MAJOR PROJECT ON

CALCULATION OF PSDF USING SIXTH ORDER ADAPTIVE FILTER

Submitted in partial fulfilment of the requirements
of the degree of
Master of Technology in Structural Engineering

By

Shreyas Gune 2K12/STR/19

Supervisor:
Shri G.P. Awadhiya
Assistant Professor in Civil Engineering

Department of Civil Engineering
DELHI TECHNOLOGICAL UNIVERSITY
DELHI

January 2016

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be the cause for disciplinary action by the University and can also evoke penal action from the sources which have not been properly cited or proper permission has not been taken when needed.

Shreyas Gune 2K12/STR/19 _____

Date:_____

Department of Civil Engineering

Delhi Technological University

CERTIFICATE

This is to certify that the project work entitled ***"Calculation of PSDF using sixth order adaptive filter"*** is being submitted by me, candidate of the M.Tech**. Structural Engineering** is a bonafide record of my work carried out by me, under guidance and direction of **Mr. G. P. Awadhiya**.

I,Shreyas Gune have not submitted the matter embodied in this report to any University or Institution for award of any Degree or Diploma

Shreyas Gune
Roll No. : 2k12/STR/19

This is to certify that the above statement made by him is correct to the best of my knowledge.

Mr. G.P. Awadhiya.
Department of Civil Engineering
Delhi Technological University

Acknowledgement

I express my sincere thanks and deep sense of gratitude to my mentor**, Mr. G.P Awadhiya**, Department of Civil Engineering, Delhi Technological University, whose contribution in stimulating suggestions and encouragement, helped me throughout my project. Without his valuable motivation and guidance this study would not have been possible.

I wish to convey my sincere gratitude to **Prof. N. Dev**, HOD and all faculties of the Civil Engineering Department, Delhi Technological University. I also consider myself fortunate for having the opportunity to learn and work with all class mates of structural engineering over the entire period of association.

Shreyas Gune

(2K12/STR/19)

Abstract

In this thesis, ground motion has been modeled as IIR sixth order adaptive filter. Three IIR filter models have been compared for various earthquakes. White noise input is given to the filter and the filter coefficients, initialized as zero are adapted such that the output is equal to the desired earthquake. Further, three adaptive algorithms, namely, standard least mean square (LMS) algorithm, normalized least mean square(NLMS) algorithm and LMS algorithm with different coefficients for forward and feedback loop have been used in this research.
Acceleration time history graphs for different earthquakes alongside the generated acceleration time history have been plotted for different algorithms. Power spectral density Function vs Frequency plot for different earthquakes with different algorithms have also been worked out.

It has been identified that NLMS algorithm gives the best result for Power Spectral Density Function.

Contents

List of figures

List of tables

List of symbols

Symbol	Description
σ	Standard deviation
m	Mass
c	Viscous damping coefficient
k	Stiffness
x	Absolute displacement of the mass
y	Base input displacement
ω_n	natural frequency
ξ	Damping ratio
ω_d	Damped frequency
$\hat{h}_a(t)$	Acceleration Impulse Response
x[n]	Input signal
y[n]	Output signal
e[n]	Error signal
d[n]	Desired signal
μ(n)	Step size parameter

Chapter-1

Introduction

In recent, digital filtering techniques for modelling of earthquake ground motion have gained immense significance. Using principles of adaptive filtering ground motion can be modelled. The filter coefficients thus obtained can be used in the estimation of Power Spectral Density Function which is a measure of power content associated with a particular frequency in an earthquake (Stoica and Moses, 2005) and can be used to derive dynamic properties of ground (Conte, Pister & Mahin, 1992) .It is used in non linear dynamic analysis of structures.

Adaptive filters have been studied to find the best adaptive algorithm giving output with the best match of the desired earthquake.

1.1 Sampling of Earthquake Signals

Seismograms are a record of the ground motion at a specific location. Seismograms come in many forms, on "smoked" paper, photographic paper, common ink recordings on standard paper, and in digital format.

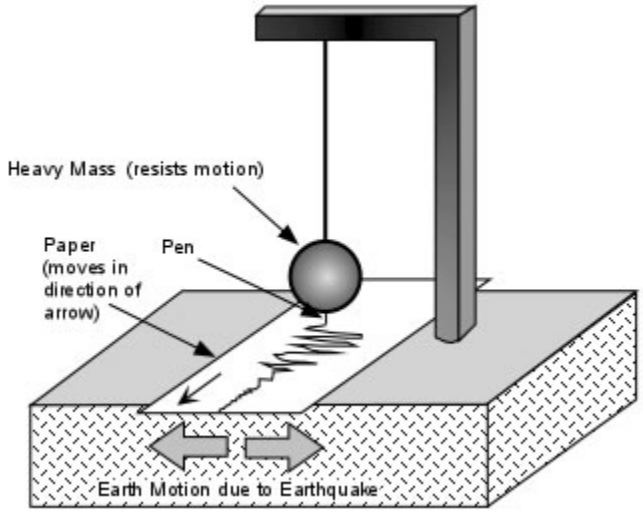


Fig 1: Seismogram

Seismograms were earlier recorded on sheet of paper, either with ink or photographically. Such records are called "analog" records. These are sampled continuously.

Most of the seismic data, today, are recorded digitally, which facilitates quick interpretations of the signals using computers. Digital seismograms are "sampled" at an even time interval that depends on the type of seismic instrument and our purpose.

Another important class of seismometers for recording large amplitude vibrations are called *strong-motion* seismometers. Strong-motion instruments are designed to record the high accelerations that are particularly important for designing structures. In this project earthquake data was recorded at sampling interval (Δt) is 0.02 sec.

1.2 Earthquake signals

a. Discrete time Earthquake Signals

A discrete signal or discrete-time signal is a time series consisting of a sequence of quantities (acceleration, velocity, displacement etc.). In other words, it is a time series that is a function over a domain of integers. In this project digital earthquake acceleration time history has been used.

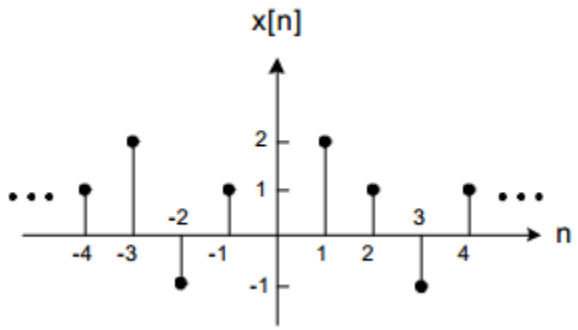


Fig 2: Discrete Time Signals

b. Continous time Earthquake Signals

A continuous signal or a continuous-time earthquake signal is a varying quantity (a signal) whose domain is a continuum. Use of continuously sampled signals in earthquake engineering is almost obsolete.

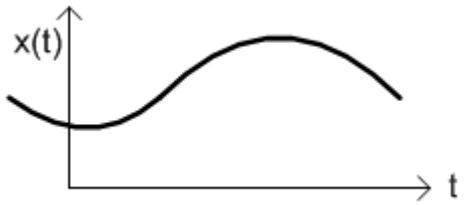


Fig 3: Continuous Time Signals

1.3 Earthquake signal modelling

Earthquake Signals are usually modelled as filtered white noise process. The filter can be IIR (Infinite impulse response) or FIR (Finite impulse response). IIR filters have definite number of filter coefficients and are less complex, and thus much easier to realize.

Adaptive filters are also used in earthquake signal modelling. An adaptive filter can be FIR or IIR. The main difference between a time invariant filter and an adaptive filter is that its coefficients are made to change with in accordance with an adaptive algorithm. Input signal is white noise process. The error signal found out by comparing the adaptive filter output with the desired signal (i.e. earthquake acceleration time history) is used to update the coefficients till they stabilize according to the adaptive algorithm used.

1.3.1 Adaptive algorithms

In this project three algorithms have been used, namely:

1. Standard LMS algorithm,
2. LMS algorithm with different step sizes for forward and feedback loop and,
3. NLMS algorithm

These algorithms minimize the cost function which is a function of error signal. These algorithms have been compared and it has been found out that NLMS algorithm gives the best estimate of the Power Spectral Density Function.

1.4 Scope

This thesis deals with the estimation of Power Spectral Density Function using adaptive filtering techniques by comparing various algorithms and finding the algorithm that gives the best estimate among them. Various earthquakes have been studied and acceleration time history graphs for different earthquakes alongside the generated acceleration time history have been plotted for different algorithms. Power spectral density Function vs Frequency plot for different earthquakes with different algorithms have also been worked out.

Chapter-2

Literature Review

Jiande Chen et al.(1987) [Ref. 1]

This paper gives the adaptive algorithm for Adaptive filter with different step sizes for forward and feedback loop. The coefficients of the filter correspond to parameters of ARMA model.

It also mentions that the error signal from the process can be used as the input white noise in the calculation of Power spectral density. This concept improved the convergence of all the algorithms used in this project drastically.

Once the algorithm converges, PSDF can be plotted using the relation:

$$P_y(\omega) = \frac{\sigma^2 \triangle t |1 + \sum_{k=1}^{q} c_k exp(-j\omega k \triangle t)|^2}{|1 + \sum_{k=1}^{p} a_k exp(-j\omega k \triangle t)|^2}$$

This relation has been used in the project.

J.P. Conte et al.(1996) [Ref. 2]

This paper gives various ARMA models and the procedure for finding the kanaitajimi parameters. It also gives the procedure for finding the PSDF and plots PSDF vs Frequency for different ARMA models.

In this paper the non-stationary behaviour of earthquake acceleration time history is also included in the model formulation. Concepts of ARMA models and PSDF used in this project have been developed from this paper.

Simon O'lafsson et al.(1992) [Ref. 3]

This paper gives various ARMA models for ground motion modelling. It uses envelope function and Gaussian white noise process and gives the process for model validation. It mentions that it is desirable to keep the number of parameters of ARMA process small making it easier to establish link with the physical parameters of the earthquake.

TH. D. Popescu and S. Demetriu(1990) [Ref. 4]

In this paper a general procedure for the analysis ans simulation of strong earthquake ground motions based on ARMA models. Power Spectral Density Function are plotted at different time instances for three different earthquakes.

Feng Chun Wang et al.(2013) [Ref. 5]

This paper gives a variable step size adaptive filter. It shows the relationship between step size and convergence rate. The adaptive algorithm mentioned in this paper is shown to perform better than the standard fixed step size filter. The NLMS filter used in this project is based on the variable step size concept of this paper.

Tom Irvine(2012) [Ref. 6]

This paper describes the derivation of digital recursive relation for a single degree of freedom structure and gives the derivation for impulse response function for the the single degree of freedom system. The concept of representation of ground motion in terms of a single degree of freedom system and the concept of its transfer function was taken from this paper.

Tom Irvine(2000) [Ref. 7]

This paper describes the filtering process for digital signals. It gives the basics of digital time invariant filter design. The basics concepts of filter design used in this project have been developed from this paper.

Simon Haykin, Adaptive filters (5^{th} Edition) [Ref. 8]

This book gives the basic adaptive filter theory and its application in Power Spectral density Function calculation. It also gives various algorithms used in adaptive signal processing and specifically the system identification procedure followed in this project.

The definition of Power spectral density is given as below:

$$P_y(\omega) = \sigma_u^2 |H(\omega)|^2,$$

It also gives the parametric method of PSDF calculation followed in this project.

Chapter-3

Mathematical Modelling Of Earthquake Signals

3.1 Introduction

Seismic Signals are usually modelled as filtered white noise process. Filter can be any operation that modifies a signal. It filters the shape of frequency spectrum of an Earthquake signal. Filters generally do not add frequency components that are not already there. They merely amplify or diminish certain regions of frequency spectrum.

Three types of difference equations are usually used for implementing filters in Earthquake modeling:

- Auto regressive (AR)
- Moving average (MA)
- Auto regressive and moving average (ARMA)

a. Auto regressive difference equations (AR):

AR difference equations described by the following equation:

$y(t) = a_1y(t - 1) + a_2y(t-2)$

In auto regressive (AR) difference equations, the output signal depends only on past outputs.

b. Moving average difference equations (MA):

MA difference equations described by the following equation:

$y(t) = b_0x(t) + b_1x(t - 1) + b_2x(t-2)$

In moving average (MA) difference equations, the output is a function of present and past inputs only.

c. Auto regressive moving average difference equations (MA):

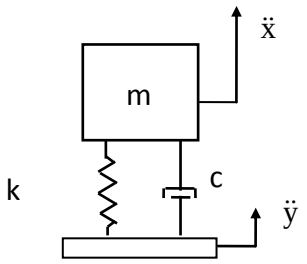
ARMA difference equations described by the following equation:

$y(t) = b_0x(t) + b_1x(t - 1) + b_2x(t-2)+ a_1y(t - 1) + a_2y(t-2)$

In Auto regressive moving average (ARMA) difference equations, the output signal is a function of present and past inputs and past outputs.

3.2 Mechanical System as Digital Filters (Tom Irvine,2012):

Consider this single-degree-of-freedom system as shown in Figure below:



m	=	Mass
c	=	Viscous damping coefficient
k	=	Stiffness
x	=	Absolute displacement of the mass
y	=	Base input displacement

Fig 4: Single Degree of freedom mechanical system

The free-body diagram the above mechanical system is shown in Figure.

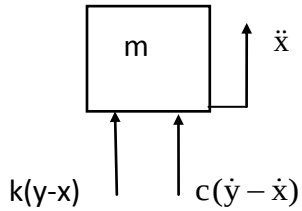


Fig 5: Free Body Diagram of Single Degree of freedom mechanical system

Balancing of forces in the vertical direction gives:

$$\sum F = m\ddot{x}$$

$$m\ddot{x} = c(\dot{y} - \dot{x}) + k(y - x)$$

Let,

$$u = x - y$$

$$\dot{u} = \dot{x} - \dot{y}$$

$$\ddot{u} = \ddot{x} - \ddot{y}$$

$$\ddot{x} = \ddot{u} + \ddot{y}$$

The relative displacement terms mentioned above are substituted into the above equilibrium equations giving:

$$m(\ddot{u} + \ddot{y}) = -c\dot{u} - ku$$

$$m\ddot{u} + c\dot{u} + ku = -m\ddot{y}$$

Dividing both sides by mass gives:

$$\ddot{u} + (c/m)\dot{u} + (k/m)u = -\ddot{y}$$

By convention,

$$(c/m) = 2\xi\omega_n$$
$$(k/m) = \omega_n{}^2$$

where, ω_n is the natural frequency of the Single Degree of freedom mechanical system in (radians/sec), and ξ is the damping ratio.

Substitute the conventional identities into equation above gives:

$$\ddot{u} + 2\xi\omega_n\dot{u} + \omega_n{}^2 u = -\ddot{y}$$

3.2.1 Acceleration impulse response

The impulse response function for the acceleration response is:

$$\hat{h}_a(t) = \exp(-\xi\omega_n t)\left[2\xi\omega_n \cos(\omega_d t) + \frac{\omega_n{}^2}{\omega_d}(1-2\xi^2)\sin(\omega_d t)\right]$$

Its derivation is available in Tom Irvine(2012). The corresponding Laplace transform for $H_a(s)$ (response/input) is

$$H_a(s) = \left[\frac{2\xi\omega_n s + \omega_n{}^2}{s^2 + 2\xi\omega_n s + \omega_n{}^2}\right]$$

The corresponding Laplace transform for $H_i(s)$ (input/response) is

$$H_i(s) = \frac{s^2 + 2\xi\omega_n s + \omega_n{}^2}{2\xi\omega_n s + \omega_n{}^2}$$

The Z-transform is found using the bilinear transform,

$$s = \frac{2}{T}\frac{z-1}{z+1}$$

$$H_i(z) = \frac{\left[\frac{2}{T}\frac{(z-1)}{(z+1)}\right]^2 + 2\xi\omega_n\left[\frac{2}{T}\frac{(z-1)}{(z+1)}\right] + \omega_n{}^2}{2\xi\omega_n\left[\frac{2}{T}\frac{(z-1)}{(z+1)}\right] + \omega_n{}^2}$$

$H_i(z) =$

$$\frac{1}{T\omega_n(4\xi + T\omega_n)} \frac{(4 + 4\xi\omega_n T + T^2\omega_n{}^2)z^2 + 2(-4 + T^2\omega_n{}^2)z + \left(4 - 4\xi\omega_n T + T^2\omega_n{}^2\right)}{z^2 + \left(\frac{2T\omega_n}{4\xi + T\omega_n}\right)z + \left(\frac{-4\xi + T\omega_n}{4\xi + T\omega_n}\right)}$$

Which is of form,

$$H_i(z) = \frac{c_0 z^2 + c_1 z + c_2}{z^2 + a_1 z + a_2}$$

Digital recursive filtering relationship corresponding to the above form of Impulse response is :

$$\ddot{y}_i = -a_1 \ddot{y}_{i-1} - a_2 \ddot{y}_{i-2} + c_0 \ddot{x}_i + c_1 \ddot{x}_{i-1} + c_2 \ddot{x}_{i-2}$$

This equation can be considered as a difference equation of the form :

$$y(m) = \sum_{k=1}^{N} a_k y(m-k) + \sum_{k=0}^{M} b_k x(m-k)$$

3.3 IIR and FIR Filters

In the time domain the input-output relationship of a filter which is also known as recursive relation is given by the following linear difference equation:

$$y(m) = \sum_{k=1}^{N} a_k y(m-k) + \sum_{k=0}^{M} b_k x(m-k)$$

where $\{a_k, b_k\}$ are the filter coefficients, and the output y(m) is a linear combination of the previous N output samples [y(m−1),…, y(m−N)], the present input sample x(m) and the previous M input samples [x(m−1),…, x(m−M)]. The characteristic of a filter is completely determined by its coefficients $\{a_k, b_k\}$. For a time-invariant filter the coefficients $\{a_k, b_k\}$ are constants calculated to obtain a specified frequency response.

The filter transfer function, obtained by taking the z-transform of the difference equation:

$$H(z)=\frac{\sum_{k=0}^{M} b_k z^{-k}}{1-\sum_{k=1}^{N} a_k z^{-k}}$$

The frequency response of this filter can be obtained from the previous equation by substituting the frequency variable $e^{j\omega}$ for the z variable, $z = e^{j\omega}$

$$H(e^{j\omega})=\frac{\sum_{k=0}^{M} b_k e^{-j\omega k}}{1-\sum_{k=1}^{N} a_k e^{-j\omega k}}$$

If $a_1, a_2, a_3, \ldots$ are zero, this implies the filter is a **Finite impulse response filter (FIR)**.However, if any one of $a_1, a_2, a_3, \ldots$ are not zero, the filter is an **Infinite impulse response filter (IIR)**. IIR filter has the advantage of finite filter coefficients that can be evaluated. The main disadvantage of IIR filter is that it has stability issues that need to be taken care of.

3.4 Description of filters

Filters can be described using the following time or frequency domain methods:

3.4.1 Time domain input-output relationship

A difference equation is used to describe the output of a discrete-time filter in terms of a weighted combination of the input and previous output samples.

For example a first-order filter may have the following difference equation

$y(m) = a\, y(m-1) + x(m)$

where $x(m)$ is the filter input, $y(m)$ is the filter output and a is the filter coefficient.

3.4.2 Impulse Response

A filter can be described in terms of its response to an impulse input.

Impulse response is useful because: (i) any signal can be described as the sum of a number of shifted and scaled impulses, hence the response a linear filter to a signal is the sum of the responses to all the impulses that constitute the signal, (ii) an impulse input contains all

frequencies with equal energy, and hence it excites a filter at all frequencies and (iii) impulse response and frequency response are Fourier transform pairs.

3.4.3 Transfer Function, Poles and Zeros

The transfer function of a digital filter $H(z)$ is the ratio of the z-transforms of the filter output and input given by

$$H(z)=Y(z)/X(z)$$

A useful method of gaining insight into the behavior of a filter is the pole zero description of a filter. Poles and zeros are the roots of the denominator and numerator of the transfer function respectively.

3.4.4 Frequency Response

The frequency response of a filter describes how the filter alters the magnitude and phase of the input signal frequencies. The frequency response of a filter can be obtained by taking the Fourier transform of the impulse response of the filter.

$$H(\quad e^{j\omega})=\frac{Y(e^{j\omega})}{X(e^{j\omega})}$$

3.5 Filter Order:

The order of a discrete-time filter is the highest discrete-time delay used in the input-output equation of the filter. For Example in the above equations the filter order is the larger of the values of N or M. For continuous time filters the filter order is the order of the highest differential term used in the input-output equation of the filter.

3.6 Modelling of Earthquake Signals using Adaptive filters:

In modelling of Earthquake ground motion, adaptive filters are especially useful. Even when the information available is less, adaptive filters give good performance. Adaptive filter consists of two parts: a digital filter and an adaptive algorithm. Filters of different filter types, such as finite impulse response (FIR) and infinite impulse response (IIR) filters can be used. Owing to their

inherent simplicity and small number of coefficients to be adapted, sixth order IIR filter is chosen.

In this project three algorithms have been used, namely:

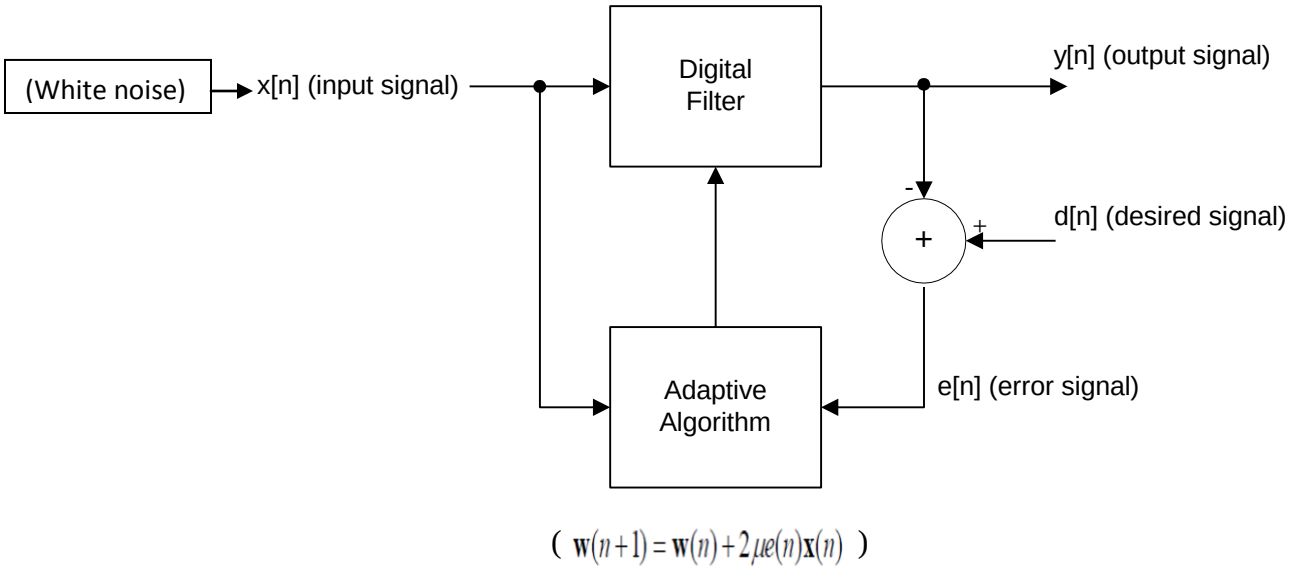
1. Standard LMS algorithm,
2. LMS algorithm with different step sizes for forward and feedback loop and,
3. NLMS algorithm

Chapter-4

Adaptive Filters

4.1. Introduction

An adaptive filter is defined as a self-designing system that relies for its operation on a recursive algorithm, which makes it possible for the filter to perform satisfactorily in an environment where knowledge of the relevant statistics is not available.



$$\left(\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu e(n)\mathbf{x}(n) \right)$$

Fig 6: Flow Diagram of Adaptive Filter

As shown in the adaptive filter diagram, adaptive filter consists of two parts: a digital filter and an adaptive algorithm. Filters of different filter types, such as finite impulse response (FIR) and infinite impulse response (IIR) filters can be used. Owing to their inherent simplicity and small number of coefficients to be adapted, sixth order IIR filter is chosen. Care has been taken to ensure stability of the adaptive filter.

As can be seen, the output signal is compared with the desired signal and the error is used to update the filter coefficients using an adaptive algorithm. The first algorithm that was used to design an adaptive filter was the least-mean-square (LMS) algorithm developed by Widrow and Hoff in 1959 in their study of a pattern recognition system called the adaptive linear element (Adaline). LMS algorithm is still used widely owing to its simplicity and robustness.

A system identification procedure was followed with input as white noise. The desired output is that of earthquake acceleration time history. The filter coefficients thus adapted so as to minimize the mean square error stabilize eventually. These filter coefficients are used to calculate the power spectral density function.

In this project three algorithms have been used, namely:

4. Standard LMS algorithm,
5. LMS algorithm with different step sizes for forward and feedback loop and,
6. NLMS algorithm

Firstly, optimal filtering is explained and then LMS algorithm is discussed. Thereafter, all three algorithms along with their implementations have been explained.

4.2 Optimal Filtering

The filter minimizes the estimation error e(n), such that the output signal y(n) resembles the desired signal d(n) as closely as possible.

In order to determine the optimal filter, a cost function J, which punishes the deviation e(n), is introduced. The larger the e(n), the higher cost.

The cost function $J(n)=E[|e(n)|^p]$ can be used for any $p\geq 1$, but usually $p= 2$ is taken. This choice gives a convex cost function which is referred to as the Mean Squared Error.

$J = E[e(n)e*(n)] = E[|e(n)|^2]$

Different algorithms have been invented to minimize this cost function.

4.3 Adaptive Algorithms

The real challenge while designing an adaptive filter is the choice of adaptive algorithm. There is always a trade-off between the implementation complexity and efficiency.

The adaptive algorithm must have following qualities:

1. Practical implementation.
2. Adapt the coefficients to their ideal value in a fast manner.
3. Provide the performance that is desired of the adaptive filter.

In this project three variants of LMS algorithm, which is the most widely used algorithm have been used.

4.4. Standard LMS Algorithm:

In LMS algorithm, with each iteration tap weights of the adaptive filter are updated according to the following equation:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu e(n)\mathbf{x}(n)$$

Here, x(n) is the input vector consisting of time delayed input values,i.e.

$x(n) = [x(n)\ x(n-1)\ x(n-2)\ ..\ x(n-N+1)]^T$.

The vector $w(n) = [w_0(n)\ w_1(n)\ w_2(n)\ ..\ w_{N-1}(n)]^T$ represents the coefficients or the tap weights of the adaptive IIR filter vector at time n.

The parameter μ is known as the step size parameter and is a small positive constant.

This step size parameter controls the influence of the updating factor. Selection of a suitable value for μ is necessary for the performance of the LMS algorithm. If it is too small, the time taken by the adaptive filter to converge on the optimal solution will be too long. If it is too large the adaptive filter becomes unstable and its output diverges.

a. Standard LMS filter implementation:

Each iteration of the standard LMS algorithm requires 3 steps:

1. Firstly, the output of the IIR filter, y(n) is calculated using equation:

$$y(n) = \sum_{i=0}^{L} a_i x(n-i) + \sum_{j=1}^{M} b_j y(n-j)$$

Where,

a_i and b_j are the input and feedback weights or filter coefficients.

y(n)is the filter output.

x_{n-1} is the filter input which is taken equal to the e_{n-i} because for PSDF estimation input signal needs to be white. Whiteness of e_{n-i} is mentioned in (Jiande Chen et. al., 1987).

M and L are the highest time delay for output and input in the input-output equation.

2. Secondly, the value of the error estimation is calculated using equation:

e(n)= d (n)-y(n)

Where, d(n) is the desired output, i.e. the earthquake acceleration time history.

3. Finally, the tap weights or filter coefficients of the IIR vector are updated in preparation for the next iteration, by equation:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu e(n)\mathbf{x}(n)$$

These equations have been implemented in python language .The program created is in annexure 1.

4.5. LMS algorithm with different step size parameters for forward and feedback loops:

The step sizes for adapting the input weights and for adapting the feedback weights are differently chosen (Jiande Chen et. al., 1987). In this way a better convergence is achieved. The structure of an adaptive IIR filter is shown in the figure.

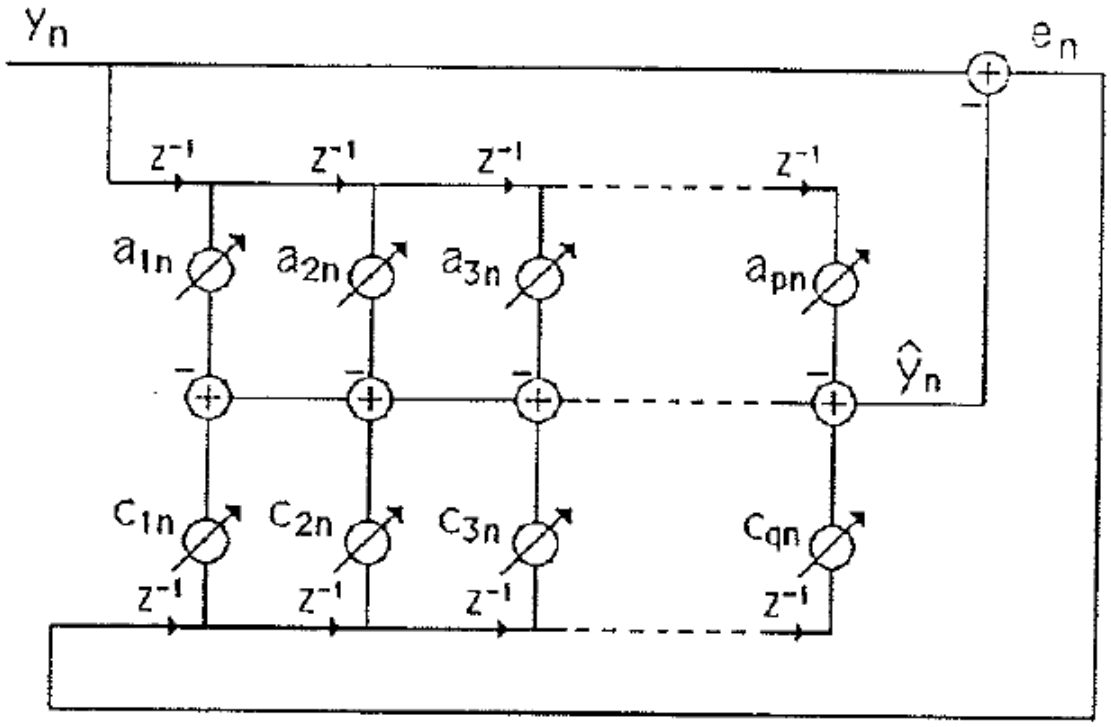


Fig 7: Structure of IIR adaptive filter with different step sizes for forward and feedback loop

In the fig. a_{in} and c_{in} are tap weights or filter coefficients.

a_{in} is input tap weight or filter coefficient.

c_{in} is output tap weight or filter coefficient.

a. Implementation of LMS algorithm with different step size parameters for forward and feedback loops:

1. Output is calculated as:

$$y_n = -\sum_{k=1}^{p} a_k y_{n-k} + \sum_{k=0}^{q} c_k n_{n-k}$$

$$= A_n^T Y_n + C_n^T \mathbf{e}_n$$

$$A_n^T = [a_{1n}, a_{2n}, \ldots, a_{pn}]$$
$$C_n^T = [c_{1n}, c_{2n}, \ldots, c_{qn}]$$
$$Y_n^T = [y_{n-1}, y_{n-2}, \ldots, y_{n-p}]$$

Where,

c_k and a_k are the input and feedback weights.
y_n is the filter output.
n_{n-k} is the filter input which is taken equal to the e_{n-k} because for PSDF estimation input needs to be white.

2. The error is calculated as:

e(n) = y(n)-d(n)

Where, d(n) is the desired output, i.e. the earthquake acceleration time history.

3. The equations used to update the filter coefficients are:

$$A_{n+1} = A_n + 2\mu_1 e_n Y_n$$

$$C_{n+1} = C_n + 2\mu_2 e_n \mathbf{e}_n$$

These equations have been implemented in python language .The program created is in annexure 2.

4.6. NLMS algorithm:

NLMS is normalised least mean square algorithm that is based on the minimisation of the least mean square error. One of the disadvantages of the LMS algorithm is that the step size parameter is fixed in all iterations. The only difference between the NLMS algorithm and the standard LMS algorithm is that the NLMS algorithm has a time-varying step size $\mu(n)$. This time varying step size improves the convergence of the adaptive algorithm.

This step size is proportional to the inverse of the total expected energy of the instantaneous values of the coefficients of the input vector x(n).The recursion formula for the NLMS algorithm is:

$$w(n+1)=w(n)+ \mu(n).x(n).e\ (n)$$

Where,

$$\mu\ (n)=\frac{\mu}{||x(n)||^2}$$

w(n) is array of filter coefficients.

e(n) is array of error signals.

x(n) is input vector.

a. Implementation of the NLMS algorithm:

The table contains the various equations that form the NLMS algorithm implementation. The implementation has been done in python language. The program created is in Annexure 3.

$y(n)$	$= \sum_{i=0}^{L} a_i x(n-i) + \sum_{j=1}^{M} b_j y(n-j)$
$\theta(n)$	$= [a_0(n), \cdots, a_{N-1}(n), b_1(n), \cdots, b_{M-1}(n)]^T$
$\Phi(n)$	$= [x(n), \cdots, x(n-N+1), y(n-1), \cdots, y(n-M+1)]^T$
$y(n)$	$= \theta^T(n)\Phi(n)$
$\varepsilon(n)$	$= d(n) - y(n)$
$\nabla_\theta y(n)$	$= \Phi(n) + \sum_{j=1}^{M} b_j \nabla_\theta y(n-j)$
$\theta(n+1)$	$= \theta(n) + \frac{\mu(n)}{\|\nabla_\theta y(n)\|^2}\varepsilon(n)\nabla_\theta y(n)$

Table 1

Here,

x(n) is the input signal.

y(n) is the filter output signal.

d(n) is the desired output signal.

θ(n) is an array consisting of the filter coefficients.

ϕ(n) is an array consisting of present and past inputs and past outputs.

ϵ(n) is the error signal.

μ(n) is the step size which controls the convergence speed of the algorithm.

$\nabla\theta y(n) = [x(n), \cdots, x(n-L), y(n-1), \cdots, y(n-M)]^T$ is an array of input and output samples.

Chapter-5

Power Spectral Density Function

5.1. Introduction:

The power-spectral density function (PSDF) provides statistical characterization forthe time-series functions. PSDF is necessary for complete statistical description for a Random process like an earthquake acceleration time history. Furthermore, the input-output relationship ofmechanical systems has been well defined in the theory of random processes in terms of PSDF. Thus, it can be used in analyzing many physical problems that are concerned with physical phenomenon that can be represented as mechanical systems.

PSDF can also be used to evaluate Kanai-Tajimi parameters for soil overburden. Kanai-Tajimi PSDF can be interpreted as PSDF corresponding to an "ideal white noise" excitation at bedrock level filtered through the soil deposits overburden. In this context, the K-T parameters can be interpreted as the soil overburden effective damping coefficient ξ_g, and natural frequency ω_g .

The spectral estimation problem is defined by the following informal definition:

> From a finite record of a stationary data sequence,
> estimate how the total power is distributed over frequency.

5.2. Mathematical definition:

The discrete-time signal { y(t); t = 0; +-1; +-2 …. } is assumed to be a sequence of random variables with zero mean:

E { y(t) } = 0 for all t ,

Where, E { . } denotes the expectation operator which averages over the ensemble of realizations. The autocovariance sequence (ACS) or covariance function of y(t)is defined as:

$$r(k) = E\{y(t)y^*(t-k)\}$$

It is assumed to depend only on the lag between the two samples averaged.

5.2.1 First Definition of Power Spectral Density:

The PSD is defined as the Discrete time Fourier transform of the covariance sequence:

$$\phi(\omega) = \sum_{k=-\infty}^{\infty} r(k)e^{-i\omega k} \qquad \text{(Power Spectral Density)}$$

The inverse transform, which recovers { r(k) } from given ϕ(ω), is

$$r(k) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \phi(\omega)e^{i\omega k}d\omega$$

Hence,

$$r(0) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \phi(\omega)d\omega$$

Since r(0) = E { $y(t)^2$ },measures the (average) power of E { y(t) }, the above equation shows that ϕ(ω) can indeed be named PSDF, as it represents the distribution of the (average) signal power over frequencies.

5.2.2 Second Definition of Power Spectral Density:

The second definition of PSDF is:

$$\phi(\omega) = \lim_{N\to\infty} E\left\{\frac{1}{N}\left|\sum_{t=1}^{N} y(t)e^{-i\omega t}\right|^2\right\}$$

5.3. Aliasing

We can see from either of these definitions that ϕ(ω) is a periodic function, with the period equal to 2 π. Hence, ϕ(ω) is completely described by its variation in the interval:

$$\omega \in [-\pi, \pi] \qquad \text{(radians per sampling interval)}$$

Since,

$$f = \frac{\omega}{2\pi} \qquad \text{(cycles per sampling interval)}$$

The PSDF can be viewed as a function of the frequency, which in accordance with above equation can be written as:

$$f \in [-1/2, 1/2]$$

The discrete time sequence { y(t) } is most commonly derived by sampling a continuous time signal. To avoid aliasing effects which might be incurred by the sampling process, the continuous time signal should be at least, approximately band limited in the frequency domain. Let F_0 denote the largest (significant) frequency component in the spectrum of the continuous signal, and let F_s be the sampling frequency. Then it follows from Shannon's sampling theorem that the continuous time signal can be exactly constructed from its samples {y(t) }, provided that

$$F_s \geq 2F_0$$

In particular, no frequency aliasing will occur when the above condition holds. Since the frequency variable, F, associated with the continuous time signal, is related to f by the equation:

$$F = f \cdot F_s$$

It follows that the interval of F is:

$$F \in \left[-\frac{F_s}{2}\ ,\ \frac{F_s}{2}\right] \qquad \text{(cycles/sec)}$$

5.4 The Spectral Estimation Problem

The spectral estimation problem can now be described in a more exact sense as follows:

From a finite–length record $\{y(1), \ldots, y(N)\}$ of a second–order stationary random process, determine an estimate $\hat{\phi}(\omega)$ of its power spectral density $\phi(\omega)$, for $\omega \in [-\pi, \pi]$

There are two basic approaches to spectral analysis:

- In the classical (or nonparametric) methods of spectral analysis,the studied signal is applied to a bandpass filter with a narrow bandwidth. Fillter output power divided by the filter bandwidth is spectral content of the input to the filter.
- In the parametric methods of spectral analysis,a model is to postulated for the data, which provides a means of parameterizing the spectrum, and hence the spectral estimation problem is reduced to that of estimating the parameters in the assumed model.

In this project parametric approach for spectral estimation is adopted. The model for data is of an ARMA process or that of a pole zero adaptive filter.

a. The method adopted for obtaining the PSDF is as follows (Jiande Chen et. al., 1987):

Equation for input-output of a digital filter with p poles and q zeros is given by:

$$y_n = -\sum_{k=1}^{p} a_k y_{n-k} + \sum_{k=0}^{q} c_k n_{n-k}$$

The initial values of the filter coefficients a_k and c_k are set as zero. The input is a white noise process and the output is compared with the desired signal, i.e. the earthquake acceleration time history. The error signal is used to update the filter coefficients till they stabilize.

Then its PSDF can be given by :

$$P_y(\omega) = \sigma_u^2 |H(\omega)|^2,$$

Here, H(ω) is transfer function of the adaptive filter. This definition of PSDF is applicable upon the stabilization of filter coefficients.

The power spectral density is then calculated as below:

$$P_y(\omega) = \frac{\sigma^2 \Delta t |1 + \sum_{k=1}^{q} c_k exp(-j\omega k \Delta t)|^2}{|1 + \sum_{k=1}^{p} a_k exp(-j\omega k \Delta t)|^2}$$

Where, σ^2 is standard deviation of the input white noise and,
$\omega = 2\pi f$,
Where, $-1/2\Delta t \leq f \leq 1/2\Delta t$.
Further, Δt represents the sampling interval.

CHAPTER-6

FILTERING OF EARTHQUAKE TIME HISTORY

1940 El Centro earthquake

Time: May 18, 1940 / 9:35 pm, Pacific Standard Time
Location: El Centro a city in Imperial valley of California region
Magnitude (M_W): 6.9
Location of epicenter: 32.733° N 115.5°W
***Component*:**N-S Component
***Sampling interval*:**0.02 Sec
***Location of Station*:** El Centro, CA-Array Sta 9 Imperial valley region district
Epicentral distance of recorded station: 11.4 km

The 1940 Imperial Valley earthquake caused at least $6 million in direct damage (not taking into consideration the crops lost due to damage of irrigation systems) and was directly responsible for the deaths of eight people, and, indirectly, for several others. At least 20 people were seriously injured. Damage to irrigation systems was widespread. Rails were bent out of line in three locations where they crossed the Imperial fault, and several railroad bridges were damaged, both in California and Mexico.

Acceleration time history of El Centro Earthquake recorded at 0.02 sec sampling interval.

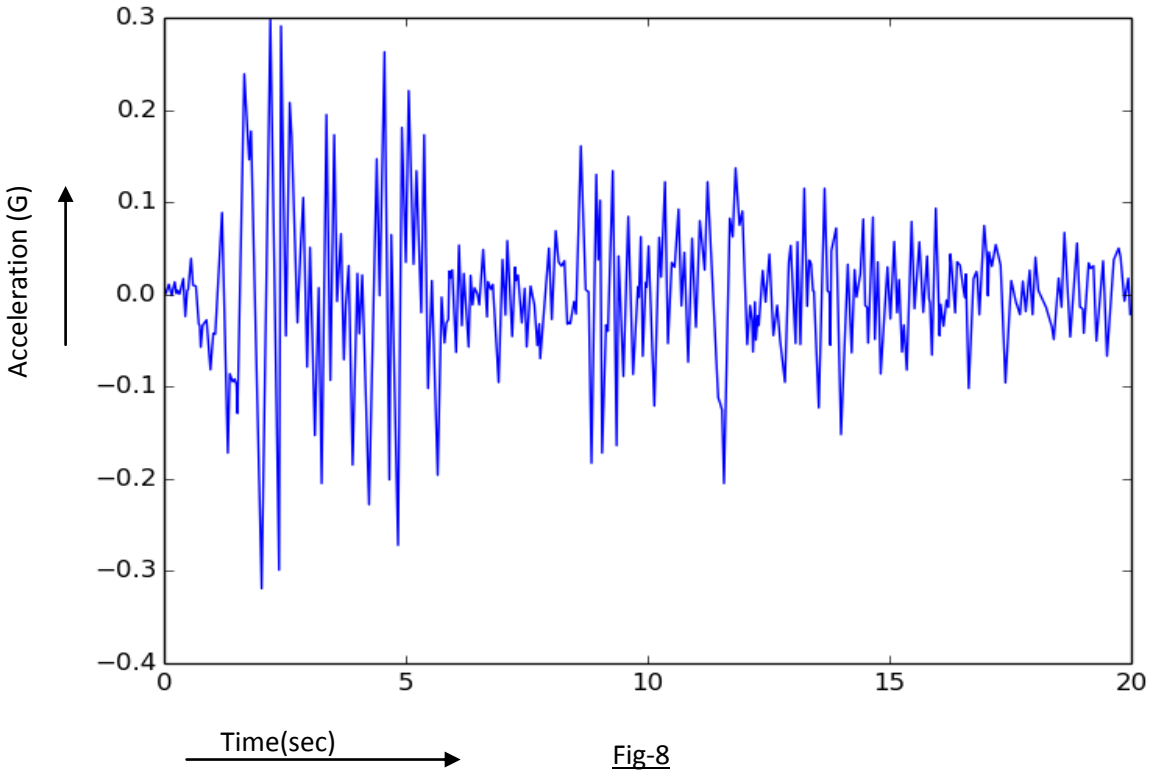


Fig-8

Filter output (red) vs Earthquake acceleration time history (green) for LMS algorithm with different step sizes for feed forward and feedback loops:

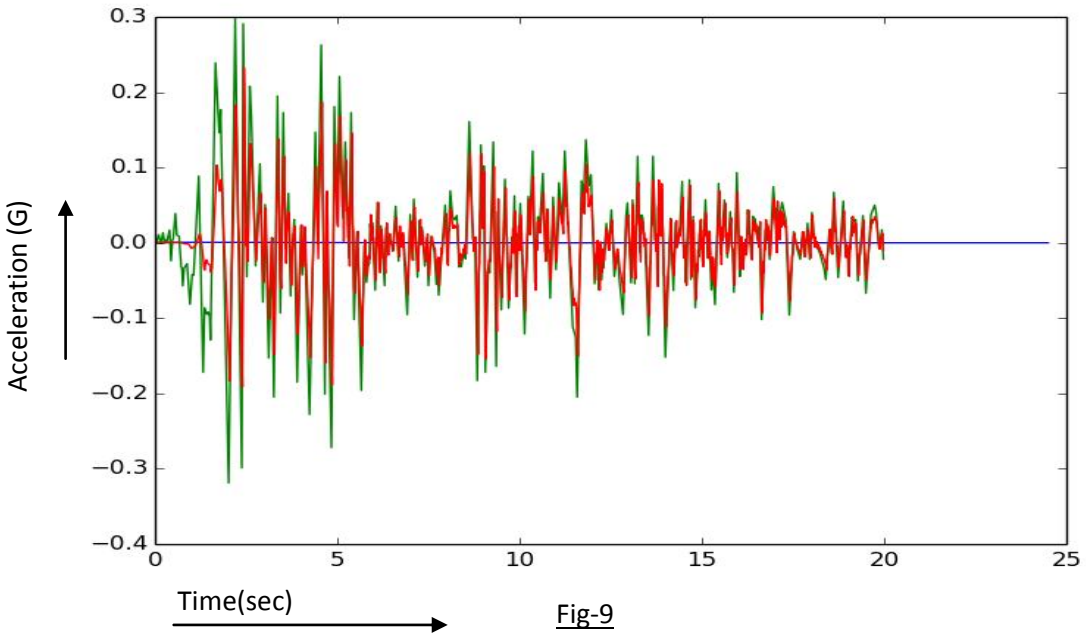


Fig-9

PSDF vs Frequency Plot for LMS algorithm with different step sizes for feed forward and feedback loops:

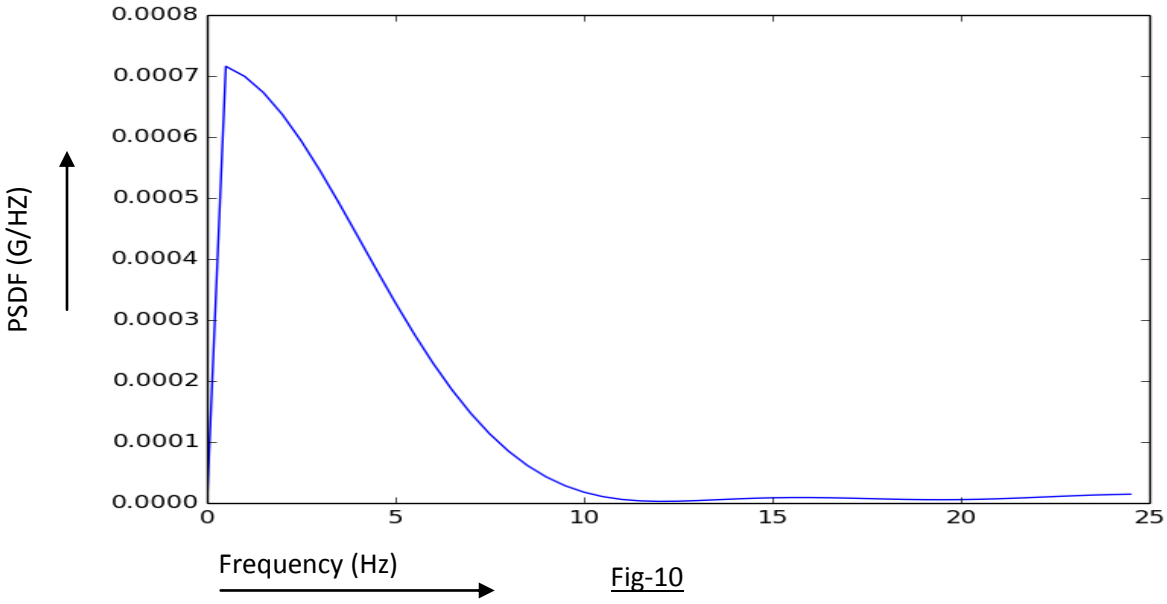


Fig-10

Filter output (red) vs Earthquake acceleration time history (green) for NLMS algorithm:

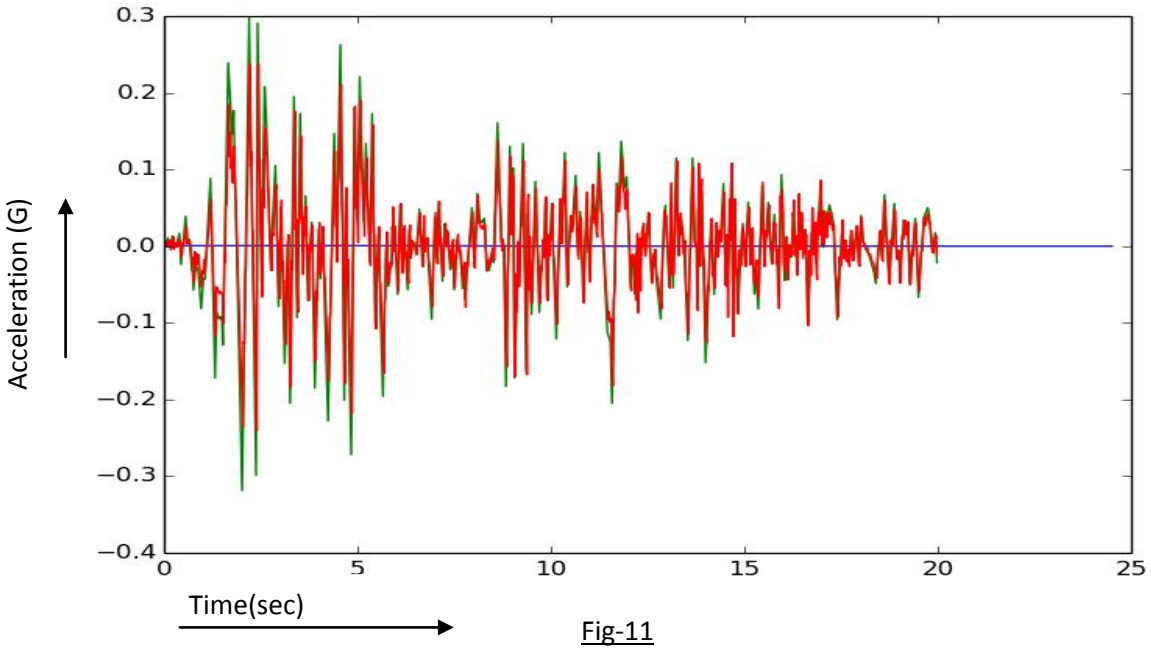


Fig-11

PSDF vs Frequency Plot for NLMS algorithm:

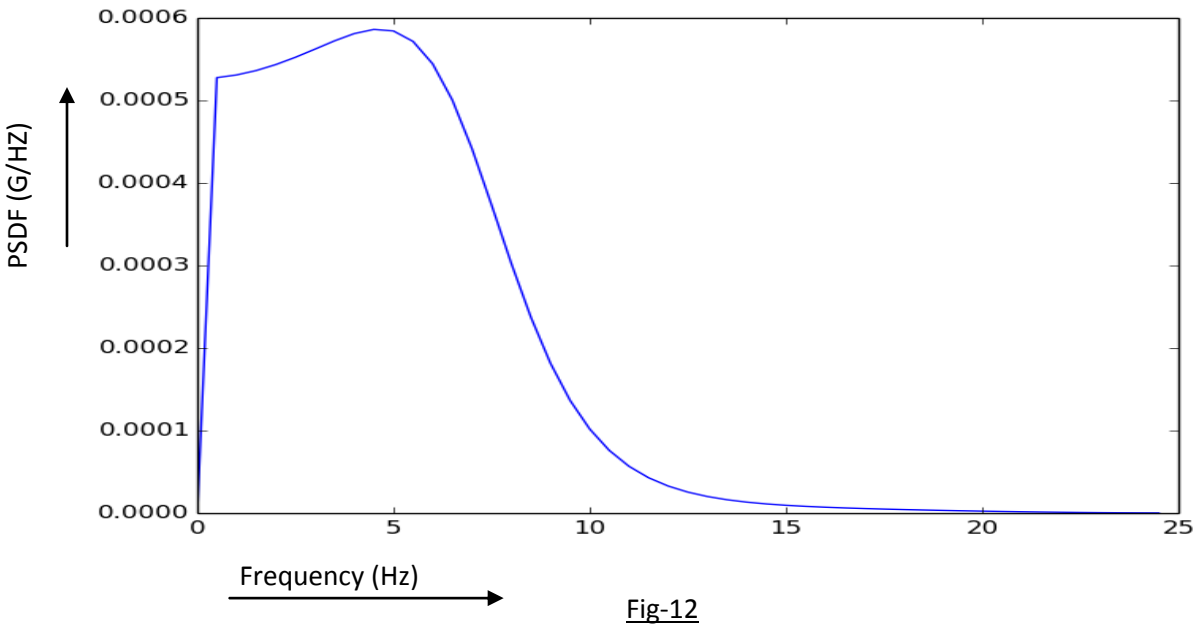


Fig-12

Filter output (red) vs Earthquake acceleration time history (green) for Standard LMS algorithm:

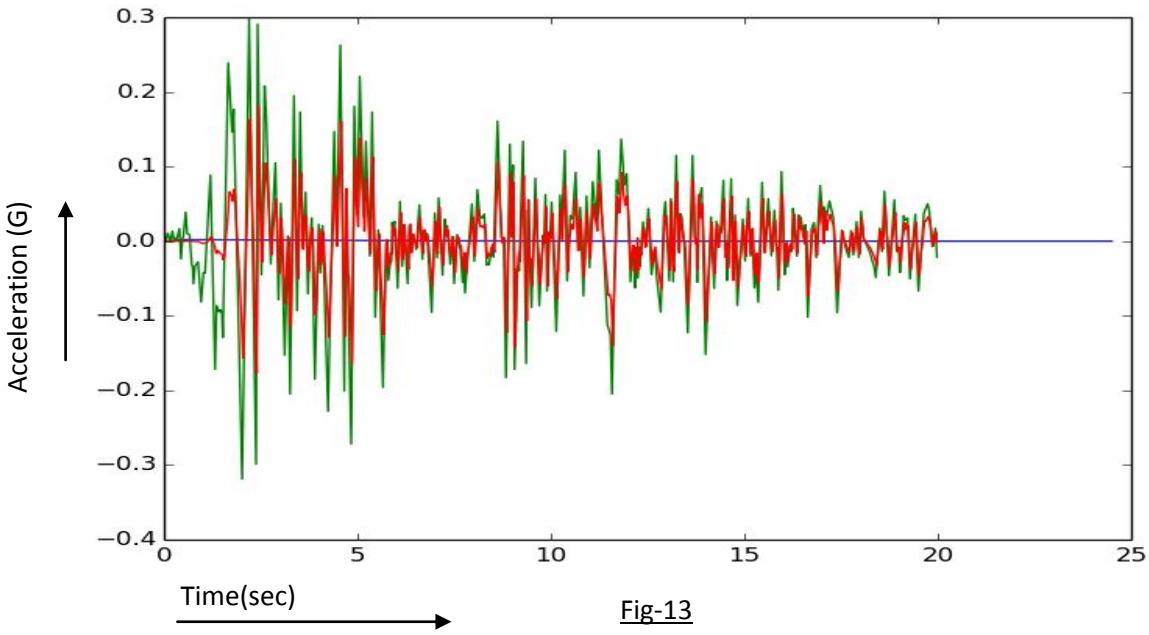


Fig-13

PSDF vs Frequency Plot for Standard LMS algorithm:

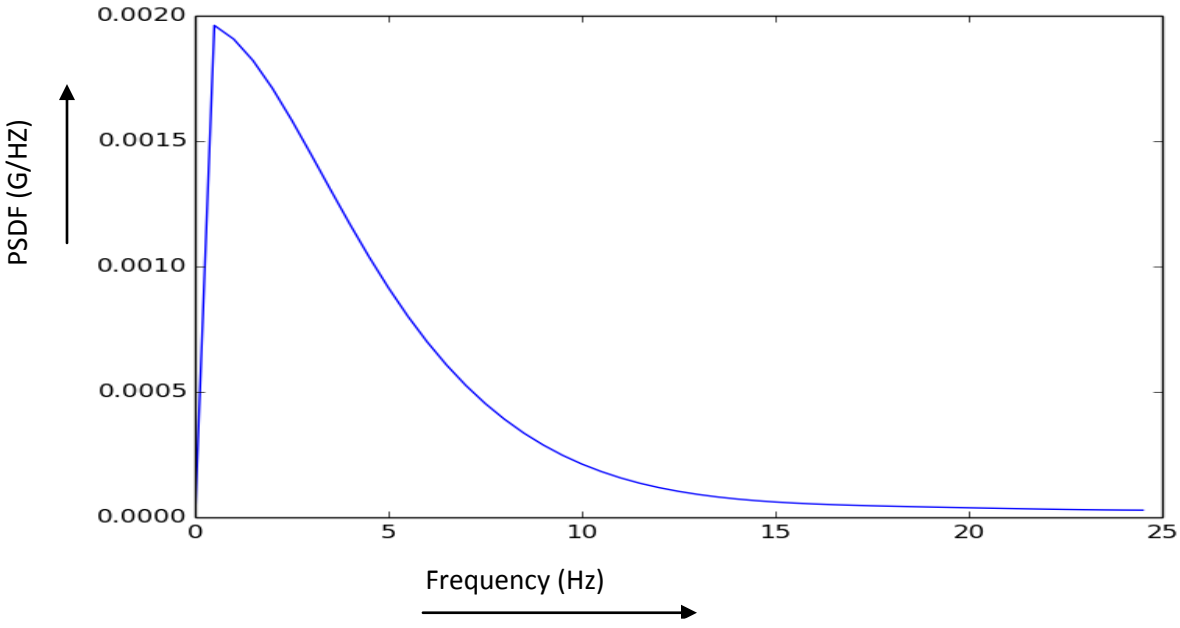


Fig-14

1933Long beach earthquake

Time: March 10, 1933 / 5:54 pm, Pacific Standard Time
Location: City of Long beach, California region
Magnitude (M_W): 6.4
Location of epicenter: 33°35' N 117°59' W
***Component*:**S Component
***Sampling interval*:**0.02 Sec
***Location of Station*:** Station no. 131, Public Utilities Building, Longbeach ,CA

Although the earthquake was only moderate in terms of magnitude, it caused serious damage to weak masonry structures on land fill from Los Angeles south to Laguna Beach. Cost of property damage was estimated at $40 million, and 115 people were killed.Severe property damage occurred at Compton, Long Beach, and other towns in the area. Most of the damage was due to land fill, or deep water-soaked alluvium or sand, and due to badly designed buildings. Minor disturbances of ground water, secondary cracks in the ground, and slight earth slumps occurred, but surface faulting was not observed. Along the shore between Long Beach and Newport Beach, the settling or lateral movement of road fills across marshy land caused much damage to the concrete highway surfaces and to approaches to highway bridges.

Acceleration time history of Long Beach Earthquake recorded at 0.02 sec sampling interval.

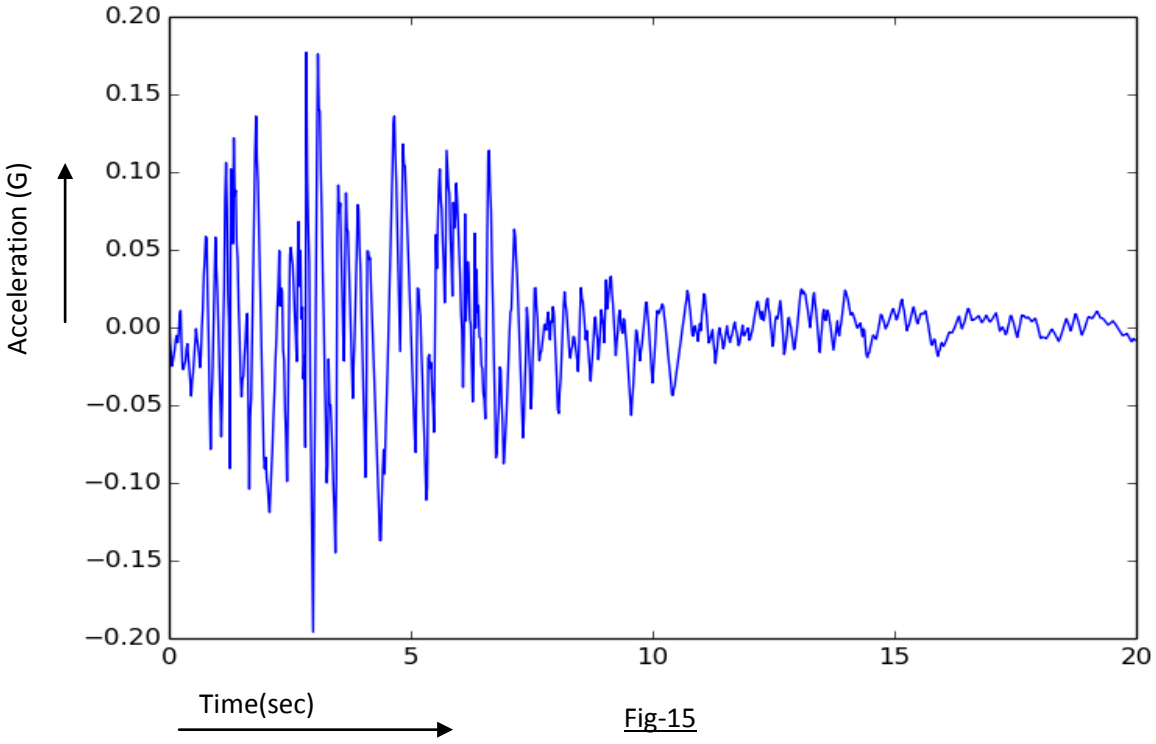


Fig-15

Filter output (red) vs Earthquake acceleration time history (green) for LMS algorithm with different step sizes for feed forward and feedback loops:

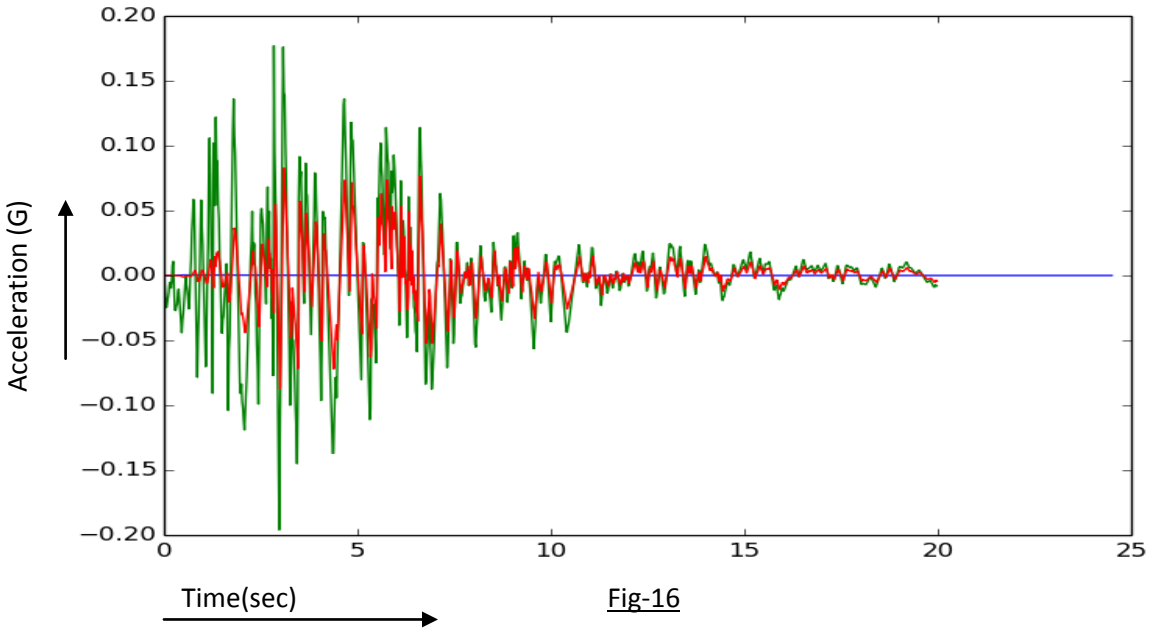


Fig-16

PSDF vs Frequency Plot for LMS algorithm with different step sizes for feed forward and feedback loops:

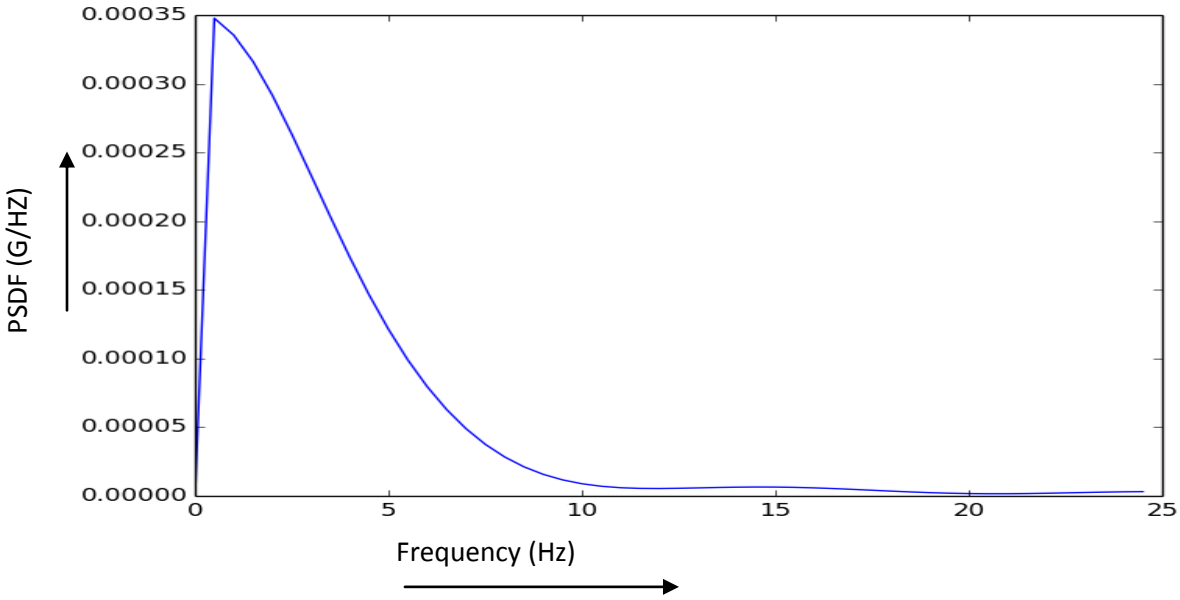


Fig-17

Filter output (red) vs Earthquake acceleration time ·n) for NLMS algorithm:

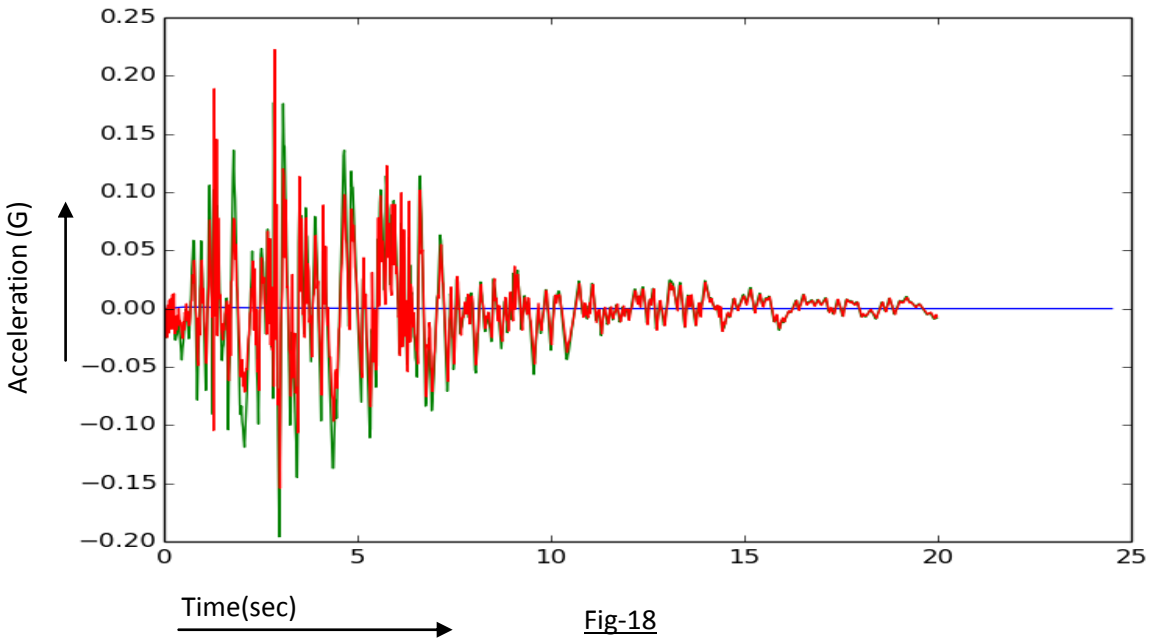


Fig-18

PSDF vs Frequency Plot for NLMS algorithm:

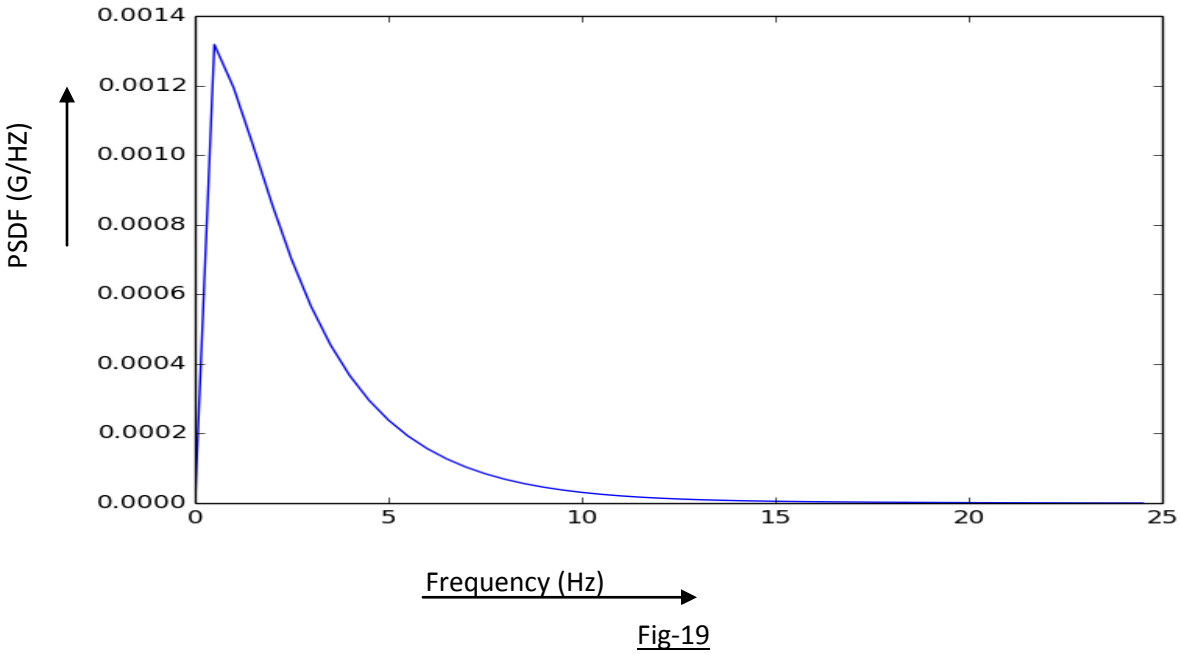


Fig-19

Filter output (red) vs Earthquake acceleration time history (green) for Standard LMS algorithm:

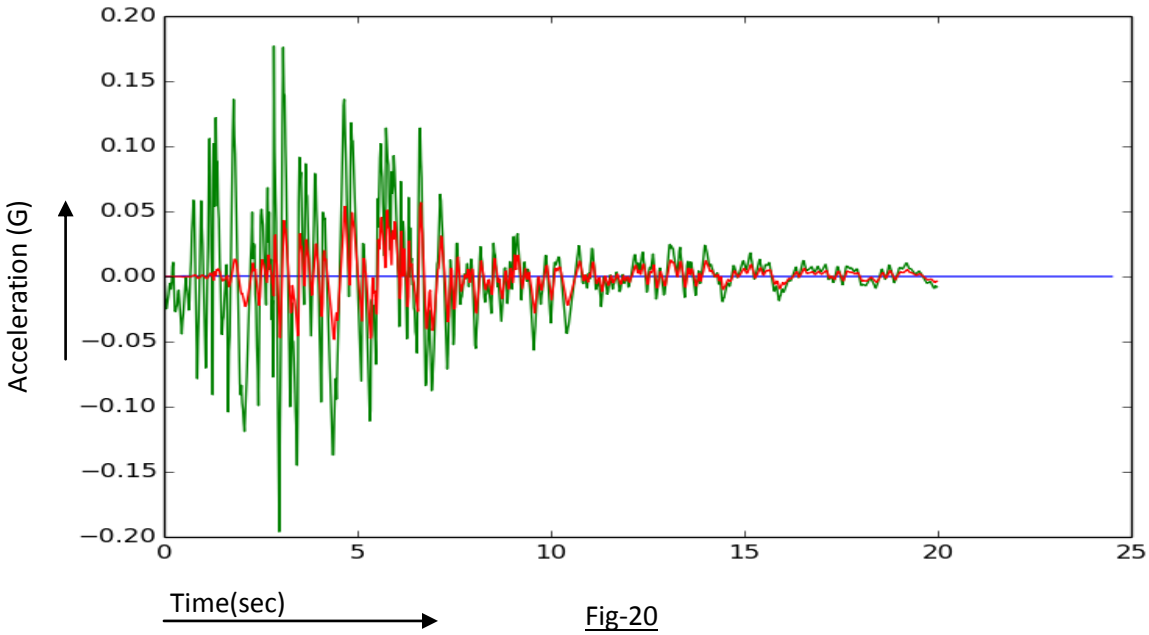


Fig-20

PSDF vs Frequency Plot for Standard LMS algorithm:

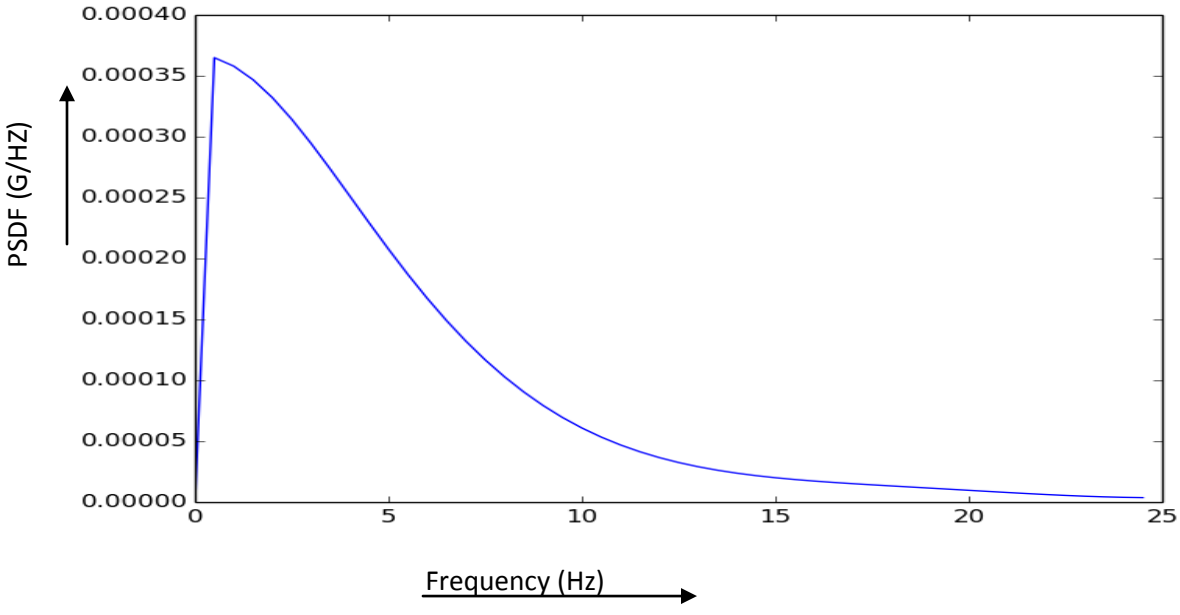


Fig-21

1952 Kern Country Earthquake

Time: July21, 1952 / 16:52:14, Pacific Daylight Time
Location: City of Long beach, California region
Magnitude (M_W): 7.3
Location of epicenter: 34.96N 119.00W
***Component*:**S69E Component
***Sampling interval*:**0.02 Sec
***Location of Station*:**Taft Lincoln School Tunnel, CA ,USA

This earthquake was the largest in the United States since the San Francisco shock of 1906. It claimed 12 lives and caused property damage estimated at $60 million. The earthquake cracked reinforced-concrete tunnels having walls 46 cms thick; it shortened the distance between portals of two tunnels by about 2.5 m and bent the rails into S-shaped curves. At Owens Lake (about 160 kilometers from the epicenter), salt beds shifted, and brine lines were bent into S-shapes.

Acceleration time history of Long Beach Earthquake recorded at 0.02 sec sampling interval.

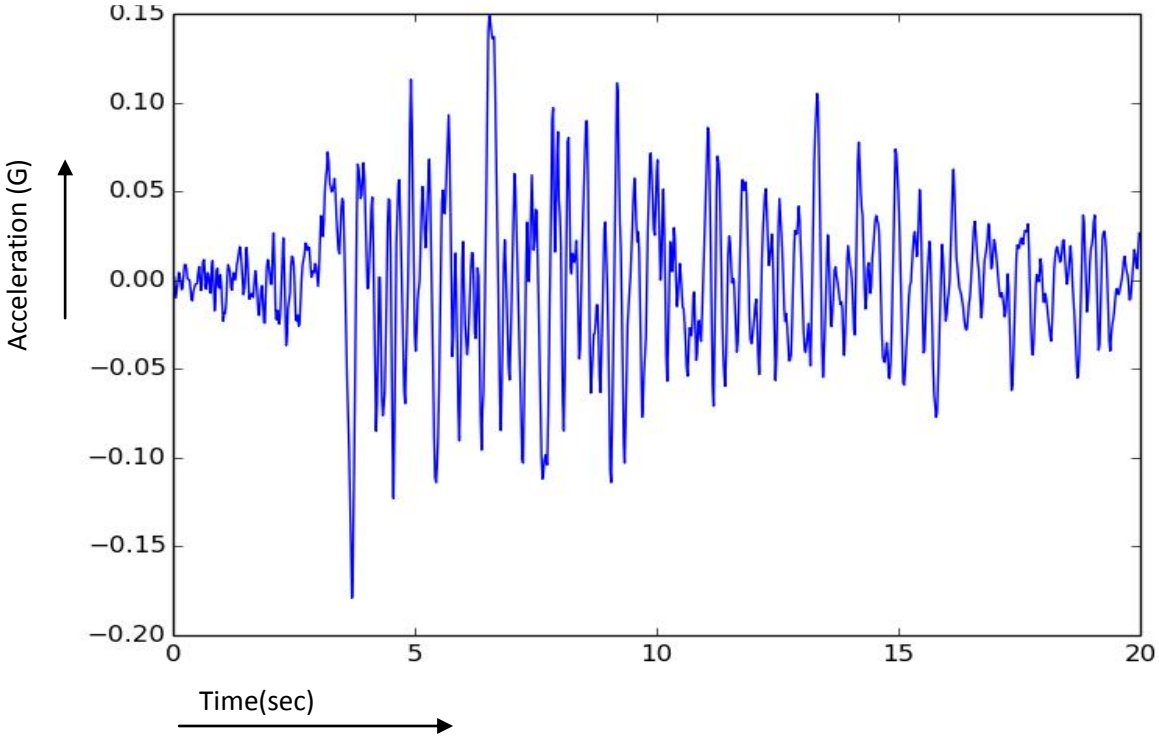


Fig-22

Filter output (red) vs Earthquake acceleration time history (green) for LMS algorithm with different step sizes for feed forward and feedback loops:

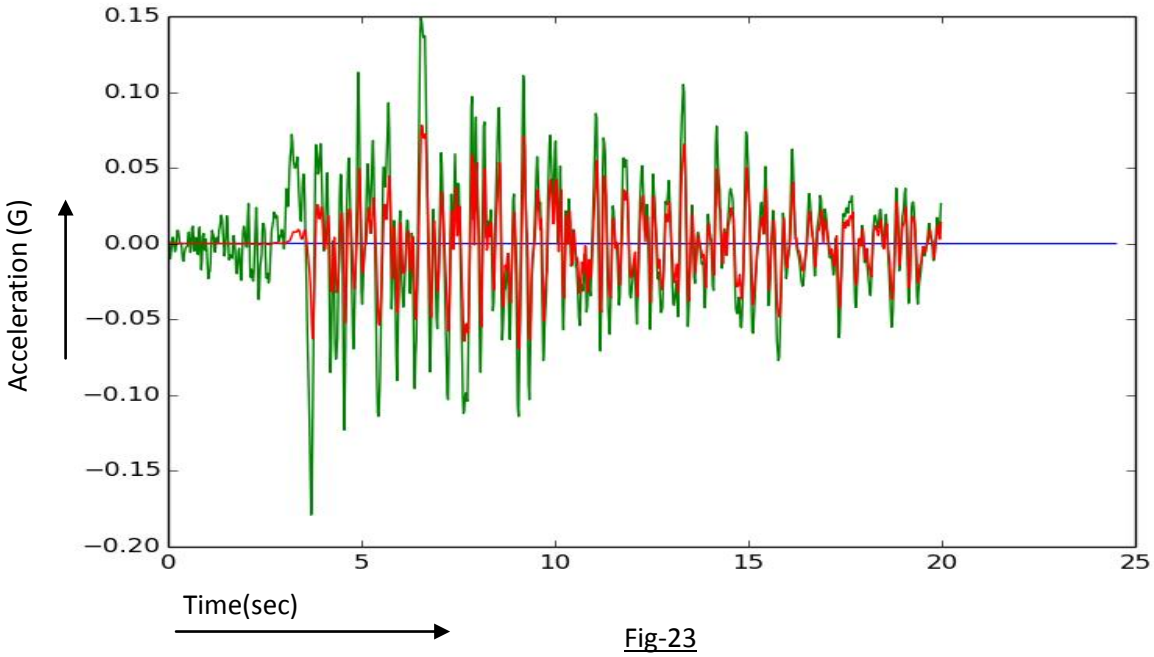


Fig-23

PSDF vs Frequency Plot for LMS algorithm with different step sizes for feed forward and feedback loops:

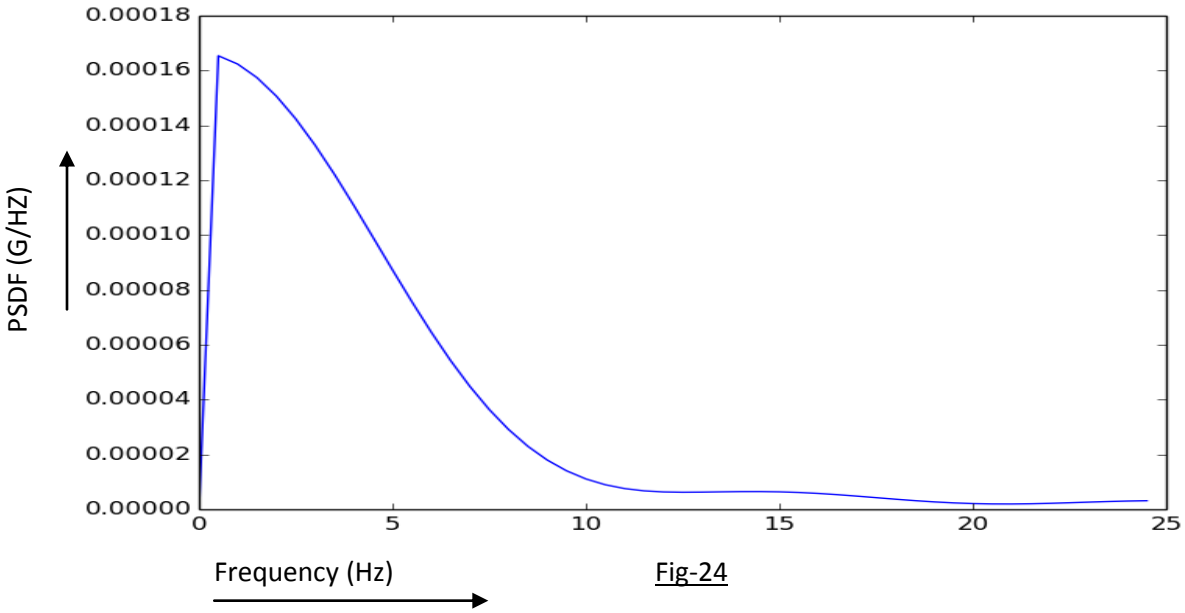


Fig-24

Filter output (red) vs Earthquake acceleration time history (green) for NLMS algorithm:

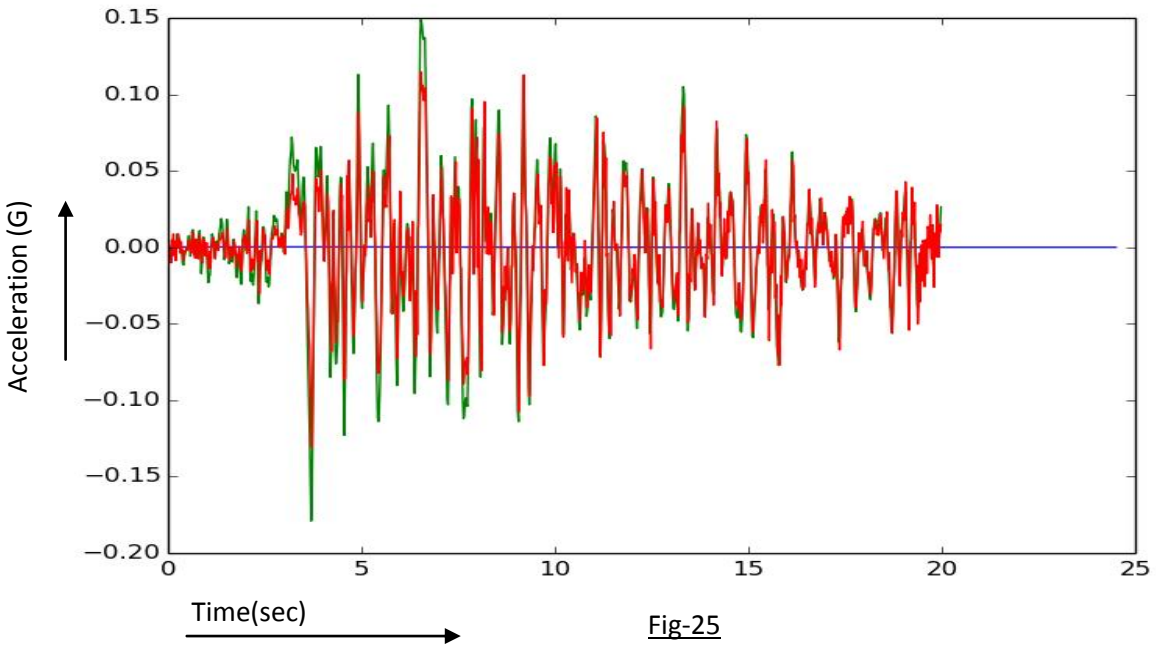


Fig-25

PSDF vs Frequency Plot for NLMS algorithm:

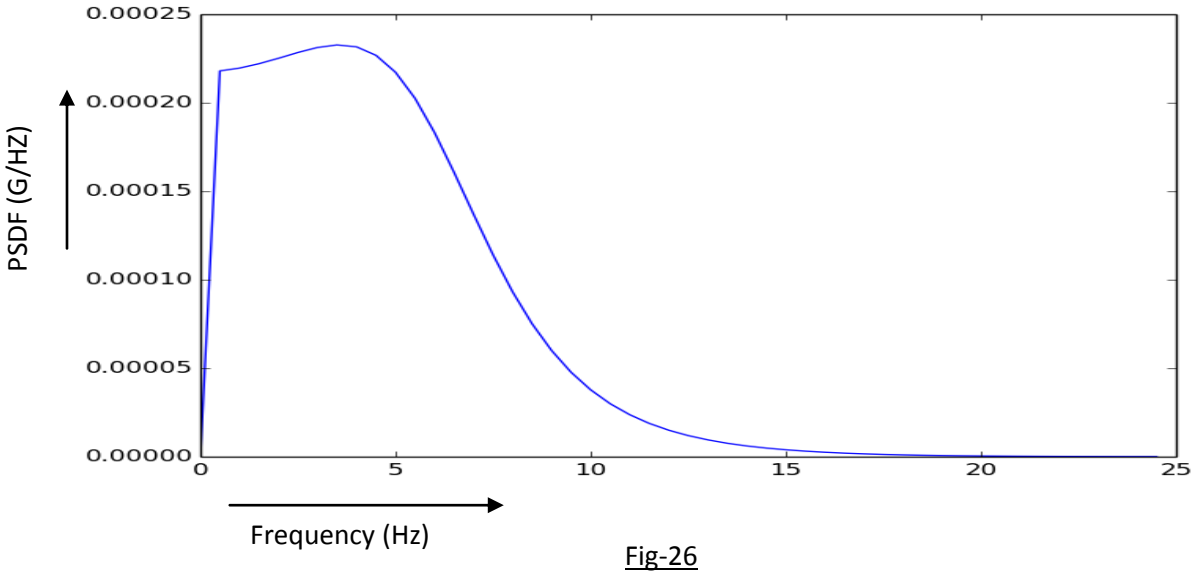


Fig-26

Filter output (red) vs Earthquake acceleration time history (green) for Standard LMS algorithm:

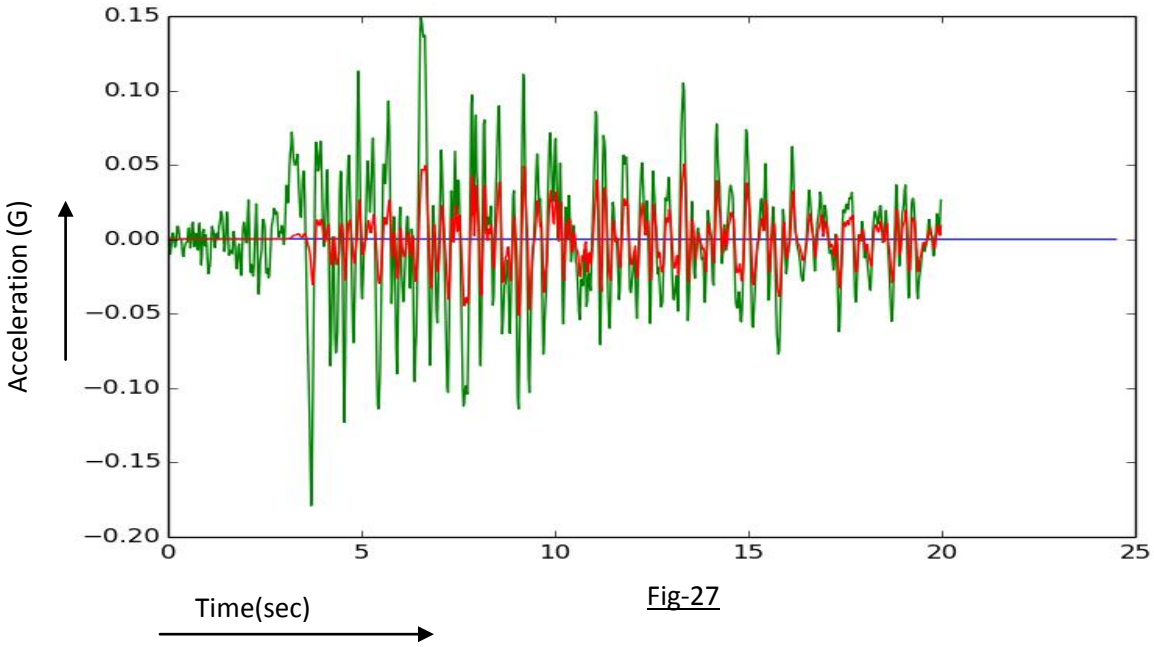


Fig-27

PSDF vs Frequency Plot for Standard LMS algorithm:

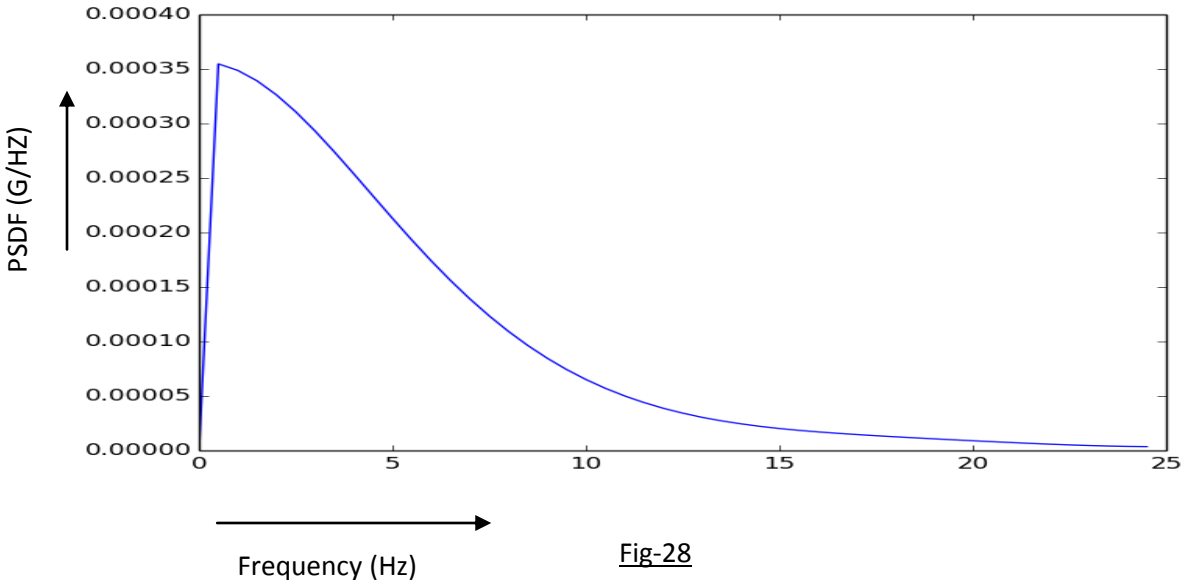


Fig-28

1971 San Fernando Earthquake

Time: February 9, 1971 / 6:01 am PST
Location: City of Long beach, California region
Magnitude (M_W): 6.5
Location of epicenter: 34° 24.67' N, 118° 24.04' W
***Component*:**S61E Component
***Sampling interval*:**0.02 Sec
***Location of Station*:**3440 University Avenue, basement, Los Angeles, Cal.

This earthquake is also known as the **Sylmar Earthquake.** It occurred on the San Fernando fault zone, a zone of thrust faulting which broke the surface in the Sylmar-San Fernando Area.The earthquake caused over $500 million in property damage and 65 deaths most of which occurred when the Veteran's Administration Hospital collapsed. Several other hospitals, including the Olive View Community Hospital in Sylmar suffered severe damage. Newly constructed freeway overpasses also collapsed, in damage scenes similar to those which occurred 23 years later in the 1994 Northridge Earthquake.

Acceleration time history of 1971 San Fernando recorded at 0.02 sec sampling interval.

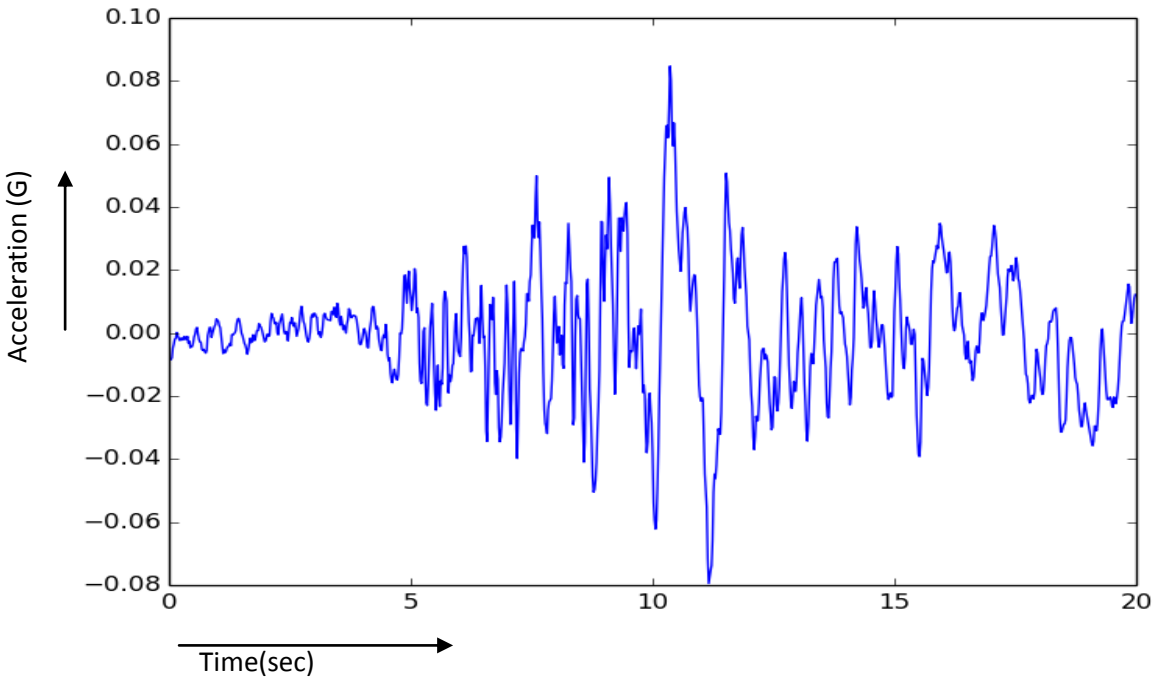


Fig-29

Filter output (red) vs Earthquake acceleration time history (green) for LMS algorithm with different step sizes for feed forward and feedback loops:

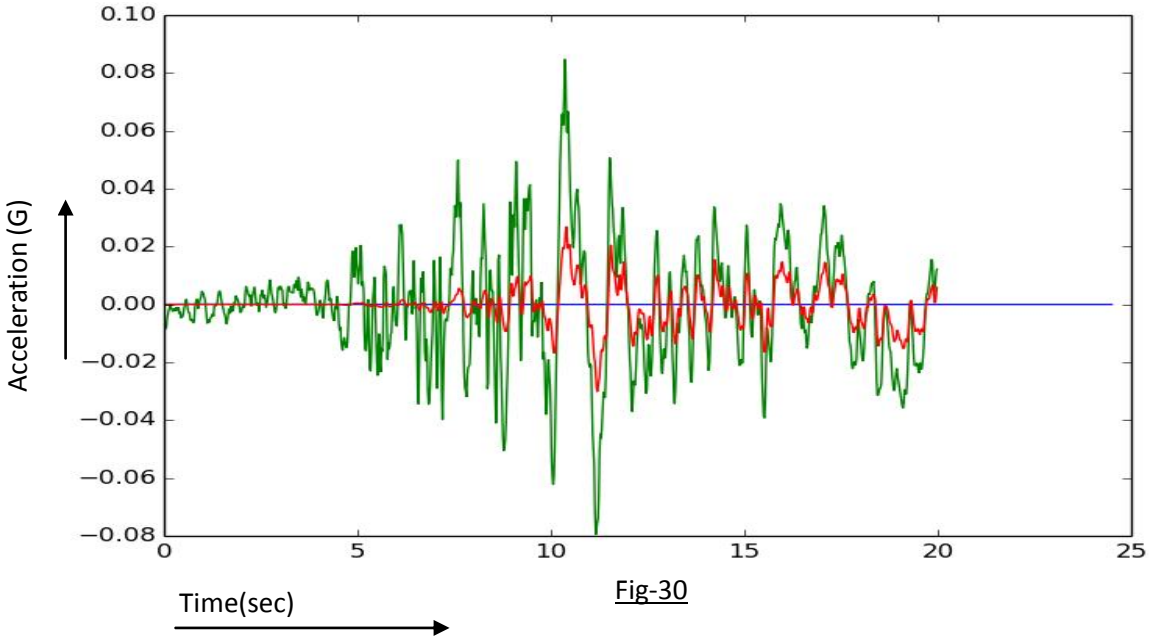


Fig-30

PSDF vs Frequency Plot for LMS algorithm with different step sizes for feed forward and feedback loops:

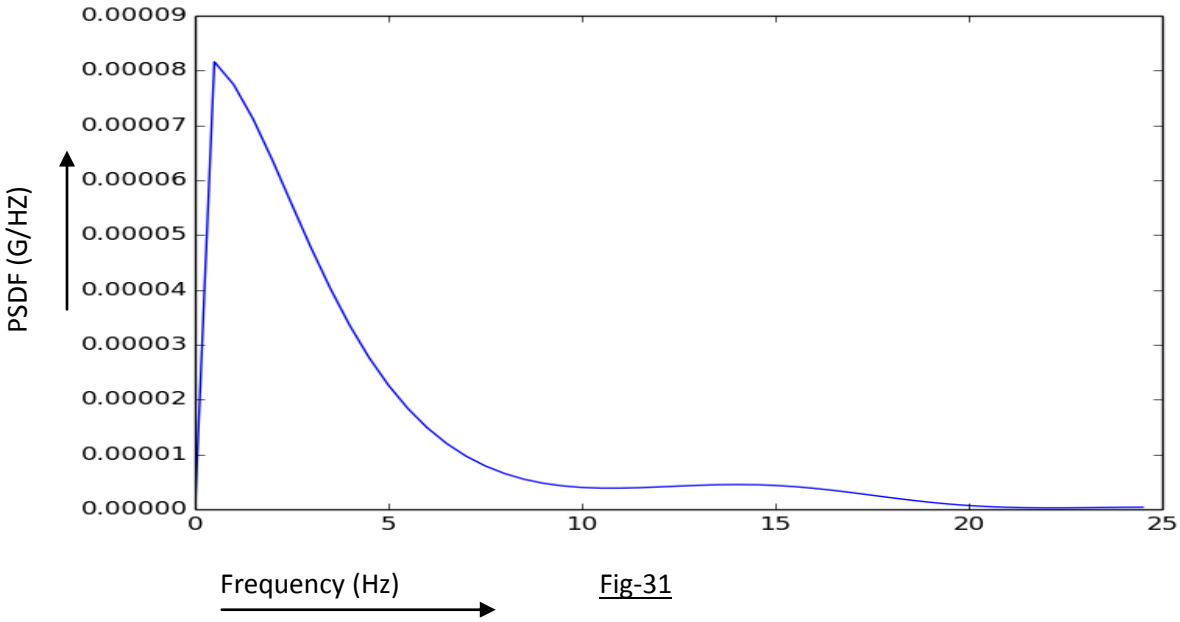


Fig-31

Filter output (red) vs Earthquake acceleration time history (green) for NLMS algorithm:

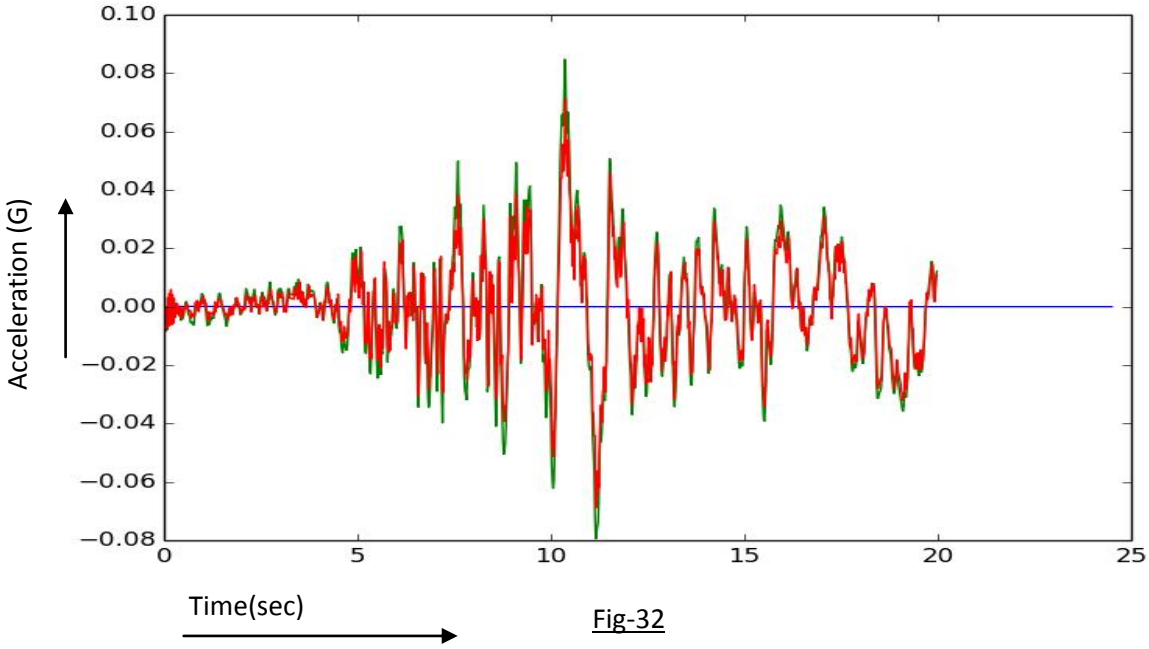


Fig-32

PSDF vs Frequency Plot for NLMS algorithm:

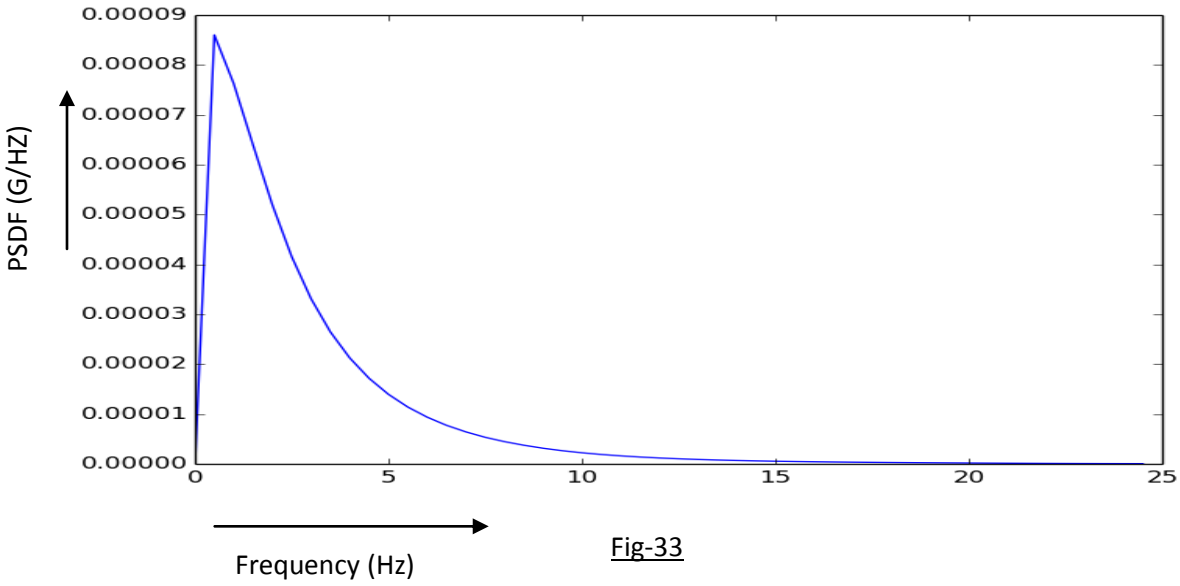


Fig-33

Filter output (red) vs Earthquake acceleration time history (green) for Standard LMS algorithm:

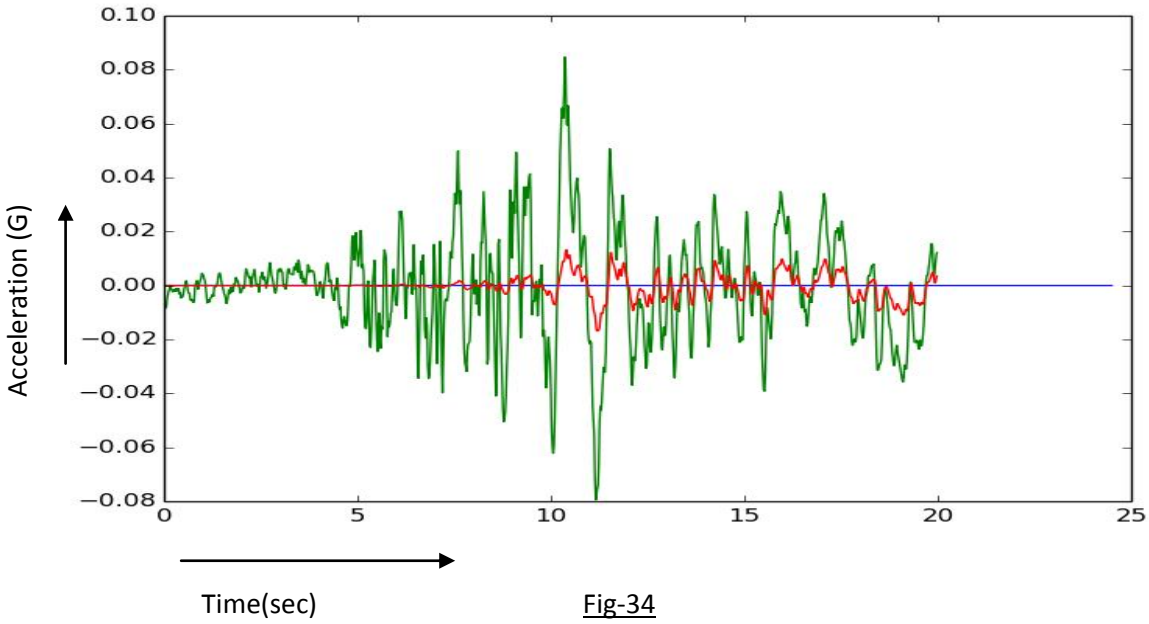


Fig-34

PSDF vs Frequency Plot for Standard LMS algorithm:

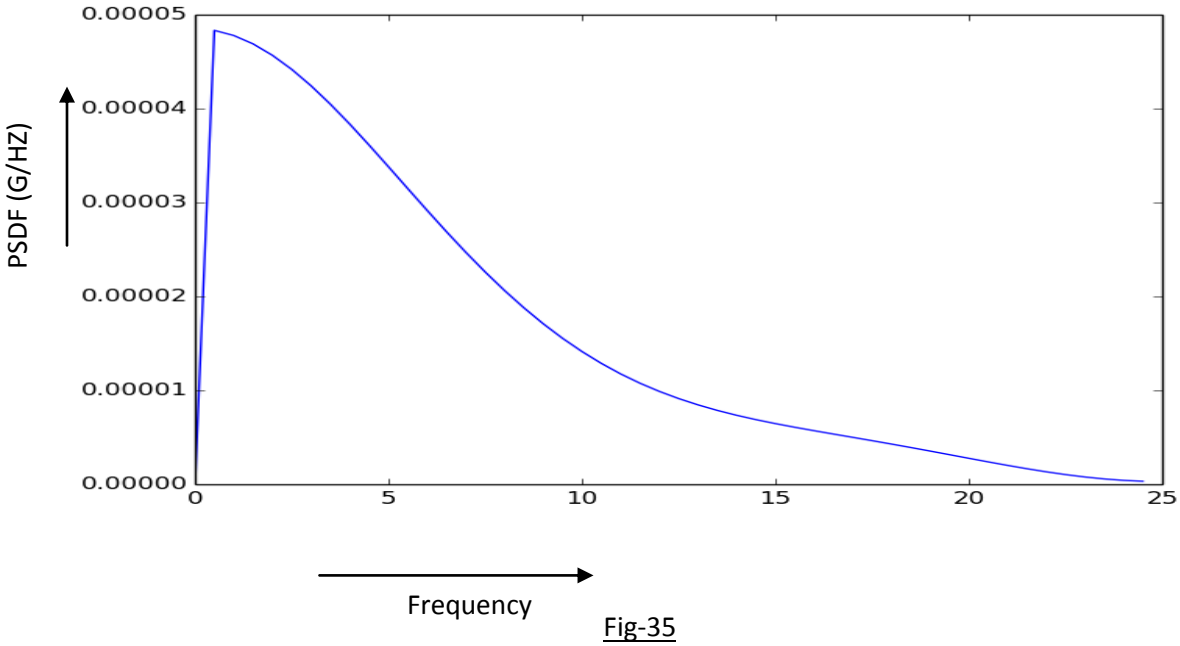


Fig-35

CHAPTER-7

CONCLUSIONS AND FUTURE SCOPE OF STUDY

7.1. Conclusions

Adaptive filter has been successfully constructed and its output is found to trace the input signal. Among the three algorithms used, it is evident from the adaptive filter output vs earthquake acceleration time history that NLMS algorithm output traces the earthquake acceleration time history most closely. Hence in this manner, we have successfully modelled the ground and replicated its behaviour.

PSDF for all three algorithms have been plotted for four different earthquakes.

A user manual to use the programs developed in an efficient manner has been developed.

7.2. Further scope of work

Further scope is to improve this filter to produce a better estimate of the power spectral density function. This can be achieved by using advanced adaptive algorithms. Further, kanaitajimi parameters can be calculated using the PSDF obtained.

APPENDIX I

1. **Python program for adaptive filter(sixth order) for standard LMS algorithm**:

```
Importnumpy as np
from math import tan,pi
importmatplotlib.pyplot as pyplot
eq= np.loadtxt('sanf.txt')

Power_spectral_density=np.zeros(54)
deffiltercoeff(uk):

        mu = 1

        u =   np.array(uk)

        b0=1
        b1=0
        b2=0
        a1=0
        a2=0

        theta=np.array([b0,b1,b2,a1,a2])
        e=np.zeros(len(u))
        x=np.zeros(len(u))
        store=np.zeros((len(u),5))

        for k in range(2,len(u)):

                x[k]= theta[0]*e[k]+theta[1]*e[k-1]+theta[2]*e[k-2]+theta[3]*x[k-1]+theta[4]*x[k-2]
                e[k]=u[k]-x[k]
                theta=theta+(mu)*e[k]*np.array([e[k-1],e[k-2],e[k-3],x[k-1],x[k-2]])
                store[k]=theta


        return theta

def filter(uk):

        mu = 1

        u =   np.array(uk)

        b0=1
        b1=0
        b2=0
        a1=0
        a2=0

        theta=np.array([b0,b1,b2,a1,a2])
        e=np.zeros(len(u))
        x=np.zeros(len(u))
        store=np.zeros((len(u),5))

        for k in range(2,len(u)):

                x[k]= theta[0]*e[k]+theta[1]*e[k-1]+theta[2]*e[k-2]+theta[3]*x[k-1]+theta[4]*x[k-2]
                e[k]=u[k]-x[k]
                theta=theta+(mu)*e[k]*np.array([e[k-1],e[k-2],e[k-3],x[k-1],x[k-2]])
                store[k]=theta


        return x
```

```
deffilter_white(uk):

        mu = 1

        u =   np.array(uk)

        b0=1
        b1=0
        b2=0
        a1=0
        a2=0

        theta=np.array([b0,b1,b2,a1,a2])
        e=np.zeros(len(u))
        x=np.zeros(len(u))
        store=np.zeros((len(u),5))

        for k in range(2,len(u)):

                x[k]= theta[0]*e[k]+theta[1]*e[k-1]+theta[2]*e[k-2]+theta[3]*x[k-1]+theta[4]*x[k-2]
                e[k]=u[k]-x[k]
                theta=theta+(mu)*e[k]*np.array([e[k-1],e[k-2],e[k-3],x[k-1],x[k-2]])
                store[k]=theta


        return e

z=5
y=[]
theta=filtercoeff(eq)

deffrange(start,end,step):
        return map(lambda x: x*step, range(int(start*1./step),int(end*1./step)))

for x in frange(0,len(eq)*0.02,0.02):
        y.append(x)
ps=[]
for x in frange(0.5,25,0.5):
        ps.append(x)
ps=np.array(ps)
ps=np.append(np.array([0,0,0,0,0]),ps)

for f in frange(0.5,25,0.5):
        w=2*pi*f
        Power_spectral_density[z]= (((np.std(filter_white(eq)))**2)*0.02*(abs(1+theta[0]*np.exp(-1j*w*0.02)+theta[1]*np.exp(-
2j*w*0.02)+theta[2]*np.exp(-3j*w*0.02)))**2)/(abs(1-theta[3]*np.exp(-1j*w*0.02)-theta[4]*np.exp(-2j*w*0.02)))**2
        z=z+1

pyplot.plot(ps,Power_spectral_density)
pyplot.savefig('PSDF')
pyplot.plot(y,eq)
pyplot.savefig('eq')
pyplot.plot(y,filter(eq))
pyplot.savefig('filtered _Data')
```

2. **Python program for adaptive filter(sixth order) for LMS algorithm with different step sizes for forward and feedback loop**:

```
Importnumpy as np
from math import tan,pi
importmatplotlib.pyplot as pyplot

eq= np.loadtxt('sanf.txt')
Power_spectral_density=np.zeros(54)

def filter(earthquake):
        mu1 = 1
        mu2 = 1
        y = np.array(earthquake)
        c0=1
        c1=0
        c2=0
        c3=0
        a1=0
        a2=0
        a3=0
        an=np.array([a1,a2,a3])
        cn=np.array([c1,c2,c3])
        y_cap=np.zeros(len(y))
        e=np.zeros(len(y))
        store=np.zeros((len(y),7))
        for k in range(3,len(y)):
                y_cap[k]= -an[0]*y_cap[k-1]-an[1]*y_cap[k-2]-an[2]*y_cap[k-3]+c0*e[k]+cn[0]*e[k-1]+cn[1]*e[k-2]+cn[2]*e[k-3]
                e[k]=y[k]-y_cap[k]
                an=an+2*(mu1)*e[k]*np.array([y_cap[k-1],y_cap[k-2],y_cap[k-3]])
                cn=cn+2*(mu2)*e[k]*np.array([e[k-1],e[k-2],e[k-3]])
                store[k]=[c0,cn[0],cn[1],cn[2],an[0],an[1],an[2]]

        returny_cap
deffilter_coeff(earthquake):
        mu1 = 1
        mu2 = 1
        y = np.array(earthquake)
        c0=1
        c1=0
        c2=0
        c3=0
        a1=0
        a2=0
        a3=0
        an=np.array([a1,a2,a3])
        cn=np.array([c1,c2,c3])
        y_cap=np.zeros(len(y))
        e=np.zeros(len(y))
        store=np.zeros((len(y),7))
        for k in range(3,len(y)):

                y_cap[k]= -an[0]*y_cap[k-1]-an[1]*y_cap[k-2]-an[2]*y_cap[k-3]+c0*e[k]+cn[0]*e[k-1]+cn[1]*e[k-2]+cn[2]*e[k-3]
                e[k]=y[k]-y_cap[k]
                an=an+2*(mu1)*e[k]*np.array([y_cap[k-1],y_cap[k-2],y_cap[k-3]])
                cn=cn+2*(mu2)*e[k]*np.array([e[k-1],e[k-2],e[k-3]])
                store[k]=[c0,cn[0],cn[1],cn[2],an[0],an[1],an[2]]
        return store[k]
deffilter_white(earthquake):

        mu1 = 1
        mu2 = 1
        y =   np.array(earthquake)
        c0=1
        c1=0
        c2=0
        c3=0
        a1=0
        a2=0
```

```
        a3=0
        an=np.array([a1,a2,a3])
        cn=np.array([c1,c2,c3])
        y_cap=np.zeros(len(y))
        e=np.zeros(len(y))
        store=np.zeros((len(y),7))
        for k in range(3,len(y)):
                y_cap[k]= -an[0]*y_cap[k-1]-an[1]*y_cap[k-2]-an[2]*y_cap[k-3]+c0*e[k]+cn[0]*e[k-1]+cn[1]*e[k-2]+cn[2]*e[k-3]
                e[k]=y[k]-y_cap[k]
                an=an+2*(mu1)*e[k]*np.array([y_cap[k-1],y_cap[k-2],y_cap[k-3]])
                cn=cn+2*(mu2)*e[k]*np.array([e[k-1],e[k-2],e[k-3]])
                store[k]=[c0,cn[0],cn[1],cn[2],an[0],an[1],an[2]]

        return e
np.savetxt("ada_output.txt",filter_white(eq))
np.savetxt("filter_coeff.txt",filter_coeff(eq))
theta=filter_coeff(eq)
z=5
y=[]
deffrange(start,end,step):
return map(lambda x: x*step, range(int(start*1./step),int(end*1./step)))
for x in frange(0,len(eq)*0.02,0.02):
        y.append(x)
ps=[]
for x in frange(0.5,25,0.5):
        ps.append(x)
ps=np.array(ps)
ps=np.append(np.array([0,0,0,0,0]),ps)
for f in frange(0.5,25,0.5):
        w=2*pi*f
        Power_spectral_density[z]= (((np.std(filter_white(eq)))**2)*0.02*(abs(1+theta[0]*np.exp(-1j*w*0.02)+theta[1]*np.exp(-
2j*w*0.02)+theta[2]*np.exp(-3j*w*0.02)+theta[3]*np.exp(-4j*w*0.02)))**2)/(abs(1-theta[4]*np.exp(-1j*w*0.02)-theta[5]*np.exp(-2j*w*0.02)-
theta[6]*np.exp(-3j*w*0.02)))**2
        z=z+1
pyplot.plot(ps,Power_spectral_density)
pyplot.savefig('PSDF')
pyplot.plot(y,eq)
pyplot.savefig('eq')
pyplot.plot(y,filter(eq))
pyplot.savefig('filtered _Data')
```

3.**Python program for adaptive filter(sixth order) for NLMS algorithm** :

```
from math import tan,pi,sqrt
importmatplotlib.pyplot as pyplot
importnumpy as np

eq= np.loadtxt('sanf.txt')
Power_spectral_density=np.zeros(54)

deffilter_white(dk):

    mu = 0.02

    y  =np.array(dk)
    y_cap  =np.zeros(len(dk))

    phi=np.array([0]*5)
    del_y=np.zeros((len(dk),5))

    b0=1
    b1=0
    b2=0
    a1=0
    a2=0
    theta=np.array([b0,b1,b2,a1,a2])
    e=np.zeros(len(dk))

    defmodsq(x):

            sum=0
            for p in x:
                    sum=p*p+sum
            return sum

    e=np.array([0.00]*(len(dk)))

    for k in range(3,len(dk)):

            y_cap[k]= theta[0]*e[k-1]+theta[1]*e[k-2]+theta[2]*e[k-3]+theta[3]*y_cap[k-1]+theta[4]*y_cap[k-2]
            phi = np.array([e[k-1],e[k-2],e[k-3],y_cap[k-1],y_cap[k-2]])
            del_y[k]=phi+theta[3]*del_y[k-1]+theta[4]*del_y[k-2]
            e[k]=y[k]-y_cap[k]
            if k > 5 :
                    theta=theta+(mu/modsq(del_y[k]))*e[k]*del_y[k]

    return    e

deffilter_coeff(dk):

    mu = 0.02

    y  =np.array(dk)
    y_cap  =np.zeros(len(dk))

    phi=np.array([0]*5)
    del_y=np.zeros((len(dk),5))
    store=np.zeros((len(dk),5))
    b0=1
    b1=0
    b2=0
    a1=0
    a2=0
    theta=np.array([b0,b1,b2,a1,a2])
    e=np.array([0.00]*(len(dk)))

    defmodsq(x):

            sum=0
```

```
                for p in x:
                        sum=p*p+sum
                return sum

        for k in range(3,len(dk)):

                y_cap[k]= theta[0]*e[k-1]+theta[1]*e[k-2]+theta[2]*e[k-3]+theta[3]*y_cap[k-1]+theta[4]*y_cap[k-2]
                phi = np.array([e[k-1],e[k-2],e[k-3],y_cap[k-1],y_cap[k-2]])
                del_y[k]=phi+theta[3]*del_y[k-1]+theta[4]*del_y[k-2]
                e[k]=y[k]-y_cap[k]
                if k >5 :
                        theta=theta+(mu/modsq(del_y[k]))*e[k]*del_y[k]
                store[k]=theta
        return  store[k]

def filter(dk):

        mu = 0.02

        y  =np.array(dk)
        y_cap  =np.zeros(len(dk))

        phi=np.array([0]*5)
        del_y=np.zeros((len(dk),5))
        store=np.zeros((len(dk),5))
        b0=1
        b1=0
        b2=0
        a1=0
        a2=0
        theta=np.array([b0,b1,b2,a1,a2])
        e=np.array([0.00]*(len(dk)))

        defmodsq(x):

                sum=0
                for p in x:
                        sum=p*p+sum
                return sum
        sum=0
        for k in range(3,len(dk)):

                y_cap[k]= theta[0]*e[k-1]+theta[1]*e[k-2]+theta[2]*e[k-3]+theta[3]*y_cap[k-1]+theta[4]*y_cap[k-2]
                phi = np.array([e[k-1],e[k-2],e[k-3],y_cap[k-1],y_cap[k-2]])
                del_y[k]=phi+theta[3]*del_y[k-1]+theta[4]*del_y[k-2]
                e[k]=y[k]-y_cap[k]
                if k > 5:
                        theta=theta+(mu/modsq(del_y[k]))*e[k]*del_y[k]

                store[k]=theta

        returny_cap
np.savetxt("ada_output.txt",filter(eq))
np.savetxt("ada_white.txt",filter_white(eq))
np.savetxt("filter_coeff.txt",filter_coeff(eq))
theta=filter_coeff(eq)
z=5
y=[]
deffrange(start,end,step):
        return map(lambda x: x*step, range(int(start*1./step),int(end*1./step)))

for x in frange(0,len(eq)*0.02,0.02):
        y.append(x)
ps=[]
for x in frange(0.5,25,0.5):
        ps.append(x)
ps=np.array(ps)
ps=np.append(np.array([0,0,0,0,0]),ps)
for f in frange(0.5,25,0.5):
        w=2*pi*f
```

```
        Power_spectral_density[z]= (((np.std(filter_white(eq)))**2)*0.02*(abs(1+theta[0]*np.exp(-1j*w*0.02)+theta[1]*np.exp(-
2j*w*0.02)+theta[2]*np.exp(-3j*w*0.02)))**2)/(abs(1-theta[3]*np.exp(-1j*w*0.02)-theta[4]*np.exp(-2j*w*0.02)))**2
        z=z+1
def integrate(y_vals, h):
        i=1
        total=y_vals[0]+y_vals[-1]
        for y in y_vals[1:-1]:
        if i%2 == 0:
        total+=2*y
        else:
        total+=4*y
        i+=1
return total*(h/3.0)
print integrate(Power_spectral_density,0.1)
np.savetxt("ada_PSDF.txt" ,Power_spectral_density)
pyplot.plot(ps,Power_spectral_density)
pyplot.savefig("PSDF.png")
pyplot.plot(y,eq)
pyplot.savefig("eq.png")

pyplot.plot(y,filter(eq))
pyplot.savefig("filtered _Data.png")
```

Pre-requisites for using the developed programs:

1. Basic knowledge of programming.
2. The Operating system must have configuration to support python 2.7.10 which is an open source development project.
3. Installation guides for Python 2.7.10 is available online.
4. Basic knowledge of python for execution of programs developed in python.
5. Libraries of matplotlib and numpy should be installed.

Step wise method for executing the developed programs:

1. The program in annexure A should be copied on a notepad (preferably Notepad++) file and the extension should be changed to .py (Let the filename be abcd.py)
2. In the same folder as this notepad file, a .txt file with acceleration time history of the earthquake to be studied in a single column must be present(Let the filename be in.txt).
3. Open windows powershell use change drive command 'cd' to reach the folder where the .py and the .txt file as created in step 6 and 7 are kept.
4. Use the command 'python abcd.py'

Results:

1. An image file of .png type with name 'filtered_data.py' will be created this will give the output of filter (in red) and original earthquake (in red).
2. An image file of .png type with name 'PSDF.py' will be created this will give the PSDF of the earthquake.
3. An image file of .png type with name'eq.py' will be created this will give the earthquake acceleration.

REFERENCES

Conference Papers

1. Chen J., Vandewalle J., De Moor B.(1987),"ARMA spectral estimation by an adaptive IIR filter". Proc. of the Int. Conf.on Linear Algebra and Applications, Valencia, Spain.

2. Conte J.P., Peng B.F.(1996), "Non stationary earthquake ground motion model",Paper No. 301, Eleventh World Conference on Earthquake Engineering, Acapulco, Mexico.

3.O'lafsson S.(1992), "The use of ARMA models in strong motion modelling".Proc. of the Tenth World Conference on Earthquake Engineering, Balkoma, Rotterdam, Netherlands.

4. Wang F., Du J.(2013), "A variable Step-Size LMS Adaptive Filtering Algorithm Based on Error Feedback", International Conference on Education Technology and Information System, ICETIS, Sanya, China .

Journals

5. Popescu TH. D.,Demetriu S.(1990), "Analysis and Simulation of Strong Earthquake Ground Motions Using ARMA Models", Automatica, Vol. 26,No. 4, pp. 721-737,Britain.

6. Tom Irvine (2000), "An Introduction to the filtering of digital signals" Revision A.

7. Tom Irvine (2012), "A Digital Recursive Filtering Method for Calculating the Base Input for a Measured Response Acceleration".

Books

8. Adaptive filter Theory by Simon O' Haykin (5th Edition).

Web Sites

9. http://www.strongmotioncenter.org

10. http://www.scedc.caltech.edu

11. http://www.cosmos-eq.org