Implementation of Neural Filters using Hopf oscillators

A Project Report

submitted by

ANAS JAFRY K K (BE13B005)

in partial fulfilment of the requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY & MASTER OF TECHNOLOGY
(DUAL DEGREE)

DEPARTMENT OF BIOTECHNOLOGY
INDIAN INSTITUTE OF TECHNOLOGY MADRAS

May 2018

THESIS CERTIFICATE

This is to certify that the report titled **"Implementation of Neural Filters using Hopf oscillators"**, submitted by **Anas Jafry K K**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelor of Technology and Master of Technology (Dual Degree)**, is a bona fide record of the research work done by him under my supervision. The contents of this report, in full or in parts, have not been submitted to any other institute or university for the award of any degree or diploma.

Dr. V Srinivasa Chakravarthy
Research Guide
Professor
Dept. of Biotechnology
IIT-Madras, 600 036

Place: Chennai

Date: 14^{th} May 2018

ACKNOWLEDGEMENTS

I would like to express my sincere and heartfelt thanks to my project guide, Dr. V Srinivasa Chakravarthy for his valuable and inspiring guidance throughout the course of this work. I am thoroughly grateful to him for the amount of time he has spent in reviewing my data and reports and for providing me with all the valuable insights and necessary facilities required. His moral support and continuous encouragement enabled me to complete the work successfully. Throughout the course of my project, I have been fortunate to receive valuable assistance and guidance from many senior students, research scholars and my peers at IIT Madras. Friends and family have been of great moral and emotional support throughout my stay at IIT Madras. Finally, I would like to place my sense of gratitude to all those who directly or indirectly contributed to my success and for the strong foundation that has been laid for the future.

ABSTRACT

KEYWORDS: Adaptive Hopf Oscillators ; Perceptron Learning Algorithm; Autoencoders; Feedforward networks; Spiking neuron model; locomotor rhythms.

All humans display five different types of brain waves across the cortex which can be observed with an Electroencephalography (EEG). Oscillatory phenomena are ubiquitous in the brain. In spite of the fact that there are oscillator-based models of brain, they don't appear to appreciate the all inclusive computational properties of rate-coded and spiking neuron arrange models. Use of oscillator-based models is often limited to special phenomena like locomotor rhythms and oscillatory attractor-based memories. Autoencoders are a special type of feedforward networks that have been utilized for development of large-scale deep networks. Despite the fact that autoencoders in view of rate-coded and spiking neuron systems have been proposed, there are no autoencoders in view of oscillators. We propose a channel for the EEG signals in light of the Perceptron Learning Algorithm.

In a nutshell, this report comprises of two parts. An adaptive Hopf oscillator which will be used for frequency tracking and a neural filter based on perceptron learning algorithm for EEG signals. Applying this model, we show effective reconstruction of the input signals and proper cutoff of the signals based on the bandwidth of the brain waves.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

EEG	Electroencephalography
MLP	Multi Layer Perceptron
FFT	Fast Fourier Transform
LPF	Low Pass Filter
BPF	Band Pass Filter
HPF	High Pass Filter
BCI	Brain Computer Interface
DSP	Digital Signal Processing

NOTATION

$P_{teach}(t)$	Input Signal
$\hat{P}(t)$	Hopf output
$Q_{learned}(t)$	Filter output

CHAPTER 1

INTRODUCTION

1.1 Motivation

People with severe neuromuscular disorders, such as late-stage amyotrophic sclerosis (ALS) and those paralyzed from higher level spinal cord injury are unable to actuate any of their muscles. Communication with the outside world is therefore problematic for the suffering people. Cognitive and sensory body functions, however, are often only minimally affected. Therefore, an electroencephalogram (EEG)-based communication which does not require any neuromuscular control is considered to be particularly helpful to enhance the disableds quality of life by increasing their independence.

This work has future applications on EEG-based brain-computer interfaces (BCIs) for decoding motor imagery left- and right-hand movements to control a robotic arm or prosthesis, contributing to the advancement in rehabilitation robotics and various other BCI applications.

Feedforward Networks

Feedfoward Neural Networks are artificial neural networks where the connections between units do not form a cycle. Feedfoward neural networks were the first type of simulated neural networks developed and are more straightforward than their counterpart, recurrent neural systems. They are called feedforward on the grounds that data just goes forward in the system , first through the input nodes, then through the hidden nodes , and finally through the output nodes.

Feedfoward neural networks are basically utilized for supervised learning in situations where the data to be learned is neither sequential nor time-dependent. That is, feedforward neural networks compute a function f on fixed size input x such that $f(x) \approx y$ for training pairs (x, y). On the other hand, recurrent neural networks learn sequential data, computing on variable length input $X_k = \{x_1, ..., x_k\}$ such that $g(X_k) \approx y_k$ for training pairs (X_n, Y_n) for all $1 \leq k \leq n$.

Multi Layer Perceptron (MLP)

A MPL takes a vector x as an input, calculates hidden activations h and outputs a vector as follows:

$$h = \alpha_1(W^{xh}x + b^h)$$

$$\hat{y} = \alpha_2(W^{h\hat{y}}h + b^{\hat{y}})$$

where W_{ij} describes the weight matrix connecting two neighboring layers. Specifically, W_{xh} are the weights connecting input layer to hidden layer, $W^{h\hat{y}}$ are the weights connecting hidden layer to output layer, b^l is a bias vector for layer l, and $\alpha_i(x)$ where $i \in \{1, 2\}$ are activation functions. The activation functions are applied element-wise when they are computed for all neurons in a layer simultaneously using matrix-vector notation as above. When multiple hidden layers are referred to, h^l is the subsequent layer to h^{l1} , where $l \in \{1, ..., L\}$ and L denotes the total number of hidden layers.

Autoencoders

An autoencoder neural network is an unsupervised learning algorithm that applies backpropagation, setting the target values to be equal to the inputs. i.e., it uses $y(i) = x(i)$.

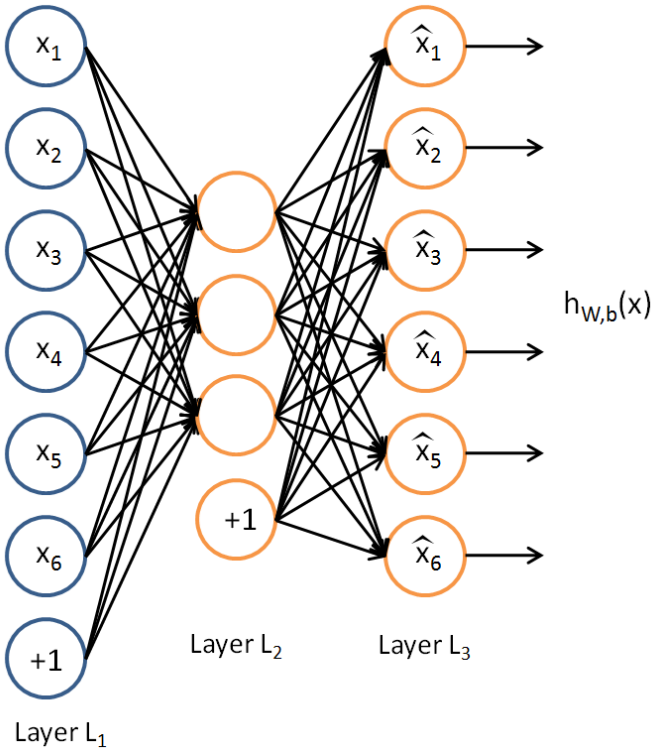

Figure 1.1: Autoencoder

The autoencoder tries to learn a function $h_W, b(x) \approx x$. In other words, it is trying to learn an approximation to the identity function, so as to output $\hat{x}$ that is similar to x. The identity function seems a particularly trivial function to be trying to learn; yet by setting requirements on the system, for example, by altering the quantity of hidden layers, we can find fascinating structure about the data.

1.2 Adaptive Hopf Oscillators

Dynamic Learning for Oscillators

Hopf oscillators are picked on the grounds that it possesses a harmonic limit cycle that will be utilized to disintegrate the periodic function we want to learn into a sum of sines and cosines.

$$\dot{x} = \gamma(\mu - r_2)x - \omega y + \epsilon F(t)$$

$$\dot{y} = \gamma(\mu - r_2)y + \omega x$$

$$\dot{\omega} = -\epsilon F(t)y/r$$

Where $F(t)$ is the input signal, frequency of the oscillator described as ω, ϵ is a coupling constant and $r = \sqrt{x^2 + y^2}$. In this system, the oscillator will learn the frequency of the periodic input $F(t)$, it means that ω will converge to the frequency of F. If the input F(t) has several frequency components, the oscillator will generally adapt its frequency to the closest frequency component of the input.

Feedback Structure and amplitude adaptation

Here we look at adaptive Hopf oscillators that learn the frequency components of a periodic input signal and their amplitudes as well. A simple feedback loop

that controls the learning is introduced to the system. Every oscillator of the system will learn one frequency part of the training signal. It will likewise adapt to the related amplitude. At the point when an oscillator has taken in the right frequency and amplitude of a frequency component of the input, this frequency segment will then vanish from the teaching signal. In order to achieve this, we use a simple feedback loop described in Figure 1.2. In this figure, we see that the learned signal is the sum of all the outputs of the oscillators, weighted by the corresponding amplitude, basically having a Fourier decomposition of the input signal. The feedback loop subtracts the already learned signal from the teaching signal. So only the frequency components that were not already learned are still sent to the oscillators. The network of N oscillators, with frequency and amplitude adaptation, is described by the following set of equations

$$\dot{x}_i = \gamma(\mu - r_i^2)x_i - \omega_i y_i + \epsilon F(t)$$

$$\dot{y}_i = \gamma(\mu - r_i^2)y_i - \omega_i x_i$$

$$\dot{\omega}_i = -\epsilon F(t) y_i / r_i$$

$$\dot{\alpha}_i = \eta x_i F(t)$$

$$F(t) = P_{teach}(t) - \Sigma \alpha_i x_i$$

where x_i , y_i and ω_i describe the i^{th} adaptive Hopf oscillator. $r_i = \sqrt{x_i^2 + y_i^2}$, η and ϵ are positive coupling constants controlling the learning rate. P_{teach} represents the input signal to learn. The amplitudes of the frequency components are learned with the $_i$ variable. The learning rule is a simple correlation based learning, here $_i$ increases when the functions $x_i(t)$ and $F(t)$ are correlated, which happens when they have the same frequencies. $_i$ will stabilize as the associated frequency component will disappear from $F(t)$ when the amplitude is correct.

With the network we presented we are able to learn the frequency components of any periodic input. As long as the frequency spectrum is finite, as each oscillator codes for one frequency component, it is sufficient to have as many oscillators as frequency components in the system. The amplitude of each frequency component is also learned with the $_i$ variable.the learned signal, $Q_{learned} = \sum_{i=0}^{N} \alpha_i x_i$, is coded in the network of oscillators.

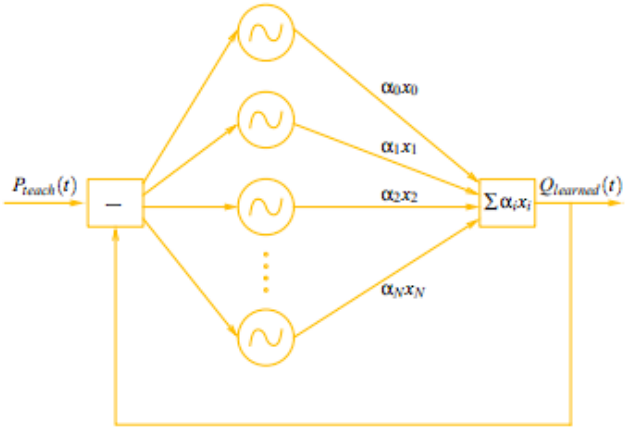

Figure 1.2: Structure of the network of adaptive Hopf oscillators with amplitude adaptation
Ludovic Righetti and Ijspeert (2005)

Phase Relation adaptation

Phase relations between oscillators can be achieved by coupling them. Two oscillators, when coupled, are entrained if their frequencies have the relationship of the form $\omega_1 \simeq \frac{p}{q}2, p, q \in N$.

Each oscillator has 5 state variables, 2 for the oscillatory motion that assure structural stability $(x_i and y_i)$, one for learning the frequency (ω_i), one for the amplitude (α_i) and one for the phase relations (ϕ_i). The network is shown in

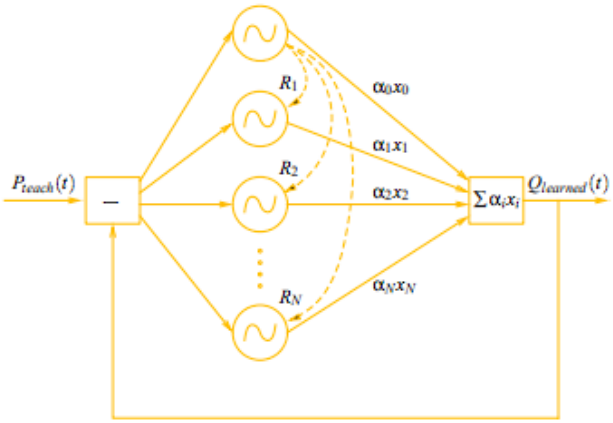

Figure 1.3: Structure of the network of adaptive Hopf oscillators with phase adaptation

Figure 1.3 and summarizes in the following equations

$$\dot{x}_i = \gamma(\mu - r_i^2)x_i - \omega_i y_i + \epsilon F(t) + \tau \sin R_i - \phi_i$$

$$\dot{y}_i = \gamma(\mu - r_i^2)y_i - \omega_i x_i$$

$$\dot{\omega}_i = -\epsilon F(t) y_i / r_i$$

$$\dot{\phi}_0 = 0$$

$$\dot{\phi}_i = \sin R_i - sgn(x_i) \arccos -\frac{y_i}{r_i} - \phi_i, \forall i \neq 0$$

with

$$R_i = \frac{\omega_i}{\omega_0} sgn(x_0) \arccos -\frac{y_0}{\sqrt{x_0^2 + y_0^2}}$$

and

$$\dot{\alpha}_i = \eta x_i F(t)$$

$$F(t) = P_{teach}(t) - \Sigma \alpha_i x_i$$

We have now presented a network of coupled adaptive Hopf oscillators that can learn a periodic input. The adaptation is done in a single stage, the learning takes place in the coupled network of oscillators. This system can be now used further for learning the frequencies and implementing the filter.

1.3 Organisation of report

Report is organised as follows -

Chapter 2 describes the architectural setup which is common to all the four methods. It starts by discussing about the input data (EEG signal and Chirp Signal). This chapter briefs how about the different methodologies adopted to produce the system. Following that is a discussion on the different trials.

Chapter 3 discusses the different methods implemented. The details of the algorithm implementation are provided. Results of experimentation after the implementation of algorithm is analysed, followed by a summary of the work.

Chapter 4 summarises the projects by a discussion of results and prospects of each algorithm.

CHAPTER 2

EXPERIMENTAL SETUP

This chapter explains the network architecture used for all the three methodologies that are discussed in the following chapters. Firstly, this chapter describes the input used for all projects. It continues to explain the significance of the filter system followed by a discussion on the techniques.

2.1 Brain Wave Frequencies

All humans display five different types of electrical patterns or "brain waves" across the cortex. The brain waves can be observed with an EEG (Electroencephalograph). Each brain wave has a purpose and helps serve us in optimal mental functioning.

Our mind's capacity to wind up adaptable and change through different brain wave frequencies assumes a substantial part in how fruitful we are at overseeing pressure, concentrating on assignments, and getting a decent night's rest. On the off chance that one of the five kinds of mind waves is either overproduced or under produced in our brain, it can cause issues. Hence, it is imperative to comprehend that there is no single mind wave that is "better" or more "ideal" than the others.

Each serves a purpose to help us adapt to different circumstances; whether it is to enable us to process and learn new data or enable us to quiet down following a long distressing day. The five brain waves in order of highest frequency to lowest are as follows: gamma, beta, alpha, theta, and delta.

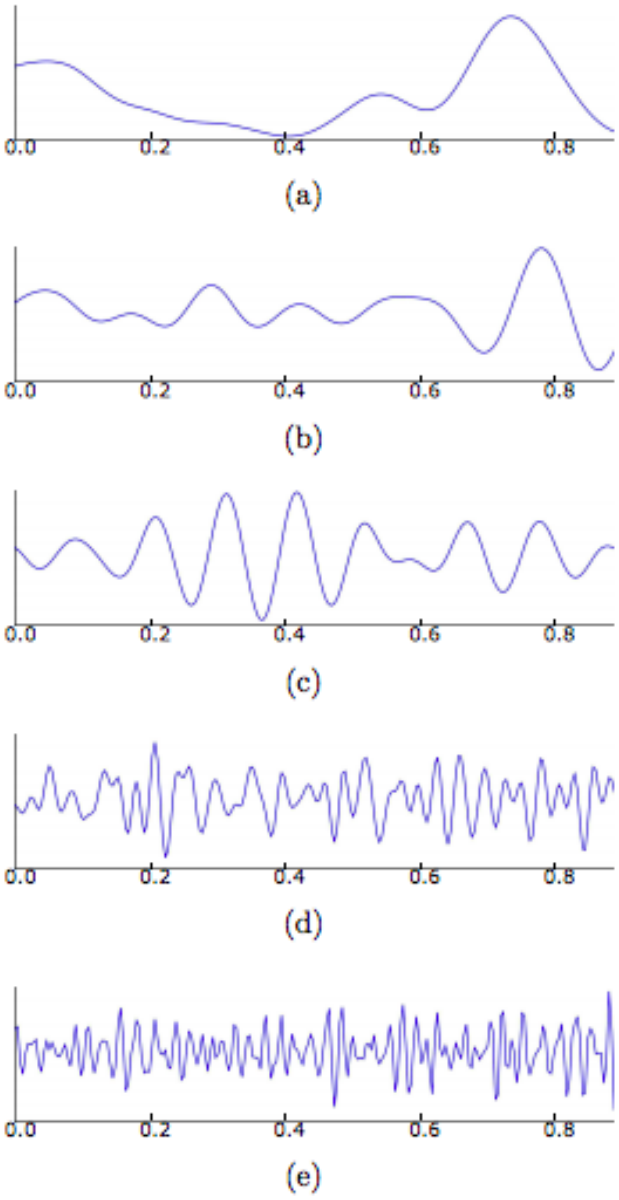

Figure 2.1: The five types of frequency bands ((a) delta, (b) theta, (c) alpha, (d) beta, (e) gamma) that is identified physiologically in human EEG signals.

K. SURESH MANIC (2014)

Gamma waves

These are involved in higher processing tasks as well as cognitive functioning. Gamma waves are critical for learning, memory and data handling. It is thought that the 40 Hz gamma wave is important for the binding of our senses in regards to perception and are involved in learning new material. It has been discovered that people who are mentally challenged and have learning handicaps have a tendency to have lower gamma activity than normal.

- **Frequency range**: 40 Hz to 100 Hz
- **Too much**: Anxiety, high arousal, stress
- **Too little**: ADHD, depression, learning disabilities
- **Optimal**: Binding senses, cognition, information processing, learning, perception, REM sleep

Beta waves

These are known as high frequency low amplitude brain waves that are commonly observed while we are alert. They are associated with cognizant idea, intelligent reasoning, and have a tendency to have an invigorating impact. Having the appropriate measure of beta waves enables us to center and finish school or work-based assignments effectively. Having excessively beta may prompt us encountering intemperate pressure or uneasiness. The higher beta frequencies are related with elevated amounts of excitement. When you drink caffeine or have another stimulant, your beta action will normally increment. Think about these as being very fast brain waves that a great many people show for the duration of the day in order to complete conscious tasks such as: critical thinking, writing, reading, and socialization.

- **Frequency range**: 12 Hz to 40 Hz
- **Too much**: Adrenaline, anxiety, high arousal, stress

- **Too little**: ADHD, daydreaming, depression, poor cognition
- **Optimal**: Conscious focus, memory, problem solving
- **Increase beta waves**: Coffee, energy drinks, various stimulants

Alpha waves

This frequency range bridges the gap between our conscious thinking and subconscious mind. In other words, alpha is the frequency range between beta and theta. It causes us quiet down when essential and advances sentiments of profound unwinding. In the event that we wind up focused on, a marvel called "alpha blocking" may happen which includes over the top beta movement and almost no alpha. Basically the beta waves "obstruct" out the generation of alpha since we turn out to be excessively stirred.

- **Frequency range**: 8 Hz to 12 Hz
- **Too much**: Daydreaming, inability to focus, too relaxed
- **Too little**: Anxiety, high stress, insomnia, OCD
- **Optimal**: Relaxation
- **Increase alpha waves**: Alcohol, marijuana, relaxants, some antidepressants

Theta waves

This particular frequency range is involved in daydreaming and sleep. Theta waves are associated with us encountering and feeling profound and crude feelings. An excessive amount of theta waves may make individuals inclined to episodes of misery and may make them "exceedingly suggestible" in view of the way that they are in a profoundly casual, semi-sleep inducing state. Theta has its advantages of enhancing our instinct, imagination, and influences us to feel more regular. It is additionally associated with remedial rest. For whatever length of

time that theta isn't created in abundance amid our waking hours, it is an exceptionally supportive brain wave range.

- **Frequency range**: 4 Hz to 8 Hz
- **Too much**: ADHD, depression, hyperactivity, impulsivity, inattentiveness
- **Too little**: Anxiety, poor emotional awareness, stress
- **Optimal**: Creativity, emotional connection, intuition, relaxation
- **Increase theta waves**: Depressants

Delta waves

These are the slowest recorded brain waves in human beings. They are found most often in infants as well as young children. As we age, we tend to produce less delta even during deep sleep. They are associated with the deepest levels of relaxation and restorative, healing sleep. They have also been found to be involved in unconscious bodily functions such as regulating heartbeat and digestion. Adequate production of delta waves helps us feel completely rejuvenated after we wake up from a good night's sleep. If there is abnormal delta activity, an individual may experience learning disabilities or have difficulties maintaining conscious awareness.

- **Frequency range**: 0 Hz to 4 Hz
- **Too much**: Brain injuries, learning problems, inability to think, severe ADHD
- **Too little**: Inability to rejuvenate body, inability to revitalize the brain, poor sleep
- **Optimal**: Immune system, natural healing, restorative / deep sleep
- **Increase delta waves**: Depressants, sleep

2.2 Input

In this part we will be looking into the input data used. A discussion of the EEG data set used and a brief about the chirp signal.

2.2.1 EEG Data

EEG data for testing was obtained from a study in R. Leeb (2007).

The subjects were right-handed, had normal or corrected-to-normal vision and were paid for participating in the experiments. All volunteers were sitting in an armchair, watching a flat screen monitor placed approximately 1 m away at eye level. For each subject 5 sessions are provided, whereby the first two sessions contain training data without feedback, and the last three sessions were recorded with feedback.

Three bipolar recordings were recorded with a samplingfrequency of 250 Hz. The recordings had a dynamic range of $\pm 100\mu V$ for the screening and $\pm 50\mu V$ for the feedback sessions.

2.2.2 Chirp Signal

A chirp is a signal in which the frequency increases (up-chirp) or decreases (down-chirp) with time. In some sources, the term chirp is used interchangeably with sweep signal.

$y = chirp(t, f0, t1, f1)$ is the Matlab function used to generate samples of a linear swept-frequency cosine signal at the time instances defined in array t, where $f0$ is the instantaneous frequency at time 0, and $f1$ is the instantaneous frequency at time $t1$. $f0$ and $f1$ are both in hertz. If unspecified, $f0$ is e^{-6} for logarithmic chirp and 0 for all other methods, $t1$ is 1, and $f1$ is 100.

$y = chirp(t, f0, t1, f1, 'method')$ specifies alternative sweep method options,

where method can be:

- linear, which specifies an instantaneous frequency sweep $f_i(t)$ given by $f_i(t) = f_0 + \beta_t$ where $\beta = (f_1 f_0)/t_1$ and the default value for f_0 is 0. β ensures that the desired frequency breakpoint f_1 at time t_1 is maintained.
- quadratic, which specifies an instantaneous frequency sweep $f_i(t)$ given by $f_i(t) = f_0 + \beta_t^2$ where $\beta = (f_1 f_0)/t_1^2$ If $f_0 > f_1$ (downsweep), the default shape is convex. If $f_0 < f_1$ (upsweep), the default shape is concave
- logarithmic, which specifies an instantaneous frequency sweep $f_i(t)$ given by $f_i(t) = f_0 \beta^t$ where $\beta = (\frac{f_1}{f_0})^{\frac{1}{t_1}}$ and the default value for f_0 is $1e^{-6}$. Both an upsweep ($f_1 > f_0$) and a downsweep ($f_0 > f_1$) of frequency is possible.

2.3 Techniques Used

In the section we will be looking at various techniques and matlab functions employed in the research.

2.3.1 Butterworth filter

The Butterworth filter is a type of signal processing filter designed to have a frequency response as flat as possible in the passband. It is also referred to as a maximally flat magnitude filter.

Lowpass butterworth filter

$[b, a] = butter(n, Wn)$ returns the transfer function coefficients of an n^{th}-order lowpass digital Butterworth filter with normalized cutoff frequency Wn.

Transfer function coefficients (b, a) Transfer function coefficients of the filter, returned as row vectors of length n + 1 for lowpass and highpass filters and 2n + 1 for bandpass and bandstop filters.

- For digital filters, the transfer function is expressed in terms of b and a as

$$H(z) = \frac{B(z)}{A(z)} = \frac{b(1) + b(2)z^{(-1)} + ... + b(n+1)z^n}{a(1) + a(2)z^1 + ... + a(n+1)z^n}$$

- For analog filters, the transfer function is expressed in terms of b and a as

$$H(s) = \frac{B(s)}{A(s)} = \frac{b(1)s^n + b(2)s^{n1} + ... + b(n+1)}{a(1)s^n + a(2)s^{n1} + ... + a(n+1)}$$

Bandpass butterworth filter

$\big[A, B, C, D\big] = butter(n, Wn)$ designs a lowpass, highpass, bandpass, or bandstop digital Butterworth filter and returns the matrices that specify its state-space representation

A,B,C,D State-space matrices State-space representation of the filter, returned as matrices. If $m = n$ for lowpass and highpass designs and $m = 2n$ for bandpass and bandstop filters, then A is $m \times m$, B is $m \times 1$, C is $1 \times m$, and D is 1×1.

- For digital filters, the state-space matrices relate the state vector x, the input u, and the output y through

$$x(k+1) = Ax(k) + Bu(k)$$

$$y(k) = Cx(k) + Du(k)$$

- For analog filters, the state-space matrices relate the state vector x, the input u, and the output y through

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

2.4 Architecture of network system

The input signal $P_{teach}(t)$ is first passed through the first layer of the system of networks, adaptive Hopf oscillator to identify the frequency components. The signal then passes through the second layer, filter bank with cut-off for different brain wave frequencies.

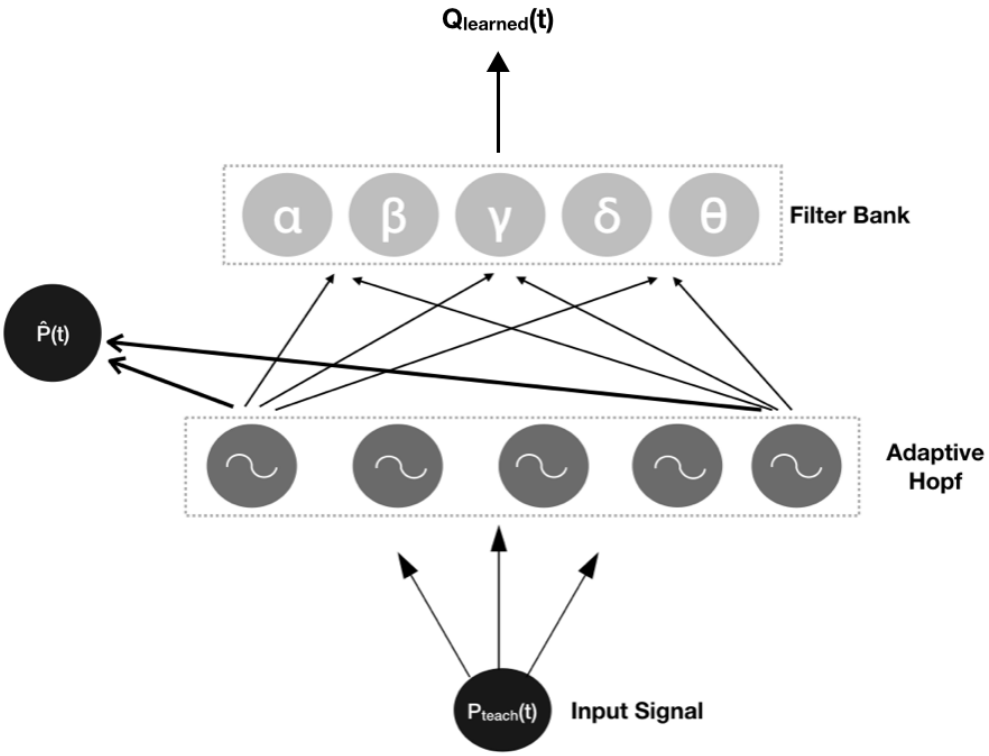

Figure 2.2: Network Architecture

CHAPTER 3

Methedology and Implementation

3.1 Introduction

EEG signal is measurement of the electrical activity is produced by the neurons network of the brain when brain is made any mental task. The EEG signal is consist of very low frequency components and amplitude so diff. types of noises and artifacts are overlapping with EEG signal like as base line wondering, eye blinking, eye movement, breathing etc. The main aim of the project is to extract the brain waves from EEG signal. The brain waves could be of great use in Brain Computer Interface applications. EEG signal is complex signal. We are filtering the EEG signal by using diff types of filters to remove the unwanted frequency components from the original signal and to extract the features or information about the signal with the help of different efficient DSP tools like FFT.

The basic architecture of the networks is the same for all the trials performed. The trained system was tested on EEG data for all the performed trials.

Fast Fourier Transform

The FFT is an important and efficient tool for the feature extraction. FFT algorithm is involved a wide range of mathematical operation from simple real and complex numbers arithmetic to group theory and no. theory. The FFT is can compute the result $O(NlogN)$ operation.

Where N is the length of the vector

There will be the range of the N is thousand or million so that DFT is not the suitable method so we are used the FFT. The calculation is very complex and time consuming to reduce the operation time and increasing the speed by using FFT. FFTs are of most importance tool to a wide variety of applications, from digital signal processing and solving partial differential equations to algorithms for quick multiplication of large integers.

Shakshi (2016)

3.2 Trial 1

3.2.1 Introduction

The input signal for training was chosen as chirp signal, which is passed through the Hopf oscillator and for identifying the frequency componenents. The output from Hopf is passed through the filter designed based on perceptron algorithm. The decide signal for the filter was also chosen as chirp signal.

The trained system was then tested with an EEG input of 2 seconds.

Chirp Parameter (Input) $Fs = 250; Ts = 1/Fs; L = 8; t = 0 : Ts : L; P = chirp(t, 0, 8, 20)$. Which is a chirp signal of 0 - 20 Hz for 8 sec.

Hopf Parameters $N = 200; R = rand(N, 1); phi = 2{*}pi{*}rand(N, 1); fmax = 50; fmin = 1; omega = 2{*}pi{*}((fmax-fmin){*}rand(N, 1)+fmin); alphaa = 0.1 * rand(N, 1); epsl = 5; meu = 1; etaa = 0.2; nepoch = 2000;$ $trainingtime$

Decide Signal Low pass Filter(LPF) $Fs = 250; Ts = 1/Fs; L = 8; t = 0 : Ts : L; P = chirp(t, 0, 8, 5)$. Which is a chirp signal of 0 - 5 Hz for 8 sec.

Decide Signal Band pass Filter(BPF) $Fs = 250; Ts = 1/Fs; L = 8; t = 0 : Ts : L; P = chirp(t, 5, 8, 15)$. Which is a chirp signal of 5 - 15 Hz for 8 sec.

Decide Signal Low pass Filter(HPF) $Fs = 250; Ts = 1/Fs; L = 8; t = 0 :$

$Ts : L; P = chirp(t, 15, 8, 20)$. Which is a chirp signal of 15 - 20 Hz for 8 sec.

3.2.2 Results

The Hopf trained the chirp signal pretty well and the filter output was almost as expected.

Hopf Output

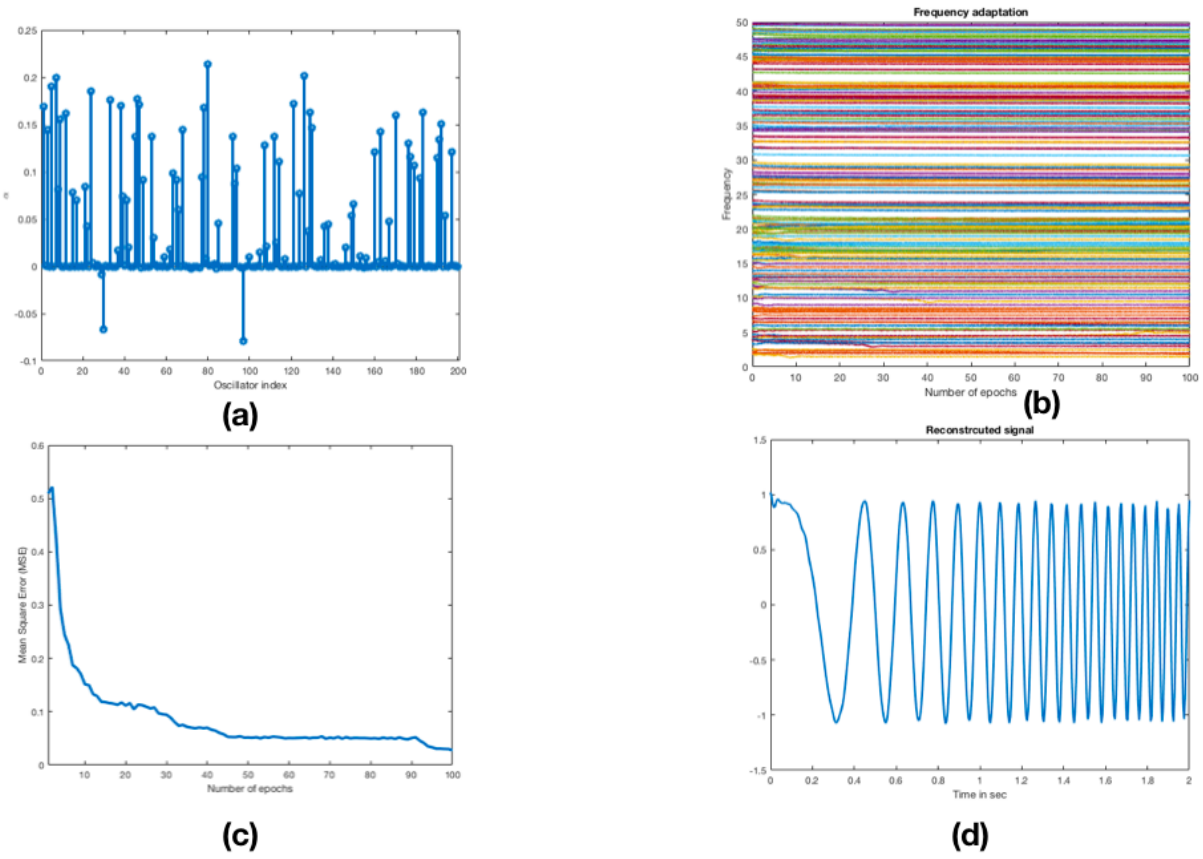

Figure 3.1: Hopf Output

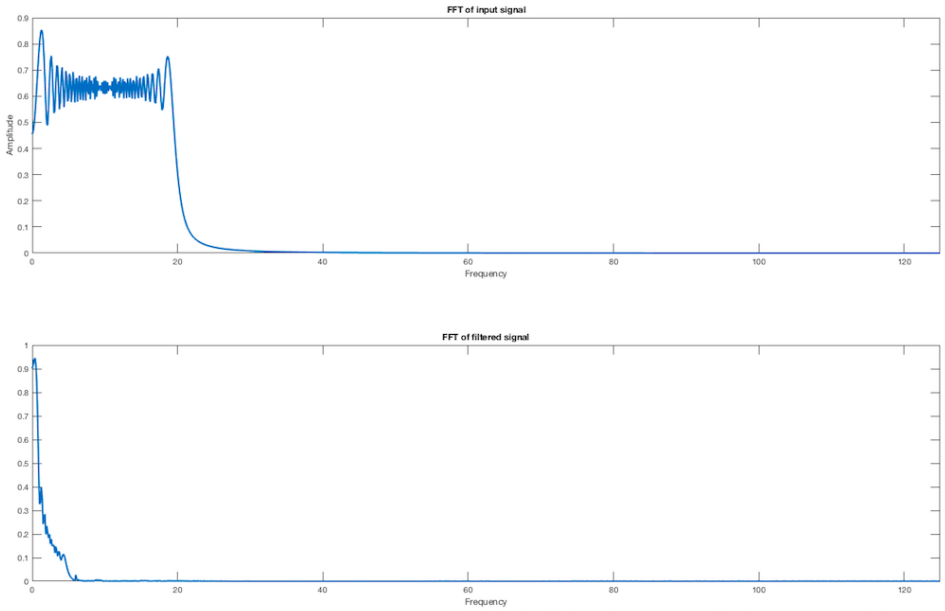

Figure 3.2: LPF output for training

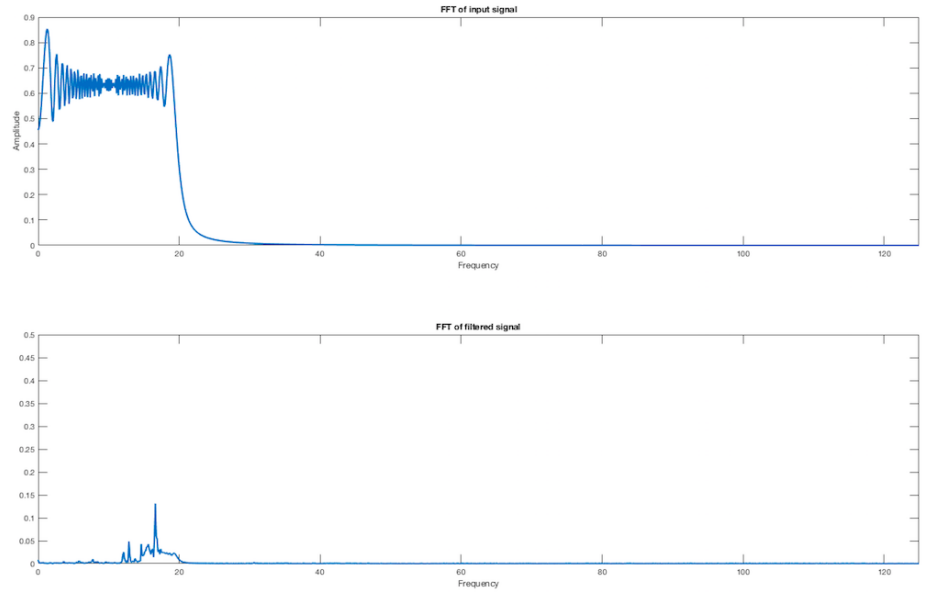

Figure 3.3: HPF Output for training

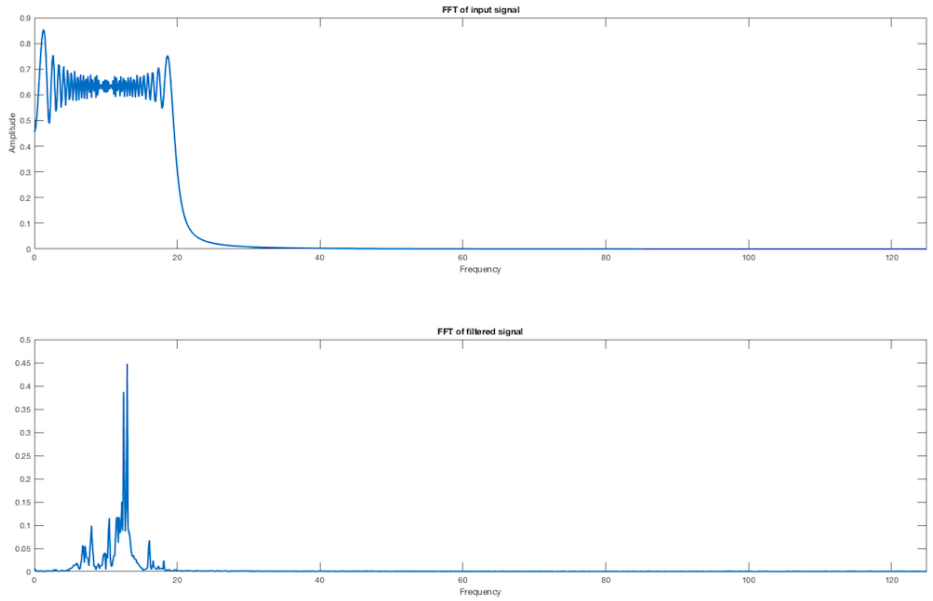

Figure 3.4: BPF Output for training

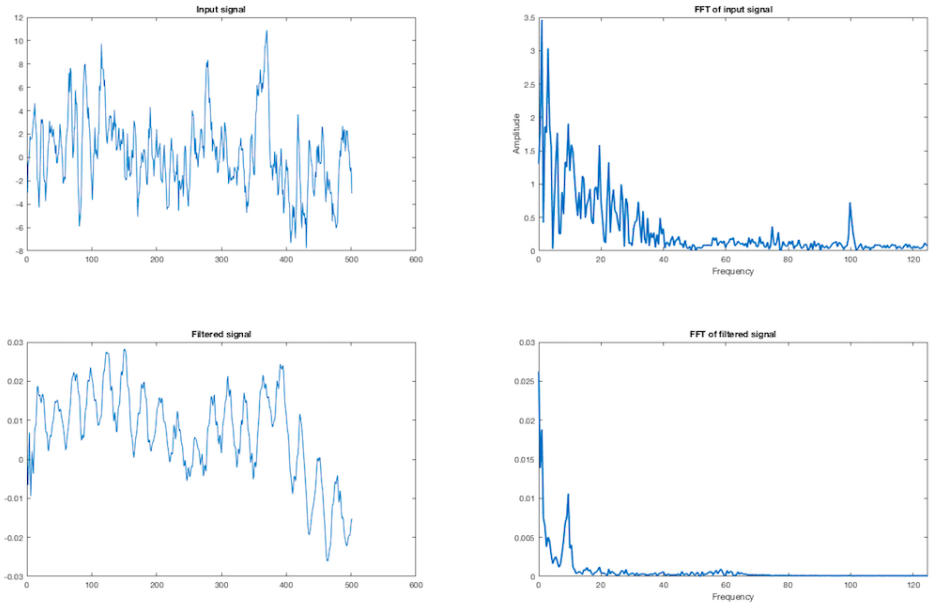

Figure 3.5: LPF Output for testing

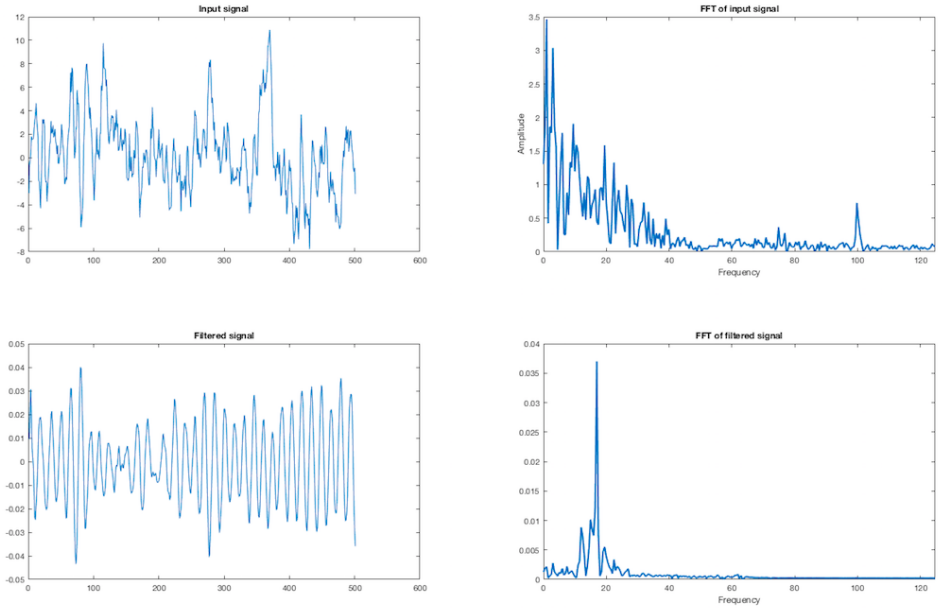

Figure 3.6: HPF Output for testing

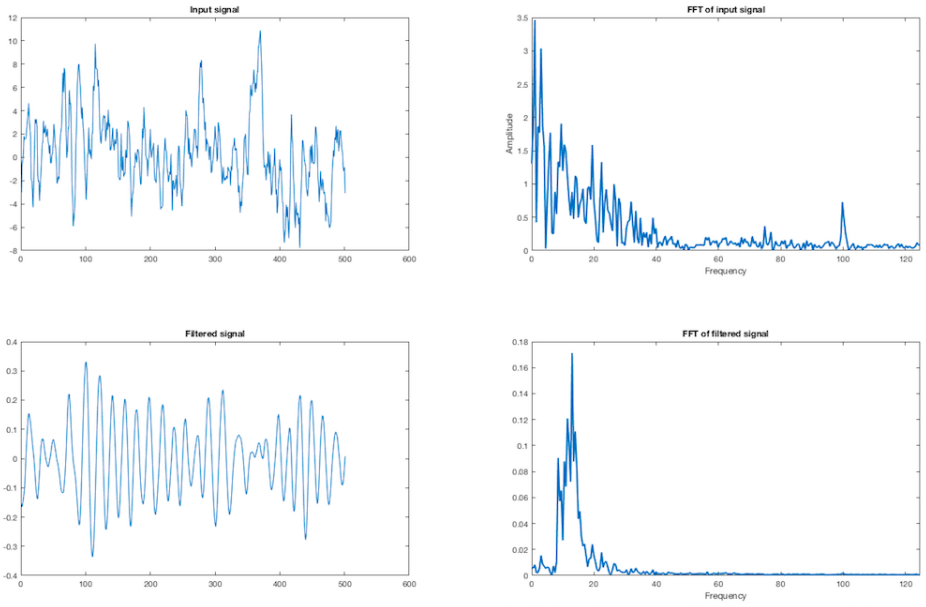

Figure 3.7: BPF Output for testing

3.3 Trail 2

3.3.1 Introduction

The input signal for training was chosen as EEG signal (0 - 125 Hz for 8 sec), which is passed through the Hopf oscillator for identifying the frequency componenents. The output from Hopf is further passed through the filter designed based on perceptron algorithm. The decide signal for the filter was chosen as chirp signal.

The trained system was then tested with an EEG input of 2 seconds.

Hopf Parameters $N = 200; R = rand(N, 1); phi = 2{*}pi{*}rand(N, 1); fmax = 50; fmin = 1; omega = 2{*}pi{*}((fmax - fmin){*}rand(N, 1) + fmin); alphaa = 0.1 * rand(N, 1); epsl = 5; meu = 1; etaa = 0.2; nepoch = 2000;$
$trainingtime$

Decide Signal Low pass Filter(LPF) $Fs = 250; Ts = 1/Fs; L = 8; t = 0 : Ts : L; P = chirp(t, 0, 8, 5)$. Which is a chirp signal of 0 - 5 Hz for 8 sec.

Decide Signal Band pass Filter(BPF) $Fs = 250; Ts = 1/Fs; L = 8; t = 0 : Ts : L; P = chirp(t, 5, 8, 15)$. Which is a chirp signal of 5 - 15 Hz for 8 sec.

Decide Signal Low pass Filter(HPF) $Fs = 250; Ts = 1/Fs; L = 8; t = 0 : Ts : L; P = chirp(t, 15, 8, 20)$. Which is a chirp signal of 15 - 20 Hz for 8 sec.

3.3.2 Results

The Hopf trained the chirp signal pretty well and the filter output was almost as expected.

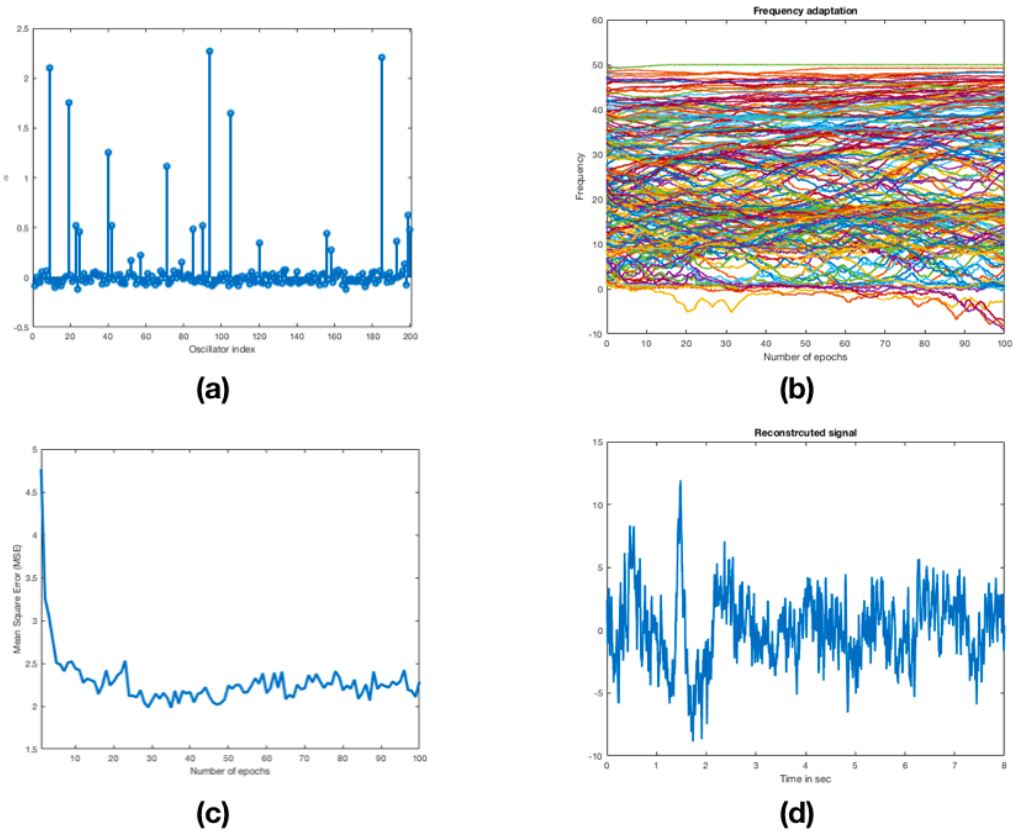

Figure 3.8: Hopf Output

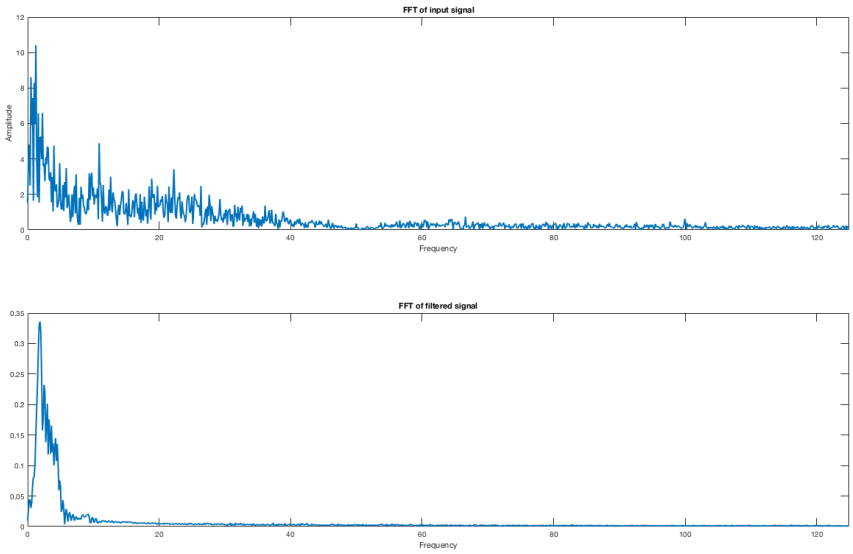

Figure 3.9: LPF output for training

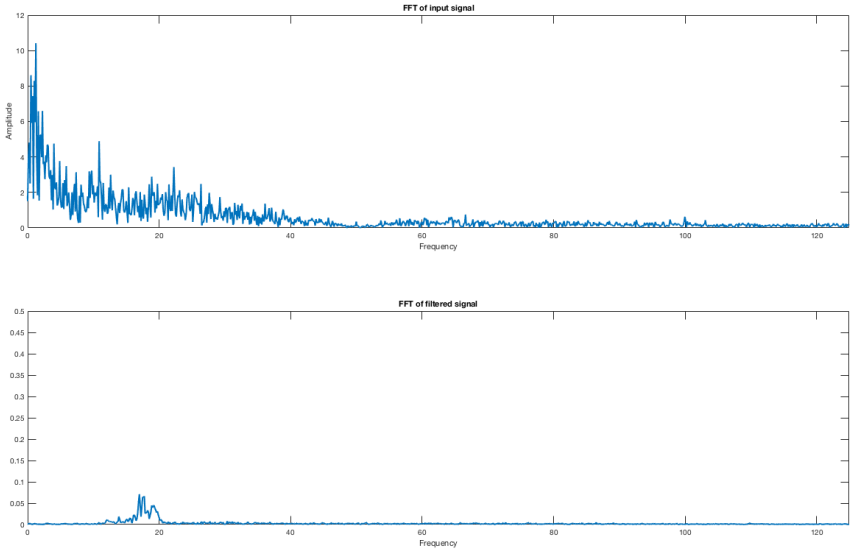

Figure 3.10: HPF Output for training

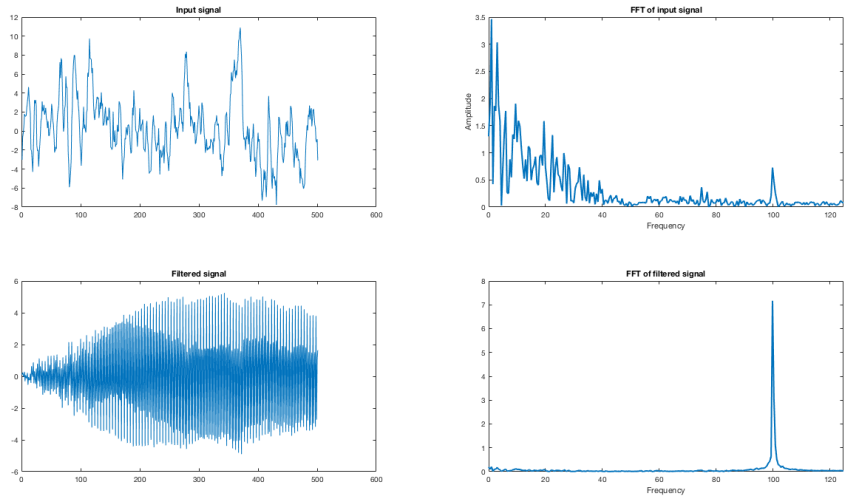

Figure 3.11: BPF Output for training

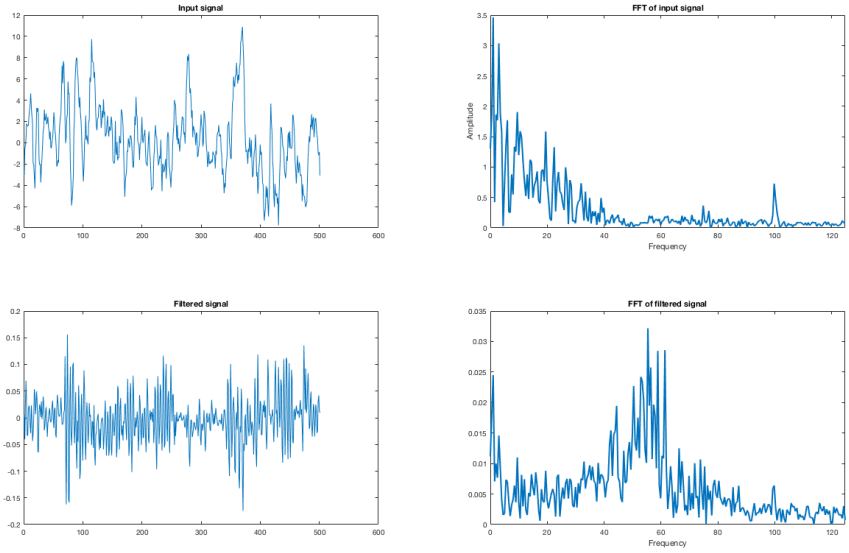

Figure 3.12: LPF Output for testing

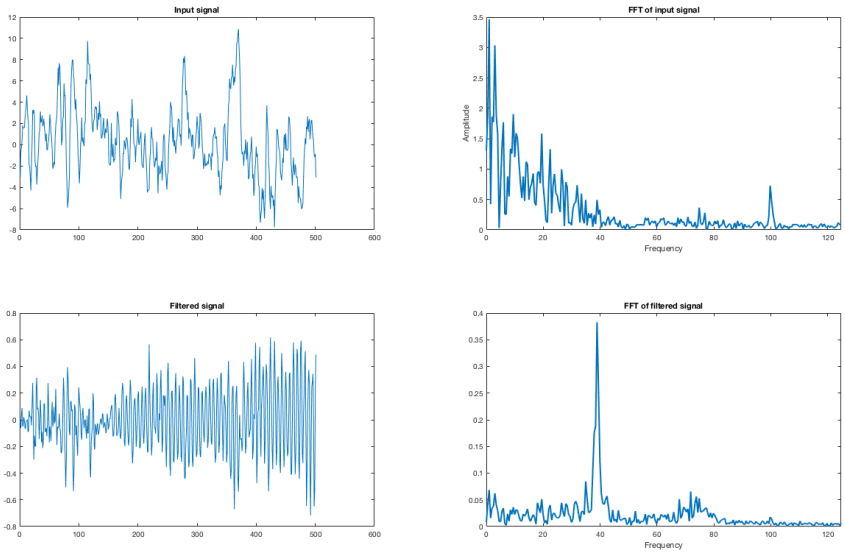

Figure 3.13: HPF Output for testing

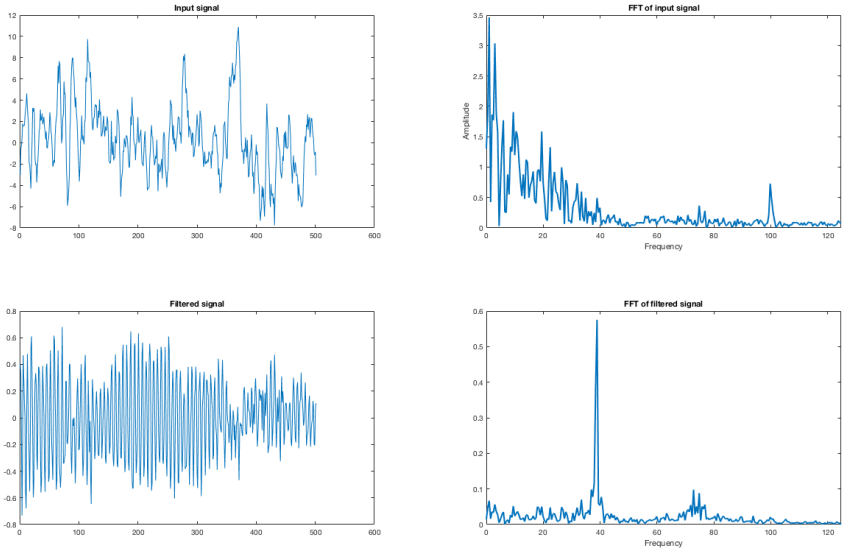

Figure 3.14: BPF Output for testing

3.4 Trail 3

3.4.1 Introduction

The input signal for training was chosen as EEG signal (0 - 125 Hz for 8 sec), which is passed through the Hopf oscillator for identifying the frequency componenents. The output from Hopf is further passed through the filter designed based on perceptron algorithm. The decide signal for the filter was chosen output from butterworth filter.

The trained system was then tested with an EEG input of 2 seconds.

Hopf Parameters $N = 200; R = rand(N, 1); phi = 2{*}pi{*}rand(N, 1); fmax = 50; fmin = 1; omega = 2{*}pi{*}((fmax - fmin){*}rand(N, 1) + fmin); alphaa = 0.1 * rand(N, 1); epsl = 5; meu = 1; etaa = 0.2; nepoch = 2000;$
$trainingtime$

Decide Signal alpha wave

$$fc = [812]; [A, B, C, D] = butter(10, fc/(Fs/2));$$

d = designfilt('bandpassiir','FilterOrder',20, ... 'HalfPowerFrequency1',8,'HalfPowerFrequency2',12, ... 'SampleRate',250); dataOutalpha = filter(d,P);

Decide Signal -

$$Dec = dataOutalpha;$$

Decide Signal beta wave

$$fc = [1240]; [A, B, C, D] = butter(10, fc/(Fs/2));$$

d = designfilt('bandpassiir','FilterOrder',20, ... 'HalfPowerFrequency1',12,'HalfPowerFrequency2',40, ... 'SampleRate',250); dataOutbeta = filter(d,P);

Decide Signal -

$$Dec = dataOutbeta;$$

Decide Signal gamma wave

$$fc = [40100]; [A, B, C, D] = butter(10, fc/(Fs/2));$$

d = designfilt('bandpassiir','FilterOrder',20, ... 'HalfPowerFrequency1',8,'HalfPowerFrequency2',12, ... 'SampleRate',250); dataOutgamma = filter(d,P);

Decide Signal -

$$Dec = dataOutgamma;$$

Decide Signal delta wave

$$fc = [04]; [A, B, C, D] = butter(10, fc/(Fs/2));$$

d = designfilt('bandpassiir','FilterOrder',20, ... 'HalfPowerFrequency1',0,'HalfPowerFrequency2',4, ... 'SampleRate',250); dataOutdelta = filter(d,P);

Decide Signal -

$$Dec = dataOutdelta;$$

Decide Signal theta wave

$$fc = [48]; [A, B, C, D] = butter(10, fc/(Fs/2));$$

d = designfilt('bandpassiir','FilterOrder',20, ... 'HalfPowerFrequency1',4,'HalfPowerFrequency2',8, ... 'SampleRate',250); dataOuttheta = filter(d,P);

Decide Signal -

$$Dec = dataOuttheta;$$

3.4.2 Results

The Hopf trained the EEG signal pretty well and the filter output was almost as expected.

Hopf Output

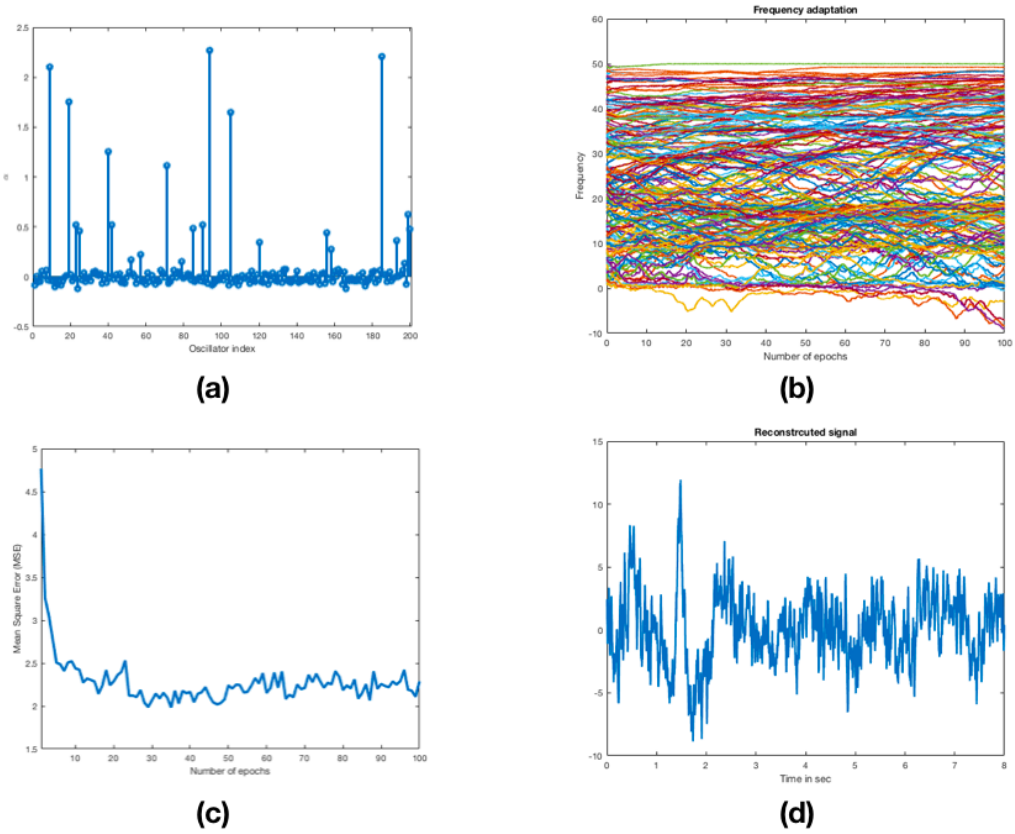

Figure 3.15: Hopf Output

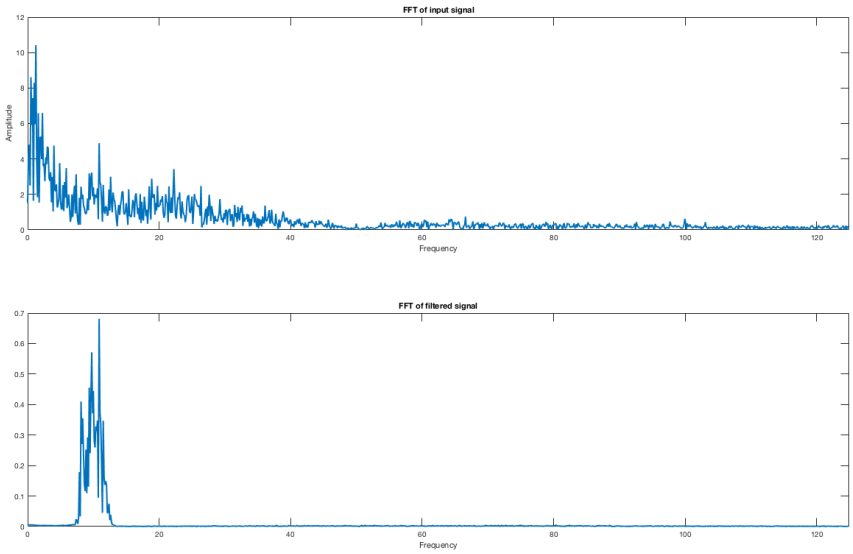

Figure 3.16: Alpha output for training

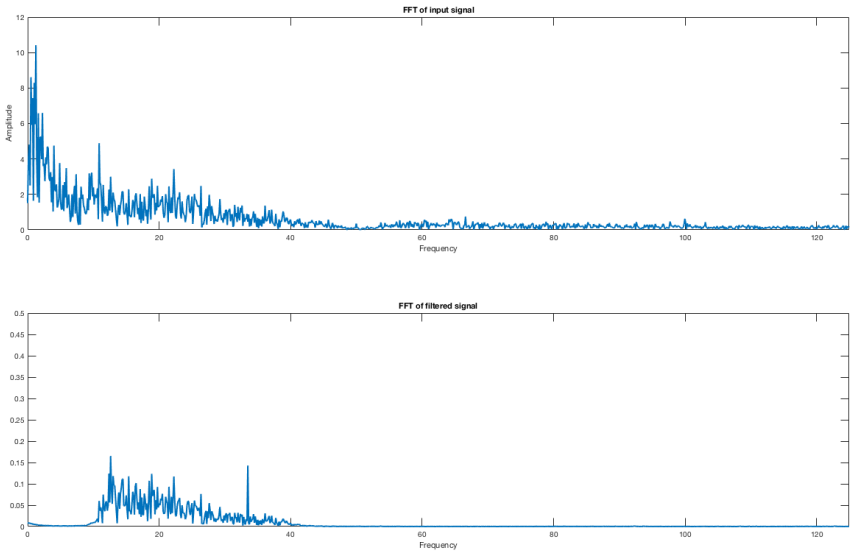

Figure 3.17: Beta Output for training

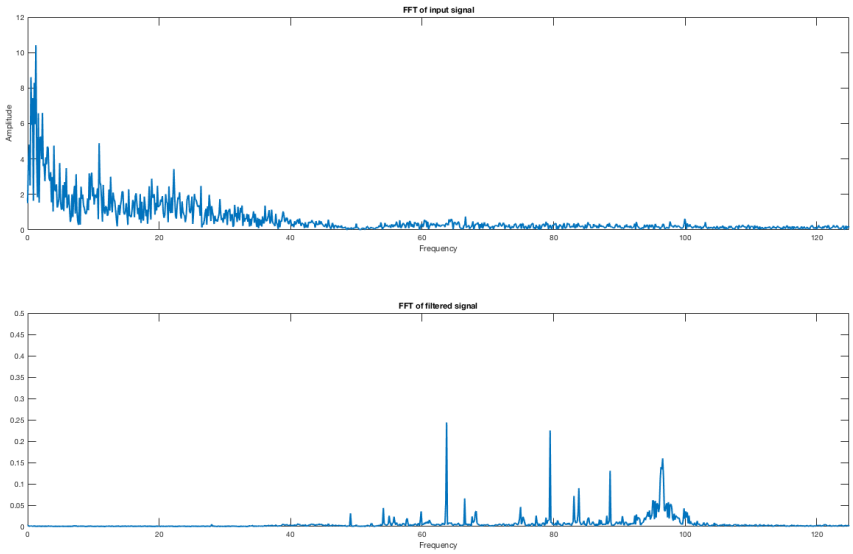

Figure 3.18: Gamma Output for training

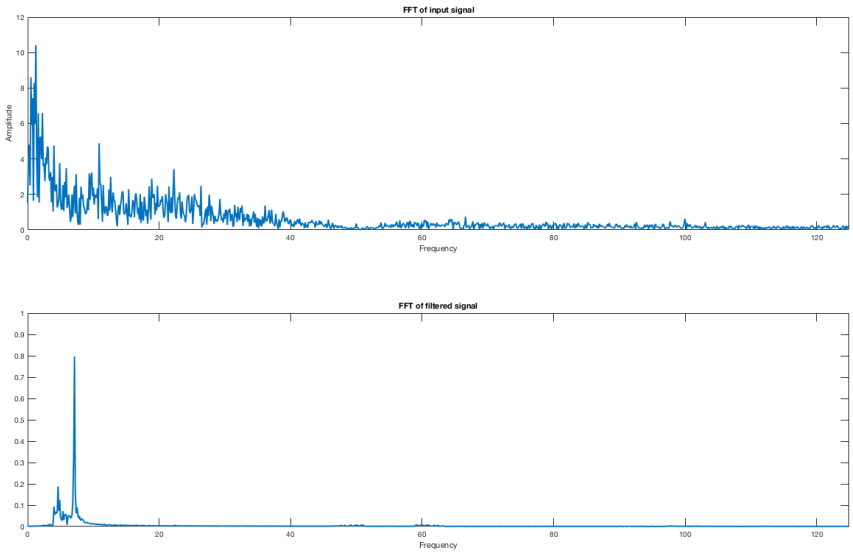

Figure 3.19: Theta Output for testing

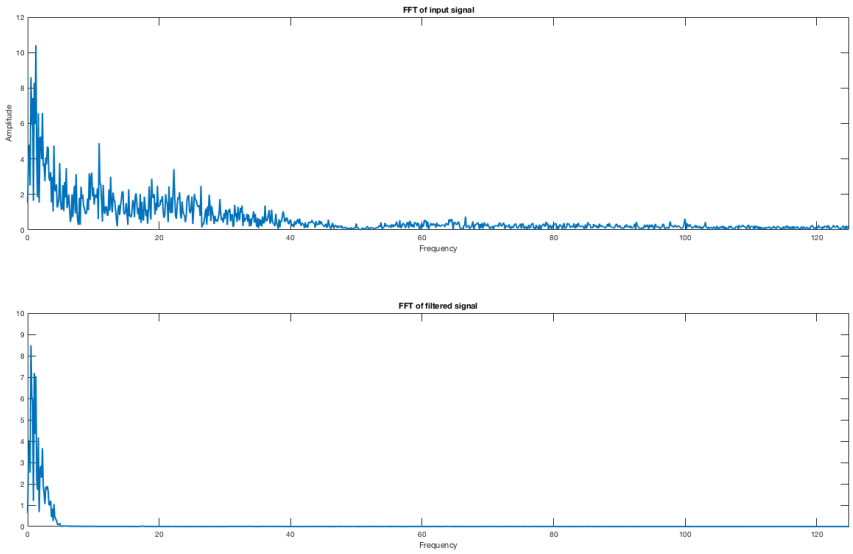

Figure 3.20: Delta Output for testing

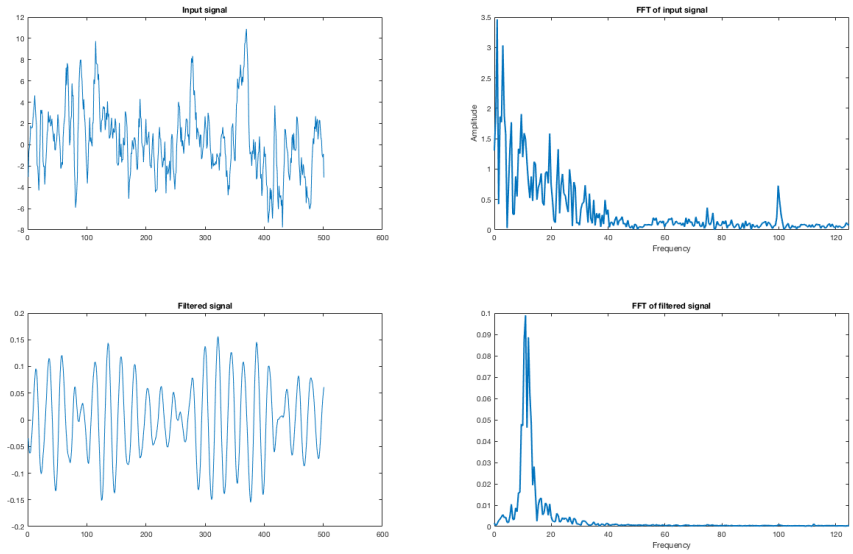

Figure 3.21: Alpha output for training

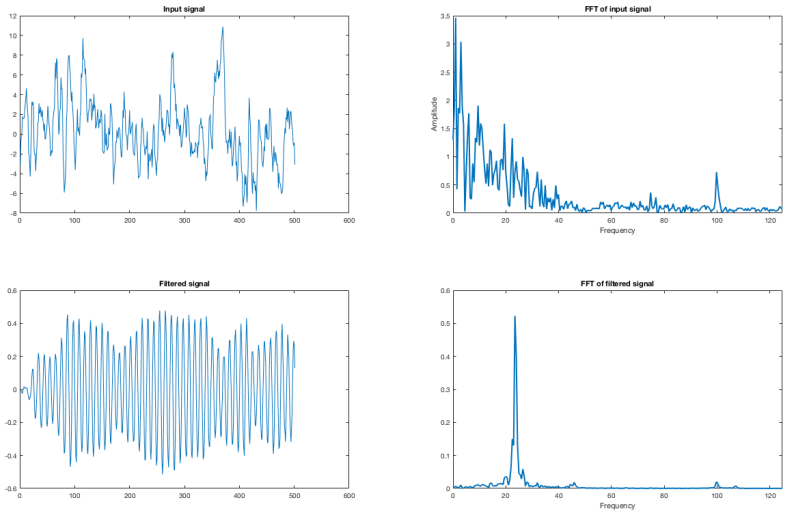

Figure 3.22: Beta Output for training

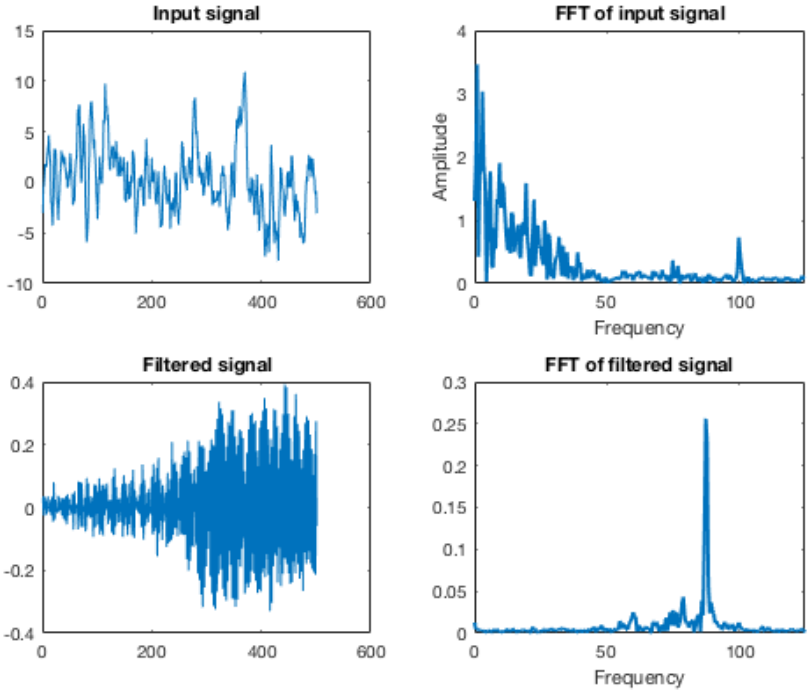

Figure 3.23: Gamma Output for training

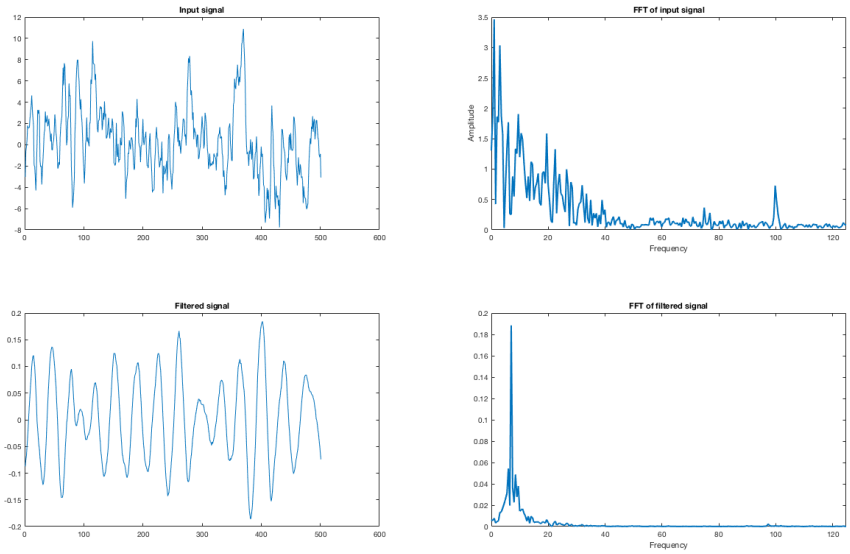

Figure 3.24: Theta Output for testing

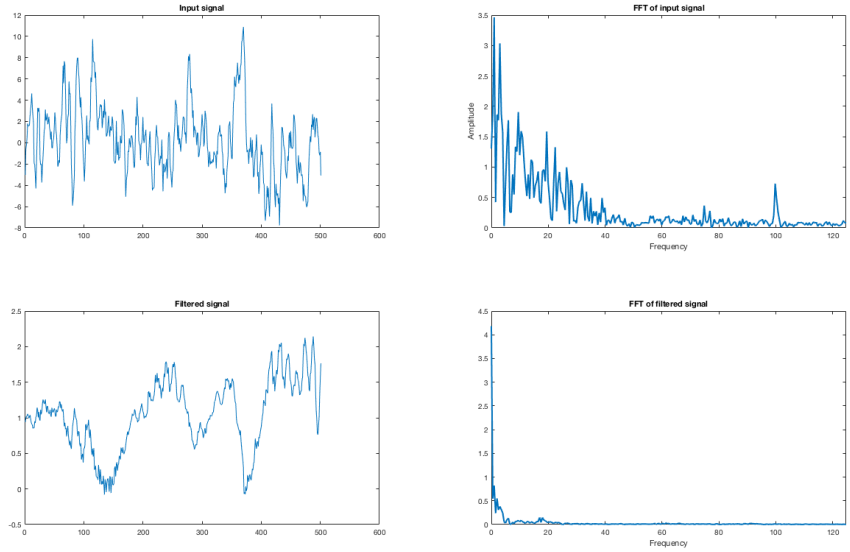

Figure 3.25: Delta Output for testing

CHAPTER 4

CONCLUSIONS

4.1 Hopf Oscillator

We have developed a very simple algorithm for Hopf oscillator. The algorithm worked good for all the three cases. The oscillator successfully reconstructed the input signal with minimal MSE.

4.2 Filter Network

The filter showed expected output in Trial 3. The peaks were not clearly distinguished in Trials 1 and 2. This could be possibly beacause of the difference in data used for training and testing

A clear shift in the bands can be seen for both training and testing in Trial 3.

4.3 Future Applications

With recent advances in signal processing and biomedical instrumentation, EEG signals can be utilized as another correspondence channel amongst human and computers. Usage of this channel is conceivable by recording and dissecting mind waves. EEG based BCI innovations that don't rely upon peripheral nerves and muscles have gotten much consideration as conceivable methods of correspondence for the disabled. The filters designed can be used for feature extraction

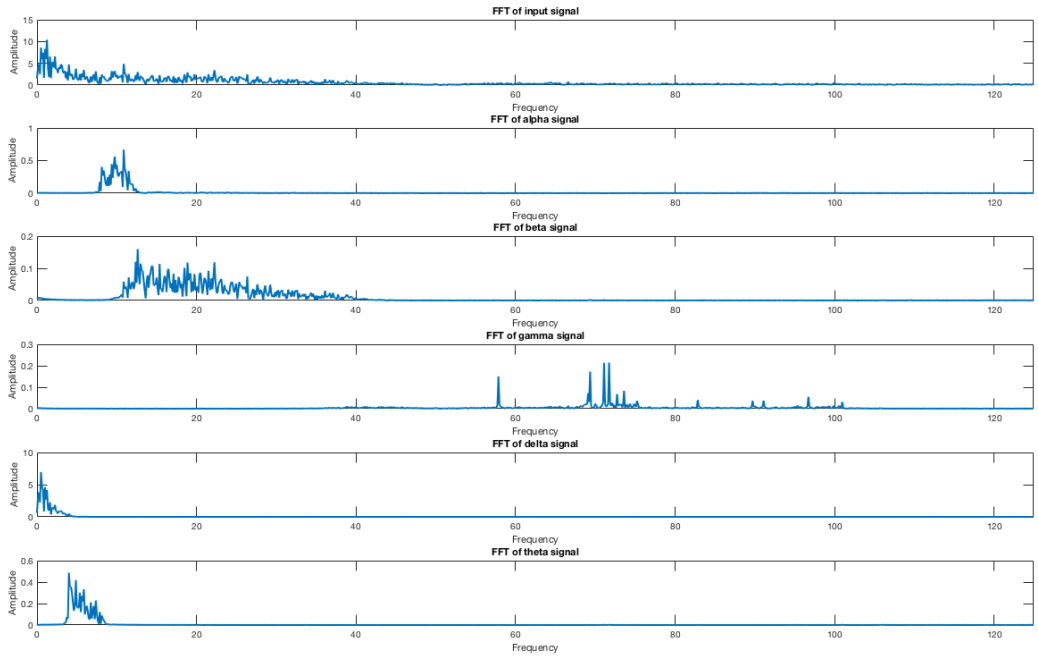

Figure 4.1: EEG Output for Training

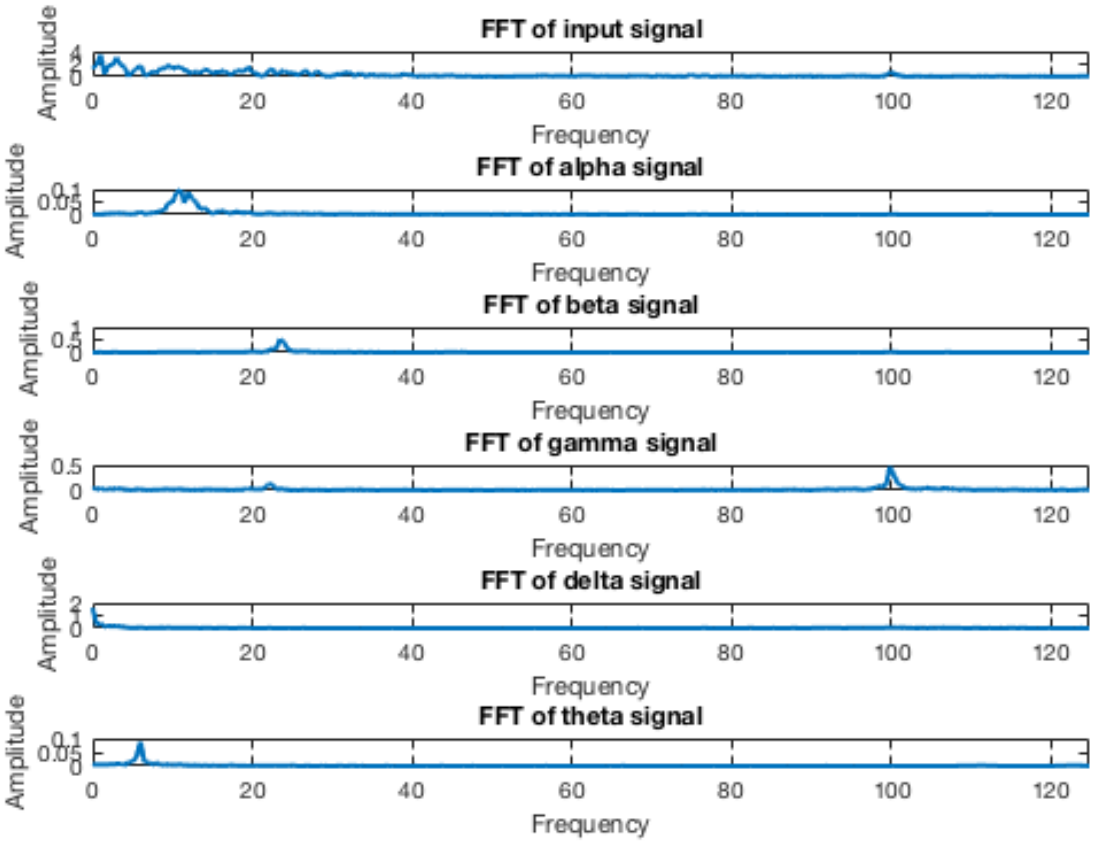

Figure 4.2: EEG Output for Testing

and classification of EEG signals, which could turn out to be of great use in BCI application.

BCI Applications

- **Medical applications**: Human services field has an assortment of uses that could exploit brain signals in all related stages including prevention, detection, diagnosis, rehabilitation and restoration.
- **Smart Environment**:Deploying brain signals is not exclusive to the medical field. Shrewd situations, for example, savvy houses, work environments or transportations could likewise exploit BCI in offering further wellbeing, extravagance and physiological control to people's every day life. They are also expected to witness cooperation between Internet Of Things (IOT) and BCI technologies
- **Games and entertainment**: VR games based on BCI
- **Security and authentication**

APPENDIX A

Matlab code used for training

```
clc
clear all
close all
%% Generation of input signal
Fs = 250;
Ts = 1/Fs;
L=8;
t = 0:Ts:L;

P = transpose(data{1,1}.X(1:2001,1));
%amp = 1e-5;
% f = [5 25]; amp = [1 0.5]; P=zeros(1,length(t));
% for ii=1:length(f)
%     P = P+amp(ii)*sin(2*pi*f(ii)*t);
% end
figure; plot(t,P,'Linewidth',2); title('Input signal'); xlabel('
    Time in seconds');
L = length(P)*Ts;      % signal duration
t = 0:1/Fs:L-1/Fs; % Time vector
f = -(Fs-1/L)/2:1/L:(Fs-1/L)/2;  % Frequency vector
fft1 = abs((2/Fs)*fft(P));
figure;
plot(f,abs(fftshift(fft1)),'Linewidth',2);
xlim([0 Fs/2]);
%% Training Hopf system
N = 200;
```

```
R = rand(N,1);
phi = 2*pi*rand(N,1);
fmax = 50; fmin = 1;
% omega = 2*pi*[2  18]';
omega = 2*pi*((fmax-fmin)*rand(N,1)+fmin);
alphaa = 0.1*rand(N,1);
epsl = 5;
meu = 1;
etaa = 0.2;
nepoch = 3000;
dt = Ts;
for ii=1:nepoch
    ii
    for jj=1:length(t)
        X = R.*cos(phi); Xarr{ii}(:,jj) = X;
        Phat = alphaa'*X; Phatarr(ii,jj) = Phat;
        I = P(jj) - Phat; err(ii,jj) = I.^2;
        Rdot = R.*(meu-R.^2) + epsl*I.*cos(phi);
        phidot = omega - (epsl*(I.*sin(phi))./R);
        omegadot = -epsl*I.*sin(phi);
        alphaadot = etaa*X.*I;

        R = R + Rdot*dt; %R = (R>0).*R;
        phi = phi + phidot*dt; %phi = (phi>0).*phi;
        omega = omega + omegadot*dt; %omega = (omega>0).*omega;
        alphaa = alphaa + alphaadot*dt;

        Rarr{ii}(:,jj) = (R>0).*R;
        phiarr{ii}(:,jj) = phi;
        omegaarr{ii}(:,jj) = omega;
        alphaarr{ii}(:,jj) = alphaa;
    end
end
```

```
59 %% Plotting the results of Adaptive Hopf oscillators
60 % figure; stem(Rarr(:,end)'); xlim([0 N+1]); xlabel('Oscillator
       index'); title('R')
61 figure; plot(t,Phatarr(nepoch,:),'Linewidth',2); title('
       Reconstrcuted signal'); xlabel('Time in sec')
62 figure; stem(alphaa,'Linewidth',3); xlim([0 N+1]); ylabel('\
       alpha'); xlabel('Oscillator index')
63 farr = cell2mat(omegaarr); farr = farr'/(2*pi); dur_epoch = 100;
        dur_len = floor(dur_epoch*length(P));
64 figure; plot(linspace(0,dur_epoch,dur_len),farr(1:dur_len,:),'
       Linewidth',2); title('Frequency adaptation'); xlabel('Number
       of epochs'); ylabel('Frequency')
65 MSE = mean(err,2);
66 figure; plot(MSE(1:dur_epoch),'Linewidth',3); xlabel('Number of
       epochs');ylabel('Mean Square Error (MSE)'); xlim([1 dur_epoch
       ])
67 %% Train the Hopf system to be a alpha filter
68 L = length(P)*Ts;      % signal duration
69 t = 0:1/Fs:L-1/Fs; % Time vector
70 Dec = dataOutalpha;
71
72 f = -(Fs-1/L)/2:1/L:(Fs-1/L)/2;  % Frequency vector
73 fftDec = abs((2/Fs)*fft(Dec));
74
75 xlim([0 Fs/8]);
76 W = rand(N,1);
77 Winit = W;
78 etaa2 = 0.05;
79 I = I/10;
80 nepoch = 4000;
81 count=1;
82 for ii=1:nepoch
83     ii
```

```
84      for jj=1:length(t)
85          X = R.*cos(phi);
86          yLPF = W'*X; yarrLPF(count) = yLPF;
87          eLPF = Dec(jj)-yLPF;
88
89          Rdot = R.*(meu-R.^2) + epsl*I.*cos(phi);
90          phidot = omega - (epsl*(I.*sin(phi))./R);
91          omegadot = -epsl*I.*sin(phi);
92          Wdot = etaa2*eLPF*X;
93
94          R = R + Rdot*dt; %R = (R>0).*R;
95          phi = phi + phidot*dt; %phi = (phi>0).*phi;
96          omega = omega + omegadot*dt; %omega = (omega>0).*omega;
97          W = W + Wdot*dt;
98          count = count + 1;
99      end
100 end
101 %freqz(1,W)
102 %s = omega.*W/(2*pi); s=(s>0).*s; s=sort(s)
103 %figure; plot(s);
104
105
106
107 % Plot LPF
108 % st = length(yarr)-200;
109 yarr_croppedLPF = yarrLPF(length(yarrLPF)-length(t)+1:length(
        yarrLPF));
110 %figure; plot(t,yarr_croppedLPF);
111
112 L = length(t)*Ts;      % signal duration
113 t = 0:1/Fs:L-1/Fs; % Time vector
114 f = -(Fs-1/L)/2:1/L:(Fs-1/L)/2;  % Frequency vector
115
```

```
116 fftalpha = abs((2/Fs)*fft(yarr_croppedLPF));
117 figure;
118 subplot(2,1,1);plot(f,fftshift(fft1),'Linewidth',2);xlim([0 max(
    f)]);title('FFT of input signal');
119 xlabel('Frequency'); ylabel('Amplitude');
120 subplot(2,1,2);plot(f,fftshift(fftalpha),'Linewidth',2);xlim([0
    max(f)]);title('FFT of filtered signal');
121 xlabel('Frequency'); %ylim([0 0.5])
122 fft1 = abs((2/Fs)*fft(P));
123 fftalpha = abs((2/Fs)*fft(yarr_croppedLPF));
124 figure; plot(f,fftshift(fft1),'Linewidth',2);xlim([0 max(f)]);
    title('FFT of input signal');
125 xlabel('Frequency'); ylabel('Amplitude');
126 figure; plot(f,fftshift(fftalpha),'Linewidth',2);xlim([0 max(f)
    ]);title('FFT of filtered signal');
127 xlabel('Frequency'); %ylim([0 0.5])
128 %% Gauusian Filter
129
130 % create filter
131 sigma = 1.25; % pick sigma value for the gaussian
132 gaussFilter = gausswin(6*sigma + 1)';
133 gaussFilter = gaussFilter / sum(gaussFilter); % normalize
134
135 filteredSig = conv(fft2, gaussFilter, 'same');
136 plot(f,fftshift(filteredSig),'Linewidth',2);xlim([0 20])
137
138 %% Sort
139 n = length(omega);
140 for j = 2:n
141     pivot = omega(j);
142     i = j;
143     while ((i > 1) && (omega(i - 1) > pivot))
144         omega(i) = omega(i - 1);
```

```
145             W(i) = W(i-1);
146             Winit(i) = Winit(i-1);
147             i = i - 1;
148         end
149         omega(i) = pivot;
150     end
151     figure; subplot(1,2,1); plot(omega/(2*pi),Winit,'Linewidth',2);
        title('FFT of System'); xlabel('Winit');
152     subplot(1,2,2); plot(omega/(2*pi),abs(W),'Linewidth',2); title('
        FFT of System'); xlabel('W');
153     %% Train the Hopf system to be a beta filter
154     Dec = dataOutbeta;
155     f = -(Fs-1/L)/2:1/L:(Fs-1/L)/2;  % Frequency vector
156     fft1 = abs((2/Fs)*fft(Dec));
157     figure;
158     plot(f,abs(fftshift(fft1)),'Linewidth',2);
159     xlim([0  150]);
160
161     W = rand(N,1);
162     Winit = W;
163     etaa2 = 0.05;
164     I = I/10;
165     nepoch = 4000;
166     count=1;
167     for ii=1:nepoch
168         ii
169         for jj=1:length(t)
170             X = R.*cos(phi);
171             yHPF = W'*X; yarrHPF(count) = yHPF;
172             eHPF = Dec(jj)-yHPF;
173
174             Rdot = R.*(meu-R.^2) + epsl*I.*cos(phi);
175             phidot = omega - (epsl*(I.*sin(phi))./R);
```

```
176             omegadot = -epsl*I.*sin(phi);
177             Wdot = etaa2*eHPF*X;
178
179             R = R + Rdot*dt; %R = (R>0).*R;
180             phi = phi + phidot*dt; %phi = (phi>0).*phi;
181             omega = omega + omegadot*dt; %omega = (omega>0).*omega;
182             W = W + Wdot*dt;
183             count = count + 1;
184         end
185 end
186 % st = length(yarr)-200;
187 yarr_croppedHPF = yarrHPF(length(yarrHPF)-length(t)+1:length(
        yarrHPF));
188 figure; plot(t,yarr_croppedHPF);
189
190 L = length(t)*Ts;      % signal duration
191 t = 0:1/Fs:L-1/Fs; % Time vector
192 f = -(Fs-1/L)/2:1/L:(Fs-1/L)/2;  % Frequency vector
193
194 fft1 = abs((2/Fs)*fft(P));
195 fftbeta = abs((2/Fs)*fft(yarr_croppedHPF));
196 figure;
197 subplot(2,1,1); plot(f,fftshift(fft1),'Linewidth',2);xlim([0 max
        (f)]);title('FFT of input signal');
198 xlabel('Frequency'); ylabel('Amplitude');
199 subplot(2,1,2);; plot(f,fftshift(fftbeta),'Linewidth',2);xlim([0
         max(f)]);title('FFT of filtered signal');
200 xlabel('Frequency'); ylim([0  0.5]);
201 %% Train the Hopf system to be a gamma filter
202 Dec = dataOutgamma;
203 f = -(Fs-1/L)/2:1/L:(Fs-1/L)/2;  % Frequency vector
204 fft1 = abs((2/Fs)*fft(Dec));
205 figure;
```

```
206 plot(f,abs(fftshift(fft1)),'Linewidth',2);
207 xlim([0  150]);
208
209 W = rand(N,1);
210 Winit = W;
211 etaa2 = 0.05;
212 I = I/10;
213 nepoch = 4000;
214 count=1;
215 for ii=1:nepoch
216     ii
217     for jj=1:length(t)
218         X = R.*cos(phi);
219         yBPF = W'*X; yarrBPF(count) = yBPF;
220         eBPF = Dec(jj)-yBPF;
221
222         Rdot = R.*(meu-R.^2) + epsl*I.*cos(phi);
223         phidot = omega - (epsl*(I.*sin(phi))./R);
224         omegadot = -epsl*I.*sin(phi);
225         Wdot = etaa2*eBPF*X;
226
227         R = R + Rdot*dt; %R = (R>0).*R;
228         phi = phi + phidot*dt; %phi = (phi>0).*phi;
229         omega = omega + omegadot*dt; %omega = (omega>0).*omega;
230         W = W + Wdot*dt;
231         count = count + 1;
232     end
233 end
234 % st = length(yarr)-200;
235 yarr_croppedBPF = yarrBPF(length(yarrBPF)-length(t)+1:length(
      yarrBPF));
236 figure; plot(t,yarr_croppedBPF);
237
```

```
L = length(t)*Ts;      % signal duration
t = 0:1/Fs:L-1/Fs; % Time vector
f = -(Fs-1/L)/2:1/L:(Fs-1/L)/2;  % Frequency vector

fft1 = abs((2/Fs)*fft(P));
fftgamma = abs((2/Fs)*fft(yarr_croppedBPF));
figure;
subplot(2,1,1); plot(f,fftshift(fft1),'Linewidth',2);xlim([0 max
    (f)]);title('FFT of input signal');
xlabel('Frequency'); ylabel('Amplitude');
subplot(2,1,2); plot(f,fftshift(fftgamma),'Linewidth',2);xlim([0
     max(f)]);title('FFT of filtered signal');
xlabel('Frequency'); ylim([0 0.5]);
%% Train the Hopf system to be a theta filter
Dec = dataOuttheta;
f = -(Fs-1/L)/2:1/L:(Fs-1/L)/2;  % Frequency vector
fft1 = abs((2/Fs)*fft(Dec));
figure;
plot(f,abs(fftshift(fft1)),'Linewidth',2);
xlim([0 150]);

W = rand(N,1);
Winit = W;
etaa2 = 0.05;
I = I/10;
nepoch = 4000;
count=1;
for ii=1:nepoch
    ii
    for jj=1:length(t)
        X = R.*cos(phi);
        ytheta = W'*X; yarrtheta(count) = ytheta;
        etheta = Dec(jj)-ytheta;
```

```
269
270             Rdot = R.*(meu-R.^2) + epsl*I.*cos(phi);
271             phidot = omega - (epsl*(I.*sin(phi))./R);
272             omegadot = -epsl*I.*sin(phi);
273             Wdot = etaa2*etheta*X;
274
275             R = R + Rdot*dt; %R = (R>0).*R;
276             phi = phi + phidot*dt; %phi = (phi>0).*phi;
277             omega = omega + omegadot*dt; %omega = (omega>0).*omega;
278             W = W + Wdot*dt;
279             count = count + 1;
280         end
281 end
282 % st = length(yarr)-200;
283 yarr_croppedtheta = yarrtheta(length(yarrtheta)-length(t)+1:
        length(yarrtheta));
284 figure; plot(t,yarr_croppedtheta);
285
286 L = length(t)*Ts;      % signal duration
287 t = 0:1/Fs:L-1/Fs; % Time vector
288 f = -(Fs-1/L)/2:1/L:(Fs-1/L)/2;  % Frequency vector
289
290 fft1 = abs((2/Fs)*fft(P));
291 ffttheta = abs((2/Fs)*fft(yarr_croppedtheta));
292 figure;
293 subplot(2,1,1); plot(f,fftshift(fft1),'Linewidth',2);xlim([0 max
        (f)]);title('FFT of input signal');
294 xlabel('Frequency'); ylabel('Amplitude');
295 subplot(2,1,2); plot(f,fftshift(ffttheta),'Linewidth',2);xlim([0
         max(f)]);title('FFT of filtered signal');
296 xlabel('Frequency'); ylim([0 1]);
297 %% Train the Hopf system to be a delta filter
298 Dec = dataOutdelta;
```

```
f = -(Fs-1/L)/2:1/L:(Fs-1/L)/2;  % Frequency vector
fft1 = abs((2/Fs)*fft(Dec));
figure;
plot(f,abs(fftshift(fft1)),'Linewidth',2);
xlim([0  150]);

W = rand(N,1);
Winit = W;
etaa2 = 0.05;
I = I/10;
nepoch = 4000;
count=1;
for ii=1:nepoch
    ii
    for jj=1:length(t)
        X = R.*cos(phi);
        ydelta = W'*X; yarrdelta(count) = ydelta;
        edelta = Dec(jj)-ydelta;

        Rdot = R.*(meu-R.^2) + epsl*I.*cos(phi);
        phidot = omega - (epsl*(I.*sin(phi))./R);
        omegadot = -epsl*I.*sin(phi);
        Wdot = etaa2*edelta*X;

        R = R + Rdot*dt; %R = (R>0).*R;
        phi = phi + phidot*dt; %phi = (phi>0).*phi;
        omega = omega + omegadot*dt; %omega = (omega>0).*omega;
        W = W + Wdot*dt;
        count = count + 1;
    end
end
% st = length(yarr)-200;
```

```
yarr_croppeddelta = yarrdelta(length(yarrdelta)-length(t)+1:
    length(yarrdelta));
figure; plot(t,yarr_croppeddelta);

L = length(t)*Ts;      % signal duration
t = 0:1/Fs:L-1/Fs; % Time vector
f = -(Fs-1/L)/2:1/L:(Fs-1/L)/2;  % Frequency vector

fft1 = abs((2/Fs)*fft(P));
fftdelta = abs((2/Fs)*fft(yarr_croppeddelta));
figure;
subplot(2,1,1); plot(f,fftshift(fft1),'Linewidth',2);xlim([0 max
    (f)]);title('FFT of input signal');
xlabel('Frequency'); ylabel('Amplitude');
subplot(2,1,2); plot(f,fftshift(fftdelta),'Linewidth',2);xlim([0
     max(f)]);title('FFT of filtered signal');
xlabel('Frequency'); ylim([0 10]);
%% Plot
figure;
subplot(6,1,1);plot(f,fftshift(fft1),'Linewidth',2);xlim([0 max(
    f)]);title('FFT of input signal');
xlabel('Frequency'); ylabel('Amplitude');
subplot(6,1,2);plot(f,fftshift(fftalpha),'Linewidth',2);xlim([0
    max(f)]);title('FFT of alpha signal');
xlabel('Frequency'); ylabel('Amplitude');
subplot(6,1,3);plot(f,fftshift(fftbeta),'Linewidth',2);xlim([0
    max(f)]);title('FFT of beta signal');
xlabel('Frequency'); ylabel('Amplitude');
subplot(6,1,4);plot(f,fftshift(fftgamma),'Linewidth',2);xlim([0
    max(f)]);title('FFT of gamma signal');
xlabel('Frequency'); ylabel('Amplitude');
subplot(6,1,5);plot(f,fftshift(fftdelta),'Linewidth',2);xlim([0
    max(f)]);title('FFT of delta signal');
```

```
356 xlabel('Frequency'); ylabel('Amplitude');
357 subplot(6,1,6);plot(f,fftshift(ffttheta),'Linewidth',2);xlim([0
        max(f)]);title('FFT of theta signal');
358 xlabel('Frequency'); ylabel('Amplitude');
```

APPENDIX B

Code for testing

```
1
2 %% Test the Hopf system for alpha filter
3 Fs = 250;
4 Ts = 1/Fs;
5 L=2;
6 t = 0:Ts:L;
7 %f = [5 25]; testsig=zeros(1,length(t));
8 %for ii=1:length(f)
9 %     testsig = testsig+sin(2*pi*f(ii)*t);
10 %end
11 testsig = transpose(data{1,1}.X(1:501,2));
12 figure; plot(t,testsig,'Linewidth',2); title('Input signal');
       xlabel('Time in seconds');
13 %HOPF
14 N = 200;
15 fmax = 250; fmin = 1;
16 epsl = 5;
17 meu = 1;
18 dt = Ts;
19
20 for jj=1:length(t)
21           X = R.*cos(phi);
22           yLPF = W'*X; yarrLPF(count) = yLPF;
23           I = testsig(jj) - Phat;
24
25           Rdot = R.*(meu-R.^2) + epsl*I.*cos(phi);
```

```
        phidot = omega - (epsl*(I.*sin(phi))./R);
        omegadot = -epsl*I.*sin(phi);


        R = R + Rdot*dt; %R = (R>0).*R;
        phi = phi + phidot*dt; %phi = (phi>0).*phi;
        omega = omega + omegadot*dt; %omega = (omega>0).*omega;

        count = count + 1;
end
% Plot LPF
yarr_croppedLPF = yarrLPF(length(yarrLPF)-length(t)+1:length(
    yarrLPF));
figure; plot(t,yarr_croppedLPF);

L = length(t)*Ts;      % signal duration
t = 0:1/Fs:L-1/Fs; % Time vector
f = -(Fs-1/L)/2:1/L:(Fs-1/L)/2;  % Frequency vector

fft1 = abs((2/Fs)*fft(testsig));
fftalpha = abs((2/Fs)*fft(yarr_croppedLPF));
figure; subplot(2,2,1); plot(testsig); title('Input signal')
subplot(2,2,2);plot(f,fftshift(fft1),'Linewidth',2);xlim([0 max(
    f)]);title('FFT of input signal');
xlabel('Frequency'); ylabel('Amplitude');
subplot(2,2,3); plot(yarr_croppedLPF); title('Filtered signal')
subplot(2,2,4);plot(f,fftshift(fftalpha),'Linewidth',2);xlim([0
    max(f)]);title('FFT of filtered signal');
xlabel('Frequency'); %ylim([0 0.5])

%% Test the Hopf system for beta filter
Fs = 250;
Ts = 1/Fs;
```

```
L=2;
t = 0:Ts:L;
%f = [5 25]; testsig=zeros(1,length(t));
%for ii=1:length(f)
%    testsig = testsig+sin(2*pi*f(ii)*t);
%end
testsig = transpose(data{1,1}.X(1:501,2));

figure; plot(t,testsig,'Linewidth',2); title('Input signal');
    xlabel('Time in seconds');
%HOPF
N = 200;
fmax = 50; fmin = 1;
epsl = 5;
meu = 1;
dt = Ts;

for jj=1:length(t)
        X = R.*cos(phi);
        yHPF = W'*X; yarrHPF(count) = yHPF;
        I = testsig(jj) - Phat;

        Rdot = R.*(meu-R.^2) + epsl*I.*cos(phi);
        phidot = omega - (epsl*(I.*sin(phi))./R);
        omegadot = -epsl*I.*sin(phi);


        R = R + Rdot*dt; %R = (R>0).*R;
        phi = phi + phidot*dt; %phi = (phi>0).*phi;
        omega = omega + omegadot*dt; %omega = (omega>0).*omega;

        count = count + 1;
end
```

```
88 % Plot LPF
89 yarr_croppedHPF = yarrHPF(length(yarrHPF)-length(t)+1:length(
       yarrHPF));
90 figure; plot(t,yarr_croppedHPF);
91
92 L = length(t)*Ts;      % signal duration
93 t = 0:1/Fs:L-1/Fs; % Time vector
94 f = -(Fs-1/L)/2:1/L:(Fs-1/L)/2;  % Frequency vector
95
96 fft1 = abs((2/Fs)*fft(testsig));
97 fftbeta = abs((2/Fs)*fft(yarr_croppedHPF));
98 figure; subplot(2,2,1); plot(testsig); title('Input signal')
99 subplot(2,2,2);plot(f,fftshift(fft1),'Linewidth',2);xlim([0 max(
       f)]);title('FFT of input signal');
100 xlabel('Frequency'); ylabel('Amplitude');
101 subplot(2,2,3); plot(yarr_croppedHPF); title('Filtered signal')
102 subplot(2,2,4);plot(f,fftshift(fftbeta),'Linewidth',2);xlim([0
       max(f)]);title('FFT of filtered signal');
103 xlabel('Frequency'); %ylim([0 0.5])
104
105 %% Test the Hopf system for gamma filter
106
107 Fs = 250;
108 Ts = 1/Fs;
109 L=2;
110 t = 0:Ts:L;
111 %f = [5 25]; testsig=zeros(1,length(t));
112 %for ii=1:length(f)
113 %     testsig = testsig+sin(2*pi*f(ii)*t);
114 %end
115 testsig = transpose(data{1,1}.X(1:501,2));
116
117 figure; plot(t,testsig,'Linewidth',2); title('Input signal');
```

```
    xlabel('Time in seconds');
%HOPF
N = 200;
fmax = 50; fmin = 1;
epsl = 5;
meu = 1;
dt = Ts;

for jj=1:length(t)
        X = R.*cos(phi);
        yBPF = W'*X; yarrBPF(count) = yBPF;
        I = testsig(jj) - Phat;

        Rdot = R.*(meu-R.^2) + epsl*I.*cos(phi);
        phidot = omega - (epsl*(I.*sin(phi))./R);
        omegadot = -epsl*I.*sin(phi);


        R = R + Rdot*dt; %R = (R>0).*R;
        phi = phi + phidot*dt; %phi = (phi>0).*phi;
        omega = omega + omegadot*dt; %omega = (omega>0).*omega;

        count = count + 1;
end
% Plot LPF
yarr_croppedBPF = yarrBPF(length(yarrBPF)-length(t)+1:length(
    yarrBPF));
figure; plot(t,yarr_croppedBPF);

L = length(t)*Ts;      % signal duration
t = 0:1/Fs:L-1/Fs; % Time vector
f = -(Fs-1/L)/2:1/L:(Fs-1/L)/2;  % Frequency vector

```

```
149 fft1 = abs((2/Fs)*fft(testsig));
150 fftgamma = abs((2/Fs)*fft(yarr_croppedBPF));
151 figure; subplot(2,2,1); plot(testsig); title('Input signal')
152 subplot(2,2,2);plot(f,fftshift(fft1),'Linewidth',2);xlim([0 max(
        f)]);title('FFT of input signal');
153 xlabel('Frequency'); ylabel('Amplitude');
154 subplot(2,2,3); plot(yarr_croppedBPF); title('Filtered signal')
155 subplot(2,2,4);plot(f,fftshift(fftgamma),'Linewidth',2);xlim([0
        max(f)]);title('FFT of filtered signal');
156 xlabel('Frequency'); %ylim([0 0.5])
157 %% Test the Hopf system for theta filter
158
159 Fs = 250;
160 Ts = 1/Fs;
161 L=2;
162 t = 0:Ts:L;
163 %f = [5 25]; testsig=zeros(1,length(t));
164 %for ii=1:length(f)
165 %     testsig = testsig+sin(2*pi*f(ii)*t);
166 %end
167 testsig = transpose(data{1,1}.X(1:501,2));
168
169 figure; plot(t,testsig,'Linewidth',2); title('Input signal');
        xlabel('Time in seconds');
170 %HOPF
171 N = 200;
172 fmax = 50; fmin = 1;
173 epsl = 5;
174 meu = 1;
175 dt = Ts;
176
177 for jj=1:length(t)
178         X = R.*cos(phi);
```

```
179         ytheta = W'*X; yarrtheta(count) = ytheta;
180         I = testsig(jj) - Phat;
181
182         Rdot = R.*(meu-R.^2) + epsl*I.*cos(phi);
183         phidot = omega - (epsl*(I.*sin(phi))./R);
184         omegadot = -epsl*I.*sin(phi);
185
186
187         R = R + Rdot*dt; %R = (R>0).*R;
188         phi = phi + phidot*dt; %phi = (phi>0).*phi;
189         omega = omega + omegadot*dt; %omega = (omega>0).*omega;
190
191         count = count + 1;
192 end
193 % Plot LPF
194 yarr_croppedtheta = yarrtheta(length(yarrtheta)-length(t)+1:
        length(yarrtheta));
195 figure; plot(t,yarr_croppedtheta);
196
197 L = length(t)*Ts;      % signal duration
198 t = 0:1/Fs:L-1/Fs; % Time vector
199 f = -(Fs-1/L)/2:1/L:(Fs-1/L)/2;  % Frequency vector
200
201 fft1 = abs((2/Fs)*fft(testsig));
202 ffttheta = abs((2/Fs)*fft(yarr_croppedtheta));
203 figure; subplot(2,2,1); plot(testsig); title('Input signal')
204 subplot(2,2,2);plot(f,fftshift(fft1),'Linewidth',2);xlim([0 max(
        f)]);title('FFT of input signal');
205 xlabel('Frequency'); ylabel('Amplitude');
206 subplot(2,2,3); plot(yarr_croppedtheta); title('Filtered signal'
        )
207 subplot(2,2,4);plot(f,fftshift(ffttheta),'Linewidth',2);xlim([0
        max(f)]);title('FFT of filtered signal');
```

```
208 xlabel('Frequency'); %ylim([0 0.5])
209 %% Test the Hopf system for delta filter
210
211 Fs = 250;
212 Ts = 1/Fs;
213 L=2;
214 t = 0:Ts:L;
215 %f = [5 25]; testsig=zeros(1,length(t));
216 %for ii=1:length(f)
217 %     testsig = testsig+sin(2*pi*f(ii)*t);
218 %end
219 testsig = transpose(data{1,1}.X(1:501,2));
220
221 figure; plot(t,testsig,'Linewidth',2); title('Input signal');
      xlabel('Time in seconds');
222 %HOPF
223 N = 200;
224 fmax = 50; fmin = 1;
225 epsl = 5;
226 meu = 1;
227 dt = Ts;
228
229 for jj=1:length(t)
230         X = R.*cos(phi);
231         yBPF = W'*X; yarrBPF(count) = yBPF;
232         I = testsig(jj) - Phat;
233
234         Rdot = R.*(meu-R.^2) + epsl*I.*cos(phi);
235         phidot = omega - (epsl*(I.*sin(phi))./R);
236         omegadot = -epsl*I.*sin(phi);
237
238
239         R = R + Rdot*dt; %R = (R>0).*R;
```

```
240         phi = phi + phidot*dt; %phi = (phi>0).*phi;
241         omega = omega + omegadot*dt; %omega = (omega>0).*omega;
242
243         count = count + 1;
244 end
245 % Plot LPF
246 yarr_croppedBPF = yarrBPF(length(yarrBPF)-length(t)+1:length(
        yarrBPF));
247 figure; plot(t,yarr_croppedBPF);
248
249 L = length(t)*Ts;      % signal duration
250 t = 0:1/Fs:L-1/Fs; % Time vector
251 f = -(Fs-1/L)/2:1/L:(Fs-1/L)/2;  % Frequency vector
252
253 fft1 = abs((2/Fs)*fft(testsig));
254 fftdelta = abs((2/Fs)*fft(yarr_croppedBPF));
255 figure; subplot(2,2,1); plot(testsig); title('Input signal')
256 subplot(2,2,2);plot(f,fftshift(fft1),'Linewidth',2);xlim([0 max(
        f)]);title('FFT of input signal');
257 xlabel('Frequency'); ylabel('Amplitude');
258 subplot(2,2,3); plot(yarr_croppedBPF); title('Filtered signal')
259 subplot(2,2,4);plot(f,fftshift(fftdelta),'Linewidth',2);xlim([0
        max(f)]);title('FFT of filtered signal');
260 xlabel('Frequency'); %ylim([0 0.5])
261
262 %% Plot
263 figure;
264 subplot(6,1,1);plot(f,fftshift(fft1),'Linewidth',2);xlim([0 max(
        f)]);title('FFT of input signal');
265 xlabel('Frequency'); ylabel('Amplitude');
266 subplot(6,1,2);plot(f,fftshift(fftalpha),'Linewidth',2);xlim([0
        max(f)]);title('FFT of alpha signal');
267 xlabel('Frequency'); ylabel('Amplitude');
```

```
subplot(6,1,3);plot(f,fftshift(fftbeta),'Linewidth',2);xlim([0
    max(f)]);title('FFT of beta signal');
xlabel('Frequency'); ylabel('Amplitude');
subplot(6,1,4);plot(f,fftshift(fftgamma),'Linewidth',2);xlim([0
    max(f)]);title('FFT of gamma signal');
xlabel('Frequency'); ylabel('Amplitude');
subplot(6,1,5);plot(f,fftshift(fftdelta),'Linewidth',2);xlim([0
    max(f)]);title('FFT of delta signal');
xlabel('Frequency'); ylabel('Amplitude');
subplot(6,1,6);plot(f,fftshift(ffttheta),'Linewidth',2);xlim([0
    max(f)]);title('FFT of theta signal');
xlabel('Frequency'); ylabel('Amplitude');
```

APPENDIX C

Code for butterworth filter

```
%% Butterworth Filter alpha
fc = [8 12]; %cut-off freq

%[b,a] = butter(3,fc/(Fs/2));
[A,B,C,D] = butter(10,fc/(Fs/2));
%freqz(b,a)
d = designfilt('bandpassiir','FilterOrder',20, ...
    'HalfPowerFrequency1',8,'HalfPowerFrequency2',12, ...
    'SampleRate',250);
%fvtool(d)
dataOutalpha = filter(d,P);
fftFilter = abs((2/Fs)*fft(dataOutalpha));
figure;
plot(f,abs(fftshift(fftFilter)),'Linewidth',2);
xlim([0 40]);
%% Butterworth Filter beta
fc = [12 40]; %cut-off freq

%[b,a] = butter(3,fc/(Fs/2));
[A,B,C,D] = butter(10,fc/(Fs/2));
%freqz(b,a)
d = designfilt('bandpassiir','FilterOrder',20, ...
    'HalfPowerFrequency1',12,'HalfPowerFrequency2',40, ...
    'SampleRate',250);
%fvtool(d)
dataOutbeta = filter(d,P);
```

```
fftFilter = abs((2/Fs)*fft(dataOutbeta));
figure;
plot(f,abs(fftshift(fftFilter)),'Linewidth',2);
xlim([0 Fs/2]);
%% Butterworth Filter gamma
fc = [40 100]; %cut-off freq

%[b,a] = butter(3,fc/(Fs/2));
[A,B,C,D] = butter(10,fc/(Fs/2));
%freqz(b,a)
d = designfilt('bandpassiir','FilterOrder',20, ...
    'HalfPowerFrequency1',40,'HalfPowerFrequency2',100, ...
    'SampleRate',250);
%fvtool(d)
dataOutgamma = filter(d,P);
fftFilter = abs((2/Fs)*fft(dataOutgamma));
figure;
plot(f,abs(fftshift(fftFilter)),'Linewidth',2);
xlim([0 Fs/2]);
%% Butterworth Filter theta
fc = [4 8]; %cut-off freq

%[b,a] = butter(3,fc/(Fs/2));
[A,B,C,D] = butter(10,fc/(Fs/2));
%freqz(b,a)
d = designfilt('bandpassiir','FilterOrder',20, ...
    'HalfPowerFrequency1',4,'HalfPowerFrequency2',8, ...
    'SampleRate',250);
%fvtool(d)
dataOuttheta = filter(d,P);
fftFilter = abs((2/Fs)*fft(dataOuttheta));
figure;
plot(f,abs(fftshift(fftFilter)),'Linewidth',2);
```

```
60 xlim([0 Fs/4]);
61 %% Butterworth Filter delta
62 fc = 4; %cut-off freq
63
64 %[b,a] = butter(3,fc/(Fs/2));x
65 [b,a] = butter(10,fc/(Fs/2));
66
67 dataOutdelta = filter(b,a,P);
68 fftFilter = abs((2/Fs)*fft(dataOutdelta));
69 figure;
70 plot(f,abs(fftshift(fftFilter)),'Linewidth',2);
71 xlim([0 Fs/8]);
```

REFERENCES

1. **K. SURESH MANIC, K. P. A. C., AMINATH SAADHA**, Characterisation and separation of brainwave signals. *In Journal of Engineering Science and Technology EURECA 2014 Special Issue*. EURECA, 2014.

2. **Ludovic Righetti, J. B.** and **A. J. Ijspeert**, From dynamic hebbian learning for oscillators to adaptive central pattern generators. *In Proceedings of 3rd International Symposium on Adaptive Motion in Animals and Machines AMAM 2005*. Verlag ISLE, Ilmenau, 2005.

3. **R. Leeb, K. C. P. G., Lee F**, Brain-computer communication: motivation, aim, and impact of exploring a virtual apartment. *In IEEE Transactions on Neural Systems and Rehabilitation Engineering 15, 2007*. IEEE, 2007, 473âĂŞ482.

4. **Shakshi, R. J.**, Brain wave classification and feature extraction of eeg signal by using fft on lab view. *In International Research Journal of Engineering and Technology, Volume 3, Issue 07*. IRJET, 2016.