftarrow \alpha_0$, $\lambda_\mathrm{f} \leftarrow \lambda_0$
Recursion:
for $n \leftarrow 1$ ***to*** N **do**
  Bandpass filtering:
  $x_f \leftarrow H(x(n))$
  Frequency Estimation:
  $f_n \leftarrow x_f + \kappa_\mathrm{f}(1+\alpha_\mathrm{f})f_{n-1} - \alpha_\mathrm{f} f_{n-2}$
  $c \leftarrow \lambda_\mathrm{f} c + f_{n-1}(f_n + f_{n-2})$
  $d \leftarrow \lambda_\mathrm{f} d + 2f_{n-1}^2$
  $\kappa_\mathrm{t} \leftarrow c/d$
  if $\kappa_\mathrm{t} > 1$ **then** $\kappa_\mathrm{t} \leftarrow 1$
  else if $\kappa_\mathrm{t} < -1$ **then** $\kappa_\mathrm{t} \leftarrow -1$
  $\kappa_\mathrm{f} \leftarrow \gamma\kappa_\mathrm{f} + (1-\gamma)\kappa_\mathrm{t}$
  $\alpha_\mathrm{f} \leftarrow \alpha_\mathrm{st}\alpha_\mathrm{f} + (1-\alpha_\mathrm{st})\alpha_\infty$
  $\lambda_\mathrm{f} \leftarrow \lambda_\mathrm{st}\lambda_\mathrm{f} + (1-\lambda_\mathrm{st})\lambda_\infty$
  Removing Harmonics:
  $\kappa_1 \leftarrow \kappa_\mathrm{f}$
  $e \leftarrow x(n)$
  for $k \leftarrow 1$ ***to*** M' **do**
    Discrete Oscillator:
    $s_1 \leftarrow \kappa_k(u_k + u'_k)$
    $s_2 \leftarrow u_k$
    $u_k \leftarrow s_1 - u'_k$
    $u'_k \leftarrow s_1 + s_2$
    $G \leftarrow 1.5 - [u_k^2 - u_k'^2(\kappa_k - 1)/(\kappa_k + 1)]$
    if $G < 0$ **then** $G \leftarrow 1$
    $u_k = Gu_k$, $u'_k = Gu'_k$
    Amplitude/Phase Estimation:
    $h_k \leftarrow (\hat{b}_k u_k + \hat{c}_k u'_k)$
    $e \leftarrow e - h_k$
    $r_{1,k} \leftarrow \lambda_\mathrm{a} r_{1,k} + u_k^2$
    $r_{4,k} \leftarrow \lambda_\mathrm{a} r_{4,k} + u_k'^2$
    $\hat{b}_k \leftarrow \hat{b}_k + e \cdot u_k / r_{1,k}$
    $\hat{c}_k \leftarrow \hat{c}_k + e \cdot u'_k / r_{4,k}$
    Harmonic Frequency Calculation:
    $\kappa_{k+1} \leftarrow 2\kappa_\mathrm{f}\kappa_k - \kappa_{k-1}$
  $\hat{s}(n) \leftarrow e$

(λ_{f}) and amplitude/phase estimator's forgetting factor (λ_{a}). Since, the proper values of these parameters depend on the signal sampling rate (f_{s}), the parameter adjustment become less intuitive. To alleviate this issue, we have chosen other representative characteristics such as notch bandwidth (related to the pole radii) and settling time (related to the forgetting factors) which can be alternatively used for parameter adjustment.

Here, we describe the relation between forgetting factors and settling time as well as the relation between pole radii and notch bandwidth. The proper values of the forgetting factors depend on the sampling rate, making it difficult to adjust their values in general condition. On the other hand, settling time is independent of the sampling rate and has a more intuitive meaning that makes the parameter tuning straightforward. In certain parts of the algorithm such as frequency estimation and phase/amplitude adaptation, settling time can be associated with the forgetting factor by the following formulation

$$0.95\frac{1}{1-\lambda} = \frac{1-\lambda^{n_{\mathrm{set}}+1}}{1-\lambda} \Rightarrow \lambda = \exp\frac{\ln(0.05)}{t_{\mathrm{set}}f_{\mathrm{s}}+1}, \tag{2.27}$$

where $n_{\mathrm{set}} = f_{\mathrm{s}}t_{\mathrm{set}}$, λ is the forgetting factor, t_{set} is the desired settling time, and f_{s} is the sampling rate. The transformation of (2.27) is used in (2.29a) to adjust α_{st}, λ_0, λ_{st}, λ_∞, and λ_{a}.

Notch bandwidth is independent of the sampling rate and can be alternatively adjusted instead of the pole radii. Given the notch bandwidth B, the pole radii α is obtained by

$$\alpha = \frac{1-\tan\pi B/f_{\mathrm{s}}}{1+\tan\pi B/f_{\mathrm{s}}}. \tag{2.28}$$

Defining four sets of variables $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{B}_1$ and $\mathcal{B}_2$ as

$$\mathcal{A}_1 = \{\alpha_{\text{st}}, \lambda_0, \lambda_\infty, \lambda_{\text{st}}, \lambda_{\text{a}}\}, \qquad \mathcal{B}_1 = \{B_{\text{st}}, P_0, P_\infty, P_{\text{st}}, W_{\text{a}}\},$$

$$\mathcal{A}_2 = \{\alpha_0, \alpha_\infty, \gamma'\}, \qquad \mathcal{B}_2 = \{B_0, B_\infty, \gamma/2\},$$

the alternative parameters are then obtained through

$$\mathcal{A}_1 = \exp \frac{\ln(0.05)}{\mathcal{B}_1 f_{\text{s}} + 1}, \tag{2.29a}$$

$$\mathcal{A}_2 = \frac{1 - \tan(\pi \mathcal{B}_2 / f_{\text{s}})}{1 + \tan(\pi \mathcal{B}_2 / f_{\text{s}})}. \tag{2.29b}$$

Here, $\mathcal{B}_1$ and $\mathcal{B}_2$ contain the alternative parameters which are independent of the sampling rate and have intuitive units. The actual parameters, defined in $\mathcal{A}_1$ and $\mathcal{A}_2$, can be obtained through (2.29b). It should be noted that improper parameter setting may lead to inadequate removal of the interference. Some guidelines on the proper adjustment of the parameters are discussed as follows.

The notch bandwidth of the frequency estimator affects both the tracking speed and the estimation bias. A wide notch allows faster tracking of the frequency at the expense of an increased estimation bias and variance. On the other hand, a narrow notch leads to a more accurate frequency estimate, but it causes very slow frequency adaptation if the desired sinusoidal component falls out of the notch bandwidth. To address this trade-off, the notch bandwidth is initially widened to allow fast initial convergence and then gradually narrowed down to achieve a lower steady-state error (described in (2.7a)). In the alternative form, B_0 is associated with α_0 and controls the initial notch bandwidth. Larger values of B_0 are preferred (e.g. tens of Hz) to achieve a faster initial convergence. Similarly, B_∞ is associated with α_∞ and controls the asymptotic notch bandwidth. Small values of B_∞ are preferred (e.g. tenths of Hz) to achieve more accurate

estimation of the frequency. B_{st} controls the rate of transition between initial notch bandwidth B_0 and the asymptotic notch bandwidth B_∞, and indicates the time in seconds, in which α_{f} reaches $0.95\alpha_\infty$ in (2.7a). When the algorithm is used to remove a large number of harmonic components, B_∞ should be set small enough to minimize the bias in the frequency estimates of higher harmonics. For example, if it is desired to remove harmonics up to 100^{th} order, setting $B_\infty = 0.001$ would be an adequate choice. In this case, although small values of B_∞ lead to slow frequency adaptation, it would not be problematic, since in practice, the drifts in the power line frequency are usually slow and the algorithm can still reasonably track the variations.

The forgetting factor of the frequency estimator λ_{f} is initially small to achieve a fast convergence and is gradually increased to achieve a more accurate estimate (described in (2.7b)). In the alternative form, P_0 is associated with λ_0 and controls the initial settling time of the frequency estimation algorithm. Smaller values of P_0 are preferred (e.g. tenths of seconds) to achieve a faster initial convergence. Similarly, P_∞ is associated with λ_∞ and controls the asymptotic settling time of the frequency estimation algorithm. Considering the fact that the power line frequency drifts are slow, larger values of P_∞ are preferred (e.g. a few seconds) to obtain a more accurate estimation of the power line frequency. P_{st} controls how fast the settling time changes from the initial value of P_0 to its final value of P_∞ and indicates the time in seconds, in which λ_{f} reaches $0.95\lambda_\infty$ in (2.7b). This transition time should be set large enough (e.g. a few seconds depending on the notch bandwidth) to allow global convergence.

The settling time of the amplitude/phase estimator (W_{a}) controls how fast it responds to the fluctuations in the amplitudes and phases of the harmonics. In the alternative form, W_{a} is associated with λ_{a}, and indicates the time in which the estimates of amplitude and phase reach 95% of their asymptotic

values. The interference frequency bands (e.g. near 50/60 Hz and multiples) contain both the interference components as well as useful neural signals which should be preserved. For this purpose, W_{a} should be selected reasonably large to obtain an accurate estimation of the interference, thus avoiding the excessive removal of neural signals, while small enough to allow tracking of the interference amplitude fluctuations. Depending on the recording environment and subject movements, W_{a} may be selected from a few tenths of seconds to a few seconds. A recommended set of parameter values are suggested in Table 2.1 which could be initially used for further tuning.

Table 2.1: Recommended Values of Parameters

Parameter	Recommended Range
B_0 (Hz)	10 – 50
B_∞ (Hz)	0.01 – 0.1
B_{st} (s)	0.5 – 10
P_0 (s)	0.01 – 0.5
P_∞ (s)	1 – 5
P_{st} (s)	1 – 10
W_{a} (s)	0.5 – 5

2.3 Results

Extensive simulations are carried out to quantitatively evaluate the performance of the proposed algorithm under various signal and parameters conditions. The algorithm performance is also compared with other popular interference removal methods. Furthermore, the algorithm is tested on extracellular, electrocorticography (ECoG) and electroencephalography (EEG) recordings to illustrate its performance on real neural data. The results of the performance evaluation

using synthetic data are described in Section 2.3.1, the performance comparison results are reported in Section 2.3.2, and the results on real data are presented in Section 2.3.3. In case the reader wishes to reproduce the results, the parameter setting in each simulation is provided.

2.3.1 Performance Evaluation on Synthetic Data

Synthetic data are used to quantitatively evaluate the important characteristics of the proposed algorithm under various signal conditions. Each test sequence was synthesised by adding a synthetic interference containing 3 harmonics, to a random portion of real ECoG and extracellular recordings that were recorded in a controlled condition with negligible amount of power line interference. In all cases, the phases of the harmonics are randomly chosen in the range of $[0, 2\pi)$, and their amplitudes are arbitrarily chosen in $[0.1, 1]$ range. After adding up the harmonics they are appropriately scaled to give the desired input SNR. The frequency and power of the interference components are specified in each simulation.

In the rest of this thesis, SNR_{in} and SNR_{out} are used to denote the SNRs of the algorithm input and output signals, i.e. $x(n)$ and $\hat{s}(n)$, respectively. It should be noted that SNR_{out} values are calculated after the algorithm reaches its steady-state, unless otherwise stated.

Sensitivity to SNR_{in}

The variations in the power of the picked-up interference are usually significant, leading to different SNR_{in} values from as low as -20 dB (severe interference), to as high as 30 dB (negligible interference). To ensure proper interference cancellation, the algorithm is desired to work reliably under various SNR_{in} conditions. To evaluate this aspect, we generated synthetic sequences whose SNR_{in} ranged from -20 dB to 20 dB. For each SNR_{in} value, 50 sequences were generated, the

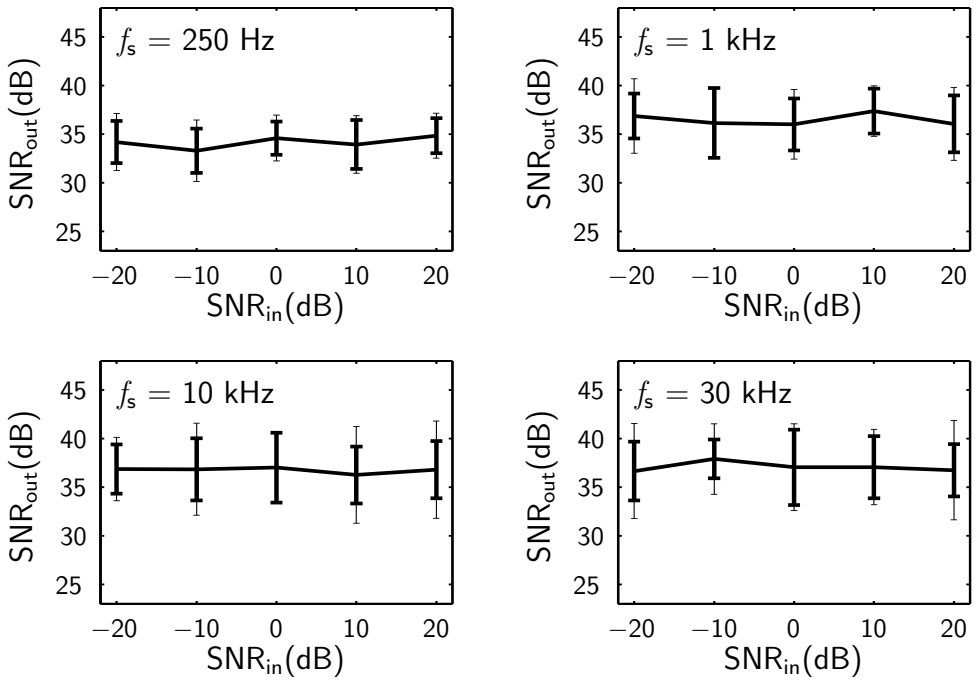

Figure 2.6: $\mathrm{SNR_{out}}$ vs. $\mathrm{SNR_{in}}$. Each figure is obtained at a different sampling rate. The horizontal plots indicate the mean, the thick bars show the standard deviation and the thin bars indicate minimum and maximum $\mathrm{SNR_{out}}$ values over 50 runs on real ECoG signals with synthetic interference containing 3 harmonics at 61 Hz, 122 Hz, and 183 Hz (2 harmonics for f_s=250 Hz). Consistent high values of $\mathrm{SNR_{out}}$ indicate the robust operation of the algorithm with regard to different $\mathrm{SNR_{in}}$ and sampling rates. Parameter setting: $\{B_0 = 50,\ B_{\mathrm{st}} = 1,\ B_\infty = 0.1,\ P_0 = 0.1,\ P_{st} = 1,\ P_\infty = 2,\ W_{\mathrm{a}} = 2\}$.

algorithm was applied to cancel the interference, and the resultant $\mathrm{SNR_{out}}$s were recorded. In addition the simulation was repeated with different sampling rates for reliability resting.

Figure 2.6 shows the mean, variance, minimum and maximum of the resultant $\mathrm{SNR_{out}}$ for each $\mathrm{SNR_{in}}$ condition and sampling rate. It can be seen that, consistent high values of $\mathrm{SNR_{out}}$ are observed in all the conditions, indicating that the performance of the algorithm is highly insensitive to $\mathrm{SNR_{in}}$.

Sensitivity to Power Line Frequency

Since the accurate value of power line frequency is a priori unknown, and may also change over time [32, 33], it is important to test the performance of the

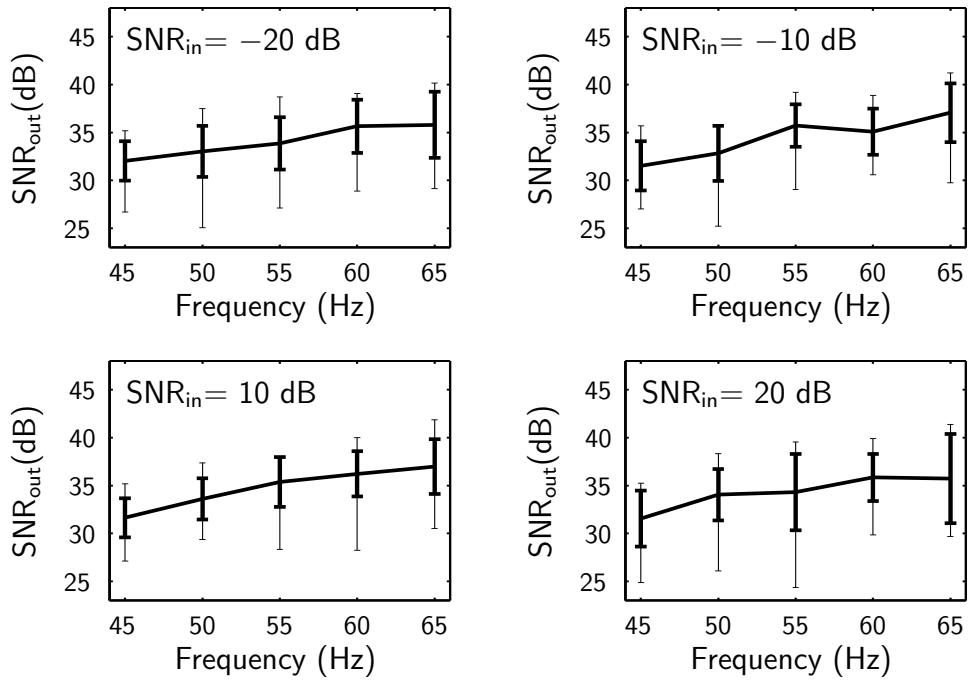

Figure 2.7: $\mathrm{SNR}_{\mathrm{out}}$ vs. power line frequency. The horizontal plots indicate the means, the thick bars show the standard deviation and the thin bars indicate minimum and maximum $\mathrm{SNR}_{\mathrm{out}}$ values over 50 runs on real ECoG signals with synthetic interference containing 3 harmonics. Consistent high values of $\mathrm{SNR}_{\mathrm{out}}$ are achieved in the wide range of power line frequencies and sampling rates. The reason for the slight increase of mean $\mathrm{SNR}_{\mathrm{out}}$ with frequency is mainly due to the $1/f$ PSD of neural signals. With a fixed $\mathrm{SNR}_{\mathrm{in}}$, at higher frequencies, the power of neural signals are less, leading to a more accurate estimation of the interference (neural signals are seen as noise to the interference estimation algorithm), hence resulting in a better cancellation and slightly higher $\mathrm{SNR}_{\mathrm{out}}$ compared with the lower frequencies. Parameter setting is the same as that of Figure 2.6.

algorithm with regard to different power line frequencies. For this purpose, synthetic sequences with fundamental frequencies ranging between 45 Hz to 65 Hz were used as the input to the algorithm, and output SNRs were measured to test the performance. This frequency range covers the worst case power line frequency deviations [32, 33]. As can be seen in Figure 2.7, high values of $\mathrm{SNR}_{\mathrm{out}}$ (> 30 dB) were consistently achieved for different power line frequencies in all the $\mathrm{SNR}_{\mathrm{in}}$ conditions. The results demonstrate the robust operation of the algorithm even in worst case power line frequency deviations. Furthermore, it can be seen that the algorithm can automatically detect the interference at 50 Hz or 60 Hz,

and no a priori setting of the nominal power line frequency is required.

Trade-off between Settling Time and $\mathrm{SNR}_{\mathrm{out}}$

As discussed in Section 2.2.4, there is a trade-off between $\mathrm{SNR}_{\mathrm{out}}$ and the amplitude settling time (W_{a}). To track the abrupt changes in the interference power, fast settling time is desired. Typically, when W_{a} is set small to have a fast tracking response, the $\mathrm{SNR}_{\mathrm{out}}$ would decrease. On the other hand, when W_{a} is set large to achieve a higher $\mathrm{SNR}_{\mathrm{out}}$, then the settling time will increase. It is desirable to achieve a high $\mathrm{SNR}_{\mathrm{out}}$ along with a reasonably fast settling time. Figure 2.8 shows the average values of $\mathrm{SNR}_{\mathrm{out}}$ versus different settling time values (W_{a}). It can be seen that high values of $\mathrm{SNR}_{\mathrm{out}}$ (≈ 30 dB) can be obtained with a reasonably low settling time (< 1 s).

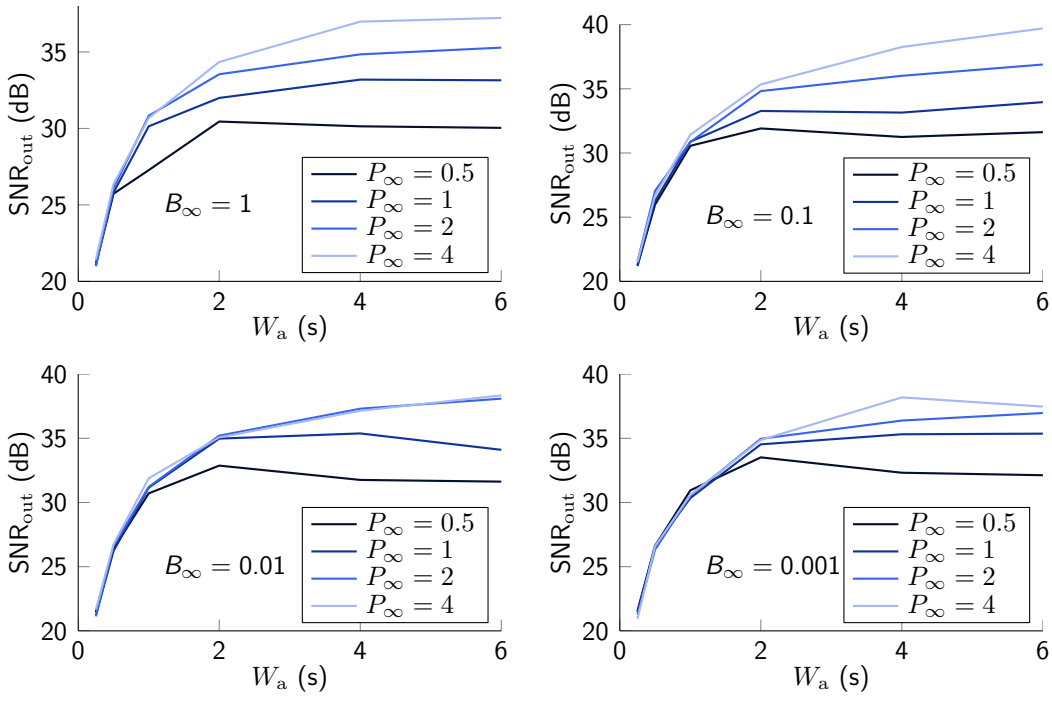

Figure 2.8: Trade-off between amplitude settling time and $\mathrm{SNR}_{\mathrm{out}}$. The plots display the $\mathrm{SNR}_{\mathrm{out}}$ versus amplitude settling time W_{a}, for different P_∞ and B_∞. $\mathrm{SNR}_{\mathrm{in}}$ is set to 0 dB for all the cases. The results show that high $\mathrm{SNR}_{\mathrm{out}}$ values (> 30 dB) can be achieved along with a reasonably fast settling time (< 1 s at $W_{\mathrm{a}} = 1$). Parameter setting: $\{f_{\mathrm{s}} = 1$ kHz, $B_0 = 50$, $B_{\mathrm{st}} = 1$, $P_0 = 0.1$, $P_{st} = 1\}$

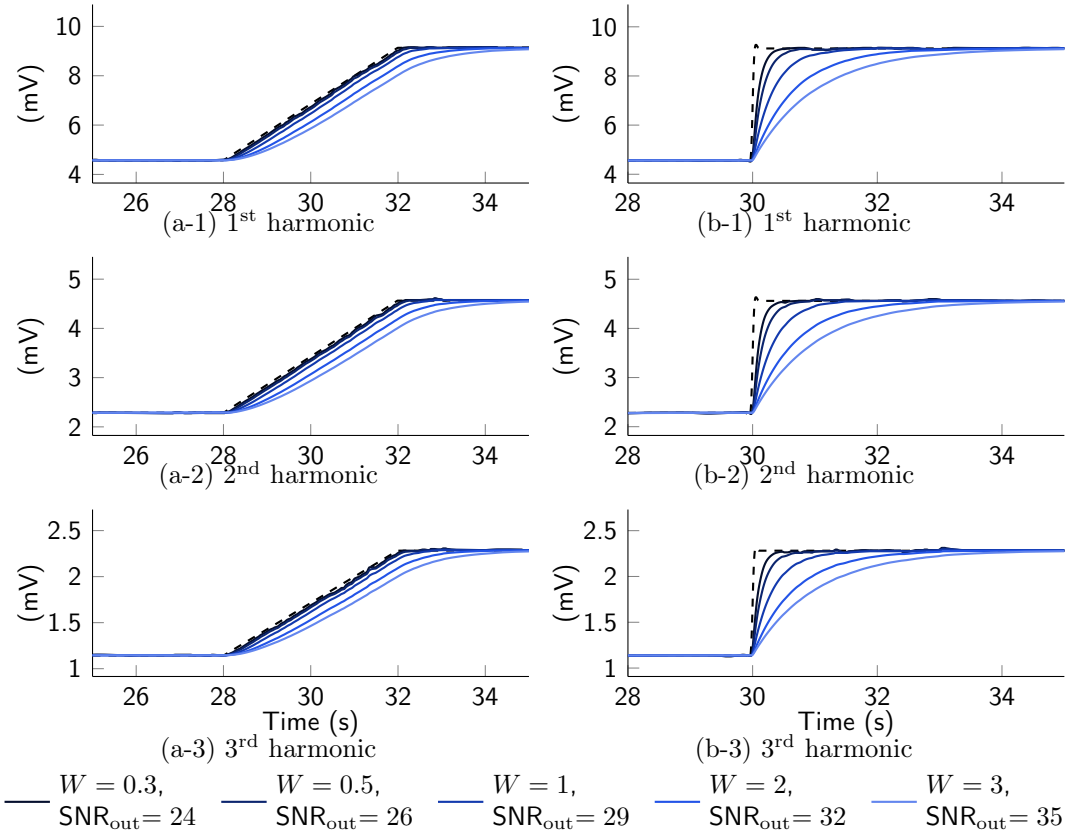

Figure 2.9: Amplitude tracking. In (a-1)–(a-3), the amplitudes of the harmonics were gradually increased to twice their initial values. In (b-1)–(b-3), the amplitudes of the harmonics underwent a step jump. The actual amplitude is displayed by (- - -). The algorithm was applied with different values of W_{a} which led to different settling times and $\mathrm{SNR}_{\mathrm{out}}$ values. The input SNR was set to $\mathrm{SNR}_{\mathrm{in}} = 0$ dB, and $\mathrm{SNR}_{\mathrm{out}}$ values were calculated after convergence ($t > 35$ s). As can be seen, smaller values of W_{a} have led to faster amplitude tracking, however yielded lower $\mathrm{SNR}_{\mathrm{out}}$. On the other hand, larger values of W_{a} resulted in a slower amplitude tracking but yielded higher $\mathrm{SNR}_{\mathrm{out}}$. Parameter setting: $\{f_{\mathrm{s}} = 1$ kHz, $B_0 = 50$, $B_{\mathrm{st}} = 1$, $B_\infty = 0.1$, $P_0 = 0.1$, $P_{st} = 1$, $P_\infty = 1\}$.

Tracking of Amplitude and Frequency Fluctuations

The drifts of the power line frequency are typically small, while the fluctuations of the harmonics amplitudes can be quite large [32, 33]. In order to effectively reject the power line interference, the algorithm should be able to adequately track the frequency and amplitude variations.

To illustrate the amplitude tracking performance, the harmonic amplitudes

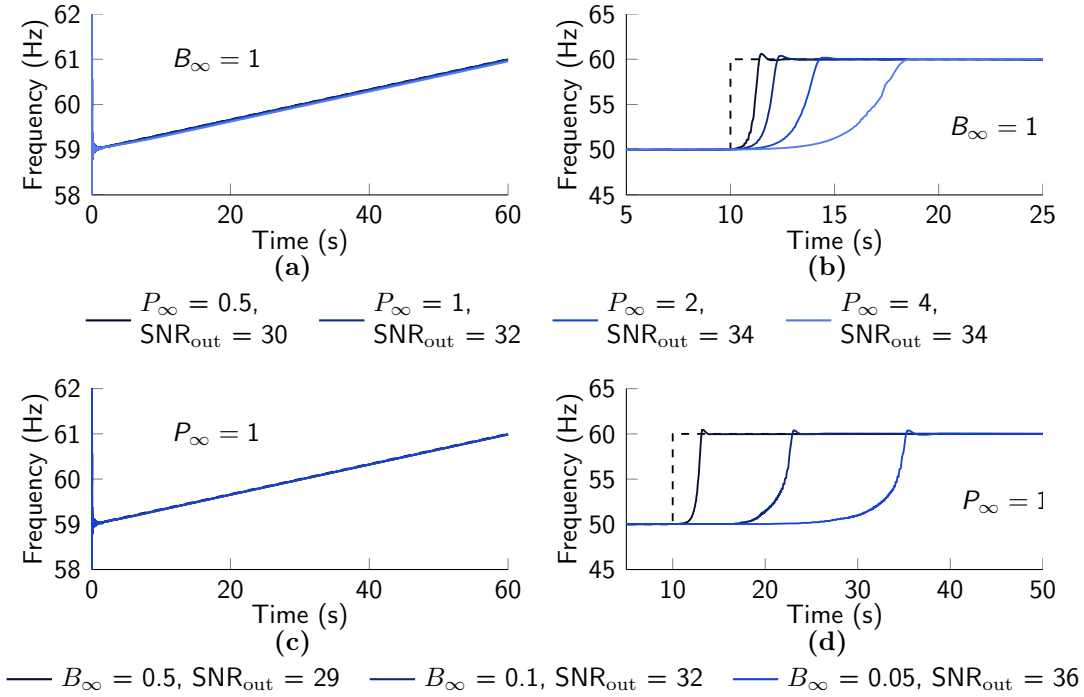

Figure 2.10: Frequency tracking. The actual frequency is shown by (- - -). In (a) and (c) the fundamental frequency is swept from 59 Hz to 61 Hz. It can be seen that, the estimated values properly track the changes of the frequency, with the minimum $\text{SNR}_{\text{out}} = 26$ dB during tracking. In (b) and (d), the fundamental frequency is abruptly changed from 50 Hz to 60 Hz, and the frequency estimates have well track the change. In this simulation, $\text{SNR}_{\text{in}} = 0$ dB and SNR_{out} values are calculated after convergence (t > 20 s in (b) and t > 40 s in (d)). In (a) and (b), $B_\infty = 1$ and P_∞ was varied. In (c) and (d), $P_\infty = 1$ and B_∞ was varied. Parameter setting: $\{f_s = 1$ kHz, $B_0 = 50$, $B_{\text{st}} = 1$, $P_0 = 0.1$, $P_{st} = 1$, $W_a = 1\}$

were increased to twice their initial values and the algorithm was applied with different settling time values (W_a). Figure 2.9 displays the first three interference harmonics, where the they underwent a ramp change and a step change. It can be seen that, the estimates of the amplitudes properly tracked the actual values.

To illustrate the frequency tracking performance, two simulations were done. In the first simulation, the fundamental frequency of the synthetic harmonics was swept from 59 Hz to 61 Hz. It can be seen in Figures 2.10(a) and 2.10(c) that, for all the parameter conditions, the frequency estimates accurately track the actual values. In the second simulation, the fundamental frequency underwent a

step change from 50 Hz to 60 Hz. Figures 2.10(b) and 2.10(d) show that, the frequency estimate converges to the actual frequency with different settling times which depend on the parameters B_∞ and P_∞. It should be noted that due to the use of time-varying parameters in (2.7), the initial convergence is much faster than is in the operating condition.

Initial Convergence

To illustrate the convergence behaviour of the algorithm, two synthetic sequences with interference fundamental frequency of 50 Hz and 60 Hz were used. The interference contained 3 harmonics. Figure 2.11 shows the frequency convergence, where the frequency estimates converged to the actual frequencies (i.e. 50 Hz and 60 Hz) in less than 100 ms, while maintaining a high SNR_{out}. This fast convergence speed is mainly due to adopting time-varying α_{f} and λ_{f}. In other words, the initial convergence is controlled by the parameters B_0, B_{st}, P_0 and P_{st}, whereas the parameters B_∞, P_∞ and W_{a} determine SNR_{out}. The convergence of the three estimated harmonics is displayed in Figure 2.12, where a quick (< 100 ms) convergence to actual harmonics is observed.

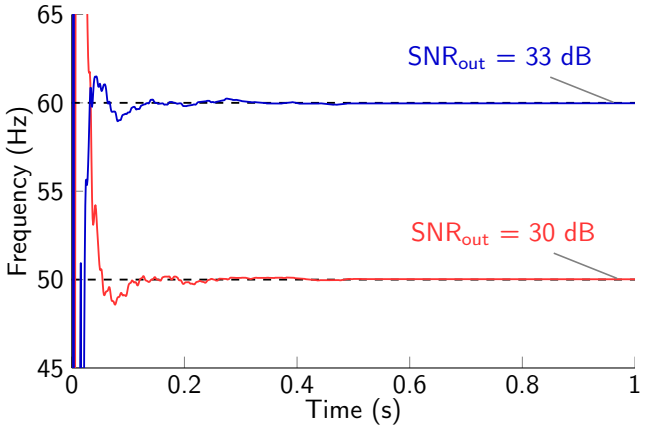

Figure 2.11: The initial convergence to two interference frequencies at 50 Hz and 60 Hz. A quick initial frequency convergence can be observed. In this simulation SNR_{in} = 0 dB, and the values of SNR_{out} were calculated after $t = 1$ s. Parameter setting: $\{f_{\text{s}} = 1$ kHz, $B_0 = 50$, $B_\infty = 0.05$, $B_{\text{st}} = 0.5$, $P_0 = 0.1$, $P_\infty = 2$, $P_{st} = 0.5$, $W_{\text{a}} = 1\}$.

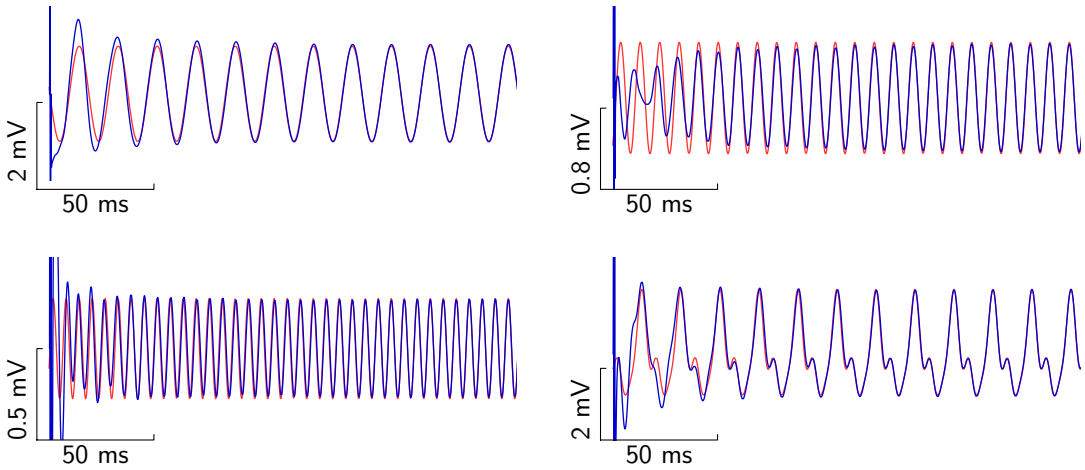

Figure 2.12: Initial convergence of the estimated harmonics. The method shows a fast adaptation of frequency, phase and amplitude. From top to bottom, 1^{st}, 2^{nd}, 3^{rd} harmonic and total interference. The plots show actual components (——) and the estimates (——). In this simulation, $SNR_{in} = 0$ dB and $SNR_{out} = 33$ dB (for $t > 1$ s). Parameter setting is the same as that of Figure 2.11.

2.3.2 Comparison with Other Methods

The algorithm is compared with narrow- and wide-band notch filtering, and two adaptive algorithms proposed by Ziarani *et al.* [20] and Martens *et al.* [18].

In this section, a performance comparison in terms of SNR improvement, mean squared error (MSE), and convergence speed is presented in presented. Further, a comparison between the effects of different interference removal methods on synthetic neural oscillations is described.

Performance Comparison

Figure 2.13 shows the effect of wide- and narrow-band notch filtering on a synthetically corrupted ECoG signal. The interference fundamental frequency was slightly deviated from 60 Hz which translated to even higher deviations in higher harmonics (Figure 2.13(b)). As can be seen in Figure 2.13(c), narrow-band notch filters fail to adequately remove the interference with changing frequency. On the other hand, wide-band notch filters distort the signal PSD (Figure 2.13(d)).

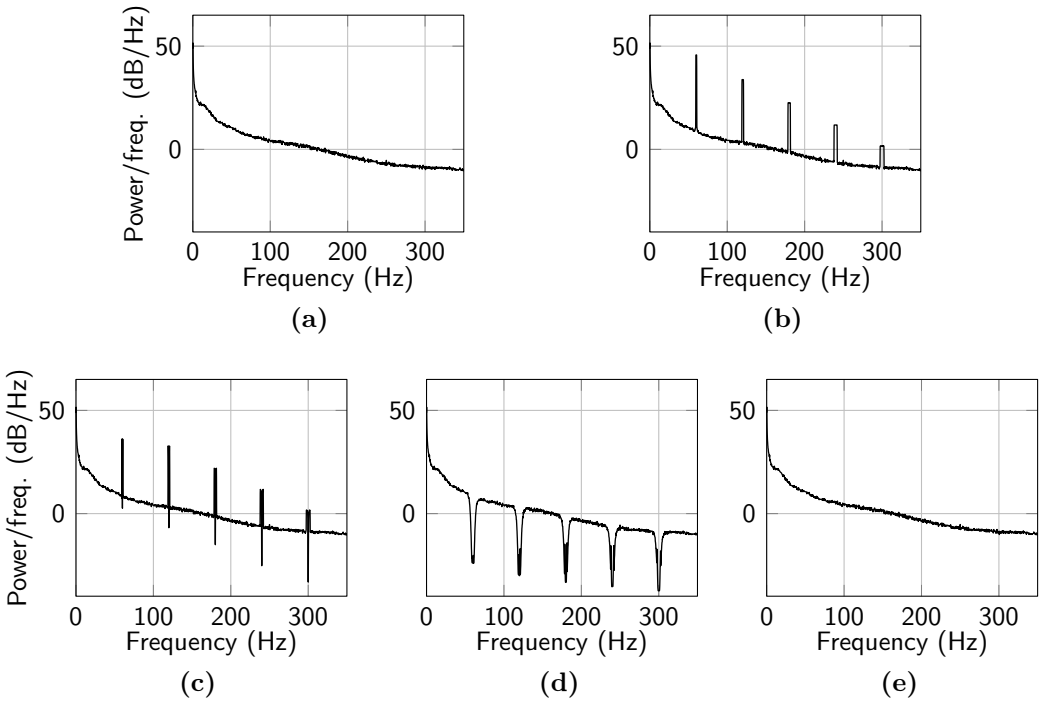

Figure 2.13: The effect of notch filtering on ECoG signal corrupted with power line interference. (a) PSD of the actual ECoG signal. (b) PSD of the synthetically corrupted signal. The interference fundamental frequency is slightly deviated from 60 Hz. (c) PSD after applying narrow-band (1 Hz) IIR notch filters centred at 60 Hz and multiples. (d) PSD after applying wide-band (8 Hz) notch filters. (e) PSD after interference cancellation with the proposed algorithm, where the interference is completely removed while the signal frequency bands are minimally affected. The narrow-band filter fails to adequately remove the interference due to its changing frequency. The wide-band filter distorts the signal spectrum at the rejection bands.

The result of interference cancellation using the proposed method is displayed in Figure 2.13(e). It can be seen that, the interference is adequately removed while the signal frequency bands are highly preserved.

The adaptive methods of Ziarani *et al.* [20] and Martens *et al.* [18] have been widely applied to ECG signals and shown effective in removing non-stationary power line interference. Here, we compare the convergence behaviour and the asymptotic performances of these methods against the proposed algorithm.

The first simulation is done to evaluate the asymptotic performances of the algorithms in terms of $\mathrm{SNR}_{\mathrm{out}}$ versus $\mathrm{SNR}_{\mathrm{in}}$. For this purpose, randomly selected

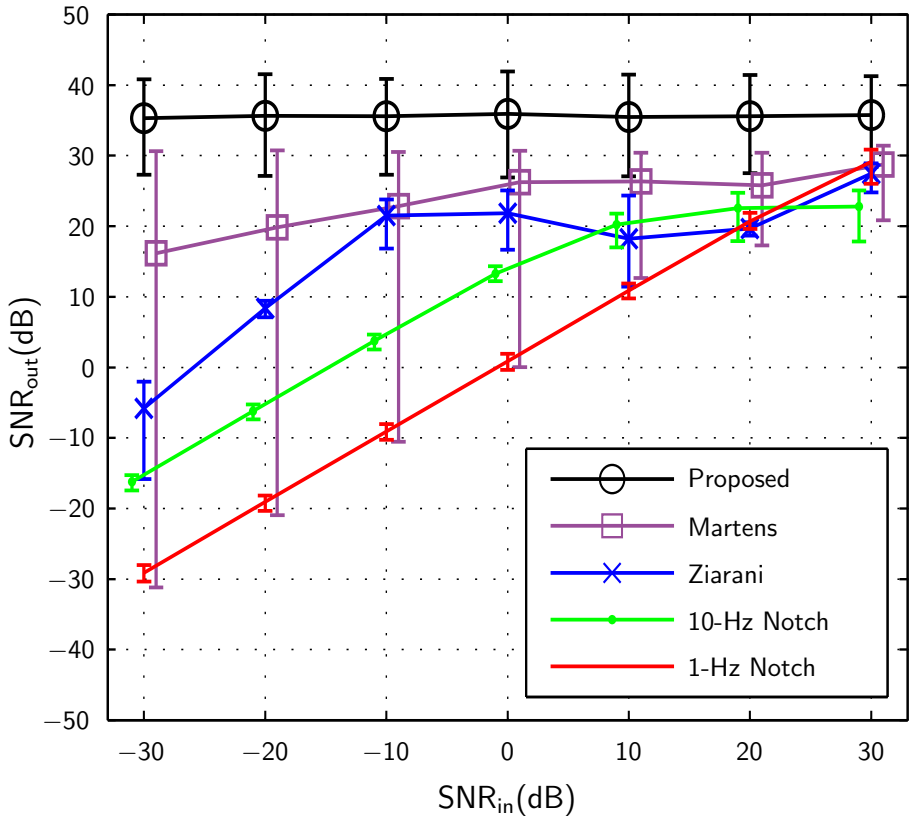

Figure 2.14: Comparison of asymptotic performance of different interference removal methods. The plots represent the average and the error bars indicate the maximum and minimum $\mathrm{SNR}_{\mathrm{out}}$. For each $\mathrm{SNR}_{\mathrm{in}}$, the $\mathrm{SNR}_{\mathrm{out}}$ values are obtained throughout 200 independent runs on synthesised sequences of corrupted ECoG signals. The interference consists of a stationary sinusoid fixed at 59 Hz. The $\mathrm{SNR}_{\mathrm{out}}$ values are calculated after $\mathrm{t} = 60$ s to ensure the full convergence of the algorithms. When applied on ECoG signals, the proposed algorithm consistently yields high $\mathrm{SNR}_{\mathrm{out}}$. The Martens' algorithm performs well in the mean sense; however, large deviations of minimum $\mathrm{SNR}_{\mathrm{out}}$ imply that it may not always converge. The Ziarani's algorithm is sensitive to the input signal power, thus same parameters cannot be used to achieve an optimal performance for different values of $\mathrm{SNR}_{\mathrm{in}}$. The 10-Hz and 1-Hz notch filters are centred at 60 Hz. The parameter setting is the same as that of Figure 2.15.

portions of an interference-free ECoG recording were used and each of which was superimposed with interference containing a single stationary sinusoid at $f_{\mathrm{I}} = 59$ Hz with a random phase and a determined amplitude. The algorithms were allowed to fully converge to their steady states and the values of $\mathrm{SNR}_{\mathrm{out}}$ were calculated for $\mathrm{t} > 60$ s. As can be seen in Figure 2.14, for all $\mathrm{SNR}_{\mathrm{in}}$ values, the proposed algorithm achieves significantly higher $\mathrm{SNR}_{\mathrm{out}}$ compared with other

methods. Furthermore, the small minimum and maximum deviations—shown by the error bars—indicate the reliable convergence and consistent performance of the proposed algorithm. In this simulation, large lower error bars indicate that the algorithm under test may fail to converge.

The second simulation is carried out to evaluate the convergence behaviour of the adaptive algorithms in the mean sense. For this purpose, random signals with $1/f$ PSD (mimicking neural signal PSD) were generated, each of which was superimposed with a single sinusoid (mimicking the interference) whose frequency was slightly deviated from 60 Hz. Subsequently, the MSEs between the output of each algorithm and the actual random signal (without the interference) were calculated. The simulation was then repeated with different values of $\mathrm{SNR}_{\mathrm{in}}$ and interference frequency (f_{I}). In this evaluation, faster convergence and lower MSE values are desirable factors. Figure 2.15 shows that the proposed algorithm consistently achieves faster convergence and lower MSE compared with the other methods. Furthermore, it can reasonably achieve its optimum performance regardless of the initial deviation of the interference frequency from its nominal value.

In the simulations, we observed that the two other adaptive methods were sensitive to the large amplitude artefacts—which are usually present in neural recording—such as electrode displacement and movement artefacts. In addition, since the performances of the algorithms depend on their parameter setting, we fine tuned the parameters of each algorithm to achieve its best performance—in terms of lower MSE and faster convergence—at $\mathrm{SNR}_{\mathrm{in}} = 0$ dB and $f_{\mathrm{I}} = 59$ Hz. Furthermore, the adaptation blocking in the Martens' algorithm is not applicable to ECoG signals, thus their SAC 2 method was used.

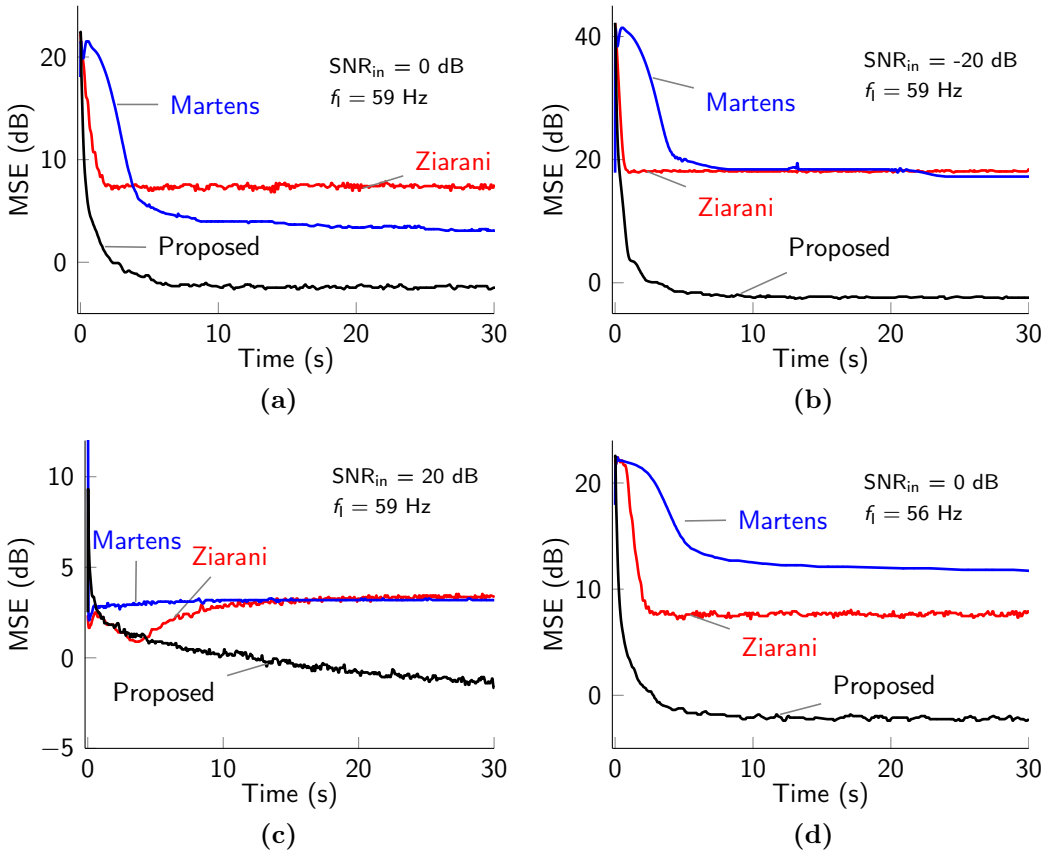

Figure 2.15: Learning curves of the proposed, Ziarani's [20] and Martens' [18] algorithms when applied to random signals with $1/f$ PSD superimposed with a single stationary sinusoid at f_{I} Hz. The MSEs are calculated through 1000 independent runs. The parameters of the algorithms were fine tuned to achieve their optimum performance at $\mathrm{SNR}_{\mathrm{in}} = 0$ dB and $f_{\mathrm{I}} = 59$ Hz which is shown in (a). (b) In low $\mathrm{SNR}_{\mathrm{in}}$, the proposed algorithm still yields low MSE, whereas the increased MSE using the other methods. (c) In high $\mathrm{SNR}_{\mathrm{in}}$, the proposed algorithm can further achieve lower MSE. (d) At $f_{\mathrm{I}} = 56$ Hz; unlike the two other methods, the performance of the proposed algorithm is highly insensitive to the initial frequency deviations. As can be seen in (a)–(d) the proposed algorithm consistently yields faster convergence along with lower MSE compared with the other methods. The nominal frequency is set to 60 Hz in the Ziarani's and Martens' algorithms; however, the proposed algorithm does not require a priori setting of nominal frequency. Parameter setting, $f_{\mathrm{s}} = 1$ kHz, Proposed:$\{B_0 = 50$, $B_{\mathrm{st}} = 0.5$, $B_\infty = 0.1$, $P_0 = 0.1$, $P_{st} = 0.5$, $P_\infty = 1$, $W_{\mathrm{a}} = 1\}$, Ziarani: $\{\mu_1 = 8$, $\mu_2 = 1000$, $\mu_3 = 0.02\}$, Martens:$\{\tau = 200$, $\zeta = 1$, $\omega_n/\omega_p^n = 0.02\}$

Effects on Synthetic Oscillations

Neural oscillations (bio-markers) can appear at, or in the vicinity of, the interference frequency bands. Since these oscillations are useful for information decoding, it is important to ensure that they are well preserved and/or undergo minimal distortion during interference cancellation. To illustrate the performance of the algorithm in this regard, it is tested on synthetic oscillations contaminated with interference. For this purpose, a sequence of patterned oscillations in the range of 50–70 Hz was generated and then added to a background random signal with $1/f$ PSD (Figure 2.16(a)). This sequence represents a synthetic neural signal. Subsequently, a sinusoid (representing the interference) was synthesised and added to the signal. The frequency of this sinusoid was swept from 59 Hz to 61 Hz, and its amplitude was logarithmically increased, setting $\mathrm{SNR_{in}}$ from 10 dB to -20 dB (Figure 2.16(b)).

Different methods of interference removal were applied to the synthesised signal. Figure 2.16(c) illustrates that the proposed algorithm has tracked and removed the interference while reasonably preserving the signal components. Figure 2.16(d) shows that in the Martens' algorithm, the phase-locked loop (PLL) has become out of lock due to the oscillations (e.g. $5 < \mathrm{t} < 15$ s). Figure 2.16(e) shows that the Ziarani's algorithm is sensitive to the interference power, thus same parameters cannot be used to obtain adequate performance for different power of the signal and/or interference. Furthermore, it has distorted the signal near the interference frequency band. Figure 2.16(f) indicates that 10-Hz notch filter has excessively removed the signal components. Figure 2.16(g) shows that the 1-Hz notch filter has only attenuated the interference near $t = 30$ s when its frequency was close to 60 Hz, and failed to remove the interference otherwise. The resultant $\mathrm{SNR_{out}}$s (calculated for $0 < t < 60$) are displayed in Table 2.2.

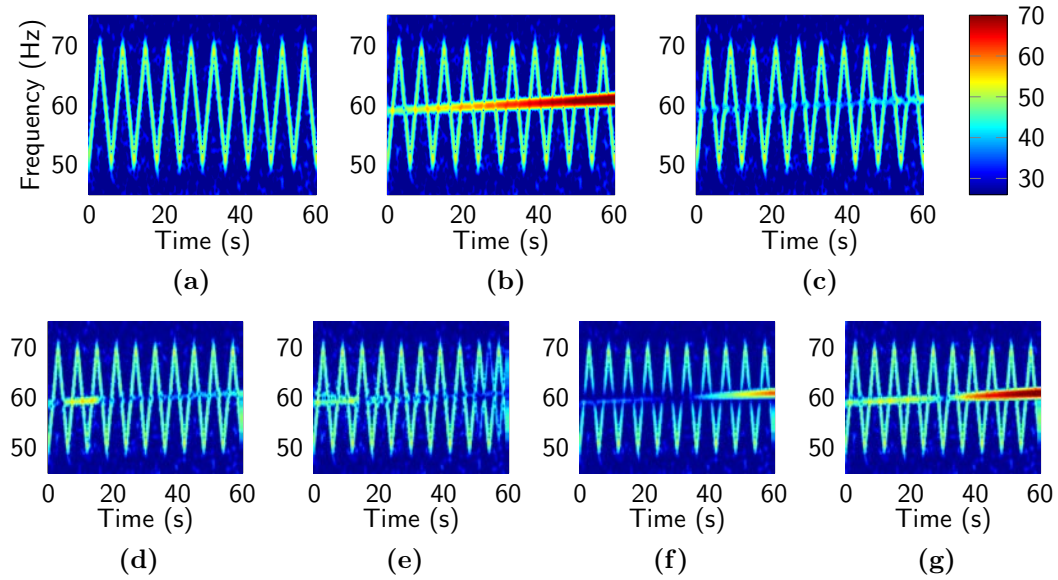

Figure 2.16: Simulation with synthetic oscillations. Time-frequency plots: (a) Synthesised signal consisting of bidirectional chirp between 50–70 Hz (representing the signal of interest) superimposed with a $1/f$ PSD background signal (representing neural noise). (b) After adding a sinusoidal interference whose frequency was swept from 59 Hz to 61 Hz, and its amplitude was logarithmically increased, setting $\mathrm{SNR_{in}}$ from 10 dB to -20 dB. (c) The proposed algorithm has tracked and removed the interference while reasonably preserving the signal components. (d) In the Martens' algorithm, the PLL has become out of lock due to the oscillations (e.g. $5 < \mathrm{t} < 15$ s). (e) The Ziarani's algorithm is sensitive to the interference power, thus same parameters cannot be used to obtain adequate performance for different power of the signal and/or interference. Furthermore, it has distorted the signal near the interference frequency band. (f) 10-Hz notch filter has excessively removed the signal components. (g) 1-Hz notch filter has only attenuated the interference near $t = 30$ s when its frequency was close to 60 Hz, and failed to remove the interference otherwise. The resultant $\mathrm{SNR_{out}}$ values (for $0 < t < 60$) are displayed in table 2.2. Parameter setting, $f_\mathrm{s} = 1$ kHz, Proposed:$\{B_0 = 20,\ B_\mathrm{st} = 0.5,\ B_\infty = 0.1,\ P_0 = 0.2,\ P_{st} = 1,\ P_\infty = 0.5,\ W_\mathrm{a} = 1\}$, Ziarani: $\{\mu_1 = 12,\ \mu_2 = 0.001,\ \mu_3 = 0.2\}$, Martens:$\{\tau = 100,\ \zeta = 1,\ \omega_n/\omega_p^n = 0.01\}$

Table 2.2: Results of Simulation with Synthetic Oscillations

Methods	SNR_{out} (dB)
Proposed	12.06
Martens	8.60
Ziarani	7.90
10-Hz Notch	2.20
1-Hz Notch	−7.85

2.3.3 Performance Evaluation on Real Data

Three types of biosignals including extracellular, ECoG and EEG recordings were used to demonstrate the performance of the algorithm on real data. These signals were recorded in ordinary environments, thus containing a significant amount of power line interference. The PSD of the recorded signals are displayed in Figures 2.17(a), 2.17(d) and 2.17(g) where the presence of the power line interference can be clearly seen. It can be observed that, the harmonics' power can be tens of dB higher than the signal power at the contaminated bands. Furthermore, both odd and even harmonics may be present in the recorded signals.

The extracellular, ECoG and EEG recordings were sampled at 40 kHz, 1 kHz and 128 Hz, respectively. The algorithm was applied to the recorded signals to cancel the interference. Figures 2.17(b), 2.17(e) and 2.17(h) show the PSDs after interference cancellation, where the harmonics have been removed. In addition, the algorithm does not distort the signal frequency components where no harmonic is present. For example in Figure 2.17(b), the PSD remained unchanged at the frequency of the 4^{th} harmonic. Figures 2.17(c), 2.17(f) and 2.17(i) display portions of the gamma band signals before and after interference cancellation.

It is worth mentioning that, a favorable property of the proposed algorithm

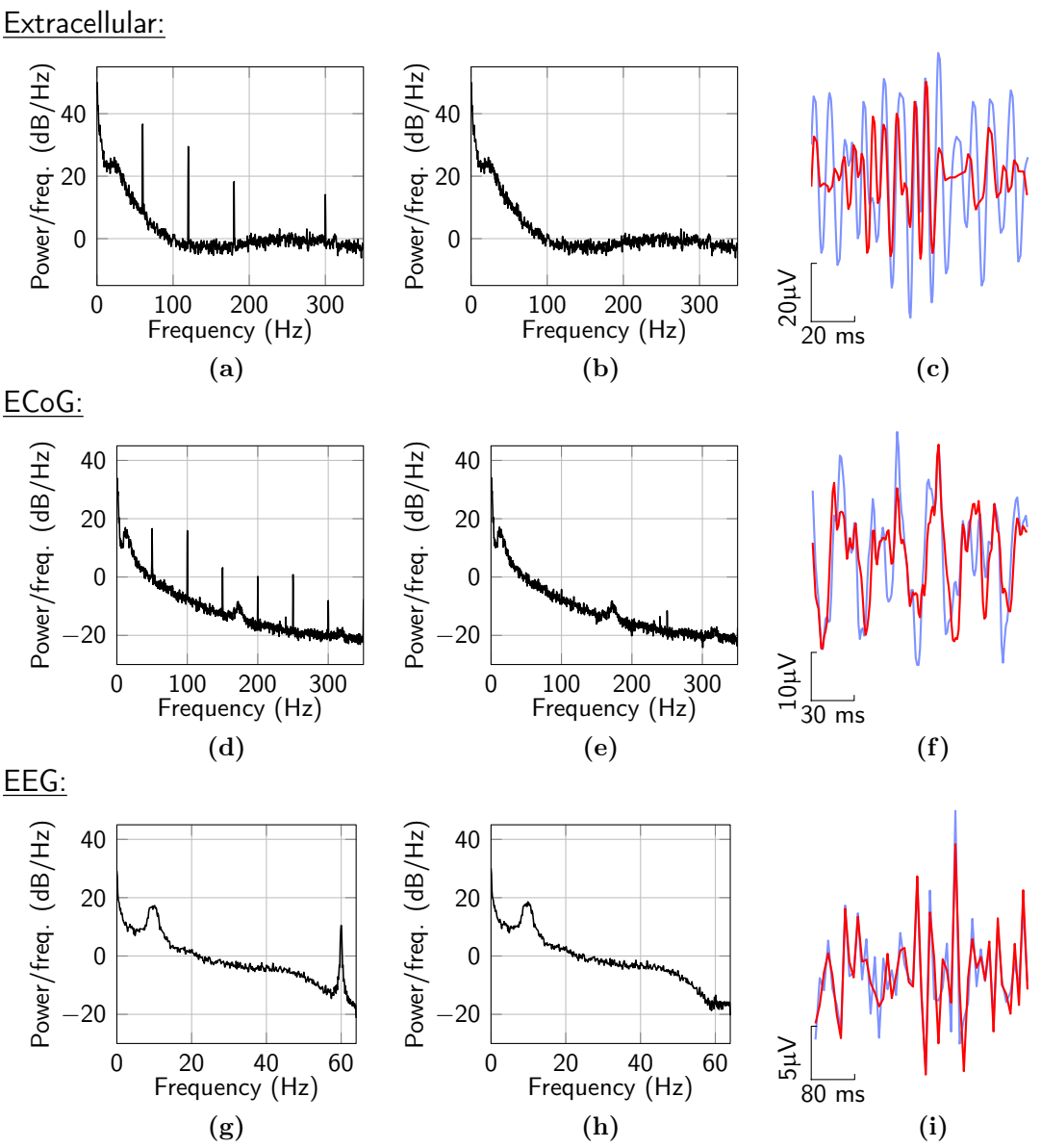

Figure 2.17: Results of the experiment with real data. First column: PSD of the actual recorded signals containing power line interference. Second column: after applying the proposed algorithm, where the interference harmonics are removed. Third column: the actual recorded signal (——), and after interference cancellation by the proposed algorithm (——), displayed in the gamma band ($>$ 30 Hz). Note that in (b), the PSD remained minimally affected at 240 Hz (4^{th} harmonic), where no harmonic was present (compare with (a)). In this experiment common parameter setting is $\{B_0 = 50$, $B_\infty = 0.05$, $B_{\text{st}} = 1$, $P_0 = 0.1$, $P_\infty = 4$, $P_{st} = 1\}$; and specific parameters settings are, extracellular: $\{f_{\text{s}} = 40$ kHz, $A = 2\}$, ECoG: $\{f_{\text{s}} = 1$ kHz, $W_{\text{a}} = 1\}$, EEG: $\{f_{\text{s}} = 128$ Hz, $W_{\text{a}} = 0.5\}$.

is that the proper parameter values remained the same (except W_a) for various signal modalities and sampling rates, confirming the usefulness of defining the alternative parameters. This property makes the algorithm easy to apply on various types of biopotential recordings, with only a slight adjustment of its parameters.

2.4 Discussion

In the design of the algorithm, a number of techniques are used to reduce its computational complexity. First, trigonometric function calculations are avoided in several parts including frequency estimation, harmonic frequencies calculation, and harmonic sinusoids generation. Second, the RLS algorithm is simplified by diagonal approximation of its covariance matrix. These considerations are particularly important in hardware implementation, and significantly reduce the circuit area.

Although the algorithm is mainly proposed for power line interference cancellation, it can also be used to cancel other types of harmonic interferences which may present in the recording. For this purpose, the corner frequencies of the bandpass filter should be adjusted accordingly. Furthermore, the algorithm is applicable to other types of biopotential recordings including ECG and electromyography (EMG) with no modification; further simulation results for these signals are presented in Chapter 3. Moreover, we tested the algorithm on signals corrupted with different types of artefacts such as muscle, eye movement, electrode displacement, and other low and high frequency artefacts. We observed that the algorithm is highly robust to such artefacts. It is important to note that the input signal is assumed to be zero-mean, thus any DC bias should be removed before applying the algorithm.

It should be noted that the algorithm relies on the first harmonic of the interference for frequency estimation. In some situations, however, the first harmonic is suppressed by the recording amplifier, but higher harmonics are still present in the signal. In such situations, the bandpass filter can be accordingly adjusted (e.g. to 90–130 Hz) to estimate the frequency of the second harmonic (κ_2), and the fundamental frequency estimate κ_f can be obtained through $\kappa_\mathrm{f} = \sqrt{(\kappa_2 + 1)/2}$ and subsequently be used for harmonic estimation.

2.5 Conclusion

A robust and efficient algorithm is proposed to remove non-stationary 50/60 Hz interference and its harmonics, from neural recordings. It is highly insensitive to the power of the interference, maintaining high output SNR ($>$ 30 dB) in a wide range of signal and interference conditions. It can effectively track the variations in the frequencies, amplitudes and phases of the harmonics to cancel the interference without compromising the actual neural signals at the interference frequency bands. Furthermore, it features low computational and memory requirements despite using no reference signal for estimation. This property makes the algorithm suitable for real-time applications and hardware implementation.

The convergence, tracking, and estimation accuracy of the algorithm can be controlled through several parameters. An alternative form of these parameters are introduced which have intuitive meaning, and make the parameter adjustment straightforward.

The performance of the algorithm is quantitatively evaluated in terms of output SNR, trade-off between settling time and $\mathrm{SNR}_\mathrm{out}$, and the convergence behaviour. High $\mathrm{SNR}_\mathrm{out}$ ($>$ 30 dB) is consistently achieved in different conditions

of $\mathrm{SNR_{in}}$ (-30 dB to 30 dB), power line frequencies (45 Hz to 65 Hz), and sampling rates (as low as 100/120 Hz). Test results on the trade-off between settling time and $\mathrm{SNR_{out}}$ as well as the tracking behaviour, demonstrate the fast adaptation of the algorithm to interference variations, while maintaining a high $\mathrm{SNR_{out}}$. This makes the algorithm highly suitable for practical applications such as in wearable recording systems, where the interference power undergoes large variations. Moreover, the algorithm features quick (< 100 ms) initial convergence.

The comparative performance evaluation between the proposed algorithm with two other adaptive methods shows the improved performance of the proposed algorithm in terms of noise immunity, output SNR and convergence behaviour.

The algorithm is tested on real extracellular, ECoG and EEG recordings, where almost complete removal of the interference—while preserving the neural signals—is observed. It is also applicable to other types of biopotential recordings including ECG and EMG, with no modification.

Chapter 3

Power Line Interference Cancellation: VLSI Architecture and ASIC

3.1 Introduction

In modern BMI applications, real-time processing of neural signals at hardware level is desired for achieving a fully integrated design. This requires the interference cancellation to be performed at the hardware side. Thus, it is desired to attenuate the interference at the circuit level. In this chapter, we present the VLSI architecture and chip implementation of the proposed interference cancellation algorithm for multichannel biomedical recording applications.

In general, the sources of the interference include displacement currents coupled to the electrodes and to the body, the imbalance in the amplifier's input impedances and/or skin-electrode impedances (leading to 'the potential divider effect'), and magnetic induction in the loops formed by the electrode wires [29].

Shielding wires, electrodes, and the subject, and twisting cables are basic ways to reduce the interference caused by coupling and magnetic induction [3, 29]. Further interference rejection can be obtained in analog by adopting a high CMRR amplifier [53, 54], a driven right leg circuit [55], active electrodes [36, 56], gain adaptation [35], and isolation [57]. These techniques are quite effective in attenuating power line interference, and help relax the required input dynamic range, but they may not completely eliminate the interference, especially in two-electrode recording amplifiers [3].

The interference shows itself in the form of common-mode and differential mode voltage across the amplifier's inputs. Common mode interference voltage can be easily rejected by adopting a high CMRR amplifier [53, 54]. In three-electrode recording, a driven right leg circuit is the most practical way to reduce the common mode voltage further [55]. Drawbacks are that, it is potentially unstable, and requires relatively high biasing current to drive the small resistance connected to the patient's body [58]. Having said that, a major portion of the interference is due to the transformation of the common mode voltage into a differential mode voltage due to the potential divider effect. This effect can be mitigated by adopting several techniques such as active electrodes, [36, 56], gain adaptation [35], and isolation [57].

Digital and analog notch filtering are practical ways to reject the residual interference by attenuating predetermined frequency bands (i.e. 50 or 60 Hz and multiples). Modern analog notch filters do not require large passives and can be fully implemented on silicon [37, 59]. Analog notch filtering reduces the risk of saturation especially when the amplifier does not have a high CMRR. Nevertheless, a drawback is that to account for the inherent variations in power line frequency, the notch bandwidth cannot be too narrow, thus resulting in an increased distortion to the signals near 50 or 60 Hz (or harmonic frequencies)

[18, 20]. This distortion may be acceptable if those bands are not included in subsequent signal analysis. However, in some applications such as neural signal analysis, the biosignals near 50 or 60 Hz (or harmonic frequencies) are especially important and should undergo minimal distortion during interference rejection. Therefore notch filtering would not be the best choice for these applications since it distorts the signal bands of interest. Another analog approach is the direct interference cancelling (DIC) scheme which rejects the interference by adding to it a compensation signal which is equal to the interference but 180° phased shifted [60]. Similar to notch filtering, this approach distorts the signals in the vicinity of 60 Hz [60]. In addition, the transient response caused by closing the DIC feedback may cause amplifier saturation and its large settling time (≈ 3 s) may be unacceptable in some real-time applications [60].

Given a sufficient dynamic range of a recording amplifier, digital adaptive filtering algorithms can be alternatively utilized to cancel out the 50 or 60 Hz noise and harmonics with minimal distortion to the actual biosignals. Since interference voltage can vary in orders of magnitude in different recording conditions, and also its frequency may undergo small drifts, these algorithms should be robust to different signal and interference strengths, and frequency drifts. Furthermore, low computational complexity would be desirable to facilitate hardware implementation for ambulatory or implanted recording systems.

In summary, the reasons we chose to perform adaptive filtering in digital domain are as follows:

- In some applications the bio-signals near 50/60 Hz and harmonic frequencies are particularly important and should undergo minimal distortion (e.g. gamma oscillations in neural recording). Although it requires the amplifier to have sufficient dynamic range, the adaptive filtering in digital better preserves signal frequency components and introduces less filtering

distortion to nearby frequencies compared with analog notch filtering.

- There are usually higher harmonics present in the interference for which the frequency deviations are higher than the fundamental harmonic. This requires the analog notch bandwidths to be larger for higher harmonics in order to effectively suppress them in case of frequency deviations. This, in turn, increases the distortion to the signal especially at the frequencies of higher harmonics. Through adaptive filtering in digital, since the fundamental and harmonic frequencies are accurately estimated, the higher harmonics can be cancelled while significantly preserving signal components, providing higher SNR compared to the fixed-frequency analog notch filtering.

- Notch filtering in analog usually requires large passives for realization leading to large circuit area. Through the state of the art approaches, the area is significantly reduced, yet still not small enough to suit large channel count or multiple harmonics (e.g. 1 mm^2 in 0.18 μm in [37] for 1 harmonic and 1 channel). The proposed architecture in this chapter scales very well with channel count or number of harmonics to be processed which suits it for multichannel interference removal. Apart from the performance, output signal quality and area, the trade-off is the power consumption which is lower for analog approaches.

In this chapter, we present the VLSI architecture and ASIC of the previously proposed algorithm in Chapter 2. The algorithm is extended to cater to multi-channel recording, and its computational complexity and performance are compared against other popular adaptive methods. The VLSI architecture is optimized for removing multiple harmonics from multi-channel recordings by making use of the same hardware blocks at higher clock rates for harmonic estimation. A prototype was fabricated in a 65-nm CMOS, and its functionality

is tested using real ECG, EMG, EEG, ECoG and extracellular recordings. It should be noted that in this chapter, the simulation and testing results are not limited to neural recordings, and we consider other modalities of biopotential recordings as well to demonstrate the applicability of the algorithm to those recording modalities.

3.2 Algorithm Extension for Multichannel Recording

Let x_i be the measured biopotential signal from channel i:

$$x_i(n) = s_i(n) + p_i(n). \tag{3.1}$$

where, $s_i(n)$ is the actual biosignal, and $p_i(n)$ is the power line interference, all recorded from channel i and sampled at f_s Hz. $p_i(n)$ consists of harmonic sinusoids with unknown frequencies, phases and amplitudes as

$$p_i(n) = \sum_{k=1}^{M} \underbrace{a_{i,k}\cos(k\omega_\mathrm{f} n + \phi_{i,k})}_{h_{i,k}(n)}. \tag{3.2}$$

Here, ω_f is the fundamental frequency in rad/s, $a_{i,k}$ and $\phi_{i,k}$ are the amplitude and phase of the k^{th} harmonic in the i^{th} channel, and M is the number of harmonics present in the interference. Denoting $\hat{h}_{i,k}$ as the estimate of $h_{i,k}$; the estimates of the actual signals $\hat{s}_i(n)$ are obtained as

$$\hat{s}_i(n) = x_i(n) - \sum_{k=1}^{M} \hat{h}_{i,k}(n). \tag{3.3}$$

For estimation of $\hat{h}_{i,k}$, the procedure is the same as in Section 2.2, except

for the harmonic estimation, which is optimized for multichannel processing. It should be noted that since the interference frequency is the same among all the channels, the obtained frequency estimate from one channel (i.e. x_j) can be used in harmonic estimation for the other channels. In other words, we can replace $x(n)$ by $x_j(n)$ in (2.4a) and obtain κ_{f} for harmonic estimation.

3.2.1 Harmonic Estimation for Multichannel Recording

As previously discussed in Section 2.2.2, harmonic estimation is carried out in the following steps: First, the harmonic frequencies (i.e. $\cos(k\omega_{\mathrm{f}})$) are directly calculated from κ_{f} by using the Chebyshev method. Subsequently, M pairs of orthogonal harmonics with fixed phases and amplitudes are generated by using a digital wave-guide oscillator (2.9a). It is worth mentioning that the gain control in (2.9b) may be omitted from hardware implementation to save circuit area with negligible degradation of the performance. Finally, amplitude and phase of each harmonic are estimated. Here, in addition to the simplified RLS algorithm, the least mean squares (LMS) algorithm is presented for amplitude and phase adaptation which leads to a lower computational cost with reasonable performance trade-off.

Given from (2.9) that u_k and u'_k are orthogonal sinusoids with the frequency of $k\omega_{\mathrm{f}}$, and fixed amplitude and phase, harmonic k in channel i is estimated through linear combination of u_k and u'_k as

$$\hat{h}_{i,k}(n) = w_{i,k}(n)u_k(n) + w'_{i,k}(n)u'_k(n), \tag{3.4}$$

where $w_{i,k}$ and $w'_{i,k}$ are adaptive coefficients.

Defining the instantaneous error as

$$e_{i,k}(n) = x_i(n) - \hat{h}_{i,k}(n),$$

the RLS algorithm minimizes the cost function

$$E_{i,k} = \sum_{l=0}^{n} \lambda_{\mathrm{a}}^{n-l} e_{i,k}^2(l),$$

where $0 \ll \lambda_{\mathrm{a}} < 1$ is the forgetting factor. As shown in Section 2.2.2, with orthogonal inputs and λ_{a} close to 1, the sample correlation matrix of the RLS algorithm can be well approximated by a diagonal matrix, leading to a considerable reduction in computational cost. The simplified RLS update equations are obtained as

$$r_{i,k}(0) = r'_{i,k}(0) = w_{i,k}(0) = w'_{i,k}(0) = 0,$$

$$r_{i,k}(n) = \lambda_{\mathrm{a}} r_{i,k}(n-1) + u_k(n)^2, \tag{3.5a}$$

$$r'_{i,k}(n) = \lambda_{\mathrm{a}} r'_{i,k}(n-1) + u'_k(n)^2, \tag{3.5b}$$

$$w_{i,k}(n+1) = w_{i,k}(n) + u_k(n) e_{i,k}(n) / r_{i,k}(n), \tag{3.5c}$$

$$w'_{i,k}(n+1) = w'_{i,k}(n) + u'_k(n) e_{i,k}(n) / r'_{i,k}(n), \tag{3.5d}$$

where, for channel i and harmonic k, $r_{i,k}$ and $r'_{i,k}$ are the diagonal elements of the sample correlation matrix.

Alternatively, LMS algorithm minimizes the cost function

$$E'_{i,k} = \mathbb{E}\{|e_{i,k}|^2\},$$

with the update equations given as

$$\begin{aligned} w_{i,k}(n+1) &= w_{i,k}(n) + \mu_1 u_k(n) e_{i,k}(n), \\ w'_{i,k}(n+1) &= w'_{i,k}(n) + \mu_2 u'_k(n) e_{i,k}(n), \end{aligned} \tag{3.6}$$

where μ_1 and μ_2 are step sizes. (3.6) can be alternatively used to update the coefficients; however, given the same convergence rate, (3.5) achieves a lower steady state error in general condition. Finally, the estimates of interference-free signals $\hat{s}_i(n)$ are obtained through (3.3).

3.3 Simulation and Comparative Results

Important factors to evaluate the algorithm performance include SNR improvement, computational complexity and real-time performance, reliability and speed of convergence, robustness to different magnitudes of the input signal and/or interference, robustness to power line frequency deviations, and the requirement for a reference signal. The proposed algorithm is compared with popular interference cancellation methods in terms of computational complexity and interference rejection performance. The computational complexity of the methods is presented in Table 3.1. The interference rejection performance was evaluated by using real, interference-free biopotential recordings contaminated with synthetic interference. In the rest of this chapter, $\mathrm{SNR_{in}}$ is defined as the ratio of the power of the clean signal (used for synthesising the chip input) to the power of the synthesised interference. $\mathrm{SNR_{out}}$ is the ratio of the power of the actual clean signal to the power of the residual interference and any added noise or distortion in the chip output. They are calculated through

$$\mathrm{SNR_{in}} = 10\log(\sum_n s_i(n)^2 / \sum_n p_i(n)^2), \tag{3.6a}$$

$$\mathrm{SNR}_{\mathrm{out}} = 10\log(\sum_n s_i(n)^2 / \sum_n (s_i(n) - \hat{s}_i(n))^2), \tag{3.6b}$$

where s_i is the clean bio-signal used for synthesizing the corrupted sequence for testing, and $\hat{s}_i$ is the signal after interference cancellation (i.e. chip output), all for channel i.

The parameters of all the methods were tuned to give their best performance at $\mathrm{SNR}_{\mathrm{in}} = 0$ dB, and interference frequency (f_{I}) of 60 Hz.

Table 3.1: Comparison of Computational Complexity

	Proposed—RLS	Proposed—LMS	[20]	[18]	ANC [40]†
Gains	$10 + 2MI$	$10 + 2MI$	$4MI$	$5I + 13MI$	$2I$
Multipliers	$4 + 2M + 6MI$	$4 + 2M + 6MI$	$4MI$	$I + 4MI$	$4I$
Dividers	$1 + 2MI$	1	0	MI	0
Adders	$13 + 4M + 7MI$	$10 + 2M + 4MI$	$5MI$	$5I + 14MI$	$4I$
Registers	$10 + 2M + 4MI$	$13 + 2M + 4MI$	$3MI$	$2I + 7MI$	$\lfloor \frac{1}{4} f_{\mathrm{s}} / f_{\mathrm{N}} \rfloor$
Sin/Cos	0	0	$2MI$	$2MI$	0
Reference	No	No	No	No	Yes

†Only applicable to fundamental sinusoidal component of the interference, i.e. $M = 1$.
f_{N} is the nominal line frequency which is either 50 Hz or 60 Hz.

Figure 3.1 shows the learning curves of different adaptive methods. The MSE curves were estimated through 1000 independent runs using synthetically corrupted biopotential signals. Wherever possible, the parameters were adjusted to achieve the same convergence rate for each method for a better comparison. The ANC used a high quality reference signal with SNR of 100 dB, while no reference signal is required for the other methods. Despite using no reference signal, the performance of the proposed methods is similar to that of the ANC which requires a high quality reference signal, while outperforming the other methods and showing a consistent performance across different signal modalities.

It should be noted that the ANC becomes ineffective if the interference contains high order harmonics. In the proposed method, the LMS and RLS adaptation yield similar results; however, the LMS step sizes are more difficult to adjust and are slightly sensitive to large frequency deviations.

Figure 3.2 shows the SNR improvement obtained using different methods. The results show that while the method of [18] and ANC yield comparable results to the proposed algorithms at $\text{SNR}_{\text{in}} = 0$ dB, their performance significantly deteriorates when the interference is strong ($\text{SNR}_{\text{in}} = -30$ dB).

3.4 VLSI Architecture

To allow an efficient and scalable implementation for multiple channels and/or harmonics, a sequential architecture is proposed. Moreover, pipelining and resource sharing techniques are utilized to increase the throughput, and optimize the area.

Scalable Sequential Architecture

The algorithm can be implemented through either a parallel or a sequential structure. On one hand, a parallel structure yields higher throughput, however requires large circuit area and is not scalable and practical with increasing number of channels and/or harmonics. On the other hand, a sequential structure is scalable, and significantly saves the area at the cost of a higher clock frequency. The frequency increase is practical since biomedical recording applications usually require low bandwidth (i.e. < 10 kHz). For example, to remove 5 interference harmonics, sampled at 10 kHz, from 100 channels, the maximum clock frequency required by the proposed sequential architecture will be 10 MHz which is in the practical range. Thus, we propose a sequential architecture to obtain a scalable

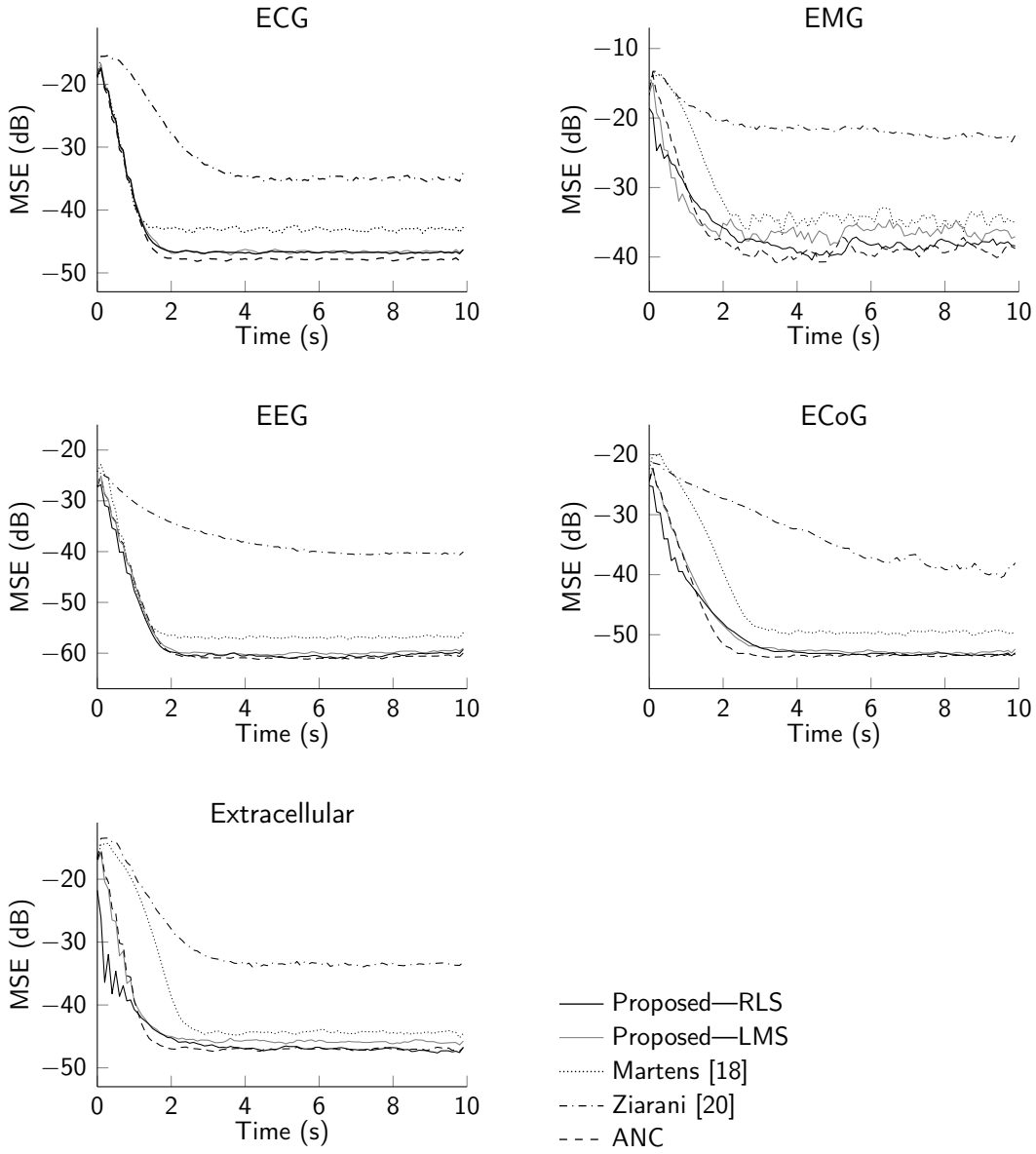

Figure 3.1: Learning curves of different adaptive methods on different modalities of biopotential signals. The interference contained a single sinusoid at 57 Hz, and $\mathrm{SNR}_{\mathrm{in}} = 0$ dB. The nominal line frequency for [20] and [18] was set to 60 Hz, and the ANC used a high quality (SNR = 100 dB) reference signal.

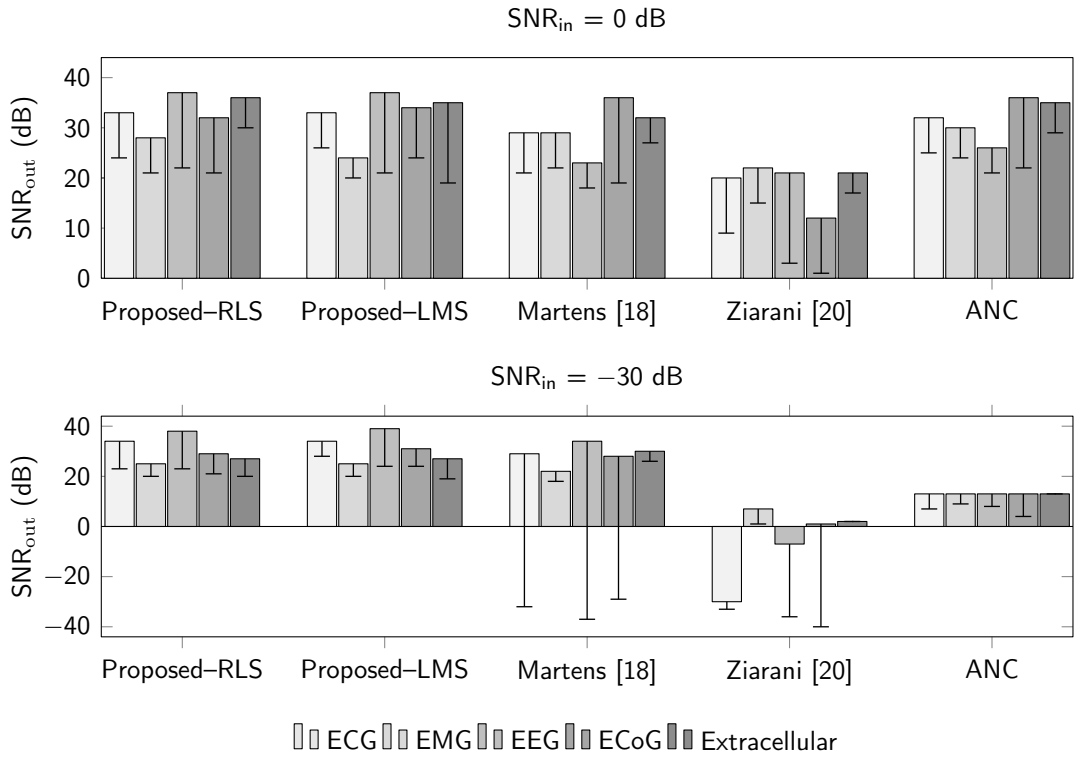

Figure 3.2: Output SNR for different methods and signal modalities. Error bars represent the minimum values. Each bar is the average $\mathrm{SNR}_{\mathrm{out}}$ for 100 runs on synthetic signals with interference frequencies of $f_{\mathrm{I}} \sim \mathcal{N}(60, 1)$. The parameters were tuned to have the same convergence rate and highest SNRout at $\mathrm{SNR}_{\mathrm{in}} = 0$ dB and $f_{\mathrm{I}} = 60$ Hz.

and area-efficient design, while utilizing the pipelining technique to increase the throughput. Having said this, the throughput without pipelining is still acceptable in biopotential recording since the sampling rate is typically not very high (i.e. <100 kHz).

The proposed architecture is shown in Figure 3.3, with the timing diagram of the signals in Figure 3.4, and the detailed signal flow graph in Figure 3.5. In each cycle of f_{s}, κ_{f} is upsampled M times to calculate $\kappa_1 \cdots \kappa_M$ which are sequentially fed to the discrete-oscillator block where M pairs of harmonic samples are generated at the rate of Mf_{s}. To estimate phase and amplitude for multiple channels, in each cycle of Mf_{s}, samples from I channels are multiplexed and sequentially fed to phase/amplitude estimation block at the rate of MIf_{s}, where

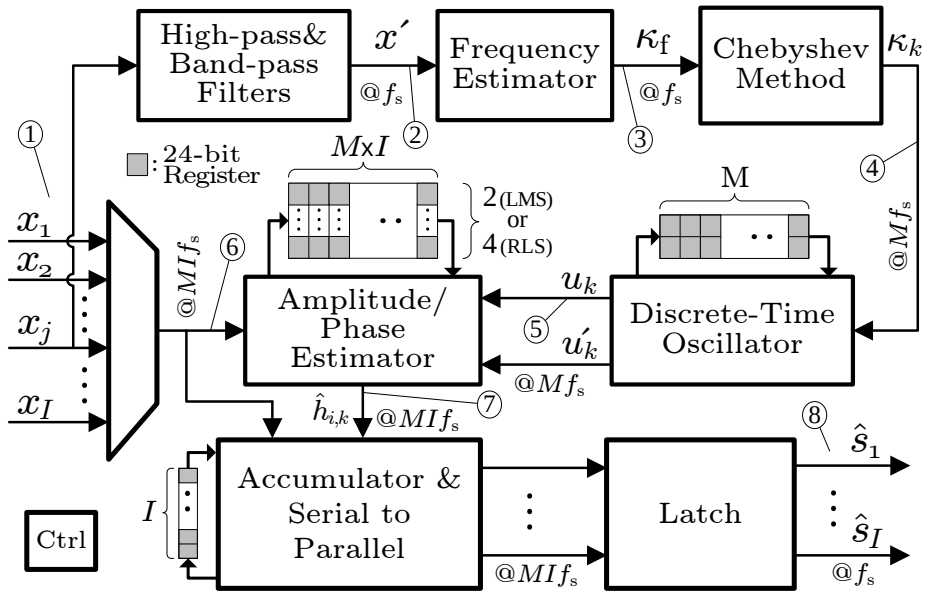

Figure 3.3: The sequential architecture of the proposed algorithm.

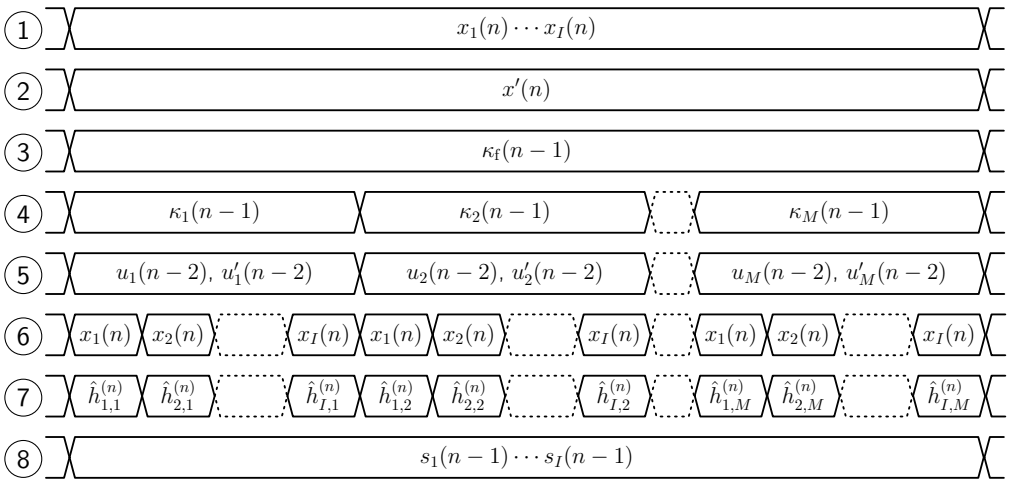

Figure 3.4: Timing diagram of the signals labeled in Figure 3.3.

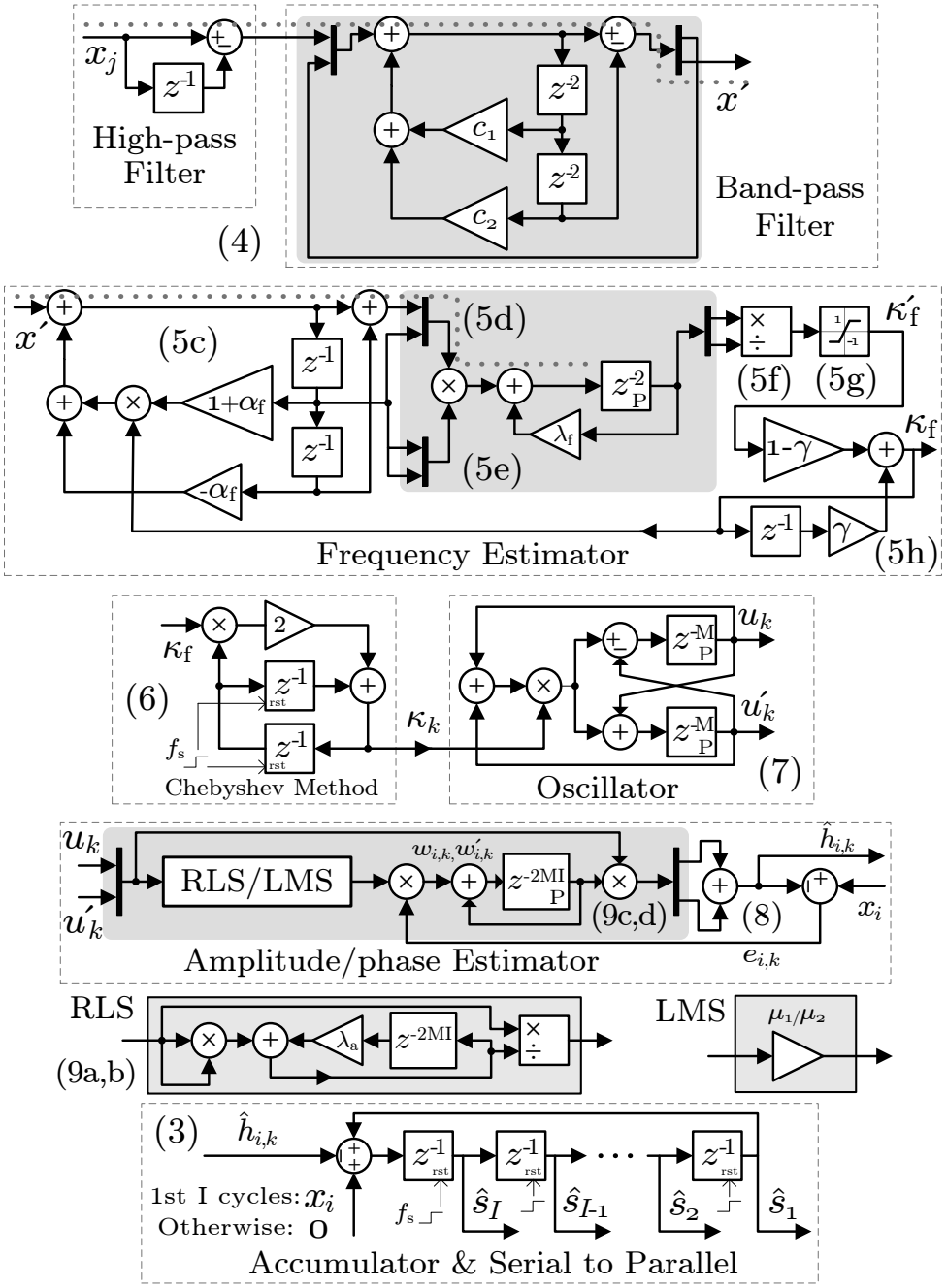

Figure 3.5: Detailed signal flow graph of the blocks. The highlighted sections are reused to save area. The critical path is shown by (·····). The numbers in parenthesis refer to the related equations in Section 3.2.

the samples of one harmonic in I channels are estimated. At the same time, the estimated samples are sequentially subtracted from input samples, and fed into serial to parallel shift registers. After MI cycles, all MI samples are subtracted from all I channels and the output is valid.

Pipelining

Pipelining is used to shorten the critical path (shown in Figure 3.5) and improve the throughput. The registers which hold the state variables are used to achieve pipelining (marked by P in Figure 3.5) without inclusion of extra pipeline registers. This causes sample latency of 1 for frequency estimator, 2 for harmonic generator, and 1 for phase/amplitude estimator. In other words, 2-sample old frequency estimate and 1-sample old adaptive coefficients are used to estimate current estimate of harmonics. The latency in harmonic generator only leads to a phase shift which does not affect the adaptation. The effect of latencies in frequency and phase/amplitude estimators is negligible since the variation in the frequency, amplitude and phase of the interference are much slower than the sampling rate. After pipelining, the critical path delay is equivalent to the delays of 6 adders (11 before), 1 multiplier (4 multipliers and 2 dividers before), and 3 multiplexers (5 before).

Resource Sharing

The hardware resources in several blocks are shared to improve the area efficiency at the cost of twice higher clock rate for the shared computational block. For this purpose, the inputs are multiplexed and sequentially fed to the shared block, and the output are demultiplexed and latched. Shift registers are used to sequentially store/retrieve the states. In the band-pass filter a biquad section is reused for implementing the 4^{th}-order filter. In the frequency estimator, the

Table 3.2: Shared Hardware Resources

Block	Gain	Multiplier	Divider	Adder	Registers	Max Clock
Band-pass Filter	2	0	0	4	5	$2f_s$
Freq. Estimator	5	2	1	5	5	$2f_s$
Oscillator	0	2	0	4	$2+2M$	Mf_s
RLS Estimator	1	3	1	4	$4MI$	$2MIf_s$
LMS Estimator	2	2	0	3	$2MI$	$2MIf_s$
Accumulator	0	0	0	2	I	MIf_s
Total—RLS	7	7	2	17	$12+I+2M+4MI$	$2MIf_s$
Total—LMS	8	6	1	16	$12+I+2M+2MI$	$2MIf_s$

averager and a multiplier is reused. In the amplitude/phase estimator, the circuitry implementing the RLS/LMS update equation is reused for calculating the two adaptive coefficients. The shared modules are highlighted in Figure 3.5. The hardware resources and the clock rates of the blocks are listed in Table 3.2.

3.5 Chip Implementation and Measurement Results

A prototype of the proposed—RLS algorithm was implemented in a 65-nm CMOS process. In this prototype M=3, I=1, f_s=1.25 kHz, and the chip input/output word-length was 16-bit. The design was implemented in fixed-point, where optimum word-lengths were obtained through worst-case simulation. The resource sharing techniques were utilized (except for the bandpass filter block) to save circuit area. The architecture was coded in Verilog, and simulated in ModelSim Altera. The hardware description was synthesised using Synopsys Design Compiler for a 65-nm CMOS process.

The chip was interfaced by using an AVR microcontroller for testing purpose. The microcontroller circuit is solely used for testing to get the data from a PC

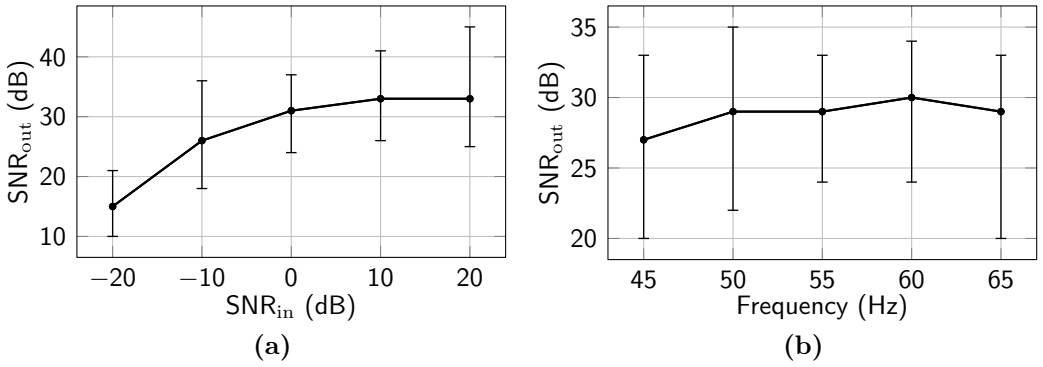

Figure 3.6: Chip testing results. Right: Chip output SNR vs. input SNR where $f_{\mathrm{I}} \sim \mathcal{N}(60, 1)$. Left: Chip output SNR vs. interference fundamental frequency where $\mathrm{SNR}_{\mathrm{in}} = 0$ dB. In both tests the interference contained 3 harmonics. Error bars show the maximum and minimum values over 20 measurements.

through a USB port and feed them to the chip through a synchronous serial interface, and similarly to read the chip output data and transfer them to the PC. For this purpose, the microcontroller generates clock signal, reset signal, and serialized data signal, and reads serialized output signal from the chip. In this chip synchronous serial interface was implemented internally to communicate with the main interference cancellation module (16-bit input and output buses). Since there were other modules from others' work implemented in the same chip, we had limitation on the number of the available pins and therefore used serial interface.

For validation, a series of synthesised and real recordings were fed into the chip and the output was measured and verified against a full-precision golden model implemented in MATLAB. The interference rejection performance of the chip was tested with regard to input SNR and interference frequency. Figure 3.6(a) shows the chip $\mathrm{SNR}_{\mathrm{out}}$ for $\mathrm{SNR}_{\mathrm{in}}$ between -20 and 20 dB, and $f_{\mathrm{I}} \sim \mathcal{N}(60, 1)$, where an average SNR improvement of 30 dB was obtained. Figure 3.6(b) shows the output SNR at interference frequency range of 45–65 Hz, and $\mathrm{SNR}_{\mathrm{in}} = 0$ dB, where a reliable operation within this frequency range can be observed. Figure 3.7

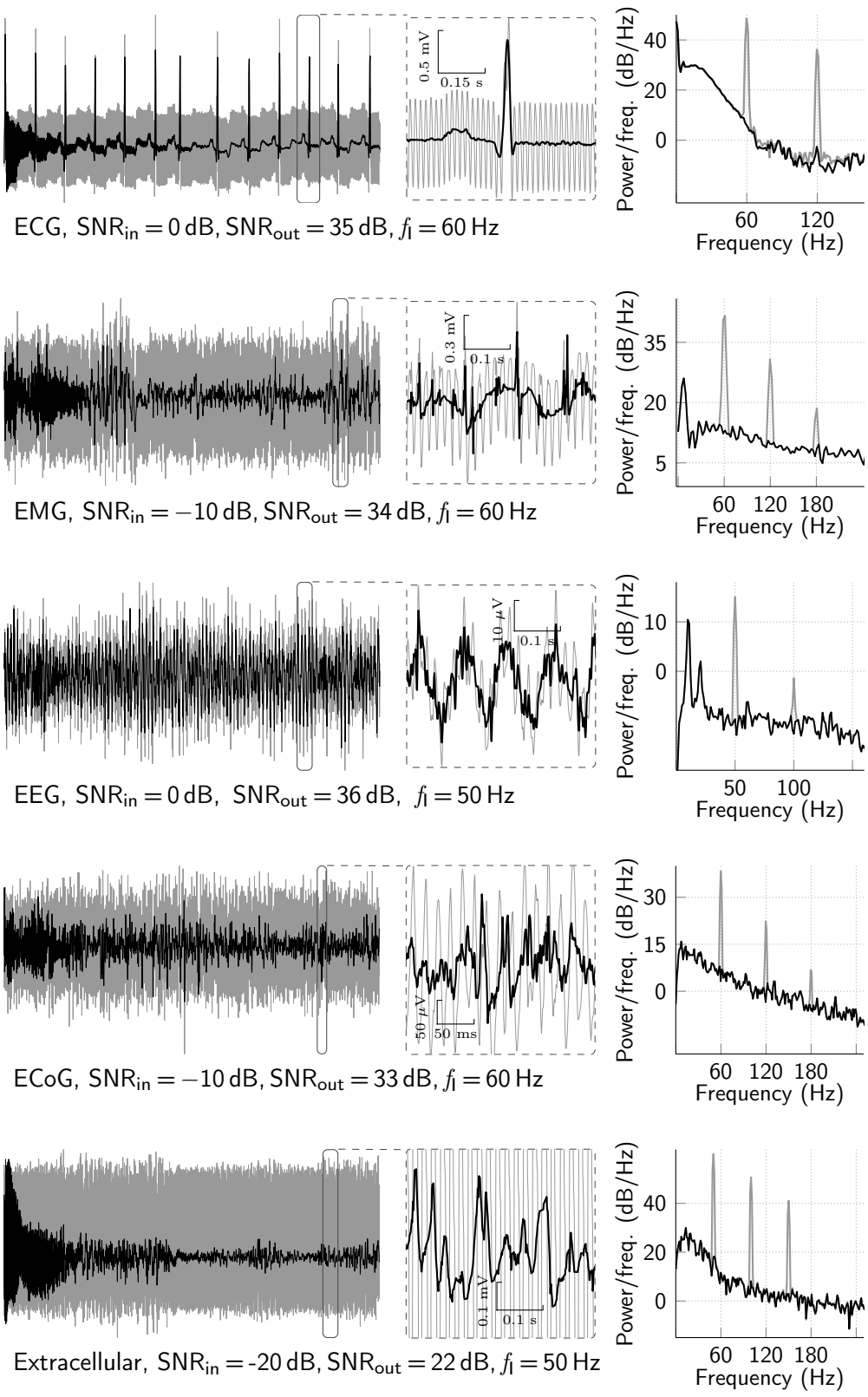

Figure 3.7: Sample chip input (——) and output (——) signals.

shows samples of chip input and output signals in time and frequency domains. It can be seen from PSDs that the harmonics are completely suppressed while the signal components are reasonably well preserved. It should be noted that the algorithm automatically converges to 50 Hz or 60 Hz frequencies regardless of the initial conditions. This is not the case with [20] and [18], since their convergence may be compromised when their initial frequency is far from actual interference frequency.

To demonstrate the chip performance when the interference changes in frequency or amplitude, we carried out three tests on ECoG signals. In all the tests, the lengths of the input signals were 1 minute, and the SNR_{out} values were calculated for $t > 20$ s.

In Test 1, $\text{SNR}_{\text{in}} = 0$ dB and the interference consisted of three stationary harmonics with the fundamental frequency of 59 Hz. The signal was then fed into the chip in real-time and the output was recorded. Figure 3.8 shows the result of Test 1. The PSD of the chip output signal can be seen in Figure 3.8 showing that the interference is significantly attenuated. Furthermore, the signals traces in the time domain clearly show that the chip output properly follows the reference model output.

In Test 2, the interference power was suddenly increased, changing the SNR_{in} from 0 dB to -10 dB. Figure 3.9 shows the chip input and output signals. It can be seen that, the chip output consistently follows the reference model output. Moreover, after the step jump in the interference power at (- - -), the chip output adapted to the change to reject the residual amount of the interference.

In Test 3, $\text{SNR}_{\text{in}} = 0$ dB and the interference fundamental frequency was changed from 60 Hz to 60.2 Hz. The chip input and output signals are shown in Figure 3.10. It can be seen that, the chip output consistently follows the reference model output. Moreover, after the frequency change at (- - -), the chip

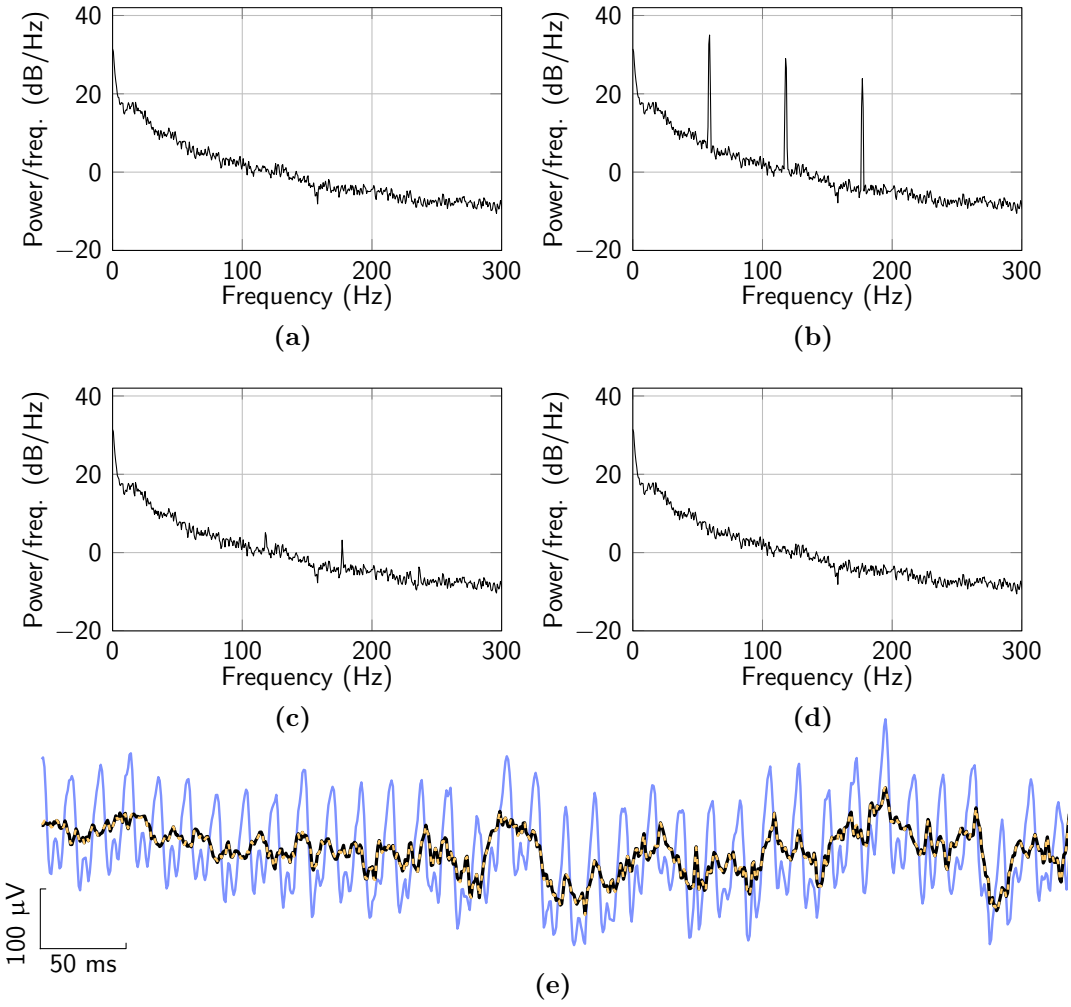

Figure 3.8: Testing result with real-time data. (a) PSD of the clean ECoG signal used for simulation. (b) PSD of the signal contaminated with synthetic interference with the fundamental frequency of 59 Hz, which served as the chip input signal (SNR_{in} = 0 dB). (c) PSD of the chip output signal (SNR_{out} = 28.9 dB). (d) PSD of the output from the full-precision reference model (SNR_{out} = 31.1 dB). The slight difference between (c) and (d) is due the precision loss in fixed-point implementation. (e) Plots of contaminated signal (——), chip output (——) and reference model output (- - - -), where the interference is removed, and the chip output accurately follows the reference model output.

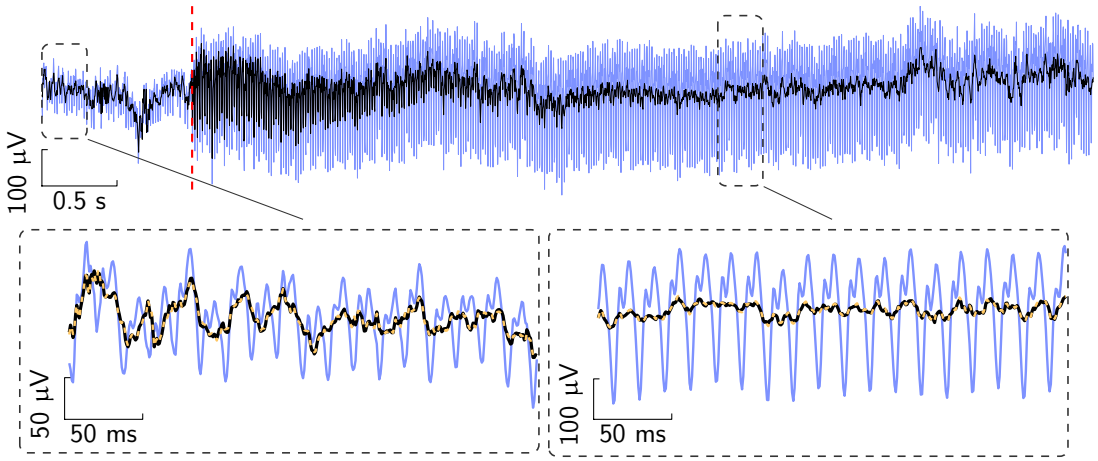

Figure 3.9: Response to a step change in interference power. The chip input signal (——) was synthesised by adding a synthetic interference, containing three harmonics, to a pre-recorded clean ECoG signal. The power of the increased at the time instance indicated by (- - -) ($\text{SNR}_{\text{in}} = 0$ dB before (- - -) and $\text{SNR}_{\text{in}} = -10$ dB after (- - -)). The plots show the contaminated input signal (——), the chip output signal (——) and the full-precision reference model output (- - - -), where the interference is removed, and the chip output closely follows the reference model output.

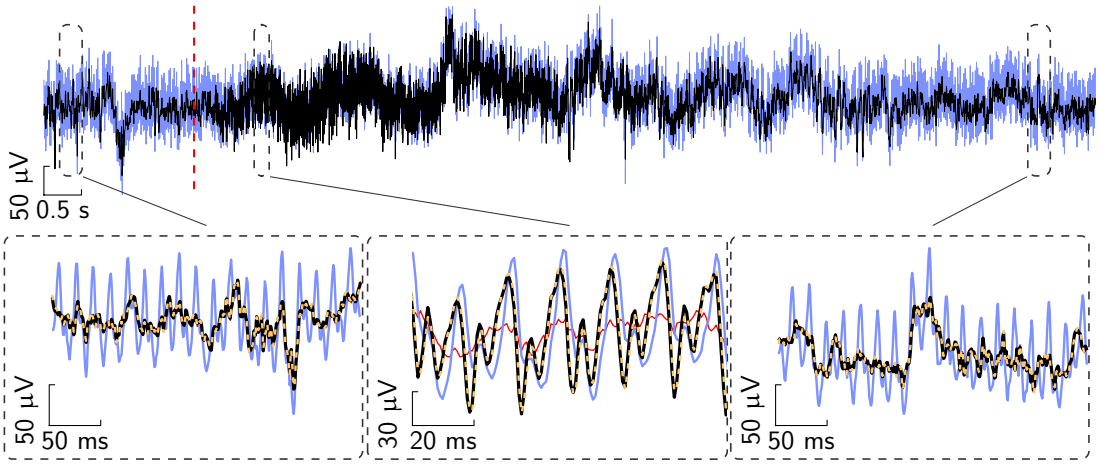

Figure 3.10: Response to a step change in the interference fundamental frequency. The fundamental frequency is changed from 60 Hz to 60.2 Hz at the time instance indicated by (- - -). The plots show the original signal (——), the contaminated signal serving as the chip input (——), the chip output signal (——) full-precision reference model output (- - - -). It can be seen that the chip output adapts to the change and closely follows the output of the reference model.

Table 3.3: Reference Model and Chip SNR_{out}

Test	SNR_{in} (dB)	SNR_{out} (dB)	
		Reference model	Chip
Test 1	0	31.1	28.9
Test 2	−10	29.5	27.8
Test 3	0	29.2	28.3

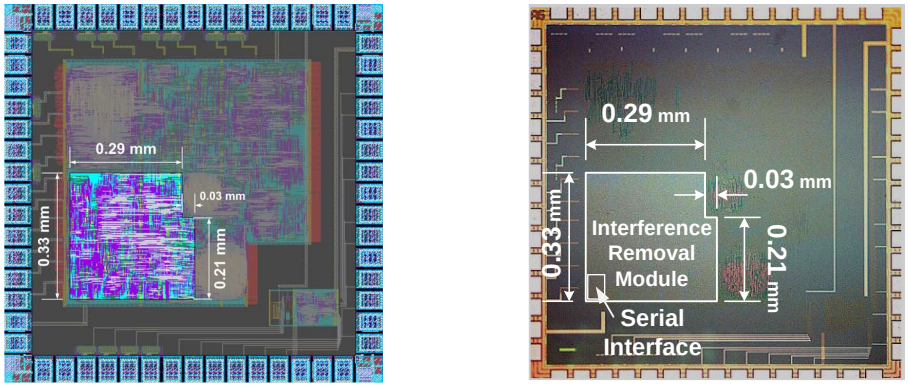

Figure 3.11: The chip layout (left) and die (right) photos. The area consumed by the interference removal module is approximately 0.11 mm^2.

output adapted to the frequency change and approached the actual signal.

The SNR_{out} values are displayed in Table 3.3. Slightly less SNR_{out} values of the chip output compared with the reference model output are due to the precision loss caused by fixed-point arithmetic used in the chip implementation.

Figure 3.11 shows the chip micrograph, where 0.11 mm^2 of area is consumed by the module. The specifications of the fabricated chip and a comparison with other works are presented in Table 3.4 and Table 3.5, respectively.

Table 3.4: Summary of Chip Specifications

Specification	Value
Technology	65-nm CMOS
Area	0.11 mm^2
Total Power	77 μW
Supply Voltage	1.2 V
Sampling Rate	1.25 kHz
Number of Harmonics	3
Number of Channels	1
Input/Output word-length	16 bits
Clock Frequency	7.5 kHz
Average SNR Improvement	30 dB

Table 3.5: Comparison with Other Works

	[61]	[62]	[63]	This Work
Tech.	FPGA	FPGA	0.18 μm	65-nm
Word-length	16-bit	16-bit	20-bit	16-bit
Gate Count	77 k	≈40 k	41 k	85 k
Power	106 mW	N/A	N/A	77 μW
SNR imprv.	29.9 dB	15 dB	N/A	31 dB
# Harmonics	1	1	4	3
Algorithm	LMS	ADALINE	LMS	Proposed—RLS
Reference	Yes	No	Yes	No
Results	Exp	Exp	Sim	Exp

3.6 Conclusion

In this chapter we presented the scalable VLSI architecture and ASIC of the previously proposed algorithm for power line interference cancellation in multi-channel biopotential recording. The proposed VLSI architecture is scalable for processing multiple channels and/or harmonics. The area is optimized through sequential implementation and resource sharing, while the throughput is enhanced through pipelining. The measurement results from a prototype implemented in a 65-nm CMOS showed an average SNR improvement of 30 dB, which was consistently achieved for input SNR of −20–20 dB and interference frequencies of 45–65 Hz for all modalities of biopotential signals.

Chapter 4

Unsupervised Spike Sorting Based on Discriminative Subspace Learning

4.1 Introduction

As discussed earlier, extracellular neural signals that are recorded by inserting microelectrodes into the brain tissue consist of LFPs, the action potentials (also called spikes) from a few surrounding neurons, and noise. For obtaining (multi-)unit activity, the signal is often filtered in 300–5000 Hz frequency band and spikes are identified through using a spike detection method. In many neuroscience studies, it is necessary to sort the spikes after detection so that the spikes generated by an individual unit fall into one cluster. The spike sorting stage is fundamental to the neuroscience studies which involves the analysis of spike rates, spike time synchrony, and inter-spike interval [16, 17].

Common spike sorting methods involve detecting neural spikes, extracting

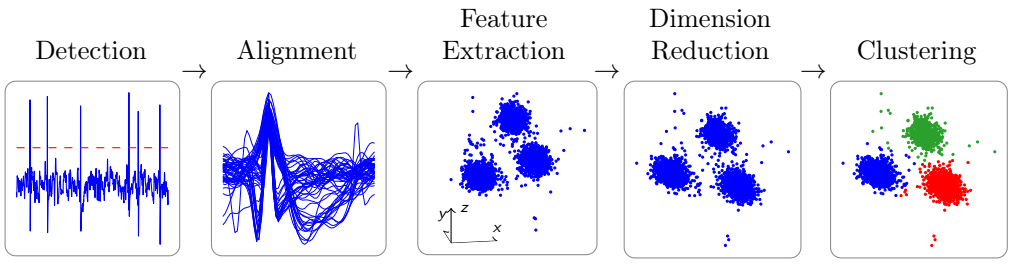

Figure 4.1: Spike sorting process.

and selecting features from detected spike waveforms, detecting the number of neurons, and assigning the spikes to their originating neurons [16]. The signal processing chain used for spike sorting is shown in Figure 4.1. For spike detection, methods based on absolute value thresholding, nonlinear energy operator, and wavelet have been widely used [17].

Among different feature extraction methods, principal component analysis (PCA) and discrete wavelet transforms (DWT) are most commonly used [16,17,64]. PCA projects the spikes to a set of orthogonal basis vectors that represent the largest variance of the data. In wavelet-based methods the spikes are decomposed into wavelets and the decomposition coefficients are used as features. A good feature extraction method should retain the most useful information for discriminating different spike shapes in a reasonably low dimension [17]. Some efforts have been done to come up with better feature extraction for spike sorting including methods based on waveform derivatives [21], Laplacian eigenmaps [22], wavelet optimization [23], and Fourier transform [24]. Although these methods can provide more discriminative features with reasonable dimension reduction, they do not necessarily seek for the most discriminative subspace for clustering [65]. Hence, the clusters which appear inherently separable in some discriminative subspace may overlap if projected using conventional features extraction methods. Such cluster overlaps increases the misclassification, may lead to incorrect

detection of number of clusters, hindering reliable clustering. Therefore, feature learning methods which seek for the most discriminative subspace is expected to provide an optimal cluster separation thus improving the clustering performance in spike sorting.

Depending on the proximity of an electrode to the surrounding neurons, the recording may contain several spike waveforms generated by different neurons. In practice, the number of the contributing neurons (i.e. clusters) is not known *a priori* and needs to be detected from the data. Incorrect selection of the number of the clusters heavily affects the performance of the clustering algorithm. One way to set the number of neurons is through visual inspection of spike waveforms by an expert observer. This method makes the spike sorting non-automatic and is prone to human error. It is generally desired that the algorithm detects the number of the clusters in an unsupervised manner so as to eliminate the user intervention [16, 17].

Automatic detection of the number of neurons often depends on the method used for clustering. Some common clustering methods used for spike sorting include k-means, mixture models, neural networks, superparamagnetic clustering (SPC), and self-organizing maps [16, 24, 66, 67]. Several approaches based on Bayesian/Akaike information criterion, gap statistics, and statistical test for Gaussian distribution have been proposed to learn the number of clusters for k-means and mixture model [68,69]. In SPC the number of clusters are detected by sweeping the temperature and choosing the clusters which are bigger than a certain size [67]. In general, various combinations of feature extraction and clustering have been applied to spike sorting with different levels of success [17, 64].

In this chapter, we propose algorithms for unsupervised spike sorting based on discriminative subspace learning to extract low dimensional and most discriminative features from the spike waveforms, and perform clustering. The core part

of the algorithms involves iterative subspace selection and clustering.

In the following sections, first the problem formulations are presented and a subspace learning method using linear discriminant analysis (LDA) and k-means is introduced. After that, a modification is proposed to the previous algorithm to make it able to detect and handle outliers including overlapping spikes. Further, another algorithm is proposed for discriminative divisive clustering which learns a hierarchy of 1-dimensional subspaces for clustering. We further introduce methods for selecting the number of neurons along with the algorithms

4.2 Robust discriminative subspace learning for spike sorting

Most of the feature extraction and dimensionality reduction techniques that have been used for spike sorting give a projection subspace which is not necessarily the most discriminative one. Feature extraction (followed by dimension reduction) is a crucial stage which determines the quality of the next stage clustering. Thus, feature extraction should transform the data in such a way that similar data points are close to each other while dissimilar ones are well-separated from each other.

4.2.1 Problem Formulation

Suppose the spikes are already detected through one of the common spike detection approaches, aligned to their peak, and stored in $X_{(m\times n)}$, where m is the number of samples stored for each spike waveform and n is the total number of the detected spikes. We are interested in finding a linear discriminative projection matrix $W_{(m\times d)}$ to transform the spike waveforms to a d-dimensional ($d < m$)

feature space $Y_{(d\times n)}$ as

$$Y = W^T X. \tag{4.1}$$

so that the potential clusters have maximum separability. Clustering is then performed in the subspace Y. For now, it is assumed that the number of clusters is known and is indicated by K; we will discuss the selection of K in Section 4.4.1 and Section 4.4.2. Let $L_{(n\times K)}$ be the cluster indicator matrix which assigns each data point (i.e. spike) x_i to its corresponding cluster $\mathcal{C}_\mathrm{k}$ so that $L_{i,k} = 1$ if $\mathbf{x}_i \in \mathcal{C}_\mathrm{k}$, and $L_{i,k} = 0$ otherwise. The cluster density and separability can be quantized by within- and between-class scatter matrices respectively defined as

$$S_\mathrm{w} = \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in \mathcal{C}_\mathrm{k}} (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T \tag{4.2}$$

$$= (X - ML^T)(X - ML^T)^T, \tag{4.3}$$

and

$$S_\mathrm{b} = \sum_{k=1}^{K} n_k \boldsymbol{\mu}_k \boldsymbol{\mu}_k^T = ML^T LM^T, \tag{4.4}$$

where μ_k is the center of $\mathcal{C}_\mathrm{k}$, $M = (\mu_1, \cdots, \mu_K)$, and n_k is the number of points in $\mathcal{C}_\mathrm{k}$. To achieve a high cluster separability, the within-class scatter should be small and/or the between-class scatter should be large. This leads to the following optimization problem:

$$\max_{W,L} \mathrm{Tr} \frac{W^T S_\mathrm{b} W}{W^T S_\mathrm{w} W}. \tag{4.5}$$

4.2.2 Discriminative Subspace Learning using LDA and k-means

When data labels (i.e. L) are known, LDA seeks for the most discriminative linear subspace for reducing the dimensionality of the data. Given the within-class scatter (S_{w}) and the between-class scatter (S_{b}) matrices, the optimization problem in (4.5) forms a generalized eigenvalue problem:

$$S_{\mathrm{b}}W = \psi S_{\mathrm{w}}W, \tag{4.6}$$

where ψ is a matrix of eigenvalues. The optimal W is given by the d eigenvectors corresponding to the largest d eigenvalues. It should be noted that the rank of S_{b} is $K-1$ where K is the number of classes in the data; thus, the dimensionality of LDA subspace is at most $K-1$.

When the clustering subspace is selected (W is known), a clustering method such as k-means can be used to obtain the labels. It can be shown that k-means specifically solves (4.5) when W is fixed [65]. Thus, the general problem in (4.5) may be solved through alternatively solving for W and L.

With W fixed, (4.5) becomes a k-means clustering in feature space Y; with L fixed, it can be solved for W by LDA. A straightforward method is to fix W (initialized by PCA) and use k-means to obtain L, and using the updated L then perform LDA to update W and iterate this procedure until convergence (i.e. L remains the same between two iterations). This method is referred to as *LDA-Km* through which, the data are simultaneously clustered while the discriminative subspaces are selected [65]. Figure 4.2 shows the flowchart of this method, with added stage for shepherding of Y before k-means, for better clustering.

Since k-means is used as a building block of the algorithm, the limitations

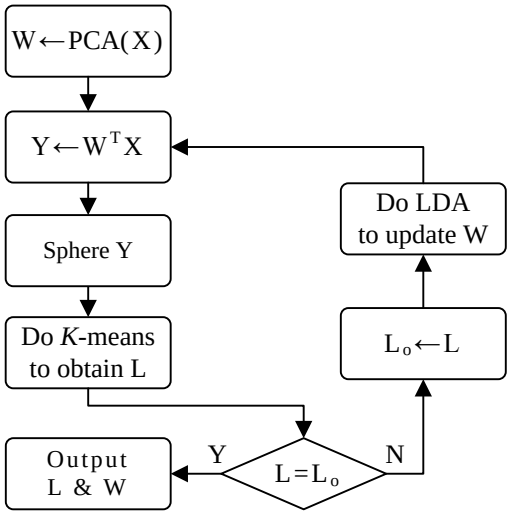

Figure 4.2: Flowchart of the subspace learning method using LDA and k-means.

inherent in k-means such as local convergence and sensitivity to initialization also exist here. However, various methods exist to mitigate these problems such as repeating k-means clustering with different initializations and picking the one which leads to the least intra-cluster distance [69].

4.2.3 Discriminative Subspace Selection through Mixture model learning with outlier handling

The refractory period forces at least 1 ms delay between successive spiking activity of a neuron. However, the recording from a single electrode is the superposition of signals from a few nearby neurons that are assumed to be firing independently. As such, it is likely for two or more neurons to fire at or around the same time, making their spike waveforms overlap in the recorded signal.

Ideally, these overlapping spikes should be detected and resolved in such a way that the (two or more) neurons contributing to the overlap be identified. In a less ideal case, at least, the overlapping spikes should be identified as outliers (or noise) and not to be assigned to any neurons.

Due to the corrupted waveform shape of overlapping spikes, samples from

these events do not usually fall into any actual cluster in the feature space. However, this may not be the case when the feature subspace is learned through conventional subspace learning methods, such as the one described in Section 4.2.2, that do not account for outliers. Furthermore, the presence of outliers may cause the algorithm to select a suboptimal subspace for clustering. This is due to the fact that the algorithm would force the outliers to be assigned to true clusters and then would try to make the clusters as compact as possible (i.e. to minimize the intra-cluster distance), which is expected to lead to a different (and suboptimal) subspace from the case that no outlier exists. Thus it would be desirable to detect and exclude the outliers during subspace learning.

In this section, we propose a new discriminative subspace learning method which can handle outliers in data. The method uses Gaussian mixture model (GMM) learning for clustering with outlier handling and LDA for subspace selection. The new method is then used for spike sorting.

The method presented in Section 4.2.2 may become ineffective if considerable number of outliers is present in the data. We are interested in handling outliers and improving the convergence in subspace learning. One way to tackle this problem is to incorporate a clustering scheme that can identify and exclude outliers in each iteration of subspace learning. That said, the clustering method chosen should match the inherent properties of spike features obtained through LDA, that tries to form dense spherical and linearly separable clusters. Clustering based on GMM seems to be a reasonable choice for this purpose for the following reasons. First, GMM can adequately model spherical (Gaussian) shaped clusters that are expected to be obtained by LDA in the subspace selection stage. Second, through using GMM, the outliers can be identified by learning an additional component—leading to $K + 1$ Gaussian mixtures—with small prior and large variance. This additional component will serve as an 'outlier collector' whose

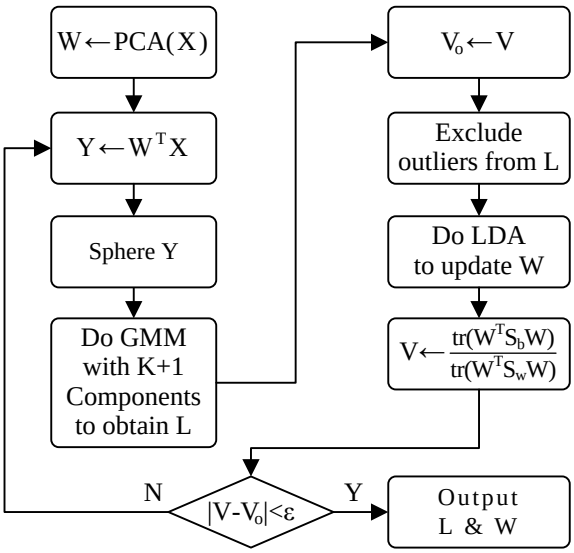

Figure 4.3: Flowchart of the subspace learning with outlier handling using LDA and GMM.

samples—in each iteration—are discarded in subspace selection using LDA. Third, GMM better handles clusters with unbalanced densities. This facilitates the extraction of small clusters formed by sparsely firing neurons. Forth, although the iterative subspace learning tends to form spherical clusters, some clusters may be more spread in a particular direction due to electrode drifts. GMM would be expected to capture such clusters better than k-means.

Figure 4.3 shows the flowchart of the proposed subspace learning method named as LDA-GMM. The method is similar to LDA-Km (Section 4.2.2), except that GMM with one extra component, which is meant to collect outlier samples, is used for clustering. The outliers are then excluded from the calculation of W by LDA. In each iteration, after obtaining W, the whole data samples (including the outliers) are projected to the feature subspace using W. The stopping criteria is the change between current and previous trace ratio value $\mathrm{Tr}(W^T S_\mathrm{b} W)/\mathrm{Tr}(W^T S_\mathrm{w} W)$. This value indicates that during one iteration how much improvement is made in discriminating the clusters. If the change in improvement is less than a small value ϵ, the algorithm stops and return the

current labels L and projection matrix W.

4.3 Detecting the Number of Neurons

Automatic detection of the number of neurons often depends on the method used for clustering. In this section, a method is proposed to detect the number of the clusters in the aforementioned subspace learning method.

It is desired to discover in the data as many potential clusters as possible while avoiding over-clustering. Simultaneous subspace learning and clustering through either LDA-Km or LDA-GMM with different values of K leads to different feature spaces. We look for a clustering with the biggest K in which the clusters are properly well-separated that is having no over clustering.

As so, we keep increasing K and perform subspace learning. For each K, the clusters are checked to detect whether over-clustering has occurred based on the following assumption: If two given clusters are formed from samples of two different actual clusters, their projection on the vector connecting their centroids would have a distribution far from the unimodal distribution. Likewise, if the two clusters are formed from samples of a single actual cluster in the data, their projection on the vector connecting their centroids would have an almost unimodal distribution. This assumption is plausible in conjunction with the subspace learning method presented earlier. This is due to the fact that given a fixed K, LDA-Km or LDA-GMM try their best to separate K clusters as far apart as possible, which translates to a multimodal distribution of clusters projected on the vectors connecting their centroids. However, when two or more clusters are formed from a single actual cluster (i.e. over-clustering), they tend to be joint and not well-separated from each other which translates to an almost unimodal distribution of clusters projected on the vectors connecting their centroids.

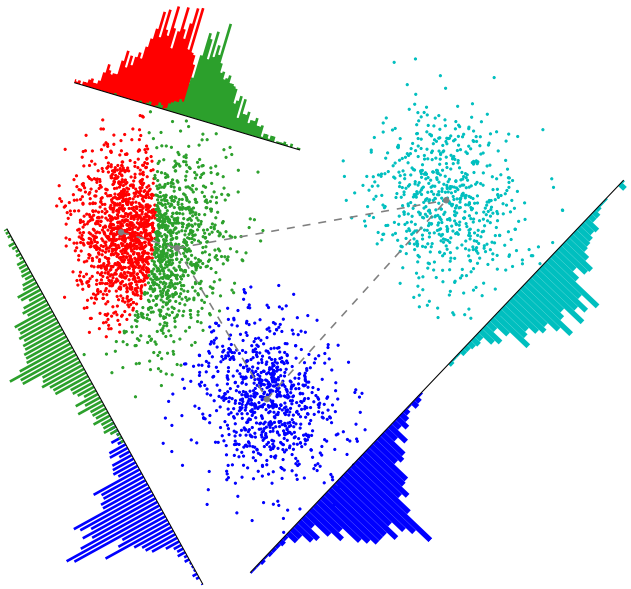

Figure 4.4: Projection of clusters (obtained through LDA-GMM on *in-vivo* data d1122101:1) onto vectors interconnecting their centroids. The well separated clusters result in higher scores in normality test on the projected data. Cluster pairs with test scores below the threshold are merged together.

Recall that the cluster centroids are represented by $\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_K$, and L denotes the cluster indicator matrix. The cluster projection on vectors connecting their centroids for each pair of cluster is given by

$$X'_{i,j} = \{\mathbf{x}_k | \mathbf{x}_k \in \mathcal{C}_\mathrm{i} \vee \mathbf{x}_k \in \mathcal{C}_\mathrm{j}\}\,,$$

$$\mathbf{Q}_{i,j} = (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^T X'_{i,j}, \tag{4.7}$$

$$S_{i,j} = \mathcal{T}(\mathbf{Q}_{i,j}), \tag{4.8}$$

where $X'_{i,j}$ a matrix containing the samples from both clusters i and j, and $\mathbf{Q}_{i,j}$ is a 1-D vector containing the projected samples onto the vector connecting the centroids of cluster i and j. $\mathcal{T}$ denotes a test of unimodality, and $S_{i,j}$ is the test score. A DIP test of unimodality [70] is suggested for $\mathcal{T}$ since it directly measures multimodality in a sample. However, other similar similar tests including Lilliefors test [71], and Anderson-Darling test [72] may be used with comparable results.

Figure 4.4 shows sample clusters and histogram on their projections.

For detecting whether the two clusters i and j, are the result of over-clustering, the test score $S_{i,j}$ is examined. If $S_{i,j}$ is less than a certain threshold, the two clusters i and j are supposed to be originated from a single actual cluster, otherwise they are correctly assigned as two clusters. $S_{i,j}$ is calculated for $i = 1, \cdots, K \quad j > i$ to obtain the number of actual clusters denoted by $K' \leq K$.

As mentioned, we are looking for a clustering with the biggest K in which the number of the actual clusters (i.e. K') is equivalent to (or not less than) K. As so, we keep increasing K and do the subspace learning until $K' \leq K$ and K' remains unchanged for $K+1$ (no well-separated cluster for $K+1$). Then, the largest value of K' indicates the true number of clusters (i.e. neurons).

4.4 Unsupervised Spike Sorting Algorithms

Based on the proposed methods for subspace learning and detection of the number of neurons, two unsupervised spike sorting algorithms are proposed.

4.4.1 Proposed Algorithm I

The first algorithm is based on the direct application of LDA-Km or LDA-GMM to learn the spike features and sort the spikes along with the method in Section 4.3 to detect the number of neurons. This method is summarized in Algorithm 4.1.

The benefit of this method compared with Algorithm I is that 1-dimensional subspace is used for clustering which leads to a faster convergence and better handling of clusters with different densities. Since this is a hierarchical clustering approach, a drawback is that misclassified samples in the top levels will affect the clustering results in subsequent levels. The summary of the proposed algorithm is provided in Algorithm 4.2.

Algorithm 4.1: Spike Sorting using Discriminative Subspace Learning

Input: $X_{m,n}$: n spikes, m samples for each spike
Initialize: $K \leftarrow 1$, $K'' \leftarrow 0$;
repeat
  $K \leftarrow K + 1$;
  do LDA-Km or LDA-GMM on X to obtain W and extract K clusters L^K ;
  obtain the true number of clusters K' using the method in Section 4.3 ;
  $K'' \leftarrow K'$;
until $K'' = K' < K$;
Cluster X using $L^{K'}$;

4.4.2 Proposed Algorithm II

Here, we propose a clustering algorithm based on divisive clustering scheme and use it for spike sorting. In this method, the data are initially partitioned into two clusters (using LDA-Km) in the most discriminative 1-dimensional subspace. The resultant clusters are marked as *child*. After that, the distribution of samples in the obtained subspace is tested for unimodality using DIP test (or similar tests). If the test score is higher than a predefined threshold, which indicates that the distribution is multimodal, each of the *child* clusters is again partitioned into two clusters and marked as *child*, otherwise they are merged into one cluster and marked as *final*. The same process is then repeated for each *child* cluster until all the clusters are marked as *final*. In this process, clusters with sizes smaller than a predefined value, or sparse samples that lie far from cluster centers, can be neglected and assigned as outliers.

It should be noted that the threshold controls how close the distribution of the clusters should be to unimodal distribution to be considered as *final*. In general, a larger threshold value causes the algorithm to extract more spread clusters, whereas a smaller value results in the extraction of more compact and smaller clusters which may lead to over-clustering.

Algorithm 4.2: Spike Sorting using Discriminative Divisive Clustering

Input: $X_{m,n}$: n spikes, m samples for each spike
Define: $\mathcal{S} = \{S_1, S_2, \cdots, S_I\}$;
Initialize: $K \leftarrow 2$, $\mathcal{S} \leftarrow \{X\}$;
repeat
  foreach $S_i \in \mathcal{S}$ **do**
    do LDA-Km on S_i to extract K *child* clusters (S_i^j);
    if *AD test score on* S_i $<$ *Threshold* **then**
      Create a *final* cluster from S_i
    else
      foreach S_i^j **do**
        if *Size(*S_i^j*)* $>$ *Minimum Cluster Size* **then**
          Add $\{S_i^j\}$ to $\mathcal{S}$
        else
          Assign S_i^j to the Outliers Cluster
      Remove S_i from $\mathcal{S}$;
until *All the spikes are assigned to clusters*;

4.5 Results

Evaluation of spike sorting algorithms is often challenging due to the lack of ground truth [16, 17]. A common approach, however, is to use synthetic datasets, which would provide the ground truth as well as the flexibility to evaluate different characteristics of the spike sorting algorithm. Synthetic neural data is usually prepared by replicating spikes from a few spike waveform templates and adding them to a background noise mimicking neural noise, with the spike arrival time distributions similar to those of real spike trains. In this work, we evaluate our algorithms using synthetic data and compare the results to two other spike sorting methods. Furthermore, we present clustering results using synthetic and *in-vivo* data, in which case the performance of the algorithms can be qualitatively assessed by looking into cluster separation.

We compared the two proposed algorithm with PCA-$k$means and `Wave_clus` based on criteria such as accuracy of spike sorting, dimensionality of feature

space for effective clustering, and sensitivity to noise.

In PCA-$k$means, PCA is used for feature extraction and dimensionality reduction, and k-means is used for clustering. This method is supervised meaning that the number of clusters (k in k-means) must be known, and parametric assuming the clusters are spherical in the PCA space. `Wave_clus` is an spike sorting method which uses DWT for feature extraction and SPC for clustering. It has been shown very practical for spike sorting since it can automatically choose the number of clusters (unsupervised), and does not impose any assumption on the shape of the clusters (nonparametric). In this method, Haar wavelet coefficients are used as features where the 10 coefficients with largest deviation from normality were selected as the input to the SPC algorithm. In many situations, SPC clusters a subset of the spikes and the remaining spikes are forced to be classified using a KNN classifier.

4.5.1 Synthetic Data with Ground Truth

For simulation with synthetic data, we have used eight challenging datasets provided by [67], which have been widely used in the literature to benchmark spike sorting algorithms. The datasets `C_difficult1*` and `C_difficult2*` are referred to as Set 1 and Set 2, respectively. Each dataset contains spikes from three different neurons. Simulations are carried out for different noise levels and feature dimension of 2–10 for PCA-$k$means and `Wave_clus`; feature dimension was 2 for Algorithm I, and 1 for Algorithm II. The number of clusters is manually set to 3 for PCA-$k$means, while `Wave_clus` and the proposed algorithms detect it automatically. Throughout our simulations, we used the default parameters of `Wave_clus` and chose the best results on multiple runs of the algorithm. `Wave_clus` correctly detected the number of clusters for all cases except for Set $2\backslash\sigma_{\mathrm{n}}$=0.2 where it only detected 2 clusters. The proposed algorithms detected

Table 4.1: Comparative Results on Synthetic Data

Dim.	Set	PCA-$k$means (%)				`Wave_clus` (%)				Algorithm I (%)			
σ_n:		0.05	0.1	0.15	0.2	0.05	0.1	0.15	0.2	0.05	0.1	0.15	0.2
2	1	93.2	75.2	50.8	40.9	97.7	89.0	63.1	62.2	99.6	99.4	99.1	99.2
	2	81.1	74.0	65.8	56.7	66.0	52.6	50.7	49.6	98.7	98.9	98.7	98.2
4	1	97.9	84.7	64.7	53.5	98.7	96.9	93.0	86.5				
	2	91.8	80.8	72.7	63.3	93.3	98.0	91.0	58.0	Algorithm II† (%)			
10	1	98.9	95.1	85.5	76.8	98.4	98.95	95.4	86.5	98.1	99.2	99.1	98.7
	2	97.8	91.3	83.0	60.6	97.3	98.4	91.7	58.9	98.6	98.7	98.8	98.3

†Feature dimension is 1.

the true number of clusters in all the cases. The results are shown in Table 4.1, where the first column is the dimension of the feature space, and the first row is the standard deviation of the noise (σ_n). Overlapping spikes are excluded in calculating the accuracies. It can be seen that the accuracy of PCA-$k$means and `Wave_clus` degrades as the dimension of the features reduces. `Wave_clus` (with forced clustering) performs better than PCA-$k$means especially when the noise level is high; however, the performance of both method degrade significantly as the noise level increases. Interestingly, the proposed methods give significantly higher sorting accuracies compared with the other methods and they are highly robust to noise.

Figure 4.5 shows the feature extraction and clustering results on the most challenging dataset Set 2\σ_{n}=0.2 (`C_Difficult2_noise02`). As can be seen in Figure 4.5(a)–(c), the 2-D subspace learned by Algorithm I provides significantly better separability of clusters compared with PCA and DWT. Figure 4.5(d) shows the average waveforms of spikes sorted by Algorithm I (upper) and Algorithm II (lower).

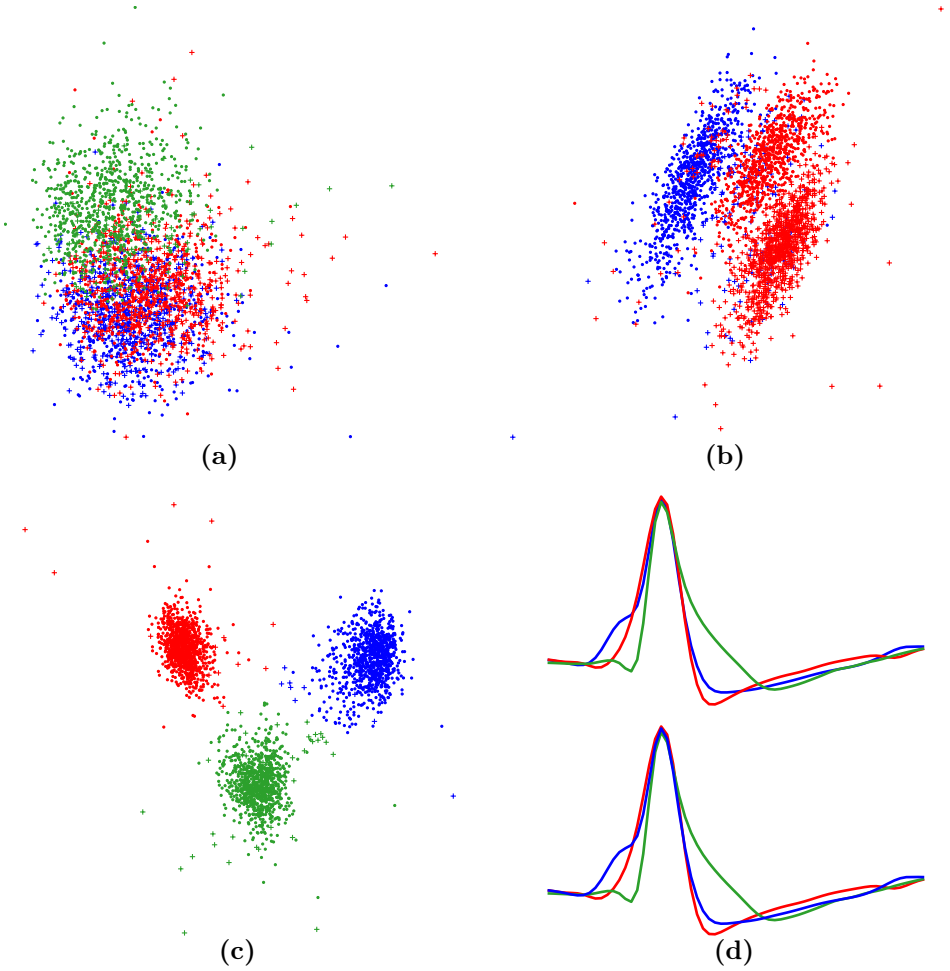

Figure 4.5: Results on synthetic data (`C_Difficult2_noise02`). (a) PCA-$k$means where feature dimension is 10 (projection on the first two principal components is shown), and K is manually set to 3. Accuracy = 60.6%. (b) `Wave_clus` where feature dimension is 10 (best 2-D DWT projection is shown); SPC fails to identify the third cluster for any temperature value. Accuracy = 58.9%. (c) Proposed Algorithm I where feature dimension is 2 (most discriminative projections). The 3 clusters are automatically identified. Accuracy = 98.2%. Misclassified spikes are shown as '+'. (d) Upper plot: Average spike waveforms of the clustering in (c). Lower plot: Average spike waveform of the sorted spikes using Algorithm II. Accuracy = 98.3%.

4.5.2 Comparison on *in-vivo* Data

The performance of the proposed algorithms is also tested on *in-vivo* data. The data is recorded from the rat hippocampus and is publicly available [73]. A sample recording `d1122101:1` is used for illustration. The spikes were extracted by negative amplitude thresholding at 3 RMS, and aligned to their peak values. In total, 4541 spikes were detected and 64 samples were stored for each spike. Figure 4.6(a) shows the PCA features on the first two principal components, where only 1 cluster can be identified. Figure 4.6(b) shows the best 2-D projection of wavelet features used in `Wave_clus`, where 2 clusters are identified by the algorithm (average waveforms in Figure 4.6(d)). Figure 4.6(c) shows the results of clustering using Algorithm I, where 3 clusters are detected (average waveforms in Figure 4.6(e)). Algorithm II also detected 3 clusters whose average spike waveforms are shown in Figure 4.6(f) with the corresponding inter-spike interval histograms in Figure 4.6(g) which indicate the qualitative validity of the extracted clusters. Figure 4.7 shows the hierarchy of 1-D features learned by the proposed hierarchical clustering algorithm (on dataset `d1122101:1`).

Regardless of the clustering algorithm used, the methods that rely on PCA for feature extraction usually fail to identify clusters with highly overlapping features in the PCA space, thus assigning spikes generated from different neurons to the same cluster [64,67]. We have observed this in several *in-vivo* recordings when the spike waveforms of different neurons are very similar. DWT has been shown more effective for spike feature extraction which provides better separability of clusters. However, similar to PCA, it does not necessarily provide the most discriminative feature subspace for clustering, and may fail to correctly discriminate spikes with similar spike waveform when the noise level is high.

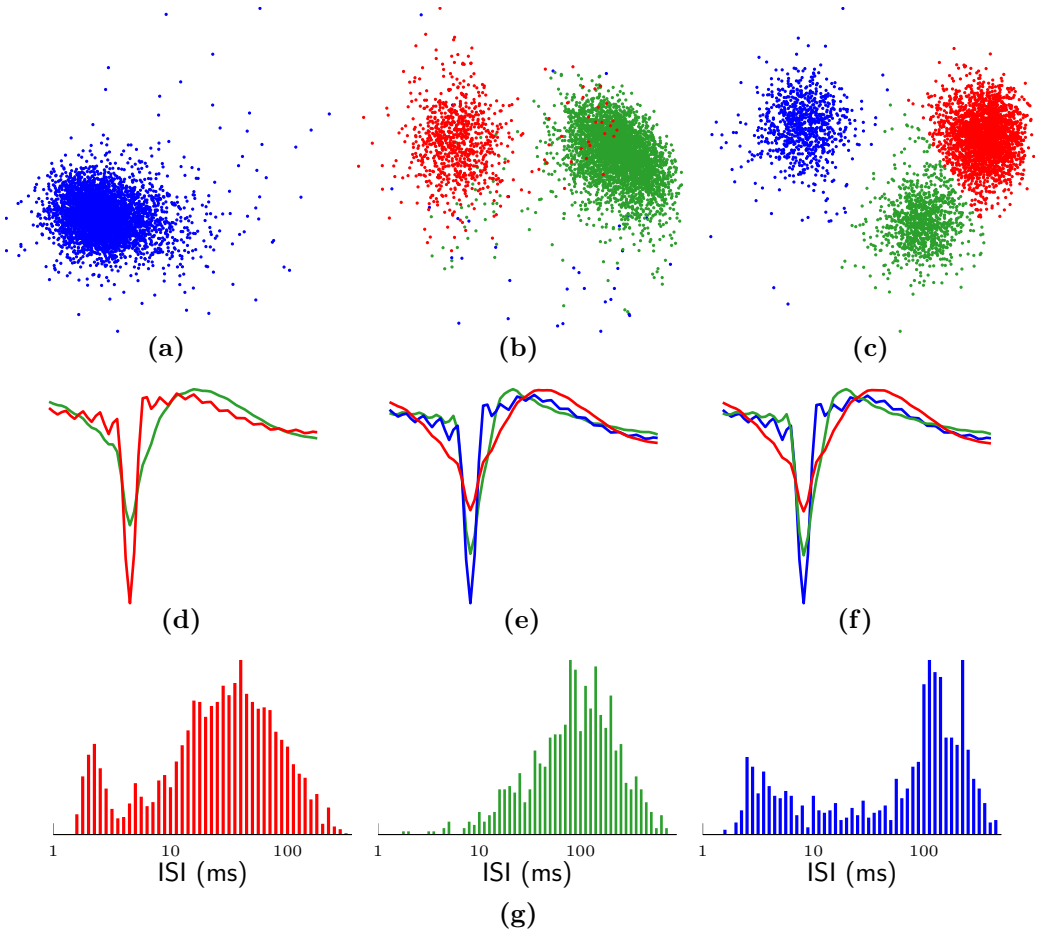

Figure 4.6: Comparative results on real data recorded from the rat hippocampus (`d1122101:1`). (a) Projection on the first two principal components. The clusters cannot be identified through PCA. (b) Best 2-D projection of DWT features used in `Wave_clus` where feature dimension is 10. Only two clusters are identified (points in blue are assigned as outliers). (c) Discriminative features learned by Algorithm I, the three clusters are automatically identified. (d) and (e) show the average spike waveforms of the clustering in (b) and (c), respectively. (f) Average spike waveform of the sorted spiked using Algorithm II. The three clusters are automatically identified and the results are highly similar to those of Algorithm I. The histogram of the log inter-spike interval of spikes sorted by Algorithm II, indicating the validity of the extracted clusters. Colors correspond to the spikes in (f).

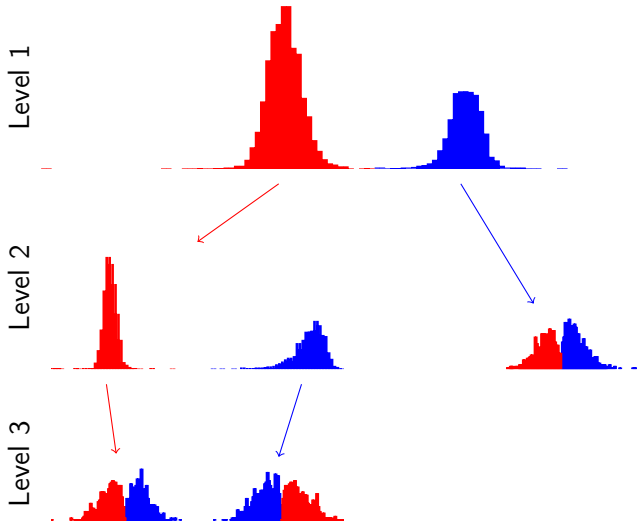

Figure 4.7: Example of the proposed hierarchical discriminative divisive clustering method on spike waveforms. In each level the clusters obtained from the previous level are separated into two clusters until the resultant clusters are almost unimodal (determined by the test of unimodality). It can be seen that the second cluster in Level 2 and the two clusters in Level 3 are almost unimodal (with the test score below the threshold), thus the algorithm stops at Level 3 and exports 3 clusters.

4.5.3 Comparison on Feature Extraction

Majority of the spike sorting methods in the literature are based on a conventional clustering procedure which involves separate stages of feature extraction and clustering [16]. In these methods, feature extraction is of crucial importance, and highly determines the overall quality of spike sorting. As discussed in Section 4.1, a good feature extraction method should discriminate the clusters originating from different spike waveforms. Thus, it is of interest to compare features extraction by the proposed algorithms with that of the other popular methods. Along this line, we have compared features extracted by popular methods such as PCA, discrete derivatives and PCA (DD-PCA), DWT, and laplacian eigenmaps (LE) with the proposed methods LDA-Km and LDA-GMM.

Figure 4.8 and Figure 4.9 show the feature extraction results on synthetic datasets `C_difficult1*` and `C_difficult2*`, respectively. It should be noted

that since different methods lead to different cluster shapes and separability, the suitable clustering algorithm should be selected based on the feature extraction method used. For example, with PCA features, clustering methods such as k-means [16], GMM [16], EM with t-distributions [74], density grid contour clustering [75], spectral clustering [76], subtractive clustering [77], Kohonen network [78], and fuzzy c-means clustering [79] have been used. Similar clustering methods may be used for DD-PCA as well. With DWT based features, clustering methods including k-means [16], SPC, [67], and variational Bayes clustering using t-distributions [80] have been applied. With LE features, k-means clustering have been used [22].

It can be seen in Figure 4.8 and Figure 4.9 that as the noise level increases, the clusters originating from different neurons mix into each other, which in turn degrades the clustering performance. This is because noise may highly interfere with the measure based on which these methods seek the projection subspace, such as variance or laplacian. However, the performance of the proposed methods is highly robust to increasing level of noise because of the objective function used. Although noise increases the final within-cluster scatter and decrease the between-cluster scatter, the optimization process itself is not much affected by high levels of noise.

Figure 4.10 shows the feature extraction results on *in-vivo* data. We chose challenging recordings for which the PCA features do not provide a good cluster separation. LE provides a slightly better separation than PCA, and DD-PCA outperforms the both. Significantly outperforming other methods, LDA-Km and LDA-GMM learned the best 2-D projection and extracted three clusters which are clearly separated. In addition, LDA-GMM identified the possible outliers.

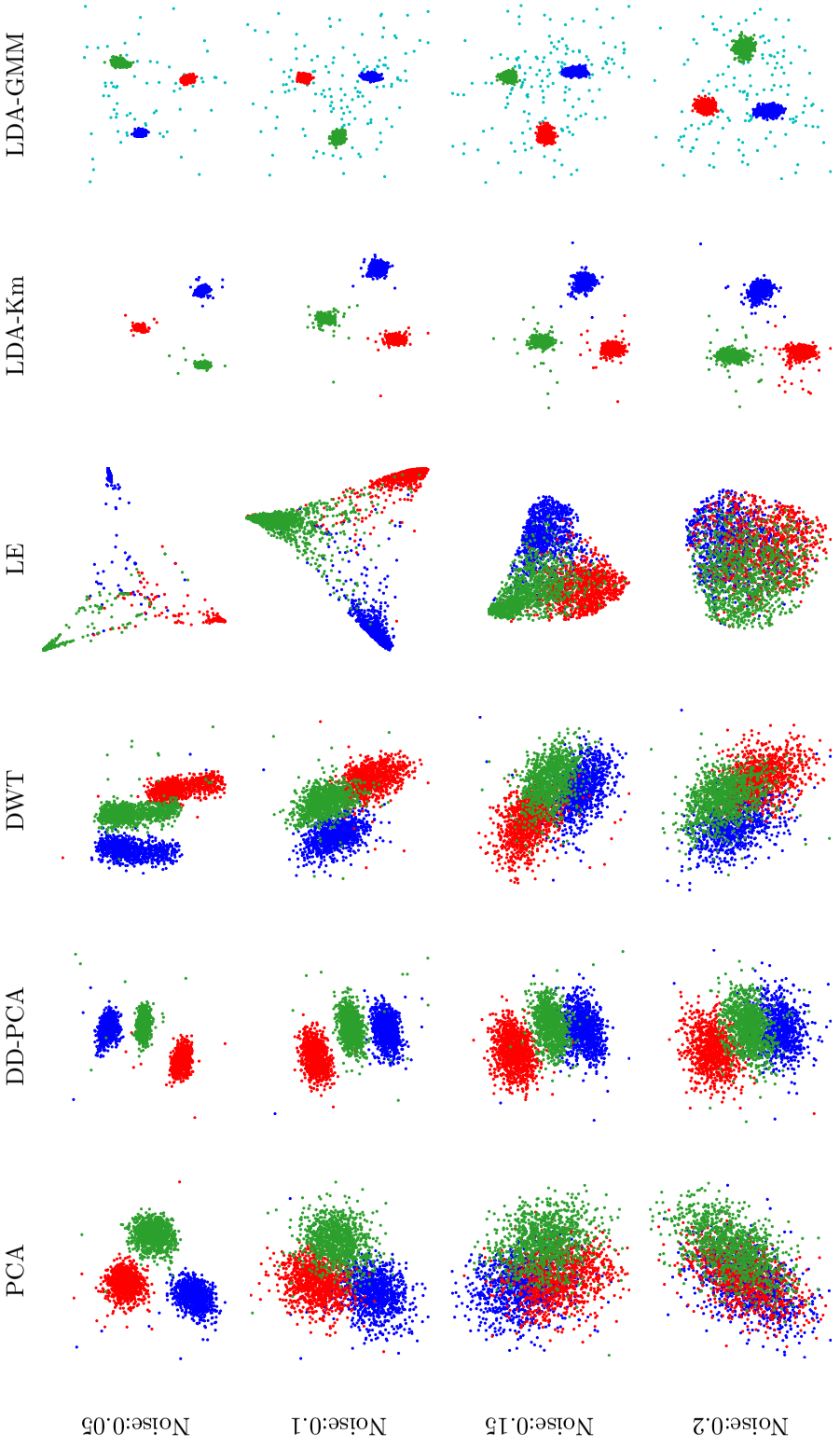

Figure 4.8: Comparison of different spike feature extraction methods on dataset `C_difficult1*` at different noise levels. The coloring is based on the ground truth, except for LDA-Km and LDA-GMM where it is based on the clustering results.

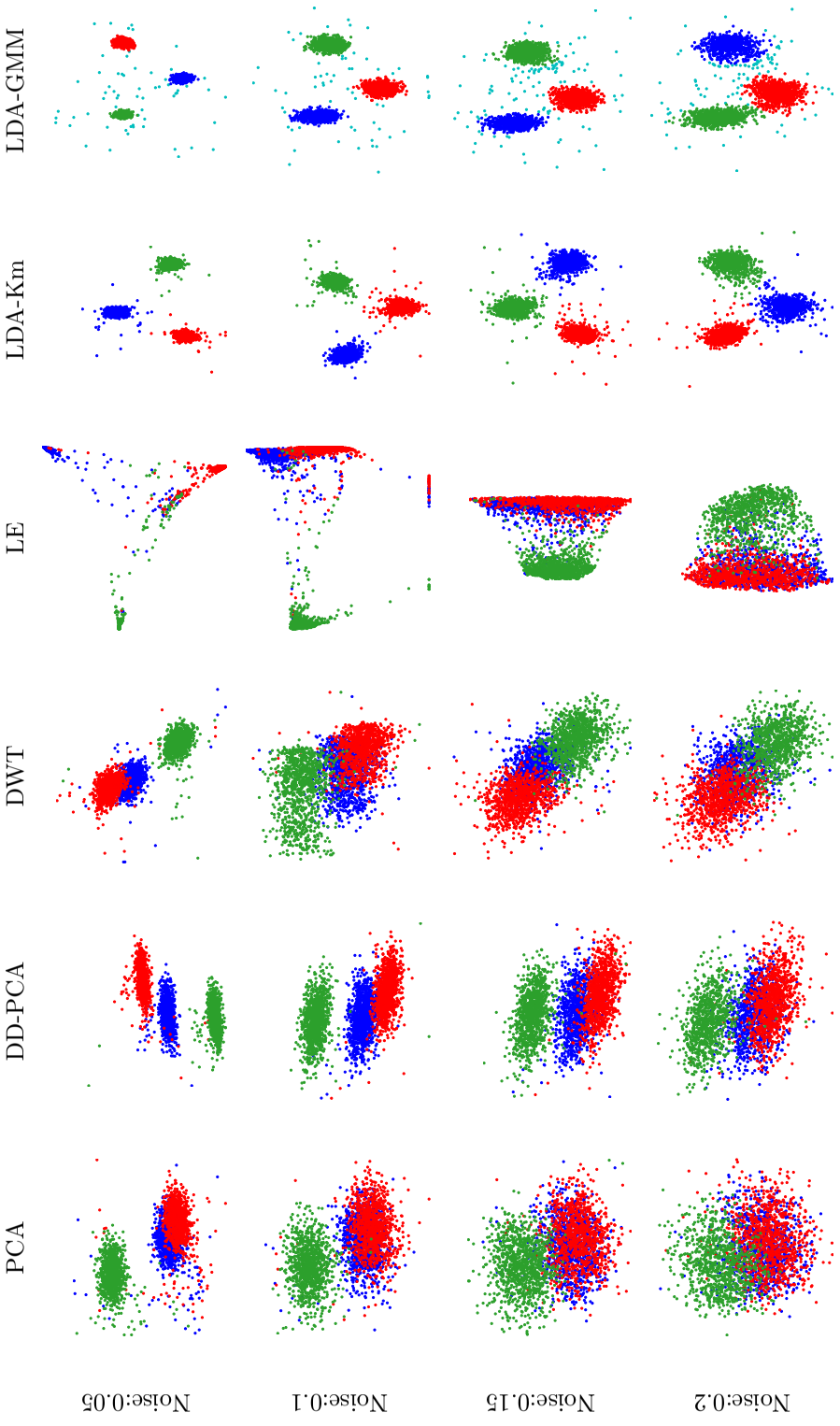

Figure 4.9: Comparison of different spike feature extraction methods on dataset `C_difficult2*` at different noise levels. The coloring is based on the ground truth, except for LDA-Km and LDA-GMM where it is based on the clustering results.

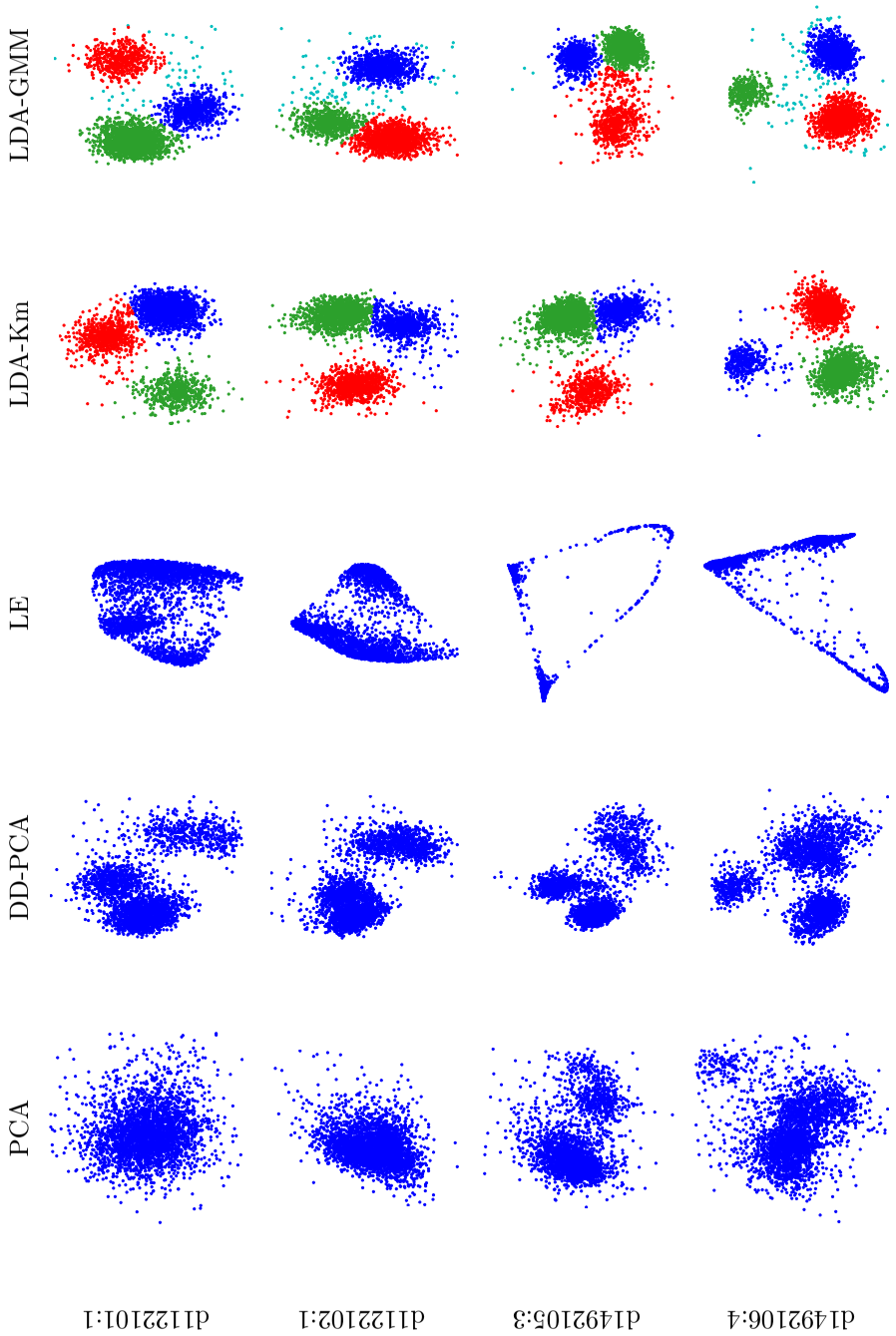

Figure 4.10: Comparison of different spike feature extraction methods on *in-vivo* data recorded from rat hippocampus.

4.5.4 Overlapping Spikes and Outliers

As discussed in Section 4.2.3, overlapping spikes, artefacts and noise usually appear as outliers in the data. Incorrectly classifying outliers to the actual spike clusters may lead to inaccurate analysis of spike trains. It is thus important to identify the outliers and exclude them from the analysis. Furthermore, in subspace learning considerable amount of outliers may lead to learning a suboptimal feature subspace. Here, we illustrate the effect of outliers and compare the performance of LDA-Km and LDA-GMM in terms of handling outliers.

To illustrate the effect of outliers in the performance of subspace learning methods, we applied the LDA-Km and LDA-GMM methods on sample data with and without outliers. The results are shown in Figure 4.11. It can be seen

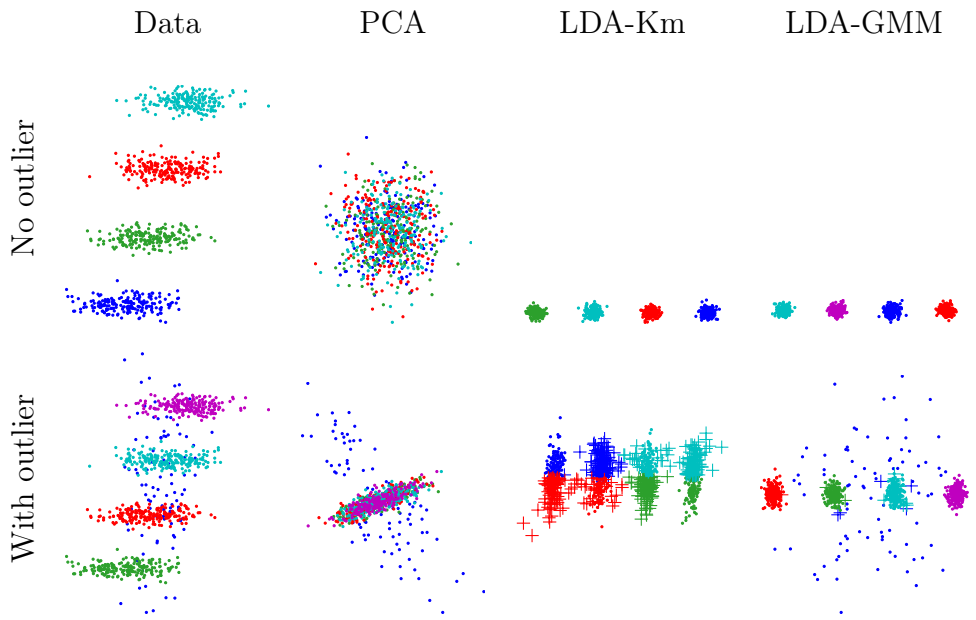

Figure 4.11: Sensitivity of the two subspace learning methods to outliers. The dimension of the data is 3. In both cases, PCA does not provide the proper subspace for clustering. When there is no (or negligible number of) outliers in the data, LDA-Km and LDA-GMM give similar results . However, when considerable number of outliers exist in the data, LDA-Km may not give the proper discriminative subspace, whereas, LDA-GMM is more robust to outliers and can adequately learn the optimal subspace for cluster discrimination.

that, PCA does not give a good low dimensional projection for this data since the direction of maximum variation is significantly different from the direction giving the maximum discriminability of clusters. Both LDA-Km and LDA-GMM algorithms give similar subspaces and clustering when no outliers exist in the data. However, when the data are contaminated with considerable number of outliers, LDA-Km does not lead to a proper clustering.

Figure 4.12 (top) shows the evolution of three clusters in subspace learning using LDA-Km. The algorithm starts from PCA subspace (iteration 1), and refine the projections in each iteration. A portion of the neural recording is also displayed in Figure 4.12 (bottom) with spike colors referring to the clusters in Iteration 15. It can be seen in the second and fourth zoom-in pictures that the overlapping spikes are incorrectly assigned to different clusters. This is a major limitation in LDA-Km algorithm which is addressed in LDA-GMM.

Similarly, Figure 4.13 (top) shows the evolution of the clusters in subspace learning using LDA-GMM on the same data sequence as in Figure 4.12. The outliers are displayed in blue. It can be seen in Figure 4.13 (bottom) that the overlapping spikes are correctly classified as outliers and are separated from other true clusters.

4.6 Conclusion

In this chapter, we have presented methods for unsupervised spike sorting based on simultaneous discriminative subspace learning and clustering. A previously proposed method namely LDA-Km has been successfully applied to spike feature learning with superior performance over traditional methods. However, since this method cannot handle overlapping spikes and outliers, a new method based on LDA for subspace selection and GMM for clustering (namely LDA-GMM)

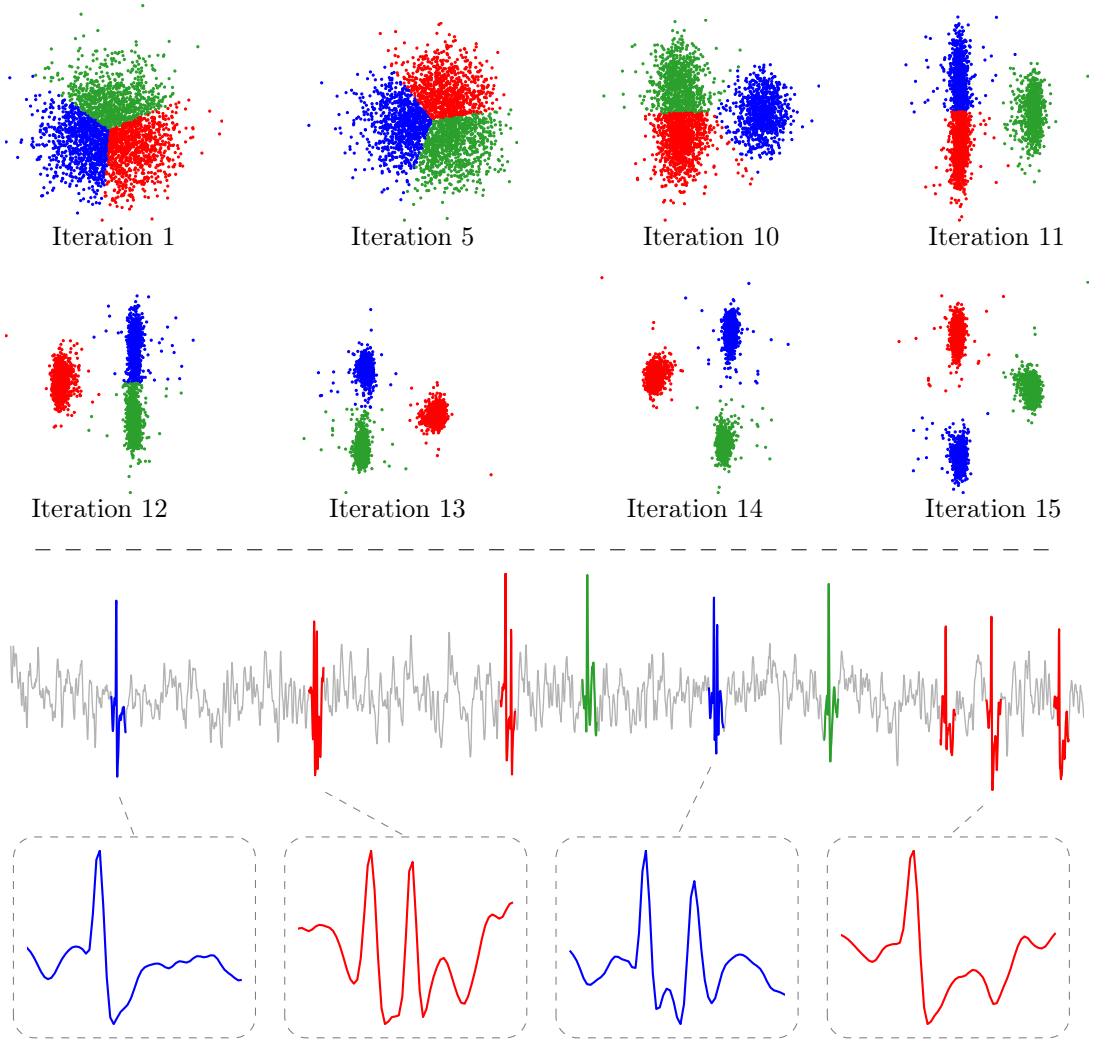

Figure 4.12: Scatter plots of spike features in each iteration of LDA-Km algorithm. Iteration 1 shows the projection on the first two principal components. It can be seen that after 15 iterations, three clusters are discovered. The sorted spike traces are shown in the lower plot, where colors refer to clusters in Iteration 15. As can be seen, the overlapping spikes are assigned to wrong clusters while they should be detected as outliers.

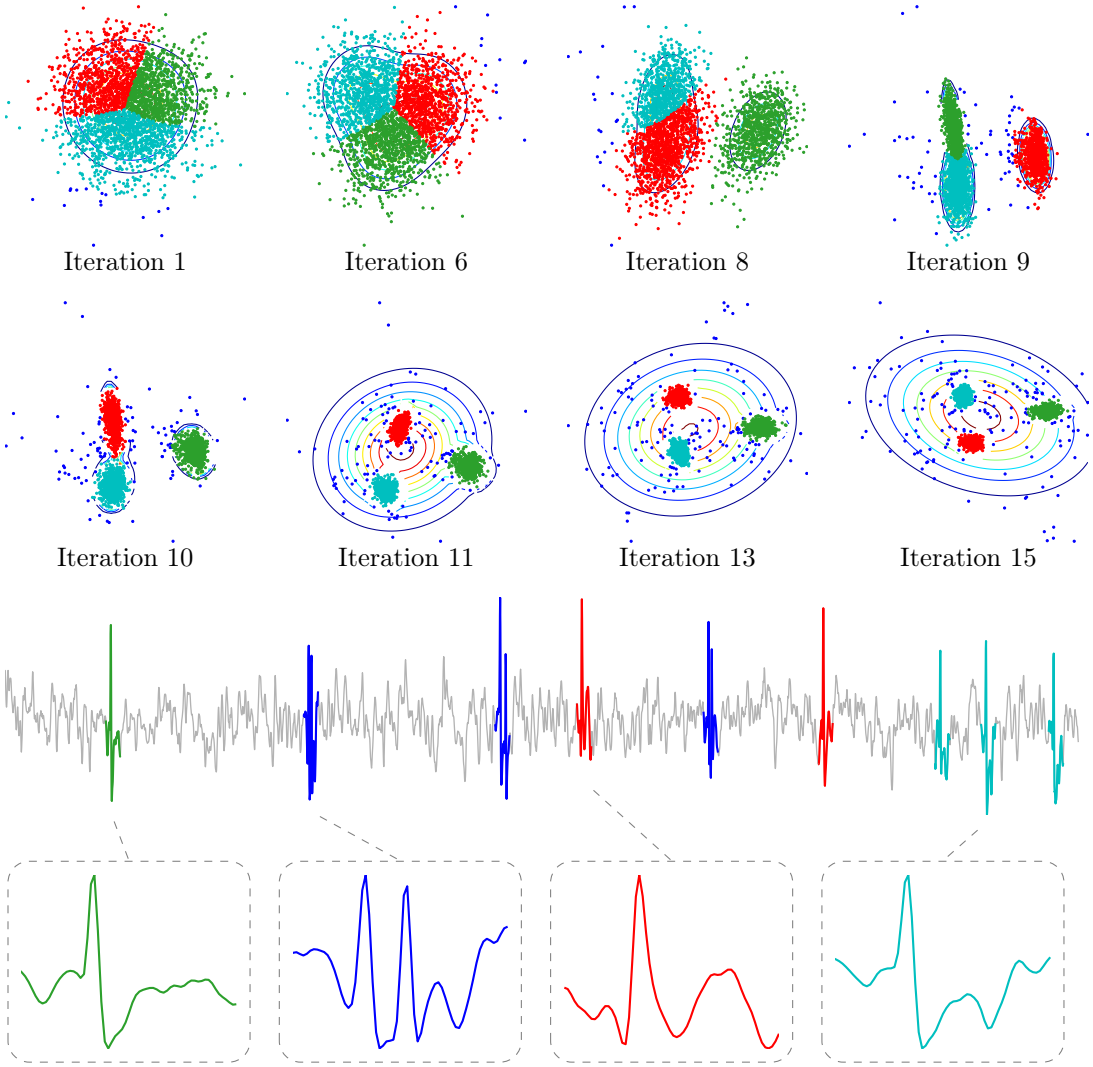

Figure 4.13: Scatter plots of spike features in each iteration of LDA-GMM algorithm on the same data set as in Figure 4.12. Iteration 1 shows the projection on the first two principal components. It can be seen that after 15 iterations, three clusters are discovered and the outliers are identified. The sorted spike traces are shown in the lower plot, where colors refer to clusters in Iteration 15. As can be seen, the overlapping spikes (shown in blue) are correctly assigned as outliers.

has been proposed. This method can properly identify overlapping spikes, thus provide more accurate spike sorting. Furthermore, a method for detecting the number of neurons in the proposed subspace learning scheme was presented.

Based on the introduced subspace learning methods, we proposed two spike sorting algorithms. The first algorithm keeps increasing the number of clusters and learns the discriminative subspace until no extra separate and significant cluster can be identified. The second algorithm utilizes a divisive clustering scheme which starts by dividing the data samples into two clusters in the most discriminative 1-dimensional projection, and keeps dividing the resultant clusters in two until achieving an almost unimodal distribution in the subspace for each final cluster. We evaluated our algorithms against two commonly used spike sorting methods PCA-$k$means and `Wave_clus`, using both synthetic and *in-vivo* data. When tested on synthetic data, the proposed algorithms achieved significantly higher accuracies in all the cases. Furthermore, the results indicate that the algorithms are highly robust to noise. Furthermore, the superior feature extraction capability of the algorithms has been illustrated and compared to PCA, DD-PCA, DWT, and LE, where better cluster separability and increased robustness to noise have been observed. Through utilizing adaptive subspace learning we have combined the three important parts of conventional spike sorting methods including feature extraction, dimensionality reduction and clustering to a single stage, which resulted in a considerable improvement in spike sorting performance.

Chapter 5

Conclusion and Future Works

5.1 Contributions

The detailed conclusion of each work presented in this thesis was discussed at the end of each chapter. Here, we present a summary of the works and highlight their significance.

This study explored two major problems in neural data analysis: power line interference cancellation and spike sorting. Two main contributions were presented for power line interference cancelation. First, a novel algorithm was proposed as a method for narrow-band harmonic estimation, which was tailored to estimate the power line interference and subsequently remove it from the neural data. The proposed algorithm outperforms other competing algorithms, and features a highly robust operation, fast adaptation to interference variations, significant SNR improvement, low computational complexity and memory requirement, and straightforward parameter adjustment. These features render the algorithm suitable for wearable and implantable sensor applications, where reliable and real-time cancellation of the interference is desired. In addition, since the algorithm

highly preserves neural oscillations while cancelling the interference, it may lead researchers to look into the frequency bands that were previously neglected due to the distortion caused by the interference or the inadequate removal of it. To further contribute to this purpose, the source code of the algorithm has been made available online to facilitate its usage by biomedical research community. It is worth mentioning that, although the algorithm is mainly proposed for power line interference cancellation, it can also be used to cancel other types of harmonic interferences which may present in the recording, for example interferences from nearby electrical appliances such as motors and switching power supplies which do not operate at the line frequency, or interferences arising from the recording amplifiers which may produce harmonic distortion. Furthermore, it is applicable to various types of biopotential recordings including ECG and EMG with no modification.

Second, for efficient hardware implementation of the interference cancellation algorithm, a VLSI architecture was proposed, and based on this architecture an ASIC was designed and tested. The proposed VLSI architecture is scalable with respect to the number of the recording channels and/or harmonics, and is optimized for area. This enables on-chip and real-time interference cancellation in high-density neural recording, which was not previously feasible using analog techniques for fully integrated and high channel count microchips. As such, the proposed architecture would be useful in the applications in which the data are required to be processed at the implant side. Based on the proposed architecture, an ASIC in a 65-nm technology was fabricated and successfully tested. The chip consumed 0.11 mm^2 of area and 77 μW of power at the clock rate of 7.5 kHz, and $f_\text{s} = 1.25$ kHz. When tested on different modalities of biomedical recordings, it achieved an average SNR improvement of 30 dB for input SNR from -20 dB to 20 dB and interference frequencies of 45–65 Hz.

Another main contribution of this theses was to propose unsupervised spike sorting algorithms based on discriminative subspace learning to achieve more reliable and accurate identification of neurons and spike waveforms especially in low SNR conditions. Two approaches for unsupervised spike sorting based on simultaneous discriminative subspace learning and clustering have been proposed. A previously proposed method namely LDA-Km has been successfully applied to spike feature learning with superior performance over traditional methods. However, since this method cannot handle overlapping spikes and outliers, a new method based on LDA for subspace selection and GMM for clustering was proposed. This method can properly identify overlapping spikes, thus provides more accurate spike sorting. Furthermore, a method for detecting the number of neurons in the proposed subspace learning scheme was presented. Furthermore, the superior feature extraction capability of the algorithms have been illustrated and compared to several popular feature extraction methods in spikes sorting, where better cluster separability and increased robustness to noise have been observed. Through utilizing adaptive subspace learning we have combined the three important parts of conventional spike sorting methods including feature extraction, dimensionality reduction and clustering to a single stage, which resulted in a considerable improvement in spike sorting performance.

The proposed spike sorting algorithms can learn the most discriminative features and extract the clusters which may appear inseparable in feature space obtained by previously proposed feature extraction methods. In other words, the proposed algorithms can automatically identify neural sources which may not be identifiable from an extracellular recording through using previously proposed approaches in the literature. This opens up the opportunity to analyse the firing patterns from more number of individual neurons (i.e. single units). In neuroscience research, this allows to decode complex brain processes encoded by

the activity of relatively large neural networks. Furthermore, applications using BMIs and neural prostheses could greatly benefit from the increased number of recorded neurons, as this allows the decoding of more complex and precise motor actions.

5.2 Future Works

The work presented in this thesis can be further continued and improved in many ways. Here, we suggest some directions for future works, and potential approaches which could be taken into consideration for further improvements.

5.2.1 Power Line Interference Cancellation

Automatic Parameter Adaptation

The proposed power line interference cancellation algorithm in Chapter 2 requires setting of several parameters to control the adaptation. To make the parameter adjustment straightforward, we presented alternative parameters in Section 2.2.4 and the guidelines how to adjust them. Nevertheless, it would be of interest if the algorithm could automatically tune its parameters. One way to achieve this would be to add a stage for adapting the parameters of the algorithm. While the added stage itself would have some parameters to tune, the sensitivity to those parameters would be low. The extra stage would increase the computational cost and should be added only if the algorithm is to be used by non-expert users or for general purpose applications.

Further Reducing the Computational Complexity

The multichannel interference cancellation algorithm presented in Chapter 3 scales quite efficiently with the number of channels and/or harmonics. Nevertheless,

there could be some margin for further decreasing the computational cost. One possible approach would be to combine the sinusoidal signal generation and phase and amplitude adaptation stages. This would lead to a structure similar to IIR comb filter. In this case further analysis on stability should be carried out to guarantee the reliable operation of the algorithm.

Low Power VLSI Implementation

The microchip presented in Chapter 3 was synthesised using standard library for 65-nm technology and no circuit level optimization was carried out. That said, through using low power design techniques the power consumption would be largely reduced. In addition, it would be interesting to have a programmable VLSI architecture in which the number of the channels and/or harmonics could be programmed and changed based on the application and the bandwidth of interest.

5.2.2 Spike Sorting

Online Learning and Real-time Spike Sorting

The spike sorting algorithms presented in Chapter 4 are based on batch processing of spike waveforms, meaning that the spikes should first be extracted, and then the algorithms require all (or at least a large number of) the spikes for subspace learning. Thus, the algorithms are only suitable for offline setting. It is, however, of practical interest to develop the online versions of the algorithms for real-time spike sorting. In this case, when a spike is detected, the algorithm should be able to adapt the subspace and classify the spike based on the detected spike and not the whole data. Although some methods exist for online clustering and subspace selection, the combination of them for subspace learning would not necessarily

lead to an appropriate solution.

Resolving Overlapping Spikes

The proposed spike sorting algorithms can adequately detect the overlapping spikes as outliers; however, they cannot identify the neurons contributed to the overlap. Several methods have been proposed in the literature which try to decompose the overlapping spike waveforms into the waveforms originated from the source neurons. If the signals are recorded from multiple electrode, another approach would be to use mutual dependence between channels to perform signal separation in order to decompose overlapping spikes or identify the source neurons. These approaches could be added as an extra stage in the proposed spike sorting scheme without modifying the main algorithms.

Multichannel Processing

Our proposed spike sorting scheme assumed the data had been recorded with single-electrode probes. In practice, however, many researchers use multi-electrode probes such as tetrodes which consist of four closely spaced electrodes. In this case, mutual information between electrodes could be used to enhance the spike sorting performance and facilitate the resolution of overlapping spikes. This could be incorporated into the algorithm through using blind source separation methods such as independent component analysis to a preliminary signal decomposition before spike sorting. Another interesting approach would be to modify the objective function and formulate an optimization problem for subspace learning which would accommodate signal dependencies between the channels.

Hardware Efficient Algorithm Design for Real-time Spike Sorting

In multichannel wireless neural recording systems, spike sorting at the implant side considerably decreases the data rate since instead of transmitting the full spike waveforms, the spike arrival times and labels are transmitted, which can lead to over 98% reduction in data rate. The reduced data rate would in turn result in a significant reduction of power consumption thus extending battery life over 270 times [81]. Thus is it of a great interest to enhance the algorithm to suit hardware design constraints such as limited memory and computational resources. This is indeed conditioned on the successful design of the online version of the algorithms as previously explained.

Appendix A

Open Source Power Line Interference Canceller Software

The power line interference cancellation algorithm introduced in Chapter 2 is implemented in MATLAB and has been made available to public at `https://github.com/mrezak/removePLI` [52]. The file `removePLI.m` implements the exact algorithm in Chapter 2 for single channel data. `removePLI_multichan.m` implements the multichannel version of the algorithm which is introduced in Chapter 3.

A graphical user interface (GUI) is also provided for easy use of the algorithm. The user may select the nominal AC frequency if known. In this case the initial bandpass filter used for frequency estimation is narrowed down to the selected frequency (i.e. 58–62 Hz or 48–52 Hz pass band) for more robust estimation. Otherwise the algorithm uses the default pass band of 40–70 Hz. Figure A.1 shows the snapshot of the software GUI.

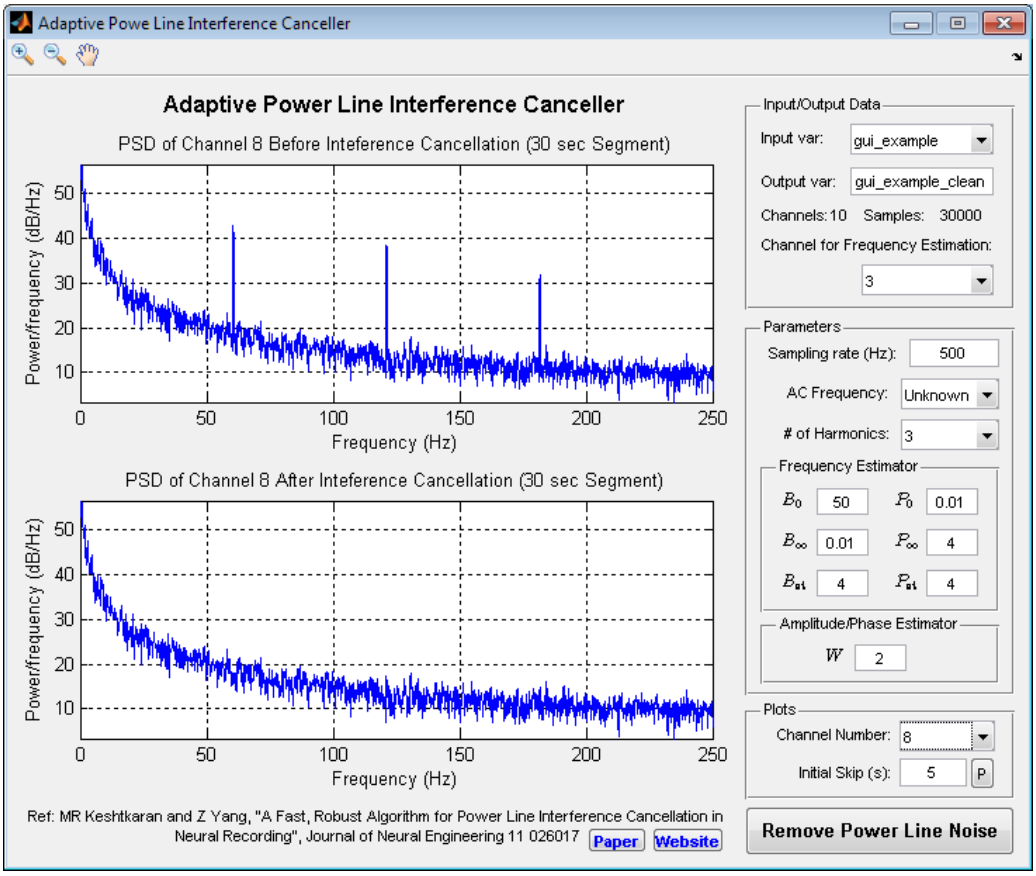

Figure A.1: Snapshot of the GUI of multichannel power line interference canceller software in MATLAB.

Bibliography

[1] G. Buzsaki, *Rhythms of the Brain.* Oxford: Oxford University Press, 2006.

[2] G. Buzsaki, "Large-scale recording of neuronal ensembles," *Nat Neurosci*, vol. 7, no. 5.

[3] M. F. Chimene and R. Pallas-Areny, "A comprehensive model for power line interference in biopotential measurements," *IEEE Transactions on Instrumentation and Measurement*, vol. 49, no. 3, pp. 535–540, 2000.

[4] P. Mitra and H. Bokil, *Observed Brain Dynamics*, 1st ed. Oxford: Oxford University Press, 2008.

[5] G. Buzsaki, Z. Horvath, R. Urioste, J. Hetke, and K. Wise, "High-frequency network oscillation in the hippocampus," *Science*, vol. 256, no. 5059, pp. 1025–1027, 1992.

[6] M. S. Jones, K. D. MacDonald, B. Choi, F. E. Dudek, and D. S. Barth, "Intracellular correlates of fast (>200 Hz) electrical oscillations in rat somatosensory cortex," *Journal of Neurophysiology*, vol. 84, no. 3, pp. 1505–1518, 2000.

[7] R. J. Staba, C. L. Wilson, A. Bragin, I. Fried, and J. Engel, "Quantitative analysis of high-frequency oscillations (80–500 Hz) recorded in human epileptic hippocampus and entorhinal cortex," *Journal of Neurophysiology*, vol. 88, no. 4, pp. 1743–1752, 2002.

[8] Z. C. Chao, Y. Nagasaka, and N. Fujii, "Long-term asynchronous decoding of arm motion using electrocorticographic signals in monkeys," *Front. Neuroeng.*, vol. 3, p. 3, 2010.

[9] E. C. Leuthardt, G. Schalk, J. R. Wolpaw, J. G. Ojemann, and D. W. Moran, "A brain–computer interface using electrocorticographic signals in humans," *Journal of Neural Engineering*, vol. 1, no. 2, pp. 63–71, 2004.

[10] K. Miller, P. Shenoy, M. den Nijs, L. Sorensen, R. Rao, and J. Ojemann, "Beyond the gamma band: The role of high-frequency features in movement classification," *IEEE Transactions on Biomedical Engineering*, vol. 55, no. 5, pp. 1634–1637, 2008.

[11] D. Moran, "Evolution of brain–computer interface: action potentials, local field potentials and electrocorticograms," *Current Opinion in Neurobiology*, vol. 20, no. 6, pp. 741–745, 2010.

[12] K. Shimoda, Y. Nagasaka, Z. C. Chao, and N. Fujii, "Decoding continuous three-dimensional hand trajectories from epidural electrocorticographic signals in japanese macaques," *Journal of Neural Engineering*, vol. 9, no. 3, p. 036015, 2012.

[13] N. Liang and L. Bougrain, "Decoding finger flexion from band-specific ecog signals in humans," *Frontiers in Neuroscience*, vol. 6, no. 91, p. 91, 2012.

[14] E. J. Hwang and R. A. Andersen, "The utility of multichannel local field potentials for brain–machine interfaces," *Journal of Neural Engineering*, vol. 10, no. 4, p. 046005, 2013.

[15] K. J. Miller, S. Zanos, E. E. Fetz, M. d. Nijs, and J. G. Ojemann, "Decoupling the cortical power spectrum reveals real-time representation of individual

finger movements in humans," *Journal of Neuroscience*, vol. 29, no. 10, pp. 3132–3137, 2009.

[16] M. S. Lewicki, "A review of methods for spike sorting: the detection and classification of neural action potentials," *Network (Bristol, England)*, vol. 9, no. 4, pp. R53–78, 1998.

[17] S. Gibson, J. Judy, and D. Markovic, "Spike sorting: The first step in decoding the brain: The first step in decoding the brain," *IEEE Signal Processing Magazine*, vol. 29, no. 1, pp. 124–143, 2012.

[18] S. M. Martens, M. Mischi, S. G. Oei, and J. W. Bergmans, "An improved adaptive power line interference canceller for electrocardiography," *IEEE Transactions on Biomedical Engineering*, vol. 53, no. 11, pp. 2220–2231, 2006.

[19] M. Ferdjallah and R. E. Barr, "Adaptive digital notch filter design on the unit circle for the removal of powerline noise from biomedical signals," *IEEE Transactions on Biomedical Engineering*, vol. 41, no. 6, pp. 529–536, 1994.

[20] A. K. Ziarani and A. Konrad, "A nonlinear adaptive method of elimination of power line interference in ECG signals," *IEEE Transactions on Biomedical Engineering*, vol. 49, no. 6, pp. 540–547, 2002.

[21] Z. Yang, Q. Zhao, and W. Liu, "Improving spike separation using waveform derivatives," *Journal of Neural Engineering*, vol. 6, no. 4, p. 046006, 2009.

[22] E. Chah, V. Hok, A. Della-Chiesa, J. J. H. Miller, S. M. O'Mara, and R. B. Reilly, "Automated spike sorting algorithm based on laplacian eigenmaps and k-means clustering," *Journal of Neural Engineering*, vol. 8, no. 1, p. 016006, 2011.

[23] V. Shalchyan, W. Jensen, and D. Farina, "Spike detection and clustering with unsupervised wavelet optimization in extracellular neural recordings," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 9, pp. 2576–2585, 2012.

[24] C. Yang, Y. Yuan, and J. Si, "Robust spike classification based on frequency domain neural waveform features," *Journal of Neural Engineering*, vol. 10, no. 6, p. 066015, 2013.

[25] M. R. Keshtkaran and Z. Yang, "A fast, robust algorithm for power line interference cancellation in neural recording," *Journal of Neural Engineering*, vol. 11, no. 2, p. 026017, 2014.

[26] M. R. Keshtkaran and Z. Yang, "Power line interference cancellation in in-vivo neural recording," in *International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2012, pp. 5214–5217.

[27] M. Keshtkaran and Z. Yang, "A robust adaptive power line interference canceller VLSI architecture and ASIC for multichannel biopotential recording applications," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol.61, no.10, pp.788,792, Oct. 2014

[28] M. R. Keshtkaran and Z. Yang, "Unsupervised spike sorting based on discriminative subspace learning," in *2014 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2014, pp. 3784–3788.

[29] A. Metting van Rijn, A. Peper, and C. Grimbergen, "High-quality recording of bioelectric events," *Medical and Biological Engineering and Computing*, vol. 28, no. 5, pp. 389–397, 1990.

[30] M. Teplan, "Fundamentals of EEG measurement," *Measurement Science Review*, vol. 2, pp. 1–11, 2002.

[31] C. K. Thorp and P. N. Steinmetz, "Interference and noise in human intracranial microwire recordings," *IEEE Transactions on Biomedical Engineering*, vol. 56, no. 1, pp. 30–36, 2009.

[32] R. C. Dugan, M. F. McGranaghan, S. Santoso, and H. W. Beaty, *Electrical Power Systems Quality.* New York: McGraw-Hill, 2012.

[33] A. Baggini, *Handbook of Power Quality.* Chichester: John Wiley & Sons, 2008.

[34] Z. Wang and A. W. Roe, "Trial-to-trial noise cancellation of cortical field potentials in awake macaques by autoregression model with exogenous input (ARX)," *Journal of Neuroscience Methods*, vol. 194, no. 2, pp. 266–273, 2011.

[35] T. Degen and H. Jackel, "Enhancing interference rejection of preamplified electrodes by automated gain adaption," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 11, pp. 2031–2039, 2004.

[36] E. Spinelli and M. Mayosky, "Two-electrode biopotential measurements: power line interference analysis," *IEEE Transactions on Biomedical Engineering*, vol. 52, no. 8, pp. 1436–1442, 2005.

[37] H. Alzaher, N. Tasadduq, and Y. Mahnashi, "A highly linear fully integrated powerline filter for biopotential acquisition systems," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 7, no. 5, pp. 703–712, 2013.

[38] S. J. Luck, *An Introduction to the Event-Related Potential Technique (Cognitive Neuroscience)*, 1st ed. Cambridge, MA: MIT Press, 2005.

[39] C. Levkov, G. Mihov, R. Ivanov, I. Daskalov, I. Christov, and I. Dotsinsky, "Removal of power-line interference from the ECG: a review of the subtraction procedure," *Biomedical Engineering Online*, vol. 4, no. 1, p. 50, 2005.

[40] B. Widrow, J. R. Glover, J. M. McCool, J. Kaunitz, C. S. Williams, R. H. Hearn, J. R. Zeidler, J. Eugene Dong, and R. C. Goodlin, "Adaptive noise cancelling: Principles and applications," *Proceedings of the IEEE*, vol. 63, no. 12, pp. 1692–1716, 1975.

[41] P. Hamilton, "A comparison of adaptive and nonadaptive filters for reduction of power line interference in the ECG," *IEEE Transactions on Biomedical Engineering*, vol. 43, no. 1, pp. 105–109, 1996.

[42] D. A. Heldman, W. Wang, S. S. Chan, and D. W. Moran, "Local field potential spectral tuning in motor cortex during reaching," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 14, no. 2, pp. 180–183, 2006.

[43] K. J. Miller, E. C. Leuthardt, G. Schalk, R. P. N. Rao, N. R. Anderson, D. W. Moran, J. W. Miller, and J. G. Ojemann, "Spectral changes in cortical surface potentials during motor movement," *Journal of Neuroscience*, vol. 27, no. 9, pp. 2424–2432, 2007.

[44] N. I. Cho, C.-H. Choi, and S. U. Lee, "Adaptive line enhancement by using an IIR lattice notch filter," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, no. 4, pp. 585–589, 1989.

[45] A. Nehorai, "A minimal parameter adaptive notch filter with constrained poles and zeros," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 33, no. 4, pp. 983–996, 1985.

[46] S. Kay, "A fast and accurate single frequency estimator," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, no. 12, pp. 1987–1990, 1989.

[47] P. A. Regalia, "Stable and efficient lattice algorithms for adaptive IIR filtering," *IEEE Transactions on Signal Processing*, vol. 40, no. 2, pp. 375–388, 1992.

[48] N. Cho and S. Lee, "On the adaptive lattice notch filter for the detection of sinusoids," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 40, no. 7, pp. 405–416, 1993.

[49] J. Klein, "Fast algorithms for single frequency estimation," *IEEE Transactions on Signal Processing*, vol. 54, no. 5, pp. 1762–1770, 2006.

[50] C. Turner, "Recursive discrete-time sinusoidal oscillators," *Signal Processing Magazine, IEEE*, vol. 20, no. 3, pp. 103–111, 2003.

[51] B. Farhang-Boroujeny, *Adaptive Filters: Theory and Applications*, 1st ed. Chichester: John Wiley & Sons, 1999.

[52] M. R. Keshtkaran, "Adaptive Power Line Interference Canceller Source Code," Github, 2013. [Online]. Available: https://github.com/mrezak/removePLI

[53] J. C. Huhta and J. Webster, "60-hz interference in electrocardiography," *IEEE Transactions on Biomedical Engineering*, vol. BME-20, no. 2, pp. 91–101, 1973.

[54] R. Pallas-Areny, "Interference-rejection characteristics of biopotential amplifiers: a comparative analysis," *IEEE Transactions on Biomedical Engineering*, vol. 35, no. 11, pp. 953–959, 1988.

[55] B. B. Winter and J. Webster, "Driven-right-leg circuit design," *IEEE Transactions on Biomedical Engineering*, vol. BME-30, no. 1, pp. 62–66, 1983.

[56] T. Degen, S. Torrent, and H. Jackel, "Low-noise two-wired buffer electrodes for bioelectric amplifiers," *IEEE Transactions on Biomedical Engineering*, vol. 54, no. 7, pp. 1328–1332, 2007.

[57] A. Metting van Rijn, A. Peper, and C. Grimbergen, "The isolation mode rejection ratio in bioelectric amplifiers," *IEEE Transactions on Biomedical Engineering*, vol. 38, no. 11, pp. 1154–1157, 1991.

[58] A. C. Metting van Rijn, A. Peper, and C. A. Grimbergen, "High-quality recording of bioelectric events: Part 2 low-noise, low-power multichannel amplifier design," *Medical & Biological Engineering & Computing*, vol. 29, no. 4, pp. 433–440, 1991.

[59] J. L. Bohorquez, M. Yip, A. P. Chandrakasan, and J. L. Dawson, "A biomedical sensor interface with a sinc filter and interference cancellation," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 4, pp. 746–756, 2011.

[60] I.-D. Hwang and J. Webster, "Direct interference canceling for two-electrode biopotential amplifier," *IEEE Transactions on Biomedical Engineering*, vol. 55, no. 11, pp. 2620–2627, 2008.

[61] M. Bahoura and H. Ezzaidi, "FPGA-Implementation of parallel and sequential architectures for adaptive noise cancelation," *Circuits Syst. Signal Process.*, vol. 30, no. 6, pp. 1521–1548, 2011.

[62] N. Jindapetch, S. Chewae, and P. Phukpattaranont, "FPGA implementations of an ADALINE adaptive filter for power-line noise cancellation in surface electromyography signals," *Measurement*, vol. 45, no. 3, pp. 405–414, 2012.

[63] G. Seibel, F. Itturriet, E. Costa, and S. Almeida, "Fixed-point adaptive filter architecture for the harmonics power line interference cancelling," in *IEEE Fourth Latin American Symposium on Circuits and Systems (LASCAS)*, 2013, pp. 1–4.

[64] D. A. Adamos, E. K. Kosmidis, and G. Theophilidis, "Performance evaluation of PCA-based spike sorting algorithms," *Computer Methods and Programs in Biomedicine*, vol. 91, no. 3, pp. 232–244, 2008.

[65] C. Ding and T. Li, "Adaptive dimension reduction using discriminant analysis and k-means clustering," in *Proceedings of the 24th International Conference on Machine Learning*, ser. ICML '07. New York, NY, USA: ACM, 2007, p. 521–528.

[66] K. H. Kim and S. J. Kim, "Neural spike sorting under nearly 0-dB signal-to-noise ratio using nonlinear energy operator and artificial neural-network classifier," *IEEE Transactions on Biomedical Engineering*, vol. 47, no. 10, pp. 1406–1411, 2000.

[67] R. Q. Quiroga, Z. Nadasdy, and Y. Ben-Shaul, "Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering," *Neural Computation*, vol. 16, no. 8, pp. 1661–1687, 2004.

[68] D. Novak, J. Wild, T. Sieger, and R. Jech, "Identifying number of neurons in extracellular recording," in *4th International IEEE/EMBS Conference on Neural Engineering, 2009. NER '09*, 2009, pp. 742–745.

[69] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[70] J. A. Hartigan and P. M. Hartigan, "The dip test of unimodality," *The Annals of Statistics*, vol. 13, no. 1.

[71] H. W. Lilliefors, "On the kolmogorov-smirnov test for normality with mean and variance unknown," *Journal of the American Statistical Association*, vol. 62, no. 318.

[72] T. W. Anderson and D. A. Darling, "A test of goodness of fit," *Journal of the American Statistical Association*, vol. 49, no. 268, pp. 765–769, 1954.

[73] [Online]. Available: http://crcns.org/data-sets/hc/hc-1

[74] S. Shoham, M. R. Fellows, and R. A. Normann, "Robust, automatic spike sorting using mixtures of multivariate t-distributions," *Journal of Neuroscience Methods*, vol. 127, no. 2, pp. 111–122, 2003.

[75] C. Vargas-Irwin and J. P. Donoghue, "Automated spike sorting using density grid contour clustering and subtractive waveform decomposition," *Journal of Neuroscience Methods*, vol. 164, no. 1, pp. 1–18, 2007.

[76] E. Wood, M. Fellows, J. Donoghue, and M. Black, "Automatic spike sorting for neural decoding," in *26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2004. IEMBS '04*, vol. 2, 2004, pp. 4009–4012.

[77] P.-M. Zhang, J.-Y. Wu, Y. Zhou, P.-J. Liang, and J.-Q. Yuan, "Spike sorting based on automatic template reconstruction with a partial solution to the overlapping problem," *Journal of Neuroscience Methods*, vol. 135, no. 1–2, pp. 55–65, 2004.

[78] P. M. Horton, A. U. Nicol, K. M. Kendrick, and J. F. Feng, "Spike sorting based upon machine learning algorithms (SOMA)," *Journal of Neuroscience Methods*, vol. 160, no. 1, pp. 52–68, 2007.

[79] J. H. Choi, H. K. Jung, and T. Kim, "A new action potential detector using the MTEO and its effects on spike sorting systems at low signal-to-noise ratios," *IEEE Transactions on Biomedical Engineering*, vol. 53, no. 4, pp. 738–746, 2006.

[80] T. Takekawa, Y. Isomura, and T. Fukai, "Accurate spike sorting for multi-unit recordings," *European Journal of Neuroscience*, vol. 31, no. 2, pp. 263–272, 2010.

[81] S. P. Gibson, "Neural spike sorting in hardware: From theory to practice," Doctoral Dissertation, University of California, Los Angeles, United States – California, 2012.

List of Publications

1. M. R. Keshtkaran and Z. Yang, "A robust adaptive power line interference canceller VLSI architecture and ASIC for multichannel biopotential recording applications," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol.61, no.10, pp.788,792, Oct. 2014.

2. M. R. Keshtkaran and Z. Yang, A fast, robust algorithm for power line interference cancellation in neural recording," *Journal of Neural Engineering*, vol. 11, no. 2, p. 026017, Apr. 2014.

3. M. R. Keshtkaran and Z. Yang, "Unsupervised spike sorting based on discriminative subspace learning," in *International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2014, pp. 3784–3788.

4. M. R. Keshtkaran and Z. Yang, "Power line interference cancellation in in-vivo neural recording," in *International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2012, pp. 5214–5217.

5. Z. Yang, W. Liu, M. R. Keshtkaran, Y. Zhou, J. Xu, V. Pikov, C. Guan, and Y. Lian, "A new EC–PC threshold estimation method for in vivo neural spike detection," *Journal of Neural Engineering,* vol. 9, no. 4, p. 046017, Aug. 2012.