University of Southampton

CONSTRAINED ADAPTIVE FILTERING AND APPLICATION TO SOUND EQUALISATION

by

Pik Shan Tam, B.Sc. M.Sc.

A thesis submitted for the award of
Doctor of Philosophy

Signal Processing and Control Group
Institute of Sound and Vibration Research
Faculty of Engineering and Applied Science
University of Southampton

December 2003

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND APPLIED SCIENCE

INSTITUTE OF SOUND AND VIBRATION RESEARCH

Doctor of Philosophy

CONSTRAINED ADAPTIVE FILTERING AND APPLICATION TO SOUND EQUALISATION

by Pik Shan Tam

This thesis is concerned with the development of algorithms for constrained adaptive filtering and their application to sound equalisation. Constraints, in particular in the frequency domain, can be useful to increase the robustness by limiting the gain of the adaptive filter. For example in the application of sound equalisation, constraints have been previously used to prevent extreme gains in the equalisation filter, improving spatial robustness. A penalty function formulation was used, so constraints were incorporated into a modified objective function, which was then treated as unconstrained. The stability, convergence time, and fluctuations of such adaptation process are governed by the convergence coefficient and penalty parameter. Recently developed frequency domain LMS algorithm with constraints, solved with the penalty method, use 'trial and error' to find suitable values of the convergence coefficient and penalty parameter, which can cause inaccurate performance and slow convergence. Therefore in this work, we focus on the development of improved constrained adaptive filtering algorithms that are faster converging than the previously proposed ones, and also provide theoretical analysis of convergence for the new algorithms.

In this thesis, a bound is constructed for the value of the convergence coefficient and penalty parameter. This will give conditions for convergence and hence enable more accurate and faster converging adaptive algorithms using the penalty method. The penalty is implemented by using the function $\{\max[c_i, 0]\}$, which is computationally efficient. The theoretical bound on the values for the adaptive LMS and FDLMS algorithms with quadratic constraints is established. The performance is significantly improved by using the new established bound as shown by simulation. In the case of using the second order penalty function $\max(c_i, 0)^2$, the quadratic constraints can be approximated using linear constraints. Using these linear constraints rather than the original quadratic constraints will mean that the new objective function will be quadratic, so that the new bound defined can also be suitable for the second order penalty function. Simulation showed that using linear constraints with a second order penalty function formulation converged to a minimum that was very close to the quadratically constrained case. Increasing the convergence rate of the FDLMS algorithm can also be achieved by incorporating the Bin-normalised FDLMS and the conjugate gradient (CG) algorithms. Efficient constrained BN-FDLMS and CG algorithms implemented in the frequency domain are developed and studied. Their performance showed that they have superior convergence rate compared with the constrained FDLMS.

The new constrained adaptive filtering algorithms studied and developed here were applied to the problem of sound equalisation in an enclosed room. It is also shown that spatial robustness can be further improved by using frequency-dependent constraints, by utilizing the structure of the spatial correlation in a reverberant sound field. A simulation study of the frequency-dependent constraints verifies the applicability of the new algorithms.

CONTENTS

List of figures

List of tables

Acknowledgements

I would like to express my great and sincere thanks to my supervisor, Dr. Boaz Rafaely, for all his help, advice, guidance, patience and encouragement throughout of this research. I would also like to thank Prof. Steve Elliott, who very patiently taught me much about acoustics. Thanks also to Dr. Vic Baston for his helpful and valuable advice in mathematical problems.

The financial support given by my supervisor, Dr. Boaz Rafaely and the Faculty of Engineering and Applied Science is gratefully acknowledged. Thanks also to the ISVR for their support. I have appreciated the support of many other colleagues at ISVR.

This thesis is dedicated to my parents Wah Tung and Sau Ling, my sisters San San and Po Shan, and my brother Wai Shun. Every part of my life has been strengthened and enriched by their tremendous love and support which is never ending. Without their support and encouragement, this would not have been possible.

Finally, I wish to acknowledge Yung for his patience and faith in me throughout my PhD.

Part I:

Introduction and Literature review

Chapter 1

Introduction

1.1 Thesis objectives and background

The objective of this thesis is to develop algorithms for constrained adaptive filtering and to apply these algorithms to an example application, such as the adaptive sound equalisation system. This is achieved by firstly studying unconstrained optimisation algorithms on filtering systems with high-order filters. High-order finite impulse response (FIR) filters are required since acoustic applications often have long impulse responses. It is essential to understand unconstrained optimisation problems before we consider constrained optimisation problems, because in this work, the penalty method will be used to convert the constrained problem into an unconstrained one. The two filtering systems studied first are system identification and prediction system. The two systems are simple to analyse compared with equalisation system, which will be used later on. The systems are designed to minimise a quadratic (mean square error) cost function, and are used to compare the performance of the algorithms. The unconstrained optimisation algorithms considered in this work are steepest descent, conjugate gradient and Newton's methods. The algorithms are then developed in a constrained optimisation framework in which a quadratic cost function is minimised with quadratic constraint. The constraints used here are all quadratic, so that the constrained optimisation problem is solved as a convex programming problem and then a penalty method is applied.

In this thesis, the penalty is implemented by using the function $\max(c_i, 0)^\rho$, where $\rho = 1$ and 2. When $\rho = 2$, the function $\max(c_i, 0)^2$ is called a quadratic penalty function or a second order penalty function, it is a continuous function and provides a smooth error surface. However we will be considering quadratic constraints and therefore high computation complexity and high order terms will be produced during the adaptive processing. We have established a way to approach the constrained optimisation problem with quadratic constraints in order to avoid high-order terms. In the case of the second order penalty function, the quadratic constraints are approximated using linear constraints, so that the objective function produced will now only contain quadratic terms. Also the linear approximations now have a simple formulation like the first order penalty function, this will also have the benefit of a continuous error surface. When $\rho = 1$, the function $\max(c_i, 0)^1$ is called a linear penalty function or a first order penalty function, this function is not a continuous function and may cause large fluctuations due to the discontinuity, however it gives a simple formulation, which facilitated a comprehensive theoretical analysis of the convergence coefficient and penalty parameter. Both penalty functions will be presented and the characteristics will be discussed.

Current algorithms, such as frequency domain least mean square (FDLMS), use a penalty method with trial and error to find suitable values of the convergence coefficient and penalty parameter. A theoretical analysis of the first order penalty function for the constrained FDLMS with quadratic constraints is therefore derived in the adaptive filtering framework; it obtains bounds on the convergence coefficient and penalty parameter, which give conditions for convergence and hence enables more accurate and faster converging adaptive algorithms using the penalty method i.e. improves the performance of the constrained adaptive filter. The theoretical bound is therefore implemented with the magnitude constraint for the FDLMS. The application of sound equalisation is used to ascertain the validity of the new theoretical bound on the convergence coefficient and the penalty parameter. The magnitude constraint on the adaptive filter is applied which could improve the performance of equalisation around the microphone in an enclosure, because large magnitudes in the equalisation filter could cause large increases in sounds away from the equalisation point. Therefore a limit in magnitude is applied to improve spatial robustness.

The LMS algorithm is widely used in adaptive filtering for solving unconstrained problems. Papers have been published which use constrained FDLMS algorithms to adapt FIR filters [Rafaely and Elliott, 2000b]. Rafaely and Elliott (2000b) apply the LMS algorithm to constrained optimisation problems for adaptive filters in the frequency domain, incorporating a second order penalty function. Increasing the convergence rate of the FDLMS algorithm can be done by incorporating other algorithms such as the frequency domain conjugate gradient algorithm (FDCGA) and the Bin-normalised frequency domain LMS algorithm (BN-FDLMS). In this thesis, new constrained adaptive algorithms were developed and studied, using first order penalty function which were incorporated in the FDCGA and BN-FDLMS algorithms, so that an improvement over the FDLMS algorithm can be made when applied to a wider range of practical applications. The algorithms were then applied to our example of sound equalisation system with the aim to improve the spatial robustness and convergence rate of adaptive filtering systems.

In sound equalisation, the equalised system may suffer from distorted sound at high frequencies in an enclosure away from the equalisation microphone. This is because the frequency response of the acoustics path can vary significantly from point to point. The effect of this distortion can be counteracted to some degree by applying a gain limit to the adaptive filter. Furthermore, the magnitude frequency-dependent constraints can be applied to further increase spatial robustness; since low frequency sound in a room is more spatially correlated, higher gains in the equalisation filter are allowed at low frequencies due to the relatively small spatial variability, while tighter gains are applied at high frequencies as the high frequency sound in a room is more uncorrelated. Therefore improved spatial robustness could potentially be achieved by using frequency-dependent constraints, by utilizing the structure of the spatial correlation in reverberant sound fields. The sinc function represents the spatial correlation of the sound in a reverberant room, which is therefore used in the design of the magnitude constraints, such that a tight limit is imposed at high frequencies and a more relaxed limit is imposed at low frequencies. The spatial robustness performance is demonstrated by using magnitude limiting frequency-dependent constraint in the simulation of one dimension duct and has a potential to further improve the spatial robustness in the application to sound equalisation in a room.

1.2 Thesis contributions

The contributions contained in this thesis are as follows:

1. The first contribution is the investigation of the characteristic of first order penalty function in constrained optimisation problems, and the theoretical analysis of the bound on the convergence coefficient and penalty parameter used in a penalty method. Current optimisation algorithms applying the penalty method use trial and error to find suitable values of the both parameters. The accuracy and performance of the constrained adaptive filter is significantly improved by using the established new bounds on the convergence coefficient and penalty parameter.

2. The second contribution is the investigation of the characteristic of second order penalty function in constrained optimisation problems. In the case of using the second order penalty function, the quadratic constraints can be approximated using linear constraints. Using linear constraints with a second order penalty function formulation converged to a minimum that is very close to the quadratically constrained case.

3. The third contribution is the improvement of the convergence rate of the past solution, constrained frequency domain LMS algorithm. It is achieved by incorporating the bin-normalised LMS and the conjugate gradient algorithms. An

efficient constrained bin-normalised LMS and conjugate gradient algorithms with first order penalty function implemented in the frequency domain are derived.

4. The fourth contribution is the study of performance and convergence rate of the conjugate gradient algorithm with a high-order FIR filter; long FIR filters are required since acoustic applications often have very long impulse responses. It was found to have a faster convergence rate compare with the steepest descent method and LMS-type algorithms, but with only minor increase in computation complexity.

5. The fifth contribution is the study of the application of adaptive sound equalisation by means of a simulations study, using the constrained adaptive filtering algorithms and theoretical analysis developed in this thesis.

6. The final contribution is the further improvement of spatial robustness of equalisation around the microphone in a one-dimension duct. Improved spatial robustness can be achieved by using magnitude frequency-dependent constraints on the equalisation filter, by utilizing the structure of the spatial correlation in reverberant sound fields. The frequency-dependent constraint is designed such that a tight limit is imposed at high frequencies and a more relaxed limit is imposed at low frequencies. Using magnitude frequency-dependent constraints can avoid most high frequency peaks and can also achieve good equalisation at low frequency. Constant magnitude constraint can only avoid most high peaks at high frequencies, but give poor equalisation at low frequencies, i.e. improved spatial robustness could potentially be achieved by using frequency-dependent constraints in an enclosure.

Publications

1. Tam, P.S., Rafaely, B., "Efficient Computation of Very Long Optimal FIR Filters", Fifth International Conference on Mathematics on Signal Processing, December 2001, Warwick, U.K.

This paper conducted the conjugate gradient algorithm, which gave fast performance for the high-order FIR filter system. Long FIR filters are required since acoustic systems are often high order. The contribution of this paper to this thesis was an understanding of the performance of the optimisation algorithms with high-order FIR filters.

2. Tam, P.S., Rafaely, B., "Adaptive Sound Equalisation using Constrained Frequency-domain Filters", Active 2002, July 2002, ISVR Southampton, U.K.

This paper presented the development of an adaptive sound equalisation algorithm, which improves both convergence speed and spatial robustness. Improved convergence speed is achieved by employing the conjugate gradient algorithm in the frequency domain. Spatial robustness is achieved by using frequency-dependent constraints. The contributions of this paper are; the convergence rate of past solution of constrained adaptive filtering can be incorporated with constrained CG algorithm, and the potential of the frequency-dependent constraints successfully increase the spatial robustness in application of sound equalisation.

Title of a potential IEEE paper to follow Rafaely and Elliott (2000b) ~ "A computationally efficient frequency-domain LMS algorithm with constraints on the adaptive filter":

3. Tam, P.S., Rafaely, B., "Theoretical analysis and simulation of an improved frequency-domain LMS algorithm with constraints", *IEEE Transaction on Signal Processing.*

This paper will include work on the theoretical bound on the convergence coefficient and penalty parameter for FDLMS algorithm with magnitude constraints, when used with first order penalty function, derived in chapter 7 and also the use of the sound equalisation application as the simulation study to ascertain the validity of the bound on the convergence coefficient.

1.3 Thesis structure

In this section, the structure of the thesis and each chapter is discussed in detail.

This thesis is divided into three parts. In Part I (chapters 1-3), the contents of the thesis are introduced and a literature review of current methods is conducted, providing some background for the later contributions. Optimisation methods that are currently in use are first considered, then adaptive filtering techniques. In Part II (chapters 4-7), several unconstrained optimisation algorithms for two high-order FIR filter design problems are studied, in order to compare their convergence and computational efficiency. In constrained optimisation problems, the penalty method is used and the characteristic of first and second order penalty functions is considered. Ways of simplifying the quadratic constraints with second order penalty function by approximating with linear constraints are discussed. Then in the case of first order penalty function, a theoretical analysis of the convergence coefficient and penalty parameter for the constrained FDLMS with quadratic constraints is derived, for which no definitive theory currently exists. Then the frequency-domain algorithms are developed which have faster converging rate and can incorporate frequency domain constraints directly, these are FDCGA and BN-FDLMS; increasing the convergence rate of the current widely used FDLMS algorithm can be achieved by both FDCGA and BN-FDLMS. In part III (chapters 8-11), various areas of sound equalisation are firstly reviewed, which form the basis of the application in the thesis. The efficient FDCGA and BN-FDLMS algorithms are then demonstrated by performing simulations of constrained optimisation for adaptive sound equalisation and results presented. The validity of the theoretical bound on the convergence coefficient and penalty parameter derived in chapter 7 is ascertained by using simulation of sound equalisation application. A frequency-dependent constraint on the magnitude of the equalisation adaptive filter is then studied, in order to further improve the appeal of the constrained adaptive filtering algorithms and spatial robustness for the application of sound equalisation. Finally conclusions of the thesis and suggestions for future work are discussed.

A detailed description of the contents of each chapter is presented below:

Chapter Two provides a review of unconstrained and constrained optimisation techniques. Firstly, steepest descent algorithm, Newton's method and conjugate gradient algorithm are review in detail. For each algorithm the search direction, update equation, method of implementation and optimal step size are stated. Any advantages and disadvantages of the algorithms are also noted. This review provides a background of the algorithms that will be used in Part II. Other unconstrained algorithms that are not used in this thesis, e.g. quasi-Newton method and Davidon-Fletcher and Powell method, are briefly discussed. Also reviewed is convexity, its importance in guaranteeing a unique solution to the optimisation problems in the rest of the thesis. Penalty and barrier methods for converting a constrained optimisation problem into an unconstrained problem are reviewed in detail. These methods are used in part II for constrained FIR filtering problems. Other constrained algorithms, e.g. semidefinite programming, quadratic programming and sequential quadratic programming are also briefly reviewed.

Chapter Three begins the review of the signal processing theory that is applicable to the thesis objectives. Wiener filters are first reviewed, then the choice of mean square error as cost function for minimisation and the advantages of adaptive filters over fixed filters. The LMS algorithm is introduced, along with its block and frequency domain formulations. This algorithm is widely used in current applications in signal processing. The CGA is then introduced, along with its block and frequency domain formulations. Then frequency domain bin-normalised LMS is presented. The efficiency of these algorithms is compared and further studied in Part II.

Chapter Four is the beginning of Part II, which starts the study of the algorithms in the thesis. In the first section, reformulation of the steepest descent, Newton's and conjugate gradient algorithms are performed, for signal processing application to two high-order FIR filter design problems: system identification and prediction systems. The two design systems are used to compare the performance of the algorithms in terms of minimising the MSE and are simple to analyse compared with equalisation system, which is used later on.

Chapter Five is concerned with penalty function formulation and investigates way to approach the minimum solution for constrained optimisation. The characteristic of first order and second order penalty functions are discussed first to see how they perform in constrained optimisation problem. We then show through simulation that the performance of the first and second order penalty functions. Then update equations for the steepest descent and conjugate gradient algorithm are derived including the penalty terms in respect of a quadratic (magnitude limiting) frequency domain constraint. Linear approximation of quadratic constraints is then constructed for the case of using the second order penalty function. The accuracy of the approximation is then discussed.

Chapter Six discusses the implementation of the block conjugate gradient and bin-normalised LMS algorithms in the frequency domain with frequency domain constraints for adaptive sound equalisation. Data is transformed to the frequency domain to reap the benefits of faster convergence rate. In this chapter, the first order penalty function is used to derive the new constrained adaptive filtering algorithms.

Chapter Seven sets out a theoretical analysis of the convergence coefficient and penalty parameter used in the penalty method of constrained optimisation, which is to control the convergence step size and the stability of the performance, and to weight the amount of penalty imposed for violating the constraints respectively. Currently, trial and error is used, as no definitive theory exists for selection of appropriate values, or sequence of values, of penalty parameters used. This chapter will detail the theoretical analysis of set bounds for suitable values of the convergence coefficient and the penalty parameter, thus establishing a new framework within which more efficient values of the parameters may be chosen.

Chapter Eight begins Part III which introduces the application of the thesis. It provides the background for sound equalisation, in particular focusing on equalisation in an enclosed room. Single and one-dimensional duct adaptive FIR filter equalisations are also discussed. The knowledge acquired in this chapter along with the new algorithms in part II is applied to the practical testing of sound equalisation application in later chapters.

Chapter Nine is concerned with the application of the adaptive sound equalisation. We first give details of the sound equalisation problem in an enclosure. Using the simulation to ascertain the validity of the theoretical stability bound on the convergence coefficient; it shows that the theoretical bound is accurate. We then show that increasing the convergence rate of the FDLMS algorithm can be done by incorporating the constrained frequency domain bin-normalised LMS and constrained frequency domain conjugate gradient algorithms.

Chapter Ten concerns the utilisation of magnitude limiting frequency domain constraints to increase the spatial robustness of the algorithm. This is achieved by using frequency-dependent constraints to provide tight limit gains of the filter at high frequencies, and a lose limit at low frequencies. Two simulations are presented: the first is the sound equalisation with measured enclosure data, the second is the sound equalisation with modified duct data. Details of the simulations and the performance of spatial robustness are presented and discussed.

Chapter Eleven is the conclusion of the thesis, which includes the contributions achieved and suggestions for future work.

Chapter 2

Optimisation

2.1 Introduction

This chapter provides the background in optimisation required for the formulation and solution of the constrained adaptive filtering problems studied in this thesis. The development of constrained adaptive filtering system discussed here involves a formulation design objective that defined from the system, then the optimal performance of an adaptive system is defined with respect to the objective function. This objective can be formulated as an unconstrained or a constrained optimisation problem. The optimisation methods discussed here are applied for the design of both fixed and adaptive filtering for acoustic problems later in this thesis. Various methods for constrained optimisation problems are presented. The constrained optimisation problem can be formulated as an unconstrained problem using penalty function, and then can be solved by using unconstrained optimisation methods. Penalty method is used here due to computation simplicity which fits real-time adaptive filtering. Therefore penalty method and unconstrained optimisation methods are presented in more detail in this chapter. Unconstrained optimisation methods presented here in detail are: steepest descent method, Newton's method and conjugate gradient method. The performance of these methods will be compared for two systems in chapter 4 and then proceed to the adaptive filtering application in later chapters. Constrained optimisation methods presented here in detail are: convex programming and penalty methods. Constraints considered in this thesis are all quadratic constraints so that constrained optimisation problems are solved as convex programming problems. Other unconstrained and constrained optimisation methods are briefly discussed which could also be used to optimisation problems.

2.2 Unconstrained optimisation problem formulation

One of the essential problems in optimisation is that of developing computational procedures for finding minimum point of an objective function. In this section we introduce the construction of algorithms for unconstrained optimisation, that is, the development of methods to find and minimise a function of n variables, where the function is assumed to be continuous. The mathematical problem becomes that of minimising the function f of n independent variables $x_1, \ldots, x_n$. It can be written in the vector form, that is $f(x_1, \ldots, x_n) = f(\boldsymbol{x})$.

When exact formulas are not available for solving the $\min f(\boldsymbol{x})$, or when such methods involve extensive effort, we can turn to numerical techniques from calculus for approximating the solution we seek. All of the numerical methods for unconstrained optimisation which are described in here are iterative; they will converge to the minimum point $\boldsymbol{x}^*$ under certain conditions on a starting point $\boldsymbol{x}_0$ and on f. In this chapter, we introduce the terminology of optimisation methods and the ways in which problems and their solutions are formulated and classified.

Numerical techniques from calculus can be used to approximate the solution of the general optimisation problem given by

$$\min_{x} f(\boldsymbol{x}) \tag{2.1}$$

where f is the objective function. In the case of this general unconstrained optimisation problem is of the form: $\min\limits_{x} f(\boldsymbol{x})$ with $\boldsymbol{x} \in \mathbf{R}^n$ where $f : \mathbf{R}^n \to \mathbf{R}^1$ is a given objective function and there are no constraints on the variables. Note that $\mathbf{R}^n$ denote the n-dimensional space of real numbers.

Many methods for unconstrained or constrained minimisation apply to a quadratic objective function with positive definite Hessian matrix $\boldsymbol{G}$, that is $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{G}\boldsymbol{x} > 0$, $\forall \boldsymbol{x} \in \mathbf{R}^n$, $\boldsymbol{x} \neq \mathbf{0}$. The quadratic objective functions considered in this thesis will have positive semidefinite $n \times n$ Hessian matrix $\boldsymbol{G}$, that is $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{G}\boldsymbol{x} \geq 0$, $\forall \boldsymbol{x} \in \mathbf{R}^n$, $\boldsymbol{x} \neq \mathbf{0}$. Quadratic forms arise in a variety of signal processing applications where squared error terms are employed. We therefore study optimisation methods, which are designed to minimise a quadratic cost function and then developing improved algorithms in which a quadratic cost function is minimised with various quadratic constraints.

The general quadratic form of f in the variables $x_1, \ldots, x_n$ that corresponds to the symmetric $n \times n$ matrix $\boldsymbol{G}$ is the function f: $\mathbf{R}^n \to \mathbf{R}^1$ defined by

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{G}\,\boldsymbol{x} + \boldsymbol{b}^{\mathrm{T}}\boldsymbol{x} + c \tag{2.2}$$

where $\boldsymbol{x}$ denotes the column vector with entries $x_1, \ldots, x_n$. The vector of the first derivative and the matrix of second derivatives, the Hessian matrices $\boldsymbol{G}$, of the function f are written as:

$$\nabla f = \frac{\partial f}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \cdots & \frac{\partial f}{\partial x_n} \end{bmatrix}^{\mathrm{T}}, \qquad \boldsymbol{G} = \nabla^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \tag{2.3}$$

From Calculus, if $f(\boldsymbol{x})$ has a minimum at some point $\boldsymbol{x}$, then all the partial derivatives of f must be zero at that point. Calculating the partial derivatives of f with respect to each $\boldsymbol{x}_i$, and setting each partial derivative to zero results in the linear system $\boldsymbol{Gx} + \boldsymbol{b} = \boldsymbol{0}$ or $\boldsymbol{Gx} = -\boldsymbol{b}$. $\boldsymbol{Gx} + \boldsymbol{b}$ is therefore the gradient of $f(\boldsymbol{x})$ written as $\nabla f(\boldsymbol{x}) = \boldsymbol{Gx} + \boldsymbol{b}$. Therefore, solving the linear system $\boldsymbol{Gx} = -\boldsymbol{b}$ is equivalent to minimising the quadratic function above, if $\boldsymbol{G}$ is positive definite. The necessary conditions for a local minimum $\boldsymbol{x}^*$ in $\mathbf{R}^n$ are that $\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}$ and $\nabla^2 f(\boldsymbol{x}^*)$ is positive semidefinite. The sufficient conditions for a local minimum $\boldsymbol{x}^*$ are $\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}$ and $\nabla^2 f(\boldsymbol{x}^*)$ is positive definite, i.e. if $\nabla f = \boldsymbol{0}$ at $\boldsymbol{x}^*$ and $\boldsymbol{G}$ is positive definite at $\boldsymbol{x}^*$, then $\boldsymbol{x}^*$ is a local minimum point of the function $f(\boldsymbol{x})$. [Fletcher, 1987, Gill et al, 1974 and Walsh, 1975]

Optimisation algorithms can be used to find this minimum. In all the iterative processes considered here, we start with points $\boldsymbol{x}_0$ and proceed to the minimum $\boldsymbol{x}^*$, by a succession of steps. At each step of the algorithm, the search direction $\boldsymbol{p}$ is updated, we are interested in finding the rate of change of the objective function with respect to the step size along the direction $\boldsymbol{p}_k$, away from a point $\boldsymbol{x}_k$, so that the next search point is given by

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k \tag{2.4}$$

where α_k is the step size at kth step, $\boldsymbol{p}_k$ are usually defined in terms of $\boldsymbol{g}_k$, where $\boldsymbol{g}$ is the gradient vector $\nabla f(\boldsymbol{x})$, defined by $\boldsymbol{g}_k = \nabla f_k = \frac{\partial f}{\partial \boldsymbol{x}_k}$. The negative of the gradient vector denotes the direction of steepest descent. It can be shown that the optimal α will produce a minimum of $f(\boldsymbol{x})$ along the search direction, and in this case, the gradient at the new point $\nabla f(\boldsymbol{x}_{k+1})$ is orthogonal to the search direction $\boldsymbol{p}_k$ (see Greig,1980 and Wolfe, 1978).

2.3 Steepest descent method

This section reviews the steepest descent method, which is simple to implement and widely used in adaptive filtering applications. It has a low computational complexity and, due its formulation, continuously decreases the value of the objective function. Later in this thesis the performance of the steepest descent method will be studied and compared with other algorithms presented in this chapter.

In this method we start from an initial point x_0 and iteratively move along the steepest descent direction of the negative gradient until the optimum point is found. The steepest descent method can be summarised by the following steps: [Greig, 1980]

1. Start with an arbitrary initial point x_0.
2. Find the search direction $p_k = -\nabla f_k \equiv -\nabla f(x_k) \equiv - g_k$.
3. Determine the optimal step length α_k^* in the direction p_k and set $x_{k+1} = x_k + \alpha_k^* p_k$.
4. Continue to (2) until the minimum is reached or stop by some stopping criteria (change in f, x, etc).

The minimum along each search direction of steepest descent method is located, and the method would perform as indicated in figure 2.1 for a function of two variables. It will be noted that successive directions are orthogonal [Wolfe, 1978] because at the minimum along any search direction, the direction of steepest descent at the next step is at a right angle to this search direction. It can also be seen that for functions of two variables, the search uses only two distinctive directions, which are determined completely by the gradient at the initial point.

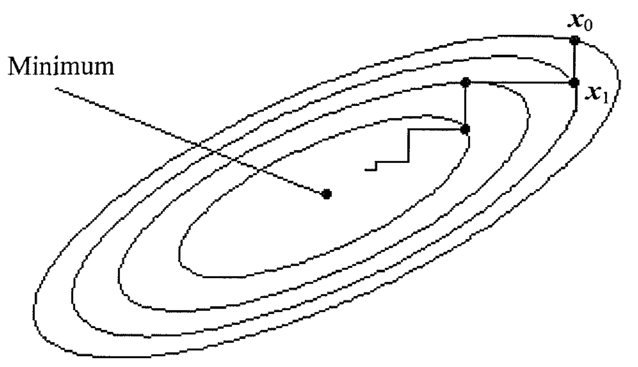


Figure 2.1: Method of steepest descent for function of two variables.

Now it remains to establish a formula for the optimal step size α_k. The gradient of (2.2) is given by $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{G}\boldsymbol{x} + \boldsymbol{b}$.

Then,
$$\begin{aligned} \boldsymbol{g}_{k+1} &= \boldsymbol{G}\boldsymbol{x}_{k+1} + \boldsymbol{b} \\ \Rightarrow \quad &= \boldsymbol{G}(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k) + \boldsymbol{b} \\ \Rightarrow \quad &= \boldsymbol{G}\boldsymbol{x}_k + \boldsymbol{b} + \boldsymbol{G}\alpha_k \boldsymbol{p}_k \\ \Rightarrow \quad &= \boldsymbol{g}_k + \alpha_k \boldsymbol{G}\boldsymbol{p}_k \end{aligned} \tag{2.5}$$

Because the search direction $\boldsymbol{p}_k$ is perpendicular to the gradient vector $\nabla f(\boldsymbol{x}_{k+1})$, i.e. $\nabla f(\boldsymbol{x}_{k+1})^{\mathrm{T}} \boldsymbol{p}_k = 0$, so from (2.5) it follows that

$$(\boldsymbol{g}_k + \alpha_k \boldsymbol{G}\boldsymbol{p}_k)^{\mathrm{T}} \boldsymbol{p}_k = 0$$

$$\Rightarrow \quad \alpha_k = -\frac{\boldsymbol{p}_k^{\mathrm{T}} \boldsymbol{g}_k}{\boldsymbol{p}_k^{\mathrm{T}} \boldsymbol{G}\boldsymbol{p}_k} \tag{2.6}$$

So, if $f(\boldsymbol{x})$ is a positive definite quadratic function, the step length of steepest descent α_k is given by equation (2.6). Therefore, steepest descent method searches for a minimum $f(\boldsymbol{x}_{k+1})$ and selects the corresponding α_k as the step length. The variable α_k is calculated for each iteration of the algorithms by using (2.6) or, a fixed α can be used as an alternative but at the expense of slowing convergence.

The method of steepest descent is easy to program. It will smoothly converge to the local minimum whatever the starting point, but it often converges slowly and therefore requires a great deal of time to do so. There are two main reasons for its slow convergence, these are 1) the method may spend time minimising f along parallel or nearly parallel search directions to the contour lines. The path becomes horizontal, and a major speed limitation lies in the fact that each step is necessarily perpendicular to the previous one [Wolfe, 1978]; 2) at each iteration we only search along a single line for the minimum of f. By doing this we use none of the information regarding previous search directions. It would be preferable to find methods that can reach the minimum quicker although the steepest descent is computational simple. In section 2.5, the conjugate gradient method will be discussed; this method searches over the plane rather just a line and also uses the information of previous steps. Before looking at the conjugate gradient method, let's look at a method that has a very rapid convergence. The next section reviews the Newton's method that offers a faster convergence.

2.4 Newton's method

In contrast to the steepest descent method, Newton's method has a very rapid convergence: it can reach the minimum in only one step. Later in this thesis, Newton's method will be compared, with steepest descent and conjugate gradient methods, with high order FIR filters.

Considering the second order Taylor expansion [Cooper,1970 and Walsh, 1975] of the objective function $f(\boldsymbol{x})$ about the minimum $\boldsymbol{x}^*$:

$$f(\boldsymbol{x}) = f(\boldsymbol{x}^* + \boldsymbol{h})$$
$$\cong f(\boldsymbol{x}^*) + \sum_{i=1}^{N} h_i \left[\frac{\partial f}{\partial x_i}\right]_{x^*} + \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} h_i h_j \left[\frac{\partial^2 f}{\partial x_i \partial x_j}\right]_{x^*} + \text{ higher order terms}$$

Neglecting the higher order terms, by assuming a quadratic function:

$$f(\boldsymbol{x}) = f(\boldsymbol{x}^*) + \nabla f^{\mathrm{T}}(\boldsymbol{x}^*)\boldsymbol{h} + \frac{1}{2}\boldsymbol{h}^{\mathrm{T}}\nabla^2 f(\boldsymbol{x}^*)\boldsymbol{h} \tag{2.7}$$

where $\boldsymbol{x} = \boldsymbol{x}^* + \boldsymbol{h}$ and $\boldsymbol{h} = (h_1, \ldots, h_n)$, and the derivatives are all evaluated at the minimum $\boldsymbol{x}^*$.

Differentiation of the above equation gives

$$\left.\frac{\partial f}{\partial x_l}\right|_{x} = \left[\frac{\partial f}{\partial x_l}\right]_{x_l^*} + \sum_{i=1}^{N} h_i \left[\frac{\partial^2 f}{\partial x_i \partial x_l}\right]_{x_l^*}, \qquad l = 1, 2, \ldots, n. \tag{2.8}$$

Since $\boldsymbol{g}(\boldsymbol{x}^*) = 0$ at the minimum, the l^{th} component, g_l, of the gradient vector $\boldsymbol{g}$ at the current point x_l satisfies

$$g_l = \frac{\partial f}{\partial x_l} = -\sum_{i=1}^{N} h_i \left[\frac{\partial^2 f}{\partial x_i \partial x_l}\right]_{x_l^*}, \qquad l = 1, 2, \ldots, n. \tag{2.9}$$

The minimum $\boldsymbol{x}^*$ is therefore obtained from the current point $\boldsymbol{x}$ by the move $\boldsymbol{x}^* = \boldsymbol{x} - \boldsymbol{h}$, where the components of the step $\boldsymbol{h}$ are determined by solving the n simultaneous linear equations above. Defining $\boldsymbol{G}$ to be the matrix with elements $\boldsymbol{G}_{ij} = \dfrac{\partial^2 f}{\partial x_i \partial x_j}$, the equations above can be written $\boldsymbol{g} = -\boldsymbol{Gh}$, $\Rightarrow \boldsymbol{h} = -\boldsymbol{G}^{-1}\boldsymbol{g}$, and $\Rightarrow \boldsymbol{x}^* = \boldsymbol{x} + \boldsymbol{G}^{-1}\boldsymbol{g}$.

Generalising the above equation, the iterative rule can be written:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{G}^{-1}\boldsymbol{g}_k \tag{2.10}$$

where the gradient vector $\boldsymbol{g}_k$ is evaluated at $\boldsymbol{x}_k$. If the objective function is quadratic and positive definite, $\boldsymbol{G}$ is constant, so that its value at the minimum is known. Consequently, the minimum will be reached in one iteration. [Fletcher, 1987]

Newton's method is known to find the roots of quadratic equations in one iteration. Although the method is very efficient, there are a number of practical issues that must be taken into account. First of all, Newton's method requires the second derivatives $\boldsymbol{G}$ that are often difficult to compute and the high-order roots matrix inversion $\boldsymbol{G}^{-1}$ can cause convergence to be slow. Secondly, Newton's method also requires that the derivative be calculated directly. In cases where the derivative is approximated by taking the slope of two points on the function, the method becomes inefficient compared to other methods.

2.5 Conjugate gradient method

The convergence of the steepest descent method described in section 2.3 can be improved by modifying it into the conjugate gradient method. This method is based on the idea that the convergence to the solution could be accelerated if f is minimised over the plane that contains all previous search directions, instead of minimising f over just the line that points in the descent gradient direction [Roberts, 1973]. The simple form of conjugate gradient method covered here is a sequence of steps. In each step the minimum is found in the plane given by two vectors: the gradient vector and the vector of the previous step. It is expected to converge towards the solution more quickly than the steepest descent method, especially when the starting point is far from the solution because iterations are not limited to perpendicular searches. Also conjugate gradient method does not require matrix $\boldsymbol{G}$ but only uses vectors, so it is more appropriate for the efficient real-time adaptive filtering rather than Newton's method. The aim of the conjugate gradient method is to try to associate conjugate properties with the steepest descent method in an attempt to increase convergence speed.

The conjugate gradient is a class of methods in which the minimum $\boldsymbol{x}^*$ of a quadratic function can be determined in at most n iteration [Fletcher et al, 1987, Greig, 1980 and Wolfe, 1978]. Methods in this class are known as quadratically convergent. Some properties of conjugate directions are now introduced in order to develop the conjugate gradient method.

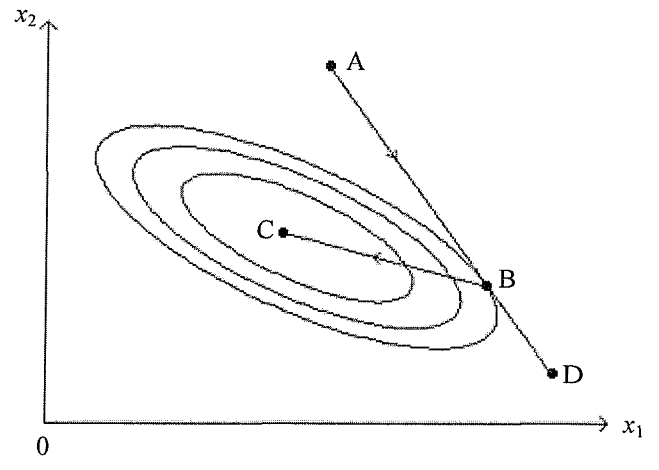


Figure 2.2: Conjugate directions.

Consider the problem of finding the minimum value of the quadratic function (2.2). Suppose we have a function in two dimensions as in figure 2.2, then we search for a minimum from the initial point A in the direction AD, and this minimum occurs at B. From B the next search direction points to C which is the optimal point of the 2-D plane. We say that the direction BC is conjugate to the direction AD. The idea of conjugate directions is extended to n dimensions by means of the following definition. [Fletcher, 1987, Greig, 1980 and Wolfe, 1978]

Definition: A set of direction vectors $\boldsymbol{p}_i$ where $i = 1, 2, \ldots, k$, $k \leq n$, are said to be conjugate with respect to the matrix $\boldsymbol{G}$ if and only if they possess the property $\boldsymbol{p}_i^{\mathrm{T}} \boldsymbol{G} \boldsymbol{p}_j = 0 \;\; (i \neq j)$.

The more important properties of conjugate directions are established in the following theorems [Fletcher, 1987, Greig, 1980 and Wolfe, 1978].

Theorem 2.1: Assuming $\boldsymbol{G}$ is symmetric positive definite, then the $\boldsymbol{G}$-conjugate vectors $\boldsymbol{p}_i$ are linearly independent. (see Appendix for proof)

Note: Theorem 2.1 would imply that the solution of the linear system $\boldsymbol{Gx} = -\boldsymbol{b}$ would be obtained in no more than n steps, where n is the number of dimensional.

Theorem 2.2: Let $\boldsymbol{x}_k$ and $\boldsymbol{x}_{k+1}$ be current points in a minimisation algorithm when the objective function $f(\boldsymbol{x})$ is quadratic. If

(a) $\boldsymbol{x}_k$ minimises $f(\boldsymbol{x})$ in the direction $\boldsymbol{p}_i$, for $i = 0, 1, \ldots, k$;

(b) x_{k+1} minimises $f(x)$ in the direction p_j for $j = 0, 1, \ldots, k+1$;

(c) p_i and p_j are conjugate directions;

then g_{k+1} is orthogonal to all the preceding descent directions p_i.

(see Appendix for proof)

The above theorem 2.2 states that under given conditions, the gradient vector at x_{k+1} is orthogonal to all the proceeding descent directions p_i.

Theorem 2.3: If (1) $x_1 \in \mathbf{R}^n$ is arbitrary;

(2) x_{k+1} is the minimum of f along the line $x_k + \alpha_k p_k$;

(3) $p_0, \ldots, p_k$ are G-conjugate;

(4) $x_{k+1} = x_k + \alpha_k p_k$.

then $x_m = x^*$ for some $m \le n$. (see Appendix for proof)

The above theorems can now be used to formulate the conjugate gradient method. Consider the problem of minimising the quadratic function (2.2), where G is positive definite and symmetric, by successive linear searches along G-conjugate directions. The initial step is in the direction of steepest descent:

$$p_0 = - g_0 = - Gx_0 - b \tag{2.11}$$

$$x_1 = x_0 + \alpha_0^* p_0 \tag{2.12}$$

where α_0^* is the minimising step length in the direction p_0, so that

$$p_0^{\mathrm{T}} \left(\nabla f \,|_{x_1} \right) = 0 \tag{2.13}$$

Therefore, equation (2.13) can be expanded as

$$p_0^{\mathrm{T}} [G(x_0 + \alpha_0^* p_0) + b] = 0 \tag{2.14}$$

$$\Rightarrow \qquad \alpha_0^* = \frac{- p_0^{\mathrm{T}} (Gx_0 + b)}{p_0^{\mathrm{T}} Gp_0} = - \frac{p_0^{\mathrm{T}} g_0}{p_0^{\mathrm{T}} Gp_0} \tag{2.15}$$

Now, expressing the second search direction as a linear combination of $\boldsymbol{p}_0$ and - $\boldsymbol{g}_1$:

$$\boldsymbol{p}_1 = -\boldsymbol{g}_1 + \beta_0 \boldsymbol{p}_0 \tag{2.16}$$

where β_0 is to chosen so as to make $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$ conjugate. This can be illustrated geometrically as shown in figure 2.3 below, [Greig, 1980].

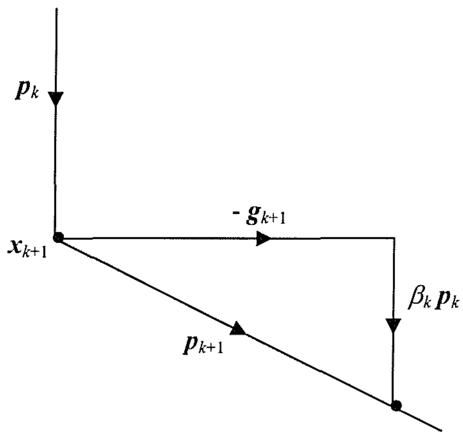


Figure 2.3: Conjugate gradient directions.

This requires that

$$\boldsymbol{p}_0^{\mathrm{T}} \boldsymbol{G} \boldsymbol{p}_1 = 0 \tag{2.17}$$

Substituting equation (2.16) into equation (2.17) implies that

$$\boldsymbol{p}_0^{\mathrm{T}} \boldsymbol{G}(-\boldsymbol{g}_1 + \beta_0 \boldsymbol{p}_0) = 0 \tag{2.18}$$

Substituting (2.12) into (2.18) gives that:

$$-\frac{(\boldsymbol{x}_1 - \boldsymbol{x}_0)^{\mathrm{T}}}{\alpha_0^*} \boldsymbol{G}(-\boldsymbol{g}_1 + \beta_0 \boldsymbol{p}_0) = 0 \tag{2.19}$$

The difference of the gradients $\boldsymbol{g}_1 - \boldsymbol{g}_0$ can be expressed as

$$(\boldsymbol{g}_1 - \boldsymbol{g}_0) = (\boldsymbol{G}\boldsymbol{x}_1 + \boldsymbol{b}) - (\boldsymbol{G}\boldsymbol{x}_0 + \boldsymbol{b}) = \boldsymbol{G}\,(\boldsymbol{x}_1 - \boldsymbol{x}_0) \tag{2.20}$$

Therefore (2.19) becomes

$$(\boldsymbol{g}_1 - \boldsymbol{g}_0)^{\mathrm{T}} (\boldsymbol{g}_1 - \beta_0 \boldsymbol{p}_0) = 0 \tag{2.21}$$

$$\Rightarrow \boldsymbol{g}_1^{\mathrm{T}} \boldsymbol{g}_1 - \boldsymbol{g}_0^{\mathrm{T}} \boldsymbol{g}_1 - \beta_0 \boldsymbol{g}_1^{\mathrm{T}} \boldsymbol{p}_0 + \beta_0 \boldsymbol{g}_0^{\mathrm{T}} \boldsymbol{p}_0 = 0 \tag{2.22}$$

But, since $\boldsymbol{g}_0^{\mathrm{T}} \boldsymbol{g}_1 = -\boldsymbol{p}_0^{\mathrm{T}} \boldsymbol{g}_1 = 0$ by (2.11) and (2.13), so (2.22) can be rearranged as:

$$\beta_0 = -\frac{\boldsymbol{g}_1^{\mathrm{T}} \boldsymbol{g}_1}{\boldsymbol{g}_0^{\mathrm{T}} \boldsymbol{p}_0} = \frac{\boldsymbol{g}_1^{\mathrm{T}} \boldsymbol{g}_1}{\boldsymbol{g}_0^{\mathrm{T}} \boldsymbol{g}_0} \tag{2.23}$$

Next, consider the third search direction as a linear combination of $\boldsymbol{p}_0$, $\boldsymbol{p}_1$, and $-\boldsymbol{g}_2$ as

$$\boldsymbol{p}_2 = -\boldsymbol{g}_2 + \beta_1 \boldsymbol{p}_1 + \delta_0 \boldsymbol{p}_0 \tag{2.24}$$

where the values of β_1 and δ_0 can be found by making $\boldsymbol{p}_2$ conjugate to $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$. By using the condition $\boldsymbol{p}_0^{\mathrm{T}} \boldsymbol{G} \boldsymbol{p}_2 = 0$, the value of δ_0 can be found to be zero. When the condition $\boldsymbol{p}_1^{\mathrm{T}} \boldsymbol{G} \boldsymbol{p}_2 = 0$ is used, the value of β_1 can be obtained as

$$\beta_1 = \frac{\boldsymbol{g}_2^{\mathrm{T}} \boldsymbol{g}_2}{\boldsymbol{g}_1^{\mathrm{T}} \boldsymbol{g}_1} \tag{2.25}$$

so that equation (2.24) becomes

$$\boldsymbol{p}_2 = -\boldsymbol{g}_2 + \beta_1 \boldsymbol{p}_1 \tag{2.26}$$

where β_1 is given by (2.25). In fact, equation (2.26) can be generalised as

$$\boldsymbol{p}_k = -\boldsymbol{g}_k + \beta_{k-1} \boldsymbol{p}_{k-1} \tag{2.27}$$

where

$$\beta_{k-1} = \frac{\boldsymbol{g}_k^{\mathrm{T}} \boldsymbol{g}_k}{\boldsymbol{g}_{k-1}^{\mathrm{T}} \boldsymbol{g}_{k-1}} \tag{2.28}$$

Equations (2.27) and (2.28) define the search direction used in the conjugate gradient method. By equation (2.27), we can define the following fact

$$\boldsymbol{p}_k = -\boldsymbol{g}_k + \beta_{k-1} \boldsymbol{p}_{k-1} \quad \Rightarrow \quad \boldsymbol{g}_k = -\boldsymbol{p}_k + \beta_{k-1} \boldsymbol{p}_{k-1}$$

$$\Rightarrow \quad \boldsymbol{g}_i = -\boldsymbol{p}_i + \beta_{i-1} \boldsymbol{p}_{i-1}$$

Multiplying the second equation by $\boldsymbol{g}_{k+1}^{\mathrm{T}}$, we have

$$\boldsymbol{g}_{k+1}^{\mathrm{T}}\boldsymbol{g}_i = -\,\boldsymbol{g}_{k+1}^{\mathrm{T}}\boldsymbol{p}_i + \beta_{i\text{-}1}\,\boldsymbol{g}_{k+1}^{\mathrm{T}}\boldsymbol{p}_{i\text{-}1}$$

$$\Rightarrow \qquad \boldsymbol{g}_{k+1}^{\mathrm{T}}\boldsymbol{g}_i = 0 \tag{2.29}$$

Hence, each successive gradient vector is orthogonal to all the preceding ones.

The conjugate gradient algorithm is a method for solving systems of linear equations derived from the stationary conditions of a quadratic function. Since the directions $\boldsymbol{p}_k$ used in this method are $\boldsymbol{G}$-conjugate, the process will converge in n iterations or less for a quadratic function [Wolfe, 1978]. The conjugate gradient algorithm can be summarised by the following relations.

Conjugate gradient algorithm (for $\boldsymbol{G}$ symmetric and positive definite) [Wolfe, 1978]:

(1) $\boldsymbol{x}_0 \in \mathbf{R}^n$ is arbitrary;

(2) $f(\boldsymbol{x})$ is of the form $\frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{G}\boldsymbol{x} + \boldsymbol{b}^{\mathrm{T}}\boldsymbol{x} + c$;

(3) $\boldsymbol{x}_k$ ($k = 0, 1, \ldots, m$+1) are generated by conjugate direction algorithm;

(4) the directions $\boldsymbol{p}_k$ ($k = 0, 1, \ldots, m\text{-}1 < n$) are generated from

$$\boldsymbol{p}_0 = -\boldsymbol{g}_0;$$

$$\boldsymbol{p}_k = -\boldsymbol{g}_k + \beta_{k\text{-}1}\,\boldsymbol{p}_{k\text{-}1}\ (k \geq 1), \text{ where } \beta_{k-1} = \frac{\boldsymbol{g}_k^{\mathrm{T}}\boldsymbol{g}_k}{\boldsymbol{g}_{k-1}^{\mathrm{T}}\boldsymbol{g}_{k-1}};$$

where $\boldsymbol{g}_k < \boldsymbol{g}_{k\text{-}1}$ and $\beta_{k\text{-}1}$ is to chosen so as to make $\boldsymbol{p}_k$ conjugate to previous search directions. We note from the above that:

(a) the direction vectors $\boldsymbol{p}_k$ are $\boldsymbol{G}$-conjugate;

(b) $\boldsymbol{g}_{k+1}^{\mathrm{T}}\boldsymbol{p}_i = 0$, $i = 0, 1, \ldots, k$;

(c) $\boldsymbol{g}_{k+1}^{\mathrm{T}}\boldsymbol{g}_i = 0$, $i = 0, 1, \ldots, k$.

One particular advantage of conjugate gradient algorithm is the form of $\boldsymbol{p}_k = -\boldsymbol{g}_k + \beta_{k\text{-}1}\,\boldsymbol{p}_{k\text{-}1}$ which requires no matrix operations to form $\boldsymbol{p}_{k\text{-}1}$. Therefore, for large problems they are usually more computationally efficient than the methods that require matrix operations. Only the function itself and gradient information are required in minimising the function. No approximation to the inverse of the second derivative matrix is needed.

2.6 Other methods for solving unconstrained optimisation problems

In sections 2.2-2.5 we have reviewed the unconstrained optimisation problems. The methods covered so far will be used later in the thesis for constrained adaptive filtering problems. However, there are many other unconstrained methods that can also be used. This section presents a brief review of other unconstrained methods such as Quasi-Newton method and Davidon-Fletcher-Powell method. Applying these methods and investigating other unconstrained optimisation methods to the constrained adaptive filtering problems is suggested for future work.

2.6.1 Quasi-Newton method

The aim of the Quasi-Newton method is to use the best features of the Newton and steepest descent method and avoid the worst. Newton's method converges rapidly but is numerically unreliable; it also requires second derivatives and a matrix inversion. The steepest descent method, however, is slow but continuously decreases the function value. Quasi-Newton method is one particular approach to practical applications, which is to use steepest descent method initially, and then use Newton's method as the minimum is approached.

It may be possible to improve computational efficiency by using a method that starts like a steepest descent method and finishes like a Newton's method but only uses first derivative. Thus, the search direction should be slowly deflected from the steepest descent direction to the Newton direction as the iterations proceed.

Now let us consider

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \boldsymbol{g}_k \qquad \text{(Steepest descent method)}$$

and

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \boldsymbol{G}^{-1} \boldsymbol{g}_k \qquad \text{(Newton's method)}$$

So, a compromise between $-\boldsymbol{g}_k$ and $-\boldsymbol{G}^{-1}\boldsymbol{g}_k$ is chosen as $-\boldsymbol{H}_k \boldsymbol{g}_k$, then we have

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \boldsymbol{H}_k \boldsymbol{g}_k \qquad \text{(Quasi-Newton method)}$$

where $\boldsymbol{H}_k$ is the kth approximation to the inverse of the matrix of second derivatives $\boldsymbol{G}^{-1}$.

Quasi-Newton method can be classified as inexact-Newton methods or approximate-Newton methods. The method was further suggested by Davidon (1959), and developed by Fletcher and Powell later on (1963). Many other authors, e.g. Broyden, Pearson, Gill and Murray, have also considered the theory and applications of Quasi-Newton method. The next section describes the Davidon-Fletcher and Powell method (DFP). The Davidon formula of the Quasi-Netwon class provides a convergent sequence $\boldsymbol{H}$ of approximations to the inverse of the Hessian $\boldsymbol{G}$ of f. Davidon-Fletcher and Powell method enables the inverse of the Hessian $\boldsymbol{G}$ of f to be approximated by $\boldsymbol{H}$, using the gradient of f.

2.6.2 Davidon-Fletcher and Powell method (DFP)

The Davidon-Fletcher-Powell method is very closely related to the steepest descent and Newton's methods. It consists of choosing an $n \times n$ matrix $\boldsymbol{H}_k$ which approximates $\boldsymbol{G}^{-1}$, and forming a direction $-\boldsymbol{H}_k \boldsymbol{g}_k$ as we discussed in the previous section. The unit matrix is usually chosen for the initial $\boldsymbol{H}_0$. If $\boldsymbol{H}_k = I_k$ (where I is a unit matrix), the method coincides with the steepest descent method, and if $\boldsymbol{H}_k = \boldsymbol{G}^{-1}$ that is equivalent to Newton's method. $\boldsymbol{H}_k$ will be selected so that the DFP method starts like the steepest descent method and finishes like Newton's method, but only uses first derivative [Davidon, 1959 and Wolfe, 1978].

Suppose the arbitrary initial choice $\boldsymbol{H}_0$ is the unit matrix, and assume that $\boldsymbol{H}_k$ are symmetrical. Also, if $\boldsymbol{H}_k$ is positive definite then $-\boldsymbol{H}_k \boldsymbol{g}_k$ is downhill for f at $\boldsymbol{x}_k$, it is because $-\boldsymbol{g}_k^{\mathrm{T}} \boldsymbol{H}_k \boldsymbol{g}_k < 0$ if $\boldsymbol{g}_k \neq \boldsymbol{0}$, [Fletcher, 1987]. Therefore, if $\boldsymbol{H}_k$ is positive definite and $\boldsymbol{g}_k \neq \boldsymbol{0}$, then there exist $\alpha_k > 0$ such that $f(\boldsymbol{x}_k - \alpha_k \boldsymbol{H}_k \boldsymbol{g}_k) < f(\boldsymbol{x}_k)$. The DFP then consists of generating $\{\boldsymbol{x}_k\}$ from

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \boldsymbol{H}_k \boldsymbol{g}_k \tag{2.30}$$

If f has continuous second partial derivatives, the gradient of the quadratic function f at a neighbouring point $\boldsymbol{s}_k$ can be written using the Taylor theorem as:

$$\boldsymbol{g}(\boldsymbol{x}_{k+1}) = \boldsymbol{g}(\boldsymbol{x}_k + \boldsymbol{s}_k). \tag{2.31}$$

Dropping the higher order terms, by assuming a quadratic function:

$$\boldsymbol{g}(\boldsymbol{x}_k + \boldsymbol{s}_k) = \boldsymbol{g}(\boldsymbol{x}_k) + \boldsymbol{G}(\boldsymbol{x}_k)\, \boldsymbol{s}_k$$

$$\Rightarrow \qquad G_k s_k = g(x_k + s_k) - g(x_k) = y_k \tag{2.32}$$

where y_k is defined as the difference of the gradients. Now if the matrix $\boldsymbol{H}$ is to behave eventually like $\boldsymbol{G}^{-1}$, then $\boldsymbol{HG} = I$. The above equation is multiplied by $\boldsymbol{H}_{k+1}$, and the identity is used, to obtain:

$$s_k = \boldsymbol{H}_{k+1} y_k \tag{2.33}$$

This is the first updating formula of the Quasi-Newton method by Davidon (1959).

To update the approximation $\boldsymbol{H}_k$ to $\boldsymbol{H}_{k+1}$, the following relation is used:

$$\boldsymbol{H}_{k+1} = \boldsymbol{H}_k + \boldsymbol{A}_k + \boldsymbol{B}_k. \tag{2.34}$$

where $\boldsymbol{A}_k$ and $\boldsymbol{B}_k$ are correction matrices. It can be shown that the updating relation is defined by

$$\boldsymbol{H}_{k+1} = \boldsymbol{H}_k + \boldsymbol{s}_k \boldsymbol{u}_k^{\mathrm{T}} - \boldsymbol{H}_k \boldsymbol{y}_k \boldsymbol{v}_k^{\mathrm{T}} \tag{2.35}$$

where $\boldsymbol{u}_k$ and $\boldsymbol{v}_k$ are any nonzero vectors such that $\boldsymbol{u}_k^{\mathrm{T}} y_k = 1$ and $\boldsymbol{v}_k^{\mathrm{T}} y_k = 1$, [Wolfe, 1978]. Hence, the updating relation satisfies Davidon's equation.

Let $\boldsymbol{u}_k = s_k / s_k^{\mathrm{T}} y_k$ and $\boldsymbol{v}_k = \boldsymbol{H}_k y_k / y_k^{\mathrm{T}} \boldsymbol{H}_k y_k$ which are satisfy (2.34). Then an approximation $\boldsymbol{H}$ of an inverse Hessian, used for unconstrained optimisation, is given by the updating formula

$$\boldsymbol{H}_{k+1} = \boldsymbol{H}_k + \frac{s_k s_k^{\mathrm{T}}}{s_k^{\mathrm{T}} y_k} - \frac{\boldsymbol{H}_k y_k y_k^{\mathrm{T}} \boldsymbol{H}_k}{y_k^{\mathrm{T}} \boldsymbol{H}_k y_k}. \tag{2.36}$$

This formula is due to Fletcher and Powell, and details can be referred to Fletcher (1987), Davidon (1959) and Wolfe (1978).

2.7 Constrained optimisation problem formulation

This section presents methods for finding the minimum of a function $f(\boldsymbol{x})$ subject to constraints $c(\boldsymbol{x}) \le 0$. Previous sections have discussed the cases of no constraints. Some of the methods which are available to solve the constrained optimisation problem are reviewed in this section. The general constrained optimisation problem is given by

$$\min_{x} f(\boldsymbol{x}) \tag{2.37}$$

where f is the objective function, subject to $i = 1, \ldots, m_e$ equality constraints

$$c_i(\boldsymbol{x}) = 0 \quad \text{for} \quad i = 1, \ldots, m_e \tag{2.38}$$

and $i = m_e + 1, \ldots, m$ inequality constraints

$$c_i(\boldsymbol{x}) \leq 0 \quad \text{for} \quad i = m_e + 1, \ldots, m. \tag{2.39}$$

where $\boldsymbol{x} \in \mathbf{R}^n$ and $c_i(\boldsymbol{x})$ are the constraint functions which can be written in a vector form as $\boldsymbol{c}(\boldsymbol{x}) = [c_1(\boldsymbol{x}), \ldots, c_m(\boldsymbol{x})]^{\mathrm{T}}$. The constraints above are restrictions on the operation of the filtering system. The objective function must now be minimised in such a way that all the constraints set by the constraints are met.

We next define the first and the second derivatives of the function c_i, which will be used later on, are written as:

$$\nabla c_i = \frac{\partial c_i}{\partial \boldsymbol{x}} = \begin{bmatrix} \dfrac{\partial c_i}{\partial x_1} & \cdots & \dfrac{\partial c_i}{\partial x_n} \end{bmatrix}^{\mathrm{T}}, \qquad \nabla^2 c_i = \begin{bmatrix} \dfrac{\partial^2 c_i}{\partial x_1^2} & \cdots & \dfrac{\partial^2 c_i}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial^2 c_i}{\partial x_n \partial x_1} & \cdots & \dfrac{\partial^2 c_i}{\partial x_n^2} \end{bmatrix} \tag{2.40}$$

The matrix of the first derivate of the $\boldsymbol{c}$ is defined as:

$$\nabla \boldsymbol{c} = \begin{bmatrix} \nabla c_1 & \cdots & \nabla c_m \end{bmatrix} \tag{2.41}$$

Since the acoustics problems in this thesis are formulated as convex, nonlinear constrained optimisation problems, the following sections present a review of some constrained optimisation methods such as, convex programming and penalty and barrier methods. When the optimisation problem is convex, as in this thesis, penalty and barrier methods can be used to find the global solution. These optimisation methods are applied for the designs of both fixed and adaptive filtering for acoustic problems in later chapters. Other specific constrained optimisation methods such as *semidefinite programming* which consists of a linear objective function subject to a linear matrix inequality constraint; *quadratic programming* which is used to minimise quadratic objective functions with linear constraints; other method such as *sequential quadratic programming* that solve more general constrained optimisation problems, are briefly presented, these are not considered in this thesis. Instead we will consider penalty method due to its computational simplicity. Penalty method is further developed and studied in more detail in section 2.9 and also in later chapters.

2.8 Convex programming

The adaptive filtering problems in this thesis are all formulated as constrained convex optimisation problems, also referred to as convex programming. In this section, the definitions and properties of convex sets and functions are first presented; they have an important characteristic which is related to the existence of a unique global minimum, this property is known as convexity. Then the properties of convex programming problems will be defined and discussed.

The convex optimisation problem (or convex programming) with constraints is formulated as below, in which the objective function f is convex and the constraints c_i are each convex (see, for example, Fletcher, 1987):

$$\begin{array}{lll} \underset{\boldsymbol{x}}{\text{minimise}} & f(\boldsymbol{x}) & \\ \text{subject to} & c_i(\boldsymbol{x}) = 0 \quad \text{for} \quad i = 1, \ldots, m_e & \\ & c_i(\boldsymbol{x}) \leq 0 \quad \text{for} \quad i = m_e + 1, \ldots, m. & \\ & \boldsymbol{x} \in \mathbf{R}^n & \end{array} \tag{2.42}$$

The objective function $f(\boldsymbol{x})$ is convex if and only if $\nabla^2 f(\boldsymbol{x})$ is positive semidefinite [Gill et al, 1981 and Walsh, 1975].

Convex sets: The domain $\boldsymbol{K}$, where $\boldsymbol{K} \subseteq \mathbf{R}^n$, is convex if a straight line between any two points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ in the domain is also part of the domain. That is $\boldsymbol{x}_1, \boldsymbol{x}_2$ in $\boldsymbol{K}$ implies $\boldsymbol{x} = \theta \boldsymbol{x}_1 + (1 - \theta) \boldsymbol{x}_2$, $0 < \theta < 1$, is also in $\boldsymbol{K}$ [Roberts, 1973].

Convex functions: The objective function is convex if its value at any point along the straight line between any two points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ in the domain has an upper bound in the chord through ($\boldsymbol{x}_1$, $f(\boldsymbol{x}_1)$) and ($\boldsymbol{x}_2$, $f(\boldsymbol{x}_2)$), see figure 2.5. More precisely stated, the criteria for convexity is, given two points $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \boldsymbol{K}$, $0 \le \theta \le 1$, the objective function is convex if and only if

$$f[\theta \boldsymbol{x}_1 + (1 - \theta)\boldsymbol{x}_2] \le \theta f(\boldsymbol{x}_1) + (1 - \theta) f(\boldsymbol{x}_2) \qquad (2.43)$$

These definitions have the graphical interpretation as shown in figure 2.4 and 2.5 for examples of the convexity criterion for domain and function, respectively.

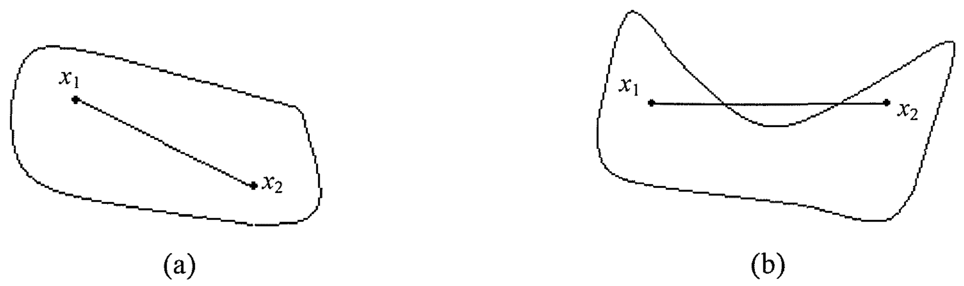


Figure 2.4: Examples of (a) a convex domain and (b) a non-convex domain.

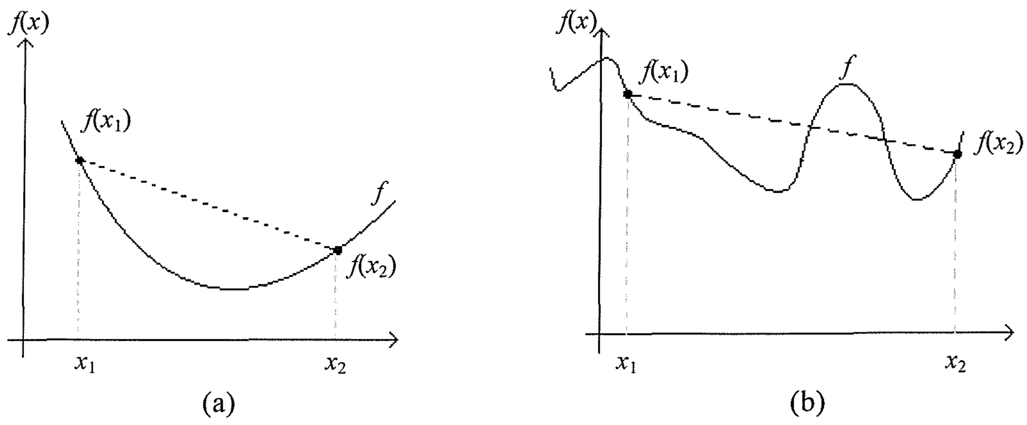


Figure 2.5: Examples of (a) a convex function and (b) a non-convex function. The solid line, f, is the function $f(\theta x_1 + (1 - \theta)x_2)$ and the dashed line is the convexity criteria $\theta f(x_1) + (1 - \theta)f(x_2)$.

Note that the sum of a finite number of convex functions is itself a convex function [Fletcher, 1978]. In addition to the definition given, the following equivalent relations can be used to identify a convex function. A function $f(\boldsymbol{x})$ is convex if for any two points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, the following holds (see Appendix):

$$f(\boldsymbol{x}_2) - f(\boldsymbol{x}_1) \geq \nabla f^{\mathrm{T}}(\boldsymbol{x}_1)(\boldsymbol{x}_2 - \boldsymbol{x}_1). \tag{2.44}$$

If both the domain and the function are convex the problem is said to be convex and in this case a local minimum is also a global minimum. Now, we define the local and global minimum points $\boldsymbol{x}^*$ of a function $f(\boldsymbol{x})$. As before, $\boldsymbol{x}^*$ denotes optimal solution. The point $\boldsymbol{x}^*$ is a local minimum if $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x}$ in the neighbourhood of $\boldsymbol{x}^*$. Similarly, $\boldsymbol{x}^*$ is a global minimum if $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x}$. Necessary conditions for a convex function $f(\boldsymbol{x})$ is that the Hessian matrix $\nabla^2 f(\boldsymbol{x}^*)$ is positive semidefinite, that is $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{G}\boldsymbol{x} \geq 0$, $\forall \boldsymbol{x} \in \mathbf{R}^n$, $\boldsymbol{x} \neq \mathbf{0}$ [Gill et al, 1981 and Walsh, 1975] (this can be proved by Taylor's theorem and equation (2.44), see Appendix). Further, if $\nabla^2 f(\boldsymbol{x}^*)$ is positive definite, the function $f(\boldsymbol{x})$ will be strictly convex, which is a sufficient condition for a local minimum $\boldsymbol{x}^*$, [Gill, 1981]. Using these definitions a very important relation can be derived, which is, any local minimum is a global minimum for a convex function see [Fletcher, 1987]. This relation can be proved by contradiction. Suppose that there exist two different local minima, say $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, for the function $f(\boldsymbol{x})$. Let $f(\boldsymbol{x}_2) < f(\boldsymbol{x}_1)$, and since $f(\boldsymbol{x})$ is convex, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ have to satisfy the relation of (2.44), hence

$$\nabla f^{\mathrm{T}}(\boldsymbol{x}_1)\boldsymbol{s} \leq 0 \tag{2.45}$$

where $\boldsymbol{s} = (\boldsymbol{x}_2 - \boldsymbol{x}_1)$ is a vector joining the point $\boldsymbol{x}_1$ to $\boldsymbol{x}_2$. Equation (2.45) indicates that the value of the function $f(\boldsymbol{x})$ can be decreased further by moving in the direction $\boldsymbol{s} = (\boldsymbol{x}_2 - \boldsymbol{x}_1)$ from point $\boldsymbol{x}_1$. This conclusion contradicts the original assumption that $\boldsymbol{x}_1$ is a local minimum. Thus there cannot exist more than one minimum for a convex function. Hence a local minimum is a global minimum for a convex function [Roberts, 1973].

Some properties of convex function are summarised as follow [Roberts, 1973]:

Positive semidefinite:	if $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{G}\boldsymbol{x} \geq 0$ for all $\boldsymbol{x}$, in which case its eigenvalues are non-negative, and the quadratic form $f(\boldsymbol{x})$ is a convex function.
Convex cones:	A set $\boldsymbol{K}$ is a convex cone if it is a convex set and if $\boldsymbol{x} \in \boldsymbol{K}$ implies $\alpha\boldsymbol{x} \in \boldsymbol{K}$ for α a non-negative scalar.

Summation: For $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ are convex, then $f(\boldsymbol{x}) + g(\boldsymbol{x})$ is convex.

Composite functions: For $f(\boldsymbol{x})$ is convex, $g(\boldsymbol{x})$ is convex and increasing, then $g(f(\boldsymbol{x}))$ is convex.

A computationally efficient and accurate solution to a constrained optimisation problem will depend not only on the size of the problem, in terms of the number of constraints and design variables, but also on characteristics of the objective function and constraints. Referring to optimisation problem formulation (2.42), the problem of minimising a convex function on a convex set $\boldsymbol{K}$ is said to be a *convex programming* problem, if the objective function $f(\boldsymbol{x})$ is a convex function on $\boldsymbol{K}$, and the constraint functions $c_i(\boldsymbol{x})$ are convex on $\boldsymbol{K}$. Introduction of constraints to the optimisation problem means that care must be taken to ensure that the minimum achieved satisfies the constraints. This can be achieved by verifying that there are no further feasible search directions at the specified point.

If $c_i(\boldsymbol{x})$ is a convex function over $\boldsymbol{K}$, then the set of points $\boldsymbol{X}$ satisfying $c_i(\boldsymbol{x}) \leq 0$ (i.e. $\boldsymbol{X}$ is the set of feasible points) is a convex set. Let $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ be points such that $c_i(\boldsymbol{x}_1) \leq 0$, $c_i(\boldsymbol{x}_2) \leq 0$. Then for any i,

$$\begin{aligned} c_i[\theta \boldsymbol{x}_2 + (1 - \theta)\boldsymbol{x}_1] &\leq \theta c_i(\boldsymbol{x}_2) + (1 - \theta)c_i(\boldsymbol{x}_1) && \text{(by convexity)} \qquad (2.46) \\ &\leq 0 && \text{(since } \theta \geq 0,\ 1-\theta \geq 0\text{)} \end{aligned}$$

which means that $\theta \boldsymbol{x}_2 + (1 - \theta)\,\boldsymbol{x}_1$ is in the set of feasible points $\boldsymbol{X}$ wherever $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are, i.e. all the points on the line segment connecting $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are also feasible. Hence, the set of feasible points $\boldsymbol{X}$ is convex. The above can be similarly shown to hold true for equality constraints. Also, the intersection of convex sets is also convex. Therefore, the above shows that the set of points, which are feasible for all the constraints, is convex, providing the constraint functions are all convex (see Fletcher, 1978 for details). This means that the convex programming problem is convex, so that every local minimum is also a global minimum. This property make convex programming very attractive, since optimisation methods that can find a local minimum, will also guarantee to find the global minimum. Convex programming problems will be considered for application in Part II & III of this thesis, so that the review above will be very useful (see Roberts and Verberg, 1973, for further reading and detailed proofs).

2.9 Penalty and barrier methods

Constrained optimisation can prove problematic as the number of constraints increase, especially where the constraints are non-linear. Penalty and barrier methods belong to the first attempts to solve constrained optimisation problems satisfactorily [Fletcher, 1987, Gill, 1981 and Rafaely and Elliott, 2000b]. This section shows how the constrained optimisation problems can be formulated as unconstrained problems using penalty and barrier methods. The basic idea is to construct a sequence of unconstrained optimisation problems and solve them by any standard minimisation method, such as steepest descent method, Newton's method or conjugate gradient method, so that the minimisers of the unconstrained problems converge to the solution of the original constrained one. This approach is used later in this thesis for adaptive filtering with constraints for acoustic applications.

The constrained optimisation problem as presented in section 2.7 (or as (2.42)) sets out a general formulation for either (penalty and barrier) type of method. The problem (2.42) can be formulated as an unconstrained problem by defining a new objective function ϕ, which is composed of the original objective function f, and another function p or b, referred to as penalty function and barrier function respectively. The two methods differ in their approach in that the penalty method approaches the boundary of the feasible region from the exterior and the barrier method approaches the boundary from the interior of the constrained (feasibility) region. In a penalty method, the feasible region of (2.42) is expanded from the feasible region to all of $\mathbf{R}^n$, but a "penalty" function is added to the objective function to penalise the points that lie outside of the original feasible region. In a barrier method, we presume that we are given an initial point that lies in the interior of the feasible region, and we impose a very large cost on feasible points that lie ever closer to the boundary of the feasible region, thereby creating a "barrier" function to exit the feasible region. Note: Only inequality constraints will be discussed here, as the problems solved in the applications later in this thesis, will have only inequality constraints. Therefore equality constraints are not considered here in order to simplify the formulation.

Penalty methods: if a constraint is violated, add a penalty to the objective such that the solution is pushed back towards to the feasible region. The penalty function is zero in the feasible region, but has large values outside of the feasible region, thus penalising the objective function for violating the constraints. The objective is modified by the product of penalty parameter σ and penalty function $p(x)$:

$$\phi(x, \sigma) = f(x) + \sigma p(x) \tag{2.47}$$

where a penalty parameter $\sigma \in [0, \infty)$ is used to control the magnitude of the extra term. As $\sigma \to \infty$, the solution of the reformulated problem tends to the solution of the original problem [Fletcher, 1987]. For convenience of notation, the extra term for penalty methods will be denoted as the penalty function $p(x)$, and the extra term for barrier methods as the barrier function $b(x)$, as appropriate.

The penalty function must satisfy the following

$$\begin{aligned} p(x) &= 0 \quad \text{if } x \text{ is in the feasible region,} \\ p(x) &> 0 \quad \text{otherwise.} \end{aligned} \tag{2.48}$$

As the parameter σ increases, the penalty term grows and the algorithm is forced towards the solution. An often-used class of penalty functions for inequality constraints is:

$$p(x) = \sum_{i=m_e+1}^{m} \max(c_i(x), 0)^{\rho}, \qquad \text{where } \rho \in [0, \infty). \tag{2.49}$$

- If $\rho = 1$, $p(x)$ in (2.49) is called a linear penalty function or a first order penalty function. Then the objective function $f(x)$ with the penalty function formulation (2.49), which corresponds to the constrained problem (2.42), is shown as follow:

$$\phi(x, \sigma) = f(x) + \sigma \sum_{i=m_e+1}^{m} \max(c_i(x), 0) \tag{2.50}$$

  The advantage of (2.50) is that the result is in lower order functions, the formulation is computationally simple and the optimum can be found in the feasible region. However, its drawback is that $p(x)$ is not differentiable at the boundary of the feasible region where $c_i(x) = 0$ for some i, i.e. may cause discontinuous gradient at the boundary of feasible domain. Figure 2.6 (a) illustrates an unconstrained problem formulation using a linear penalty function as in equation (2.50).

- Setting $\rho = 2$ in (2.49) is a most common penalty function that is used in practice [Fletcher, 1978, Gill et al, 1981 and Walsh, 1975], and is called a quadratic penalty function or a second order penalty function. Then, the modified objective function ϕ would become

$$\phi(x, \sigma) = f(x) + \sigma \sum_{i=m_e+1}^{m} \max(c_i(x), 0)^2 \tag{2.51}$$

The quadratic penalty function has an advantage over the linear penalty function since it is differentiable on **R**, and is a "smooth" penalty function. Therefore, the continuous gradients reduces numerical problems. However the second order derivatives of ϕ can be discontinuous at points where $c_i(x) = 0$, due to the max(-) operator. Hence, one cannot safely use Newton's method to minimise the penalty function. The disadvantage of the quadratic penalty method is that the optimum of modified problem could be infeasible. Figure 2.6 (b) illustrates an unconstrained problem formulation using a quadratic penalty function as in equation (2.51).

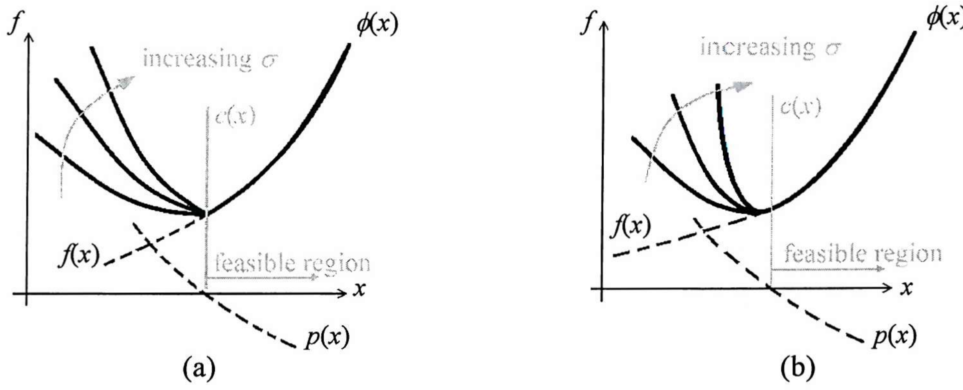


Figure 2.6: The constrained problem of minimising a quadratic objective function $f(x)$, subject to the constraint $c(x)$, is formulated as an unconstrained problem of minimising the function $\phi(x)$, using (a) a linear penalty function and (b) a quadratic penalty function.

Generally, the minimum of $\phi(x, \sigma)$ can be found iteratively, increasing the value of σ as the number of iterations increases. The unconstrained optimisation algorithms described in this chapter can be used to find this minimum. This solution will be very close to the solution of the constrained problem. However, care must be taken as ill-conditioning may occur when high values of σ are used. When x is in the feasible region, all $c_i(x) \leq 0$ and $p(x) = 0$. That is, no penalty is incurred. When x is not in the feasible region, that is, $c_i(x) > 0$ for some i. Also, $p(x) > 0$ and a high penalty is therefore imposed if σ is high.

The problem is solved by first choosing an increasing sequence $\{\sigma_k\} \rightarrow \infty$. Then, for each σ_k, the x_k that minimises $\phi(x,\sigma)$ in the objective function above is calculated. Finally, iterations are terminated when sufficiently close to the solution. The termination condition could be, for instance, when the changes in ϕ or x are smaller than a set criterion.

Other types of penalty functions also exist, such as the *multiplier or augmented Lagrangian functions*. As we have discussed above, the smooth quadratic penalty function never generate the optimum solutions in the feasible region to the constrained optimisation problem. Therefore we need to solve the unconstrained problems with very large values of the penalty parameter σ in order to obtain solutions that are close to being feasible and optimal. However ill-conditioning may occur when σ increases, and therefore it will be hard to solve the problem. The multiplier or augmented Lagrangian functions is the method that trys to avoid the use of a large penalty parameter σ, that lead to ill-conditioned unconstrained subproblems, by shifting the origin of the penalty function to the solution point x^*. Details of the multiplier or augmented Lagrangian functions will not be discussed further here, as this method would not be considered, because later in the thesis we will be concentrating on using the linear penalty function. The linear penalty function gives a simple formulation, which will facilitate a comprehensive theoretical analysis that is derived in chapter 7. For more information about augmented Lagrangian functions and other types of penalty functions, the reader can refer to Fletcher (1987) and Leunberger (1973).

Barrier methods imposes a penalty for reaching the boundary of an inequality constraint. The idea in a barrier method is to prevent points x from ever approaching the boundary of the feasible region. The barrier methods differ from the penalty methods in that the iterations converge to the solution from interior points rather than exterior. Figure 2.7 illustrates the idea of the unconstrained problem formulations by using barrier function.

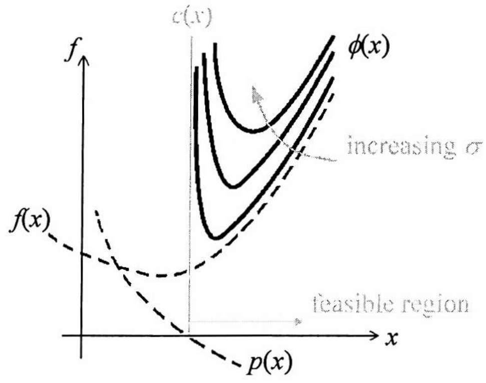


Figure 2.7: The idea of barrier function method (interior point method).

The advantage of the barrier method is that intermediate solutions are always feasible, and the disadvantage is that equality constraints cannot be handled. Barrier methods will be discussed briefly here, because penalty function method will only be used later in the thesis for constrained filtering problem due to its simplicity and its formulation structure, which will be needed for further development in the theoretical analysis, chapter 7. Barrier functions are more complicated to implement and its search can more easily go unstable. The modified formulation of the objective function is similar to that of the penalty function above, that is:

$$\phi(\boldsymbol{x}, \sigma) = f(\boldsymbol{x}) + \sigma b(\boldsymbol{x}) \tag{2.52}$$

where $b(\boldsymbol{x})$ is the barrier function and σ is the penalty parameter. Barrier methods construct a sequence of unconstrained problems and a parameter σ controls the weight of the barriers. The difference between the methods lies in the choice of the penalty function and barrier function. The objective function ϕ of barrier method is defined only in the feasible region and it must have a feasible initial starting point. The most popular barrier functions are [Fletcher 1987 and Leunberger 1973]:

$$\begin{aligned} &\text{Logarithmic barrier term:} && b(\boldsymbol{x}) = -\sum_{i=1}^{m} \log\left(c_i(\boldsymbol{x})\right) \\ &\text{Inverse barrier term:} && b(\boldsymbol{x}) = \sum_{i=1}^{m} \left[c_i(\boldsymbol{x})\right]^{-1} \end{aligned} \tag{2.53}$$

The problem is solved by first taking a decreasing sequence $\{\sigma_k\} \rightarrow \infty$, the weight of the constraints is then decreased step by step, then finding the minimum of $\phi(\boldsymbol{x},\sigma)$ for each σ_k. The barrier functions tend to infinity if a feasible iteration converges to the border of the feasible region. The sequence is chosen so that the barrier term is very small near the boundary, and hence the solutions $\boldsymbol{x}_k$ converge to the solution $\boldsymbol{x}^*$ of the constrained problem. The sequence $\boldsymbol{x}_k$ begins in the feasible region, and moves towards the boundary; the barrier method is also therefore called an interior point method. Note that figures 2.6 and 2.7 are used above to be consistent with the literature on penalty and barrier methods [Ing, 2002].

Many practical applications require the solving of constrained optimisation problems. This section has demonstrated how the penalty and barrier methods can be used to reformulate these into an equivalent unconstrained problem, which will generally be more computationally efficient to solve, using the algorithms reviewed earlier in this chapter.

2.10 Other methods for solving constrained optimisation problems

2.10.1 Semidefinite programming

Many convex optimisation problems can be cast as semidefinite programming (SDP) problems. It may therefore be possible, using SDP, to derive algorithms for a wide variety of convex optimisation problems. Also, the purpose of using SDP is to solve problems with matrices of large dimensions. SDP is an extension of linear programming (LP), where the variables can be symmetric matrices and the non-negativity constraints are replaced by a positive semidefinite constraint.

A standard form of semidefinite programming is [Boyd et al, 1996]:

$$\begin{aligned} &\text{minimise } \boldsymbol{d}^{\mathrm{T}}\boldsymbol{x} \\ &\text{subject to } \boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{F}_0 + \sum_{i=1}^{m} x_i \boldsymbol{F}_i \geq \mathbf{0}. \end{aligned} \tag{2.54}$$

An SDP problem is a optimisation problem, consisting of a linear objective function of a vector of variables $\boldsymbol{x} \in \mathbf{R}^m$, subject to a linear matrix inequality (LMI) constraint. This constraint is a combination of symmetric matrices $\boldsymbol{F}_0, \ldots, \boldsymbol{F}_m \in \mathbf{R}^n$. The inequality sign in $\boldsymbol{F}(\boldsymbol{x}) \geq \mathbf{0}$ means that $\boldsymbol{F}(\boldsymbol{x})$ is positive semidefinite. The problem of finding $\boldsymbol{x}$ that satisfies the LMI constraint, or proving that no such $\boldsymbol{x}$ exists, is called an SDP feasibility problem; these can be cast and solved as ordinary SDPs.

The acoustic applications that we wish to solve in this thesis have quadratic objective functions subject to quadratic constraints. These can be formulated by building a matrix to use in the SDP, and constructing a linear matrix inequality. SDP therefore extends LP by taking the component-wise inequalities between the vectors and converting them to matrix inequalities.

Since the main concern of this thesis is acoustic applications for constrained optimisation, it is interesting to see that quadratically constrained quadratic programming (QCQP) can be cast as an SDP. In general, the SDP problems generated are too complicated to solve analytically and must be solved numerically. However, it has been suggested that SDP may not perform well numerically and that using other methods would be preferable. Research into this area showed that even when the functions involved were quite straight forward, a lot of time and effort is still needed to convert and solve the problem using SDP. Furthermore, the complex matrix structure tends to reduce the insight into the problem. It is possible that practical acoustic problems, solved using QCQP's, would be very complex and time consuming to set up and solve.

From experimentation and consultation with professors from other countries (e.g S.P. Boyd from Stanford University 1996, H. Wolkowicz from Waterloo University 2000, F. Jarre from Norte Dame University 1998, M.J. Todd from Cornell University 2001, C. Helmberg from Germany 2000, R.M Freund from Cambridge 1994, S.E. Karisch from Denmark Copenhagen University 1998 and K.M. Anstreicher from Lowa University 1998 etc) it was found that using SDP to solve problems with quadratic constraints was computationally expensive and inefficient. In this case, the penalty function method should be used instead. This would mean directly applying penalty method to the problem rather than transforming it to a larger SDP problem. Please note that the dates above are not of contact dates but of papers/ references dates.

2.10.2 Quadratic programming and Sequential quadratic programming

Since the constrained optimisation problems considered in the thesis are nonlinear programming problem, which have quadratic objective function with nonlinear constraint functions (quadratic constraints) as defined in (2.42). One method for solving equation (2.42) involves dividing the problem into a sequence of subproblems, each of which can be solved using quadratic programming (QP). Quadratic programming problems are a fundamental type of optimisation programming problem. The standard form quadratic programming problem is characterised by a quadratic objective and linear constraints. QP is introduced next, while the sequential method which solves equation (2.42), refereed to as sequential quadratic programming (SQP), is discussed after QP. Note that these two programming methods are only briefly discussed here, because they are not considered in this thesis due to the computational complexity compared with penalty method; the simple formulation of penalty method is considered throughout the thesis and is further developed for the constrained adaptive filtering problems.

Quadratic programming problem have quadratic objective function with linear constraints, and formulated as:

$$\text{Minimise} \qquad f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{G}\boldsymbol{x} + \boldsymbol{d}^{\mathrm{T}}\boldsymbol{x}$$

$$\text{Subject to} \quad \boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x} - b_i = 0 \quad i = 1, \ldots, m_e \tag{2.55}$$
$$\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x} - b_i \leq 0 \quad i = m_e+1, \ldots, m$$

where $\boldsymbol{G}$ is a symmetric positive semidefinite $n \times n$ matrix, $\boldsymbol{d}$ is an n-dimensional row vector describing the coefficients of the linear terms in the objective function and $\boldsymbol{a}_i$ refers to the ith row of the $m \times n$ matrix $\boldsymbol{A}$, where $\boldsymbol{A}$ is the matrix included rows $\boldsymbol{a}_i$. The constraints $\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x} - b_i = 0$ are referred to as equality constraints while $\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x} - b_i \leq 0$ are known as inequality constraints.

One of the methods for solving QP problem is an active set method (also known as a projection method) which use the principle of active set, where only the active constraints are considered along the solution process. This method has been modified for both linear programming and quadratic programming problems. The active set method is iterative, and in each iteration k, a new solution $\boldsymbol{x}_k$ is found. After finite number of iterations, the solution of problem (2.55) is found, provided that it exits. The solution procedure involves two parts: 1) the calculation of a feasible point, if one exits; 2) the generation of an iterative sequence of feasible points that converge to the solution. Let the matrix $\boldsymbol{A}$ include rows $\boldsymbol{a}_i$ which correspond to active constraints at the solution point, i.e. equality constraints always remain in the active set $\boldsymbol{A}$, and thus these rows correspond to all the equality constraints and the active inequality constraints. At each iteration $\boldsymbol{A}$ is estimated by $\boldsymbol{A}_k$ which corresponds to the active constraints of the estimate of the solution point $\boldsymbol{x}_k$. $\boldsymbol{A}_k$ is updated at each iteration k, and this is used to form a basis for a search direction. The estimate of the solution point is updated by the update equation $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$, where $\boldsymbol{d}_k$ is the search direction and α_k is the step size at kth iteration. The search direction is calculated and minimises the objective function while remaining on any active constraint boundaries. The main idea behind the active set method is that the new solution $\boldsymbol{x}_{k+1}$ will maintain the active constraints of the point $\boldsymbol{x}_k$. The technical details of solving quadratic programs will not be dealt with here. Further descriptions of active set method for QP problems can be referred to Grace (1995) or Gill at el (1981) and Fletcher (1987) for more detail.

Lets now briefly described the basic idea of the sequential quadratic programming method. A general SQP problem has a quadratic objective function with nonlinear constraints. The idea of SQP is to model nonlinear programming defined in equation (2.42) at a given approximate solution, say $\boldsymbol{x}_k$, by a quadratic programming subproblem, and then to use the solution to this subproblem to construct a better approximation $\boldsymbol{x}_{k+1}$. Basically, the nonlinear programming can be solved iteratively by calculating the solution to a QP subproblem at each iteration, and then the solution to the QP subproblem gives the search direction for the next iteration. This process is iterated to create a sequence of approximations that will converge to a solution $\boldsymbol{x}^*$. A more detailed description and formulation of the SQP can be found in Fletcher (1987), Gill (1981), Walsh (1975) and Rafaely (1997).

2.11 Conclusions

General unconstrained and constrained optimisation problems have now been introduced and various methods for their solution have been presented. Also, convex programming has shown that any minimum point found will be a global minimum, if the functions are convex. In the acoustic application considered later in this thesis, convexity will be assumed to hold, and the methods presented in this chapter will be used to solve the problem. In particular, the penalty method will be used to convert a constrained optimisation problem to an unconstrained optimisation one, and versions of the unconstrained optimisation methods discussed will be applied to the constrained adaptive filtering in acoustic applications.

Chapter 3

Adaptive filtering

3.1 Introduction

This chapter presents a review of the signal processing theory that is applicable to the objectives of this thesis. Adaptive filters are useful in many signal processing applications because of their ability to give satisfactory results in situations with unknown input data and their ability to track time variation of input data. Acoustic applications are often required to be adaptive when the acoustic plant is time-varying, or when the disturbance is changing in time, in order to achieve good performance and maintain stability over time.

Adaptive filtering has been applied to a very wide range of acoustic applications including echo cancellation, active noise control systems and hearing aids [Widrow and Stearns, 1985]. These applications all have in common the fact that a set of filter coefficients is controlled by minimising an error, where the error is the difference between an output signal and a desired response. An adaptive filter relies on a recursive algorithm to update its structure. This chapter provides a background on adaptive filters and also highlights several recursive algorithms that will be used in the constrained adaptive filtering problems later in the thesis. The chapter provides reviews on the time-domain and frequency-domain LMS algorithms, bin-normalized LMS algorithm and conjugate gradient algorithm.

Many algorithms are available for solving adaptive filtering problems. Different algorithms have different characteristics: rate of convergence, or number of iterations to reach solution; misadjustment, or difference in mean square error (MSE) between the optimal and estimated solution; tracking, which is affected by rate of convergence and fluctuation in data; robustness, in terms of numerical behaviour or behaviour when data is ill-conditioned; computation complexity, such as number of operations involved, the size of the memory required and the cost of computations; structure of the information flow in the algorithm; numerical properties such as stability and accuracy. The requirement for different adaptive filtering problems is different, and depends on the application. The appropriate algorithm must be chosen for the application to achieve good performance at minimum cost. This chapter develops algorithms based on Wiener filter theory.

The chapter is presented as follows. The Wiener filter is first reviewed, with the choice of MSE as cost function for minimisation. Then the Least Mean Squares (LMS) algorithm and bin-normalized LMS algorithm are introduced, along with its block and frequency domain formulations. The LMS algorithm is widely used in current applications in signal processing. The conjugate gradient algorithm (CGA) is then introduced in the view of adaptive filtering. The convergence rate and computational complexity of these algorithms are compared in Part II and III.

3.2 The Wiener filter

The algorithms discussed later in this chapter all minimises the variance of the error signal and convergence to an optimum solution or the Wiener filter. In this section we present the optimal filtering problem, and its solution: the Wiener filter. The Wiener filter is presented in, for example, Cowan (1985) and Widrow and Stearns (1985), where given d the desired signal and x the reference signal, then the output signal y is required to be as close as possible to the desired signal d. Thus the solution is the optimal filter: the Wiener filter. This is done on the assumption that x and d are stationary random signals, and $\mathbf{w}$ is a finite impulse response (FIR) filter of length N. As with many signal processing problems, the comparison between output signal y and desired signal d is measured by the mean squared value. The MSE is therefore a criterion chosen as the cost function for optimisation.

The general filtering problem can be illustrated as shown in the figure below [Haykin, 1986]. A random stationary signals x is passed through a filter $\mathbf{w}$; the output is a signal y. The difference between the output signal and the desired signal is the error e, defined as $e = d - y$.

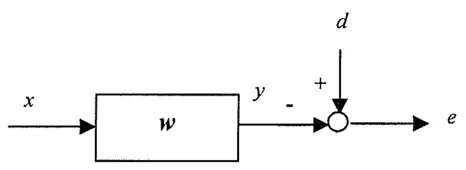


Figure 3.1: Illustration of the optimal filtering problem.

The Wiener filter minimises the MSE which is given by $E[e(n)^2]$, where E is the expectation operator, e is the error signal and n is the current discrete time. The error e can be stated explicitly as:

$$e(n) = d(n) - \sum_{i=0}^{N-1} w_i x(n-i) \tag{3.1}$$

where w_i are the filter coefficients, $x(n)$ is the current sample, N is the length of the adaptive filter and n is the sampling instant. Equation (3.1) can be rewritten using the inner product as follows:

$$e(n) = d(n) - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}(n) \tag{3.2}$$

In equation (3.2), $\boldsymbol{w} = [w_0 \ w_1 \ \dots \ w_{N-1}]^{\mathrm{T}}$ and $\boldsymbol{x}(n) = [x(n) \ x(n-1) \dots \ x(n-N+1)]^{\mathrm{T}}$ are both $N \times 1$ vectors.

The variance of the error signal MSE produces the error surface equation given by:

$$\begin{aligned} E[e^2(n)] &= E[\,(d(n) - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}(n))^2\,] \\ &= E[\,(d(n) - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}(n)) \cdot (d(n) - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}(n))^{\mathrm{T}}\,] \\ &= E[\,\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}(n) \boldsymbol{x}(n)^{\mathrm{T}} \boldsymbol{w} - d(n) \boldsymbol{x}(n)^{\mathrm{T}} \boldsymbol{w} - d(n) \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}(n) + d^2(n)] \end{aligned}$$

The filter coefficients in $\boldsymbol{w}$ are assumed to be time invariant, so the equation above can be written as:

$$\begin{aligned} &= \boldsymbol{w}^{\mathrm{T}} E[\boldsymbol{x}(n) \boldsymbol{x}(n)^{\mathrm{T}}] \boldsymbol{w} - 2\boldsymbol{w}^{\mathrm{T}} E[\boldsymbol{x}(n) d(n)] + E[\,d^2(n)] \\ &= \boldsymbol{w}^{\mathrm{T}} \boldsymbol{R}\, \boldsymbol{w} - 2\boldsymbol{w}^{\mathrm{T}} \boldsymbol{p} + E[d^2(n)] \end{aligned} \tag{3.3}$$

where,

$\boldsymbol{R} = E[\boldsymbol{x}(n)\boldsymbol{x}^{\mathrm{T}}(n)]$ is the symmetric autocorrelation Toeplitz $N \times N$ matrix of x, and $R_{xx}(i) = E[x(n)x(n+i)] = R_{xx}(-i)$ is the auto-correlation function of x:

$$\boldsymbol{R} = \begin{bmatrix} R_{xx}(0) & R_{xx}(1) & \dots & \dots & R_{xx}(N-1) \\ R_{xx}(1) & R_{xx}(0) & \dots & \dots & R_{xx}(N-2) \\ \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ R_{xx}(N-1) & R_{xx}(N-2) & \dots & \dots & R_{xx}(0) \end{bmatrix} \tag{3.4}$$

$\boldsymbol{p} = E[d(n)\boldsymbol{x}^{\mathrm{T}}(n)]$ is the cross-correlation $N \times 1$ vector between d and x, and $R_{xd}(i) = E[x(n)d(n+i)] = E[x(n-i)d(n)]$ is cross-correlation functions of x and d:

$$p = [R_{xd}(0) \quad R_{xd}(1) \quad \ldots \quad \ldots \quad R_{xd}(N-1)]^{\mathrm{T}} \tag{3.5}$$

It can be seen that equation (3.3) is a quadratic function of the coefficients w_i. It is a second order function of the tap weights. Plotting any two filter coefficients against the MSE produces a convex, bowl shape surface. This implies that the MSE is a quadratic function of the w_i and that it will have a unique minimum if the matrix $\boldsymbol{R}$ is positive definite that is, $w^{\mathrm{T}}\boldsymbol{R}w > 0$ for all w [Widrow and Stearns, 1985]. Another condition for $\boldsymbol{R}$ to be positive definite, is if the reference signal x is *persistently existing* [Elliott, 2000] which means the signal x has at least half as many spectral components as the filter weights w. The minimum can be found when the derivative of the MSE with respect to each w_i is zero, at the optimal filter coefficient vector, w^*. The derivative with respect to the filter coefficients w_i is:

$$\frac{\partial E[e^2(n)]}{\partial w} = 2w_{opt}^{\mathrm{T}}\boldsymbol{R} - 2p = 0 \tag{3.6}$$

The optimal filter can therefore be calculated as:

$$w^* = \boldsymbol{R}^{-1}p. \tag{3.7}$$

The FIR filter in equation (3.7) is known as the Wiener filter.

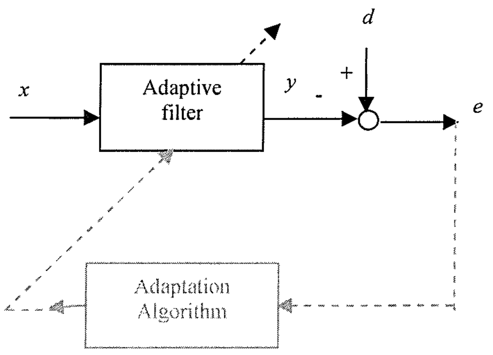


Figure 3.2 Illustration of adaptive implementation of a Wiener filter

A typical adaptive implementation of a Wiener filter is illustrated in figure 3.2. The adaptive filter works by filtering the input signal x using the current filter weights w_i at time n, then using the error to update the filter coefficients in the direction of the optimal coefficients. If the input signals are stationary, the system can learn the pattern of the inputs x and d, so that eventually the filter coefficients will reach the optimal $\boldsymbol{w}^*$. In practical implementations of adaptive filters, the length of time or number of computations taken to reach the optimal coefficients will depend on the algorithm and filter length used. If the input signal changes, the filter will respond by adjusting the direction of the updating process towards the new optimal $\boldsymbol{w}^*$. In the following, recursive algorithms are developed to calculate the filter coefficients of the optimal filter.

3.3 The LMS algorithm

The optimal Wiener filter as previously shown in equation (3.7) is computationally expensive because of the inversion of matrix $\boldsymbol{R}$, which cannot be easily accomplished in real time. An alternative way of calculating the optimal solution that is to approach it iteratively by using gradient based search methods which work well when the error surface is convex. Therefore a steepest descent search can be used here to approach the minimum point. By updating the coefficients of the filter $\boldsymbol{w}$ with steepest descent search, in such an iterative process, we can obtain the minimum point $\boldsymbol{w}^*$. The filter $\boldsymbol{w}$ is updated iteratively using the following update equation:

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) - \mu\nabla(n) \tag{3.8}$$

where μ is the convergence coefficient and ∇ is the gradient vector which can be defined by:

$$\nabla(n) = \frac{\partial E[e^2(n)]}{\partial \boldsymbol{w}} = 2\boldsymbol{w}(n)^{\mathrm{T}}\boldsymbol{R} - 2\boldsymbol{p} \tag{3.9}$$

where $\boldsymbol{R}$ is an auto-correlation matrix and $\boldsymbol{P}$ is the cross-correlation vector as defined in the previous section. Equation (3.9) can be written in terms of the error signal and the reference signal as:

$$\nabla(n) = \frac{\partial E[e^2(n)]}{\partial \boldsymbol{w}} = E\left[2e(n)\frac{\partial e(n)]}{\partial \boldsymbol{w}}\right] = -2E[e(n)\boldsymbol{x}(n)] \tag{3.10}$$

We can see that $\boldsymbol{w}$ in the update equation (3.8) starts from an initial point $\boldsymbol{w}_0$ and updated in the opposite direction of the gradient of the error surface as shown in (3.10). It will iteratively move to the minimum along the steepest descent direction on the error surface.

The gradient vector (3.10) is known as the stochastic gradient descent since it estimates the value of the gradient at each iteration; the exact measurements of correlation matrix $\boldsymbol{R}$ and cross-correlation vector $\boldsymbol{p}$ can be computationally expensive and cannot be easily accomplished in real time. However, we can estimate the gradient vector $E[e(n)\boldsymbol{x}(n)]$ by using the instantaneous values of signals e and $\boldsymbol{x}$. Widrow and Stearns (1985) present the LMS algorithm where instead of using gradient vector $E[e(n)\boldsymbol{x}(n)]$ in (3.10), the update equation with the instantaneous gradient estimates $e(n)\boldsymbol{x}(n)$ becomes:

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + 2\mu\, e(n)\boldsymbol{x}(n) \tag{3.11}$$

The LMS algorithm is introduced as a method that uses an estimate of the gradient from the available data. The approximation may seem very rough from a theoretical perspective. However, each successive iteration will be using new values of $\boldsymbol{x}$, and the effect of this selection of sample values is that it will tend to bring the estimated value more in line with the actual expected value. The LMS algorithm has the advantage of simple implementation, and does not require complex computations such as matrix inversion.

The convergence speed of the LMS adaptive algorithms is limited by the eigenvalue spread of the autocorrelation matrix $\boldsymbol{R}$, where eigenvalue spread is the ratio between the maximum and the minimum of magnitude of the eigenvalues of $\boldsymbol{R}$, which also correspond to the spread in the magnitude of the Fourier transform of the input data. Fast convergence towards the optimal solution is obtained when the eigenvalue spread of the matrix $\boldsymbol{R}$ is small i.e. when the power spectrum of the reference signal $\boldsymbol{x}$ is flat, which means the reference signal $\boldsymbol{x}$ is white-noise. The LMS algorithm has the disadvantages of slow convergence when the matrix $\boldsymbol{R}$ has a wide eigenvalues spread, i.e. when the power spectrum of the reference signals $\boldsymbol{x}$ is non-flat. The convergence coefficient μ of the LMS has a significant effect on the efficiency of the algorithm. If μ is too large, this could results in divergence from the solution. If μ is too small, the filter might converge, but convergence might be too slow for practical applications. Therefore the bound of the convergence coefficient for stability of the LMS to be maintained is now considered and is defined as follow (see Widrow and Stearns 1985, Cowan 1985, Haykin 1991). The detail of the convergence coefficient of the LMS algorithm defined here will be used later for our theoretical analysis chapter 8.

In section 3.2, the gradient with respect to $\boldsymbol{w}$ at the minimum point is defined as:

$$\nabla = 2\boldsymbol{R}\boldsymbol{w}^{*} - 2\boldsymbol{p} \tag{3.12}$$

and, an update equation of the steepest descent algorithm with a step size μ is

$$w_{n+1} = w_n + \mu(-\nabla)$$

$$\Rightarrow$$

$$w_{n+1} = w_n + 2\mu(\boldsymbol{p} - \boldsymbol{R}w_n) \tag{3.13}$$

Rewriting the equation above, it can be seen that

$$w_{n+1} = (\boldsymbol{I} - 2\mu\boldsymbol{R})w_n + 2\mu\boldsymbol{p} \tag{3.14}$$

Let the Wiener solution be $\boldsymbol{w}^*$. Then the convergence coefficient is calculated using the difference,

$$w_{n+1} - \boldsymbol{w}^* = (\boldsymbol{I} - 2\mu\boldsymbol{R})w_n + 2\mu\boldsymbol{p} - \boldsymbol{w}^*$$

$$\Rightarrow$$

$$w_{n+1} - \boldsymbol{w}^* = (\boldsymbol{I} - 2\mu\boldsymbol{R})(w_n - w^*) \tag{3.15}$$

Substituting $\boldsymbol{s}_n = w_n - \boldsymbol{w}^*$,

$$\boldsymbol{s}_{n+1} = (\boldsymbol{I} - 2\mu\boldsymbol{R})\,\boldsymbol{s}_n \tag{3.16}$$

Using induction, it follows that:

$$\boldsymbol{s}_n = (\boldsymbol{I} - 2\mu\boldsymbol{R})^n\,\boldsymbol{s}_0 \tag{3.17}$$

The above series of equations will converge only if

$$|\boldsymbol{I} - 2\mu\boldsymbol{R}| < 1 \tag{3.18}$$

$\boldsymbol{R}$ is a Hermitian Toeplitz matrix, so it can be decomposed into the orthogonal representation:

$$\Lambda = \boldsymbol{QRQ}^{\mathrm{T}} \tag{3.19}$$

where $\boldsymbol{Q}$ is the orthogonal matrix composed of eigenvectors of $\boldsymbol{R}$, $\Lambda = \mathrm{diag}(\lambda_0, \ldots, \lambda_{N-1})$ is the diagonal matrix of eigenvalues of $\boldsymbol{R}$ and $\boldsymbol{QQ}^{\mathrm{T}} = \boldsymbol{I} = \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q}$ [Moon, 2000]. Applying equation (3.19) into (3.17), and let $z_n = \boldsymbol{Q}^{\mathrm{T}}\,s_n$, then using induction to obtain:

$$z_n = (\boldsymbol{I} - 2\mu\Lambda)^n\,z_0 \tag{3.20}$$

Since ($\boldsymbol{I}$ - $2\mu\Lambda$) is diagonal, the above can be expressed as:

$$\begin{aligned} z_n^{[1]} &= (1-2\mu\lambda_1)^n z_0^{[1]} \\ z_n^{[2]} &= (1-2\mu\lambda_2)^n z_0^{[2]} \\ &\vdots \\ z_n^{[N]} &= (1-2\mu\lambda_N)^n z_0^{[N]} \end{aligned} \qquad (3.21)$$

It is clear that for convergence of this series, it is required that

$$|\boldsymbol{I} - 2\mu\lambda_i| < 1 \qquad i = 1, 2, \ldots, N$$

hence

$$0 < \mu < \frac{1}{\lambda_i} \qquad i = 1, 2, \ldots, N \qquad (3.22)$$

The maximum λ_i must be taken so that all the constraints are satisfied. Therefore:

$$0 < \mu < \frac{1}{\lambda_{\max}} \qquad (3.23)$$

To avoid computation of the eigenvalues of the matrix $\boldsymbol{R}$, the trace of $\boldsymbol{R}$ is often used in practice instead:

$$0 < \mu < \frac{1}{\mathrm{tr}[\boldsymbol{R}]} \qquad (3.24)$$

If the matrix $\boldsymbol{R}$ is positive definite, then $\mathrm{tr}[\boldsymbol{R}] = \Sigma\lambda_i > \lambda_{\max}$. Therefore $\mu < 1/\mathrm{tr}[\boldsymbol{R}]$ guarantees that $\mu < 1/\lambda_{\max}$. The $\mathrm{tr}[\boldsymbol{R}]$ can be easily computed because $\mathrm{tr}[\boldsymbol{R}] = N\mathrm{Var}\{x(n)\}$, where $\mathrm{Var}\{x(n)\}$ estimated from the data. So equation (3.24) can be written as:

$$0 < \mu < \frac{1}{NVar\{x(n)\}} \qquad (3.25)$$

Note that the convergence coefficient μ is essentially limited by the eigenvalue spread of the data matrix $\boldsymbol{R}$ as we mentioned above. The material revised here will be used later in the thesis.

3.4 The Block LMS algorithm

The previous chapter showed how the LMS algorithm was useful for optimisation problems for adaptive filters. However, when considering problems with very long FIR filters, it would be preferable to find another method of finding the solution. A more efficient way to calculating the FIR weights is to wait until a block of samples has arrived, the block size can be chosen as any length L, then updating the adaptive filter every L samples instead of every sample. This method is referred to as the block LMS (BLMS) algorithm [Clark et al, 1981, Lee and Un, 1989, and Shynk, 1992]. The BLMS updating technique will therefore be used to achieve an efficient method for solving problems with very long filter lengths. The BLMS algorithm is presented in this section, as its fundamental principle is used for frequency domain implementations.

The previous section showed that the LMS algorithm minimises the MSE by updating the filter coefficients for each new input signal $\boldsymbol{x}(n)$ and corresponding desired signal $d(n)$. The BLMS approach is to keep the filter coefficients fixed until L data items have been gathered; the block of data is then used to update the filter coefficients. It can be shown that the block recursion for filter coefficients at time $n+L$ is given by (see Shynk, 1992):

$$\boldsymbol{w}(n+L) = \boldsymbol{w}(n) + 2\mu\sum_{m=0}^{L-1} e(n+m)\boldsymbol{x}(n+m) \tag{3.26}$$

Whereas the standard LMS algorithm uses a new weight vector for each recursion, the block implementation above uses the same weight vector $\boldsymbol{w}$ for each calculation of the error. That is, each error

$$e(n+m) = d(n+m) - y(n+m), \qquad \text{for } m = 0, \ldots, L\text{-}1 \tag{3.27}$$

where $y(n+m) = \boldsymbol{x}^{\mathrm{T}}(n+m)\boldsymbol{w}(n)$. For convenience of notation, the time index is rewritten using a new index k, which represents each block instant. Taking $n = kL$, and rewriting (3.26) as: [Shynk, 1992]

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + 2\mu L\left[\frac{1}{L}\sum_{m=0}^{L-1} e(kL+m)\boldsymbol{x}(kL+m)\right] \tag{3.28}$$

It can be seen from equation (3.28) that the BLMS differs from the standard formulation in that the gradient estimate it uses is a more accurate representation of the true gradient. This is because the BLMS averages the gradient over the sample data in the block, i.e. it can approach the true gradient for a large L. Hence a smoother convergence can be obtained due to the gradient estimate which is less noisy. Equation (3.28) also shows that the maximum step size for stable convergence is lower for the block formulation of the LMS than the standard formulation, since the convergence coefficient is now μL. The bounds of the convergence coefficient for stability of the BLMS to be maintained is now given by:

$$0 < \mu_B < \frac{1}{N Var\{x(n)\}} \tag{3.29}$$

where $\mu_B = \mu L$. This tighter bound on the step size can lead to slow convergence for the BLMS on problems where the eigenvalue spread of the correlation matrix is large, or for non-flat power spectrum of the reference signal.

The BLMS algorithm has been introduced in the time domain; in the next section we present the implementation of the BLMS in the frequency domain, thus known as the frequency domain LMS algorithm.

3.5 The frequency domain LMS algorithm

In this section the implementation of frequency domain LMS (FDLMS) is presented, it will then be used later in the thesis for constrained adaptive filtering problems. In some acoustic applications such as in equalisation systems where the acoustic plant impulse response is long, a large number of coefficients is usually required for good performance. The LMS algorithm can be inefficient for very long filter lengths as many multiply and add operations are needed due to the direct convolution calculation required. For an adaptive filter of length N, every N samples processed using the LMS in the time domain requires N^2 operations. However, if the filtering operation of the BLMS algorithm is performed in the frequency domain using the fast Fourier transforms (FFT), only approximately $N(\log_2 N)$ operations are required, the computational complexity is greatly reduced i.e. greatly increasing efficiency [Bershad and Fenintuch, 1986, Clark et al, 1981, Cowan, 1985, Ferrara, 1980, Janssen, 1987, Lee and Un, 1989, Rafaely 1997, Saeef and Wiley, 1996, Sommen et al, 1987, and Shynk, 1992].

We now show how the BLMS can be implemented in the frequency domain. It is known that linear convolution in the time domain corresponds to multiplication in the frequency domain, which can be realised using the discrete Fourier transform (DFT). However, when multiplying filters in the frequency domain with frequency domain data, the result is a circular convolution. This circular convolution would mean that the data is periodically repeated in the time domain. However, the filtering problem requires linear convolution, and correct convolution of the data must be achieved by adding some extra conditioning to the problem. This can be done by applying overlap-save method [Shynk, 1992] which implements the FDLMS with linear convolution. Others methods such as the overlap-add method have also been developed [Shynk, 1992]. In this thesis only the overlap-save method is applied.

The overlap-save method performs linear convolutions using FFT algorithms by discarding part of the DFT product, and keeping only a subset of the results. This can be computed by performing DFT of twice the length $2N$, where for the input signal, the first block is the old data and second block is new. Then when a frequency domain multiplication occurs, the first half of the output corresponds to a circular convolution, while the second half is exactly the same as a linear convolution. Then the error signal needs to be set with zeros since the first block of the output of the multiplication with the input is discarded in the causality constraint. The causality constraint discards the second half of the output as opposed to the first, basically the reverse of a convolution, and then zero extends to bring the vector to the same size as the input ready for the filtering. Note a block length of $L = N$ will be used.

The formulation in the frequency domain of the BLMS algorithm [Shynk, 1992] is now considered, discussing the update of the filter coefficients. The frequency domain filter coefficient matrix can be written as

$$\boldsymbol{W}(k) = \boldsymbol{F}\,[w_0(k) \quad \cdots \quad w_{N-1}(k) \quad \boldsymbol{0}]^{\mathrm{T}} \tag{3.30}$$

where $\mathbf{0}$ is the vector containing N zeros representing the non-causal part of the filter, used when applying the overlap-save method to produce the N output samples of the linear convolution. $\boldsymbol{F}$ is the DFT matrix with elements $F_{ml} = e^{-2\pi ml/N}$ and $\boldsymbol{W}(k)$ is therefore a $2N\times 1$ vector of the complex filter coefficients in the frequency domain.

Two matrices are introduced to simplify notation:

$$\boldsymbol{D}_{\mathrm{f}} = [\,\boldsymbol{I}_N \quad \boldsymbol{0}_N\,], \quad \boldsymbol{D}_{\mathrm{b}} = [\,\boldsymbol{0}_N \quad \boldsymbol{I}_N\,] \tag{3.31}$$

where $\boldsymbol{I}_N$ is $N\times N$ identity matrix and $\mathbf{0}_N$ is the $N\times N$ zero matrix [Lim and Un, 1996]. These are used in conjunction with DFT operator to ensure linear convolution. So, the matrix of filter coefficients (3.30) can be rewritten as

$$\boldsymbol{W}(k) = \boldsymbol{F}\,\boldsymbol{D}_{\mathrm{f}}^{\mathrm{T}}\,\boldsymbol{w}(k) \tag{3.32}$$

The input signal is defined by:

$$\boldsymbol{X}(k) = diag\left\{\boldsymbol{F}\left[x(kN-N) \quad \cdots \quad x(kN-1) \quad x(kN) \quad \cdots \quad x(kN+N-1)\right]^{\mathrm{T}}\right\} \tag{3.33}$$

$\boldsymbol{X}(k)$ is a $2N\times2N$ diagonal matrix of the reference signal, this contains $2N$ elements, N from the previous block and N from the current block. This is, hence, overlapping half of the vector and providing only N new data points for each iteration.

The time-domain output of the filtered signal can now be defined as:

$$\begin{aligned} y(k) &= \boldsymbol{F}^{-1}\,\boldsymbol{X}(k)\,\boldsymbol{W}(k) \\ &= [y(kN-N) \quad \cdots \quad y(kN-1) \quad y(kN) \quad \cdots \quad y(kN+N-1)]^{\mathrm{T}} \end{aligned} \tag{3.34}$$

where $\boldsymbol{F}^{-1}$ is the inverse of the DFT, defined as $\boldsymbol{F}^{-1} = (1/2N)\boldsymbol{F}^{\mathrm{H}}$ where the superscript H denotes complex conjugate transpose. The first N output samples in our vector $y(k)$ are not used as they correspond to previous data points. The last N elements correspond to the current data points. These two facts will be used to ensure that a linear convolution is obtained when using the inverse DFT to transform data back to the time domain [Shynk, 1992].

The filter coefficients are fixed for the duration of the block, and the gradient of the algorithm is a linear convolution. To compute the error, $\boldsymbol{e}(k)$ is first calculated in the time domain

$$\boldsymbol{e}(k) = [e(kN),\ldots,e(kN+N-1)]$$

which is then transformed to the frequency domain as

$$\boldsymbol{E}(k) = \boldsymbol{F}\,[\mathbf{0} \quad e(kN) \quad \cdots \quad e(kN+N-1)\,]^{\mathrm{T}} = \boldsymbol{F}\,\boldsymbol{D}_{\mathrm{b}}^{\mathrm{T}}\,\boldsymbol{e}(k) \tag{3.35}$$

The presence of the zeros is to ensure linear rather than circular convolution when transforming the problem back to the time domain.

The gradient in the frequency domain can be calculated [Shynk, 1992] as shown below:

$$\hat{\nabla}(k) = \sum_{m=0}^{N-1} e(n+m)\boldsymbol{x}(n+m) = \boldsymbol{F}^{-1}[\boldsymbol{X}^{\mathrm{H}}(k)\boldsymbol{E}(k)] \tag{3.36}$$

As with the output signal, some elements of the gradient vector (3.36) are zeros, namely the last N elements. Again, this is to ensure that linear convolution is achieved. It can be shown that the block gradient estimate $\hat{\nabla}(k)$ has $2N$ elements, but only the first N components are used because the latter N correspond to updating the non-causal coefficient of $\boldsymbol{w}(k)$, which are assumed to be zero to achieve linear convolution.

Now, the update equation for the BLMS in frequency domain can be defined in matrix form as:

$$\boldsymbol{W}(k+1) = \boldsymbol{W}(k) + 2\boldsymbol{F}\boldsymbol{g}\boldsymbol{F}^{-1}\mu(k)\boldsymbol{X}^{\mathrm{H}}(k)\boldsymbol{E}(k) \tag{3.37}$$

where

$$\boldsymbol{g} = \begin{bmatrix} \boldsymbol{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{3.38}$$

This choice of matrix $\boldsymbol{g}$ allows the non-causal elements of the time domain gradient to be set to zero. This $\boldsymbol{g}$ must not be confused with the gradient $\boldsymbol{g}$ that defined in chapter 2. The matrix $\boldsymbol{F}^{1}\boldsymbol{g}\boldsymbol{F}$ represents a gradient constraint, which is introduced to ensure linear convolution between $\boldsymbol{W}(k)$ and $\boldsymbol{X}(k)$. If this matrix is omitted, as it can be, the accuracy of the algorithm will suffer due to the presence of circular convolution. This may lead to a solution that is not the Wiener filter, which is undesirable. The gradient constraint is usually defined as a restriction on the signals in the time domain, and then the results transformed to the frequency domain. The complete frequency domain BLMS is shown in figure 3.3 [Shynk, 1992].

Clark et al (1981) has been shown that the convergence coefficient of the BLMS in frequency domain is the same as that of the BLMS in the time domain given the same block length and filter length. $\mu(k)$ is equal to μ_B for all k to implement the BLMS algorithm. For long block lengths, non-stationary input signals or large eigenvalues spread, poor performance of the convergence can occur since $\mu(k)$ is equal to μ_B for all k. A frequency dependent convergence coefficient can be used for faster convergence. It is known as bin-normalized frequency domain LMS (BN-FDLMS), which will be presented in the next section. Note that although matrix equations were used here, in practice the FDLMS and BN-FDLMS described are implemented using vectors only.

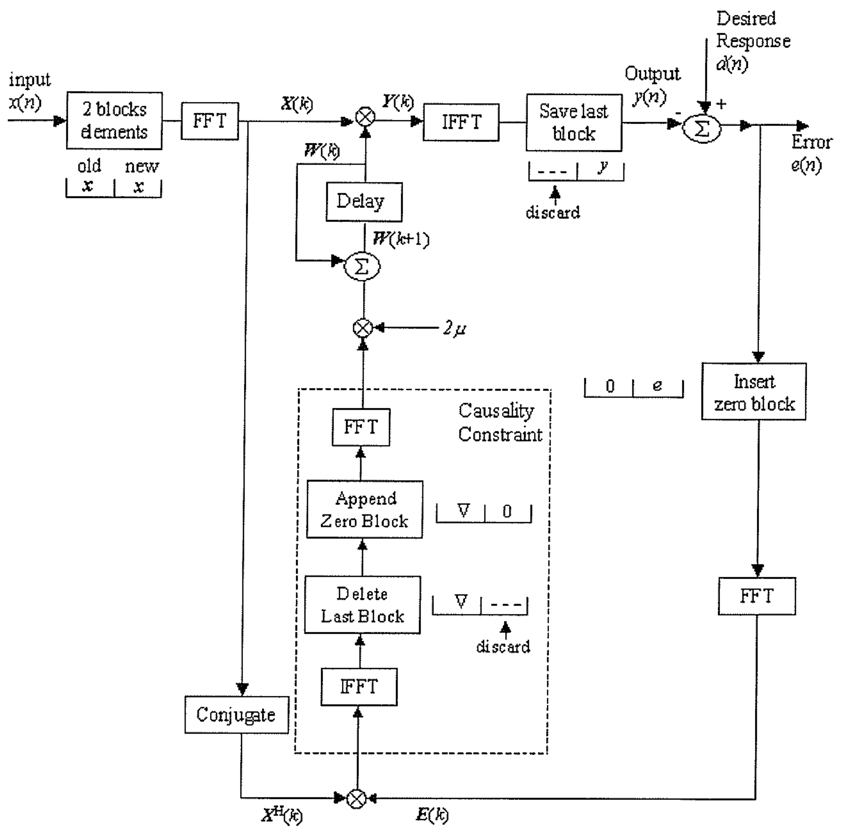


Figure 3.3: Overlap-save frequency domain LMS algorithm.

3.6 The Bin-normalized frequency domain LMS algorithm

The BN-FDLMS is a further development from FDLMS, which improves the convergence speed by applying frequency dependent convergence coefficient at each bin. Each convergence coefficient is varied according to the power of the reference signal in that frequency bin. The general form of the update equation of BN-FDLMS algorithm is expressed as follow which is given by Sommen et al (1987) and Shynk (1992):

$$\boldsymbol{W}(k+1) = \boldsymbol{W}(k) + 2\boldsymbol{\mu}(k)\boldsymbol{F}\boldsymbol{g}\boldsymbol{F}^{-1}\boldsymbol{X}^{\mathrm{H}}(k)\boldsymbol{E}(k) \tag{3.39}$$

where $\mu(k)$ is diagonal time-varying matrix that contains the coefficients $\mu_m(k)$, where $m = 0, \ldots, M\text{-}1$. In order to make all weights converge at the same rate, the adaptation constant can be made different for each frequency bin according to $\mu_m(k)$. The coefficient $\mu_m(k)$ is defined as

$$\mu_m(k) = \frac{\mu}{\hat{\mathrm{P}}_m(k)} \tag{3.40}$$

where μ is a fixed scalar, m is a frequency bin and $\hat{\mathrm{P}}_m(k)$ is the power estimate of the input signal at the mth bin. It has been studied by Cowan (1985), Jansssen (1987), Sommen et al (1987) and shown to significantly improve the convergence rate of the FDLMS. $\mu_m(k)$ is to compensate for the power variation by using convergence coefficients that are inversely proportional to the power levels in the frequency bins. The power estimate $\hat{\mathrm{P}}_m(k)$ is calculated as

$$\hat{\mathrm{P}}_m(k) = \beta\, \hat{\mathrm{P}}_m(k-1) + (1-\beta)\left|X_m(k)\right|^2 \tag{3.41}$$

where β is a forgetting scalar within the range $0 < \beta < 1$, and the initial value $\hat{\mathrm{P}}_m(0)$ is chosen which depends on the initial power spectrum estimate of $X_m(k)$. The convergence coefficients matrix can then be written as

$$\boldsymbol{\mu}(k) = \mu\, \mathrm{diag}\left\{\hat{\mathrm{P}}_0^{-1}(k),\ \cdots\ ,\hat{\mathrm{P}}_{2N\text{-}1}^{-1}(k)\right\} \tag{3.42}$$

$\hat{\mathrm{P}}_m(k)$ can be presented as $\hat{\mathbf{P}}(k) = diag[\hat{\mathrm{P}}_0(k),\ldots,\hat{\mathrm{P}}_{2N-1}(k)]$ is a diagonal matrix of size $2N\times2N$ which is the power spectrum estimate of the reference signal using exponential or rectangular averaging.

A convergence coefficient in equation (3.39) is allocated to each frequency bin depending on the power of the input signal at that bin. Similar time-constants occur for every mode of convergence when the power estimate of the reference signal is accurate (see Cowan, 1985). Sommen et al (1987) produced expressions which determine the maximum value of μ to ensure stability of the BN-FDLMS and showed that the maximum value is dependent on β and further the stable range for μ is given as (3.43). Note that for the FDLMS multiplying μ by N means the latter has the same maximum convergence coefficient as the LMS.

$$0 < \mu N < \frac{4}{E_{1,1}(\beta) + \dfrac{E_{2,2}(\beta)}{E_{1,1}(\beta)} + \dfrac{E_{1,0}(\beta)E_{1,2}(\beta)}{E_{1,1}(\beta)}} = \alpha_0 \tag{3.43}$$

where in the limit of infinite iterations, or $k = \infty$ and $\beta \to 1$, $E_{1,1}(\beta) \approx 1$, $E_{1,0}(\beta)E_{1,2}(\beta) \approx 1$ and $E_{2,2}(\beta) \approx 2$ (Sommen et al, 1987). This reduces equation (3.43) to $0 < \mu N < 1$, where $E_{p,q}$ is given by:

$$E_{p,q} = E((\hat{\mathrm{P}}_m(k))^{-q}|X_m(k)|^{2p}) = \lim_{n\to\infty} E\left[\frac{|X_m(k)|^{2p}}{\left((1-\beta)\sum_{j=0}^{n}|\beta^j X_m(k-j)|^2\right)^q}\right] \tag{3.44}$$

Further detail of the BN-FDLMS can be found in Sommen et al, (1987) and Jassen (1987). The BN-FDLMS, with some additional complexity on the FDLMS associated with the power estimation, can improve the convergence of the FDLMS. The convergence speed of the BN-FDLMS will be examined in the simulation in part III.

3.7 Conjugate gradient algorithm in adaptive filtering

In this section, we present the conjugate gradient algorithm (CGA) for adaptive filtering applications. It has been studied by Boray and Srinath (1992), and showed that it has the ability to perform sample-by-sample updating of the filter coefficients, and its performance can be comparable to LMS based algorithms, giving fast convergence rate [also see Chang and Willson, 1996]. Over recent years many algorithms have been developed to solve the adaptive filtering problems. LMS type algorithms have been widely used due to their simplicity of use and low computational complexity. However, the convergence rate depends on the eigenvalue spread of the autocorrelation matrix $\boldsymbol{R}$, as presented in section 3.3-3.5. Conjugate gradient type algorithms belong to a family of algorithms that attempt to accelerate the slow convergence rate of the method of steepest descent while avoiding manipulations of matrix $\boldsymbol{R}$ associated with Newton's method. It was therefore chosen in this study together with the BN-FDLMS, although other algorithms could also be used (see section 2.6 for examples).

Similar to the LMS or other adaptive filtering algorithms, the gradient estimation needs to be calculated in order to update the filter coefficients and to obtain the minimum. In the CGA, the gradient estimate is obtained by averaging the instantaneous gradient estimate over a specified window size. The CGA's are flexible in that it is possible to choose the size of the gradient average window to change the complexity of the calculations, and hence choose the level of accuracy and efficiency required. The stability can be affected by the gradient of the descent direction in the CGA. The problem of finding the optimal $\boldsymbol{w}^*$ is briefiy reviewed, then the formulation of the direction vectors $\boldsymbol{b}_k$ is presented. Issues arising from using adaptive filters are then considered, namely using the instantaneous value of the gradient.

The problem is to solve $\boldsymbol{Rw} = \boldsymbol{p}$, as before $\boldsymbol{R}$ is the $N \times N$ correlation matrix of the input data, $\boldsymbol{w}$ are the filter coefficients and $\boldsymbol{p}$ is the cross correlation vector between the input signal and the desired response signal. The approach in the CGA is to obtain a set of linearly independent direction vectors $\boldsymbol{b}_0, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_{N-1}$ which are conjugate with respect to $\boldsymbol{R}$, that is, $\boldsymbol{b}_i^{\mathrm{T}} \boldsymbol{R} \boldsymbol{b}_j = 0$ for all $i \neq j$, so that the solution $\boldsymbol{w}^*$ can be expressed as a linear combination of search direction vectors [Chang and Willson, 1996 and Lim and Un, 1996],

$$\boldsymbol{w}^* = \alpha_0 \boldsymbol{b}_0 + \alpha_1 \boldsymbol{b}_1 + \cdots + \alpha_{N-1} \boldsymbol{b}_{N-1} \tag{3.45}$$

where

$$\alpha_i = \frac{\boldsymbol{b}_i^T \boldsymbol{p}}{\boldsymbol{b}_i^T \boldsymbol{R} \boldsymbol{b}_i} \tag{3.46}$$

where N is the length of the filter and α is the step size for each search direction (refer to section 2.5). The problem is then to compute the appropriate search direction vectors and the step size α. The notation $\boldsymbol{b}$ represents the search direction and $\boldsymbol{p}$ represents the cross correlation vector, which must not be confused the notations used in chapter section 2.5. The idea of CGA is an iterative N steps process to find the minimum $\boldsymbol{w}^*$, by adding $\alpha_k \boldsymbol{b}_k$ at the kth iteration. The update equation of CGA is written as:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \alpha_k \boldsymbol{b}_k \tag{3.47}$$

The solution $\boldsymbol{w}^*$ is approached by a recursive formula (3.47). Each successive $\boldsymbol{b}_k$ is calculated such that it is conjugate to the preceding linear combination of direction vectors, $\boldsymbol{b}_{k-1}, \boldsymbol{b}_{k-2}, \ldots, \boldsymbol{b}_0$. The initial step $\alpha_0 \boldsymbol{b}_0$ is the same as the steepest descent method. Each step after the initial step is a linear combination of the previous direction vectors and the current gradient, and is calculated by:

$$\boldsymbol{b}_{k+1} = -\boldsymbol{g}_{k+1} + \beta_k \boldsymbol{b}_k \tag{3.48}$$

The negative gradient is taken at each step, and added to the existing linear combination of direction vectors to evaluate the new search direction $\boldsymbol{b}_{k+1}$. β_k is chosen to ensure that $\boldsymbol{b}_{k+1}$ is conjugate to $\boldsymbol{R}$ with respect to all the preceding direction vectors $\boldsymbol{b}_{k-1}, \boldsymbol{b}_{k-2}, \ldots, \boldsymbol{b}_0$.

The CGA is now considered for minimising the MSE for adaptive filters. If an instantaneous value of the gradient is used, it would not be possible to find a search direction conjugate to the initial step. Since the data is being continuously updated, the search direction that the

CGA follows may not be conjugate to the next sample of data. Hence, a gradient estimate is used, taking an average of the instantaneous values. This can be done over a specified data window of past values. The choice of the size of this window will affect convergence speed of the algorithm [Boray and Srinath, 1992 and Lim and Un, 1993].

One type of data window is Finite Sliding Data Window. Here, the size of the data window is fixed at an integer M. We estimate $\boldsymbol{R}$ and $\boldsymbol{p}$, the auto-correlation and cross-correlation matrices [Chang and Willson, 1995], as follows:

$$\boldsymbol{R}(n) = \frac{1}{M} \sum_{j=n-M+1}^{n} \boldsymbol{x}(j)\boldsymbol{x}(j)^{\mathrm{T}} \tag{3.49}$$

$$\boldsymbol{p}(n) = \frac{1}{M} \sum_{j=n-M+1}^{n} d(j)\boldsymbol{x}(j) \tag{3.50}$$

The gradient vector ∇ is then

$$\nabla = \boldsymbol{p}(n) - \boldsymbol{R}(n)\boldsymbol{w}(n) \tag{3.51}$$

$$= \frac{1}{M} \sum_{j=n-M+1}^{n} [(d(j) - \boldsymbol{w}(j)^{\mathrm{T}} \boldsymbol{x}(j))\boldsymbol{x}(j)] \tag{3.52}$$

When M is less than the length of the input vector, then (3.52) above will be more efficient to implement than the formulation (3.51) [Chang and Willson, 1995]. However, it can be shown that although this method will give a fast convergence rate, it has a high level of misadjustment [Chang and Willson, 1995].

Another type of data window is the Exponentially Decaying Data Window [Chang and Willson, 1995]. The estimates of the auto-correlation matrix $\boldsymbol{R}$ and the cross-correlation matrix $\boldsymbol{p}$ are:

$$\boldsymbol{R}(n) = \lambda_f \boldsymbol{R}(n-1) + \boldsymbol{x}(n)\boldsymbol{x}(n)^{\mathrm{T}} \tag{3.53}$$

$$\boldsymbol{p}(n) = \lambda_f \boldsymbol{p}(n-1) + d(n)\boldsymbol{x}(n) \tag{3.54}$$

where λ_f is the forgetting factor.

A recursion for the gradient vector can be found, resulting in:

$$\begin{aligned}\nabla &= \boldsymbol{p}(n) - \boldsymbol{R}(n)\boldsymbol{w}(n) \\ &= \lambda_f \nabla(n-1) - \alpha(n)\boldsymbol{R}(n)\boldsymbol{p}(n) + \boldsymbol{x}(n)[d(n) - \boldsymbol{x}(n)^{\mathrm{T}}\boldsymbol{w}(n-1)]\end{aligned} \tag{3.55}$$

This method has an advantage over the finite sliding data window method in that has a fast convergence rate whilst keeping a low level of misadjustment [Chang and Willson, 1995].

When not using block processing of input data, it is important to periodically reset the search direction with the true gradient to ensure convergence to the solution. The performance of the algorithm will be affected by the frequency of the resets. It should be noted that if a set of direction vectors do not increase the cost function, then the algorithm will converge to the minimum since the CGA takes an initial steepest descent step each time it is reinitialised.

Termination criteria for the algorithm must also be considered. One choice would be to terminate when a solution sufficiently close to the minimum has been reached. Alternatively, the number of iterations could be limited. In most cases, the CGA must be run several times before convergence to the minimum is achieved. This is computationally inefficient for sample-by-sample processing of adaptive filters, but is better for block-by-block updating, as will be shown in the next section.

3.8 Block conjugate gradient algorithm

The BCGA proposed by Lim and Un (1996) uses the block MSE as a performance criterion. The block MSE (BMSE) and its estimate are first considered. The algorithm will be derived using the least-squares solution so that an estimate of the BMSE is minimised along the conjugate direction vectors that are generated to be mutually conjugate. The standard adaptive filtering problem is now considered. A filter of length N, and blocks of size L are used. Using j as a block index, an $L\times N$ input signal $\boldsymbol{X}_j$, an $L\times 1$ desired signal $\boldsymbol{d}_j$ and a $N\times 1$ filter coefficients vector $\boldsymbol{w}$. The structure of the input signal $\boldsymbol{X}_j$ is as follows: (details refer to Lim and Un, 1996)

$$\boldsymbol{X}_j = [\boldsymbol{x}_{(j-1)L+1}, \boldsymbol{x}_{(j-1)L+2}, \ldots, \boldsymbol{x}_{jL}]^{\mathrm{T}} \tag{3.56}$$

with

$$\boldsymbol{x}_l = [x_l, x_{l-1}, \ldots, x_{l-N+1}]^{\mathrm{T}} \tag{3.57}$$

where $l = (j-1)L+1, (j-1)L+2, \ldots, jL$. The error is defined as

$$\boldsymbol{e}_j = \boldsymbol{d}_j - \boldsymbol{X}_j \boldsymbol{w}_j \tag{3.58}$$

The BMSE as the expectation of the block averaged errors is defined as follows:

$$BMSE = \frac{1}{L} E[\boldsymbol{e}_j^{\mathrm{T}} \boldsymbol{e}_j] \tag{3.59}$$

The above can also be written as

$$BMSE = \frac{1}{L}(E[\boldsymbol{d}_j^{\mathrm{T}} \boldsymbol{d}_j] - 2\boldsymbol{p}^{\mathrm{T}} \boldsymbol{w} + \boldsymbol{w}^{\mathrm{T}} \boldsymbol{R} \boldsymbol{w}) \tag{3.60}$$

In the above, $\boldsymbol{R} = E[\boldsymbol{X}_j^{\mathrm{T}} \boldsymbol{X}_j]$ is the $N \times N$ block auto-correlation matrix and $\boldsymbol{p} = E[\boldsymbol{X}_j^{\mathrm{T}} \boldsymbol{d}_j]$ is the $N \times 1$ block cross-correlation vector between the block desired signal $\boldsymbol{d}$ and the block input signal $\boldsymbol{X}$.

Next, an estimate of the BMSE is defined, using the least-squares approach, to formulate the BCGA.

$$\begin{aligned} Est.BMSE &= \frac{1}{L}([\boldsymbol{d}_j^{\mathrm{T}} \boldsymbol{d}_j] - 2[\boldsymbol{d}_j^{\mathrm{T}} \boldsymbol{X}_j]\boldsymbol{w} + \boldsymbol{w}^{\mathrm{T}}[\boldsymbol{X}_j^{\mathrm{T}} \boldsymbol{X}_j]\boldsymbol{w}) \\ &= \frac{1}{L}[\boldsymbol{d}_j - \boldsymbol{X}_j \boldsymbol{w}]^{\mathrm{T}}[\boldsymbol{d}_j - \boldsymbol{X}_j \boldsymbol{w}] \end{aligned} \tag{3.61}$$

In the above, the expectation operators of the terms $E[\boldsymbol{X}_j^{\mathrm{T}} \boldsymbol{X}_j]$ and $E[\boldsymbol{d}_j^{\mathrm{T}} \boldsymbol{X}_j]$ have been dropped in order to form instantaneous approximation of those terms. This estimate is not as rough as it may at first seem. Since blocks of data are being collected, the instantaneous estimates will average out over the block. Whereas the BMSE is constant, *Est. BMSE* may vary from block to block. If the term $\boldsymbol{X}_j^{\mathrm{T}} \boldsymbol{X}_j$ is positive definite, then the *Est. BMSE* above can be regarded as a quadratic function with a unique minimum. The minimum of the *Est. BMSE* can therefore be found by:

$$\boldsymbol{d}_j - \boldsymbol{X}_j \boldsymbol{w} = \boldsymbol{0} \tag{3.62}$$

where $\boldsymbol{0}$ is a vector of zeros. $\boldsymbol{X}_j^{\mathrm{T}} \boldsymbol{X}_j$ in the *Est. BMSE* will be positive definite, as required, if $L \geq N$ and $\boldsymbol{X}_j$ is a matrix that is non-singular [Lim and Un, 1996]. Under this assumption, it is appropriate to use the formulation above. Since iterations will be performed to find the minimum error in each block, the error update equation for the j^{th} block is defined as:

$$\boldsymbol{e}_{k+1} = \boldsymbol{d}_j - \boldsymbol{X}_j \boldsymbol{w}_{k+1} \tag{3.63}$$

Since the update for the CGA is given by $\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \alpha_k \boldsymbol{b}_k$, this is substituted into the above equation, to obtain a recursive estimate for $\boldsymbol{e}_{k+1}$:

$$\boldsymbol{e}_{k+1} = \boldsymbol{e}_k - \alpha_k \boldsymbol{X}_j \boldsymbol{b}_k \tag{3.64}$$

Next, $\boldsymbol{g}_{k+1}$ is defined as the gradient vector of the *Est. BMSE* at the k^{th} iteration. This is given by [Lim and Un, 1996]:

$$\boldsymbol{g}_{k+1} = -\nabla Est.BMSE = \frac{2}{L} \boldsymbol{X}_j^{\mathrm{T}} \boldsymbol{e}_{k+1} \tag{3.65}$$

The above vector will be orthogonal to $\boldsymbol{b}_k$ because the update equation for $\boldsymbol{w}_{k+1}$ minimises the *Est. BMSE* in the direction $\boldsymbol{b}_k$. This is written as:

$$\boldsymbol{b}_k^{\mathrm{T}} \boldsymbol{g}_{k+1} = 0 \tag{3.66}$$

Using the above equations, the optimal step size for iterative step k can be written as:

$$\alpha_k = \frac{L}{2} \frac{\boldsymbol{b}_k^{\mathrm{T}} \boldsymbol{g}_k}{\boldsymbol{b}_k^{\mathrm{T}} \boldsymbol{X}_j^{\mathrm{T}} \boldsymbol{X}_j \boldsymbol{b}_k} \tag{3.67}$$

The direction vector $\boldsymbol{b}_{k+1}$ is obtained by adding a multiple of the previous direction vector to the gradient vector. The update equation for the direction vector is written again here:

$$\boldsymbol{b}_{k+1} = \boldsymbol{g}_{k+1} + \beta_k \boldsymbol{b}_k \tag{3.68}$$

The constant is chosen to ensure that each direction vector is conjugate to the previous vector. That is, β_k is chosen such that $\boldsymbol{b}_k^{\mathrm{T}} \boldsymbol{X}_j^{\mathrm{T}} \boldsymbol{X}_j \boldsymbol{b}_{k+1} = 0$.

Taking these two conditions β_k can be defined as:

$$\beta_k = -\frac{\boldsymbol{b}_k^{\mathrm{T}} \boldsymbol{X}_j^{\mathrm{T}} \boldsymbol{X}_j \boldsymbol{g}_{k+1}}{\boldsymbol{b}_k^{\mathrm{T}} \boldsymbol{X}_j^{\mathrm{T}} \boldsymbol{X}_j \boldsymbol{b}_k} \tag{3.69}$$

The above requires an extra convolution of $\boldsymbol{X}_j$ with $\boldsymbol{X}_j\ \boldsymbol{g}_{k+1}$. The above expression can be simplified to:

$$\beta_k = \frac{\boldsymbol{g}_{k+1}^{\mathrm{T}} \boldsymbol{g}_{k+1}}{\boldsymbol{b}_k^{\mathrm{T}} \boldsymbol{g}_k} \tag{3.70}$$

From the above expressions, it should be noted that $\boldsymbol{b}_k^{\mathrm{T}} \boldsymbol{g}_k = \boldsymbol{g}_k^{\mathrm{T}} \boldsymbol{g}_k$, and the expressions for the step size and direction vector coefficient can be written as:

$$\alpha_k = \frac{L}{2} \frac{\boldsymbol{g}_k^{\mathrm{T}} \boldsymbol{g}_k}{\boldsymbol{b}_k^{\mathrm{T}} \boldsymbol{X}_j^{\mathrm{T}} \boldsymbol{X}_j \boldsymbol{b}_k}$$

and (3.71)

$$\beta_k = \frac{\boldsymbol{g}_{k+1}^{\mathrm{T}} \boldsymbol{g}_{k+1}}{\boldsymbol{g}_k^{\mathrm{T}} \boldsymbol{g}_k}$$

***Block conjugate gradient algorithm*:**

The BCGA can now be formulated. The number of blocks is set to be j, and the maximum number of iterations for each block to be M. For each block, a value for $\boldsymbol{w}_0$ is chosen and the initial error vector and initial direction vectors computed. A termination criterion bounding the value of $\boldsymbol{g}_{k+1}^{\mathrm{T}} \boldsymbol{g}_{k+1}$ can also be set to prevent unnecessary iterations [Lim and Un, 1996]. This can be done by determining the level of accuracy of the solution for the given application. The initial vectors are given by:

$$\boldsymbol{e}_0 = \boldsymbol{d}_j - \boldsymbol{X}_j \boldsymbol{w}_0 \tag{3.72}$$

and

$$\boldsymbol{b}_0 = \boldsymbol{g}_0 = \frac{2}{L} \boldsymbol{X}_j^{\mathrm{T}} \boldsymbol{e}_0 \tag{3.73}$$

For each of the first (M-1) iterations in each block, the filter coefficients can be updated using

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \alpha_k \boldsymbol{b}_k \tag{3.74}$$

On the final M^{th} iteration of the block, the updated error, gradient and direction vectors are computed as:

$$\boldsymbol{e}_{k+1} = \boldsymbol{e}_k - \alpha_k \boldsymbol{X}_j \boldsymbol{b}_k, \tag{3.75}$$

$$\boldsymbol{g}_{k+1} = \frac{2}{L}\boldsymbol{X}_j^{\mathrm{T}} \boldsymbol{e}_{k+1}, \tag{3.76}$$

$$\boldsymbol{b}_{k+1} = \boldsymbol{g}_{k+1} + \beta_k \boldsymbol{b}_k. \tag{3.77}$$

These are then substituted into the first iteration of the filter coefficient update equation for the next block. This is continued for j blocks [Lim and Un, 1996].

Since $\boldsymbol{X}_j^{\mathrm{T}}\boldsymbol{X}_j$ is positive definite and symmetric, and if the number of distinct eigenvalues of the matrix is less than the filter length, then the minimum point can be written as the sum of the initial condition and a linear combination of the direction vectors and the direction coefficients as

$$\boldsymbol{w}^* = \boldsymbol{w}_0 + \alpha_0 \boldsymbol{b}_0 + \ldots + \alpha_{M-1}\boldsymbol{b}_{M-1}. \tag{3.78}$$

The algorithm will clearly reach this minimum after M iterations in the direction $\boldsymbol{b}_k$. It is rarely practical to explicitly calculate and know the eigenvalues of the matrix. The number of iterations can be chosen to suit our application. As iterations begin, a high value of M, up to a maximum number of iterations equal to the filter length, will result in a high convergence rate, but also high computational complexity. Conversely, a low value of M will provide a more computationally efficient algorithm, with a lower convergence rate. As the minimum is approached the number of iterations per block will tend to one [Boray and Srinath, 1992].

An essential condition in the BCGA is that the successive $\boldsymbol{b}_{k+1}$ direction vectors are conjugate to each other. This is to ensure that the estimate of the BMSE can be treated as a quadratic function, and that the direction vectors are hence always pointing towards the solution $\boldsymbol{w}^*$. When the update equation is very near to the solution, the accuracy of the final result may suffer if iterations continue [Lim and Un, 1996].

3.9 Frequency domain block conjugate gradient algorithm

In section 3.5 it was shown that when implementing algorithms in the frequency domain, to obtain better computational efficiency, the overlap-save method can be used to avoid circular convolution when transforming back to the time domain. The overlap-save method will also be used to implement the BCGA in the frequency domain. An overlap of half of the vector will be used and the filter length and block size will be set equal. The matrices $\boldsymbol{D}_{\mathrm{f}}$ and $\boldsymbol{D}_{\mathrm{b}}$, are as defined in section 3.5, to simplify notation. These will be used in conjunction with the DFT operator to ensure linear convolution [Lim and Un, 1996].

The frequency domain BCGA is now stated as follow. The matrix of filter coefficients, at block j and at iteration k, of size $2N\times1$, is as given in section 3.5. Zeros have been added to the first N columns of the matrix, then transformed to the frequency domain using the DFT. The filter output signal transformed to the frequency domain is also as written in section 3.5. It is $2N\times2N$, and is a diagonal matrix.

The $2N\times1$ error vector in the frequency domain is given by:

$$\boldsymbol{E}_k = \boldsymbol{F}\boldsymbol{D}_{\mathrm{b}}^{\mathrm{T}}\boldsymbol{e}_k \tag{3.79}$$

The direction vectors in the frequency domain are defined by:

$$\boldsymbol{B}_k = \boldsymbol{F}\boldsymbol{D}_{\mathrm{f}}^{\mathrm{T}}\boldsymbol{b}_k \tag{3.80}$$

All the vectors have now been transformed to the frequency domain, with additional zero and identity matrices added in to ensure linear convolution. The frequency domain block conjugate gradient algorithm (FDCGA) can now be stated as follows.

To implement the algorithm in the frequency domain, set a value for $\boldsymbol{w}_0$ and take as initial vectors for the first block, the following [Lim and Un, 1996]:

$$\boldsymbol{X}_j = [\boldsymbol{D}_{\mathrm{f}}^{\mathrm{T}}\ \boldsymbol{D}_{\mathrm{b}}^{\mathrm{T}}]\boldsymbol{F}[\boldsymbol{x}_{j-1}^{\mathrm{T}}\ \boldsymbol{x}_j^{\mathrm{T}}]^{\mathrm{T}} \tag{3.81}$$

$$\boldsymbol{W}_0 = \boldsymbol{F}\boldsymbol{D}_{\mathrm{f}}^{\mathrm{T}}\boldsymbol{w}_0 \tag{3.82}$$

$$\boldsymbol{e}_0 = \boldsymbol{d}_j - \boldsymbol{D}_{\mathrm{b}}\boldsymbol{F}^{-1}[\boldsymbol{X}_j\boldsymbol{W}_0] \tag{3.83}$$

$$\boldsymbol{E}_0 = \boldsymbol{F}\boldsymbol{D}_\mathrm{b}^\mathrm{T}\boldsymbol{e}_0 \tag{3.84}$$

$$\boldsymbol{b}_0 = \boldsymbol{g}_0 = \frac{2}{L}\boldsymbol{D}_\mathrm{f}\boldsymbol{F}^{-1}[\boldsymbol{X}_j^\mathrm{H}\boldsymbol{E}_0] \tag{3.85}$$

For each iteration $k = 0, \ldots, M\text{-}1$ update the filter coefficients by calculating:

$$\boldsymbol{B}_k = \boldsymbol{F}\boldsymbol{D}_\mathrm{f}^\mathrm{T}\boldsymbol{b}_k \tag{3.86}$$

$$\alpha_k = \frac{L}{2}\frac{\boldsymbol{g}_k^\mathrm{T}\boldsymbol{g}_k}{[\boldsymbol{D}_\mathrm{b}\boldsymbol{F}^{-1}[\boldsymbol{X}_j\boldsymbol{B}_k]]^\mathrm{T}[\boldsymbol{D}_\mathrm{b}\boldsymbol{F}^{-1}[\boldsymbol{X}_j\boldsymbol{B}_k]]} \tag{3.87}$$

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \alpha_k\boldsymbol{b}_k \tag{3.88}$$

For each iteration except $k = M\text{-}1$, calculate:

$$\boldsymbol{e}_{k+1} = \boldsymbol{e}_k - \alpha_k\boldsymbol{D}_\mathrm{b}\boldsymbol{F}^{-1}[\boldsymbol{X}_j\boldsymbol{B}_k] \tag{3.89}$$

$$\boldsymbol{E}_{k+1} = \boldsymbol{F}\boldsymbol{D}_\mathrm{b}^\mathrm{T}\boldsymbol{e}_{k+1} \tag{3.90}$$

$$\boldsymbol{g}_{k+1} = \frac{2}{L}\boldsymbol{D}_\mathrm{f}\boldsymbol{F}^{-1}[\boldsymbol{X}_j^\mathrm{H}\boldsymbol{E}_{k+1}] \tag{3.91}$$

$$\boldsymbol{b}_{k+1} = \boldsymbol{g}_{k+1} + \beta_k\boldsymbol{b}_k \text{ with } \beta_k = \frac{\boldsymbol{g}_{k+1}^\mathrm{T}\boldsymbol{g}_{k+1}}{\boldsymbol{g}_k^\mathrm{T}\boldsymbol{g}_k}. \tag{3.92}$$

As was the case in the time domain, a termination criterion on $\boldsymbol{g}_{k+1}^\mathrm{T}\boldsymbol{g}_{k+1}$ can be set to stop needless iterations. Finally, the above is substituted into the first iteration of the filter coefficient update equation for the subsequent block, and the process above repeated for blocks 2 to j. (see Lim and Un, 1996, for more detail).

For practical implementations, it can be shown that the FDCGA is more flexible for applications of adaptive filtering than the BCGA, as varying the value of M has a more noticeable impact on the convergence performance and computational complexity. When implementing the FDCGA, care needs to be taken setting the value of the parameters. In particular, if the stopping criterion is set too large, this may result in too few iterations being performed, especially in early blocks of data.

The FDCGA is obtained from the BCGA using the fast circular convolution technique. The convergence rates of the BCGA and FDCGA are equal, but the FDCGA is far superior in terms of computational complexity when long FIR filter is involved. This is due to the use of fast Fourier transforms in the implementation of the algorithm. Note that although matrix equations were used here, in practice the FDCGA described is implemented using vectors only.

3.10 Conclusions

Adaptive filters are able to track changes in input data; they can help to achieve stability and good performance in practical applications. Some adaptive filtering algorithms have been presented and we have reviewed the LMS, bin-normalized LMS and conjugate gradient algorithms in both time and frequency domains. The algorithms implemented in time domain have slow convergence in the case of colored reference signals. Also when the adaptive filter has a long impulse response, the computational complexity of algorithms is increased significantly. However, block implementation of FIR filter provides considerable computational savings and faster convergence. Furthermore the computational complexity can be greatly reduced if the algorithms are formulated in frequency domain such as the algorithms of FDLMS, BN-FDLMS and FDCGA presented in this chapter. The performance of the time and frequency domains algorithms is compared and further studied in following chapters. We then show in chapter 6 how the FDLMS, BN-FDLMS and FDCGA can be modified to include frequency domain constraints. Then the theoretical analysis of the convergence coefficient of the constrained FDLMS is conducted in chapter 7, the stability and performance are significantly improved.

Part II:

Algorithms

Chapter 4

Unconstrained optimisation with high-order FIR filters

4.1 Introduction

Over the next few chapters, constrained optimisation of adaptive filters for acoustic application in the frequency domain is studied. Long FIR filters are required since acoustic applications are often with very long impulse responses. Here three algorithms from the previous chapter are studied through simulations: steepest descent algorithm; Newton's method; conjugate gradient algorithm. The trade off between computational efficiency and rate of convergence of these three algorithms is evaluated.

The aim of this chapter is to investigate the unconstrained optimisation algorithms with long FIR filter. It is essential to understand unconstrained optimisation problems before we consider constrained optimisation problems, because in this work, penalty method will be used to convert the constrained problem into an unconstrained one. To simplify the analysis, simulations are conducted in an optimisation setting in the time domain with a known data set. The simulations used analytic gradients (not signals), to focus on the study of the various optimisation formulations and avoid stochastic gradient noise. Constrained adaptive filters with stochastic gradient are studied later in the thesis. The two systems studied are: system identification and prediction system. These systems are used because they are simple to analyse compared with equalisation system which is used later in part III. Various different filter lengths are used in order to further compare their performance. This will be done by comparing performance of three algorithms in minimising the MSE. The results of the simulations will be used to develop a constrained adaptive filter later in the thesis.

The simulations performed for this chapter were conducted using the mathematical program MATLAB, using a PC with a Pentium III 32-bit processor, 128 MB RAM, and Windows 98 (2nd edition) operating system. The time taken to run the simulations obviously depends on the hardware used.

4.2 Formulation of the steepest descent, Newton's and conjugate gradient methods

In this section, reformulated versions of the three optimisation algorithms discussed in part I of this thesis are set out. Note that the gradients of the systems used here are analytic gradients, which is not estimated from signals.

Steepest descent algorithm:

The updated value of the coefficient vector at time n+1 is defined by

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \frac{1}{2}\alpha(-\nabla(n)) \tag{4.1}$$

where α is step length, α controls the speed of convergence and is used in the update of the weight vector, with an optimal α given by

$$\alpha_k = \frac{\nabla_k^{\mathrm{T}}\nabla_k}{\nabla_k^{\mathrm{T}}\boldsymbol{R}\nabla_k} \tag{4.2}$$

$\nabla(n)$ is the gradient vector at time n, and the factor 1/2 is for convenience, since $-\nabla(n) = 2E[\boldsymbol{e}(n)\boldsymbol{x}(n)]$.

Newton's method:

Solving for the optimal weight vector it can be seen that:

$$\boldsymbol{w}^* = \boldsymbol{w}_0 - \frac{1}{2}\boldsymbol{R}^{-1}\nabla \tag{4.3}$$

where $\boldsymbol{w}_0$ is any given initial value. This result can be changed into a form of Newton's method, as follows:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \frac{1}{2}\boldsymbol{R}^{-1}\nabla_k \tag{4.4}$$

Then, Newton's method can be generalised by reintroducing the constant α:

$$w_{k+1} = w_k - \alpha \boldsymbol{R}^{-1} \nabla_k \tag{4.5}$$

The one-step formula can be obtained by setting $\alpha = \frac{1}{2}$, as above, which is the optimal step size. Otherwise, any other value of α in the stable range could be chosen.

Conjugate gradient algorithm:

The update equation is given by:

$$w_{k+1} = w_k + \frac{1}{2} \alpha_k \boldsymbol{b}_k$$

where (4.6)

$$\boldsymbol{b}_0 = -\nabla_0,\ \boldsymbol{b}_{k+1} = -\nabla_{k+1} + \beta_k \boldsymbol{b}_k,\ \beta_k = \frac{\nabla_{k+1}^{\mathrm{T}} \nabla_{k+1}}{\nabla_k^{\mathrm{T}} \nabla_k} \text{ and } \alpha_k = -\frac{\boldsymbol{b}_k^{\mathrm{T}} \nabla_k}{\boldsymbol{b}_k^{\mathrm{T}} \boldsymbol{R} \boldsymbol{b}_k}.$$

The derivation for these equations is set out in chapter 2. These three methods will now be applied to optimisation problems for very long FIR filters by minimising the MSE in the following two case studies: system identification and prediction problems. The performance of the algorithms will also be compared in the two case studies.

4.3 Algorithm analysis: System Identification simulation

One application of adaptive filters is system identification. In this case, the filter provides a linear model of an unknown filter *H*, representing the best fit. The same input data is used for the unknown filter *H* to provide the desired signal and for the filter to provide the output signal, so that the error can be calculated as the difference between the two. The system identification problem is illustrated in figure 4.1.

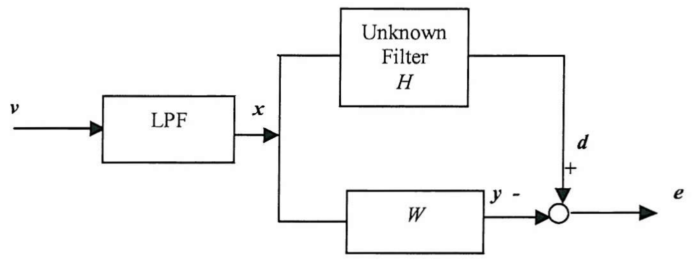


Figure 4.1 Block diagram of system identification.

The filter W attempts to match its output y with the output d of the unknown filter H. The error e is then the difference between the two signals; we use the MSE as a means of minimising the error. In this case, a 4th order butterworth low-pass filter with cutoff frequency 0.2 of the Nyquist frequency is used to generate the reference signals, x, from a white noise process of unit variance, v. This low-pass filter is illustrated in figure 4.2. The impulse response of H is generated by windowing a white noise sequence with an exponentially decaying sequence, illustrated in figure 4.3. The performance of the algorithms is compared for three different filter lengths: 512, 1024 and 2048. The simulations were conducted in MATLAB using a PC with a Pentium III 32-bit processor and 128 MB RAM. The stopping criteria of 0.1dB above the optimal minimum and a maximum of 1,000 iterations were set to enable simple comparison of the performance of the algorithms. Summary of the results is presented in table 4.1, and detailed results are presented. The 'floating point operations' in the results table 4.1 is the number of computation operations performed by the MATLAB program for each algorithm. Depending on how MATLAB codes are developed, we see that the number of floating point operations are not absolute values but an indication of values only. The reduction in error in the table 4.1 is calculated by dividing the mean square error by the mean square desired signal, that is $E[e^2]/E[d^2]$. The reduction in error, the time taken to run an algorithm and the number of floating point operations will be used to compare the performance of the three algorithms with different filter lengths. The floating point operations per iteration is also presented in order to reflect real-time efficiency, which shows computation per new sample.

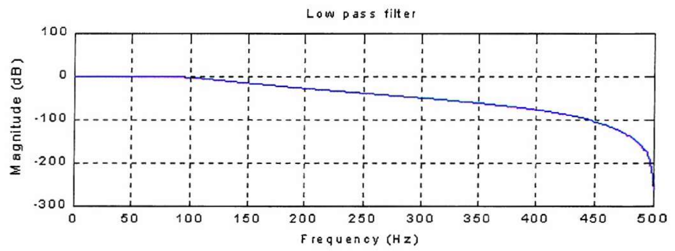


Figure 4.2: The 4th order butterworth low-pass filter with cutoff frequency 0.2.

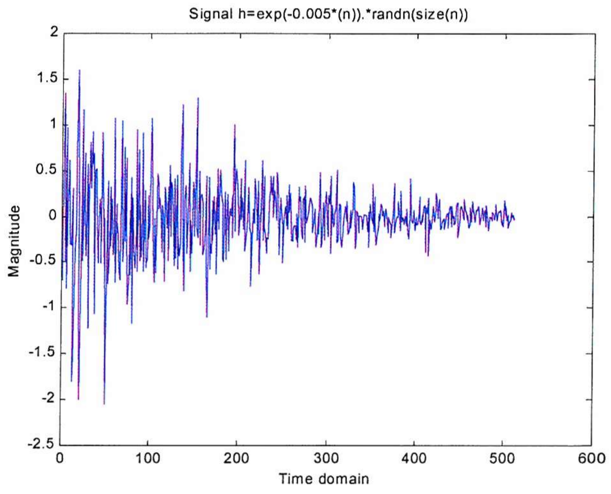


Figure 4.3: The impulse response of H is generated by windowing a white noise signal with an exponentially decaying sequence.

	Methods	**Iterations**	**Total floating point operations**	**Floating points per iteration**	**Time (sec)**	**Reduction in error (dB)**
(i)	Filter length $H = W = 512$, stopping criterion: 1,000 iterations					
	Newton's	1	273.6e+006	273.60e+006	8.52	-94.44
	Steepest descent	1000	1849.1e+006	1.84e+006	184.88	-43.63
	Conjugate gradient	1000	1852.2e+006	1.85e+006	147.37	-70.82
(ii)	Filter length $H = 512$ and $W = 450$, stopping criteria: 0.1dB above optimal minimum; maximum 1,000 iterations					
	Newton's	1	186.2e+006	186.20e+006	3.79	-21.43
	Steepest descent	1000	1432.7e+006	1.43e+006	106.32	-21.31
	Conjugate gradient	74	107.2e+006	1.45e+006	4.34	-21.32
(iii)	Filter length $H = 1024$ and $W = 600$ stopping criteria: 0.1dB above optimal minimum; maximum 1,000 iterations					
	Newton's	1	439.1e+006	439.10e+006	13.07	-27.42
	Steepest descent	1000	2536.6e+006	2.53e+006	208.16	-27.05
	Conjugate gradient	357	908.0e+006	2.54e+006	48.55	-27.31
(iv)	Filter length $H = 2048$ and $W = 800$ stopping criteria: 0.1dB above optimal minimum; maximum 1,000 iterations					
	Newton's	1	1036.5e+006	1036.50e+006	30.87	-33.85
	Steepest descent	1000	4502.5e+006	4.50e+006	369.21	-33.29
	Conjugate gradient	143	648.2e+006	4.53e+006	37.52	-33.74

Table 4.1: Comparison of convergence time, number of iterations and total number of floating point operations.

Since the white noise input signal consists of a random combination of all frequencies at equal amplitude, when the filter lengths of H and W are equal, the system will be able to identify the signal with a high degree of accuracy. Case (i) in table 4.1 shows the results of the simulation with $H = W = 512$, with a stopping criterion of 1,000 iterations.

The table shows that the number of total floating point operations for Newton's method were significantly lower than for both the steepest descent algorithm (SD) and conjugate gradient algorithm (CGA); this is because the filter length of 512 is relatively short. Although the number of floating point operations used by the CGA was slightly more than the SD, this is compensated by the fact that the reduction in error is significantly better than the SD, and that the convergence time is smaller. There was a large reduction in error for all three algorithms, indicating that the filter W was a good match to the unknown filter H. Figure 4.4 shows a comparison of the convergence of the error signal for the three algorithms for a filter length of 512. The optimal minimum value indicated in figure 4.4 was calculated by using MATLAB and equations (3.3, 3.7).

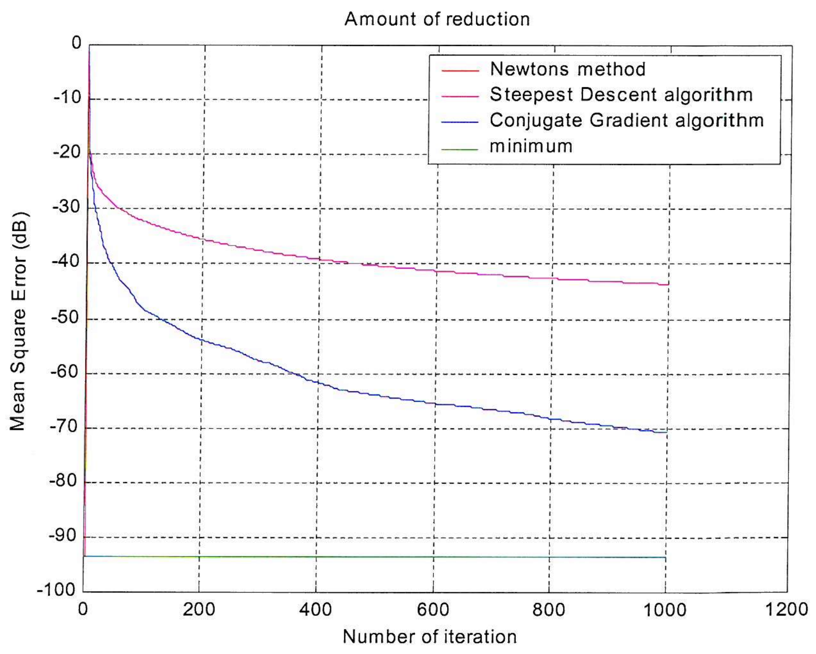


Figure 4.4: Comparison of three algorithms for system identification with filter length of 512. The filter length W is same as the unknown filter length H, stopping criterion is 1000 iterations.

The maximum length that was chosen for W is equal to the length of H, as chosen for case (i). Therefore, in each of the cases (ii), (iii) and (iv), the filter W was set shorter than the filter H

in order to more effectively compare the performance of the three algorithms. The length of W was chosen to ensure that the reduction in error was at a reasonable level: around 30dB. The stopping criteria of 0.1dB above the optimal minimum and a maximum of 1,000 iterations were set to enable simple comparison of the performance of the algorithms. The stopping criterion of 0.1dB above the optimal minimum was set since this would give results which are sufficiently accurate for our purposes. The optimal minimum value was calculated by using MATLAB and equations (3.3, 3.7) as in case (i). The secondary criterion of a maximum of 1,000 iterations was set since preliminary runs of the remaining cases indicated that 1,000 would be a sufficient number of iterations for a satisfactory comparison of the algorithms.

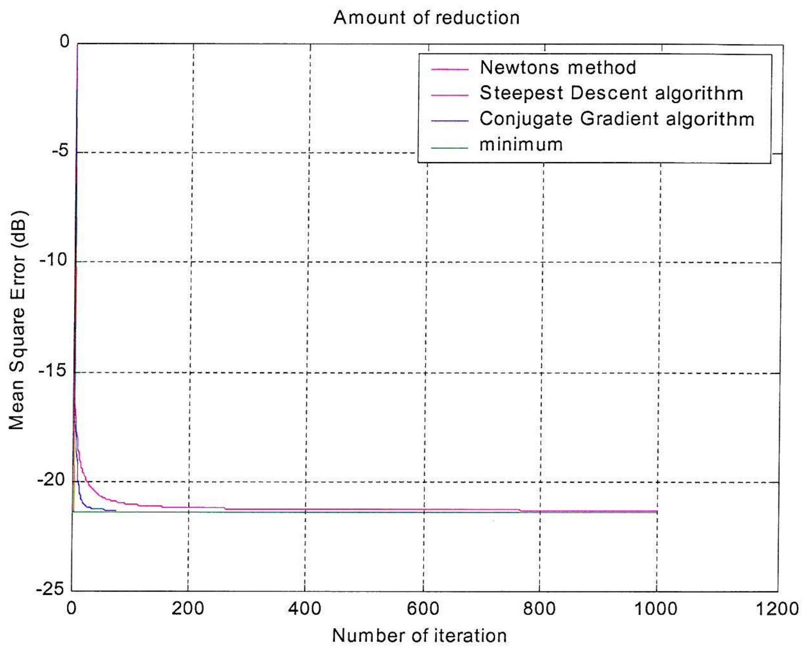


Figure 4.5: Comparison of three different algorithms for system identification with a filter length H of 512, and filter length W of 450. Stopping criterion 0.1dB above the minimum, maximum 1000 iterations.

Case (ii) in table 4.1 shows the results of the simulation with $H = 512$ and $W = 450$. Newton's method took just one iteration to reach the minimum, due to the construction of the algorithm. The CGA took 74 iterations to reach the minimum, whereas the SD reached the stopping criteria of 1,000; Figure 4.5 shows clearly that the Newton's method and CGA are very close to the minimum MSE after very few iterations, and hence Newton's method and CGA had faster convergence rates than the SD. The figure also shows that, since the MSE was still being reduced at each iteration, the SD was still converging to the minimum at the 1,000th iteration, but at a slow rate. The number of iterations is linked to the number of total floating point operations; since the SD performed 1,000 iterations, it used many more floating point operations in total than the CGA and Newton's method. However, the number of floating point operations per iteration showed that the SD required far less floating point operations at each iteration. Although Newton's method only performed one iteration, it used more floating point operations (per iteration) than the SD and CGA. So, Newton's method uses one computationally heavy step to reach the minimum, whereas the SD and CGA uses a fairly small number of iterations, each of which has a lower computational complexity than the step performed by Newton's method. Convergence time for Newton's method and CGA were very similar, but the SD had by far the slowest convergence time.

The results of the simulations for case (iii), with $H = 1024$ and $W = 600$, were similar to case (ii), and are presented in table 4.1. Again, Newton's method took one iteration to reach the minimum; the CGA took significantly less iterations than the SD. Figure 4.6 shows that the CGA was fairly close to the minimum MSE after 100 iterations, and that the SD only approached the same level at around 1,000 iterations. Hence, the CGA had an overall faster convergence rate than the SD. Also, the SD used more total floating point operations, to reach the minimum, than the CGA and Newton's method. This is due to the number of iterations used, which is in turn due to the search directions employed by the algorithm. However, the SD had the smallest number of floating point operations at each iteration because of the simple formulations of the SD. The table also shows that for the longer W, Newton's method uses a significantly increased number of floating point operations. This is due to the increase in the dimensions of the matrix that the algorithm inverts. The computation time for Newton's method and CGA were both shorter than the SD.

Case (iv) in table 4.1 shows the results of the simulation with $H = 2048$ and $W = 800$. The results here are very similar to case (ii). Figure 4.7 shows that the convergence rates of both the Newton's method and CGA were much faster than the SD, since the gradient of the plot is much steeper for both methods. In this case, the CG had the smallest number of total floating point operations, followed by Newton's method. But Newton's method had the largest number of floating point operations per iteration due to its large inverse matrix. The SD had a significantly higher number of total floating point operations than either the CGA or Newton's method. This means that the CGA was fast converging and the most computationally efficient of the three algorithms. The computation time for the Newton's method and CGA were very similar. However, if a longer filter length W is used, e.g. 1024, 2048, CGA could also be shorter in Time (sec) compare to Newton's method. Finally, the SD had a much longer computation time than the CGA and Newton's method, although it had the small number of floating point operations at each iteration.

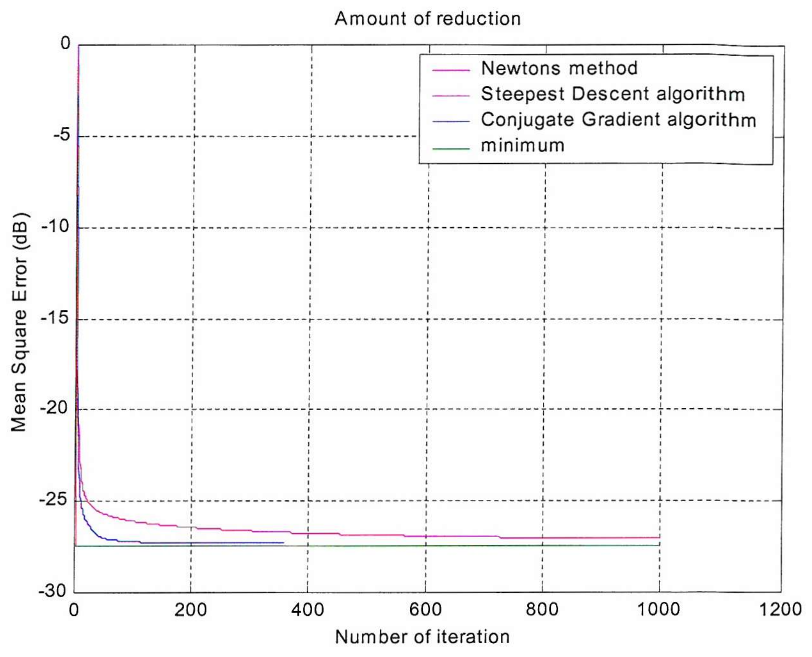


Figure 4.6: Comparison of three different algorithms for system identification with a filter length H of 1024, and filter length W of 600. Stopping criterion 0.1dB above the minimum, maximum 1000 iterations.

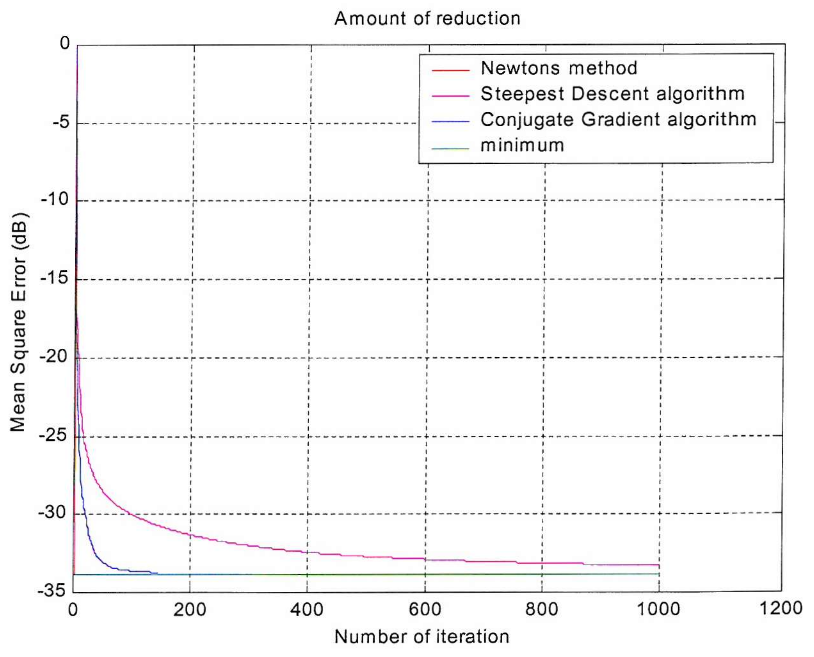


Figure 4.7: Comparison of three different algorithms for system identification with a filter length H of 2048, and filter length W of 800. Stopping criterion 0.1dB above the minimum, maximum of 1000 iterations.

4.4 Algorithm analysis: Prediction problem simulation

In prediction adaptive filtering, the filter must predict the present value of a random signal; the present value is therefore used as the desired signal and past values of the signal are input values into the filter. The system output will be taken as either the output from the filter, in which case the system is known as a predictor system, or the system output will be taken as the error in the prediction, in which case the system operates as a prediction error filter. The choice of system output will depend on the application. The block diagram is shown as figure 4.8.

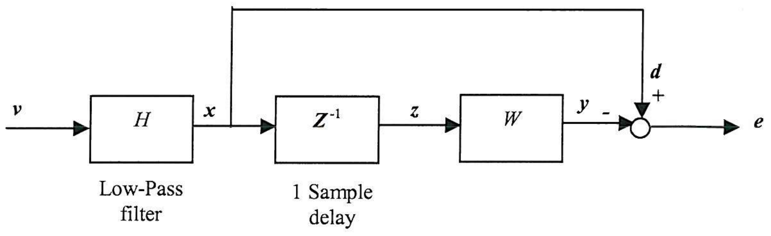


Figure 4.8: Block diagram of prediction system.

An example of a prediction system is illustrated in figure 4.8. The input signal *x* is passed through a modelling delay in order to produce the delayed signal *z*. This is then filtered through *W* to produce *y*. Comparing the desired signal *d* with the delayed signal *y* produces the error in the system. In this case, Δ is one sample. The reference signal is generated by passing white noise through a low pass filter whose transfer function is $H(z) = [(1-0.5z^{-1})/(1-0.6z^{-1})]^{16}$, [Rafaely and Elliott, 2000a]. Figure 4.9 shows a plot of the low pass filter used in this system. The floating point operations and reduction in error are as described in the previous section. The stopping criteria used are 0.1dB above the minimum, with a maximum of 1,000 iterations. The minimum value is calculated by using MATLAB and equations (3.3, 3.7) as in section 4.3. The performance of the algorithms is compared for three different filter lengths: 512, 1024 and 2048.

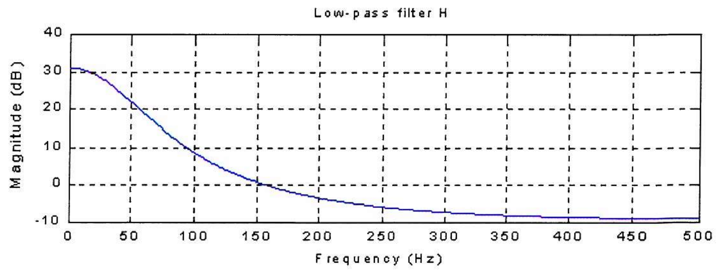


Figure 4.9: Low pass filter with transfer function is $H(z) = [(1-0.5z^{-1})/(1-0.6z^{-1})]^{16}$.

Methods	**Filter length** *W*	**Iterations**	**Floating point operations**	**Time (sec)**	**Reduction in error (dB)**
(i) Stopping criteria: 0.1dB above optimal minimum; maximum 1,000 iterations					
Newton's	512	1	273.1e+006	8.35	-15.30
Steepest descent	512	1000	1849.1e+006	183.56	-15.04
Conjugate gradient	512	72	134.9e+006	7.97	-15.18
(ii) Stopping criteria: 0.1dB above optimal minimum; maximum 1,000 iterations					
Newton's	1024	1	2168.1e+006	64.21	-16.66
Steepest descent	1024	1000	7369.9e+006	556.12	-16.31
Conjugate gradient	1024	85	633.4e+006	35.49	-16.55
(iii) Stopping criteria: 0.1dB above optimal minimum; maximum 1,000 iterations					
Newton's	2048	1	17262.0e+006	471.70	-17.56
Steepest descent	2048	1000	29433.0e+006	5642.60	-17.12
Conjugate gradient	2048	90	2676.6e+006	139.84	-17.45

Table 4.2: Comparison of convergence time, number of iterations and total number of floating point operations.

Case (i) in table 4.2 shows that the CGA had the best overall performance of the three algorithms. Although the Newton's method had the fastest convergence rate, the CGA used fewer floating point operations and less computation time than both the Newton's method and SD. Figure 4.10 illustrates that the convergence rate of the CGA was fairly close to that of the Newton's method and that the convergence rate of the SD was poor compared to both the CGA and Newton's method. The CGA reached the minimum MSE after 72 iterations, whereas the SD had not quite reached the minimum after 1,000 iterations. The reduction in error for all three algorithms was sufficiently good.

Case (ii) showed similar results to case (i) in that the CGA outperformed Newton's method and SD. The CGA took fewer iterations to reach the minimum and used less floating point operations than both the Newton's method and SD. The CGA also used the least computation time of the three algorithms and therefore outperformed the Newton's method and SD in every category. Figure 4.11 further illustrates the convergence rates of the algorithms. The results show that the time and number of floating point operations for Newton's method have greatly increased over case (i). This would seem to suggest that Newton's method is more suitable for application with short filter lengths. With the increased filter length, the reduction in error increased for each of the three algorithms, over case (i).

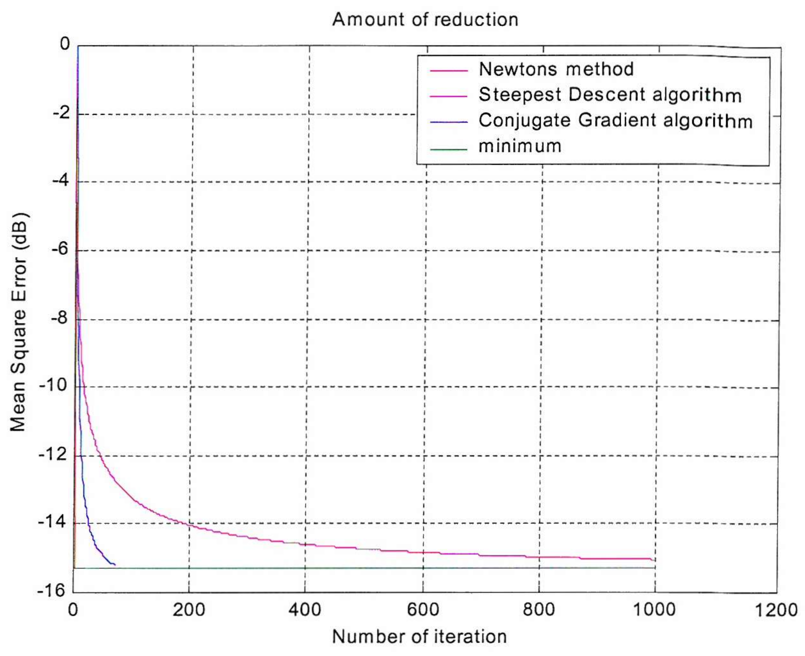


Figure 4.10:Comparison of three different algorithms for prediction system with a filter length W of 512. Stopping criterion 0.1dB above the minimum, maximum of 1000 iterations.

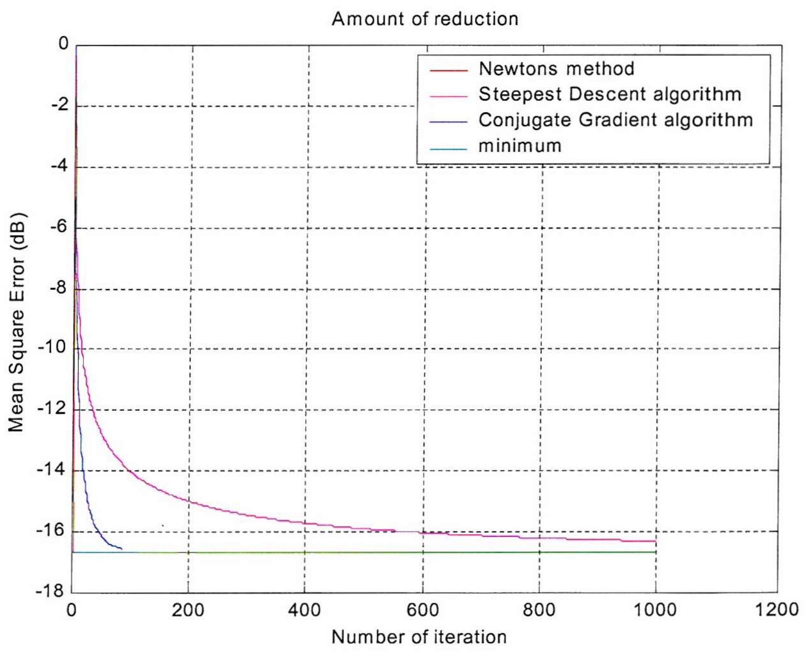


Figure 4.11: Comparison of three different algorithms for prediction system with a filter length W of 1024. Stopping criterion 0.1dB above the minimum, maximum of 1000 iterations.

The final case used a filter length of 2048. The results of case (iii) clearly demonstrates that the CGA is the best of the three algorithms tested in terms of computational efficiency and time. The CGA used far less floating point operations and less computation time than either SD or Newton's method. Since acoustic applications often require long filter lengths, these results suggest that the CGA should be consider for such applications. It should be noted that the number of floating point operations and computation time for the Newton's method have now increased to high levels, and it is therefore not suitable for applications requiring high order filters. A comparison of the convergence rates of the algorithms minimising the MSE is shown in figure 4.12.

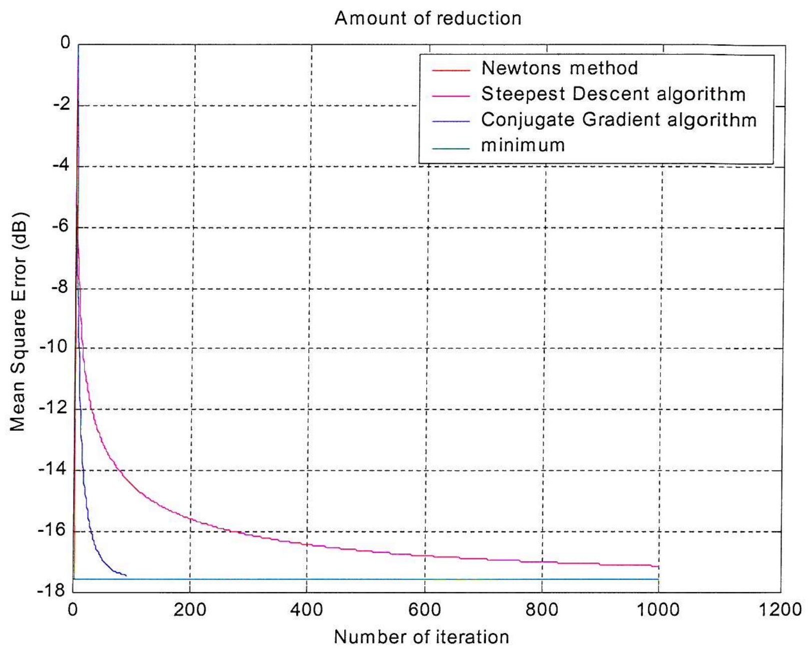


Figure 4.12 Comparison of three different algorithms for prediction system with a filter length W of 2048. Stopping criterion 0.1dB above the minimum, maximum of 1000 iterations.

4.5 Conclusions

The convergence rate, computation time, total number of floating point operations and reduction in error for each algorithm have been used as performance criteria to compare the methods for system identification and prediction problems.

Based on the above criteria, Newton's method gave the best convergence speed performance of the three algorithms for the system identification simulation. However, the CGA was the most computationally efficient for long filters. The SD was outperformed by both CGA and Newton's method in each of the performance criteria. It should be noted that the results are data dependant, particularly with respect to the size of W in relation to the size of H in the system identification simulations. For the prediction system problem, the CGA was the best performing algorithm. Although Newton's method converged in one iteration, the CGA had far fewer floating point operations, much less computation time and an almost identical reduction in error. As with the system identification problem, SD had the worst performance of the three algorithms.

The results of the simulations above suggest that Newton's method is particularly suitable for applications requiring short filter lengths, but less suitable for long filtering applications due to the computations involved in the correlation matrix inversion. The CGA performed well for both applications, but gave the best performance of the three algorithms for the prediction system problem. However, the aim of this thesis is to develop adaptive constrained filtering for acoustic applications, by using a penalty function formulation. Because acoustic applications require long filter lengths, it would not be suitable to use Newton's method. Also, there is the difficulty that if the estimated matrix inverse loses the property of positive definiteness, it will cause the algorithm to diverge.

This chapter showed that matrix based methods, e.g. Newton's method, are less suitable for long filtering application due to the computational complexity and it might cause inefficiency performance, therefore matrix based methods are not used in the thesis. The vector/gradient based methods, e.g. CGA and SD, are considered to further develop the long adaptive filtering problem due to the convergence time and computation efficiency. The next chapter approaches the solution of constrained problems which are investigated; constraints will be added to the problem to further develop and study constrained FIR optimisation, leading to constrained adaptive optimisation in following chapters. Although SD and CGA were used here, other vectors based on methods such as Quasi-Newton's method and Davidon-Fletcher-Powell method, which was discussed in chapter 2, could also be used in a similar manner.

Chapter 5

Formulation of constrained optimisation as unconstrained

5.1 Introduction

Various methods for optimisation problems were presented and the results of the simulations in chapter 4 showed that the CGA produced good performances. The aim of the thesis is to develop constrained adaptive filters, therefore in this chapter, approaches to the solution of constrained problems are investigated. The new constrained formulations are investigated using simulation examples that use SD and CGA as presented in the previous chapters. In the thesis, the constrained optimisation problem is formulated as an unconstrained problem using the penalty function, and then can be solved by using unconstrained optimisation methods. The penalty method is used here due to its computation simplicity which is more suitable for real-time adaptive filtering. Although the penalty method is used here, other possible ways to approach constrained optimisation problems could also be considered (refer to chapter 2 for other various methods).

The ability to specify performance objective functions together with practical constraints can produce solutions that are more suitable for practical implementation than the minimisation of unconstrained objective functions. The constraints considered here are all quadratic constraints so that the constrained optimisation problem can be solved as convex programming problem. A penalty function formulation is then used, so constraints are incorporated into a modified objective function, which is then treated as unconstrained. Linear penalty function (first order) and quadratic penalty function (second order) are considered in this thesis. Second order penalty function is a continuous function and provides a smooth error surface, however high computation complexity and high order terms will be produced during the adaptive processing. First order penalty function is not a continuous function and may cause large fluctuations due to the discontinuity, but it gives a simple formulation, which facilitates a comprehensive theoretical analysis (chapter 7) of convergence, thus helping to improve convergence performance. The formulation of both penalty functions will be presented and the characteristic will be discussed.

To avoid high-order terms in the second order penalty function formulation, the quadratic constraints are approximated using linear constraints, so that the objective function produced by the second order penalty function will now only contain quadratic terms, i.e. using these linear constraints rather than the original quadratic constraints will mean that the new objective function will be quadratic. This way the second order penalty function can have the simple formulation of the first order penalty function, and also have the benefit of the second order i.e. continuous error surface. The second order penalty function with linear approximation to the constraints is therefore investigated.

A brief review of the use of various constraints is presented, as its underlying principle is used for the constrained adaptive filtering problems. For example, constraints can be useful to increase the robustness, e.g. in the application of sound equalisation, constraints can be used to prevent extreme gains in the equalisation filter and therefore improve spatial robustness. Another example is in the application of active sound control, where a useful constraint can impose limit on the disturbance enhancement outside the control frequency range. Another constraints can be used to guarantee robust stability in feedback systems. It is clear that the design of constraint is dependent on the applications.

This chapter has four parts: (1) The discussion of various constraints; (2) Study of the characteristic of penalty functions max(c_i, 0) and max(c_i, 0)^2; (3) Approximation of quadratic constraints using linear constraints in the case of using a second order penalty function; (4) The performance of the various formulations is compared using the steepest descent and CG algorithms.

We now first reformulate the constrained optimisation problem as an unconstrained problem using the penalty method. The reader is referred to chapter 2, for further detail on the penalty and barrier methods.

5.2 Constrained optimisation by Penalty function

Selecting a penalty function and penalty parameters can be problem dependent, and so there is no general rule for penalty function selection. We will discuss here the two penalty functions max(c_i, 0)^2 and max(c_i, 0). Because the form of max(-) gives a simple formulation, which facilitates a comprehensive theoretical analysis of convergence when quadratic constraints involved.

Let's briefly revise the constrained problem we are considering here. Most optimisation problems have constraints, and so the solutions are obtained as the final result of a recursive search and must be feasible, that is, satisfy all constraints. The constraints define the feasible region, meaning that if vector $\boldsymbol{x}$ complies with all constraints $c_i(\boldsymbol{x}) = 0$ and $c_i(\boldsymbol{x}) \leq 0$ then it belongs to the feasible region.

In general, a constrained numerical optimisation problem is defined as (see also chapter 2):

$$
\begin{array}{llll}
\text{Minimise} & f(\boldsymbol{x}) & \boldsymbol{x} \in \mathbf{R}^n & \\
\text{Subject to} & c_i(\boldsymbol{x}) = 0 & \text{for } i = 1, \ldots, m_e & \quad (5.1) \\
 & c_i(\boldsymbol{x}) \leq 0 & \text{for } i = m_e+1, \ldots, m &
\end{array}
$$

Here, $\boldsymbol{x}$ is an n-dimensional parameter vector and $f(\boldsymbol{x})$ is the objective function or cost function to be minimised under nonlinear equality and inequality constraints given by $c_i(\boldsymbol{x})$, $i = 1, \ldots, m$. It is assumed that these functions are continuously differentiable in $\mathbf{R}^n$.

To construct the unconstrained problems, a large penalty term is added to the objective function whenever $\boldsymbol{x}$ lies outside the feasible region. A parameter σ controls the degree of penalising f. In this approach, a constrained problem is constructed into an unconstrained one. The function under consideration is transformed as follows:

$$
\phi(\boldsymbol{x},\sigma) = \begin{cases} f(\boldsymbol{x}) & \boldsymbol{x} \in \textit{feasible region} \\ f(\boldsymbol{x}) + \sigma\, p(\boldsymbol{x}) & \boldsymbol{x} \notin \textit{feasible region} \end{cases} \qquad (5.2)
$$

where σ is the penalty parameter, $p(\boldsymbol{x})$ is the penalty function, and the problem described in (5.1) turns into the one of minimising (5.2) given the selection of the penalty function. The method generates a sequence of points whose limit can be shown to be an optimal solution to the original constrained problem. The advantage of this approach is, of course, that algorithms for the solution of unconstrained problems are generally simpler and more robust than those for constrained problems. The main disadvantage is, when σ becomes too large, the Hessian matrix $\nabla^2\phi(\boldsymbol{x},\sigma)$ becomes more ill-conditioned, so that the unconstrained minimisation of $\phi(\boldsymbol{x},\sigma)$ becomes unstable.

Sections 5.4 and 5.5 will present the quadratic penalty function $\max(c_i, 0)^2$ and the linear penalty function $\max(c_i, 0)$. Before that, let's consider some practical constraints for constrained optimisation problems.

5.3 Various practical constraints

This section describes examples of practical constraints for adaptive filtering. Choosing constraints are problem dependent; different constraints are required by different applications. Sound equalisation is considered as the application example in this thesis. Therefore the constraints introduced here are based on this application, although other constraints will also be presented.

The first constraints introduced with a limit on the magnitude of the system. This is useful to avoid excess amplification of the filter at given frequencies and is used in section 5.7 in the prediction system example and later on in the sound equalisation problem. This constraint can increase the robustness of the system by avoiding too high gains. A limit L is placed on the magnitude of the filter W for any given frequency k. This can be written as:

$$|W(k)| < \sqrt{L(k)} \qquad k = 0, \ldots, N\text{-}1 \tag{5.3}$$

where the filter gain in decibels is limited to less than $10\log_{10}L(k)$ dB. The filter W is calculated as the FFT of $\boldsymbol{w}$ at any frequency. Equation (5.3) can be rewritten to define a constraint function c_k as:

$$c_k = |W(k)|^2 - L(k) < 0 \tag{5.4}$$

where the square value of W is used for convenience of calculation.

Another constraint is the limit on the power of the filter output signal. It is a constraint on the average power of signal y; in control applications, it can be applied to ensure the actuator driven by signal y is not overloaded. Note x and y are represented as input and output signals respectively. The power limit constraint can therefore be written as:

$$\|y\|_2^2 < p \tag{5.5}$$

where the constant p is the power limit and $\|\cdot\|_2^2$ denotes the 2-norm.

Equation (5.5) can be written in the frequency domain using a block of input data, which is:

$$\frac{1}{N}\sum_{k=0}^{N-1}|\,X(k)W(k)\,|^2 < p \tag{5.6}$$

Equation (5.6) can be rewritten to define a constraint function c as:

$$c = \frac{1}{N}\sum_{k=0}^{N-1}|\,X(k)W(k)\,|^2 - p < 0 \tag{5.7}$$

Note that in equation (5.7) there is only one constraint involved.

Other constraints can also be considered. The reader can refer to Rafaely (1997), for further detail on other design constraints for different applications. Constraint (5.3) will be used throughout this thesis to develop algorithms for constrained adaptive filtering.

5.4 Second order penalty function (Quadratic penalty function)

One of the earliest methods for solving constrained optimisation problem (5.1), and the most widely used penalty function is the quadratic penalty function. It dates back to an idea of Courant (1943), the detail can be found in the book by Fiacco and McCormick (1968). Let us consider problem (5.1) and let $p(\boldsymbol{x})$, p: $\mathbf{R}^n \rightarrow \mathbf{R}$ be a continuous function such that $p(\boldsymbol{x}) = 0$ for all $\boldsymbol{x} \in$ *feasible region*, $p(\boldsymbol{x}) > 0$ for all $\boldsymbol{x} \notin$ *feasible region*. The function $\phi(\boldsymbol{x})$ is obtained by using penalty terms that are the squares of the constraint violations and is given by:

$$\phi(\boldsymbol{x},\sigma) = f(\boldsymbol{x}) + \sigma \sum_{i=m_e+1}^{m} \max\left(c_i(\boldsymbol{x}),0\right)^2 \tag{5.8}$$

The constrained problem (5.1) is reformulated using the penalty method to obtain a new unconstrained problem (5.8). The cost function $f(\boldsymbol{x})$ considered here is the mean square error while inequality constraints are only considered in this thesis. The objective function J, following equation (5.9), can be written as

$$J = E[\boldsymbol{e}(n)^{\mathrm{T}}\,\boldsymbol{e}(n)] + \sigma\sum_{i=1}^{I} \max\left(c_i(\boldsymbol{w}),0\right)^2 \tag{5.9}$$

Let us first consider the structure of constraint (5.4) in the time domain, we then apply this constraint in equation (5.9). The DFT operation on $\boldsymbol{w}$ is written in vector form [Widrow and Stearns, 1985]

$$W(k) = \sum_{n=o}^{N-1} w_n e^{-j2\pi nk/N} = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{a}_k \tag{5.10}$$

where $\boldsymbol{a}_k$ is a column vector of the complex Fourier exponential at frequency k. c_k in (5.4) can now be written in the time domain, using the results above. The constraint at frequency k is [Rafaely and Elliott, 2000b]:

$$c_k = \boldsymbol{w}^{\mathrm{T}} (\boldsymbol{a}_k^* \boldsymbol{a}_k^{\mathrm{T}}) \boldsymbol{w} - L(k) \tag{5.11}$$

Hence equation (5.11) is a quadratic equation of the form $c_k = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{A} \boldsymbol{w} + d$. In the above $|W(k)|^2$ is $W(k)$ times its complex conjugate, and $\boldsymbol{a}_k^*$ is the complex conjugate of $\boldsymbol{a}_k$. Thus all the constraints are convex because the term $\boldsymbol{w}^{\mathrm{T}} (\boldsymbol{a}_k^* \boldsymbol{a}_k^{\mathrm{T}}) \boldsymbol{w}$ is positive semi-definite, due to its construction from $|W(k)|^2$. Since c_k is convex, it follows that the penalty term is convex [Roberts and Varberg, 1973]. Hence, the new objective function J is also convex, and the SD, CG and other gradient-based search algorithms can be applied to find a unique solution of the original constrained problem.

Since the SD and CGA algorithms are the gradient search methods, the gradient of the objective function (5.9) are required for the update equations. First look at the gradient of the penalty term in equation (5.9), the gradient of each term in the summation of the penalty term is given by

$$\frac{\partial}{\partial \boldsymbol{w}} \max(c_i(\boldsymbol{w}), 0)^2 = \begin{cases} 0, & \text{if } c_i(\boldsymbol{w}) \le 0 \\ 2c_i(\boldsymbol{w}) \dfrac{\partial}{\partial \boldsymbol{w}} c_i(\boldsymbol{w}), & \text{if } c_i(\boldsymbol{w}) > 0 \end{cases} \tag{5.12}$$

For convenience, a new operator $[c_i]_z$, as defined below [Rafaely and Elliott, 2000b], is introduced:

$$[c_i(\boldsymbol{w})]_z = c_i(\boldsymbol{w}) \frac{\mathrm{sign}(c_i(\boldsymbol{w})) + 1}{2}, \tag{5.13}$$

where $\mathrm{sign}(c_i)$ is the function that returns the value 1 when the constraints are active and the value -1 otherwise [Rafaely and Elliott, 2000b].

The partial derivative above can now be rewritten as

$$\frac{\partial}{\partial \boldsymbol{w}} \max(c_i(\boldsymbol{w}),0)^2 = 2\,[c_i(\boldsymbol{w})]_z \frac{\partial}{\partial \boldsymbol{w}} c_i(\boldsymbol{w}) \tag{5.14}$$

Thus, the gradient of the penalty term is:

$$\frac{\partial}{\partial \boldsymbol{w}} \sum_{i=1}^{I} \max(c_i(w)\,,0)^2 = 2\sum_{i=1}^{I}\, [c_i(\boldsymbol{w})]_z \frac{\partial}{\partial \boldsymbol{w}} c_i(\boldsymbol{w}) \tag{5.15}$$

The gradient of the new objective function can now be written as:

$$\nabla J_n = -2(\boldsymbol{p} - \boldsymbol{R}\boldsymbol{w}_n) + 2\sigma \sum_{i=1}^{I} [c_i(\boldsymbol{w}_n)]_z \frac{\partial}{\partial \boldsymbol{w}_n} c_i(\boldsymbol{w}_n) \tag{5.16}$$

Now, the partial derivative with respect to $\boldsymbol{w}$ for c_k is required, so that the penalty term in the new update equations can be reformulated. This is given by

$$\frac{\partial}{\partial \boldsymbol{w}} c_k = 2\,\boldsymbol{a}_k^* \boldsymbol{a}_k^{\mathrm{T}} \boldsymbol{w} = 2\,W(k)\boldsymbol{a}_k^* \tag{5.17}$$

The gradient of the new objective function is

$$\nabla J_n = -2(\boldsymbol{p} - \boldsymbol{R}\boldsymbol{w}_n) + \left(4\,\sigma \sum_{k=0}^{N-1} [\,|W(k)|^2 - L(k)]_z\, W(k)\, \boldsymbol{a}_k^* \right) \tag{5.18}$$

Due to the term W in the penalty term, the update equation is still not completely in the time domain. The inverse DFT can be used to transform the last few terms from the frequency domain to the time domain. The final gradient of the new objective function therefore becomes:

$$\nabla J_n = 2(\boldsymbol{R}\boldsymbol{w}_n - \boldsymbol{p}) + \boldsymbol{F}^{-1} \left(4\,\sigma N\,[\,|W(k)|^2 - L(k)\,]_z\, W(k) \right) \tag{5.19}$$

The above can now be substituted to obtain the time domain update equations for the CGA, SD and other algorithms. As an example, the time domain update equation of SD and CG algorithms are now rewritten to include the reworking of the penalty function above.

The update equation for the SD is given by

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n + \mu\ (-\nabla J_n) \tag{5.20}$$

The update equation using the CG algorithm is:

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n + \mu\ \boldsymbol{b}_n,$$

where

$$\boldsymbol{b}_0 = -\nabla J_0,\ \ \boldsymbol{b}_{n+1} = -\nabla J_{n+1} + \beta_n \boldsymbol{b}_n \ \text{ and } \ \beta_k = \frac{\nabla J_{n+1}^{\mathrm{T}} \nabla J_{n+1}}{\nabla J_n^{\mathrm{T}} \nabla J_n}. \tag{5.21}$$

The advantage of the second order penalty function is that the continuous gradients avoids numerical problems, it is differentiable at points at the boundary of the feasible region where $c_i(\boldsymbol{x}) = 0$ and is therefore a "smooth" penalty function.

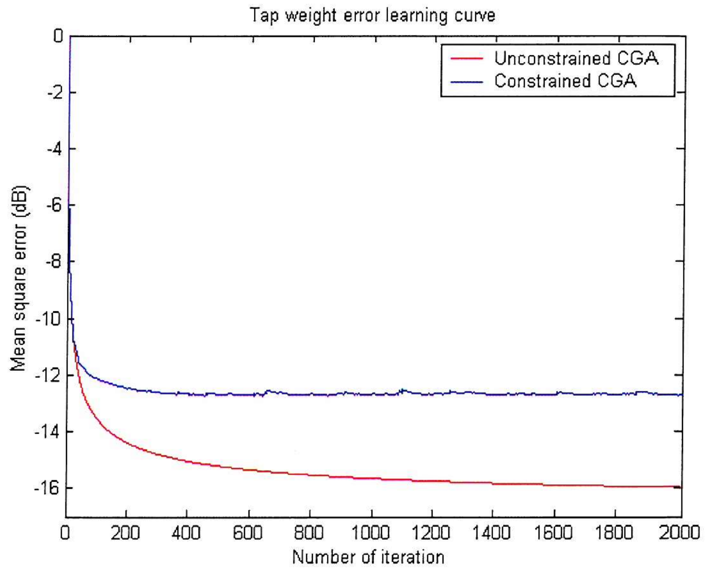


Figure 5.1: The performance of the convergence characteristics of unconstrained and constrained cases using second order penalty function with $\sigma = 0.005$.

Figure 5.1 is a simulation of the prediction system with the second order penalty function. The results of a prediction problem are presented, in which the reference signal is equal to the desired signal delayed by one sample. The desired signal is generated by passing white noise through a simple low pass filter H (data referred to section 4.4). The CG algorithm is simulated in MATLAB, using a sampling frequency of 1kHz, a filter length W of 128, with 2000 iterations and using the magnitude constraints $|W(k)|^2 < L(k)$, where L is selected as a linearly line from 0dB to 15dB. Figure 5.1 shows the convergence performance of unconstrained and constrained cases, it also shows that the algorithm with second order penalty function and $\sigma = 0.005$, smoothly converges to the minimum. Figure 5.2 shows the magnitude responses of W after convergence, which illustrates the effect of the constraint. Note that, since the quadratic form constraints are only considered in this thesis, so the objective function (5.9) produced by the second order penalty function will involve higher order terms rather than only quadratic term, as we can see from equation (5.19).

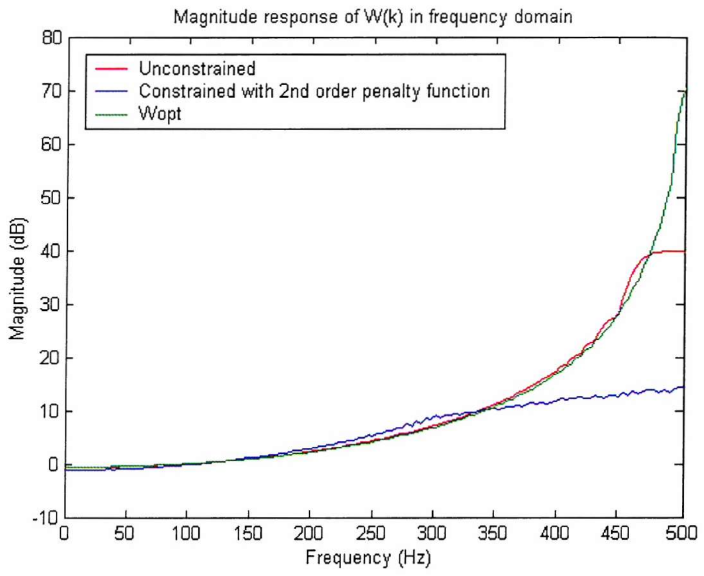


Figure 5.2: Comparison of the magnitude responses after convergence.

5.5 First order penalty function (Linear penalty function)

Another type of penalty function is the first order, or linear penalty function, which is given by:

$$\phi(x,\sigma) = f(x) + \sigma \sum_{i=m_e+1}^{m} \max(c_i(x),0) \tag{5.22}$$

In spite of its simplicity, function (5.22) is not continuously differentiable at the border point where constraint c_i is active, due to the presence of the term $\max(c_i, 0)$. This case produces an unstable or noisy convergence at points where constraint c_i is active.

The constrained problem is reformulated using the penalty method to obtain a new unconstrained problem (5.22). The penalty term considered here is,

$$\sum_{i=1}^{I} \max\left(c_i(\boldsymbol{w}), 0\right) \tag{5.23}$$

The objective function of (5.22) can be written as

$$J = E[\boldsymbol{e}(n)^{\mathrm{T}} \boldsymbol{e}(n)] + \sigma \sum_{i=1}^{I} \max\left(c_i(\boldsymbol{w}), 0\right) \tag{5.24}$$

Since c_i is convex, it follows that the penalty term is convex. Hence, the objective function J (5.24) is also convex, which is similar to the case of a second order penalty function. As an example the SD and CG algorithms are also applied here to find a unique solution of the original constrained problem.

The gradient of each term in the summation of the penalty term is given by

$$\frac{\partial}{\partial \boldsymbol{w}} \max\left(c_i(\boldsymbol{w}), 0\right) = \begin{cases} 0, & \text{if } c_i(\boldsymbol{w}) \leq 0 \\ \dfrac{\partial}{\partial \boldsymbol{w}} c_i(\boldsymbol{w}), & \text{if } c_i(\boldsymbol{w}) > 0 \end{cases} \tag{5.25}$$

Similarly as previous section, for convenience, a new operator $[c_i]_z$, is introduced:

$$\left[\frac{\partial}{\partial \boldsymbol{w}} c_i(\boldsymbol{w})\right]_z = \frac{\partial}{\partial \boldsymbol{w}} c_i(\boldsymbol{w}) \cdot \frac{\operatorname{sign}(c_i(\boldsymbol{w})) + 1}{2} \tag{5.26}$$

where $\operatorname{sign}(c_i)$ is the function that returns the value 1 when the constraints are active and the value -1 otherwise [Golub and Van, 1983]. The partial derivative of (5.25) can now be written as

$$\frac{\partial}{\partial \boldsymbol{w}} \max\left(c_i(\boldsymbol{w}), 0\right) = \left[\frac{\partial}{\partial \boldsymbol{w}} c_i(\boldsymbol{w})\right]_z \tag{5.27}$$

Thus, the gradient of the penalty term is:

$$\frac{\partial}{\partial w}\sum_{i=1}^{I}\max\left(c_i(w),0\right)=\sum_{i=1}^{I}\left[\frac{\partial}{\partial \boldsymbol{w}}c_i(\boldsymbol{w})\right]_z \tag{5.28}$$

The gradient of the objective function (5.24) therefore becomes:

$$\nabla J_n=-2(\boldsymbol{p}-\boldsymbol{R}\boldsymbol{w}_n)+2\sigma\sum_{k=0}^{N-1}\left[W(k)\,\boldsymbol{a}_k^*\right]_z \tag{5.29}$$

The final gradient of the objective function with the constraint penalty can be written using the inverse discrete Fourier transform as

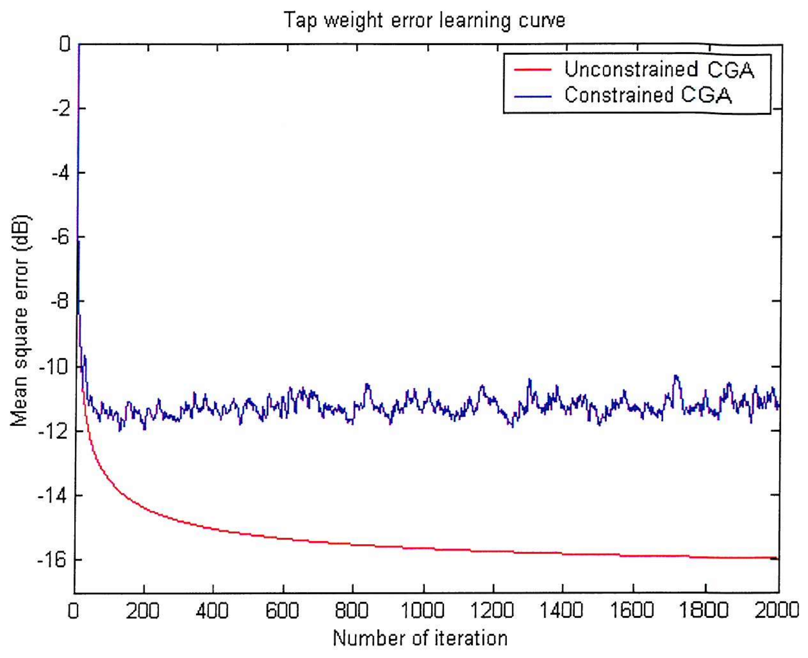
$$\nabla J_n=2(\boldsymbol{R}\boldsymbol{w}_n-\boldsymbol{p})+\boldsymbol{F}^{-1}\left(2\sigma N\left[\,W(k)\,\right]_z\right). \tag{5.30}$$

The equation (5.30) can now be substituted to the update equation of SD (5.20) and CGA (5.21) and then the minimum point can be found. The advantage of using first order penalty function is its simplicity and the optimum can be found exactly with a finite penalty parameter. However, the disadvantage is that max($c_i(\boldsymbol{w})$, 0) is not differentiable at the boundary of the feasible region where $c_i(\boldsymbol{x})$ = 0 i.e. this results in a discontinuous gradient at the boundary of the feasible domain and can lead to fluctuations in the convergence. However, if σ is chosen to be small enough, the noisy convergence can be reduced. Figure 5.3(a), shows the convergence of the CG algorithm when using a first order penalty function, in the prediction system. The prediction system used here is the same system used in previous section. MATLAB simulation of the CG algorithm is performed with a sampling frequency of 1kHz, a filter length W of 128, with 2000 iteration.

Figure 5.3(a) shows that the first order penalty function can have noisy convergence due to the discontinuity at the boundary of the feasible region. However, figure 5.4(a) shows that this can be reduced if σ is reduced. Of course, there is a trade off between convergence time and smooth performance. Although the first order penalty function is discontinuous and cause noisy convergence, it is not far off from the second order penalty function. Figure 5.3(b) and 5.4(b) show the magnitude responses when using σ = 0.005 and σ = 0.001.

In the following section we investigate a linear approximation approach to the quadratic constraints, so that we can have the smoothness of the second order penalty function with the simplicity of the first order penalty function. The theoretical analysis of the first order penalty function will also be conducted in chapter 7.

(a)



(b)

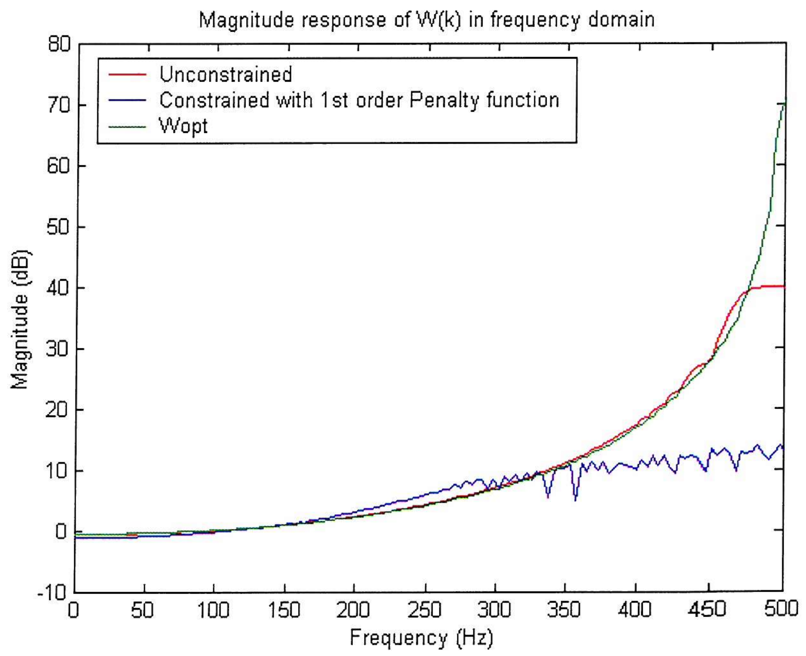
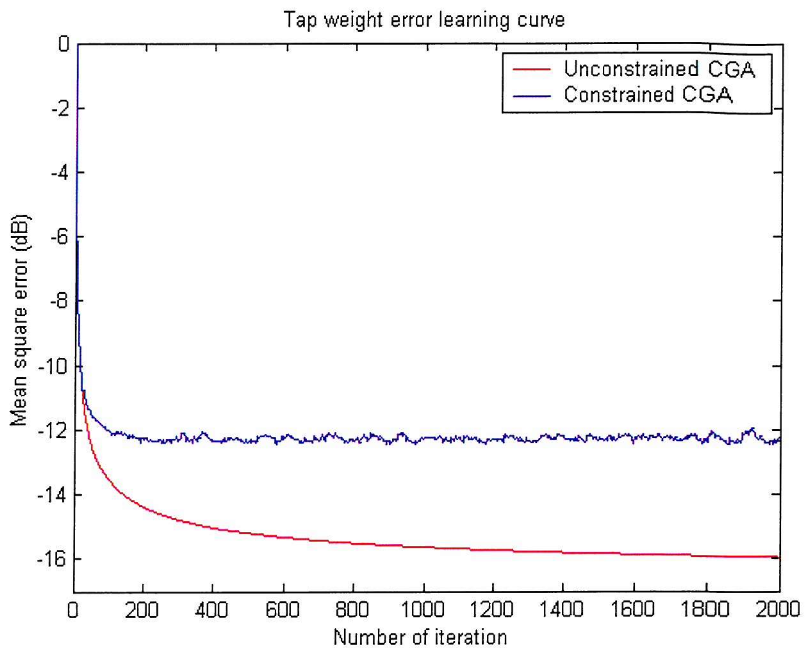


Figure 5.3: The performance of using first order penalty function with (a) the convergence characteristics when σ = 0.005 and (b) the magnitude response after convergence.

(a)



(b)

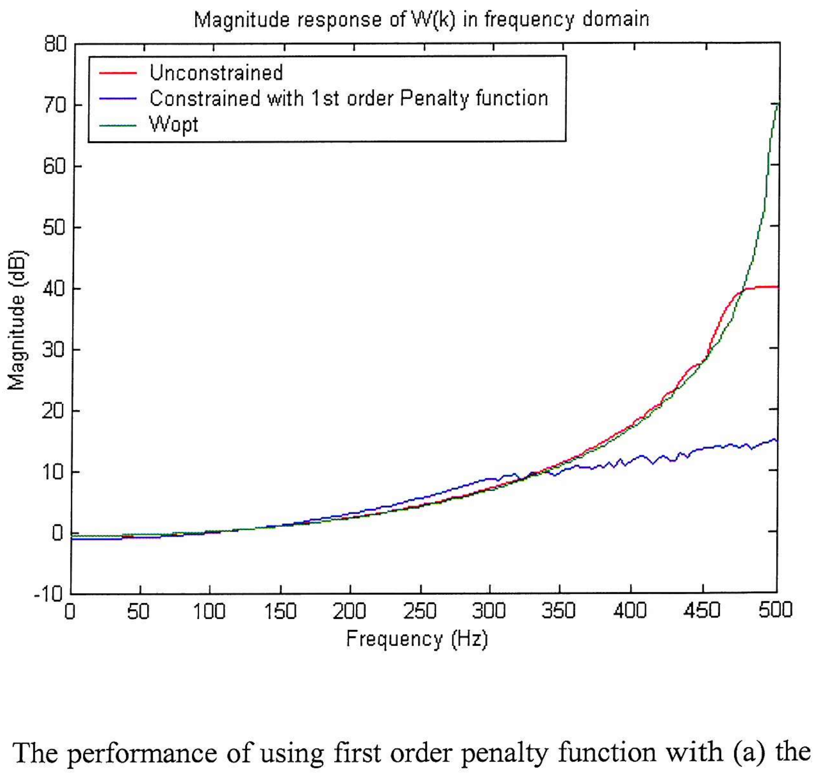


Figure 5.4: The performance of using first order penalty function with (a) the convergence characteristics when $\sigma = 0.001$ and (b) the magnitude response after convergence.

5.6 Second order penalty function with linear approximation

Quadratic constraints with second order penalty function tend to be quite computationally complex, since the penalty term of the penalty function involves a quadratic term, using quadratic constraints would mean an update equation of order four. A method of dealing with quadratically constrained problems is required. This section discusses approximating quadratic constraints with linear constraints; the new objective function produced by the penalty function will then be quadratic. The linear approximation approach to quadratically constrained optimisation using the penalty function has not been previously attempted, the suitability of the approach will therefore be tested, along with its efficiency with respect to number of floating point operations and computation time in the following sections. The formulations developed in this chapter will be investigated via simulations with a prediction system (as in section 4.4) using MATLAB in section 5.7. The linear approximation method is now discussed in the following.

The constraint is on the magnitude of the frequency, $|W(k)|^2 < L(k)$, on the complex plane, which can be illustrated as shown below in figure 5.5.

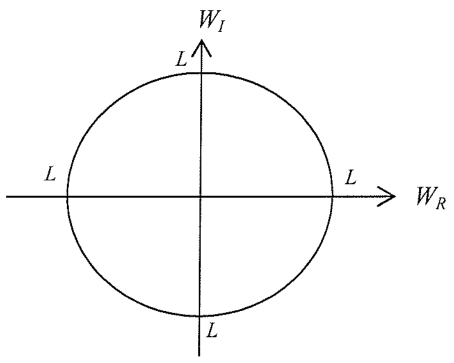


Figure 5.5: Illustration of frequency magnitude constraint.

The simplest linear approximation to the above is to use just four linear constraints, as in figure 5.6(a). One method of constructing these is to connect the points on the axis that the circle crosses. This has the effect of more tightly constraining W, but guarantees that any solution found would also satisfy the original quadratic constraints. An alternative method for using four linear constraints would be to extend the feasible region of the approximation by having the edges of a diamond shape intersecting the circumference of the circle. However, due to the geometry of this approximation, an estimated optimal minimum might be achieved that was not in the feasible region of the original quadratic constraints. Hence, this alternative approach was not used in the simulation runs. Figures 5.6(b) and 5.6(c) show that increasing the number of linear constraints can improve the accuracy of the approximation.

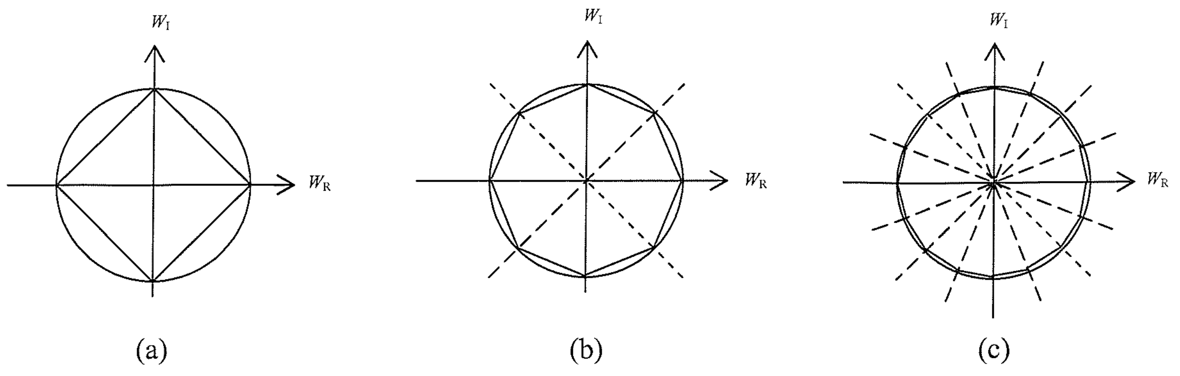


Figure 5.6: Using four, eight and sixteen linear constraints to approximate a quadratic constraint.

5.6.1 Approximating with four linear constraints

The simplest example is considered first: using four linear constraints. These can be represented by the diagram below, in figure 5.7.

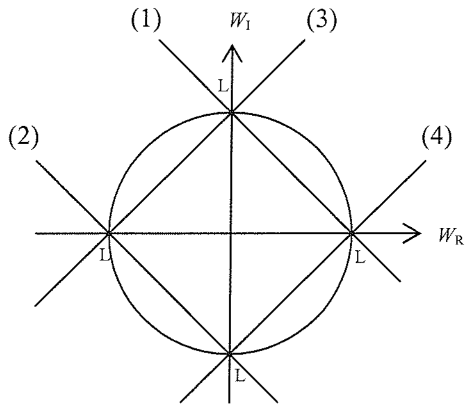


Figure 5.7: Approximating with four linear constraints.

The equations for the four linear constraints are:

$$\begin{array}{ll} (1) & W_I < -W_R + L \\ (2) & W_I < -W_R - L \\ (3) & W_I < W_R + L \\ (4) & W_I < W_R - L \end{array} \tag{5.31}$$

From figure 5.7 it can be see that there are several areas of the original feasible region that are not included in the approximation; the maximum regions which are not included in the approximation is between L and $\frac{L}{\sqrt{2}}$, so error is 3dB maximum. Hence, it would seem that these linear constraints offer a fairly rough approximation of the original quadratic constraint. The above four constraints can be rewritten as:

$$\begin{array}{ll} c_1: & W_R + W_I - L < 0 \\ c_2: & -W_R - W_I - L < 0 \\ c_3: & -W_R + W_I - L < 0 \\ c_4: & W_R - W_I - L < 0 \end{array} \tag{5.32}$$

Since the constraints are in the frequency domain, the Fourier transform is used to convert them to the time domain [Haykin et al, 1984 and Orfanidis, 1985]:

$$W_I(k) = \sum_{n=0}^{N-1} w(n)\sin\left(\frac{2\pi\, nk}{N}\right) = \boldsymbol{w}^{\mathrm{T}} \sin\left(\frac{2\pi\, nk}{N}\right)$$

$$W_R(k) = \sum_{n=0}^{N-1} w(n)\cos\left(\frac{2\pi\, nk}{N}\right) = \boldsymbol{w}^{\mathrm{T}} \cos\left(\frac{2\pi\, nk}{N}\right) \tag{5.33}$$

Reformulating the original constraints, it can be seen that:

$$\begin{array}{ll} c_1: & \boldsymbol{w}^{\mathrm{T}}\mathbf{cos}(k) + \boldsymbol{w}^{\mathrm{T}}\mathbf{sin}(k) - L < 0 \\ c_2: & -\boldsymbol{w}^{\mathrm{T}}\mathbf{cos}(k) - \boldsymbol{w}^{\mathrm{T}}\mathbf{sin}(k) - L < 0 \\ c_3: & -\boldsymbol{w}^{\mathrm{T}}\mathbf{cos}(k) + \boldsymbol{w}^{\mathrm{T}}\mathbf{sin}(k) - L < 0 \\ c_4: & \boldsymbol{w}^{\mathrm{T}}\mathbf{cos}(k) - \boldsymbol{w}^{\mathrm{T}}\mathbf{sin}(k) - L < 0 \end{array} \tag{5.34}$$

Where
$$\mathbf{cos}(k) = \left[\cos(0) \quad \cos\left(\frac{2\pi k}{N}\right) \quad \cdots \quad \cos\left(\frac{2\pi (N-1) k}{N}\right)\right],$$
$$\mathbf{sin}(k) = \left[\sin(0) \quad \sin\left(\frac{2\pi k}{N}\right) \quad \cdots \quad \sin\left(\frac{2\pi (N-1) k}{N}\right)\right]. \tag{5.35}$$

It is now necessary to calculate the gradient of the penalty term. The gradients of the four constraints are as follows:

$$\begin{aligned}\frac{\partial}{\partial w} c_1 &= \mathbf{cos}(k) + \mathbf{sin}(k)\\ \frac{\partial}{\partial w} c_2 &= -\mathbf{cos}(k) - \mathbf{sin}(k)\\ \frac{\partial}{\partial w} c_3 &= -\mathbf{cos}(k) + \mathbf{sin}(k)\\ \frac{\partial}{\partial w} c_4 &= \mathbf{cos}(k) - \mathbf{sin}(k)\end{aligned} \tag{5.36}$$

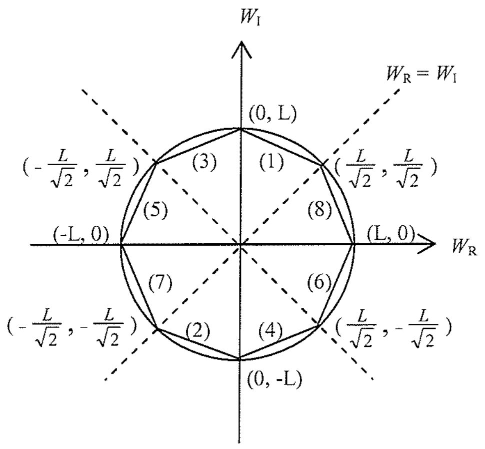
Taking the penalty term as defined in the previous section, it can be seen that:

$$\begin{aligned} &2\sigma \sum_{i=1}^{I} [c_i(\boldsymbol{w})]_z \frac{\partial}{\partial \boldsymbol{w}} c_i(\boldsymbol{w})\\ &= \sum_{k=0}^{N-1} 2\sigma \{ \left[\boldsymbol{w}^{\mathrm{T}} \mathbf{cos}(k) + \boldsymbol{w}^{\mathrm{T}} \mathbf{sin}(k) - L \right]_z (\mathbf{cos}(k) + \mathbf{sin}(k))\\ &\quad + \left[-\boldsymbol{w}^{\mathrm{T}} \mathbf{cos}(k) - \boldsymbol{w}^{\mathrm{T}} \mathbf{sin}(k) - L \right]_z (-\mathbf{cos}(k) - \mathbf{sin}(k))\\ &\quad + \left[-\boldsymbol{w}^{\mathrm{T}} \mathbf{cos}(k) + \boldsymbol{w}^{\mathrm{T}} \mathbf{sin}(k) - L \right]_z (-\mathbf{cos}(k) + \mathbf{sin}(k))\\ &\quad + \left[\boldsymbol{w}^{\mathrm{T}} \mathbf{cos}(k) - \boldsymbol{w}^{\mathrm{T}} \mathbf{sin}(k) - L \right]_z (\mathbf{cos}(k) - \mathbf{sin}(k)) \} \end{aligned} \tag{5.37}$$

The update equation for the CGA is obtained by substituting the above into (5.21). Similarly for the SD algorithm.

5.6.2 Approximating with eight linear constraints

The graph below shows how eight linear constraints represent a much more accurate approximation of the quadratic constraint than four linear constraints. This is because the majority of the original feasible region is covered by the approximations. The maximum regions which are not included in the approximation is between L and $\frac{\sqrt{2+\frac{2}{\sqrt{2}}}}{2}L$, so error is only 0.7dB maximum.



W_I
$W_R = W_I$
(0, L)
$(-\frac{L}{\sqrt{2}}, \frac{L}{\sqrt{2}})$
$(\frac{L}{\sqrt{2}}, \frac{L}{\sqrt{2}})$
(3)
(1)
(5)
(8)
(-L, 0)
(L, 0)
W_R
(7)
(6)
$(-\frac{L}{\sqrt{2}}, -\frac{L}{\sqrt{2}})$
(2)
(4)
$(\frac{L}{\sqrt{2}}, -\frac{L}{\sqrt{2}})$
(0, -L)

Figure 5.8: Approximating with eight linear constraints.

The equations for the eight linear constraints are:

$$\begin{aligned}
&(1) \quad W_I < \left(1-\sqrt{2}\right)W_R + L \\
&(2) \quad W_I > \left(1-\sqrt{2}\right)W_R - L \\
&(3) \quad W_I < \left(\sqrt{2}-1\right)W_R + L \\
&(4) \quad W_I > \left(\sqrt{2}-1\right)W_R - L \\
&(5) \quad W_I < \left(\frac{1}{\sqrt{2}-1}\right)W_R + \frac{L}{\sqrt{2}-1} \\
&(6) \quad W_I > \left(\frac{1}{\sqrt{2}-1}\right)W_R - \frac{L}{\sqrt{2}-1} \\
&(7) \quad W_I > \left(\frac{1}{1-\sqrt{2}}\right)W_R - \frac{L}{\sqrt{2}-1} \\
&(8) \quad W_I < \left(\frac{1}{1-\sqrt{2}}\right)W_R - \frac{L}{\sqrt{2}-1}
\end{aligned} \tag{5.38}$$

The eight linear constraints can be rewritten as:

$$
\begin{aligned}
&c_1: && \left(\sqrt{2}-1\right)W_R + W_I - L < 0 \\
&c_2: && \left(1-\sqrt{2}\right)W_R - W_I - L < 0 \\
&c_3: && \left(1-\sqrt{2}\right)W_R + W_I - L < 0 \\
&c_4: && \left(\sqrt{2}-1\right)W_R - W_I - L < 0 \\
&c_5: && \left(\frac{1}{1-\sqrt{2}}\right)W_R + W_I - \frac{L}{\sqrt{2}-1} < 0 \\
&c_6: && \left(\frac{1}{\sqrt{2}-1}\right)W_R - W_I - \frac{L}{\sqrt{2}-1} < 0 \\
&c_7: && \left(\frac{1}{1-\sqrt{2}}\right)W_R - W_I - \frac{L}{\sqrt{2}-1} < 0 \\
&c_8: && \left(\frac{1}{\sqrt{2}-1}\right)W_R + W_I - \frac{L}{\sqrt{2}-1} < 0
\end{aligned}
\tag{5.39}
$$

Once again, the DFT must be used to transform the problem to the time domain. By (5.33), the constraints become:

$$
\begin{aligned}
&c_1: && \left(\sqrt{2}-1\right)\boldsymbol{w}^{\mathrm{T}}\mathbf{cos}(k) + \boldsymbol{w}^{\mathrm{T}}\mathbf{sin}(k) - L < 0 \\
&c_2: && \left(1-\sqrt{2}\right)\boldsymbol{w}^{\mathrm{T}}\mathbf{cos}(k) - \boldsymbol{w}^{\mathrm{T}}\mathbf{sin}(k) - L < 0 \\
&c_3: && \left(1-\sqrt{2}\right)\boldsymbol{w}^{\mathrm{T}}\mathbf{cos}(k) + \boldsymbol{w}^{\mathrm{T}}\mathbf{sin}(k) - L < 0 \\
&c_4: && \left(\sqrt{2}-1\right)\boldsymbol{w}^{\mathrm{T}}\mathbf{cos}(k) - \boldsymbol{w}^{\mathrm{T}}\mathbf{sin}(k) - L < 0 \\
&c_5: && \left(\frac{1}{1-\sqrt{2}}\right)\boldsymbol{w}^{\mathrm{T}}\mathbf{cos}(k) + \boldsymbol{w}^{\mathrm{T}}\mathbf{sin}(k) - \frac{L}{\sqrt{2}-1} < 0 \\
&c_6: && \left(\frac{1}{\sqrt{2}-1}\right)\boldsymbol{w}^{\mathrm{T}}\mathbf{cos}(k) - \boldsymbol{w}^{\mathrm{T}}\mathbf{sin}(k) - \frac{L}{\sqrt{2}-1} < 0 \\
&c_7: && \left(\frac{1}{1-\sqrt{2}}\right)\boldsymbol{w}^{\mathrm{T}}\mathbf{cos}(k) - \boldsymbol{w}^{\mathrm{T}}\mathbf{sin}(k) - \frac{L}{\sqrt{2}-1} < 0 \\
&c_8: && \left(\frac{1}{\sqrt{2}-1}\right)\boldsymbol{w}^{\mathrm{T}}\mathbf{cos}(k) + \boldsymbol{w}^{\mathrm{T}}\mathbf{sin}(k) - \frac{L}{\sqrt{2}-1} < 0
\end{aligned}
\tag{5.40}
$$

As in the four linear constraints case, the gradient of the penalty term must be calculated. The gradients of the eight linear constraints are:

$$\frac{\partial}{\partial w}c_1(w) = \left(\sqrt{2}-1\right)\mathbf{cos}(k) + \mathbf{sin}(k)$$

$$\frac{\partial}{\partial w}c_2(w) = \left(1-\sqrt{2}\right)\mathbf{cos}(k) - \mathbf{sin}(k)$$

$$\frac{\partial}{\partial w}c_3(w) = \left(1-\sqrt{2}\right)\mathbf{cos}(k) + \mathbf{sin}(k)$$

$$\frac{\partial}{\partial w}c_4(w) = \left(\sqrt{2}-1\right)\mathbf{cos}(k) - \mathbf{sin}(k) \tag{5.41}$$

$$\frac{\partial}{\partial w}c_5(w) = \left(\frac{1}{1-\sqrt{2}}\right)\mathbf{cos}(k) + \mathbf{sin}(k)$$

$$\frac{\partial}{\partial w}c_6(w) = \left(\frac{1}{\sqrt{2}-1}\right)\mathbf{cos}(k) - \mathbf{sin}(k)$$

$$\frac{\partial}{\partial w}c_7(w) = \left(\frac{1}{1-\sqrt{2}}\right)\mathbf{cos}(k) - \mathbf{sin}(k)$$

$$\frac{\partial}{\partial w}c_8(w) = \left(\frac{1}{\sqrt{2}-1}\right)\mathbf{cos}(k) + \mathbf{sin}(k)$$

Taking the penalty term as defined in the previous section, it can be seen that the gradient of the penalty term, over all frequencies k, is:

$$2\sigma\sum_{i=1}^{I}[c_i(w)]_z \frac{\partial}{\partial w}c_i(w) \tag{5.42}$$

$$\begin{aligned}
=\sum_{k=0}^{N-1} 2\sigma\Big\{ & \left[\left(\sqrt{2}-1\right) w^{\mathrm{T}}\mathbf{cos}(k) + w^{\mathrm{T}}\mathbf{sin}(k) - L\right]_z \left[\left(\sqrt{2}-1\right)\mathbf{cos}(k) + \mathbf{sin}(k)\right] \\
+ & \left[\left(1-\sqrt{2}\right) w^{\mathrm{T}}\mathbf{cos}(k) - w^{\mathrm{T}}\mathbf{sin}(k) - L\right]_z \left[\left(1-\sqrt{2}\right)\mathbf{cos}(k) - \mathbf{sin}(k)\right] \\
+ & \left[\left(1-\sqrt{2}\right) w^{\mathrm{T}}\mathbf{cos}(k) + w^{\mathrm{T}}\mathbf{sin}(k) - L\right]_z \left[\left(1-\sqrt{2}\right)\mathbf{cos}(k) + \mathbf{sin}(k)\right] \\
+ & \left[\left(\sqrt{2}-1\right) w^{\mathrm{T}}\mathbf{cos}(k) - w^{\mathrm{T}}\mathbf{sin}(k) - L\right]_z \left[\left(\sqrt{2}-1\right)\mathbf{cos}(k) - \mathbf{sin}(k)\right] \\
+ & \left[\left(\frac{1}{1-\sqrt{2}}\right) w^{\mathrm{T}}\mathbf{cos}(k) + w^{\mathrm{T}}\mathbf{sin}(k) - \frac{L}{\sqrt{2}-1}\right]_z \left[\left(\frac{1}{1-\sqrt{2}}\right)\mathbf{cos}(k) + \mathbf{sin}(k)\right] \\
+ & \left[\left(\frac{1}{\sqrt{2}-1}\right) w^{\mathrm{T}}\mathbf{cos}(k) - w^{\mathrm{T}}\mathbf{sin}(k) - \frac{L}{\sqrt{2}-1}\right]_z \left[\left(\frac{1}{\sqrt{2}-1}\right)\mathbf{cos}(k) - \mathbf{sin}(k)\right] \\
+ & \left[\left(\frac{1}{1-\sqrt{2}}\right) w^{\mathrm{T}}\mathbf{cos}(k) - w^{\mathrm{T}}\mathbf{sin}(k) - \frac{L}{\sqrt{2}-1}\right]_z \left[\left(\frac{1}{1-\sqrt{2}}\right)\mathbf{cos}(k) - \mathbf{sin}(k)\right] \\
+ & \left[\left(\frac{1}{\sqrt{2}-1}\right) w^{\mathrm{T}}\mathbf{cos}(k) + w^{\mathrm{T}}\mathbf{sin}(k) - \frac{L}{\sqrt{2}-1}\right]_z \left[\left(\frac{1}{\sqrt{2}-1}\right)\mathbf{cos}(k) + \mathbf{sin}(k)\right]\Big\}
\end{aligned}$$

The update equation for the CG is obtained by substituting the above into (5.21). Similarly for the SD algorithm.

The process for using higher numbers of linear constraints is very similar to that described above for four and eight linear constraints. It should be noted that although increasing the number of linear constraints improves the accuracy of the approximation, it also greatly increases the number of equations involved, and hence the computation time required by the algorithm to find the minimum.

5.7 Simulation example: Various formulations with SD algorithm

In previous sections, various formulations of constrained optimisation problems were presented. In the sections below these are studied using simulations with SD and CG as example algorithms. Simulations using MATLAB of the formulations investigated in previous sections have been conducted and the prediction system is considered. The performance of the formulations with two algorithms are performed for the prediction system problem with quadratic constraints $|W(k)|^2 < L(k)$. The results of the problem with linear approximations of quadratic constraints are compared with the results of the original quadratically constrained problem. Results of first and second order penalty functions are also compared and studied. Therefore four main performances will be compared in this section, they are: the uses of first order penalty function, the uses of second order penalty function, four linear constraints approximation and eight linear constraints approximation of quadratic constraints.

The prediction system considered here is the same system used in previous sections; using a sampling frequency of 1kHz, the reference signal is equal to the desired signal delayed by one sample and the desired signal is generated by passing white noise through a simple low pass filter. The performance of the algorithms will be compared with a filter length of 128. The stopping criterion used is 2,000 iterations. The magnitude constraints $|W(k)|^2 < L(k)$ is considered, where L is selected as a linearly line from 0dB (at 0Hz) to 15dB (at 500Hz). In the simulations, analytic gradient is used in order to focus on the study of the various optimisation formulations and avoid stochastic gradient noise. Constrained adaptive filter with stochastic gradient are studied later in chapter 9.

The values of the penalty parameter σ and convergence coefficient μ were established by trial and error, as there is currently no theory concerning the values. The values that gave the most stable convergence to the solution (from a range of chosen values), were used for the remainder of the simulations, these were $\sigma = 0.001$ and $\mu = 2$. Values of σ that are too large, result in instability and ill conditioning; values that are too small do not effectively penalise the objective function, and therefore lead to a solution that is like an unconstrained problem. Currently there are no definitive theories regarding bounds or optimal values of μ and σ, if these can be established they can be very useful in increasing the accuracy and efficiency of algorithms for solving constrained optimisation problems for adaptive filters in the frequency domain. Chapter 7 will conduct the theory of this analysis. In these simulations fixed values of μ and σ were therefore used.

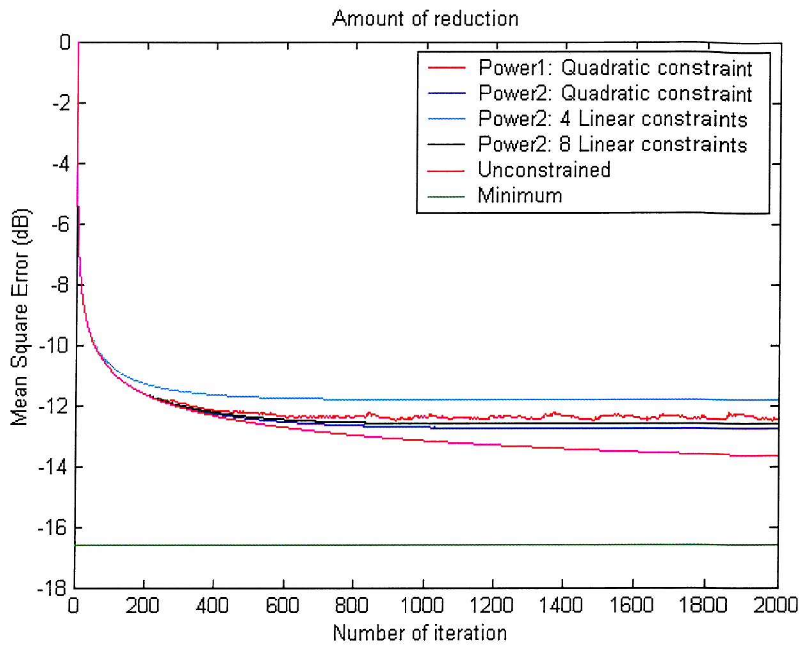
The table 5.1 below shows results from the simulation that was run for various formulations and constraints. The minimum point (mean square error), the number of floating point operations and the time taken to run an algorithm will be used to compare the performance between the linear approximations and the original quadratically constraint; also being compared are the difference of using first and second order penalty functions. The minimum point indicates to the value of mean square error at 2000th iteration. The floating point operations is the number of computation operations generated by Matlab program. The starting active point in table 5.1 is the iteration at which the constraint becomes active. The starting active point can be used to compare the linear approximations with the original quadratic constraint. The simulations performed for this chapter were conducted using the mathematical program MATLAB, using a PC with a Pentium III 32-bit processor and 128 MB RAM. The time taken to run the simulations therefore depends on the hardware used.

	Minimum point	**Starting active**	**Floating point**	**Time (in sec)**
Optimal	-16.59			
Unconstrained	-13.68		153.9e+006	4.69
First order penalty function:				
Quadratic constraint	-12.47	221	173.6e+006	6.10
Second order penalty function:				
4 Linear constraints	-11.96	38	684.6e+006	35.81
8 Linear constraints	-12.63	160	1349.3e+006	65.90
Quadratic constraint	-12.77	232	193.8e+006	6.57

Table 5.1: Performance of using first and second order penalty functions, and the performance of the approximation of the quadratic constraint with four and eight linear constraints, for the steepest descent algorithm.

We can see from figure 5.9(a) that the performance of second order penalty function with original quadratically constraint smoothly converges to the minimum, where as using first order penalty function convergence is noisier. This is due to their continuity and discontinuity at the boundary of the feasible region. However, the convergence performance of using first order penalty function is not far off from the second order penalty function, it also has less number of floating point operations. Also the noisy convergence performance can be reduced if smaller σ is used. Table 5.1 shows that the four and eight linear approximations converged to a minimum that was close to that of the quadratically constrained case, within 1dB. The linear constraints were giving a fairly accurate approximation of the original quadratic constraint. The approximation became more accurate as the number of linear constraints increased, but it also greatly increase the number of equations involved as shown in the floating point operations in table 5.1. The performance of the linear approximations can also be compared by the starting active iteration. The starting active iteration of four and eight linear approximation constraints is 38 and 160 respectively, and the original quadratically constraint is 232, i.e. increasing the numbers linear constraints can improve the accuracy of the approximations. Figure 5.9(b) shows the magnitude responses after convergence, illustrating the effect of the quadratic constraint, the four linear approximation constraints, the eight linear approximation constraints, using first order and second order penalty functions.

(a)



(b)

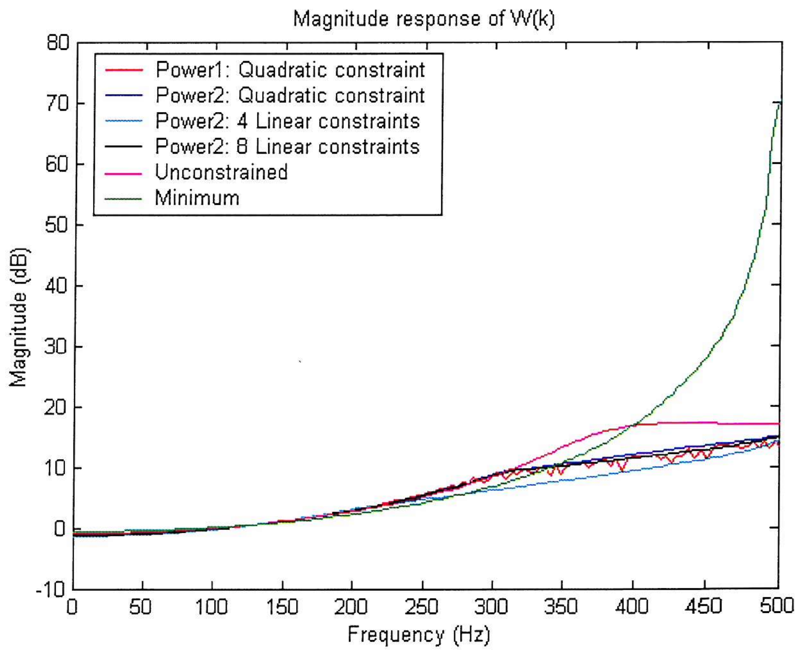


Figure 5.9: Comparison of the performance between unconstrained, four linear constraints, eight linear constraints, first and second order penalty functions with quadratic constraint by using steepest descent algorithm in a prediction system: (a) comparison of the convergence characteristics, (b) comparison of the magnitude responses after convergence.

5.8 Simulation example: Various formulations with CG algorithm

MATLAB simulation using the formulations investigated in this chapter with conjugate gradient algorithm is now conducted. The table below sets out the results from the simulations. For ease of comparison with the performance of steepest descent algorithm, the prediction system from the previous section is used here.

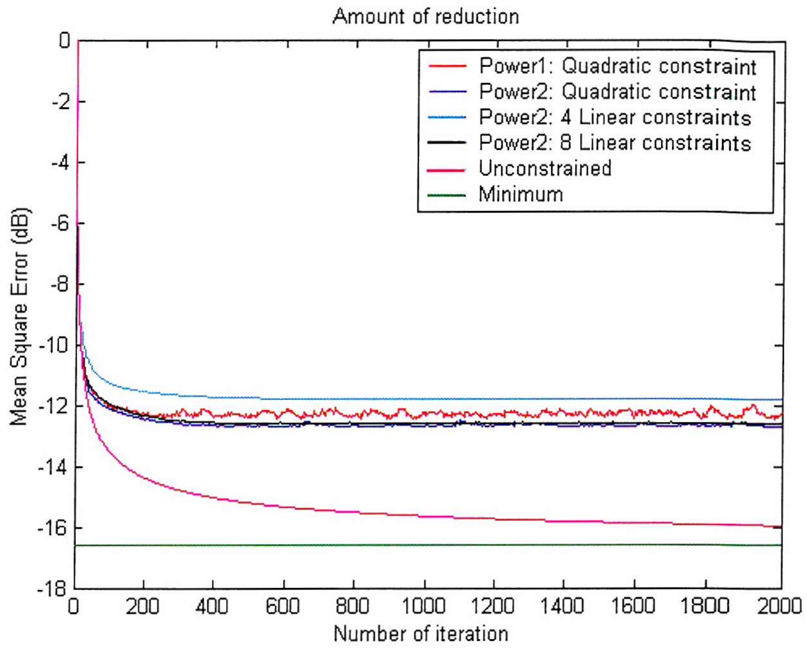
	Minimum point	**Starting active**	**Floating point**	**Time (sec)**
Optimal	-16.59			
Unconstrained	-15.96		155.3e+006	4.99
First order penalty function:				
Quadratic constraint	-12.26	19	174.7e+006	6.52
Second order penalty function:				
4 Linear constraints	-12.06	9	686.4e+006	35.89
8 Linear constraints	-12.64	16	1351.1e+006	67.94
Quadratic constraint	-12.80	36	194.8e+006	6.76

Table 5.2: Performance of using first and second order penalty functions, and the performance of the approximation of the quadratic constraint with four and eight linear constraints, for the conjugate gradient algorithm.

We can see from tables 5.1 and 5.2 for each of the cases, the CGA achieves a minimum much closer to the optimal minimum than the SD. Tables 5.1 and 5.2 also show that the number of floating point operations and the computation time were very similar for the two algorithms, for all cases. The CGA used slightly more floating point operations and computation time than the SD, but the convergence rate was much faster. Because the comparison of the algorithms can be conducted on the part where the constraints had become active. That is, the sections of the graph which relates to iterations after the starting active point. For example, the starting active points for the original quadratically constraints were 36 and 232 for the CGA and SD respectively, i.e. it showed that the CGA has a better convergence performance then SD algorithm. The convergence performance of all cases with CG algorithm is shown in figure 5.10(a). It shows the performance of the approximation of the quadratic constraint with four and eight linear constraints; the linear approximations converged to a minimum, which was close to the quadratically constrained case. With eight linear constraints the difference was very small: within 0.2dB of the minimum achieved using quadratic constraints. This shows that using eight linear constraints gives a very accurate approximation of the quadratic constraints. Figure 5.10(a) also shows the performance of using first and second order penalty functions with the quadratically constraint, it gave a similar result as SD algorithm in figure 5.9; it had a noisy convergence when first order penalty function is used and a smoothly convergence when second order penalty function is used, but the performance is not far from each together. Figure 5.10(b) shows the magnitude responses after convergence, illustrating the effect of the quadratic constraint, the four linear approximation constraints, the eight linear approximation constraints, using first order and second order penalty functions.

For ease to compare both algorithms, figure 5.11(a) and (b) shows the comparison of the two algorithms for the unconstrained and linear approximation constraints cases. Figure 5.11(a) shows the unconstrained and four linear constraints cases, it can be seen that the CGA with constraints reached the minimum approximately 250 iterations after the first starting active point, whereas the SD took much longer, over 400 iterations. Also in tables 5.1 and 5.2 show that the starting active points were 9 and 38 for the CGA and SD respectively. It can also be seen that in the unconstrained case, CG has a better convergence performance than the SD. Figure 5.11(b) shows a comparison of the two algorithms for the unconstrained and eight linear approximation constraints cases. It can be seen that the algorithms reached the minimum approximately 450 and over 700 iterations after the starting active point for the CGA and SD respectively, and the starting active points for the algorithms were 16 for the CGA and 160 for the SD. This therefore demonstrates that the CGA had a faster convergence rate than the SD, in this example application.

(a)



(b)

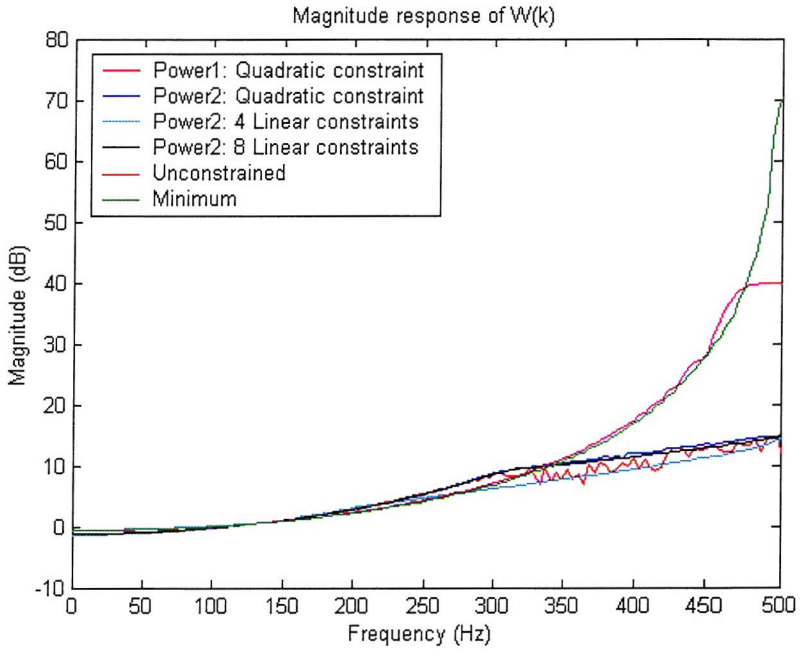
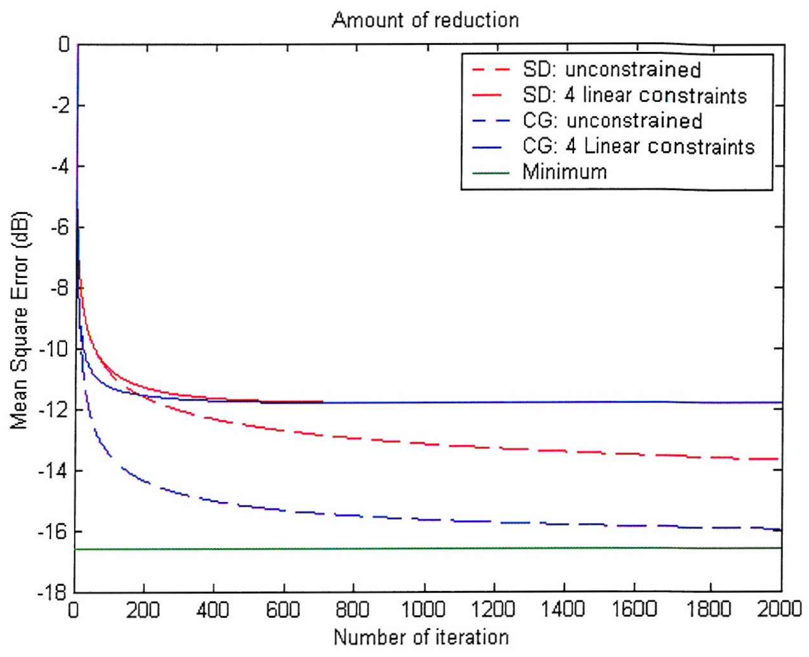


Figure 5.10: Comparison of the performance between unconstrained, four linear constraints, eight linear constraints, first and second order penalty functions with quadratic constraint by using conjugate gradient algorithm in a prediction system: (a) comparison of the convergence characteristics, (b) comparison of the magnitude responses after convergence.

(a)



(b)

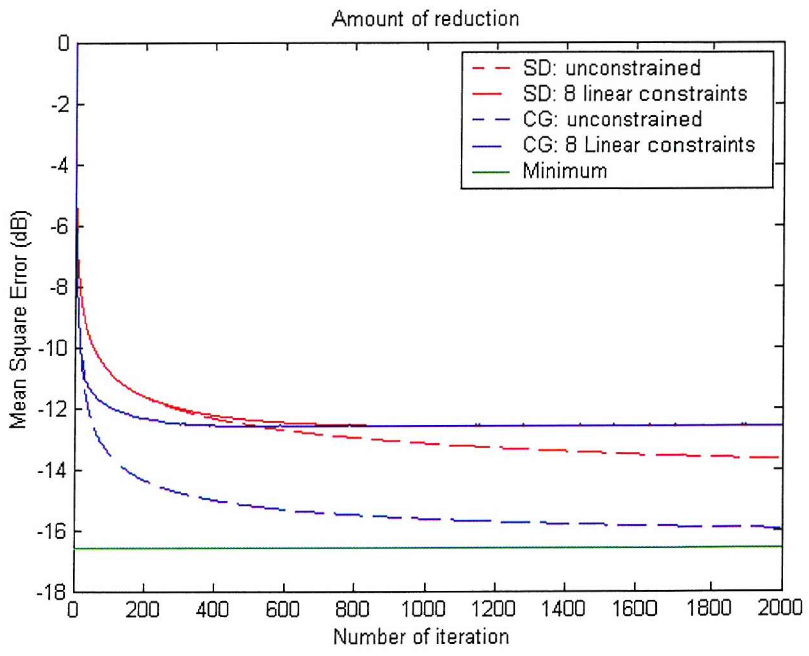


Figure 5.11: Comparison of the convergence performance between conjugate gradient and steepest descent algorithms: (a) with unconstrained and four linear constraints, (b) with unconstrained and eight linear constraints.

5.9 Conclusions

In this chapter, methods to solve constrained optimisation problems suitable for use in adaptive filtering were developed. SD and CGA algorithms were used as example algorithms. The penalty function method was chosen to convert the constrained problems into an unconstrained one with simple penalty function forms of $\max(c_i, 0)$ and $\max(c_i, 0)^2$. The constrained optimisation problem considered here had quadratic cost function (mean square error) with quadratic constraints (magnitude constraint $|W(k)|^2 < L(k)$). Due to the computational complexity of quadratic constraints with second order penalty function, the approximation of the original quadratic constraints with linear constraints was suggested. The performances of the formulations discussed in this chapter were: (1) Using first order penalty function with quadratic constraint; (2) Using second order penalty function with quadratic constraint; (3) Second order penalty function with linear approximations: four linear constraints and eight linear constraints; (4) the performance between SD and CGA algorithms.

For the prediction problem, a set of four and a set of eight linear constraints were used to compare the unconstrained, linearly constrained and quadratically constrained problems. The approximation became more accurate as the number of linear constraints increased. The simulations showed that using four linear constraints gave results that were within 1dB of the minimum achieved by the quadratic constraints. Using eight linear constraints gave results that very closely approximated the quadratically constrained case; the difference between the two was within 0.2dB. However, due to the large number of equations involved, the computation time needed to achieve such a high degree of accuracy of approximation using linear constraints was significantly increased. Overall, the simulations showed that using linear constraints with a penalty function formulation converged to a minimum that was very close to the quadratically constrained case.

The linear approximation case had a starting active point at a smaller number of iterations than the quadratically constrained case. This is due to the construction of the linear approximations; the feasible region has been reduced from that of the quadratic case. As the number of linear approximation constraints increased, the starting active point also increased. This is because the feasible region gave a more accurate approximation of the region defined by the quadratic constraint. As the number of linear approximation constraints increases, it is expected that the starting active point will tend to that of the quadratically constrained case.

The results demonstrated that the CGA had a better convergence performance than the SD for the constrained optimisation problem in the time domain in every case tested. The CGA will be modified for solving constrained optimisation problems for adaptive filters in the frequency domain in the next chapter. The LMS and bin-normalized LMS frequency domain with constraints will also be modified in the following chapter. We then will show that increasing the convergence rate can be achieved by incorporating other algorithms, for example the FDCGA and the Bin-normalised FDLMS algorithms. The performance of FDCGA, FDLMS and bin-normalized FDLMS will be compared later in part III.

In this chapter, the penalty method was used to convert constrained optimisation problems into unconstrained optimisation problems. Second and first order penalty functions have been investigated, and the results showed that using first order penalty function is not far off from using second order penalty function. Therefore the first order penalty function will be further developed for adaptive filtering applications. In constrained applications, the stability, convergence time, and fluctuations of the process are governed by the convergence coefficient μ and the penalty parameter σ. However, there is no definitive convergence theory that currently exists, and 'trial and error' has been used to find suitable values, i.e. it can cause inaccurate performance and slow convergence. A theoretical analysis of constrained optimisation with first order penalty function will therefore be constructed for the value of the convergence coefficient μ and the penalty parameter σ in chapter 7. Lets now modify the algorithms for the constrained adaptive filtering in the frequency domain with first order penalty function first in the following chapter 6, then we will implement the theoretical analysis in chapter 7.

Chapter 6

Algorithms for constrained adaptive FIR filters

6.1 Introduction

In chapter 5, formulations of constrained optimisation were studied and initial simulations of time-domain algorithm were performed using analytic gradient information. In this chapter algorithms are presented which can be used for constrained adaptive filters in practice. Firstly the algorithms use stochastic gradients estimated from measured signal, and so can be implemented in practice. Secondly, frequency-domain algorithms are developed which are more computationally efficient, faster converging and can incorporate frequency domain constraints directly. These algorithms are then used in part III for sound equalisation application.

The LMS is probably the most widely used algorithm in adaptive filtering for solving unconstraint problems. Some papers have been published that use constrained LMS-type algorithms to adapt FIR filters [Rafaely and Elliott, 2000b]. Rafaely and Elliott (2000b) apply the LMS algorithm to constrained optimisation problems for adaptive filters in the frequency domain, incorporating a second order penalty function. There have also been papers that present unconstrained frequency domain CGA [Lim and Un, 1993] and unconstrained frequency domain bin-normalized LMS algorithm (BN-FDLMS) [Shynk, 1992]. However there is no author so far that has presented the constrained CGA and constrained bin-normalized LMS algorithms. The aim of this thesis is to design efficient algorithms for constrained adaptive filtering, and the algorithms of CGA and BN-FDLMS are one of the possibilities. The constrained CGA in frequency domain and BN-FDLMS with constraints are therefore developed and implemented. The penalty method will be used for the implementation and first order penalty function is considered. This thesis adopts the new approach of adding constraints to a FDCGA and BN-FDLMS formulations of the problem, so that the derived method may have a wider range of practical applications, particularly in acoustics and audio. Performing calculations in the frequency domain will take advantage of the computational savings and frequency domain constraints will be applied. The performance of the algorithms will be shown and compared in chapter 9, and it will also show that increasing the convergence rate of the conventional LMS algorithm can be done by incorporating the FDCGA and BN-FDLMS algorithms.

In this chapter, the constrained FDLMS with a first order penalty function is first formulated, then followed by the implementation of the BN-FDLMS and FDCGA with constraints. As with the time domain implementation of chapter 4 and 5, penalty functions will be used to write a new objective function, so that the constrained optimisation problem becomes one of unconstrained optimisation. Simulations and results of the constrained algorithms will be presented in chapter 9 for the application of sound equalisation.

6.2 Formulation of the constrained frequency domain LMS algorithm

Rafaely and Elliott (2000b) have published constrained FDLMS algorithm solved with second order penalty function. In the following, we develop a modified constrained FDLMS algorithm that is solved by using a first order penalty function. The implementation of the formulation of the constrained FDLMS algorithm with first order penalty function is presented below.

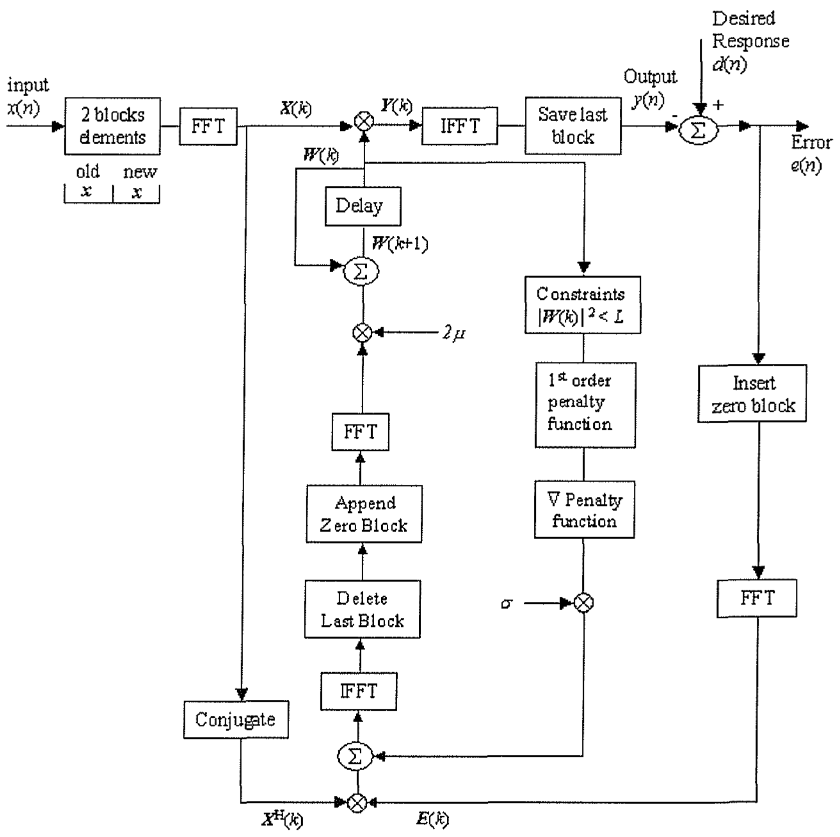


Figure 6.1: Block diagram of the FDLMS algorithm with constraints.

UNIVERSITY OF SOUTHAMPTON LIBRARY

As in section 3.5, the overlap-save method is applied to ensure the filtering achieve linear convolution. A block diagram of the constrained FDLMS algorithm as implemented in this section is presented in figure 6.1 and follows some of the notation used in Shynk (1992).

The time domain filtering equation and the update of the error signal is defined as:

$$e(n) = d(n) - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}(n) \tag{6.1}$$

where

$$\boldsymbol{w} = [w_0 \ w_1 \ \dots \ w_{N-1}]^{\mathrm{T}} \tag{6.2}$$

Let the frequency domain representation of the filter $\boldsymbol{W}$ be derived in matrix notation from the time domain representation as below:

$$\boldsymbol{W}(k) = \boldsymbol{F}\,[w_0(k) \ \dots \ w_{N-1}(k) \ \ \boldsymbol{0}]^{\mathrm{T}} \tag{6.3}$$

The gradient estimate is implemented as a correlation between the error and input signals. The time domain gradient estimate can be calculated using DFT's:

$$\tilde{\nabla}(k) = \boldsymbol{F}^{-1}\boldsymbol{X}^{\mathrm{H}}(k)\boldsymbol{E}(k) \tag{6.4}$$

The frequency domain input vector $\boldsymbol{X}(k)$ and error vector $\boldsymbol{E}(k)$ of size $2N$ are formulated using FFT on the corresponding time domain vectors as (detail refer to section 3.5):

$$\boldsymbol{X}(k) = diag\left\{\boldsymbol{F}\left[x(kN-N) \ \cdots \ x(kN-1) \ \ x(kN) \ \cdots \ x(kN+N-1)\right]^{\mathrm{T}}\right\} \tag{6.5}$$

where N elements from the previous block and N elements from the current block.

$$\boldsymbol{E}(k) = \boldsymbol{F}\,[\boldsymbol{0} \ \ e(kN) \ \cdots \ e(kN+N-1)\,]^{\mathrm{T}} \tag{6.6}$$

where $\boldsymbol{0}$ is the vector containing N zeros and is used to ensure linear correlation in the update of the adaptive filter.

Now consider the constrained optimisation problem as follows:

$$\begin{aligned} &\text{Minimise} && J = E\,[e^{\mathrm{T}}(n)e(n)] \\ &\text{Subject to} && c_i(k) = |\boldsymbol{W}(k)|^2 - L(k) < 0 \end{aligned} \tag{6.7}$$

where a limit L is placed on the magnitude of the filter W for any given frequency k. L can take the form of a scalar or functions.

Using the first order penalty function, the new objective function can be written as:

$$\text{Minimise} \quad J = E[\boldsymbol{e}^{\mathrm{T}}(n)\boldsymbol{e}(n)] + \sigma\sum_{i=1}^{I}\max(c_i(\boldsymbol{w}),0) \tag{6.8}$$

The gradient of the new objective function is calculated as:

$$\nabla J = \boldsymbol{X}^{\mathrm{H}}(k)\boldsymbol{E}(k) + \sigma\sum_{i=1}^{I}\left[\frac{\partial}{\partial \boldsymbol{w}}c_i(\boldsymbol{w})\right]_z \tag{6.9}$$

The partial derivative with respect to $\boldsymbol{w}$ for c_k is defined as equation (5.17), which is:

$$\frac{\partial}{\partial \boldsymbol{w}}c_k = 2\boldsymbol{a}_k^*\boldsymbol{a}_k^{\mathrm{T}}\boldsymbol{w} = 2\boldsymbol{W}(k)\boldsymbol{a}_k^* \tag{6.10}$$

Therefore the equation (6.9) becomes,

$$\nabla J = \boldsymbol{X}^{\mathrm{H}}(k)\boldsymbol{E}(k) + 2\sigma\sum_{k=0}^{N-1}\left[\boldsymbol{W}(k)\boldsymbol{a}_k^*\right]_z \tag{6.11}$$

Using inverse DFT transforms the gradient (6.11) to the time domain, so the gradient of the new objective function becomes:

$$\Rightarrow \qquad \nabla J = \boldsymbol{F}^{-1}\left(\boldsymbol{X}^{\mathrm{H}}(k)\boldsymbol{E}(k) + 2\sigma N\left[\boldsymbol{W}(k)\right]_z\right) \tag{6.12}$$

The update equation incorporating the penalty term for FDLMS is:

$$\boldsymbol{W}(k+1) = \boldsymbol{W}(k) + 2\mu\boldsymbol{F}\boldsymbol{g}\boldsymbol{F}^{-1}\left(\boldsymbol{X}^{\mathrm{H}}(k)\boldsymbol{E}(k) + 2\sigma N\left[\boldsymbol{W}(k)\right]_z\right) \tag{6.13}$$

where $\boldsymbol{g}$ is chosen, equation (3.38), to allow the non-causal elements of the time domain gradient to be set to zero. Note that the FDLMS is based on block LMS, therefore the convergence coefficient μ in (6.13) is divided by the block size.

6.3 Formulation of the constrained frequency domain bin-normalised LMS algorithm

In this section we will develop the formulation of the frequency domain bin-normalised LMS algorithm with constraints. The BN-FDLMS builds on the FDLMS where the convergence rate of the FDLMS is improved by having frequency dependent convergence coefficient μ introduced in the update equation. Therefore the implementation of constrained BN-FDLMS is basically the same as the FDLMS, except the convergence coefficient μ.

The general form of the update equation of BN-FDLMS algorithm is expressed as [Shynk, 1992]

$$\boldsymbol{W}(k+1) = \boldsymbol{W}(k) + 2\boldsymbol{\mu}(k)\boldsymbol{F}\boldsymbol{g}\boldsymbol{F}^{-1}\boldsymbol{X}^{\mathrm{H}}(k)\boldsymbol{E}(k) \tag{6.14}$$

Generally, each convergence coefficient is varied according to the power of the reference signal in that frequency bin, and $\boldsymbol{\mu}(k)$ is diagonal time-varying matrix that contains the coefficients $\mu_m(k)$, where $m = 0, \ldots, M\text{-}1$. It is to compensate for the power variation by using convergence coefficients that are inversely proportional to the power levels in the frequency bins. The coefficient $\mu_m(k)$ is defined as

$$\mu_m(k) = \frac{\mu}{\hat{\mathrm{P}}_m(k)} \tag{6.15}$$

where μ is a fixed scalar and $\hat{\mathrm{P}}_m(k)$ is the power spectrum estimate of the reference signal in the mth bin. It is calculated as

$$\hat{\mathrm{P}}_m(k) = \beta\,\hat{\mathrm{P}}_m(k-1) + (1-\beta)\left|X_m(k)\right|^2 \tag{6.16}$$

where β is a forgetting scalar and the initial value $\hat{\mathrm{P}}_m(0)$ is chosen that depends on the initial power spectrum estimate of $X_m(k)$. Detail can be referred to chapter 3. Also see Shynk (1987) and Janssen (1987).

Now consider the constrained optimisation problem (6.7) and using first order penalty function. The objective function considered for the BN-FDLMS is the same as the FDLMS which stated in equation (6.8). The constrained adaptive BN-FDLMS can be formulated by the following steps. Note that the BN-FDLMS implemented here is also based on the overlap-save method. To simplify notation, we first define:

$$\boldsymbol{D}_{\mathrm{f}} = [\boldsymbol{I}_N \quad \boldsymbol{0}_N], \qquad \boldsymbol{D}_{\mathrm{b}} = [\boldsymbol{0}_N \quad \boldsymbol{I}_N],$$

Matrices $\boldsymbol{D}_{\mathrm{f}}$ and $\boldsymbol{D}_{\mathrm{b}}$ are used below to set to zero the non-causal and causal parts (see section 3.5).

Start with initialisation:

$$\boldsymbol{W}(0) = [0 \ldots \ldots 0]^{\mathrm{T}} \tag{6.17}$$

and

$$\hat{\mathrm{P}}_m(0) = \delta_m, \qquad m = 0, \ldots, 2N\text{-}1 \tag{6.18}$$

where δ_m is a positive number which estimated by the initial power spectrum $X_m(k)$.

Then, set the magnitude limit L and formulate the constraints by setting:

$$c_i(w(k)) = |\boldsymbol{W}(k)|^2 - L(k) < 0 \tag{6.19}$$

Consider the first order penalty function,

$$\sum_{i=1}^{I} \max(c_i(w), 0) \tag{6.20}$$

and define,

$$[\boldsymbol{W}(k)]_z = \boldsymbol{W}(k)\left(\frac{sign(c_i(\boldsymbol{w}_k)) + 1}{2}\right) \tag{6.21}$$

For each new block, compute:

$$\boldsymbol{X}(k) = diag\left\{\boldsymbol{F}[x(kN-N) \quad \cdots \quad x(kN-1) \quad x(kN) \quad \cdots \quad x(kN+N-1)]^{\mathrm{T}}\right\} \tag{6.22}$$

$$\boldsymbol{Y}(k) = \boldsymbol{X}(k)\boldsymbol{W}(k) \tag{6.23}$$

$$\boldsymbol{y}(k) = \boldsymbol{F}^{-1}\{\boldsymbol{D}_{\mathrm{b}}\boldsymbol{Y}(k)\} \tag{6.24}$$

$$\boldsymbol{e}(k) = \boldsymbol{d}(k) - \boldsymbol{y}(k) \tag{6.25}$$

$$\boldsymbol{E}(k) = \boldsymbol{F}\{\boldsymbol{D}_{\mathrm{b}}^{\mathrm{T}}\boldsymbol{e}(k)\} \tag{6.26}$$

Now compute the gradient search direction with constraints,

$$\nabla J(k) = \boldsymbol{X}^{\mathrm{H}}(k)\boldsymbol{E}(k) + 2\sigma \sum_{k=0}^{N-1} \left[\boldsymbol{W}(k)\boldsymbol{a}_k^*\right]_z \tag{6.27}$$

The gradient of the new objective function in time domain is then calculated as:

$$\nabla J(k) = \boldsymbol{F}^{-1}\left(\boldsymbol{X}^{\mathrm{H}}(k)\boldsymbol{E}(k) + 2\sigma N\left[\boldsymbol{W}(k)\right]_z\right) \tag{6.28}$$

Then calculate the convergence coefficients at each frequency bins:

$$\boldsymbol{\mu}(k) = \mu_m(k) = \mu\, \mathrm{diag}\left\{\hat{\mathrm{P}}_0^{-1}(k),\ \cdots\ ,\hat{\mathrm{P}}_{2N\text{-}1}^{-1}(k)\right\} \tag{6.29}$$

where

$$\hat{\mathrm{P}}_m(k) = \beta\, \hat{\mathrm{P}}_m(k-1) + (1-\beta)\left|X_m(k)\right|^2,\quad m = 0, \ldots, 2N\text{-}1 \tag{6.30}$$

Compute the update equation incorporating with the penalty term:

$$\boldsymbol{W}(k+1) = \boldsymbol{W}(k) + 2\boldsymbol{\mu}(k)\boldsymbol{F}\boldsymbol{g}\boldsymbol{F}^{-1}\left(\boldsymbol{X}^{\mathrm{H}}(k)\boldsymbol{E}(k) + 2\sigma N\left[\boldsymbol{W}(k)\right]_z\right) \tag{6.31}$$

Then replace $\boldsymbol{W}_0$ with $\boldsymbol{W}_{k+1}$. Continue iterating until the minimum is reached.

Note that since the BN-FDLMS is based on the FDLMS, the convergence coefficient $\mu(k)$ needs to be dividing by the block size.

6.4 Formulation of the constrained frequency domain conjugate gradient algorithm

Similar to FDLMS, the overlap-save method is also used to implement FDCGA. The BCGA will be developed in the frequency domain with constraints using FFT to transform the data from the time domain to the frequency domain. When the BCGA is implement in the frequency domain, the one block output delay that is enforced by the system must be considered.

A block size of N and hence an FFT of size $2N$ are used. The FFT has this size to produce a 50% overlap, as discussed in previous chapters, to allow for the causal and non-causal parts of the vector and prevent circular convolution when transforming back to the time domain [Lim and Un, 1996]. The constrained frequency-domain block CG algorithm outlined below [Chang and Willson, 2000] is implemented using FFT, a block diagram of the constrained FDCGA algorithm implemented in this section is presented in figure 6.2.

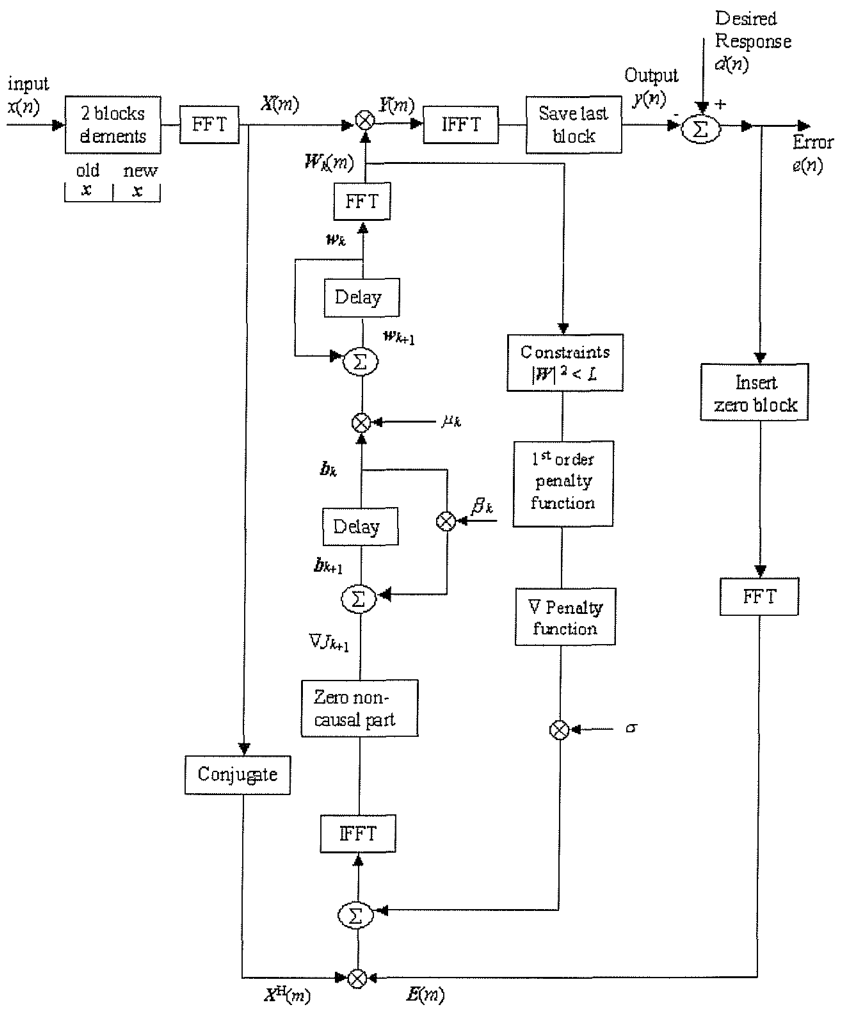


Figure 6.2: Block diagram of the FDCGA algorithm with constraints.

Again, to simplify notation, we use:

$$\boldsymbol{D}_{\mathrm{f}} = [\boldsymbol{I}_N \quad \boldsymbol{0}_N], \qquad \boldsymbol{D}_{\mathrm{b}} = [\boldsymbol{0}_N \quad \boldsymbol{I}_N], \tag{6.32}$$

The $2N\times1$ frequency domain vector $\boldsymbol{W}_j$ is given by

$$\boldsymbol{W}_k = \boldsymbol{F}\{\boldsymbol{D}_\mathrm{f}^\mathrm{T}\boldsymbol{w}_k\} \tag{6.33}$$

$\boldsymbol{W}_k$ is transformed, using the FFT, from $\boldsymbol{w}_k$, which has been padded with N zeros.

The input signal $\boldsymbol{x}$ contains $2N$ elements: N from the previous block and N from the current block. This is, hence, overlapping half of the vector and providing only N new data points for each iteration,

$$\boldsymbol{x}_{j\text{-}1} = [x_{jN\text{-}N}, \ldots, x_{jN\text{-}1}]^\mathrm{T}, \qquad \boldsymbol{x}_j = [x_{jN}, \ldots, x_{jN+N\text{-}1}]^\mathrm{T} \tag{6.34}$$

Therefore, the input signal $\boldsymbol{X}_j$ transformed to the frequency domain is written as below, is $2N\times2N$, and is a diagonal matrix:

$$\boldsymbol{X}_j = [\boldsymbol{D}_\mathrm{f}^\mathrm{T}\ \boldsymbol{D}_\mathrm{b}^\mathrm{T}]\boldsymbol{F}\{[\boldsymbol{x}_{j-1}^\mathrm{T}\ \boldsymbol{x}_j^\mathrm{T}]^\mathrm{T}\} \tag{6.35}$$

$\boldsymbol{X}_j$ is transformed from a circulant matrix in the first column.

Also, the $2N\times1$ frequency domain error vector $\boldsymbol{E}_k$, the search direction vector $\boldsymbol{B}_k$, and the gradient vector are defined by:

$$\boldsymbol{E}_k = \boldsymbol{F}\{\boldsymbol{D}_\mathrm{b}^\mathrm{T}\boldsymbol{e}_k\},\ \boldsymbol{B}_k = \boldsymbol{F}\{\boldsymbol{D}_\mathrm{f}^\mathrm{T}\boldsymbol{b}_k\},\ \text{and}\ \boldsymbol{g}_{k+1} = \frac{2}{N}\boldsymbol{D}_\mathrm{f}\boldsymbol{F}^{-1}\{[\boldsymbol{X}_j^\mathrm{H}\boldsymbol{E}_{k+1}]\} \tag{6.36}$$

where $\boldsymbol{b}_k$ is the search direction in the time domain which can be formulated as

$$\boldsymbol{b}_{k+1} = \boldsymbol{g}_{k+1} + \beta_k\boldsymbol{b}_k \qquad \text{and} \qquad \beta_k = \frac{\boldsymbol{g}_{k+1}^\mathrm{T}\boldsymbol{g}_{k+1}}{\boldsymbol{g}_k^\mathrm{T}\boldsymbol{g}_k} \tag{6.37}$$

where the constant β_k is chosen such that $\boldsymbol{b}_{k+1}$ is conjugate to $\boldsymbol{b}_k$ [Cooper and Steinberg, 1970], that is, $\boldsymbol{b}_k^\mathrm{T}\boldsymbol{X}_j^\mathrm{T}\boldsymbol{X}_j\boldsymbol{b}_{k+1} = 0$ by Lim and Un (1996). All the vectors have now been transformed to the frequency domain, with additional zero and identity matrices added in to ensure linear convolution.

Now consider the constrained optimisation problem in equation (6.7). The gradient of new objective function with respect to the filter coefficients, for the constrained block CG algorithm in frequency domain, can be written as

$$\nabla J_k = \frac{2}{N} \boldsymbol{D}_{\mathrm{f}} \left\{ [\boldsymbol{X}_j^{\mathrm{H}} \boldsymbol{E}_{k+1}] + 2\sigma \sum_{k=0}^{N\text{-}1} \left[\boldsymbol{W}_k(m) \boldsymbol{a}_k^* \right]_z \right\} \tag{6.38}$$

Using inverse DFT transforms the gradient (6.38) to the time domain,

$$\nabla J_k = \frac{2}{N} \boldsymbol{D}_{\mathrm{f}} \boldsymbol{F}^{-1} \left([\boldsymbol{X}_j^{\mathrm{H}} \boldsymbol{E}_{k+1}] + 2\sigma N \left[\boldsymbol{W}_k(m) \right]_z \right) \tag{6.39}$$

For the jth data block, the constrained adaptive frequency-domain CG algorithm is then formulated as follows.

For each data block j, initialise $\boldsymbol{w}_0$ and $\boldsymbol{W}_0$, then

Start:

Compute (3.79) - (3.82) as in the case of unconstrained.

Then, set the magnitude limit L:

L can take the form of a scalar or functions.

Formulate the constraints by setting:

$$c_i(\boldsymbol{w}(k)) = |\boldsymbol{W}(k)|^2 - L(k) < 0 \tag{6.40}$$

Consider the first order penalty function,

$$\sum_{i=1}^{I} \max(c_i(\boldsymbol{w}), 0)\,, \tag{6.41}$$

and define,

$$\left[\boldsymbol{W}(k) \right]_z = \boldsymbol{W}(k) \left(\frac{sign(c_i(\boldsymbol{w}_k)) + 1}{2} \right) \tag{6.42}$$

Then, compute the initial search direction,

$$\boldsymbol{b}_0 = -\nabla J_0 = \frac{2}{N}\boldsymbol{D}_\mathrm{f}\boldsymbol{F}^{-1}\left([\boldsymbol{X}_j^\mathrm{H}\boldsymbol{E}_0] + 2\sigma N[\boldsymbol{W}_0]_z\right) \tag{6.43}$$

For $k = 1$ to M-1 do:

$$\boldsymbol{B}_k = \boldsymbol{F}\{\boldsymbol{D}_\mathrm{f}^\mathrm{T}\boldsymbol{b}_k\}; \tag{6.44}$$

Then, compute the update equation for the adaptive filter,

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \mu_k\boldsymbol{b}_k; \tag{6.45}$$

Transform $\boldsymbol{w}$ in frequency domain,

$$\boldsymbol{W}_{k+1} = \boldsymbol{F}\{\boldsymbol{D}_\mathrm{f}^\mathrm{T}\boldsymbol{w}_k\}. \tag{6.46}$$

If k is not equal to M-1, then compute:

$$\boldsymbol{e}_{k+1} = \boldsymbol{e}_k - \mu_k\ \boldsymbol{D}_\mathrm{b}\boldsymbol{F}^{-1}\{[\boldsymbol{X}_j\ \boldsymbol{B}_k]\}; \tag{6.47}$$

$$\boldsymbol{E}_{k+1} = \boldsymbol{F}\{\boldsymbol{D}_\mathrm{b}^\mathrm{T}\boldsymbol{e}_{k+1}\}; \tag{6.48}$$

$$\nabla J_{k+1} = \frac{2}{N}\boldsymbol{D}_\mathrm{f}\boldsymbol{F}^{-1}\left([\boldsymbol{X}_j^\mathrm{H}\boldsymbol{E}_{k+1}] + 2\,\sigma\,N[\boldsymbol{W}_{k+1}]_z\right); \tag{6.49}$$

$$\boldsymbol{b}_{k+1} = \nabla J_{k+1} + \beta_k\boldsymbol{b}_k, \text{ with } \beta_k = \frac{\nabla J_{k+1}^\mathrm{T}\nabla J_{k+1}}{\nabla J_k^\mathrm{T}\nabla J_k}. \tag{6.50}$$

Then replace $\boldsymbol{w}_0$ with $\boldsymbol{w}_{k+1}$. Go to Start for next block.

As was the case in the time domain, a termination criterion can be set on $\nabla J_{k+1}^\mathrm{T}\nabla J_{k+1}$ to stop needless iterations. Finally, the above is substituted into the first iteration of the filter coefficient update equation for the subsequent block, and the process above repeated for blocks 2 to j.

6.5 Conclusions

In this chapter, new constrained adaptive algorithms were developed, which used first order penalty function and incorporate the FDCGA and BN-FDLMS algorithms. This chapter adopts the FDCGA and BN-FDLMS algorithms, so that the improvement over the conventional LMS algorithm can be made when applied to a wider range of practical applications, particularly in acoustics and audio. The performance of the algorithms will be studied and compared in chapter 9, in the application to the sound equalisation system. It is shown that increasing the convergence rate of the FDLMS algorithm can be done by incorporating the BN-FDLMS and FDCGA algorithms.

This chapter presented formulations of constrained adaptive filters using a first order penalty function since this was found most suitable for efficient realization. However, similar formulation can be developed for constrained adaptive filters such as the FDCGA and BN-FDLMS that use second order, with or without linear approximation, in the penalty term.

The penalty method has been used in this thesis to convert constrained optimisation into unconstrained optimisation. The penalty parameter σ and convergence coefficient μ are used to control the stability and convergence speed. A novel theoretical analysis for the values of the convergence coefficient μ and penalty parameter σ that ensure stability will be developed in the next chapter.

Chapter 7

Theoretical analysis of convergence of the constrained FDLMS algorithm

7.1 Introduction

In this chapter, a theoretical analysis of the constrained algorithm for adaptive filtering implemented in this thesis is developed. For this thesis, the penalty method is used to convert constrained optimisation problems into unconstrained optimisations; the constraints of the original problem are incorporated into a new objective function. The penalty method then uses penalty parameter σ to penalise violation of the constraints. A penalty function formulation is used with a steepest descent search to adapt the filter so that it converges to the new constrained minimum.

Recently developed algorithms, e.g. frequency domain LMS algorithm with constraints, solved with penalty method [Rafaely and Elliott, 2000b], used trial and error to find suitable values of the penalty parameter σ and the convergence coefficient μ since no definitive convergence theory exists. This approach may result in inaccurate setting for μ and σ and therefore slow convergence. In this chapter a bound will be constructed for the value of the penalty parameters, which will give conditions for convergence and hence enable more accurate and faster converging adaptive algorithms using the penalty method. Performance and stability could be significantly improved if a bound or theory concerning these parameters could be established, as available for the constrained LMS and FDLMS, for example [Rafaely and Elliott, 2000b]. In this work the theoretical bound on the parameters of the adaptive LMS and FDLMS algorithms with quadratic constraints, applying first order penalty function max(c_i, 0), is developed. The analysis is performed for the first order penalty function since it results in a quadratic cost function. The cost function for the second order penalty function is not quadratic and so developing a theoretical analysis of convergence is not simple, and left for further work.

The following sets out the theoretical work in establishing bounds for the penalty parameter σ and the convergence coefficient μ for the constrained adaptive filter with quadratic constraints, and with first order penalty function max(c_i, 0). The parameter μ is a convergence factor that regulates the speed and stability of adaptation, and the parameter σ controls the tightness of the penalty function. The development of the new bounds is formulated for the block LMS algorithm (BLMS) and the block LMS algorithm in frequency domain (FDLMS). The theoretical analysis is presented in sections 7.2, 7.3, 7.4 and the simulation study in chapter 9 section 9.5.

Note: The use of frequency domain adaptive filter has a substantial computation reduction compared with time domain adaptive filtering (refer to chapter 3 for detail), and also can incorporate the frequency domain constraint. Therefore FDLMS is considered as the algorithm of our theoretical analysis chapter.

7.2 Bound on the convergence coefficient for quadratic constraints

To introduce the theoretical analysis of convergence of the BLMS and the FDLMS algorithms for the constrained case, we first review the bounds on the convergence coefficient μ for the unconstrained case. Details can be referred to in chapter 3.

Consider the variance of the error signal as the cost function:

$$J = E\,[e^{\mathrm{T}}(n)\, e(n)] \tag{7.1}$$

where $e(n) = d(n) - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}(n)$ is the new error signal.

For BLMS and FDLMS algorithms, the stability bound for μ is inversely proportional to the block size. This is because adaptation is performed only once per block, the FIR weights is to wait until a block of samples has arrived, and gradient estimates are summed over an entire block of data before being used to update the weights. The weights are updated every L samples rather than every sample instant. The nature of block processing requires that the weight filter coefficients be held fixed during the block [Shynk, 1992]. The adaptive weight-update equation of BLMS is defined as (see section 3.4):

$$\boldsymbol{w}(n+L) = \boldsymbol{w}(n) + 2\mu \sum_{m=0}^{L-1} e(n+m)\boldsymbol{x}(n+m) \tag{7.2}$$

The update equation of BLMS in frequency domain (FDLMS) is implemented as

$$\boldsymbol{W}(k+1) = \boldsymbol{W}(k) + 2\mu\, \boldsymbol{FgF}^{-1}\, \boldsymbol{X}^{\mathrm{H}}(k)\, \boldsymbol{E}(k) \tag{7.3}$$

where $\boldsymbol{g}$ is the causality constrained matrix.

Since the BLMS adaptive filter is essentially identical to the frequency domain BLMS algorithm [Cowan and Grant, 1985], it is sufficient to study the convergence coefficient of the BLMS. Using equations (7.2), a recursion for the expected value of the weight vector can be obtained under the assumption that $d(n)$ and $x(n)$ are stationary and that the $x(n)$ are uncorrelated in time or $\boldsymbol{w}$ changes very slowly [Cowan and Grant, 1985 and Widrow and Stearns, 1985]:

$$\begin{aligned} E[\boldsymbol{w}(k+1)] &= E[\boldsymbol{w}(k)] + 2\mu N\left(E[d(n)\boldsymbol{x}(n)] - E[\boldsymbol{x}(n)\boldsymbol{x}(n)^{\mathrm{T}}\boldsymbol{w}(k)]\right) \\ &= E[\boldsymbol{w}(k)] + 2\mu N\left(\boldsymbol{p} - \boldsymbol{R}E[\boldsymbol{w}(k)]\right) \end{aligned} \tag{7.4}$$

where $\boldsymbol{R} = E[\boldsymbol{x}(n)\boldsymbol{x}(n)^{\mathrm{T}}]$, $\boldsymbol{p} = E[d(n)\boldsymbol{x}(n)]$ and a block size of N points is assumed here which is equal to filter length (size $2N$ for FDLMS). Then the bound of the convergence coefficient for stability, with BLMS and FDLMS algorithms, can be defined by equation (7.4), which is

$$0 < \mu < \frac{1}{N\lambda_{\max}(\boldsymbol{R})} \tag{7.5}$$

Chapter 3 showed that the bound for the step size μ to ensure convergence of the BLMS or FDLMS algorithm was $0 < \mu < 1/N\lambda_{\max}$, where $\lambda_{\max}$ is the maximum eigenvalues of the autocorrelation matrix $\boldsymbol{R}$, and $\boldsymbol{R}$ is positive definite matrix. In other words, to ensure the stability of the matrix $(\boldsymbol{I} - 2\mu N\Lambda)$ it is necessary that $0 < \mu < 1/N\lambda_{\max}$, where $\Lambda = \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{RQ}$ and $\lambda_{\max}$ is the largest diagonal element in Λ. To avoid computation of the eigenvalues of the autocorrelation matrix $\boldsymbol{R}$, or sometime in practice $\lambda_{\max}(\boldsymbol{R})$ may not be known a priori, we can replace $\lambda_{\max}(\boldsymbol{R})$ with an upper bound [Widrow and Stearns, 1985],

$$\lambda_{\max}(\boldsymbol{R}) \le \mathrm{trace}(\boldsymbol{R}) \tag{7.6}$$

For positive definite matrices, trace of $\boldsymbol{R}$ denoted by tr[$\boldsymbol{R}$], is the sum of the diagonal elements of $\boldsymbol{R}$ (i.e. the sum of diagonal elements of Λ, thus is the sum of all the eigenvalues). Therefore

$\mu < 1/N\mathrm{tr}[\boldsymbol{R}]$ guarantees that $\mu < 1/N\lambda_{\max}$, however it may provide a more conservative boundary for μ. Furthermore, with a transversal adaptive filter, an autocorrelation matrix gives $\mathrm{tr}[\boldsymbol{R}]$ as just $N\mathrm{Var}\{x(n)\}$, or N times the input signal power [Widrow and Stearns, 1985]. So, a more conservative, sufficient condition for convergence, equation (7.5) becomes

$$0 < \mu < \frac{1}{N^2 E\{x(n)^2\}} \tag{7.7}$$

The advantage of using $\mathrm{tr}[\boldsymbol{R}]$ is that it is easier to compute because $\mathrm{Var}\{x(n)\}$ can be estimated from the input data. To see the conservative nature of (7.7), the eigenvalue spread of $\boldsymbol{R}$ of the reference signals x needs to be considered, because the convergence rate is inherently limited by the eigenvalue spread of the data correlation matrix. If x is a white noise, which implies the power spectrum of x is flat i.e. λ_i all the same, so

$$\frac{1}{N\lambda_{\max}} = \frac{1}{N\lambda} >> \frac{1}{N\mathrm{tr}[\boldsymbol{R}]} \tag{7.8}$$

because in this case $\mathrm{tr}[\boldsymbol{R}] = N\lambda$. Hence the limit is conservative. If x is not a white noise, i.e. with tonal components, then the $\lambda_{max} >> \lambda_i$'s. Therefore,

$$\frac{1}{N\lambda_{\max}} \approx \frac{1}{N\mathrm{tr}[\boldsymbol{R}]} \tag{7.9}$$

Hence, the limit is not conservative.

The bounds (7.5) (7.7) of the convergence coefficient μ for the unconstrained case have been introduced. The theoretical convergence coefficient μ for the constrained case is now implemented as follows.

The derivation of the bound on μ for the constrained algorithm would suggest that when penalty functions are used to reformulate the new objective function and the new update equation, that the bounds on μ will involve the penalty parameter σ. It is therefore reasonable to study the eigenvalues of the matrices in the penalty term as a first step in establishing bounds for the constrained optimisation case.

To begin with, a cost function with penalty term $J = E[e^{\mathrm{T}}e] + \sigma\sum_{i=1}^{l}\{\max[c_i(w), 0]\}$ is considered. The penalty method is implemented by using the first order penalty function $\{\max[c_i, 0]\}$, due to the computational complexity of the second order penalty function when quadratic constraint are involved. Details of first and second order penalty functions have been discussed in chapters 2 and 5. The constraint $|W|^2 < L$ will be used; detail can be found in chapter 5. When no constraints are active, the penalty term does not apply. The cost function can then be rewritten as $J = E[e^{\mathrm{T}}e] = w^{\mathrm{T}}A_0 w + \text{lower order terms}$. This sub-problem now resembles unconstrained optimisation and the above bound for the convergence coefficient can therefore be applied. Once one of the constraints becomes active, the cost function becomes $J = E[e^{\mathrm{T}}e] + \sigma c_1$. This first constraint will have a structure of the form $c_1 = w^{\mathrm{T}}A_1 w - L$, providing another Hessian (or Hermitian) matrix, A_1. It can be seen from the above that the step size bound for the unconstrained case is based on the eigenvalue of the Hessian matrix A_0. The bound for the case with one active constraint is therefore a function of the eigenvalues of matrices A_0 and A_1. As iterations of the algorithm proceed and more constraints become active, a sequence of Hessian matrices is formed from the constraints, as described above. As each constraint becomes active, a new maximum bound can be found for the convergence coefficient, based on the eigenvalues of the sequence of Hessian matrices generated from the constraints active at that point. Continuing this argument, assuming the worst-case scenario: all the constraints are active, the maximum eigenvalue from the Hessian matrices of all constraints are found in order to set a bound on the maximum stable convergence coefficient. The sequence generated is as follows:

$$\begin{aligned}
&\lambda_{\max}(A_0) \to \mu_{\max} \\
&\lambda_{\max}(A_0 + \sigma A_1) \to \mu_{1\max} \\
&\lambda_{\max}(A_0 + \sigma A_1 + \sigma A_2) \to \mu_{2\max} \\
&\qquad\vdots \\
&\lambda_{\max}(A_0 + \sum_i \sigma A_i) \to \mu_{i\ \max}
\end{aligned} \tag{7.10}$$

It is now required to determine the maximum eigenvalues of the matrix A, where $A = A_0 + \sum_i \sigma A_i$ for all i. Instead, the maximum eigenvalues of the individual matrices A_i will be calculated. There is a relation between the eigenvalues of A and the eigenvalues of the matrices A_i, which can be shown to be [Golub and Van Loan, 1983]:

$$\lambda_{\max}(A_0 + \sigma A_1 + \sigma A_2 + \cdots) \le \lambda_{\max}(A_0) + \sigma\,\lambda_{\max}(A_1) + \sigma\,\lambda_{\max}(A_2) + \cdots \tag{7.11}$$

Thus, if a bound can be established using the eigenvalues of the individual matrices (RHS), it will also serve as a bound on the matrix A, by the above inequality. Taking the maximum eigenvalue at each stage means that any bound established would lead to a stable convergence coefficient.

Let's now look at the constrained problem in equation (6.7), which is written again here:

$$\begin{aligned} &\text{Minimise} && J = E\,[e^{\mathrm{T}}(n)\,e(n)] \\ &\text{Subject to} && c_k = |W(k)|^2 - L(k) < 0 \qquad k = 0, 1, \ldots, N\text{-}1 \end{aligned} \tag{7.12}$$

The discrete Fourier transform operation on w is written as [Rafaely and Elliott, 2000b and Widrow and Stearns, 1985]:

$$W(k) = \sum_{n=0}^{2N-1} w_n e^{-j2\pi nk/2N} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{a}_k \tag{7.13}$$

The constraint c_k at frequency k in the time domain can be written as follows [Rafaely and Elliott, 2000b],

$$c_k = |W(k)|^2 - L(k) = \boldsymbol{w}^{\mathrm{T}}(\boldsymbol{a}_k^{*}\boldsymbol{a}_k^{\mathrm{T}})\boldsymbol{w} - L(k) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{A}_k\boldsymbol{w} - L(k) \tag{7.14}$$

where $\boldsymbol{a}_k^{*}$ is the column vector with components equal to the complex conjugates of those of the column vector $\boldsymbol{a}_k$. Equation (7.14) shows the constraint c_k is a quadratic function and the constraints limiting the magnitude of $\boldsymbol{W}$ are all convex, since matrix A_k is positive definite Hermitian matrix, due to its construction $A_k = \boldsymbol{a}_k^{*}\,\boldsymbol{a}_k^{\mathrm{T}}$ (Refer to chapter 5 for further detail).

The constrained problem in (7.12) is now reformulated using the first order penalty function. Also as we mentioned above, we assume the worst-case scenario with all the constraints active and so all the constraints are consider here. For convenience of notation, the constraint index i is used; A_i therefore represents the constraint on the gain of the adaptive filter at frequency k. The new objective function with penalty term can be written as

$$\text{Minimise} \qquad J_{new} = E[\boldsymbol{e}^{\mathrm{T}}\boldsymbol{e}] + \sigma\sum_{i=1}^{N}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{A}_i\boldsymbol{w} - L) \tag{7.15}$$

Expanding equation (7.15), then we get

$$\boldsymbol{J}_{new} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{w} - 2\boldsymbol{w}^{\mathrm{T}}\boldsymbol{p} + E[d^2] + \sigma\sum_{i=1}^{N}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{A}_i\boldsymbol{w} - L) \tag{7.16}$$

The gradient of the new objective function $\boldsymbol{J}_{new}$ with respect to the parameter vector $\boldsymbol{w}$ is defined by:

$$\nabla\boldsymbol{J}_{new} = \frac{\partial E[\boldsymbol{e}^{\mathrm{T}}\boldsymbol{e}]}{\partial \boldsymbol{w}} + \sigma\sum_{i=1}^{N}\frac{\partial}{\partial \boldsymbol{w}}\left[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{A}_i\boldsymbol{w} - L\right] \tag{7.17}$$

As we know that the BLMS (7.2) is the block processing algorithm. The gradient estimates are summed over an entire block of data before being used to update the weights. i.e. N gradient are added, and the weights are only updated every N samples. The update equation without constraints for BLMS was defined as

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + 2\mu\sum_{m=0}^{L-1}\boldsymbol{ex}, \qquad \text{where } L = N \tag{7.18}$$

However the BLMS algorithm of equation (7.18) can be compared to the LMS algorithm (without constraints) by multiplying $\boldsymbol{x}$ by N, which is given as

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + 2\mu\boldsymbol{ex}\cdot N \tag{7.19}$$

Therefore from equations (7.18) and (7.19), we use here $\boldsymbol{R}\cdot N$ as the equivalent Hessian matrix. So $\boldsymbol{R}$ in equation (7.16) should consider the fact that the gradient estimate from $\boldsymbol{R}$ is summed over an entire block of data before being used to update the weights. Hence the objective function of equation (7.16) should be written as the below equation (7.20) for the constrained BLMS:

$$\boldsymbol{J}_{new} = \boldsymbol{w}^{\mathrm{T}}(\boldsymbol{R}\cdot N)\boldsymbol{w} - 2\boldsymbol{w}^{\mathrm{T}}(\boldsymbol{p}\cdot N) + E[d^2] + \sigma\sum_{i=1}^{N}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{A}_i\boldsymbol{w} - L) \tag{7.20}$$

$$\Rightarrow \qquad = \boldsymbol{w}^{\mathrm{T}}\left(\boldsymbol{R}\cdot N + \sigma\sum_{i=1}^{N}\boldsymbol{A}_i\right)\boldsymbol{w} - 2\boldsymbol{w}^{\mathrm{T}}(\boldsymbol{p}\cdot N) + E[d^2] - \sigma L \tag{7.21}$$

Note that the gradient of penalty term can be shown that is not summed over an entire block of data before being used to update the weights, which implies the constraints are independent and individual to each other at difference frequency k. Therefore the gradient of penalty term is not proportional to the block size N. The constraints will be shown in the next section to be independent at frequency k.

Equation (7.21) is a quadratic equation of the form

$$\boldsymbol{J}_{new} = \boldsymbol{w}^{\mathrm{T}}\hat{\boldsymbol{R}}\boldsymbol{w} - 2\boldsymbol{w}^{\mathrm{T}}\boldsymbol{p}N + constant \tag{7.22}$$

where

$$\hat{\boldsymbol{R}} = N\boldsymbol{R} + \sigma\sum_{i=1}^{N}\boldsymbol{A}_i \tag{7.23}$$

Since $\boldsymbol{R}$ and $\boldsymbol{A}_i$ are positive definite Hermitian matrices, therefore it implies $\hat{\boldsymbol{R}}$ is a positive definite Hermitian matrix. The gradient at any point on the new constrained surface is obtained by differentiating the new objective function $\boldsymbol{J}_{new}$ with respect to the parameter vector $\boldsymbol{w}$,

$$\nabla\boldsymbol{J}_{new} = 2\hat{\boldsymbol{R}}\boldsymbol{w} - 2\boldsymbol{p}N \tag{7.24}$$

Since J_{new} is a quadratic function, so by setting the new gradient vector ∇J_{new} to zero we can define the optimal solution. For stationary input processes, the expected value of the new weight vector $E[\boldsymbol{w}_{new}]$ after a sufficient number of iterations will converge to the Wiener optimal solution [Cowan and Grant, 1985 and Widrow and Stearns, 1985], that is:

$$\boldsymbol{w}^*_{new} = \hat{\boldsymbol{R}}^{-1}(\boldsymbol{p}N) \tag{7.25}$$

Using the assumption that the input vector and the weight vector are independent, and the gradient of the objective function in (7.22), the weight-update equation of the BLMS for constrained problem (7.12) can be simplified and written as follows,

$$E[\boldsymbol{w}(k+1)] = E[\boldsymbol{w}(k)] + 2\mu\left(N\boldsymbol{p} - \hat{\boldsymbol{R}}E[\boldsymbol{w}(k)]\right) \tag{7.26}$$

where the usual assumption of uncorrelated between $\boldsymbol{x}$ and $\boldsymbol{w}$ is still held. Now using equation (7.25), then equation (7.26) can be written as

$$E[\boldsymbol{w}_{k+1}] = \left(I - 2\mu\hat{\boldsymbol{R}}\right)E[\boldsymbol{w}_k] + 2\mu\hat{\boldsymbol{R}}\boldsymbol{w}^* \tag{7.27}$$

Since matrix $\hat{\boldsymbol{R}}$ is Hermitian matrix, hence $\hat{\boldsymbol{R}}$ can be decomposed as $\Lambda = \boldsymbol{Q}^{\mathrm{T}} \hat{\boldsymbol{R}} \boldsymbol{Q}$, where $\boldsymbol{Q}$ is the orthogonal matrix composed of eigenvectors of $\hat{\boldsymbol{R}}$, and Λ is the diagonal matrix of eigenvalues. The following theorem states the property of Hermitian matrices. This theorem is known as spectral theorem, and the set of eigenvalues of a Hermitian matrix is known as its spectrum.

Theorem 7.1: Every Hermitian $m \times m$ matrix $\boldsymbol{R}$ can be diagonalised by a unitary matrix:

$$\Lambda = \boldsymbol{Q}^{\mathrm{T}} \boldsymbol{R} \boldsymbol{Q}, \tag{7.28}$$

where $\boldsymbol{Q}$ is unitary and Λ is diagonal [Moon and Stirling, 2000].

Note that a matrix which satisfies $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q} = \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}} = \boldsymbol{I}$ is called a unitary matrix. A real unitary matrix is called an orthogonal matrix. This theorem is true when $\boldsymbol{R}$ has distinct eigenvalues; also it is true even when $\boldsymbol{R}$ has repeated eigenvalues.

By theorem 7.1, equation (7.27) can be decomposed into the form of equation (7.28) to obtain the convergence bound on μ. The sufficient condition for μ of equation (7.27) can be calculated in the same manner as unconstrained problem. i.e. To ensure the stability of the matrix $(\boldsymbol{I} - 2\mu\Lambda)$ it is necessary that,

$$0 < \mu < \frac{1}{\lambda_{\max}(\hat{\boldsymbol{R}})} \tag{7.29}$$

where $\Lambda = \boldsymbol{Q}^{\mathrm{T}} \hat{\boldsymbol{R}} \boldsymbol{Q}$, and $\lambda_{\max}$ is the maximum eigenvalues of matrix $\hat{\boldsymbol{R}}$ (i.e. $\lambda_{\max}$ is the largest diagonal element in Λ).

Condition (7.29) is necessary and sufficient for convergence of the constrained BLMS or FDLMS algorithms with a quadratic performance surface. It can be obtained under the condition that $\boldsymbol{J}_{new}$ is a quadratic function.

Using equation (7.23), then equation (7.29) can be written as,

$$0 < \mu < \frac{1}{\lambda_{\max}\left(N\boldsymbol{R} + \sigma\sum_{i=1}^{N} \boldsymbol{A}_i\right)} \tag{7.30}$$

Using equation (7.11), then a sufficient coefficient for convergence is

$$0 < \mu < \frac{1}{\lambda_{\max}\left(N\boldsymbol{R} + \sigma\sum_{i=1}^{N}\boldsymbol{A}_i\right)} \leq \frac{1}{N\left[\lambda_{\max}(\boldsymbol{R})\right] + \sigma\lambda_{\max}\left(\sum_{i=1}^{N}\boldsymbol{A}_i\right)} \tag{7.31}$$

If $\lambda_{\max}(\boldsymbol{R})$ is not known a priori, equation (7.31) can be written as follows by using equation (7.6) and (7.11),

$$\mu < \frac{1}{N\left[tr(\boldsymbol{R})\right] + \sigma\sum_{i=1}^{N}\lambda_{\max}(\boldsymbol{A}_i)} \tag{7.32}$$

$$\Rightarrow \qquad \mu < \frac{1}{N\left[NE\{x(n)^2\}\right] + \sigma\sum_{i=1}^{N}\lambda_{\max}(\boldsymbol{A}_i)} \tag{7.33}$$

The theoretical analysis of the convergence coefficient bound on μ, for constrained problem, is defined in (7.31) or (7.33); it determines rate of convergence and stability of the adaptive process. Equation (7.31) and (7.33) are the general forms of the bound on μ that calculated by using the penalty method with BLMS algorithm in constrained case. Hence, the weight-update equation of FDLMS eventually minimises the mean square error at the *k*th frequency bin, provided that μ is chosen an accurate number that is determined from the bound condition (7.31) or (7.33).

The analysis of $\lambda_{\max}(\boldsymbol{A}_i)$ is now performed in the following section, and shows that the constraint c_k is independent and individual at any frequency k. Then the final form of the theoretical convergence coefficient will be defined.

7.3 Analysis of the Fourier matrix in the magnitude constraints

The frequency domain block LMS algorithm considered in this thesis performs strictly linear convolution. It allows an efficient frequency domain implementation while maintaining performance equivalent to that of the widely used LMS adaptive filter. In chapter 3, the BLMS algorithm has been implemented in the frequency domain, FDLMS, using the overlap-save method [Shynk, 1992]. This is because the frequency domain filter produces circular convolution of the input signal with the adaptive filter impulse response, i.e. the input blocks are circularly convolved with the impulse response. Due to the overlap redundancy at the input, the first N points of the circular convolution in the output can simply be discarded. Then the overlap-save method FFT filter block outputs remains the last N points, which is equivalent to the linear convolution. According to this method, the weights must be padded with N zeros, so that $2N$ point FFTs must be used.

The constraint c_k considered is a limit on the magnitude of the adaptive filter $\boldsymbol{W}$ at frequency k, $|\boldsymbol{W}(k)|^2 < L(k)$, so that the filter gain is constrained to L. The frequency domain vector $\boldsymbol{W}(k)$ of size $2N$ is calculated using FFT's on the corresponding time domain vector as equation (7.13). i.e. Matrix $\boldsymbol{A}_i$ (i.e. $\boldsymbol{a}_k^* \, \boldsymbol{a}_k^{\mathrm{T}}$), is the size of $2N \times 2N$ matrix formed from constraint c_k. Since matrix $\boldsymbol{A}_i$ is formed from complex Fourier exponential vectors $\boldsymbol{a}_k^* \, \boldsymbol{a}_k^{\mathrm{T}}$, then $\lambda_{\max}(\boldsymbol{A}_i)$ can be calculated, for $i = 1, \ldots, N$. The structure of the Fourier transform matrix $\boldsymbol{A}_i$ is now considered, and the constraints are then shown to be independent and individual at different frequency k, where $k = 0, 1, 2, \ldots, N\text{-}1$. The statement that we want to prove is: "no matter how many constraints are actually active, the maximum eigenvalue is still the same". More specifically, the statement can be stated as follows:

- In the general case,

$$\lambda_{\max}\left(\sum_{i=1}^{N} \boldsymbol{A}_i\right) \le \sum_{i=1}^{N} \lambda_{\max}(\boldsymbol{A}_i) \tag{7.34}$$

- But for the constraint $|\boldsymbol{W}(k)|^2 < L$,

$$\lambda_{\max}\left(\sum_{i=1}^{N} \boldsymbol{A}_i\right) = \lambda_{\max}(\boldsymbol{A}_i) = 2N \tag{7.35}$$

i.e. the maximum eigenvalue when one constraint is active

= the maximum eigenvalue when all constraints are active.

Statement (7.34) means that all constraints will be summed over an entire block before being used to update the weights, in BLMS or FDLMS algorithms. Therefore, $\boldsymbol{A}_i$ is dependent on $\boldsymbol{A}_j$, where $i \neq j$, and hence they are not independent and individual.

Statement (7.35) means that constraints are not summed over an entire block for the update processing. In fact, it means that $\boldsymbol{A}_i$ is independent on $\boldsymbol{A}_j$, and hence constraints happen individually.

In order to show (7.35), lets now consider the structure of the Fourier transform matrix $\boldsymbol{A}_i$. According the equation (7.13), $\boldsymbol{a}_k$ is a column vector of size $2N$ of the complex Fourier exponential at frequency k,

$$\boldsymbol{a}_k = \left(e^{-j2\pi nk/2N}\right) = \begin{pmatrix} 1 \\ e^{-j2\pi k/2N} \\ e^{-j4\pi k/2N} \\ \vdots \\ e^{-j2\pi(2N-2)k/2N} \\ e^{-j2\pi(2N-1)k/2N} \end{pmatrix}, \quad \text{where } n = 0, 1, 2, \ldots, 2N\text{-}1. \tag{7.36}$$

The structure of Fourier transform matrix $\boldsymbol{A}_i$ is then,

$$\boldsymbol{A}_i = \overset{*}{\boldsymbol{a}_k}\boldsymbol{a}_k^{\mathrm{T}} = \begin{pmatrix} 1 \\ e^{j2\pi k/2N} \\ e^{j4\pi k/2N} \\ e^{j6\pi k/2N} \\ \vdots \\ \vdots \\ e^{j2\pi(2N-1)k/2N} \end{pmatrix} \begin{pmatrix} 1 & e^{-j2\pi k/2N} & e^{-j4\pi k/2N} & e^{-j6\pi k/2N} & \cdots & \cdots & e^{-j2\pi(2N-1)k/2N} \end{pmatrix} \tag{7.37}$$

Therefore the structure of the Fourier transform matrix $\boldsymbol{A}_i$ at frequency k is

$$= \begin{pmatrix} 1 & e^{-j2\pi k/2N} & e^{-j4\pi k/2N} & e^{-j6\pi k/2N} & \cdots & e^{-j2\pi(2N-1)k/2N} \\ e^{j2\pi k/2N} & 1 & e^{-j2\pi k/2N} & e^{-j4\pi k/2N} & \cdots & e^{-j2\pi(2N-2)k/2N} \\ e^{j4\pi k/2N} & e^{j2\pi k/2N} & 1 & e^{-j2\pi k/2N} & \cdots & e^{-j2\pi(2N-3)k/2N} \\ e^{j6\pi k/2N} & e^{j4\pi k/2N} & e^{j2\pi k/2N} & 1 & & e^{-j2\pi(2N-4)k/2N} \\ \vdots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ e^{j2\pi(2N-3)k/2N} & \ddots & \ddots & \ddots & \ddots & e^{-j4\pi k/2N} \\ e^{j2\pi(2N-2)k/2N} & e^{j2\pi(2N-3)k/2N} & \ddots & \ddots & \ddots & e^{-j2\pi k/2N} \\ e^{j2\pi(2N-1)k/2N} & e^{j2\pi(2N-2)k/2N} & e^{j2\pi(2N-3)k/2N} & e^{j2\pi(2N-4)k/2N} & \cdots & 1 \end{pmatrix} \tag{7.38}$$

In (7.38), the matrix elements along each diagonal have the same entry, i.e. matrix A_i is positive definite Hermitian matrix; it is also a Toeplitz matrix.

The matrices A are constraining the explicit dependence on k, which means

$$\begin{aligned}
&A_1 \text{ is constraining on the gain of the adaptive filter at frequency } k = 0;\\
&A_2 \text{ is constraining on the gain of the adaptive filter at frequency } k = 1;\\
&\vdots\\
&\vdots\\
&A_N \text{ is constraining on the gain of the adaptive filter at frequency } k = N\text{-}1;
\end{aligned} \tag{7.39}$$

Therefore there is a constraint on the magnitude at each frequency in the adaptive process. Note: we were assuming the worst-case scenario; all the constraints are active, so all A_i will be considered, where $i = 1, 2, \ldots, N$.

The eigenvalue of A_i is now considered. For a general square matrix A one seeks a scalar λ and a vector x such that $Ax = \lambda x$. The vector x is called an eigenvector of A associated with eigenvalue λ. The matrix equation of eigenvalue problems of A_i can be written in the form:

$$\begin{pmatrix} 1 & e^{-j2\pi k/2N} & \cdots & e^{-j2\pi(2N-1)k/2N} \\ e^{j2\pi k/2N} & 1 & \cdots & e^{-j2\pi(2N-2)k/2N} \\ \vdots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ e^{j2\pi(2N-1)k/2N} & e^{j2\pi(2N-2)k/2N} & \cdots & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_{2N} \end{pmatrix} = \begin{pmatrix} \lambda & 0 & \cdots & 0 \\ 0 & \lambda & \cdots & 0 \\ \vdots & & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_{2N} \end{pmatrix} \tag{7.40}$$

where A_i is $2N \times 2N$ matrix and $k = 0, 1, 2, \ldots N\text{-}1$.

The matrix equation becomes,

$$\Rightarrow \begin{pmatrix} 1-\lambda & e^{-j2\pi k/2N} & \cdots & e^{-j2\pi(2N-1)k/2N} \\ e^{j2\pi k/2N} & 1-\lambda & \cdots & e^{-j2\pi(2N-2)k/2N} \\ \vdots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ e^{j2\pi(2N-1)k/2N} & e^{j2\pi(2N-2)k/2N} & \cdots & 1-\lambda \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_{2N} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{pmatrix} \tag{7.41}$$

The short form of the matrices (7.40) and (7.41) can be written $A_i x = \lambda x$ and $(A_i - \lambda I)x = \mathbf{0}$, where i = 1, 2, ..., N. The eigenvalues $\lambda_i(A_i)$ is equivalent to the roots of the determinant of the matrix (A_i - λI), which is written as det (A_i - λI) = 0.

The determinant of a matrix may be defined in several ways. For simplicity, the following definition is used here. The determinant of a $2N \times 2N$ matrix is given by

$$\det(A) = \sum_{j=1}^{2N} (-1)^{j+1} a_{1j} \det(A_{1j}) \tag{7.42}$$

where A_{1j} is an ($2N$-1) × ($2N$-1) matrix obtained by deleting the first row and the j-th column of A [Axler, 1996].

Let's now find the eigenvalues of A_1, where A_1 is the $2N \times 2N$ constraint matrix at frequency k = 0 (refer to 7.39), so from equation (7.41) we have:

$$\det(A_1 - \lambda I) = \begin{vmatrix} 1-\lambda & 1 & \cdots & 1 \\ 1 & 1-\lambda & \cdots & 1 \\ \vdots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 1 & 1 & \cdots & 1-\lambda \end{vmatrix} = 0 \tag{7.43}$$

By using equation (7.42), the solutions of the characteristic equation (7.43) (or the eigenvalues) are

$$\lambda_1 = \lambda_2 = \ldots = \lambda_{2N-1} = 0, \text{ and } \lambda_{2N} = 2N$$

Hence, the maximum eigenvalue $\lambda_{\max}(A_1)$ is $2N$. (see Appendix 1 for detail calculation)

Let's define the eigenvalues of A_2, which is the constraint matrix at frequency k = 1 (refer to 7.39):

$$\det(A_2 - \lambda I) = \begin{vmatrix} 1-\lambda & e^{-j2\pi/2N} & \cdots & e^{-j2\pi(2N-1)/2N} \\ e^{j2\pi/2N} & 1-\lambda & \cdots & e^{-j2\pi(2N-2)/2N} \\ \vdots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ e^{j2\pi(2N-1)/2N} & e^{j2\pi(2N-2)/2N} & \cdots & 1-\lambda \end{vmatrix} = 0 \tag{7.44}$$

During the process of finding the eigenvalues λ in equation (7.44) by using equation (7.42), the complex exponential elements cancel each other out (see Appendix 1), then equation (7.44) becomes

$$\Rightarrow \qquad \lambda^{2N} - 2N\lambda^{2N-1} = 0$$

$$\Rightarrow \qquad \lambda^{2N-1}(\lambda - 2N) = 0 \qquad (7.45)$$

Hence the eigenvalues of the Fourier transform matrix A_2 are

$$\lambda_1 = \lambda_2 = \ldots = \lambda_{2N\text{-}1} = 0, \text{ and } \lambda_{2N} = 2N$$

Therefore the maximum eigenvalue $\lambda_{\max}(A_2)$ is equal to $2N$.

Now, let's compute the eigenvalues of A_3, which is the constraint matrix at frequency $k = 2$:

$$\det(A_3 - \lambda I) = \begin{vmatrix} 1-\lambda & e^{-j4\pi/2N} & \cdots & e^{-j4\pi(2N-1)/2N} \\ e^{j4\pi/2N} & 1-\lambda & \cdots & e^{-j4\pi(2N-2)/2N} \\ \vdots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ e^{j4\pi(2N-1)/2N} & e^{j4\pi(2N-2)/2N} & \cdots & 1-\lambda \end{vmatrix} = 0 \qquad (7.46)$$

The matrix of A_3 has the same feature of A_2, it only uses $k = 2$ for A_3. Hence the eigenvalues of A_3 are the same as the eigenvalues of A_2. Hence the eigenvalues of A_3 are

$$\lambda_1 = \lambda_2 = \ldots = \lambda_{2N\text{-}1} = 0, \text{ and } \lambda_{2N} = 2N$$

Again, the maximum eigenvalue $\lambda_{\max}(A_2)$ is $2N$.

Since the Fourier transform matrix is generated from the complex Fourier exponentials, the matrices $A_4, A_5, A_6, \ldots, A_N$, at frequency $k = 3, 4, 5, \ldots, N\text{-}1$, will have the same feature as A_1, A_2 or A_3. Therefore the eigenvalues of the matrices $A_4, A_5, A_6, \ldots, A_N$ at frequency $k = 3, 4, 5, \ldots, N\text{-}1$, are

$$\lambda(A_1) = \lambda(A_2) = \lambda(A_3) = \cdots = \lambda(A_N) = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 2N \end{pmatrix} \tag{7.47}$$

Hence,

$$\lambda_{\max}(A_1) + \lambda_{\max}(A_2) + \cdots + \lambda_{\max}(A_N) = \sum_{i=1}^{N} \lambda_{\max}(A_1) = 2N + 2N + \cdots + 2N = 2N \cdot N \tag{7.48}$$

Thus, the sum of the maximum eigenvalues of A_i is

$$\sum_{i=1}^{N} \lambda_{\max}(A_i) = 2N^2 \tag{7.49}$$

The eigenvalues of $\lambda(A_i)$ and $\sum_{i=1}^{N} \lambda_{\max}(A_i)$ are just defined. Let's now define the eigenvalues of $\lambda\,(A_1 + A_2 + A_3 + \ldots + A_N)$, and then compare it with (7.47) to (7.49).

Consider the complex Fourier exponential, the column vectors $a_k = e^{-j2\pi nk/2N}$ as before, n = 0, 1, 2, ..., $2N$-1 at the frequency k = 0, 1, 2, ..., N-1. The vectors are stated in the table 7.1. Since matrix $A_i = a_k^* a_k^{\mathrm{T}}$, it implies the following:

Sum of the first row of the matrix $(A_1+A_2+A_3+\ldots+A_N)$

$$= \text{Sum of the row vectors } a_0^{\mathrm{T}} + a_1^{\mathrm{T}} + a_2^{\mathrm{T}} + \cdots + a_{N-1}^{\mathrm{T}} \tag{7.50}$$

and

Sum of the first column of the matrix $(A_1+A_2+A_3+\ldots+A_N)$

$$= \text{Sum of the column vectors } a_0^* + a_1^* + a_2^* + \cdots + a_{N-1}^*. \tag{7.51}$$

From Table 7.1, it shows that the sum of the matrices at first sample at all frequencies, where n = 0 at k = 0, 1, ..., N-1, which is equal to N. Also from table 7.1, we can see some special pattern when n = even, which are:

- When $n = 2q$, where $q = 1, 3, 5, 7, \ldots, N - 1$

$$e^{-jnk(\frac{\pi}{N})} = -e^{-jn(\frac{N}{2}+k)(\frac{\pi}{N})}, \qquad \text{where } k = 0, 1, 2, \ldots, \frac{N}{2} - 1. \tag{7.52}$$

- When $n = 2q$, where $q = 2, 4, 6, 8, \ldots, N - 2$

$$e^{-jnk(\frac{\pi}{N})} = -e^{-jn(\frac{N}{4}+k)(\frac{\pi}{N})}, \qquad \text{where } k = 0, 1, 2, \ldots, \frac{N}{2} - 1. \tag{7.53}$$

Hence it shows that the sum of all frequencies is equal to zero if n = even.

	Table 7.1: Column vectors $\boldsymbol{a}_k$ at the frequency k					
Samples n	$k = 0$	$k = 1$	$k = 2$	$k = 3$	…	$k = N - 1$
$n = 0$	1	1	1	1		1
$n = 1$	1	$e^{-j(\pi/N)}$	$e^{-j2(\pi/N)}$	$e^{-j3(\pi/N)}$		$e^{-j(N-1)(\pi/N)}$
$n = 2$	1	$e^{-j2(\pi/N)}$	$e^{-j4(\pi/N)}$	$e^{-j6(\pi/N)}$		$e^{-j2(N-1)(\pi/N)}$
$n = 3$	1	$e^{-j3(\pi/N)}$	$e^{-j6(\pi/N)}$	$e^{-j9(\pi/N)}$	……	$e^{-j3(N-1)(\pi/N)}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$
$n = 2N - 2$	1	$e^{-j2(N-1)(\pi/N)}$	$e^{-j4(N-1)(\pi/N)}$	$e^{-j6(N-1)(\pi/N)}$		$e^{-j(2N-2)(N-1)(\pi/N)}$
$n = 2N - 1$	1	$e^{-j(2N-1)(\pi/N)}$	$e^{-j2(2N-1)(\pi/N)}$	$e^{-j3(2N-1)(\pi/N)}$		$e^{-j(2N-1)(N-1)(\pi/N)}$

Therefore from the equation (7.50) to (7.53) and Table 7.1, the first row of the matrix $\sum_{i=1}^{N} \boldsymbol{A}_i$ is:

$$= \left(N \quad \sum_{k=0}^{N-1} e^{-jk(\pi/N)} \quad 0 \quad \sum_{k=0}^{N-1} e^{-j3k(\pi/N)} \quad 0 \quad \cdots \quad \sum_{k=0}^{N-1} e^{-j(2N-3)k(\pi/N)} \quad 0 \quad \sum_{k=0}^{N-1} e^{-j(2N-1)k(\pi/N)} \right) \tag{7.54}$$

and the first column of the matrix $\sum_{i=1}^{N} \boldsymbol{A}_i$ is,

$$\boldsymbol{a}_0^* + \boldsymbol{a}_1^* + \boldsymbol{a}_2^* + \cdots + \boldsymbol{a}_{N-1}^* = \begin{pmatrix} N \\ \sum_{k=0}^{N-1} e^{jk(\pi/N)} \\ 0 \\ \sum_{k=0}^{N-1} e^{j3k(\pi/N)} \\ 0 \\ \vdots \\ \sum_{k=0}^{N-1} e^{j(2N-3)k(\pi/N)} \\ 0 \\ \sum_{k=0}^{N-1} e^{j(2N-1)k(\pi/N)} \end{pmatrix} \tag{7.55}$$

From equation (7.50) to (7.55) and the fact that A is Hermitian matrix, the matrix $\sum_{i=1}^{N} \boldsymbol{A}_i$ is conducted in the following special form,

$$= \begin{pmatrix} N & \sum_{k=0}^{N-1} e^{-jk(\pi/N)} & 0 & \sum_{k=0}^{N-1} e^{-j3k(\pi/N)} & \cdots\cdots & \sum_{k=0}^{N-1} e^{-j(2N-1)k(\pi/N)} \\ \sum_{k=0}^{N-1} e^{jk(\pi/N)} & N & \sum_{k=0}^{N-1} e^{-jk(\pi/N)} & 0 & \cdots\cdots & 0 \\ 0 & \sum_{k=0}^{N-1} e^{jk(\pi/N)} & N & \sum_{k=0}^{N-1} e^{-jk(\pi/N)} & \cdots\cdots & \sum_{k=0}^{N-1} e^{-j(2N-3)k(\pi/N)} \\ \sum_{k=0}^{N-1} e^{j3k(\pi/N)} & 0 & \sum_{k=0}^{N-1} e^{jk(\pi/N)} & N & & 0 \\ 0 & \sum_{k=0}^{N-1} e^{j3k(\pi/N)} & 0 & \sum_{k=0}^{N-1} e^{jk(\pi/N)} & \ddots & \sum_{k=0}^{N-1} e^{-j(2N-5)k(\pi/N)} \\ \vdots & & & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \sum_{k=0}^{N-1} e^{j(2N-3)k(\pi/N)} & 0 & \sum_{k=0}^{N-1} e^{j(2N-5)k(\pi/N)} & & \sum_{k=0}^{N-1} e^{-jk(\pi/N)} \\ \sum_{k=0}^{N-1} e^{j(2N-1)k(\pi/N)} & 0 & \sum_{k=0}^{N-1} e^{j(2N-3)k(\pi/N)} & 0 & \cdots\cdots & N \end{pmatrix} \tag{7.56}$$

The matrix equation (7.56) contains zero in columns and rows, which can be negated when we calculate the determinant or the eigenvalues of the matrix.

The determinant of (7.56) is written as,

$$\det\left(\left(\sum_{i=1}^{N} \boldsymbol{A}_i\right) - \lambda \boldsymbol{I}\right) = \left|\left(\sum_{i=1}^{N} \boldsymbol{A}_i\right) - \lambda \boldsymbol{I}\right| = 0 \tag{7.57}$$

Using equation (7.42), then (7.57) becomes: (see Appendix 2 for calculation)

$$\Rightarrow \qquad \lambda^N(\lambda - 2N)^N = 0 \tag{7.58}$$

Hence the eigenvalues of the matrix (7.56) are

$$\lambda_1 = \lambda_2 = \ldots = \lambda_N = 0 \qquad \text{and} \qquad \lambda_{N+1} = \lambda_{N+2} = \ldots = \lambda_{2N} = 2N$$

It can be written as,

$$\Rightarrow \qquad \lambda(A_1 + A_2 + A_3 + \ldots + A_N) = \lambda\left(\sum_{i=1}^{N} A_i\right) = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 2N \\ 2N \\ \vdots \\ 2N \end{pmatrix} \tag{7.59}$$

Hence, the maximum eigenvalues of the summation $\boldsymbol{A}_i$ is

$$\lambda_{\max}\left(\sum_{i=1}^{N} \boldsymbol{A}_i\right) = 2N \tag{7.60}$$

It shows that no matter how many constraints are actually active, the maximum eigenvalue is still the same. It means,

the maximum eigenvalue when one constraint is active

= the maximum eigenvalue when all constraints are active.

This result, equation (7.60), gives a very interesting fact in the theoretical convergence coefficient μ.

*Remark: There might be a more elegant way to show this using the properties of matrices $\boldsymbol{A}_i$ and some theorems from linear algebra, but we could leave that for future work.

7.4 Bound on the convergence coefficient for magnitude constraints

The structure of the Fourier transform matrix and its eigenvalues have been analysed and the results which are defined in equation (7.49) and equation (7.60) are:

$$\sum_{i=1}^{N} \lambda_{\max}(\boldsymbol{A}_i) = 2N^2 \qquad \text{and} \qquad \lambda_{\max}\left(\sum_{i=1}^{N} \boldsymbol{A}_i\right) = 2N$$

The statement of (7.35) is true, for all cases,

$$\lambda_{\max}\left(\sum_{i=1}^{N} \boldsymbol{A}_i\right) = \lambda_{\max}(\boldsymbol{A}_i) = 2N$$

Since the equation (7.59) shows that

$$\lambda_{\max}(\boldsymbol{A}_1) = \lambda_{\max}(\boldsymbol{A}_1 + \boldsymbol{A}_2) = \cdots = \lambda_{\max}(\boldsymbol{A}_1 + \boldsymbol{A}_2 + \ldots + \boldsymbol{A}_N) = 2N \tag{7.61}$$

it shows that all constraints are independent. No matter how many constraints are actually active, they will be treated individually, and the maximum eigenvalue of $\lambda_{\max}(\boldsymbol{A}_i)$ or $\sum\lambda_{\max}(\boldsymbol{A}_i)$ will be equal. i.e. $\boldsymbol{A}_i$ is independent on $\boldsymbol{A}_j$, where $i \neq j$. Hence, all constraints work individually.

Therefore the theoretical convergence coefficient μ for the constrained FDLMS algorithm with magnitude constraint, equation (7.31), can be written as

$$\mu < \frac{1}{N\lambda_{\max}(\boldsymbol{R}) + 2\sigma N} \tag{7.62}$$

So a sufficient condition for convergence stability of the constrained FDLMS adaptive algorithm can be implemented as follows,

$$\mu < \frac{1}{N^2 E\{x(n)^2\} + 2\sigma N} \tag{7.63}$$

where N is the block size and is also the length of the FIR filter $\boldsymbol{w}$. $E\{x(n)^2\}$ is the variance of the input signal x, and σ is used to control tightness of the penalty function.

If σ is chosen significantly small, the constrained part $2\sigma N$ in equation (7.62) or (7.63) will have a small value, the constraints will thus not affect the bound on μ even when the constraints are active. In practice, we choose σ large enough to maintain the constraint but not too large so that it will not slow the convergence. The value of σ can be designed by the ratio between the value of the unconstrained part, $N^2E\{x(n)^2\}$, and constrained part, $2\sigma N$. As shown in (7.62) and (7.63), the ratio is an essential point to choose the value σ, because:

- If $\sigma = 0 \Rightarrow$ Equivalent to the unconstrained case.

- If the value of $2\sigma N$ is much smaller then $N^2E\{x(n)^2\}$, e.g. when σ is significantly small or $\sigma \to 0 \Rightarrow$ the constrained case will act like an unconstrained case although the constraints can be active. Thus, it has fast convergence but the constraints might not be maintained effectively.

- If the value of $2\sigma N$ is equal or larger then $N^2E\{x(n)^2\}$, e.g. when σ is sufficiently large $\Rightarrow$ the constraints could be maintain properly. However if σ is too large, it may slow the convergence or ill-conditioning might occur.

Therefore the value of σ can be chosen based on the following conditions: (i) to make $2\sigma N \approx N^2E\{x(n)^2\}$ and (ii) to ensure constraint is sufficiently maintained. In chapter 9, the constrained FDLMS, using the new theoretical bound on the convergence coefficient μ, with the magnitude limiting constraint, will be applied to the adaptive sound equalisation system. We use Matlab simulation to ascertain the validity of the theoretical convergence coefficient μ which is defined in (7.62) and show how the parameter σ is chosen to achieve a better trade-off between speed of convergence and maintenance of the constraints.

7.5 Conclusions

A theoretical analysis of the bounds on the penalty parameter σ and the convergence parameter μ, for BLMS and FDLMS algorithms with constraint $|W(k)|^2 < L(k)$, have been developed. It used the first order penalty function with quadratic constraints in the implementation. The new bounds give conditions for the parameters, thus establishing a new framework within which more efficient values of the parameters can be chosen.

We have investigated the validity of the theoretical convergence coefficient bounds by comparing the latter with practical convergence coefficients obtained from simulation tests. The results show that the theoretical convergence coefficient is not far off the practical one. This means that the bounds facilitate improved performance in constrained problems without any "trial and error". This is detailed in chapter 9.

Part III:

Applications

Chapter 8

Introduction to sound equalisation

8.1 Introduction

This thesis is concerned with the development of constrained adaptive filtering algorithms and uses sound equalisation as the application. In order to place the algorithms and theoretical analysis developed in part II into context, this chapter starts with a brief introduction of the field of sound equalisation and describes the problems to be solved. New approaches will be applied in the following chapter with the aim of finding a more efficient means of solving equalisation problems than those that are currently in use.

The chapter is presented as follows. Firstly, the background of sound equalisation in an enclosed room is introduced, followed by an analysis of the sound field in a one-dimensional duct. The adaptive filtering of single channel FIR equalisation is then presented. Most of the material in this chapter is taken from Elliott and Nelson (1992) and so further details can be found in this textbook and other papers.

8.2 Sound equalisation in an enclosed room

In this section, we will give a brief background of some elementary ideas of sound equalisation in an enclosed room that will be used in chapter 9.

Sound equalisation attempts to set the output of the control system's transfer function equal to that of a specified, desired transfer function. The type of function used is determined by the application being studied. A typical equalisation application is shown figure 8.1, and the path between source and the receiver is further described on figure 8.2:

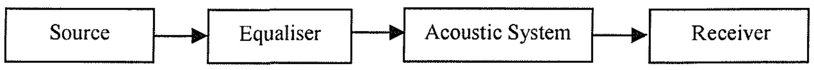


Figure 8.1: A typical equalizer application.

Inverse filtering plays an important role in sound equalisation. Inverse filtering is when the frequency response from the acoustic system to the receiver is inverted. The filter is designed to cancel the distortions that occur and so lessen the error between the desired sound and the actual output. This error can include physical errors from the speaker, transmission paths and reverberations, when the system is based in a room.

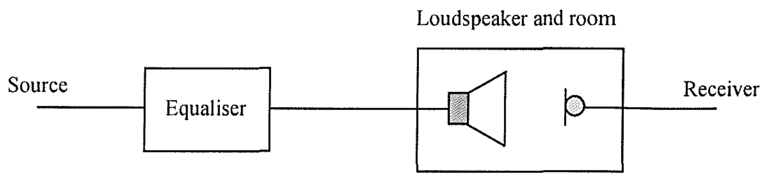


Figure 8.2: Filtering of equaliser and loudspeaker in a room.

Most methods used minimise the MSE as a way of optimising the performance of the system. It can be shown that the cost function can be expressed as a quadratic function and will have a unique global minimum that represents the optimal solution to the equalisation problem. An adaptive filter will be used and a microphone as an on-line reference to enable us to perform the equalisation. Changes in location of the source and microphone can greatly affect the frequency response and therefore performance of the system. Figure 8.3 below represents a typical MSE minimisation in an equalisation system.

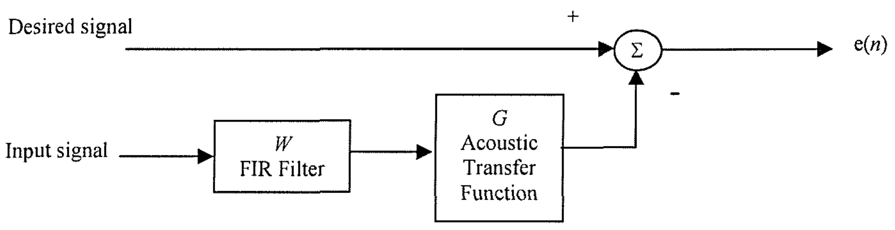


Figure 8.3: Block diagram of equalisation system.

In the above, G represents the transfer function of the room and loudspeaker, and W represents the inverse filter. To find an optimal W, a cost function containing the MSE can be minimised as in previous optimisation problems.

Many papers have been published on equalisation of frequency response for either one or several receiving points, e.g. [Elliott and Nelson, 1989][Nelson, Hamada and Elliott, 1992][Elliott, Bathia, Deghan, Fu, Stewart and Wilson, 1994][Nelson, Orduna-Bustamante and Hamada, 1995][Asano, Suzuki and Sone, 1996]. These systems have been able to achieve good equalisation at the specified receiving points. However, the equalisation areas become small zones around each receiving point as the distance between these points increases. The aim of part III is to increase the robustness of equalisation area around the microphone so that in practical applications people movements can be allowed for whilst maintaining high sound quality, by using the techniques in part II. Since the acoustic path can change due to the movements of people, the adaptive filtering is therefore necessary and used here in order to maintain good equalisation at all time and applying constraints to increase the robustness. The techniques of constrained adaptive filtering algorithms developed in part II will be used in this application.

Distortion often occurs in room-based acoustic applications due to the frequency response of the room. Equalisation filters can be used to minimise this distortion. The undesirable effect of linearly distorted sound can occur under normal listening conditions, particularly at low frequencies and in small rooms [Santillan, 2001]. To improve quality of the output, an adaptive filter can be used to track the changes in the environment using a single microphone. Another method is to use multiple control points [Elliott and Nelson, 1989 and Miyoshi and Kaneda, 1988]. In this approach, extra receiving points are placing near the original point. The objective is to achieve the same result at each of the multiple receiving points. In this way, areas in the room are set where satisfactory equalisation will occur. This may take the form of several disjointed areas, rather than one, since the areas between the equalisation zones may not be smoothly interpolated [Asano, Suzuki and Sone, 1996]. Another alternative approach is to constrain the partial derivatives of the transfer functions at the receiving point to zero, which will give a wider and smoother area of equalisation [Asano, Suzuki and Sone, 1996]. In this approach, the transfer function is smoothly interpolated over the frequency range, and will therefore have one area of equalisation.

The sound pressure at a microphone consists of the sound wave directly from the source and all the waves that reflect onto the microphone, at various incidence angles, from the walls of the room. These reflections are linear distortions of the source and impair the quality of the output to the listener. They are represented in the frequency domain by peaks and troughs in the frequency response of the room, the nature of which will depend on the phase shift and amplitude produced by the sound waves at a given point. In the time domain, these reflections are delayed, attenuated versions of the original source signal, and are represented by reverberations and echoes. By using an FIR filter to approximate the inverse filter, the variation in transfer function at the microphone position is compensated for, and it becomes possible to lessen the effect of these reflections and hence increase the quality of the received soundwave. Equalisation is very difficult to achieve in rooms that have highly reflective surfaces. This is because small changes at the receiving point will have a large effect on the transfer function [Asano and Swanson, 1995].

8.3 Acoustic sound field in a one-dimensional thin duct

In the following, we will introduce briefly some elementary ideas of sound propagation in one-dimensional duct that will be used in chapter 10.

Let's begin with the sound resonance of a thin duct, of length L closed at both ends. The 1-D duct will be observed to exhibit resonance frequencies under the following conditions [Elliott, 2001]:

$$f_n = nc/2L, \qquad c = f\lambda \tag{8.1}$$

where L is the length of the duct, c is the speed of sound in the room and n is an integer (n = 1, 2, 3, 4, ...). The distance travelled by the sound during one cycle of motion is defined by cT (where T is the period of time for one cycle of motion); this distance is one wavelength λ. i.e. λ is the wavelength of the sound. Also the frequency of the motion $f = 1/T$ Hz (cycles per second), so that frequency and wavelength are related by $c = f\lambda$. Since our simulation assumes room temperature, the speed of sound c is taken as around 343ms^{-1}. For example, if the length of the duct is 1 metre, the motion of the sound would be transmitted from one end to the other within about 1/343 of a second.

When plane waves of equal amplitude are travelling in opposite directions, they add in such a way as to produce a wave that oscillates in amplitude but does not move in space. Such a wave field is called a standing wave or a mode. The shape of the standing wave, or mode shape, looks like $\cos(n\pi x/L)$ where n is an integer. The locations where $\cos(n\pi x/L) = 0$ are called nodes. At nodes the acoustic pressure from this mode is always zero (regions of the duct where the pressure is zero). The places where $\cos(n\pi x/L)$ are ± 1 are called anti-nodes. If a source that has energy at the resonance frequency is located at a node, it cannot excite the mode and sound will not build up. If a source is located at an anti-node, it can fully excite a mode and maximum sound levels can be obtained. The mode number, n, corresponds to the number of nodes in the standing wave. For n = 1, there is one pressure node at the centre of the duct at $L/2$ and for n = 2, there are two nodes at $L/4$ and so on for higher values of n, as shown in figure 8.4.

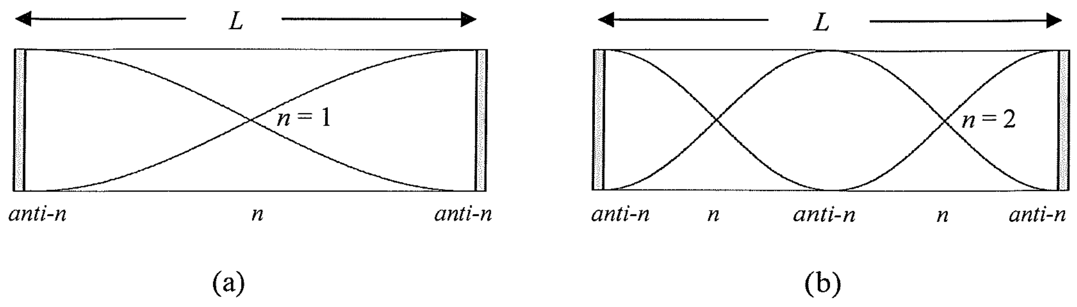


Figure 8.4: The mode shapes in a duct of length L: (a) n = 1 and (b) n = 2.

In a 3-D rectangular rooms, mode shapes can be set up between all three sets of walls. The mode shape end up being of the form $\cos(n_x \pi x/L_x) \cos(n_y \pi y/L_y) \cos(n_z \pi z/L_z)$ where n_x, n_y, and n_z are all integers. However, we only concentrate on a 1-D duct in here, for simplicity.

There are some factors that will affect sensitivity to the position of microphones that needs to be considered:

- Mode shape: greater sensitivity occur near the nodes, especially at the low frequencies when model density is low.

- Poor equalisation will occur at other positions, away from the equalisation microphone, at the nodes; because the frequency response of the acoustic path can vary significantly with position.

Let's now look at some basic formulation for a 1-D duct. The acoustic pressure field $p(x)$ of the enclosed sound field can be expressed using acoustic modes,

$$p(x) = \sum_{n=1}^{N} a_n \Psi_n(x) \tag{8.2}$$

where a_n, $\Psi_n(x)$ (n = 1, …, N) are the nth modal coefficients and mode shape function.

The mode shape function in 1-D duct is calculated by:

$$\Psi_n(x) = \sqrt{\varepsilon_n}\cos\frac{n\pi x}{L} \tag{8.3}$$

where L is the length of the duct, n is the mode number, x is co-ordinate direction, and ε_n is the normalisation factor, $\varepsilon_n = 1$ if $n = 0$ and $\varepsilon_n = 2$ if $n > 0$.

The coefficients a_n (the mode amplitudes) in one-dimensional duct is defined by:

$$a_n = \frac{\omega\rho_0 c_0^2}{2\zeta_n\omega_n\omega + j(\omega^2 - \omega_n^2)} \tag{8.4}$$

where ω_n is the angular frequency of vibration, ρ air density, c speed of sound, ω natural frequency, ζ_n modal damping ratio of nth mode.

In the simulation, as an example of the application of the frequency-dependent constraints developed in the chapter 10 we attempt to equalise total acoustic pressure (8.2) in an enclosed one dimensional space, a long thin duct (figure. 8.4). This measured data is generated by MATLAB programming.

8.4 Adaptive filtering of single channel FIR equalisation

In this section, the theory of single-point equalisation is considered, and later in this thesis the results of a simulation run for a single point equalisation in an enclosed room are presented.

Adaptive filters are often used for identifying time-varying systems. The performance of the adaptive filter is affected by the time it takes for the filter to track the changes occurring and the performance can be judged on the power of the error signal that is left after the equalisation takes place.

The most widely used and simplest to implement algorithm is the LMS. Illustrated below for comparison purposes are the equalisation system and the system applicable for approximating the optimal filter using the LMS algorithm. Details can be referred to Nelson and Elliott (1992).

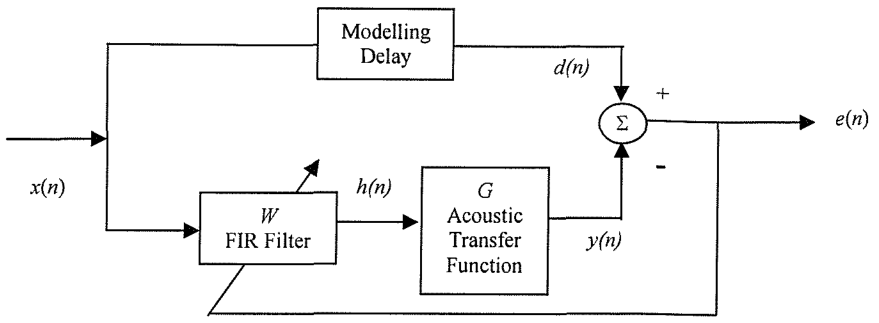


Figure 8.5: Block diagram of equalisation problem with sampled signals.

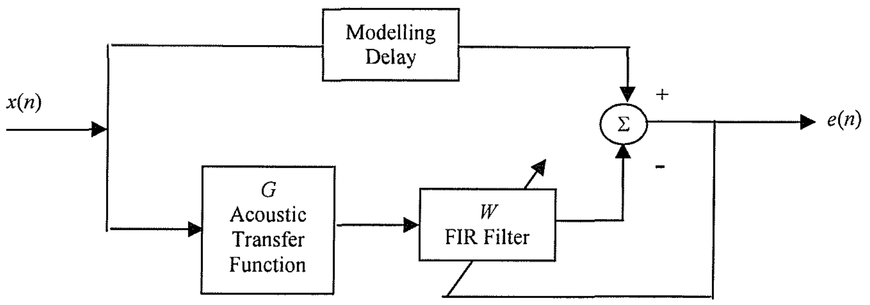


Figure 8.6: Conventional LMS applied to equalisation system.

The positions of the adaptive filter and acoustic plant for the LMS figure 8.6 have changed from those of the standard system figure 8.5. Figure 8.5 is the conventional way of adaptive equalisation and requires the algorithm of filtered-x LMS. The system in figure 8.5 does not correspond to the condition for finding a Wiener filter using the LMS that was stated in section 3.3. It is therefore necessary to change the system as figure 8.6 to enable the use of the LMS and so find the optimal solution. Unfortunately, figure 8.6 does not perform equalisation, since W is connected after the microphone (G-output) and not before the loudspeaker (G-input). This altered system may produce biased results based on any noise that might occur on the output of G, which will subsequently be used to update the filter coefficients of W. However, if W changes slowly compared to the delay in G, figure 8.6 can be considered as an equivalent block diagram as figure 8.5, in the sense that W_{opt} in both cases is the same. Therefore, figure 8.6 can be used to do simple simulations using the LMS (or FDLMS) which do not require the filtered-x LMS, for the analysis of performance of the equalisation. Also, when the filtered-x LMS is used one must find an estimate of G, which usually requires on-line system identification. For simplicity we study equalisation using figure 8.6 to avoid the need for filtered-x LMS and identification of G, but in practice figure 8.5 is usually implemented.

Let's now construct an LMS algorithm that can be used on the system of figure 8.5 and figure 8.6, as will be shown below. (see Nelson and Elliott, 1992)

Taking the standard equalisation system, our sample source signal is $x(n)$. The input to the loudspeaker is:

$$h(n) = \sum_{i=0}^{I-1} w_i x(n-i) \tag{8.5}$$

The input to the microphone from the acoustic plant is then $y(n)$, written as:

$$y(n) = \sum_{j=0}^{J-1} G_j h(n-j) \tag{8.6}$$

$$y(n) = \sum_{j=0}^{J-1} G_j \sum_{i=0}^{I-1} w_i x(n-i-j) \tag{8.7}$$

Equation (8.7), W and G can be exchanged and the system of figure 8.5 can then be solved as figure 8.6. This is only correct if W changes slowly compared to the delay in G. It is given as:

$$y(n) = \sum_{i=0}^{I-1} w_i r(n-i) \tag{8.8}$$

where

$$r(n) = \sum_{j=0}^{J-1} G_j x(n-j) \tag{8.9}$$

is the vector produced by filtering the input signal through the acoustic plant and it is called filtered-x LMS since w acts on r and not x. The above equation for y can be written in vector form as:

$$y(n) = \boldsymbol{r}^{\mathrm{T}}(n)\,\boldsymbol{w}(n) \tag{8.10}$$

where

$$\boldsymbol{r}^{\mathrm{T}}(n) = [r(n), r(n-1), \ldots, r(n-I+1)]$$

$$\boldsymbol{w}^{\mathrm{T}}(n) = [w_0, w_1, \ldots, w_{I-1}]$$

The error is defined as the difference between the actual and desired response:

$$e(n) = d(n) - y(n) \tag{8.11}$$

To find the optimal solution to the equalisation system, the error defined above must be minimised. The MSE is taken as the cost function. That is:

$$J = E[e^2(n)] \tag{8.12}$$

where E{.} is the expectation operator. This choice of cost function is particularly useful as it can be expressed as a quadratic that has a unique global minimum, hence a unique optimal set of filter coefficients. The equation (8.12) can be expanded to:

$$J = E[d^2(n)] - 2\,\boldsymbol{w}^{\mathrm{T}}(n)E[\boldsymbol{r}(n)d(n)] + \boldsymbol{w}^{\mathrm{T}}(n)E[\boldsymbol{r}(n)\boldsymbol{r}^{\mathrm{T}}(n)]\,\boldsymbol{w}(n) \tag{8.13}$$

and the optimal set of filter coefficients written as:

$$\boldsymbol{w}^* = E[\boldsymbol{r}(n)\boldsymbol{r}^{\mathrm{T}}(n)]^{-1}\,E[\boldsymbol{r}(n)d(n)] \tag{8.14}$$

The coefficients of $\boldsymbol{w}$ in the equations above can be adjusted adaptively in practical applications so that a good approximation to the optimal $\boldsymbol{w}^*$can be achieved. The update equation for the system can be written as:

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \alpha\,\boldsymbol{r}(n)e(n) \tag{8.15}$$

where α is the step size, which controls the speed of convergence.

For both systems in figures 8.5 and 8.6 above, there is obviously a trade off between accuracy and convergence speed when choosing the value of α to use in practical applications. If the value of α becomes too large, the algorithm becomes unstable, and may not converge to the optimal solution. Work by Elliott and Nelson suggested that an estimate of the largest stable step size is given by:

$$\alpha_{\max} \approx \frac{2}{\bar{r}^2(1+\delta)} \tag{8.16}$$

where δ is a pure delay in the error path and detail can be referred to Elliott and Nelson, 1989. In practical applications, the filtered reference signal $r(n)$ must be estimated; the filtered-x LMS algorithm is, however, able to cope quite effectively with the errors caused by these estimates.

Some current papers [Elliott, 1994][Elliott and Nelson, 1996] have shown that although it has been possible to achieve satisfactory equalisation at a single receiving position, the quality of equalisation drops off away from the receiving point. At points far away from the specified receiving point, the equalised response may in fact be of a lower quality than the un-equalised response. It is possible to increase the robustness of sound equalisation away from the receiving point, in relation to movement of the sound source or receiver in an enclosed space, by using the technique in part II, and is shown in the following chapter.

8.5 Conclusions

The aim of this chapter is to first review equalisation in general, then to look at literature relating to enclosed room applications in particular. The current algorithms and techniques that are used for optimisation problems in this area have also been discussed. The knowledge gained from this chapter will be combined with the algorithms developed in part II for practical testing. In the next two chapters, the algorithms and the theoretical analysis developed in part II will be used for adaptive sound equalisation with the aim of improving the spatial robustness and convergence rate.

Chapter 9

Application to adaptive sound equalisation

9.1 Introduction

In chapter 6, a constrained FDCGA algorithm and constrained BN-FDLMS algorithm for adaptive filtering were formulated with calculations performed in the frequency domain offering computational savings. In chapter 7, the new theoretical bound on the convergence coefficient μ and the penalty parameter σ, for the constrained FDLMS, were also developed. In this chapter, the constrained approach with FDCGA and BN-FDLMS will be used for adaptive sound equalisation with the aim of improving the spatial robustness and convergence rate. We will also use the sound equalisation application to ascertain the validity of the bound on μ, and also see how σ is chosen.

A fast convergence rate is required in order to maintain good equalisation at all times because the acoustic path can change due to the movements of people around the room, for example. Improved convergence rate is achieved by employing the FDCGA and BN-FDLMS due to the improved search directions employed by the CG algorithm over the LMS, and the frequency-dependent convergence coefficient used by the BN-FDLMS. Simulations will be used here to demonstrate that the algorithms can improve the performance over the current LMS algorithm for the adaptive filtering sound equalisation problem, in terms of convergence rate.

Spatial robustness is achieved by constraining the magnitude of the adaptive equalisation filter, because large magnitudes in the equalisation filter could cause large increases in sounds away from the equalisation point. The magnitude constraint is therefore applied to limit the magnitude and prevent high sound amplification, so that it can improve the equalisation performance at positions away from the microphone. The performance of the algorithms with a constant magnitude constraint is discussed in this chapter. Since low frequency sounds in a room are more spatially correlated, higher gains in the equalisation filter are allowed at low frequencies due to the relatively small spatial variability, while tighter gains are applied at high frequencies to further increase the spatial robustness. This spatially robust sound equalisation which uses frequency-dependent constraints is discussed in the chapter 10.

For ease of comparison with the LMS-based algorithms, the acoustic plant and modelling delay data from Rafaely and Elliott's (2000b) is used for the simulation; Rafaely and Elliott (2000b) specifically solved the equalisation problem by using the LMS algorithm with constant constraints using second order penalty function. In the chapter, the performance of the FDLMS, FDCGA and BN-FDLMS algorithms, with constant magnitude constraints and with first order penalty function is compared for both unconstrained and constrained cases, using the measured data as described in Rafaely and Elliott (2000b). The reader is referred to this paper for further information on the data set. The results from the simulations are presented and discussed in this chapter.

This chapter is presented as follows. We first state the problems in sound equalisation in an enclosure that are required to be improved. Then ascertain the validity of the theoretical bound on the convergence coefficient μ which is developed in chapter 7. Finally the performance of the FDLMS, FDCGA and BN-FDLMS algorithms with constant magnitude constraint are discussed.

9.2 The sound equalisation problem in an enclosure

The purpose of sound equalisation is to compensate for amplitude distortion caused by the frequency response of the acoustic path and the loudspeaker in the sound reproduction system, i.e. equalise the source such that the magnitude of the response from source to receiver is unity. A number of authors have considered using adaptive filtering for audio equalisation. Adaptive filters are used in sound equalisation systems to control the frequency response of a loudspeaker at a specific position in a room. In many sound applications, although good equalisation can be achieved at the microphone positions, the equalisation at near-by positions, as illustrated in figure 9.1, may be poor.

A good equaliser will be spatially robust to ensure good performance at some region around the microphone. Figure 9.1, illustrates the potential problem; good equalisation can be achieved at the microphone position G_1, but poor equalisation might occur at position G_2. This is because the frequency response of the acoustic path can vary significantly with position. An equalisation filter W may produce peaks to compensate for notches in the acoustic path G at the microphone position. At positions away from the microphone, because of the interference between acoustic modes, the notches may occur at different frequencies, and the peaks in W will cause undesired amplification. On the other hand, if the peaks in G

move or change magnitude when the microphone position is changing, then the notches in W will cause a notch in the equalised response, which will usually not result in much degraded audibility. The effects of the distortion caused by the peaks in the equalisation filter W can be counteracted to some degree by applying a frequency gain limit to the adaptive filter.

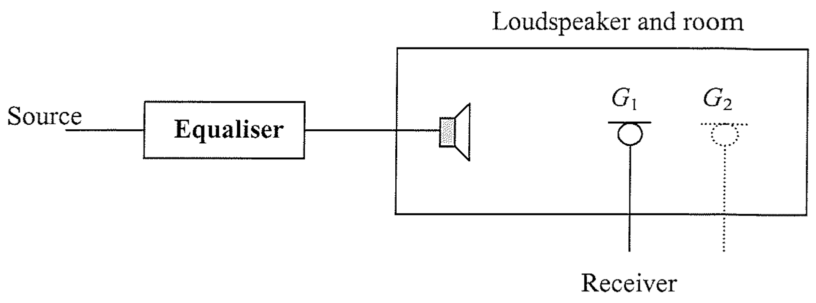


Figure 9.1: Equalisation problem in an enclosure (room) - Filtering of equaliser and the loudspeaker-room system.

This chapter proceeds to apply the optimisation algorithms to an adaptive filtering equalisation system as illustrated in figure 9.2, by using the magnitude constraint. When a microphone is used to record the output of a loudspeaker, this can be used to adaptively filter the input to the loudspeaker through an adaptive sound equalisation system, in real time. An example of this is sound reproduction, where the distortion caused by the frequency response of the loudspeaker and acoustic path can be compensated for by using an adaptive equalisation filter.

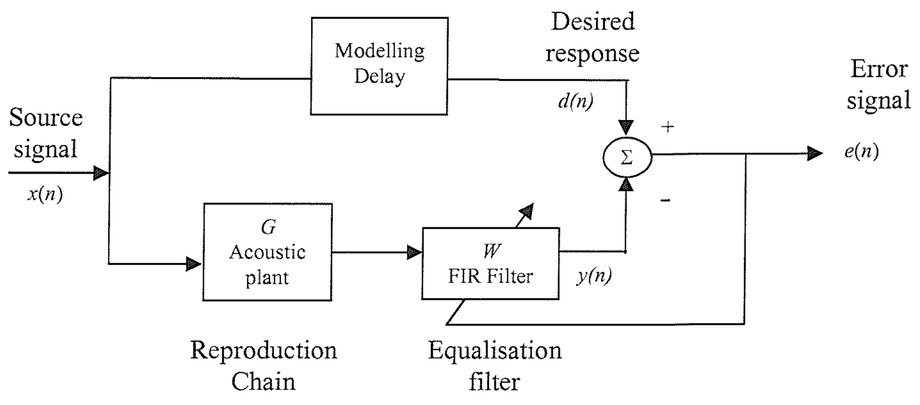


Figure 9.2: Block diagram of an adaptive sound equalisation system.

The figure 9.2 shows a block diagram of a sound equalisation system, in which the inverse of the acoustic plant G is modelled by an adaptive filter W. The causality constraints on the adaptive filter that are included in the system may limit performance; a modelling delay is therefore introduced to negate these effects. The acoustic plant will not significantly affect the amplitude response of the equalised input signal when we achieve the minimum error, and the output signal y is close to the input signal x, i.e. $|WG| = 1$ can be achieved.

The following figures give the idea of how the responses look like in sound equalisation system and give an idea of the problems attempted to solve here. Figures below are obtained by using the FDLMS algorithm. The detail of the simulation is stated in the next section. Figure 9.3 shows the resulting magnitude response $|W|$ after convergence. Figure 9.4 shows the equalised response $|WG_1|$, where G_1 is the magnitude response of the acoustic path from loudspeaker to equalisation microphone. Figure 9.5 shows the equalised response $|WG_2|$, where G_2 is the response between the microphone and the loudspeaker and is measured at a position 10 cm away from G_1. Figure 9.6 shows the different between the response G_1 and G_2. It can be seen that response G_2 is different from G_1, and therefore the filters designed to equalise G_1 cannot produce good sound equalisation of G_2, i.e. the sound quality significantly drops off away from the equalisation microphone. Therefore the performance at positions away from the microphone need to improve robustness and a fast convergence rate is also required in order to maintain good equalisation at all time.

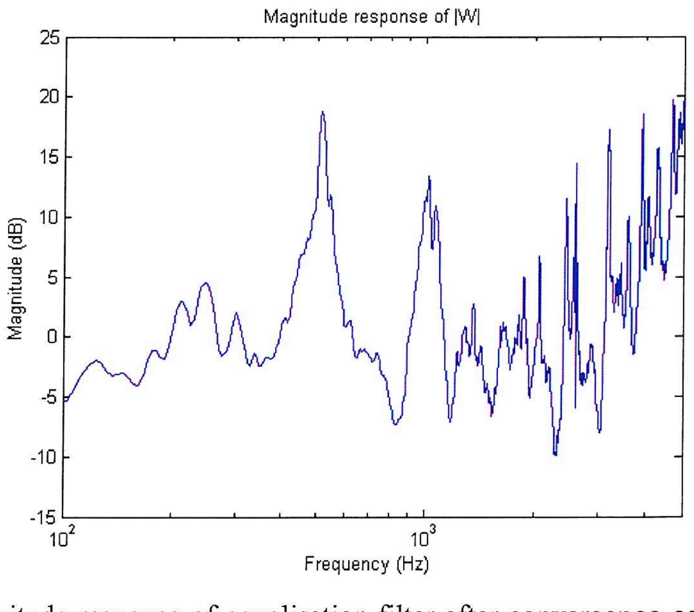


Figure 9.3: Magnitude response of equalisation filter after convergence corresponding to the FDLMS.

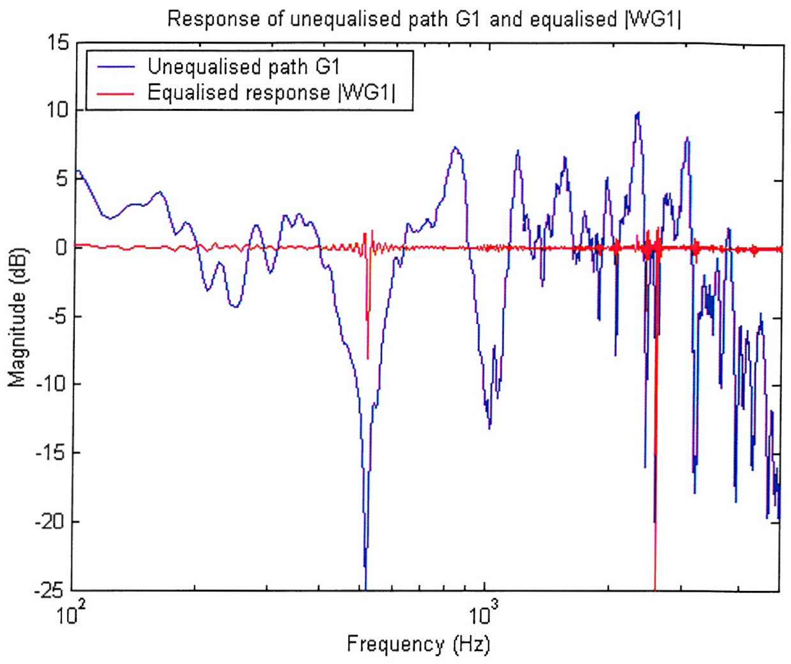


Figure 9.4: Magnitude response of unequalised path G_1 from loudspeaker to microphone, and equalised response $|WG_1|$, using the FDLMS algorithm.

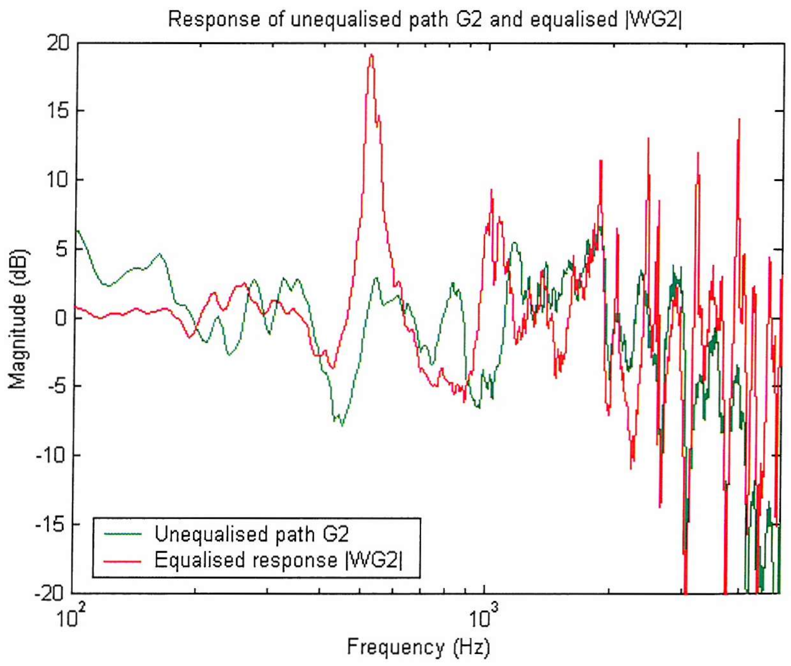


Figure 9.5: Magnitude response of unequalised path G_2 from loudspeaker to location 10cm from the equalisation microphone, and corresponding equalised response $|WG_2|$, using the FDLMS algorithm.

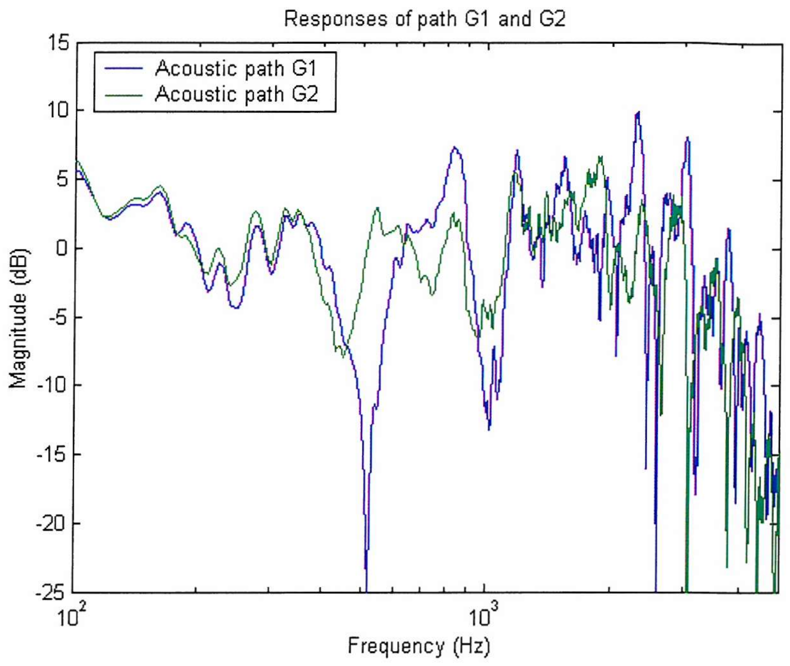


Figure 9.6: Magnitude response of the acoustic paths G_1 and G_2.

9.3 Verification of the theoretical stability bound

The constrained FDLMS with the new theoretical bound on the convergence coefficient μ, with the magnitude limiting constraint and the first order penalty function, is now applied to the adaptive filtering equalisation as illustrated in figure 9.2. In this section, the sound equalisation simulation is used to ascertain the validity of the theoretical bound on the convergence coefficient μ which is developed in chapter 7, and also investigate how σ is chosen.

The maximum theoretical value of the convergence coefficient is first calculated, and then the practical maximum convergence coefficient is investigated using simulations and compared with the theoretical one. The practical maximum convergence coefficient is defined as the value just before instability occurs. The practical maximum convergence coefficient is found by redoing the simulations until the adaptive filter starts going unstable. The practical maximum convergence coefficient just before instability was then chosen. The results in this section show that the theoretical convergence coefficient is not far off the practical one.

The constrained FDLMS adaptive algorithm with the first order penalty function is simulated in MATLAB with 5000 iterations, using a sampling frequency of 10kHz, a block size of 1024 samples, a filter length $N = 1024$ and constraint sets $|W(k)|^2 < 2\text{dB}$. The measured frequency response of the acoustic path G_1 between a loudspeaker and a microphone in an enclosure is taken from Rafaely and Elliott (2000b), and used in the simulation to estimate the equalisation at the microphone location. Figures 9.3 - 9.6 in previous section are based on the simulation described here, we can therefore refer back to the figures for the all magnitude responses. The impulse response of the acoustic plant path model used in the simulation is shown in figure 9.7 at a sampling frequency of 10kHz.

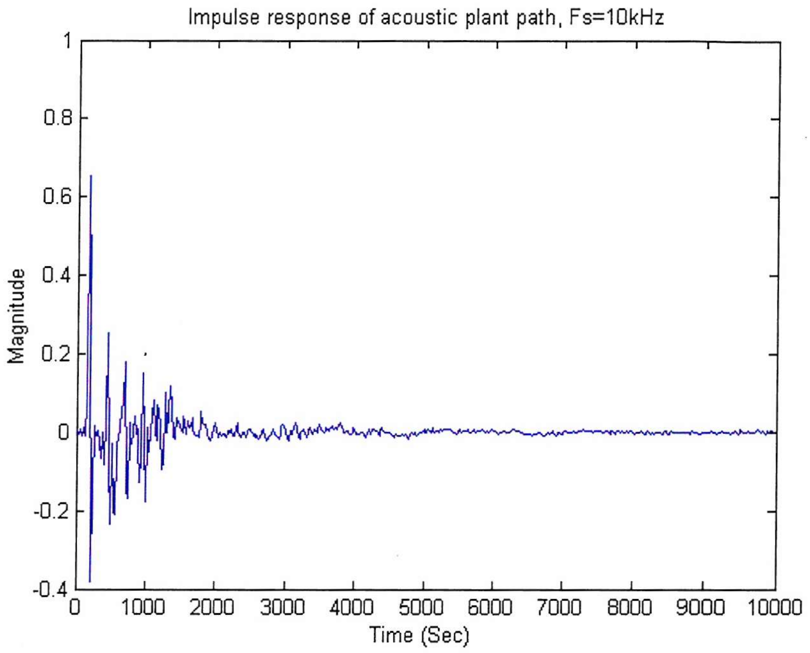


Figure 9.7: Impulse response of acoustic plant model at sample frequency of 10kHz.

Before we conduct the simulations, we need to consider the eigenvalue spread of the autocorrelation matrix $\boldsymbol{R}$ of the reference signal x (the power spectrum of the reference signal x). If the eigenvalue spread of $\boldsymbol{R}$ is small (i.e. the power spectrum of x is flat), then the sufficient condition for convergence will give a more conservative boundary. This is because equation (7.63) is defined by an upper bound $\lambda_{\max}(\boldsymbol{R}) \leq$ trace $(\boldsymbol{R}) = NE\{x(n)^2\}$, since $\lambda_{\max}(\boldsymbol{R})$ cannot be found a priori in some cases. So if the eigenvalue spread of $\boldsymbol{R}$ is small, it implies $\lambda_{\max}(\boldsymbol{R})$ is much smaller than tr[$\boldsymbol{R}$]. In other words, the larger the eigenvalue spread of the matrix (a non-flat power spectrum) is, the value of $\lambda_{\max}(\boldsymbol{R})$ will get closer to the value of tr[$\boldsymbol{R}$], which means that equation (7.63) will become less conservative. Therefore as in the deterministic case the convergence rate is essentially limited by the eigenvalue spread of the data correlation matrix.

Hence if $\lambda_{\max}(\boldsymbol{R})$ is known a priori, then equation (7.62) can be used, and there is no need to consider the eigenvalue spread of $\boldsymbol{R}$. In this simulation, $\lambda_{\max}(\boldsymbol{R})$ is used because the matrix $\boldsymbol{R}$ can be obtained a priori. Therefore equation (7.62) can be used here as the sufficient condition for convergence. The maximum theoretical convergence coefficient μ is chosen as follows to compare with the practical maximum μ. The maximum theoretical μ is

$$\mu_{\max} = \frac{1}{N\lambda_{\max}(\boldsymbol{R}) + 2\sigma N} \tag{9.1}$$

The value of σ is to chosen so as to ensure constraint is sufficiently maintained. In general, the steepness of unconstrained surface should be equal or similar to the steepness of constrained surface. Therefore the condition $2\sigma N \approx N\lambda_{\max}(\boldsymbol{R})$ is required in order to make the penalty function large enough to maintain the constraint; otherwise, if $2\sigma N << N\lambda_{\max}(\boldsymbol{R})$, then the penalty function is much less steep than the unconstraint function hence it will act like an unconstrained case although the constraints can be active; or if $2\sigma N >> N\lambda_{\max}(\boldsymbol{R})$, then it will slow the convergence time and will result in large penalties even for small violation of the constraint hence it will produce a more rapidly varying error surface.

Let's now look at the results of the simulations. Comparing the maximum theoretical and practical convergence coefficients μ, with different penalty parameter σ, in the following cases 1, 2, 3 and 4 that was mentioned in section 7.4.

The following table 9.1 summarise the comparison of the maximum theoretical μ and the maximum practical μ that we obtained in the simulation study. The result of table 9.1 shows that $\sigma = 1$ and 5 which implies $2\sigma N < N\lambda_{\max}(\boldsymbol{R})$ and $2\sigma N \approx N\lambda_{\max}(\boldsymbol{R})$ respectively, the values of practical convergence coefficient μ are almost the same as the maximum theoretical one. When the value of σ is chosen as $2\sigma N << N\lambda_{\max}(\boldsymbol{R})$ or $2\sigma N >> N\lambda_{\max}(\boldsymbol{R})$, then the value of practical convergence coefficient μ needs to be decreased or increased to compensate the different between the constraint term and the unconstraint term. However the result of table 9.1 shows that the theoretical μ is not far off the practical μ. The new established bounds defined in chapter 7 therefore actually enable the design of constrained adaptive filters with accurate choice of μ and σ.

The detail of table 9.1 is presented in the following various simulations in cases 1 – 4. The results of the simulations below show the convergence stability of the error learning curve.

σ	Constraint term $2\sigma N$	Practical μ $\left(\text{where } \mu_{\max} = \dfrac{1}{N\lambda_{\max}(\boldsymbol{R}) + 2\sigma N}\right)$
0	$<< N\lambda_{\max}(\boldsymbol{R})$	$0.6 \times \mu_{\max}$
0.01	$<< N\lambda_{\max}(\boldsymbol{R})$	$0.65 \times \mu_{\max}$
1	$< N\lambda_{\max}(\boldsymbol{R})$	$0.8 \times \mu_{\max}$
5	$\approx N\lambda_{\max}(\boldsymbol{R})$	$1 \times \mu_{\max}$
10	$>> N\lambda_{\max}(\boldsymbol{R})$	$1.3 \times \mu_{\max}$

Table 9.1: Comparison of maximum theoretical μ and maximum practical μ.

Case 1: Set $\sigma = 0$, then compare the maximum theoretical value and the practical convergence coefficient for the unconstraint case.

The results of case 1 show that when $\sigma = 0$, equation (7.62) becomes $\mu < \dfrac{1}{N\lambda_{\max}(\boldsymbol{R})}$, which basically equals to the unconstrained case as in equation (7.5). To study the stability of the adaptive filters, several convergence coefficients have been chosen and tested, and are shown in figures 9.8, 9.9 and 9.10. In figure 9.8, the practical convergence coefficient is chosen as $\mu = 0.6\mu_{\max}$, and it shows noisy performance but stability is maintained. In figure 9.9, $\mu = 0.7\mu_{\max}$ and the adaptive filter started to go unstable. Instability also occurs when using the maximum theoretical convergence coefficient $\mu = \mu_{\max}$, as shown in figure 9.10. Hence from figures 9.8, 9.9 and 9.10, we have the following results:

- With the maximum theoretical convergence coefficient $\mu = \mu_{\max} \Rightarrow$ instability occurs;

- With the practical convergence coefficient $\mu = 0.7\mu_{\max} \Rightarrow$ the algorithm starts to go unstable;

- With the practical convergence coefficient $\mu = 0.6\mu_{\max} \Rightarrow$ this is just before instability occurs and the algorithm converges to the optimum solution, although misadjustment noise is large.

Therefore the practical maximum convergence coefficient (just before instability) with the FDLMS algorithm for unconstrained case ($\sigma = 0$) is:

$$\mu_{\underset{\max}{\text{Practical}}} = \frac{0.6}{N\lambda_{\max}(\boldsymbol{R})} \tag{9.2}$$

$$\Rightarrow \qquad \mu_{\underset{\max}{\text{Practical}}} = 0.6\mu_{\max}\,, \qquad \text{when } \sigma = 0 \tag{9.3}$$

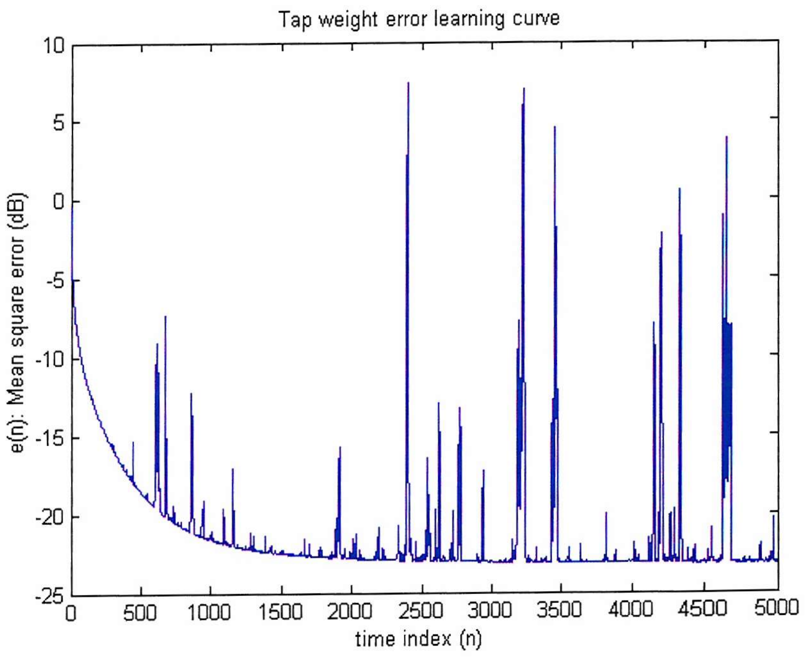


Figure 9.8: Mean square error estimate, with σ = 0, practical convergence coefficient used is $\mu = 0.6 / N\lambda_{\max}(\boldsymbol{R}) = 0.6 \times \mu_{\max} = 0.000061415$.

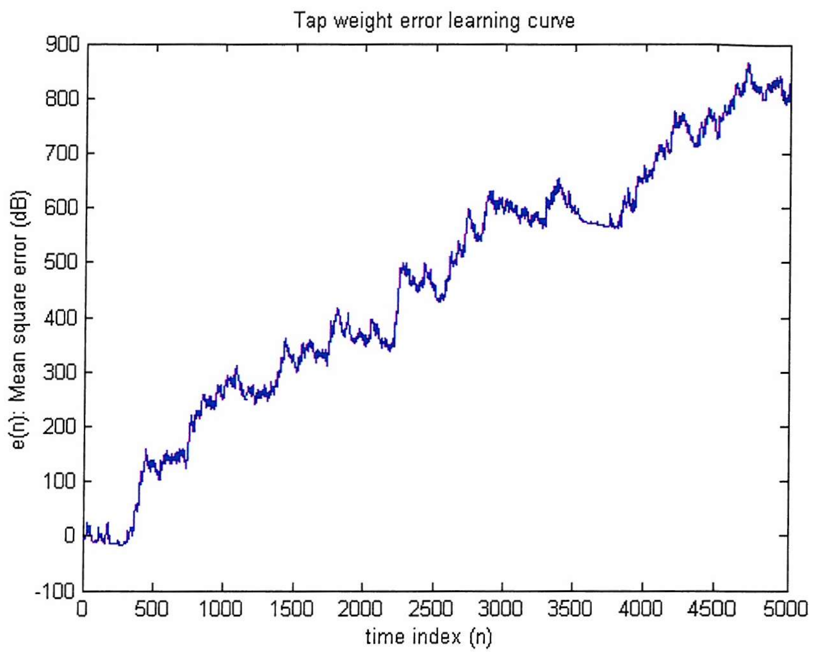


Figure 9.9: Mean square error estimate, with $\sigma = 0$, practical convergence coefficient used is $\mu = 0.7 / N\lambda_{\max}(\boldsymbol{R}) = 0.7 \times \mu_{\max} = 0.000071651$.

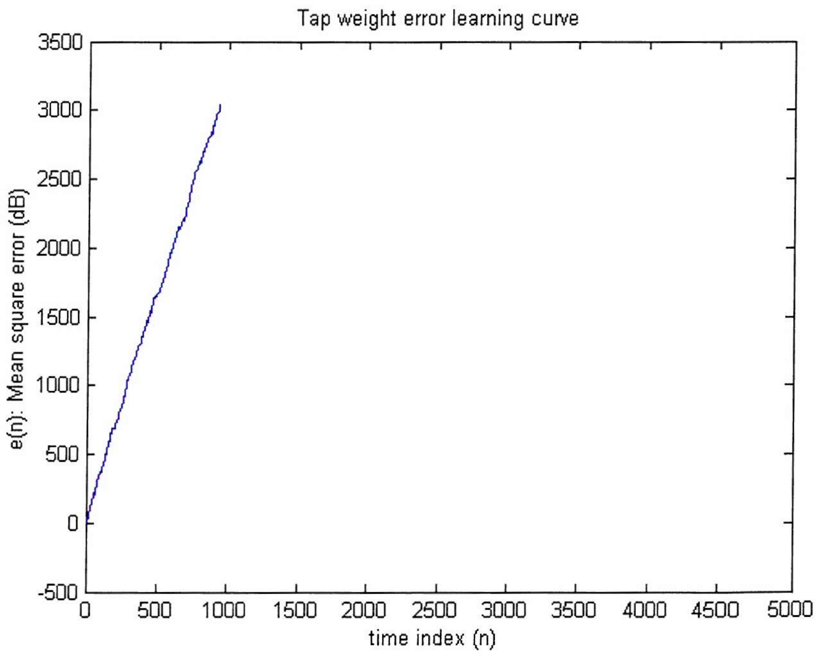


Figure 9.10: Mean square error estimate, with $\sigma = 0$, maximum theoretical convergence coefficient used is $\mu = \mu_{\max} = 0.00010236$.

Case 2: Set σ to a small value, then compare the maximum theoretical and practical convergence coefficients.

To see how the constrained algorithm with a small σ works, it will be compared with the unconstrained case as shown in the figures. Since the performances of case 1 shows $\mu_{\max}$ is too large for the unconstrained case, the maximum practical convergence coefficient $\mu = 0.6/N\lambda_{\max}(\boldsymbol{R})$ is therefore used for the unconstrained case here.

Now we set $\sigma = 0.01$ for this case, which implies that the constraint term $2\sigma N = 20.48$, and the unconstraint term $N\lambda_{\max}(\boldsymbol{R}) = 9769.6$. Thus the difference between the two terms is large, i.e. the ratio of $\frac{\text{Unconstraint term}}{\text{Constraint term}}$ in equation (9.1) is large. This means that if $2\sigma N << N\lambda_{\max}(\boldsymbol{R})$, then the penalty function is much less steep than the unconstraint function, it is also likely that the constraint will be violated due to the weak penalty, i.e. the penalty parameter $\sigma = 0.01$ is too small for the constraints to be maintained. The performance of the maximum theoretical convergence coefficient with $\sigma = 0.01$ is shown in figure 9.11, and is similar to figure 9.10 in case 1. This is because the value $\mu = \mu_{\max}$ used here is similar to the one we used in case 1 since the penalty parameter σ is small, i.e. the constraint part $2\sigma N$ has no effect for $\mu_{\max}$. Therefore $\mu = \mu_{\max}$ with $\sigma = 0.01$ is too large for convergence, like in case 1.

The result of case 2 (figure 9.11 - 9.13) shows that when $\sigma = 0.01$, the practical maximum convergence coefficient (just before instability) with the FDLMS algorithm is:

$$\mu_{\substack{\text{practical}\\ \max}} = \frac{0.65}{N\lambda_{\max}(\boldsymbol{R}) + 2\sigma N}$$

$$\Rightarrow \qquad \mu_{\substack{\text{Practical}\\ \max}} = 0.65\mu_{\max}, \qquad \text{when } \sigma = 0.01 \qquad (9.4)$$

The validity of the maximum practical convergence coefficient (9.4) is ascertained in figure 9.12. In figure 9.12, equation (9.4) gives convergence, but just before instability occurs. Also as shown in figure 9.13, the adaptive filter started going unstable when $\mu = 0.7/[N\lambda_{\max}(\boldsymbol{R}) + 2\sigma N]$, i.e. equation (9.4) is chosen as the maximum practical convergence coefficient for constrained case with $\sigma = 0.01$. Figure 9.14 shows the magnitude of the filter after convergence, which is not constrained to 2dB as required by the constraints, even though figure 9.12 shows that the constraints are active. This is because the small penalty parameter results in relaxed constraints.

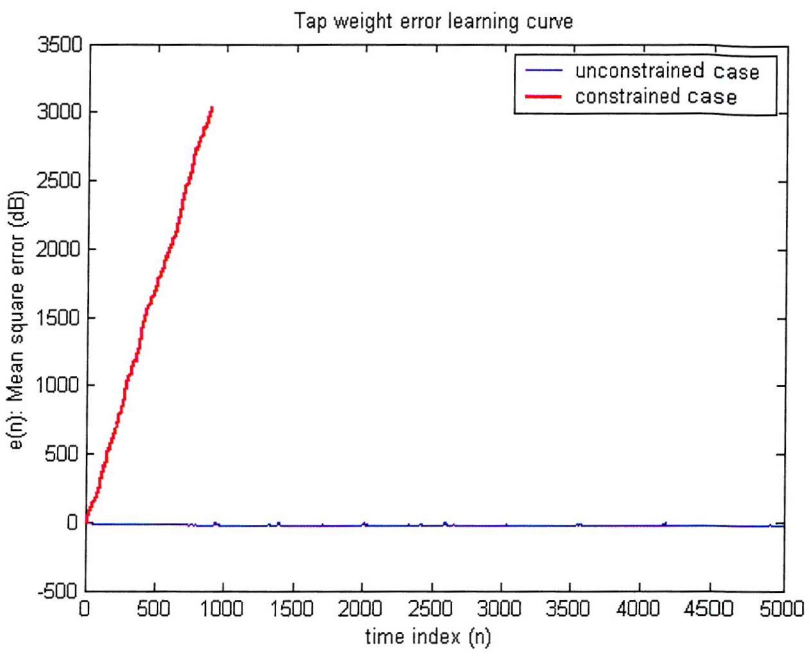


Figure 9.11: Mean square error estimate, with $\sigma = 0.01$, maximum theoretical convergence coefficient $\mu_{\max}$ used for constrained case (red) is $\mu_{\max} = 1/(N\lambda_{\max}(\boldsymbol{R}) + 2\sigma N)$ $= 0.00010214$ and $\mu = 0.6/N\lambda_{\max}(\boldsymbol{R})$ is used for unconstrained case (blue).

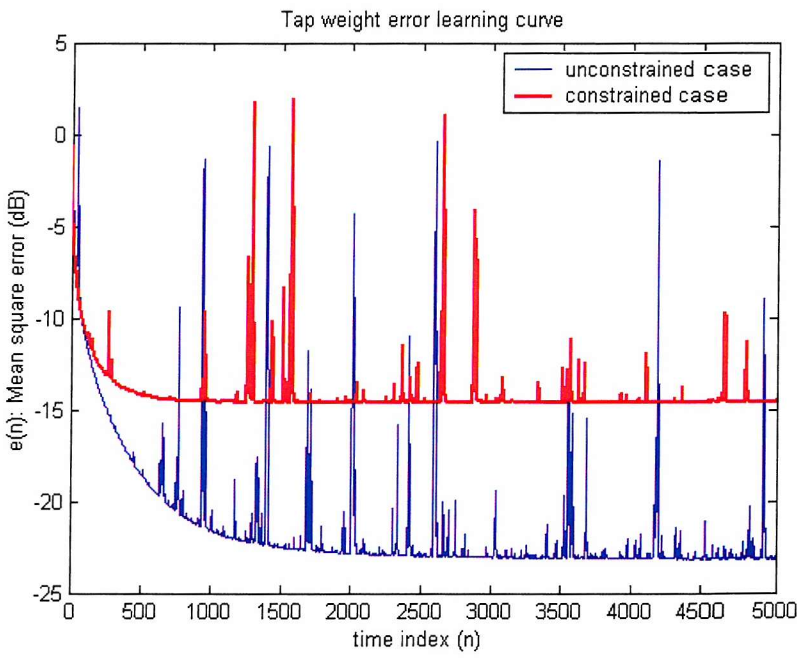


Figure 9.12: Mean square error estimate, with $\sigma = 0.01$, maximum practical convergence coefficient μ used for constrained case (red) is $\mu = 0.65\mu_{\max}$, and $\mu = 0.6/N\lambda_{\max}(\boldsymbol{R})$ is used for unconstrained case (blue).

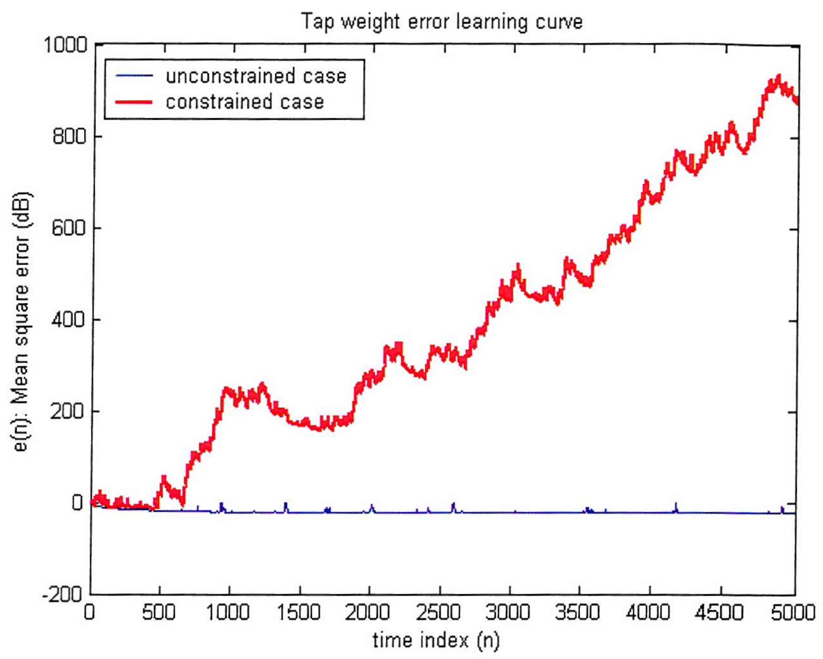


Figure 9.13: Mean square error estimate, with $\sigma = 0.01$, practical convergence coefficient μ used for constrained case (red) is $\mu = 0.7\mu_{\max}$, and $\mu = 0.6 / N\lambda_{\max}(\boldsymbol{R})$ is used for unconstrained case (blue).

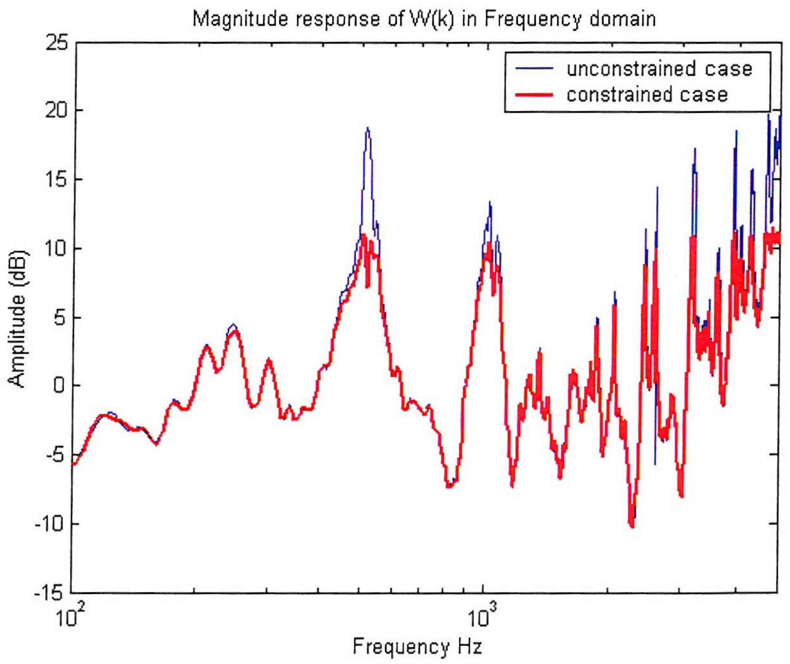


Figure 9.14: Set $\sigma = 0.01$. Magnitude response of equalisation filter after convergence corresponding to the FDLMS (blue), and the FDLMS with constraint on its magnitude (red) by using maximum practice convergence coefficient $\mu = 0.65\mu_{\max}$ with $\sigma = 0.01$.

Case 3: Set σ to a value larger then case 2, then compare the maximum theoretical and practical convergence coefficients.

We set $\sigma = 1$ in this case, which implies that the ratio of $\frac{\text{Unconstraint term}}{\text{Constraint term}} = \frac{N\lambda_{\max}(\boldsymbol{R})}{2\sigma N} = 4.77$ in equation (9.1) which is small. This means that constraint part will play an essential role in the boundary (9.1).

As shown in figure 9.15, the theoretical maximum convergence coefficient is still too large for convergence stability, like cases 1 and 2. The maximum practical convergence coefficient $\mu_{\text{Practical}}$ used in the case of $\sigma = 1$ is:

$$\mu_{\substack{\text{practical}\\ \max}} = \frac{0.8}{N\lambda_{\max}(\boldsymbol{R}) + 2\sigma N},$$

$$\Rightarrow \qquad \mu_{\substack{\text{Practical}\\ \max}} = 0.8\mu_{\max}, \qquad \text{when } \sigma = 1 \tag{9.5}$$

Once again in figure 9.16, the result ascertains the validity of equation (9.5). Figure 9.16 shows that equation (9.5) gives the value just before instability occurs, and figure 9.17 shows that the adaptive filter started going unstable when using $\mu = 0.85/[N\lambda_{\max}(\boldsymbol{R}) + 2\sigma N]$.

Using the maximum practical convergence coefficient (9.5) with $\sigma = 1$, figure 9.18 clearly shows how the magnitude of the filter with constraint is limited to 2dB. This is because, as we mentioned before, the ratio between $2\sigma N$ and $N\lambda_{\max}(\boldsymbol{R})$ is small, and hence the constraint part $2\sigma N$ will affect the constrained performance. However, σ cannot be too large as it may slow the convergence and ill-conditioning might occurs.

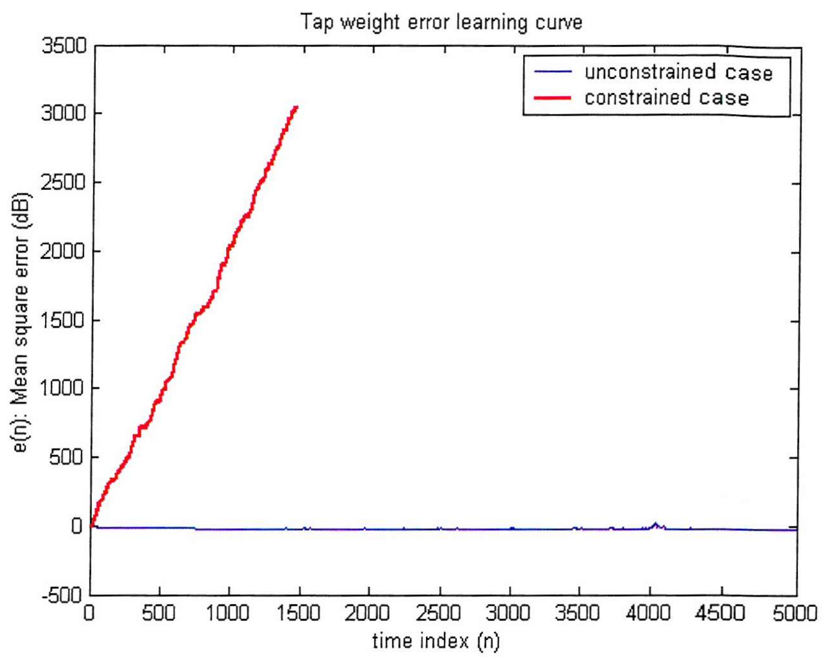


Figure 9.15: Mean square error estimate, with $\sigma = 1$, maximum theoretical convergence coefficient $\mu_{\max}$ used for constrained case (red) is $\mu_{\max} = 1/(N\lambda_{\max}(\boldsymbol{R}) + 2\sigma N)$ = 8.4620e-005, and $\mu = 0.6 / N\lambda_{\max}(\boldsymbol{R})$ = 6.1415e-005 is used for unconstrained case (blue).

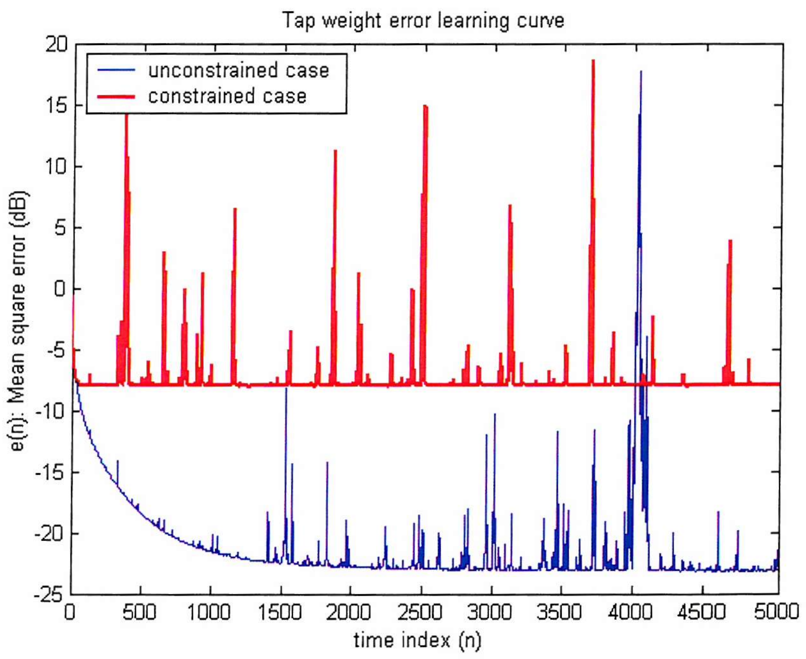


Figure 9.16: Mean square error estimate, with $\sigma = 1$, maximum practical convergence coefficient used is μ = 0.8 $\mu_{\max}$ for constrained case (red), and $\mu = 0.6 / N\lambda_{\max}(\boldsymbol{R})$ is used for unconstrained case (blue).

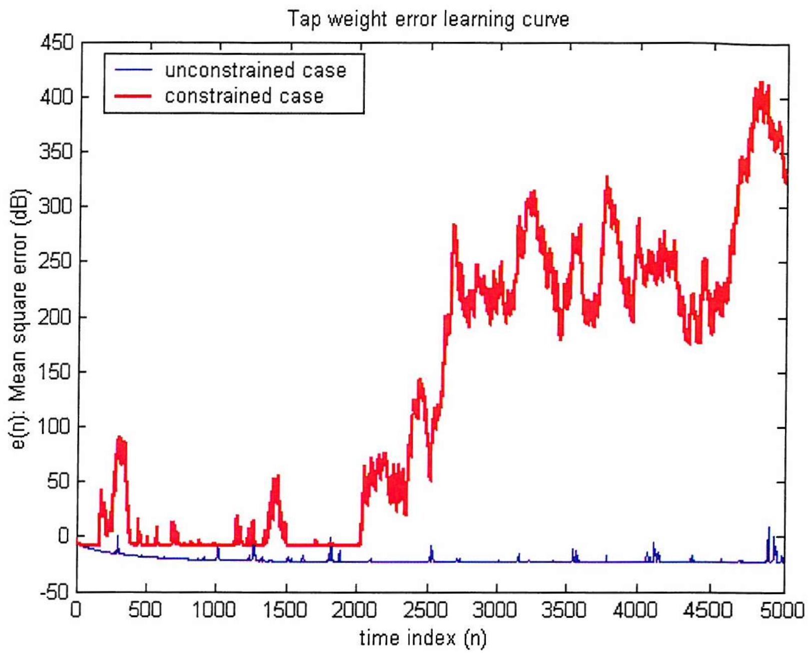


Figure 9.17: Mean square error estimate, with $\sigma = 1$, practical convergence coefficient used is $\mu = 0.85\ \mu_{\max}$, for constrained case (red), and $\mu = 0.6 / N\lambda_{\max}(\boldsymbol{R})$ is used for unconstrained case (blue).

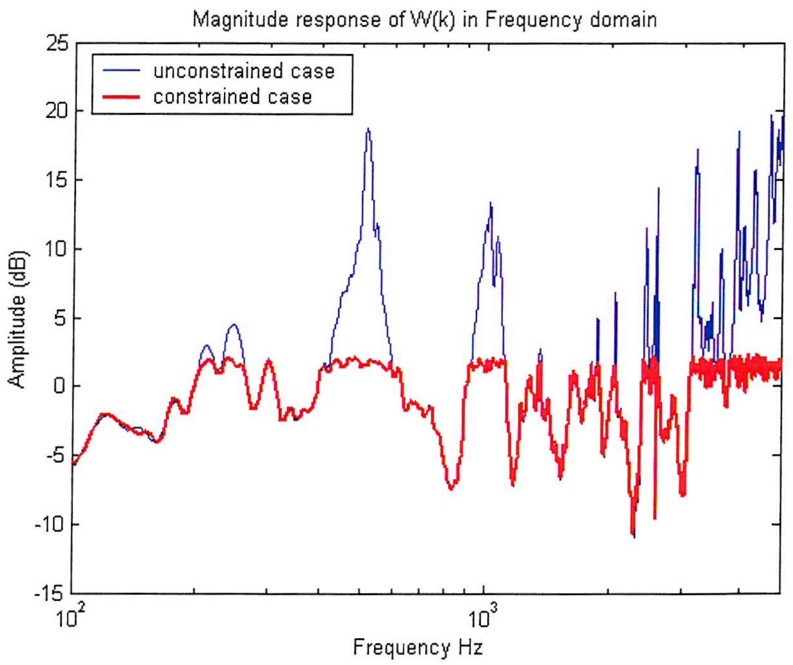


Figure 9.18: Set $\sigma = 1$. Magnitude response of equalisation filter after convergence corresponding to the FDLMS (blue), and the FDLMS with constraint on its magnitude (red) by using maximum practice convergence coefficient $\mu = 0.8\ \mu_{\max}$ with $\sigma = 1$.

Cases 2 and 3 showed that the value of σ can be designed by the ratio of $2\sigma N$ and $N\lambda_{\max}(R)$. We choose σ large enough to maintain the constraint but not too large so that it may cause instability and slow convergence. Let's now look at the performance of the convergence when using larger σ.

Case 4: Set $\sigma = 10$, then compare the maximum theoretical value and the practical convergence coefficients.

When $\sigma = 10$, it implies that the constraint term $2\sigma N = 20480$, hence $2\sigma N >> N\lambda_{\max}(R)$. Therefore the ratio between the two terms in equation (9.1) is very small, which is 0.477. Due to the large σ used here, the penalty parameter is very close to the constraints boundary, and it results in large penalties even for small violation of the constraints; thus ensuring tight maintenance of the constraints, but at the price of producing a more rapidly varying error surface, as shown in figure 9.22. Figure 9.22 shows that not all frequencies reached the constraint boundary, as in figure 9.18. This might also cause a result of slow convergence due to the smaller μ, although the effect of the mean square error is small and so this might not significantly affect the total error and is not seen is figure 9.20. Using $\sigma = 10$ (figure 9.20), the error learning curve gives a smoother convergence than using $\sigma = 1$ (figure 9.16) due to the smaller μ involved. The performance of the maximum practical convergence coefficient with $\sigma = 10$ is shown in figure 9.20. Figure 9.21 shows that the adaptive filter started going unstable when using $\mu = 1.4/[N\lambda_{\max}(\boldsymbol{R}) + 2\sigma N]$.

Figure 9.19 shows the theoretical maximum convergence coefficient is not too large for convergence stability in this case due to the large σ used. When $\sigma = 10$, the maximum practical convergence coefficient obtained by the simulation is:

$$\mu_{\underset{\max}{\text{practical}}} = \frac{1.3}{N\lambda_{\max}(\boldsymbol{R}) + 2\sigma N}$$

$$\Rightarrow \qquad \mu_{\underset{\max}{\text{Practical}}} = 1.3\mu_{\max} \text{ , when } \sigma = 10 \tag{9.6}$$

The following figures 9.19 - 9.22 illustrates the performances when using $\sigma = 10$: 1) the theoretical maximum convergence coefficient $\mu = 1/[N\lambda_{\max}(\boldsymbol{R}) + 2\sigma N]$; 2) the maximum practical convergence coefficient $\mu = 1.3/[N\lambda_{\max}(\boldsymbol{R}) + 2\sigma N]$; 3) the convergence value starts to go unstable $\mu = 1.4/[N\lambda_{\max}(\boldsymbol{R}) + 2\sigma N]$; 4) the magnitude response of the filter.

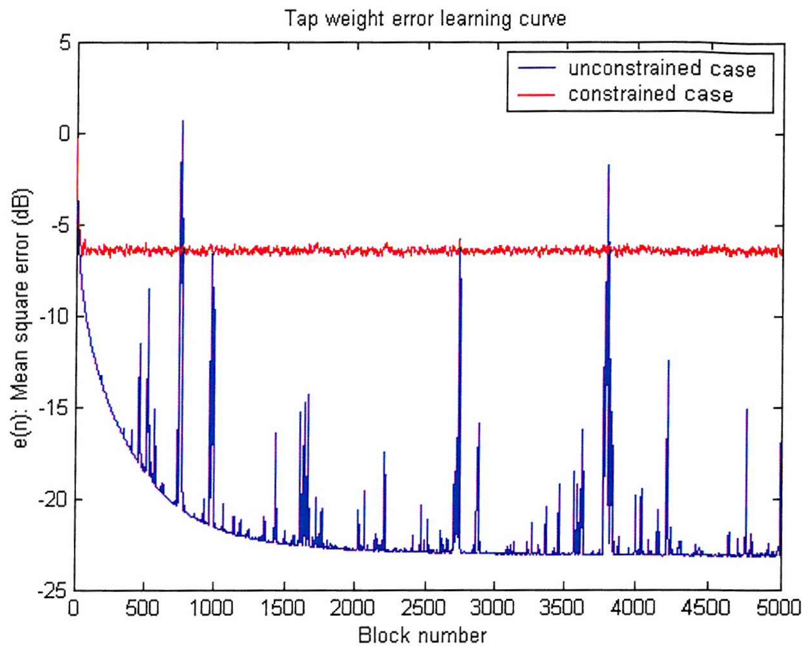


Figure 9.19: Mean square error estimate, with $\sigma = 10$, maximum theoretical convergence coefficient $\mu_{\max}$ used for constrained case (red) is $\mu_{\max} = 1/(N\lambda_{\max}(\boldsymbol{R}) + 2\sigma N)$ = 3.3058e-005, and $\mu = 0.6/N\lambda_{\max}(\boldsymbol{R})$ = 6.1415e-005 is used for unconstrained case (blue).

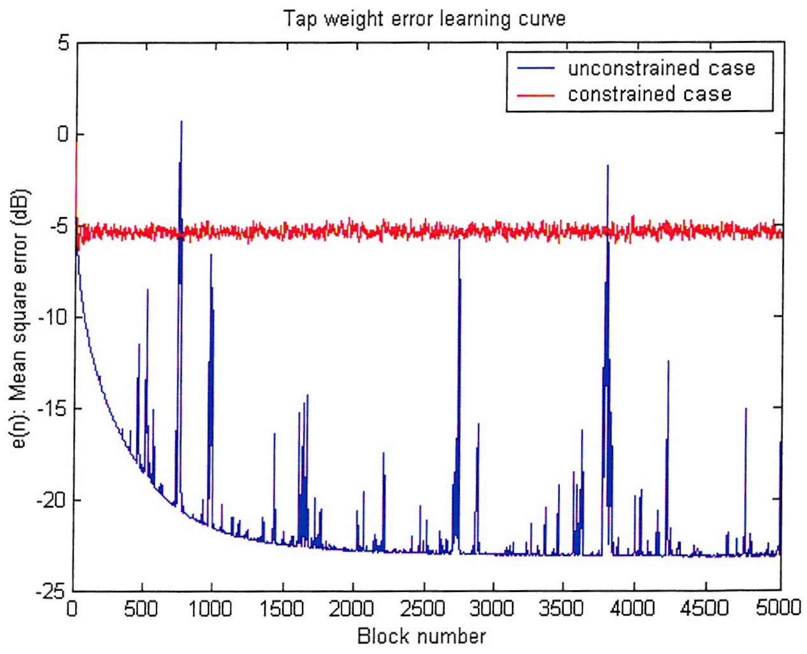


Figure 9.20: Mean square error estimate, with $\sigma = 10$, maximum practical convergence coefficient used is $\mu = 1.3/[N\lambda_{\max}(\boldsymbol{R}) + 2\sigma N]$ = 4.2976e-005 for constrained case (red), and $\mu = 0.6/N\lambda_{\max}(\boldsymbol{R})$ is used for unconstrained case (blue).

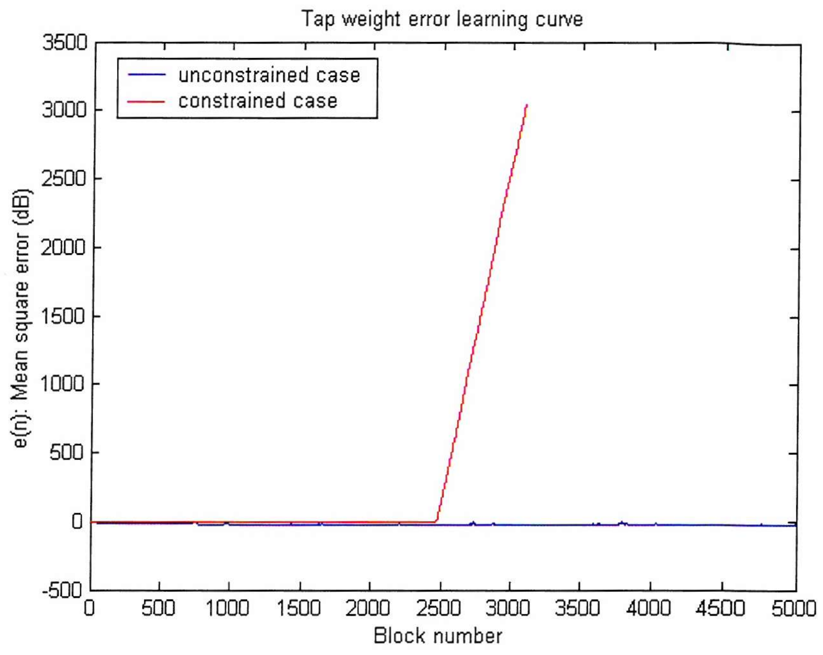


Figure 9.21: Mean square error estimate, with $\sigma = 10$, practical convergence coefficient used is $\mu = 1.4\mu_{\max} = 4.6282\text{e-}005$, for constrained case (red), and $\mu = 0.6 / N\lambda_{\max}(\boldsymbol{R})$ is used for unconstrained case (blue).

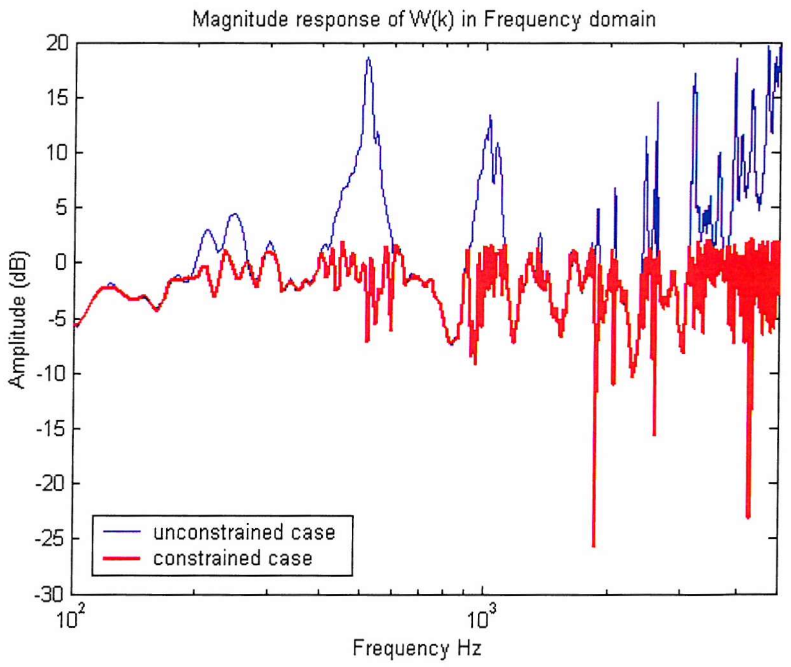


Figure 9.22: Set $\sigma = 10$. Magnitude response of equalisation filter after convergence corresponding to the FDLMS (blue), and the FDLMS with constraint on its magnitude (red) by using maximum practice convergence coefficient $\mu = 1.3\mu_{\max}$ with $\sigma = 10$.

In this simulation section, the validity of the maximum theoretical and practical convergence coefficients has been tested. The value of σ can be actually chosen from the sufficient bound condition, as equation (9.8) below. We choose σ large enough to maintain the constraint but not too large so that it will not cause instability. From cases 1, 2, 3 and 4 we have shown that the maximum practical convergence coefficients are:

$$
\begin{aligned}
&\text{Case 1:} \quad && \mu_{\text{Practical}} = 0.6\mu_{\max} \quad && \text{with } \sigma = 0 \\
&\text{Case 2:} \quad && \mu_{\text{Practical}} = 0.65\mu_{\max} \quad && \text{with } \sigma = 0.01 \\
&\text{Case 3:} \quad && \mu_{\text{Practical}} = 0.8\mu_{\max} \quad && \text{with } \sigma = 1 \\
&\text{Case 4:} \quad && \mu_{\text{Practical}} = 1.3\mu_{\max} \quad && \text{with } \sigma = 10
\end{aligned}
\tag{9.7}
$$

Refer to table 9.1 which summarise the results of (9.7). From the simulation studies, and therefore in practice, a sufficient practical condition for the convergence coefficient is defined as:

$$\mu \leq \frac{0.6}{N\lambda_{\max}(\boldsymbol{R}) + 2\sigma N} \tag{9.8}$$

The theoretical analysis of the penalty parameter σ and convergence parameter μ, for BLMS and FDLMS algorithms with constraint $|\boldsymbol{W}(k)|^2 < L(k)$ and first order penalty function, have been set out and established. We have then investigated the validity of the theoretical convergence coefficient bounds of equation (7.62) by comparing it with the practical convergence coefficient bounds which obtained from simulation tests. The results show that the theoretical convergence coefficient is not far off the practical one. This means that the bounds enable us to select μ and σ accurately and improve performance. The new bounds give conditions for these parameters, thus establishing a new framework within which more efficient values of the parameters can be chosen, avoiding trial and error as in past methods. The framework can enable an efficient design and can be applied to a wide rage of applications.

9.4 Simulations with the constrained frequency domain LMS algorithm

In this section, the formulation of the constrained FDLMS algorithm with first order penalty function developed in chapter 6 is used for the adaptive sound equalisation simulation. The theoretical convergence coefficient bound in equation (9.8) will be used to obtain the value of μ for the constrained FDLMS algorithm in this simulation, since the μ in equation (9.8) was implemented by using FDLMS.

The constrained FDLMS algorithm is now applied to the adaptive filtering equalisation system, as illustrated in figure 9.2. The equalisation at the microphone location is estimated by measuring the frequency response G_1 between the microphone and a loudspeaker in an enclosure. The measured response of the acoustic path G_1 is taken from the paper of Rafaely and Elliot (2000b) as used throughout this chapter. The simulation is performed in MATLAB, using a sampling frequency of 10 kHz, a block size of 1024 samples, a filter length $N = 1024$, with 3000 iterations and using the magnitude constraint $|\boldsymbol{W}(k)|^2 < 2\text{dB}$. The convergence coefficient μ used in this simulation is $\mu = 0.6/\ [N\lambda_{\max}(\boldsymbol{R}) + 2\sigma N]$ for both constrained and unconstrained cases, and $\sigma = 0.1$ is chosen.

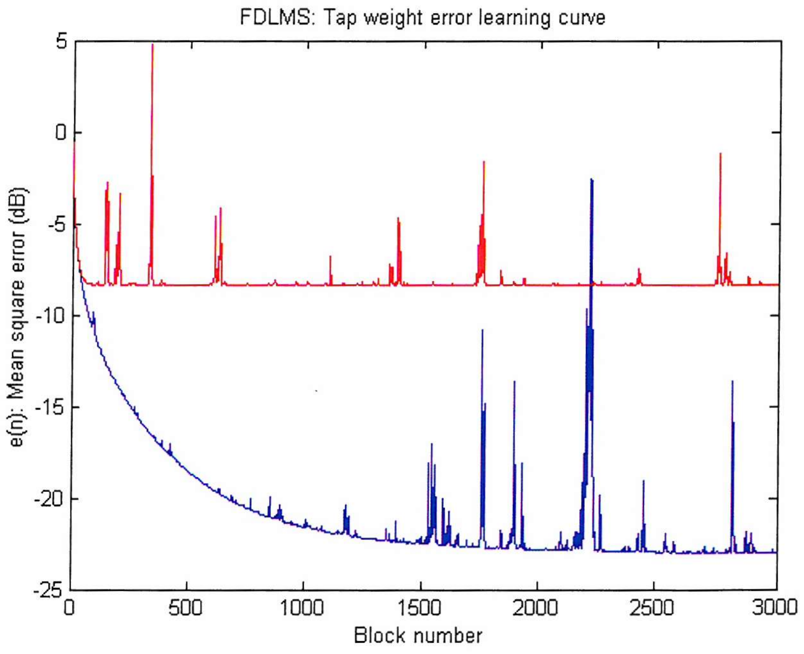


Figure 9.23: The performance of the convergence characteristics between constrained FDLMS (red) and unconstrained FDLMS (blue).

Figure 9.23 shows the convergence performance of the constrained and unconstrained FDLMS algorithm using theoretical convergence coefficient $\mu = 0.6\mu_{\max}$ and $\sigma = 0.1$. It presents the block-averaged error as a function of the block number for the duration of the adaptation for both cases. Figure 9.24 shows the magnitude responses after convergence, illustrating the effect of the quadratic constraint, $|W|^2 < 2$dB. The convergence coefficient μ used above is the maximum μ, although a smaller μ can be used to reduce the noise in figure 9.23, but at the expense of reducing the convergence speed. We can see from figure 9.23 that for the unconstrained case, the FDLMS algorithm did not reach the minimum until 2500 blocks iteration even though the maximum μ is used. With constraints, the algorithm converged after fewer iterations but produces larger equalisation errors. This is illustrated in figure 9.25. It can also be seen that without any constraints, the FDLMS algorithm gives good equalisation.

The magnitude response of the acoustic path G_1 (green) is shown in figure 9.25, and when equalized (WG_1) with the FDLMS algorithm (blue), and with constrained FDLMS algorithm on the filter magnitude (red). Figure 9.25 shows, in the unconstrained case, that the FDLMS algorithm compensates for almost all notches and peaks in the magnitude, and hence produce a good equalized response. The constrained FDLMS cannot compensate for all the notches; this would require a filter response with a larger magnitude. However, good equalisation is achieved at most frequencies with constraints.

To observe the quality of the equalisation away from the microphone position, the frequency response between the microphone and the loudspeaker were measured at a position 10 cm away from the equalisation microphone [Rafaely and Elliot, 2000b], denoted by G_2. The plot of the magnitude responses of this point is shown in figure 9.26; these responses are clearly different from those of figure 9.25. This is because the adaptive filter has been designed to equalise magnitude responses at G_1; the equalisation at G_2 will therefore not produce sound equalisation as good as that at G_1.

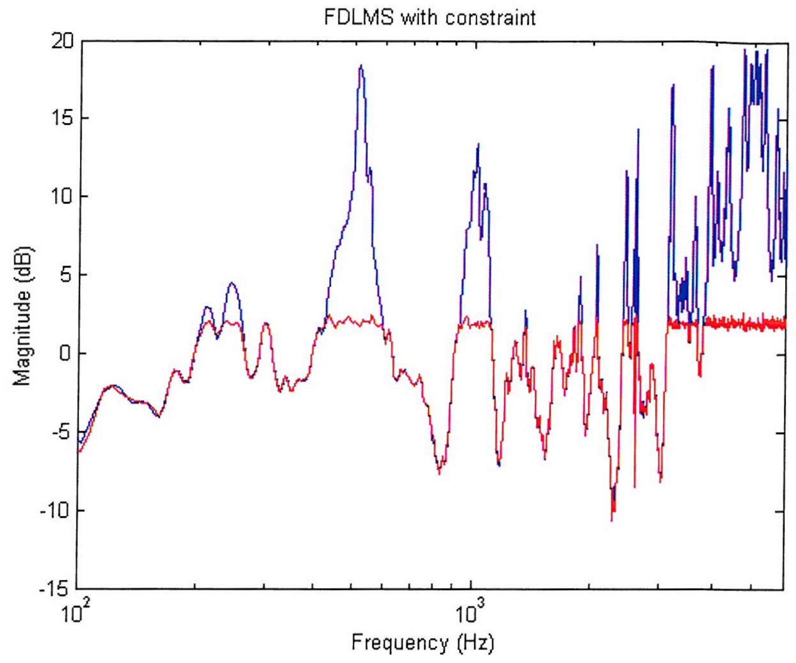


Figure 9.24: Magnitude responses after convergence corresponding to the FDLMS and FDLMS with constraint $|\boldsymbol{W}(k)|^2 < 2\text{dB}$.

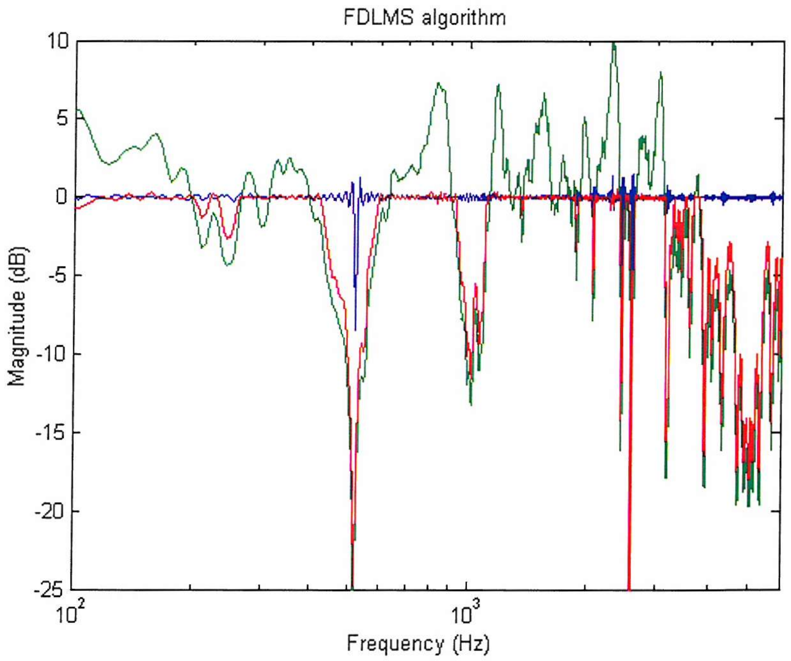


Figure 9.25: Magnitude response of unequalised path G_1 from loudspeaker to microphone (green), and equalized response, $|WG_1|$ using the FDLMS (blue), and the FDLMS with magnitude constraint at 2dB (red).

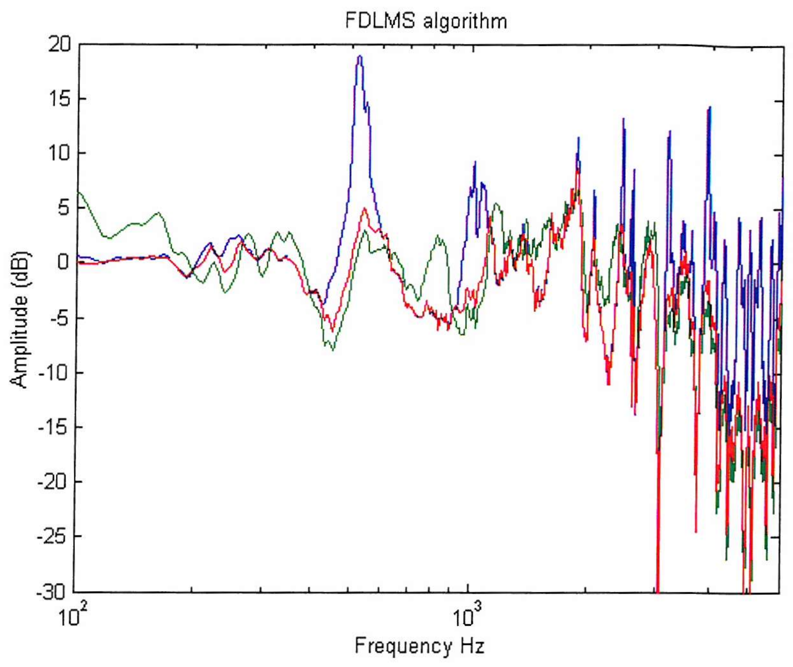


Figure 9.26: Magnitude response of unequalised path G_2 from loudspeaker to location 10cm from the equalization microphone (green), and corresponding equalized response, $|WG_2|$ using the FDLMS (blue) and the FDLMS with magnitude constraint (red), where both are designed to equalize the response to microphone G_1.

Figure 9.26 shows that no extreme peaks are observed, but that the equalisation with the constrained FDLMS result in equalisation of a poorer quality than at G_1. On the other hand, due to the amplification produced by the unconstrained equalisation filter W, a peak of about 20 dB is observed at around 500 Hz. Further peaks occur at higher frequencies. These peaks will mean that sound quality away from the equalisation microphone will be poor. Figure 9.26 demonstrates that satisfactory equalisation away from the microphone can be successfully achieved by adding a frequency-domain magnitude limiting constraint. This is in contrast to unconstrained algorithms that can allow magnitude peaks, which are undesirable.

The constrained FDLMS algorithm with quadratic constraint and first order penalty function, and with theoretical convergence coefficient bound on μ has been simulated in an adaptive sound equalisation system. In the next two sections, we will consider the constrained FDCGA and constrained BN-FDLMS algorithms and then compare the performance with the FDLMS algorithm.

9.5 Simulations with the constrained frequency domain conjugate gradient algorithm

The constrained frequency-domain block CG algorithm is now applied to the adaptive filtering equalisation system, as illustrated in figure 9.2, using the magnitude limiting constraint discussed previously. In order to compare the performance, the same measured data and setting of the sound equalisation system from section 9.4 are used in this section, except the convergence coefficient μ. The convergence coefficients μ used in this simulation are $\mu = 4\times10^{-4}$ for unconstrained case, and $\mu = 4\times10^{-6}$ for constrained case. These values are chosen by trial and error, it gave most stable convergence to the solution and near the maximum, from a range of chosen values. The performance of FDCGA algorithm will be presented for both unconstrained and constrained cases.

Figure 9.27 shows that the FDBCG algorithm has a better performance in terms of the convergence rate for both unconstrained and constrained cases compared to FDLMS algorithm in figure 9.23. It can be seen that convergence to the minimum using FDCGA was achieved in far less blocks than the FDLMS algorithm, in both constrained and unconstrained cases. For the unconstrained case, the FDCGA reached the minimum after very few blocks; approximately 20 blocks. With constraints, it reached the minimum after only approximately 10 blocks. Figure 9.28 shows the magnitude responses after convergence, illustrating the effect of the quadratic constraint, $|W|^2 < 2\text{dB}$, with FDCGA.

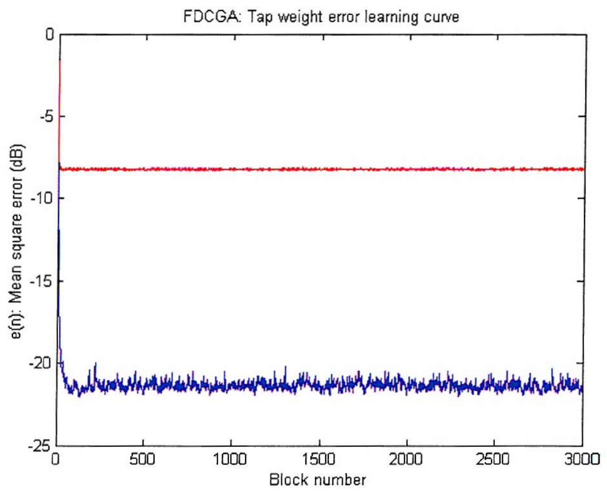


Figure 9.27: The performance of the convergence characteristics between constrained FDCGA (red) and unconstrained FDCGA (blue).

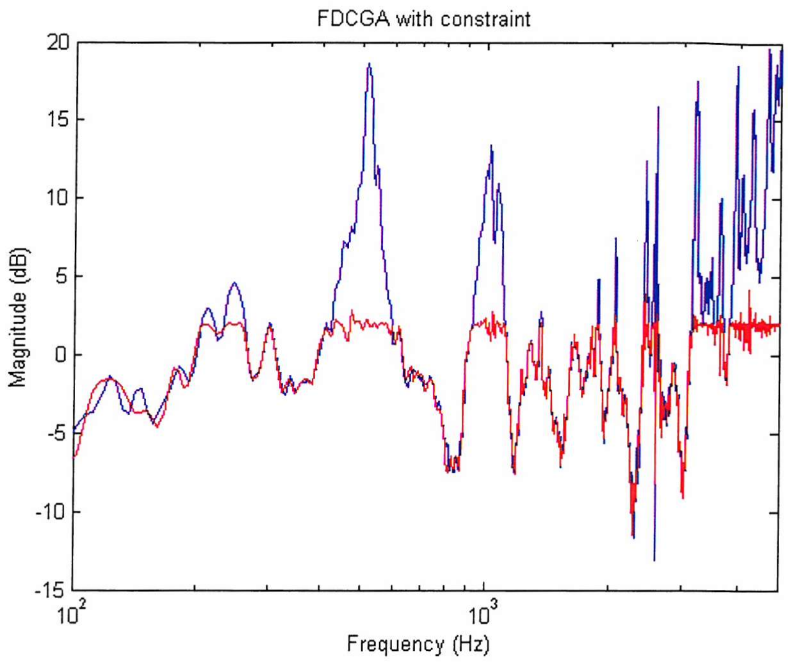


Figure 9.28: Magnitude responses after convergence corresponding to the FDCGA and FDCGA with constraint $|\boldsymbol{W}(k)|^2 < 2\text{dB}$.

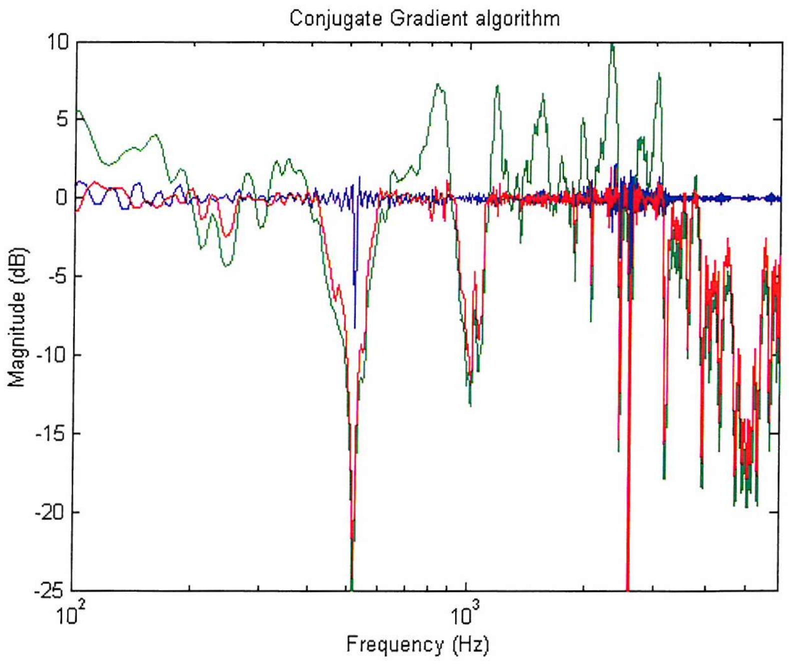


Figure 9.29: Magnitude response of unequalised path G_1 from loudspeaker to microphone (green), and equalized response, $|WG_1|$ using the FDCGA (blue), and the FDCGA with magnitude constraint at 2dB (red).

The magnitude response of the acoustic path G_1 (green) is shown in figure 9.29, and when equalized $|WG_1|$ with the FDCGA (blue), and with constrained FDCGA on the filter magnitude (red). Figures 9.25 and 9.29 show, in the unconstrained case, that both FDLMS and FDCGA compensate for almost all notches and peaks in the magnitude, and hence produce a good equalized response. The FDCGA produces a similar equalisation to the FDLMS. However the convergence coefficient μ is chosen by trial and error, with a smaller μ used here in order to reduce the noise and fluctuation that is shown in figure 9.27 and 9.28. Although the constrained FDCGA and FDLMS cannot compensate for all the notches, good equalisation is achieved at most frequencies with constraints in both algorithms. We can also see from figures 9.25 and 9.29 that without any constraints, both the FDLMS and FDCGA give a good equalisation.

Figure 9.30 shows the quality of the equalisation away from the microphone position. The frequency response between the microphone and a loudspeaker were measured at a position 10 cm away from the equalisation microphone as in section 9.4. Figure 9.30 shows that equalisation at G_2 cannot produce sound equalisation as good as that at G_1. However, it demonstrates that satisfactory equalisation away from the microphone can be successfully achieved by adding a magnitude limiting constraint.

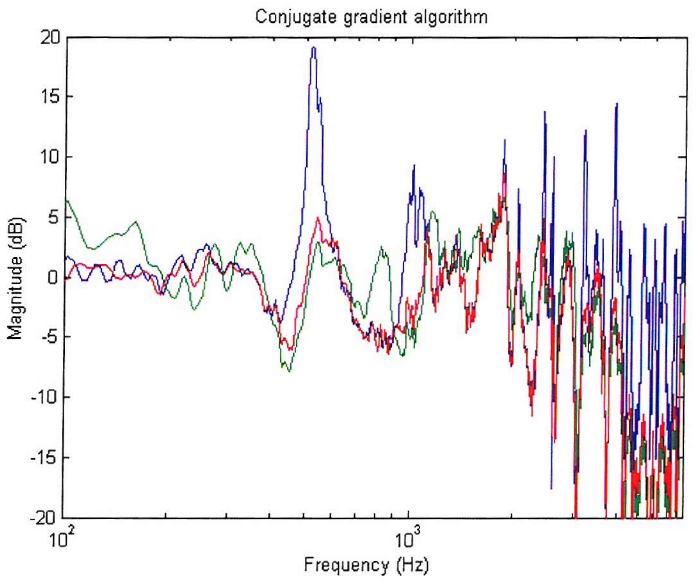


Figure 9.30: Magnitude response of unequalised path G_2 from loudspeaker to location 10cm from the equalization microphone (green), and corresponding equalized response, $|WG_2|$ using the FDCGA (blue) and the FDCGA with magnitude constraint (red), where both are designed to equalize the response to microphone G_1.

9.6 Simulations with the constrained frequency domain bin-normalised LMS algorithm

In this section, the formulation of the constrained BN-FDLMS algorithm presented in chapter 6 is used for the adaptive sound equalisation simulation. The same measured data and setting of the system from previous is used, except the convergence coefficient μ. Here, the convergence coefficients $\mu = 0.03$ is used for both unconstrained and constrained cases in $\mu_{\max}(k)$, where $\mu_{\max}(k) = \dfrac{\mu}{\hat{\mathrm{P}}_m(k)}$ (see chapter 7 for details). Again, the value of μ is chosen by trial and error and it gave most stable convergence to the solution. The performance of the unconstrained and constrained BN-FDLMS algorithm is presented as follows.

Figure 9.31 shows that the BN-FDLMS algorithm has a better convergence rate than FDLMS algorithm in the unconstrained case, and has a similar convergence rate in the constrained case. It can be seen that convergence to the minimum using BN-FDLMS is achieved in far less blocks than the FDLMS algorithm, in the case of the unconstrained algorithm. For the unconstrained case, the BN-FDLMS reached the minimum after only approximately 150 blocks. With constraints, it reached the minimum after approximately 100 blocks, which has a similar performance as FDLMS.

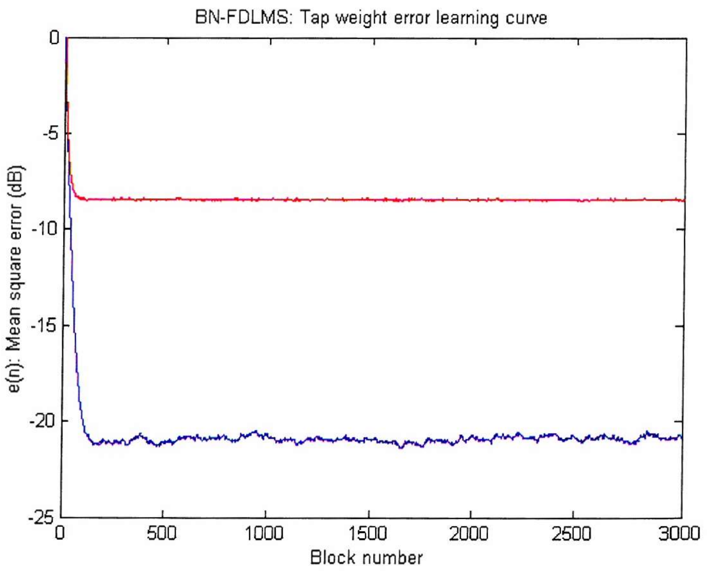


Figure 9.31: The performance of the convergence characteristics between constrained BN-FDLMS (red) and unconstrained BN-FDLMS (blue).

The following figure 9.32 shows the magnitude responses after convergence, illustrating the effect of the quadratic constraint, $|\boldsymbol{W}|^2 < 2\text{dB}$, with BN-FDLMS.

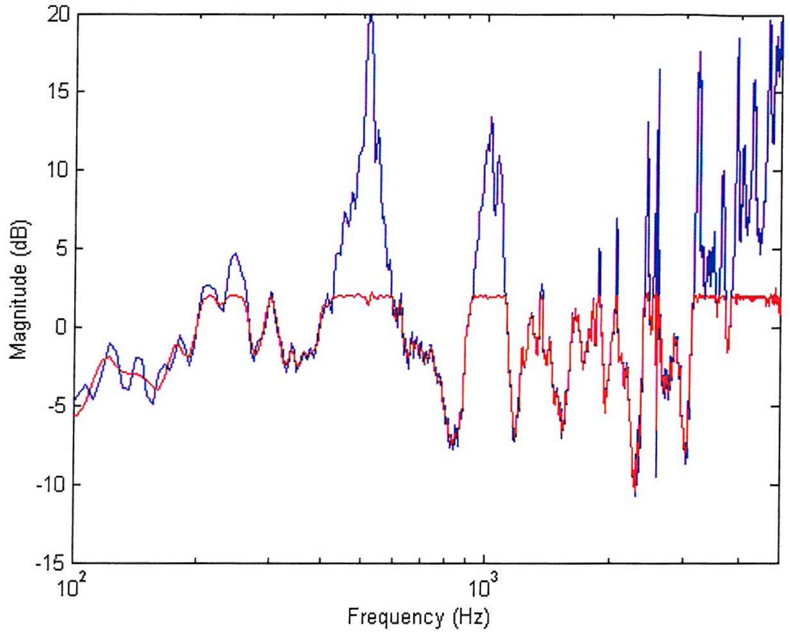


Figure 9.32: Magnitude responses after convergence corresponding to the BN-FDLMS and BN-FDLMS with constraint $|\boldsymbol{W}(k)|^2 < 2\text{dB}$.

The magnitude response of the acoustic path G_1 (green) is shown in figure 9.33, when equalized $|WG_1|$ with the BN-FDLMS algorithm (blue), and with constrained BN-FDLMS algorithm on the filter magnitude (red). Similar to the result of FDLMS and FDCGA, in the unconstrained case, the algorithm compensates for almost all notches and peaks in the magnitude.

Again, to observe the quality of the equalisation away from the microphone position, the frequency response between the microphone and a loudspeaker were measured at a position 10 cm away, as with the previous two sections. The plot of the magnitude responses of this point is shown in figure 9.34. Figure 9.34 shows that equalisation at G_1 produced better sound equalisation than that at G_2. By adding a magnitude limiting constraint to G_2, satisfactory equalisation away from the microphone can be successfully achieved.

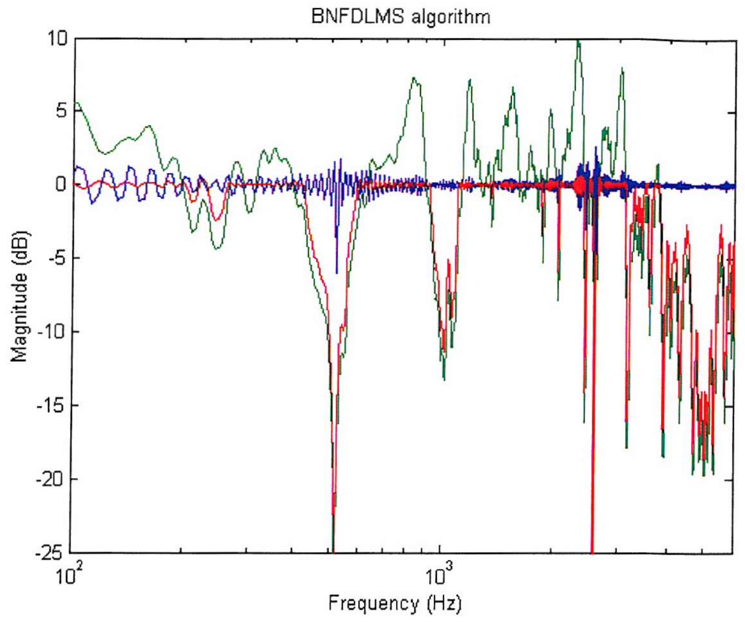


Figure 9.33: Magnitude response of unequalised path G_1 from loudspeaker to microphone (green), and equalized response, $|WG_1|$ using the BN-FDLMS (blue), and the BN-FDLMS with magnitude constraint at 2dB (red).

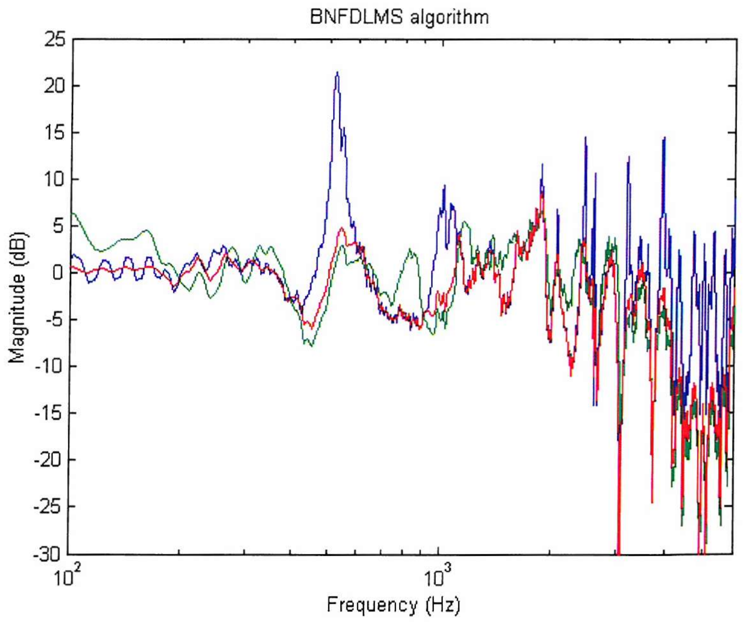


Figure 9.34: Magnitude response of unequalised path G_2 from loudspeaker to location 10cm from the equalization microphone (green), and corresponding equalized response, $|WG_2|$ using the BN-FDLMS (blue) and the BN-FDLMS with magnitude constraint (red), where both are designed to equalize the response to microphone G_1.

9.7 Conclusions

In section 9.3, the theoretical convergence coefficient μ, for FDLMS algorithms with constraint $|W(k)|^2 < L(k)$, has been studied and we have investigated the validity of the theoretical convergence coefficient bound by comparing it with practical convergence coefficients obtained from simulation tests. The results show that the theoretical convergence coefficient is not far off the practical one. This means that the bound enables accurate design for constrained problems without any "trial and error". The convergence coefficient μ for other algorithms could also be developed in a similar manner to the FDLMS algorithm. The bound on μ can also be developed for different constraints (depending on the application) as long as the constraints are quadratic functions. Furthermore it can also be applied to other algorithms, for example to the BN-FDLMS algorithm. By taking the simple case of the algorithm ($\beta \approx 1$) when the analysis is like the FDLMS, we can apply the constraints analysis, so that the convergence coefficient μ defined for the FDLMS can be suitable for the BN-FDLMS as well. However for the case of $\beta \neq 1$, the convergence analysis of the BN-FDLMS algorithm will be more complicated than FDLMS algorithm [Cowan, 1985, Janssen, 1987 and Sommen et al, 1987]. The further investigation of the convergence analysis of the BN-FDLMS algorithm is suggested for future work.

In sections 9.4, 9.5 and 9.6, the constrained FDLMS, FDCGA and BN-FDLMS algorithms with quadratic constraint have been implemented in an adaptive sound equalisation system. It shows that increasing the convergence rate of the widely used FDLMS algorithm can be done by incorporating the FDCGA and BN-FDLMS algorithms. The performance shows that the FDCGA has superior convergence rate for both unconstrained and constrained cases compared with the algorithms of BN-FDLMS and FDLMS. The results also show that the BN-FDLMS algorithm has better convergence performance than the FDLMS algorithm for the case of unconstrained, and has similar convergence rate for constrained case. Figure 9.35 shows the comparison of the performance of the convergence characteristics between FDLMS, FDCGA and BN-FDLMS algorithms in an adaptive sound equalisation system.

These sections also showed that applying constraints could improve the performance of equalisation around the microphone in an enclosure. In the next chapter, we will discuss how to increase the spatial robustness by using an appropriate frequency-dependent constraint, instead of a constant constraint, in sound equalisation in an enclosed room. Chapter 10 shows that spatial robustness can be improved by using frequency-dependent constraints, by utilizing the structure of the spatial correlation in the reverberant sound field.

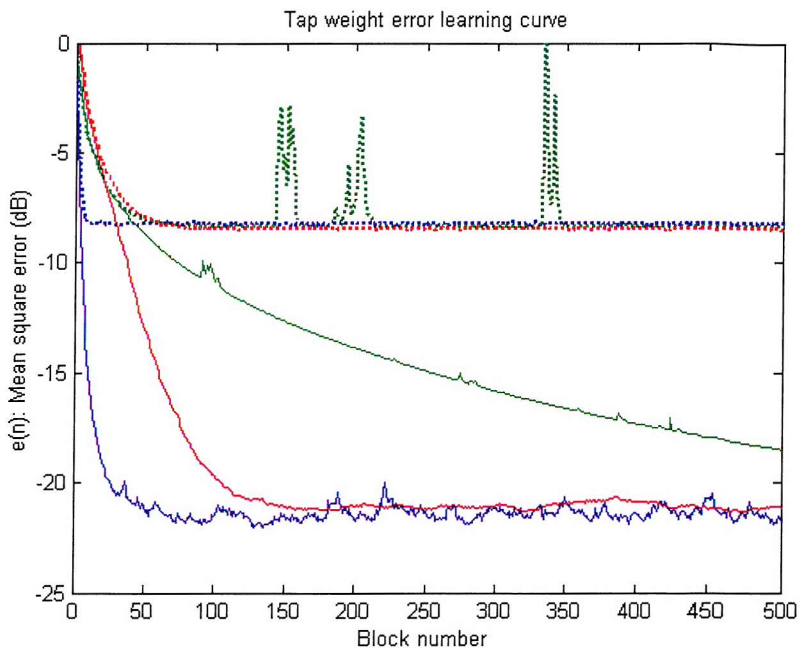


Figure 9.35: Comparison of the performance of the convergence characteristics between FDLMS, FDCGA and BN-FDLMS, with unconstrained and constrained in an adaptive sound equalisation system. FDLMS (green), FDCGA (blue) and BN-FDLMS (red). Dotted and solid lines represent constrained and unconstrained cases, respectively.

Chapter 10

Spatially robust adaptive sound equalisation by constrained adaptive filtering

10.1 Introduction

In this chapter, spatially robust performance will be demonstrated by using various magnitude limiting frequency dependency constraints in the simulation; since low frequency sound in a room is more spatially correlated, higher gains in the equalisation filter are allowed at low frequencies, since we expect the notches not to vary significantly over space. The sinc function which represents the spatial correlation of the sound in a reverberant room, is used in the design of the magnitude constraints, such that a tight limit is imposed at high frequencies where the sound is relatively uncorrelated, and a more relaxed limit is imposed at low frequencies where the sound is more correlated.

10.2 Implementation of frequency-dependent constraints

In the previous chapter, we showed that in sound equalisation applications, although good equalisation can be achieved around the microphone point, the equalisation at other points may be poor since the frequency response of the acoustic path can vary significantly from point to point. The equalised system may therefore suffer from distorted sound at high frequencies, away from the microphone point. The effects of this distortion can be counteracted to some degree by applying a gain limit to the adaptive filter. Improved spatial robustness could potentially be achieved by using frequency-dependent constraints, by utilizing the structure of the spatial correlation in reverberant sound fields. The spatial robustness performance will be demonstrated by using magnitude limiting frequency-dependent constraint in the simulation; since low frequency sounds in a room are more spatially correlated, higher gains in the equalisation filter are allowed at low frequencies.

When comparing a point near the microphone in a reverberation sound field to the microphone position, Cook et la (1955) showed that the average spatial correlation function is given by the sinc function. The function sinc (kr) is equivalent to sin $(kr)/kr$, where $k = 2\pi/\lambda$.

In this formulation, r is the distance between the microphone point and the selected point, and λ is the wavelength. λ is calculated by dividing the speed of propagation c by the frequency of vibration f. Since the sinc function is based on the spatial correlation of two points in a reverberation room, using the sinc function to design a frequency-dependent constraint has the potential to improve the spatial robustness of the equalisation performed.

A practical frequency-domain constraint was introduced in chapter 5; the constraint introduced limit the magnitude of the signal, $|\boldsymbol{W}(k)|^2 < L$. This is useful to avoid excess amplification of the filter at given frequencies. A limit L is placed on the magnitude of the filter $\boldsymbol{W}$ for any given frequency k, as described in previous chapters. L can take the form of a scalar or a function. In this chapter, the structure of a sinc function is used to motivate the design of an appropriate constraint for the sound equalisation in an enclosed room. Since the sinc function represents the spatial correlation away from the microphone, low frequencies are more correlated. This means that at low frequencies, the frequency response away from the microphones is unlikely to change significantly, as long as kr is less then π. While at high frequencies, the response is more likely to change, when kr is larger then π. To improve spatial robustness, we therefore limit the gain on $\boldsymbol{W}$ to avoid peaks in the equalised response near the microphone. This limit will be tighter at high frequencies, where the response is more likely to change. Therefore, low and high frequency gain limits are defined, with the roll-off between the behaving as the structure of sinc function. Therefore we use sinc function as a motivator, then the limit L designed is chosen and illustrated as figure 10.1.

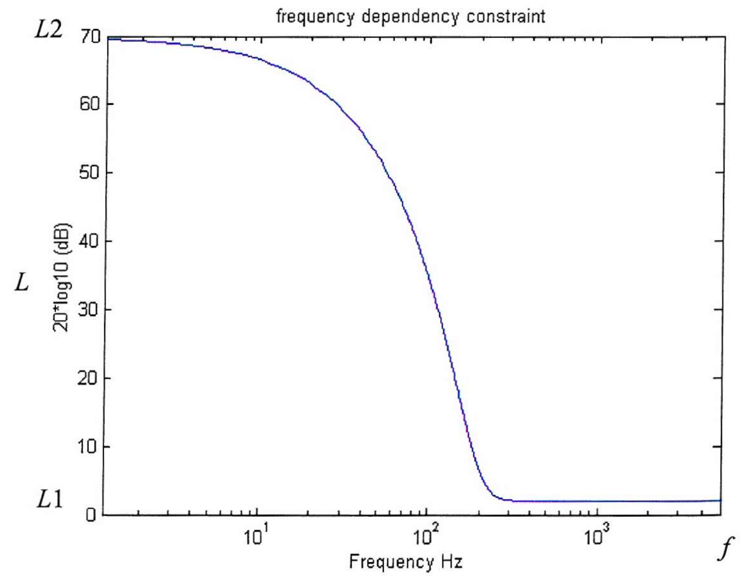


Figure 10.1 The frequency-dependent constraint with exponential function shape.

The frequency-dependent constraint chosen here is based on the exponential function, and is given as:

$$L\,(f, L1, L2) = (L2 - L1)\exp(-f) + L1 \qquad (10.1)$$

The function of (10.1) is depended on the valuables of f, $L1$ and $L2$. Such that as $f \rightarrow \infty$, $20\log10(L) \rightarrow L1$, and $f \rightarrow 0$, $20\log10(L) \rightarrow L2$. In equation (10.1), the L is the constraint function, f is an integer and $L1$, $L2$ are the lower and upper bounds of the function respectively. This frequency-dependent constraint illustrated in figure 10.1 gives us a more relaxed limit at low frequencies and a tight limit at high frequencies. We will not apply limits on excessively low frequencies, this is because these will be in an inaudible frequency range. In the next section we will apply the limit L in equation (10.1) to the adaptive sound equalisation system to avoid excessively high peaks due to distortion of the signal.

10.3 Simulation of sound equalisation with measured enclosed data

Due to the simplicity of the FDLMS algorithm, it was chosen to investigate the performance of the limit L of (10.1) in this chapter. The limit L is now applied to the adaptive sound equalisation system, which was used in chapter 9. The measured data, acoustic plant and modelling delay data from Rafaely and Elliott (2000b) is used here as the previous chapter. The performance of the limit L in the simulation is presented and discussed as follows.

As in chapter 9, in order to investigate the limit L, we needed to observe the quality of the equalisation away from the microphone position. Because good equalisation can be achieved at the microphone point, but poor equalisation might occurs at other points, constraints can then be applied to improve the equalisation. Therefore the frequency response between the microphone and a loudspeaker were measured at a position 10cm away from the equalisation microphone, denoted by G_2. The plot of the magnitude responses of this point is shown in figure 10.2. The response G_2 are clearly different from the response at G_1.

Figure 10.2 can illustrate the potential of the frequency-dependent constraint to be a useful constraint function. We need to look at the differences between G_1 and G_2 across the frequency range. We can see that they are very similar below about 400Hz, and starts changing above 400Hz. This means that we can allow higher gains in $\boldsymbol{W}$ below about 400Hz, and should restrict the gain above about 400Hz, which is more or less what the current frequency-dependent constraint does, as shown in figure 10.1.

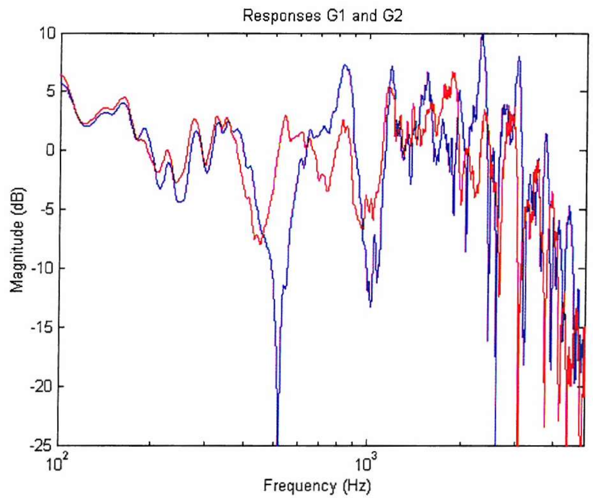


Figure 10.2: Magnitude response of unequalized path G_1 from loudspeaker to microphone (blue), and magnitude response of unequalized path G_2 from loudspeaker to location 10cm from the equalization microphone (red).

Since the adaptive filter has been designed to equalise the magnitude responses at G_1; the equalisation at G_2 will therefore not produce sound equalisation as good as that at G_1. Figure 10.3 shows good equalisation can be achieved at the microphone point. But the equalisation at other points may be poor since the frequency response of the acoustic path can vary significantly from point to point as shown in figure 10.4.

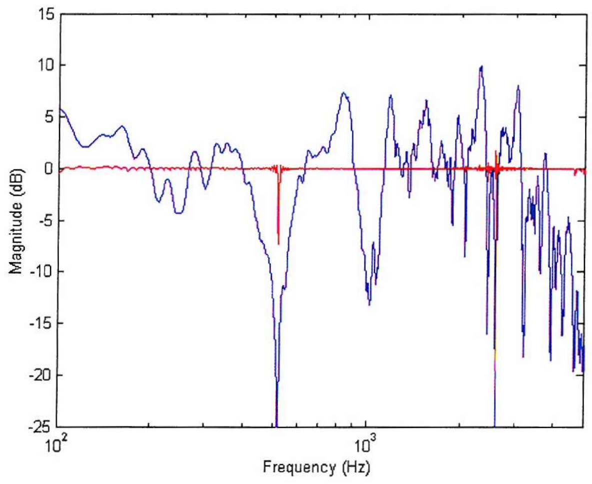


Figure 10.3: Magnitude response of unequalized path G_1 from loudspeaker to microphone (blue), and equalized response, $|WG_1|$ (red) using the FDLMS.

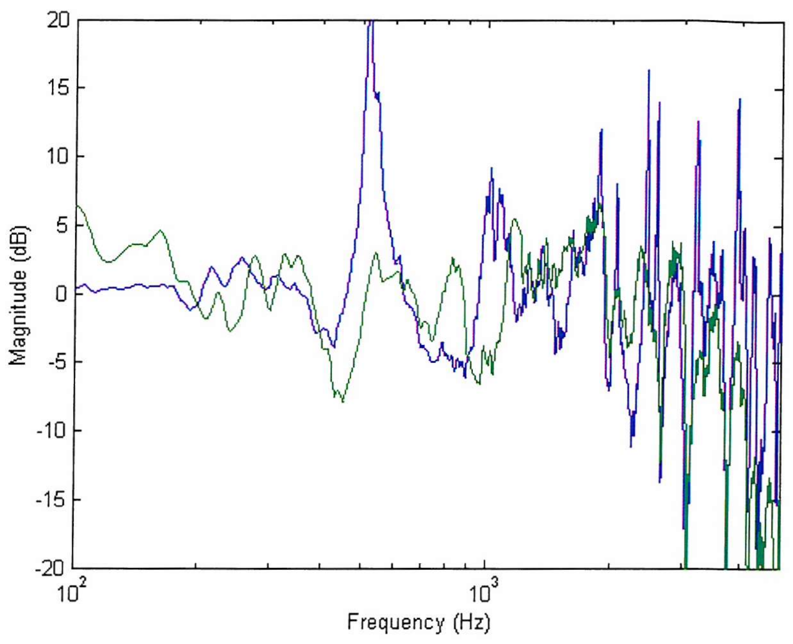


Figure 10.4: Magnitude response of unequalized path G_2 from loudspeaker to location 10cm from the equalization microphone (green), and corresponding equalized response, $|\boldsymbol{W}G_2|$ (blue) with the FDLMS.

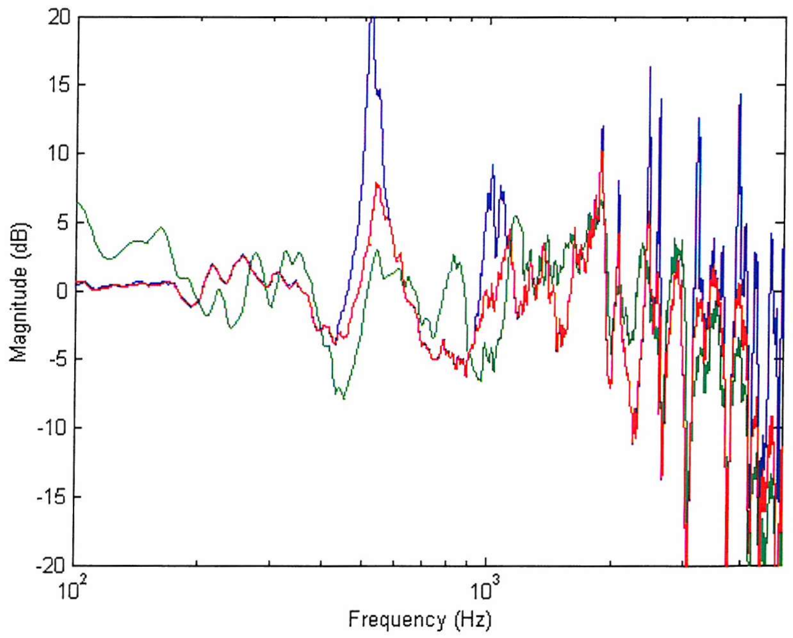


Figure 10.5: Magnitude response of unequalized path G_2 from loudspeaker to location 10cm from the equalization microphone (green), and corresponding equalized response with the unconstrained FDLMS, $|\boldsymbol{W}G_2|$, (blue) and the FDLMS with frequency-dependent magnitude constraint (red), where both are designed to equalize the response to microphone G_1.

Due to the amplification produced by the unconstrained equalisation filter W, a peak of over 20 dB is observed at around 400 Hz. Further peaks occur at higher frequencies. Figure 10.5 shows that satisfactory equalisation away from the microphone can be successfully achieved by adding a magnitude limiting constraint.

Comparing between the frequency-dependent and constant constraints, L is selected to the highest constant, which does not give rise to significant peaks (when changing from G_1 to G_2). Although the definition of what is a significant peak is also quite subjective and again could depend on the example, the peaks up to 2dB are considered reasonable in this simulation. The frequency-dependent constraint is then set such that its lowest value is the same as the constant constraint, but higher at frequencies where peaks are less likely to occur (e.g. low frequencies).

The potential of the frequency-dependent constraint may not be shown in this simulation, because the high peaks only occur after approximately 500Hz. Before 500Hz, the magnitude responses are almost less then $L1$. So, the constraints are not active until the magnitude responses are above $L1$, as shown in figure 10.6. Figure 10.6 shows the magnitude responses corresponding to the L's. Then in figure 10.7 shows the comparison of equalised responses $|WG_2|$ with the constant constraint and frequency-dependent constraint. We can see that there is no big different between constant and frequency-dependent constraint in these examples. It is because the unconstrained equalisation filter at low frequencies does not have high amplitude and so all constraints (constant and frequency-dependent constraint) are inactive anyway at this region. Therefore, this example may not be a good case to show the potential of the frequency-dependent constraint. Nevertheless, in other cases the frequency-dependent constraint could be useful, as the fact that when measuring at a location near the microphone, it is likely that the low frequency notches will not change much while the high frequency notches will change more.

In order to establish the potential of the characteristics of frequency-dependent magnitude constraint, we need to find an application when the responses have significant notches at both high and low frequencies. In the following section, one dimensional duct acoustic simulation is considered; it shows that the frequency-dependent constraints has the potential to improve the spatial robustness of equalisation around the microphone in an enclosure. The desired response of the simulation can be designed using this simplified model, i.e. notched at low frequencies that change very little when the sensor location is moved, while notches at high frequencies change more. The results from the simulations are presented and discussed.

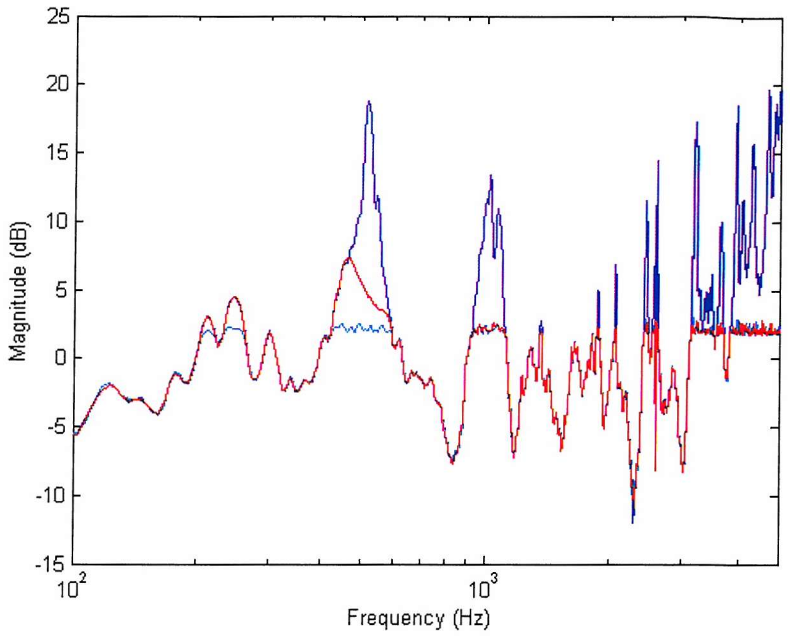


Figure 10.6: Magnitude response of equalisation filters after convergence corresponding to the FDLMS, with frequency-dependent constraint (red), and with constant constraint on its magnitude (light blue).

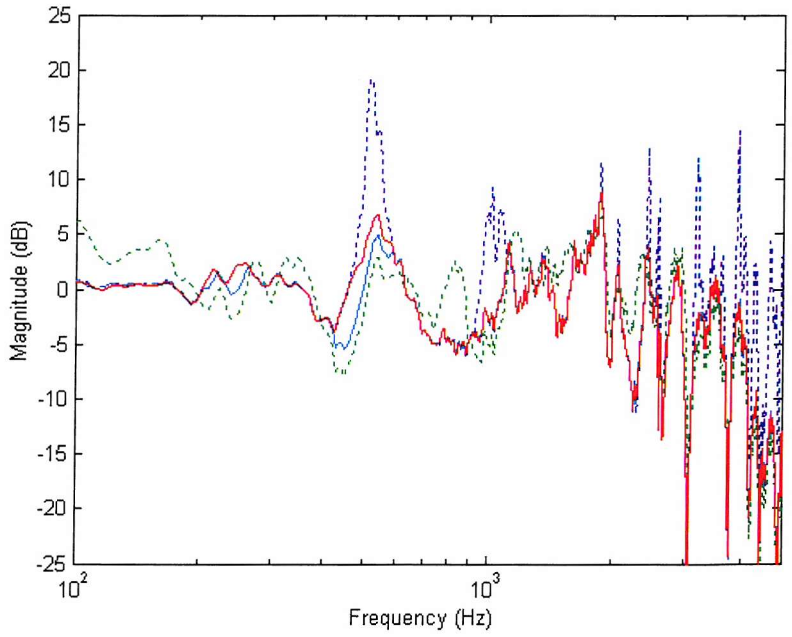


Figure 10.7: Magnitude response of equalised response $|WG_2|$ with the frequency-dependent constraint (red) and constant constraint (light blue).

10.4 Simulation of sound equalisation with modified simulated duct data

This thesis is concerned with the development and application of adaptive filtering algorithms, for problems that require constraints, such as sound equalisation. In this chapter the use of frequency-dependent constraints to improve performance is further investigated. In order to facilitate this study a suitable acoustic simulation was sought which was simple, such as a one dimensional duct, so as not to unnecessarily complicate the analysis but which is sufficiently general to represent the performance of acoustical applications.

The simulation of sound equalisation in an enclosed duct is used as a motivating model problem, to study the effects of frequency-dependent constraints. Although a one dimensional duct simulation may have different responses from a measurement in a room, it was simpler to obtain the responses from a computer model, when a careful control of the responses and the room parameters is needed.

In this simulation, as an example of the application of the frequency-dependent constraints developed in the section 10.2 we attempt to equalise the acoustic response in an enclosed one dimensional space, a long thin duct (figure 10.8). This simulated data is generated by MATLAB programming. The duct is modelled as an enclosure measuring 1 metre long and 28 acoustic modes are considered. The location of the loudspeaker is present at one end of the duct and the equalisation microphone G_1 is located 30cm away from the loudspeaker. We want to observe (1) the quality of the equalisation away from the microphone position; and (2) the potential of the frequency-dependent constraint, the frequency responses between the microphone and a loudspeaker are measured at a positions 3cm, 5cm and 8cm away from the equalisation microphone, denoted by G_2, G_3 and G_4. The performance of the frequency-dependent constraints will be compared to the constant constraints. A sampling frequency of 10kHz is used, as in the previous section. A modal damping of 0.01, a block size of 4096 samples, and thousands block iterations to allow for convergence, are used in the MATLAB simulations.

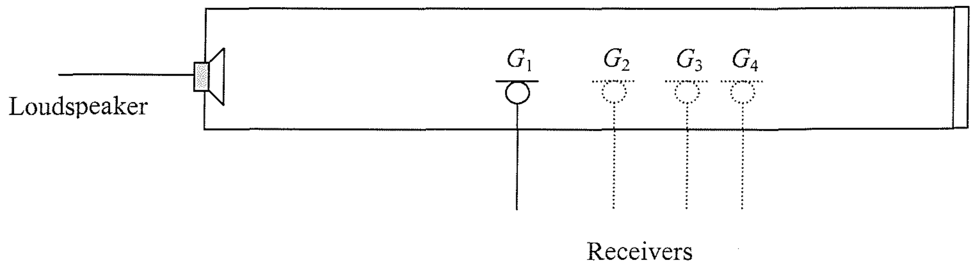


Figure 10.8: The position of the loudspeaker, equalisation microphone G_1 and the other microphones G_2, G_3, G_4 in the one dimensional duct.

In section 8.3, we have introduced briefly some elementary ideas of sound propagation in one dimensional duct that are useful in the following. The mode amplitude coefficients in equation (8.4) shows that the modal damping term $2\zeta_n\omega_n\omega$ is increasing when the mode number increased, where ω_n is defined by $\omega_n = c(n\pi/L)$. This means that the mode amplitudes will decrease, i.e. the variations in the acoustic responses when moving from microphone G_1 to microphone G_2, will not have a large magnitude, as shown in figure 10.9 which represents $\left|\frac{G_2}{G_1}\right|$. This shows that the variations have a similar magnitude response at both low and high frequencies.

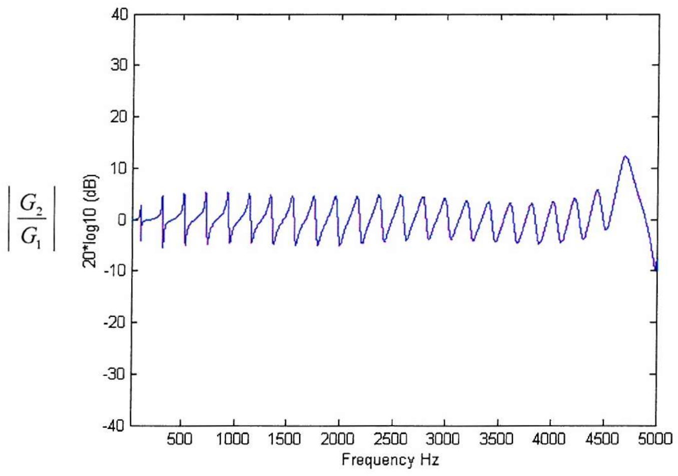


Figure 10.9: The normalised acoustic responses between the equalisation microphone G_1 and microphone position G_2.

However, in this case the frequency-dependent constraints might not be required since the variation are similar at the entire frequency range. As we mentioned previously, we want an example that have small change at low frequencies and large change at high frequencies. Figure 10.10 has the response that can provide a good example to show the potential of frequency-dependent constraints. The equalisation of a modified duct model with this example using constant damping will be presented now.

As shown in figure 10.9, we noticed that the modal damping term $2\zeta_n\omega_n\omega$ is increasing and causing a frequency independent variability in the acoustic responses. Let's now introduce a modified duct mode amplitude by changing the modal damping term into a constant. Although it is a modified duct model and therefore might be less realistic, it demonstrates that the frequency-dependent constraints have a good potential. i.e. they can be applied to an acoustic problem when a similar response happened, e.g. when damping is smaller than normal at high frequencies.

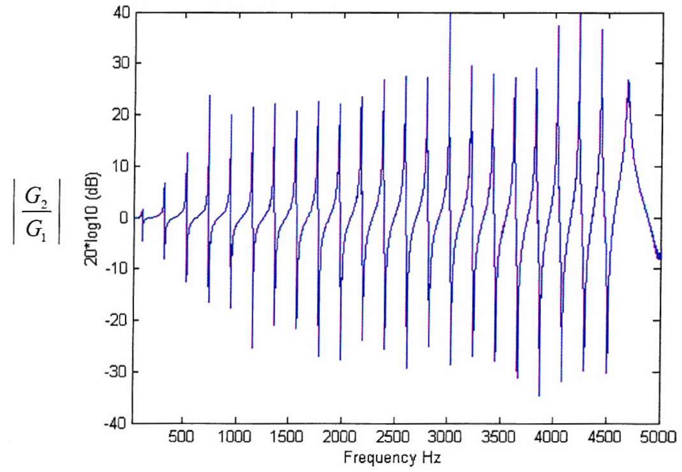


Figure 10.10: Modified duct model example: the normalised acoustic responses $\left(\left|\frac{G_2}{G_1}\right|\right)$ between the equalisation microphone G_1 and microphone position G_2.

In the simulation, the duct is modelled as an enclosure measuring 1 metre long and 28 acoustic modes are considered. The loudspeaker is present at one end of the duct and the equalisation microphone G_1 is located at 30cm away from the loudspeaker. Microphones G_2, G_3 and G_4 are located at positions 3cm, 5cm and 8cm away from the equalisation microphone. A modal damping ratio 0.01, air density 1.21 kgm^{-3}, and speed of sound 343ms^{-1} are used here. A sampling frequency of 10kHz is used as before. A block size of 4096 samples, and thousands of block iterations to allow for convergence, are used in the simulation. The frequency-dependent constraint in equation (10.1) is used here.

Figure 10.11 shows the frequency response of the equalisation microphone G_1, which is located 30cm away from the loudspeaker. Figures 10.12 – 10.14 shows the difference in the frequency responses between the equalisation microphone position G_1 and microphone positions G_2, G_3, and G_4, which are measured at a position 3cm, 5cm and 8cm away from the equalisation microphone G_1. We can see from the figures 10.12 – 10.14 that the frequency responses between G_1 and the positions away from G_1 are more correlated at low frequencies, and are relatively uncorrelated at high frequencies. Figure 10.15 (a), (b), (c) plots a close up of the high frequencies responses of figures 10.12, 10.13 and 10.14 respectively, so that we can easily see the behaviour between the responses at high frequencies.

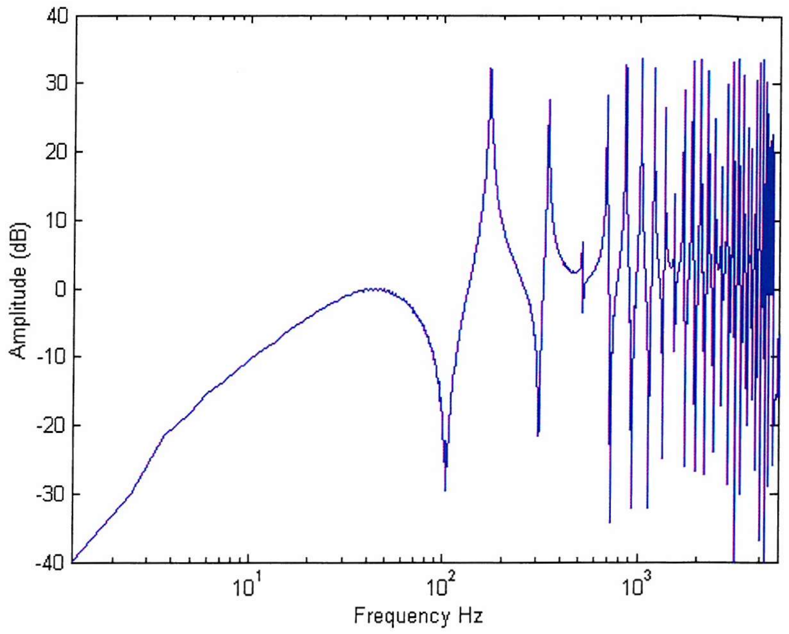


Figure 10.11: Frequency response between the equalisation microphone G_1 and a loudspeaker.

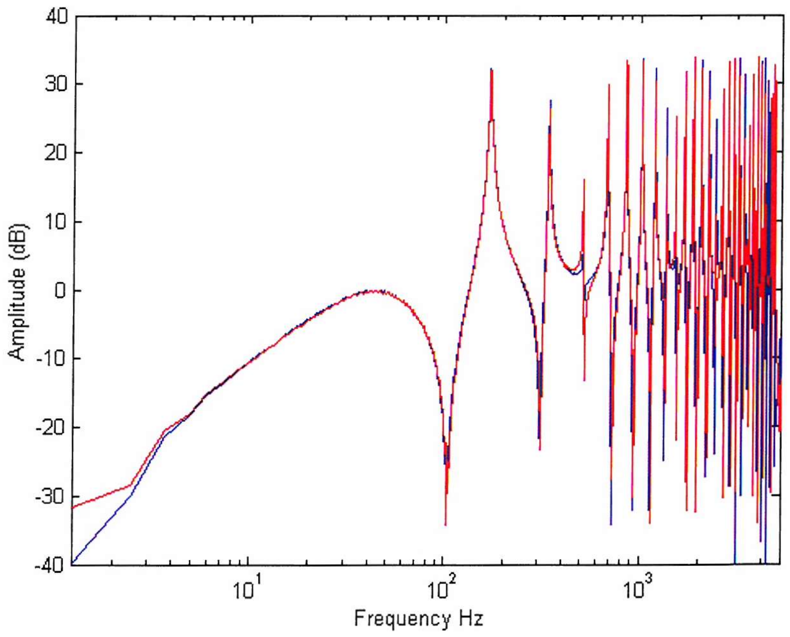


Figure 10.12: Frequency responses to the equalisation microphone G_1 (blue) and microphone position G_2 (red), which is measured at 3cm away from G_1.

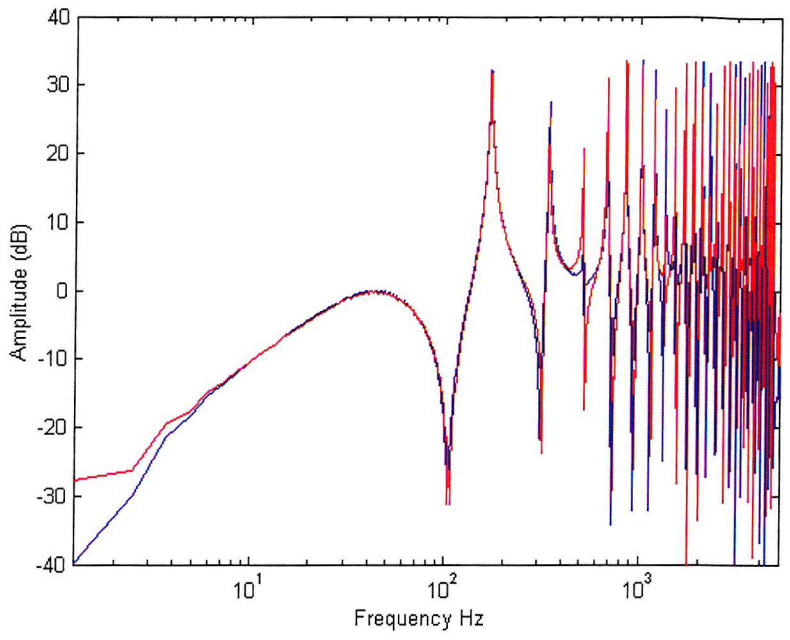


Figure 10.13: Frequency responses to the equalisation microphone G_1 (blue) and microphone position G_3 (red), which is measured at 5cm away from G_1.

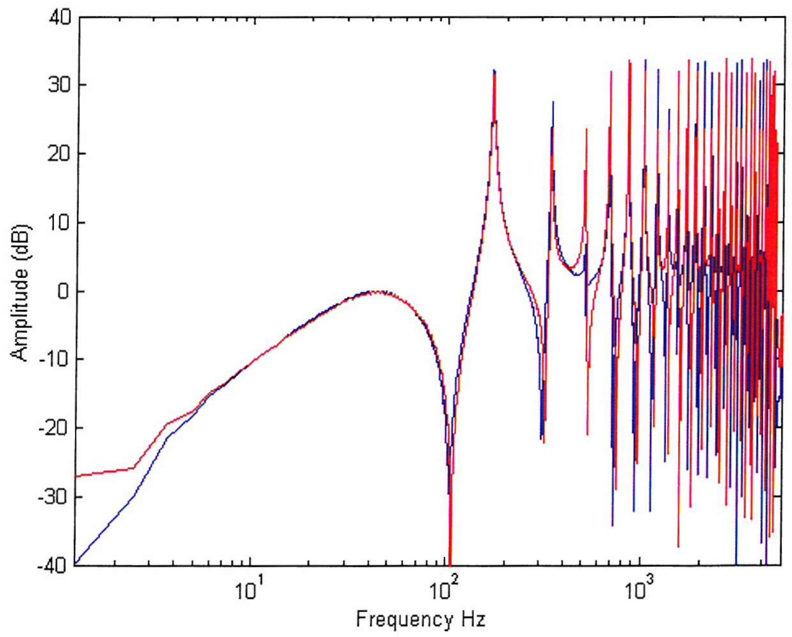
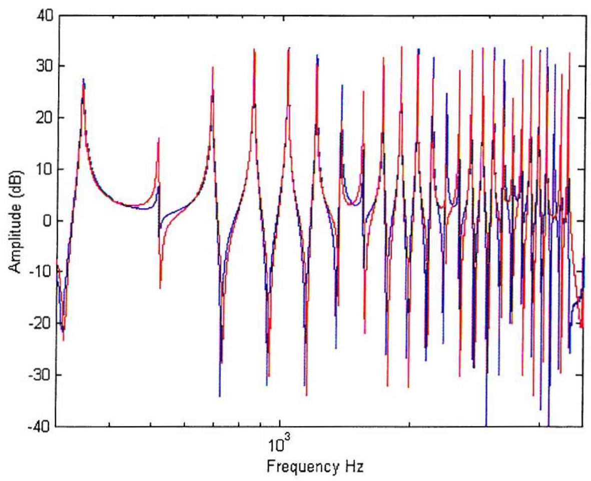
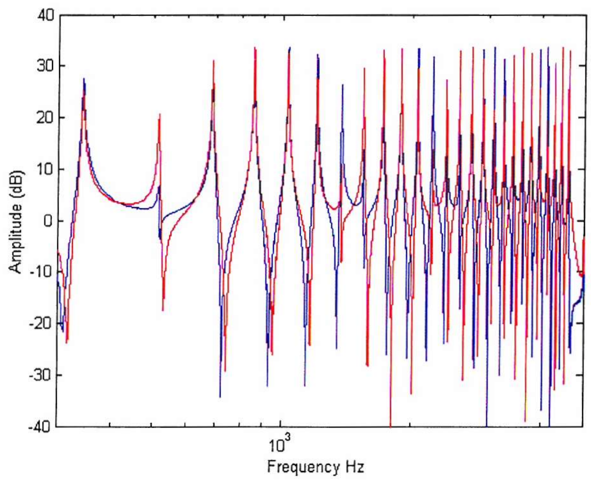


Figure 10.14: Frequency responses to the equalisation microphone G_1 (blue) and microphone position G_4 (red), which is measured at 8cm away from G_1.

(a)



(b)



(c)

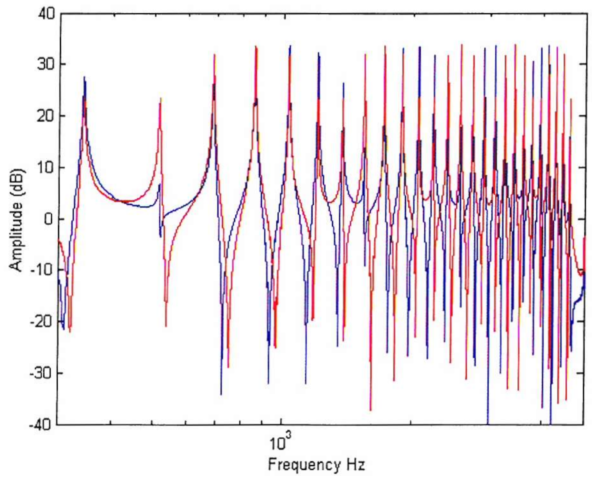


Figure 10.15: (a), (b) and (c) represents the range of frequency response between 300Hz to 5000Hz of the figures 10.12, 10.13 and 10.14, respectively.

Figure 10.16 shows the magnitude response of the unconstrained equalisation filters described in the previous sections producing large peaks, and with frequency-dependent and constant constraints having more moderate responses. The figure clearly shows how the magnitude of the filter with the constant constraint is accurately limited to 2dB and that with the frequency-dependent constraint to an exponential function at several frequency ranges.

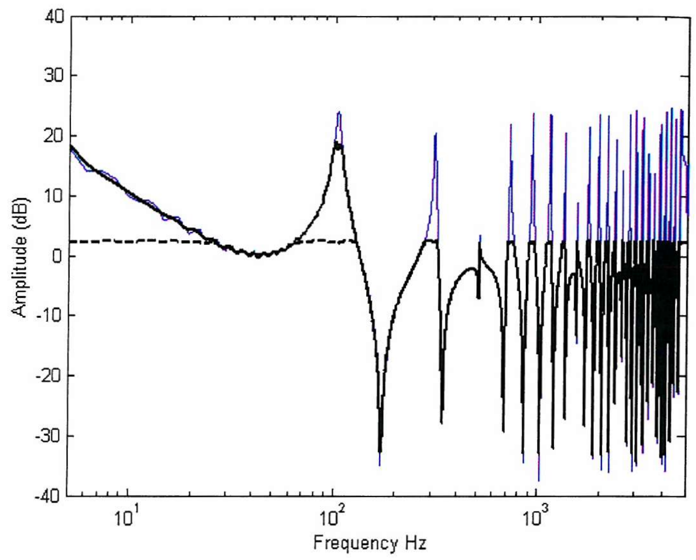


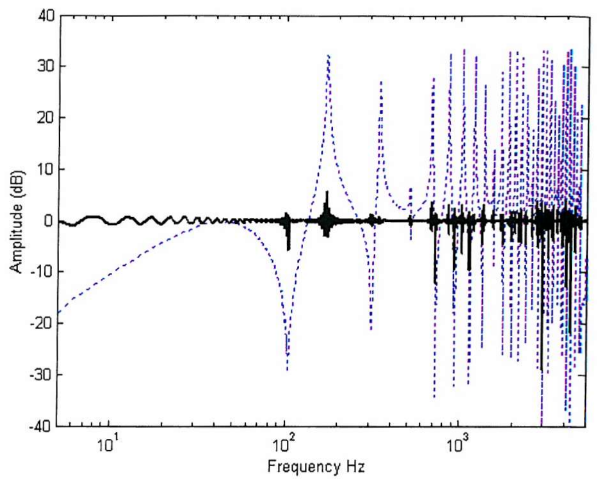
Figure 10.16: Magnitude response of equalisation filters after convergence corresponding to the FDLMS algorithm (blue), the FDLMS with frequency-dependent constraint (black solid line), and with 2dB constant constraint (dotted line) on its magnitude.

Figure 10.17 shows the magnitude response of the acoustic path G_1 (blue), equalised response without constraints, and with constraints on the filter magnitude. It is clear that without any constraints, it produces a better equalised response since it compensates for almost all notches and peaks in the magnitude. The one with constant constraint cannot compensate for all the notches; this would require a filter response with a larger magnitude. Especially there is a big notch at low frequency around 100Hz which cannot be compensated with a constant constraint. However, the simulation with frequency-dependent constraint achieves good equalisation at most frequencies, even at the large notch at low frequencies.

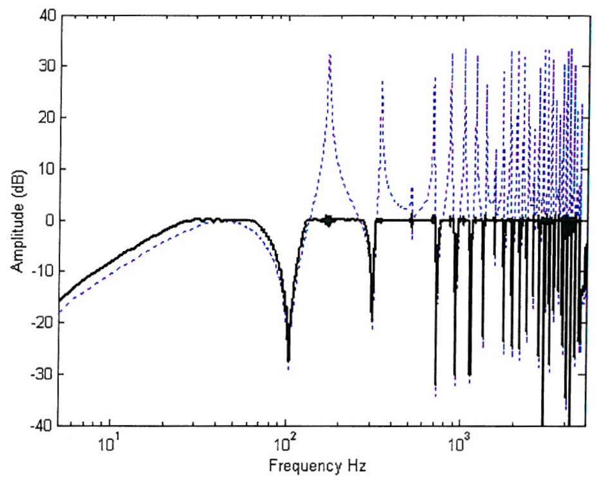
The plots of the magnitude responses shown in figures 10.18, 10.19 and 10.20 represent similar results but for equalisation positions G_2, G_3 and G_4 respectively. These responses are clearly different from those of figure 10.17. This is because the adaptive filter has been designed to equalise magnitude responses at G_1; the equalisation at G_2, G_3 and G_4 will therefore not produce sound equalisation as good as that at G_1 (as it clearly shows in figures 10.12 – 10.14).

Figures 10.18(a), 10.19(a) and 10.20(a) shows that equalised responses $|WG_2|$, $|WG_3|$ and $|WG_4|$ without using any constraint produces high peaks at high frequencies, due to the distorted sound at high frequencies away from the microphone point. The effect of this distortion can be counteracted by applying constraints. Figures 10.18(b), 10.19(b) and 10.20(b) shows that using constant constraint can avoid most high frequencies peaks, however, poor equalisation at low frequency still exist. Figures 10.18(c), 10.19(c), 10.20(c) shows that most extreme peaks were avoided and also achieves better equalisation at low frequency with frequency-dependent constraints, compared to both the equalisation without any constraints and constant constraint, as shown in figures 10.18(a), 10.19(a), 10.20(a) and figures 10.18(b), 10.19(b), 10.20(b), respectively. However, some peaks do occur at higher frequencies. These peaks will mean that sound equality away from the equalisation microphone will have poor audibility. Figures 10.18–10.20, demonstrate that satisfactory equalisation away from the microphone can be successfully achieved by adding a frequency-dependent magnitude constraint. It clearly shows that the performance of the frequency-dependent constraint has a potential in sound equalisation application, or possibly other applications.

(a)



(b)



(c)

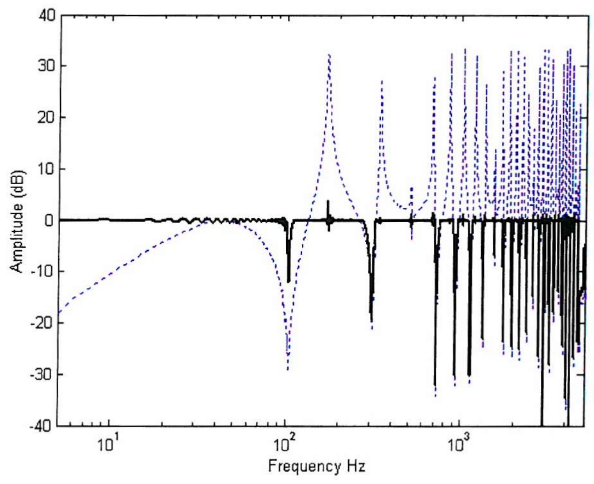
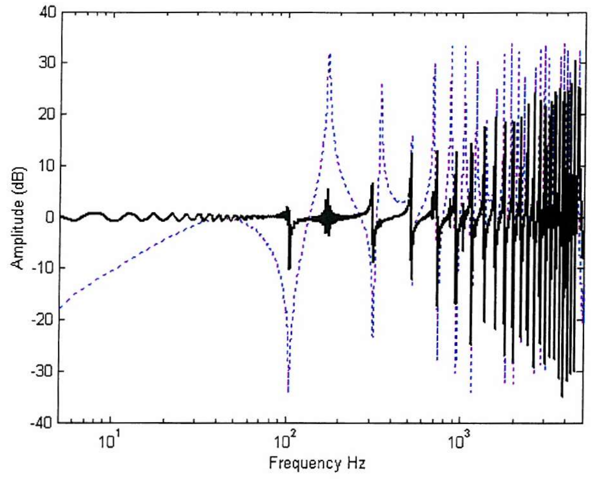
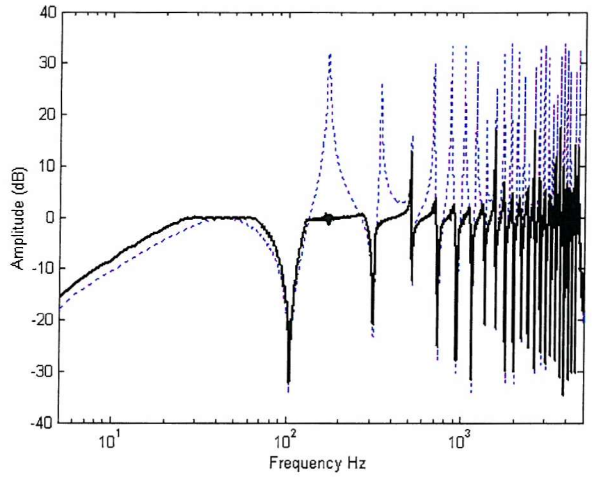


Figure 10.17: Magnitude response of unequalized path G_1 from loudspeaker to microphone (blue). (a) equalized response, $|WG_1|$ (black); (b) equalize response with magnitude 2dB constant constraint (black); (c) equalize response with frequency-dependent constraint (black).

(a)



(b)



(c)

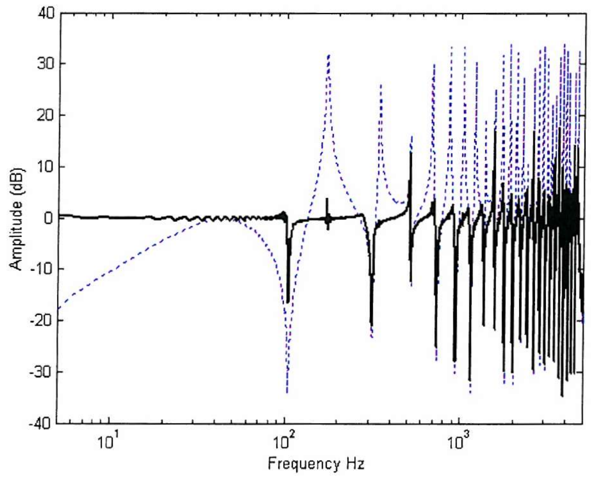
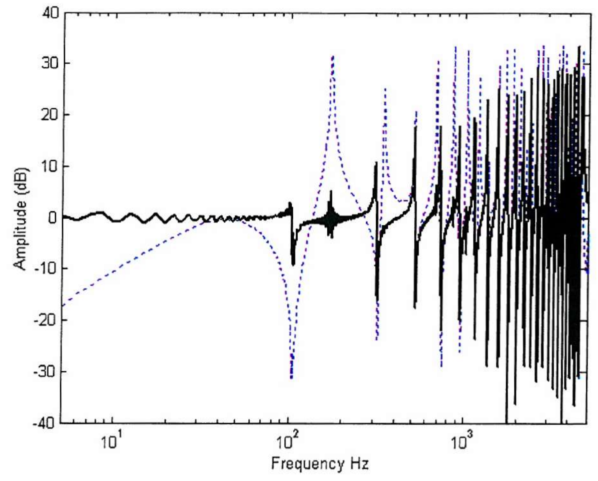
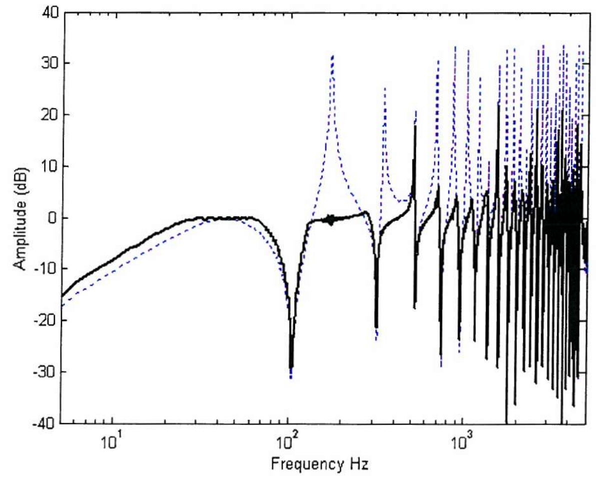


Figure 10.18: Magnitude response of unequalized path G_2 from loudspeaker to location 3cm from the equalisation microphone (blue), and corresponding (a) equalized response, $|WG_2|$ (black); (b) equalize response with magnitude 2dB constant constraint (black); (c) equalize response with frequency-dependent constraint (black).

(a)



(b)



(c)

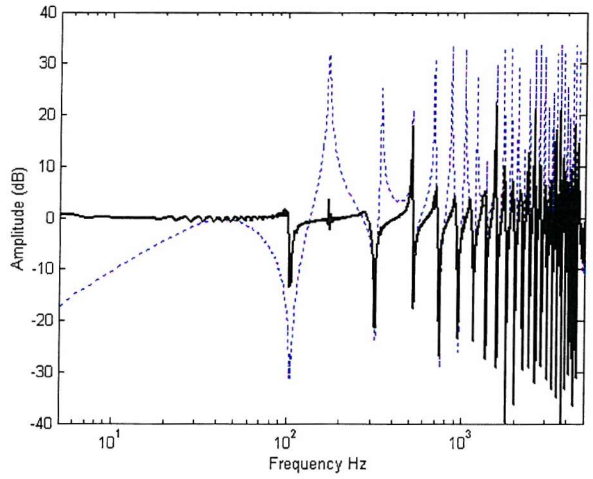
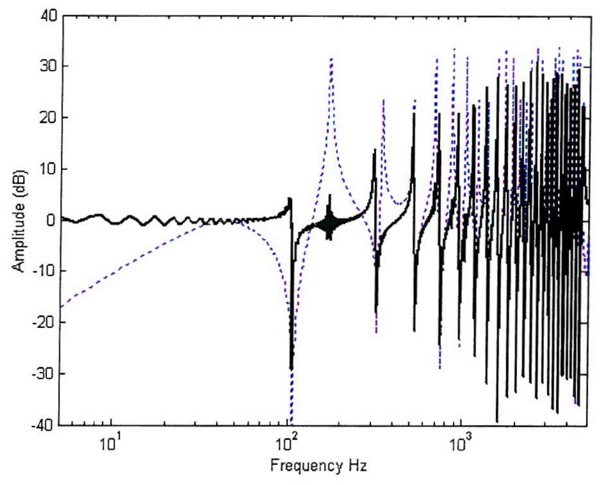
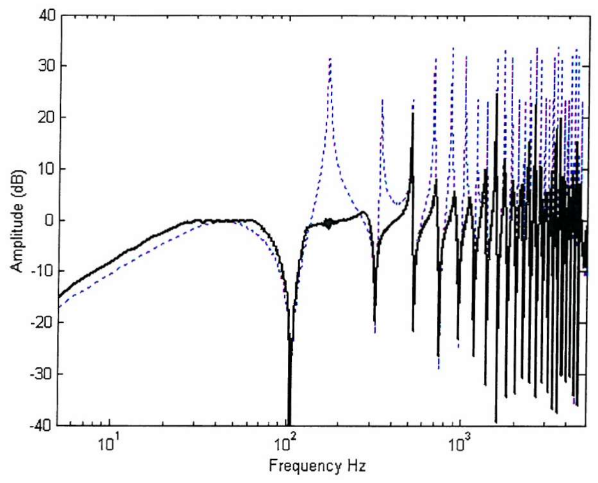


Figure 10.19 Magnitude response of unequalized path G_3 from loudspeaker to location 5cm from the equalisation microphone (blue), and corresponding (a) equalized response, $|WG_3|$ (black); (b) equalize response with magnitude 2dB constant constraint (black); (c) equalize response with frequency-dependent constraint (black).

(a)



(b)



(c)

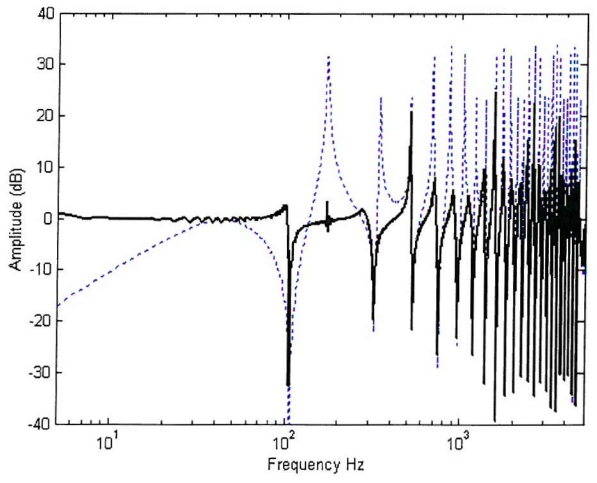


Figure 10.20: Magnitude response of unequalized path G_4 from loudspeaker to location 8cm from the equalisation microphone (blue), and corresponding (a) equalized response, $|WG_4|$ (black); (b) equalize response with magnitude 2dB constant constraint (black); (c) equalize response with frequency-dependent constraint (black).

10.5 Conclusions

Practical frequency-domain magnitude constraints for adaptive filtering have been developed and implemented in an adaptive sound equalisation system, using a constrained formulation of the FDLMS algorithm.

When measuring at a location near the microphone, it is likely that the low frequency notches will not change much while the high frequency notches will change more. In this case, we want to impose a strict limit at high frequency and a more loose limit at low frequency. Numerical simulation for a simple one-dimensional sound field in a duct has shown a strong possibility for improved equalisation of the acoustic response in an enclosed space using frequency-dependent constraint. Due to the spatial variability of sound field, the equalisation filter must not produce large peaks if a reasonable equalisation is to be achieved at positions other than the equalisation microphone. This can be successfully achieved by adding a magnitude frequency-dependent constraint.

Chapter 11

Conclusions and suggestions for future research

11.1 Introduction

In this thesis constrained adaptive filtering algorithms have been developed. A theoretical analysis of the algorithms and frequency-dependent constraint technique have been developed, which can facilitate a better convergence rate and increase the robustness. The constrained adaptive filtering algorithms, the theoretical results and the frequency-dependent constraint technique have been applied to the sound equalisation system in an enclosure. In this chapter we summarise the conclusions from this work, and the suggestions for future research.

11.2 Conclusions

The summary and conclusions of the work in this thesis are presented below.

- An investigation of the unconstrained optimisation algorithms for long FIR filters typical in acoustic applications has been presented. The efficiency of the conjugate gradient algorithm over both Newton's method and the widely used steepest descent algorithm, in terms of minimising the mean square error, was established for system identification, and prediction system problems by means of simulations. The results show that the conjugate gradient algorithm or other vector/gradient-based algorithms are better for long filtering application due to the computation time and efficiency compared with matrix-based algorithms. Matrix-based algorithms, e.g. Newton's method, are less suitable for long filtering applications because of the computations involved in the correlation matrix inversion, i.e. computational complexity and it might cause inefficient performance.

- For constrained optimisation problems, the characteristic of the penalty functions $\max(c_i, 0)$ and $\max(c_i, 0)^2$ has been investigated. It was shown by simulations and through theory that the second order penalty function is a continuous function and provides smooth convergence, however high computation complexity and high order terms will be involved during the adaptive processing. First order penalty function is not a continuous function and cause fluctuations, but it gives a simple formulation. This thesis showed the benefits of using the first order penalty function.

- Quadratic constraints were used throughout this thesis. We studied quadratic constraints that can be approximated by linear constraints, so that the objective function produced by the second order penalty function will only contain quadratic penalty term. A method of constructing linear approximations of quadratic constraints was therefore introduced, and it showed that the algorithms that use these linear approximations converged to a minimum that was close to the algorithms in the quadratically constrained case.

- Increasing the convergence rate of the constrained LMS algorithm was shown to be possible by incorporating the conjugate gradient algorithm and bin-normalised LMS algorithm. An efficient constrained CGA and BN-LMS implemented in the frequency domain are therefore developed and studied. The constrained approach with FDCGA and BN-FDLMS is used for adaptive sound equalisation with the aim of improving the convergence rate. Their performance showed that they have superior convergence rate compared with the widely used FDLMS algorithm, and therefore have the potential to track faster changing systems.

- A new theoretical analysis of the bound is constructed for the value of the convergence coefficient, which will give sufficient condition for convergence in the constrained problems studied here. Hence the theoretical analysis of the bound enables more accurate design of optimisation algorithms using the penalty method. The sound equalisation application is used to ascertain the validity of μ, and also see how σ is chosen. The results showed that the theoretical convergence coefficient is not far off the practical one. It means the bound was able to give accurate design in constrained problems without any "trial and error".

- To further improve the appeal of the constrained algorithms for sound equalisation, spatial robustness is successfully achieved by means of frequency-dependent constraints on the magnitude of the adaptive filter. Spatial robustness was improved by using frequency-dependent constraints, by utilizing the structure of the spatial correlation in the reverberant sound field. A frequency-dependent constraint was therefore designed that only limits large gains of the filter at high frequencies. The robustness of the frequency-dependent constraint was assessed in the simulation of sound equalisation with modified simulated one-dimensional duct data. The simulation has shown a strong possibility for improved equalisation of the acoustic response in an enclosed space using frequency-dependent constrained.

11.3 Recommendations for future work

- Due to the promising performance of the algorithm, further investigation may show that it would be suitable for real-time implementation in many signal processing applications on conventional hardware. The algorithm could be implemented alongside existing methods to compare convergence rates and suitability for real-time application.

- The results of the thesis demonstrated that the formulation of the constrained FDCGA performs very well, however further research could be conducted to find ways of increasing the computational efficiency of the algorithm and lowering computation time. These may be achieved by finding optimal termination criteria for the block implementation. Research could be conducted to improve the tracking ability of the algorithm for adaptive filters and to improve the convergence rate.

- The robustness of the algorithm may be improved by the addition of further frequency domain constraints. The constraints that have been studied here were chosen for their appropriateness to the sound equalisation application. Other constraints could be considered that may improve efficiency and robustness of performance of the algorithm, and be even more suitable for sound equalisation or other applications.

- Further research is suggested regarding the theoretical bound on μ, by studying the bound on μ for different quadratic constraints (depend on the application needed), and developing limits on μ for other algorithms. This could follow the derivation for the FDLMS algorithm. Further investigation of the convergence analysis of the BN-FDLMS algorithm is suggested for future work. The properties of solving different matrices and their theorems from linear algebra are also suggested for further study which might provide a more elegant way to obtain the theoretical bound on μ.

- Frequency-dependent constraint has the strong potential to increase the spatial robustness of the equalisation away from the microphone position and the performance is shown in the simulation with modified simulated one-dimensional duct data. Further investigation is therefore to apply frequency-dependent constraints in the practical application and then further expend its use.

- Finally developing other unconstrained vector/gradient-based optimisation algorithms and other constrained optimisation algorithms to constrained adaptive filtering applications are suggested for future work.

Bibliography

[1] Anstreicher K. M. (1998), "A Long-Step Path Following Algorithm for Semidefinite Programming Problems," *The Fields Institute for Research in Mathematical Sciences Communications Series*, pp. 181-196.

[2] Asano F. and Swanson D. C. (1995), "Sound Equalization in enclosures using modal reconstruction," *J. Acoust. Soc. Am.* 98 (4), pp. 2062-2069, Oct. 1995.

[3] Asano F., Suzuki Y., and Sone T. (1996), "Sound Equalization using derivative constraints," Acustica acta acustics, Vol. 82, pp. 311-320.

[4] Bershad N. J. and Feintuch P. L. (1986), "A Normalized Frequency Domain LMS Adaptive Algorithm," *IEEE Transactions on Acoustics, Speech, and Signal Progressing,* Vol. 34, No. 3, pp.452-461, June 1986.

[5] Boray G. K. and Srinath M. D. (1992), "Conjugate gradient techniques for adaptive filtering," *IEEE Transactions on Circuits and Systems I,* Vol. 39, No. 1, pp. 1-10, Jan. 1992.

[6] Boroujeny B. F. and Chan K. S. (2000), "Analysis of the Frequency-Domain Block LMS Algorithm," *IEEE Transactions on Signal Processing*, Vol. 48, No. 8, pp. 2332-2342, Aug. 2000.

[7] Boyd S. and Vandenberghe L. (1996), "Semidefinite programming", *SIAM Review,* 38(1): 49- 5, pp. 49-95, March 1996

[8] Boyd S. and Vandenberghe L. (1998), "SP Software for Semidefinite Programming, User's Guide," *Version 1.0. Stanford University*, November 1998.

[9] Boyd S. and Vandenberghe L. (1999), "Applications of semidefinite programming," *Applied Numerical Mathematics*, 29:283-299, 1999.

[10] Chang P. S. and Willson A. N. (1995), "Adaptive Filtering using Modified Conjugate Gradient," in *Proc. 38th Midwest Symp. Circuits Syst.*, Rio de Janeiro, Brazil, pp. 243-246, Aug. 1995.

[11] Chang P. S. and Willson A. N. (1996), "Adaptive spectral estimation using the conjugate gradient algorithm," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.,* Atlanta, GA, pp. 2979-2982, May 1996.

[12] Chang P. S. and Willson A. N. (2000), "Analysis of conjugate gradient algorithms for adaptive filtering," *IEEE Transactions on Signal Processing,* Vol. 48, No. 2, pp. 409-418, February 2000.

[13] Clark G. A., Mitra S. and Parker S. R. (1981), "Block Implementation of Adaptive Digital Filters," *IEEE Transactions on Circuits and Systems*, Vol. 28, No. 6, pp. 584-592, June 1981.

[14] Cook R. K., Waterhouse R. V., Berendt R. D., Edelman S. and Thompson M. C. (1955), "Measurement of Correlation Coefficients in Reverberant Sound Fields," *The Journal of the Acoustical Society of America*, Vol. 27, No. 6, Nov.

[15] Cooper L. and Steinberg D. (1970). *Introduction to Methods of optimization*. W.B. Saunders company.

[16] Cowan C. F. N. and Grant P. M. (1985). *Adaptive filters*. Prentice-Hall, Inc. 1985.

[17] Davidon W. C. (1959), "Variable metric method for minimization," A.E.C. Research and Development Report, ANL-5990 (Rev.).

[18] Dennis J. E. and Schnabel R. B. (1983). *Numerical methods for Unconstrained optimization and nonlinear equations*. Prentice-Hall, Inc.

[19] Elliott S. J. (1994), "Practical Implementation of Low-Frequency Equalisation Using Adaptive Digital Filters," *Journal of Audio Engineering Society*, Vol. 42, No. 12, pp. 988-998, Aug. 1994.

[20] Elliott S. J. (2001). *Signal Processing for Active Control*. Academic Press. 2001.

[21] Elliott S. J., Bathia L. P., Deghan F. S., fu A. H., Stewart M. S. and Wilson D. W. (1994), "Practical implementation of low-frequency equalization using adaptive digital filters," *Journal of Audio Engineering Society*, Vol. 42, pp. 988-998.

[22] Elliott S. J. and Nelson P. A. (1989), "Multiple-Point Equalisation in a room using Adaptive Digital Filters," *Journal of Audio Engineering Society*, Vol. 37, No. 11, pp. 899-907, Nov. 1989.

[23] Eilliott S. J. and Rafaely R. (2000), "Frequency-Domain Adaptation of Causal Digital Filters," *IEEE Transactions on Signal Processing*, Vol. 48, No. 5, pp. 1354-1364, May 2000.

[24] Ferrara E. R. (1986), "Fast Implementation of LMS Adaptive Filters," *IEEE Transactions on Acoustics, Speech, and Signal Progressing,* Vol. 28, No. 4, pp.474-475, Aug. 1986.

[25] Fletcher R. (1970), "A new approach to variable metric algorithms," *The Computer Journal*, Vol. 13, No. 3, pp. 317-322, Aug. 1970.

[26] Fletcher R. and Powell M. J. D. (1963), "A rapidly convergent descent method for minimization," *Computer Journal*, Vol. 6, pp.163-68, 1963.

[27] Fletcher R. and Reeves C.M. (1964), "Function Minimization by Conjugate Gradients," *The computer Journal*, Vol. 7, pp.149-153, 1964.

[28] Fletcher R. (1987). *Practical Methods of Optimization.* New York: Wiley, 1987.

[29] Fonseca P. D., Sas P., and Brussel H. V., "Robust design and robust stability analysis of active noise control systems," *Katholieke University Van Leuven.*

[30] Freund R. M. (1994), "Complexity of an Algorithm for Finding an Approximate Solution of a Semi-Definite Program, with no Regularity Condition," *O.R. Center Working Paper* pp. 302-94, Dec 1994.

[31] Fuller L. E. (1962). *Basic Matrix Theory.* Prentice-Hall, Inc. 1962.

[32] Gill P. E. and Murray W. (1974). *Numerical methods for constrained optimization.* Academic press. 1974.

[33] Gill p., Murray W., and Wright M. H. (1981). *Practical optimization.* Academic press. 1981.

[34] Golub G. H. and Van Loan C. F. (1983). *Matrix Computations*. Oxford Academic, 2nd and 3rd Edition.

[35] Grace, A. (1995). *Matlab Optimization Toolbox*. The MathWorks, Inc.

[36] Greig D.M. (1980). *Optimisation*. New York: Longman, 1980.

[37] Haykin S. (1984). *Introduction To Adaptive Filters*. New York: Macmillan Publishing Company, 1984.

[38] Haykin S. (1986). *Adaptive Filter Theory*. Prentice hall, Englewood Cliffs, NJ, 1986.

[39] Helmberg, C. (2000), "Semidefinite Programming for Combinatorial Optimization," *Habilitationsschrift*, TU Berlin, January 2000. ZIB-Report ZR-00-34, Konrad-Zuse-Zentrum Berlin, October 2000.

[40] Hestenes M. (1980). *Conjugate Direction methods in optimization*. Springer-Verlag 1980.

[41] Ing Kai-Uwe Bletzinger (2002), "Structural Optimization," *Lectures notes*. Technische University Munchen, Germany. Summer 2002

[42] Jarre F. (1998), "A QQP-Minimization Method for Semidefinite and Smooth Nonconvex Programs," *University of Norte Dame*, 1998.

[43] Janssen A. J. E. M. (1987), "On certain integrals occurring in the analysis of frequency-domain power-compensated adaptive filter," *Phillips J. Res.* Vol. 42, pp. 131-171, 1987.

[44] Johansen L. G. and Rubak P. (1998), "On the equalisation of loudspeakers in closed rooms," *Acoustical Meeting, Stockholm*, Aalborg University, Denmark.

[45] Kirkeby O. and Nelson P. A. (1999), "Digital filter design for inversion problems in sound reproduction," *Journal of Audio Engineering Society*, Vol. 47, pp. 583-595.

[46] Krauss T. P., Shure L., and Little J. N. (1994). *Matlab Signal Processing Toolbox*, The MathWorks, Inc., 1994.

[47] Lee J. C. and Un C. K. (1989), "Performance analysis of Frequency-domain Block LMS Adaptive Digital Filters," *IEEE Transactions on Circuits and Systems*, Vol. 36, No. 2, pp. 173-189, Feb. 1989.

[48] Lim J. S. and Un C. K. (1993), "Conjugate gradient algorithm for block FIR adaptive filtering," *Electronics Letters,* Vol. 29, No. 5, 4th March 1993.

[49] Lim J. S. and Un C. K. (1996), "Block conjugate algorithms for adpative filtering," *Signal Processing* Vol. 55, pp. 65–77, June 1996.

[50] Lim J. S. and Un C. K. (1997), "Optimal block adaptive filtering algorithms using the preconditioning technique," *IEEE Transactions on Signal Processing,* Vol. 45, No. 3, March 1997.

[51] Leunberger D. G. (1973). *Introduction to Linear and Non-Linear programming.* Addison Wesley Publishing Company.

[52] Mikhael W. B. and Wu F. H. (1987), "Fast Algorithms for Block FIR Adaptive Digital Filtering," *IEEE Transactions on Circuits and Systems*, Vol. 34, No. 10, pp.1152-1160, Oct.

[53] Miyoshi M. and Kaneda Y. (1988), "Inverse filtering of room acoustics," *IEEE Transactions on Acoustics, Speech, and Signal Progressing,* Vol. 36, pp. 145-152.

[54] Moon T. K. and Stirling W. C. (2000). *Mathematical methods and algorithms for signal processing.* Prentice Hall.

[55] Nelson P. A., Orduna-Bustamante F. and Hamada H. (1995), "Inverse filter design and equalization zones in multichannel sound reproduction," *IEEE Transaction on Speech and Audio Processing*, Vol. 3, pp. 185-192.

[56] Orfanidis S. J. (1985). *Optimum signal processing.* Macmillan Publishing Company.

[57] Pearson J. D. (1969), "Variable metric methods of minimisation," *Computer Journal.* pp. 171-178.

[58] Pillo G. D. and Palagi L. (2000), "Nonlinear programming: Introduction Unconstrained and Constrained Optimization," *In Handbook of Applied Optimization, Springer Verlag, P. Pardalos and M. Resende eds*. Pp. 263-298

[59] Powell M. J. D. (1964), "An efficient method for finding the minimum of a function of several variables without calculating derivatives," pp. 155-162.

[60] Radlovic B. D., Williamson, r. C. and Kennedy R. A. (2000), "Equalization in an Acoustic Reverberant Environment: Robustness Results," *IEEE Transactions on Speech and Audio Processing*, Vol. 8, No. 3, pp. 311-319, May.

[61] Rafaely B. (1997), "Feedback control of sound," *Ph.D. thesis*, ISVR, University of Southampton, UK. 1997.

[62] Rafaely B. and Elliott S. J. (1997), "Frequency-Domain Adaptive Control With Design Constraints," *IEE Computing and Control Divison*, Adaptive Controllers in Practice '97. April.

[63] Rafaely B. and Elliott S. J. (1999), "H_2/H_∞ Active Control of Sound in a Headrest: Design and Implementation," *IEEE Transaction on Control Systems Technology*, Vol. 7, No. 1, pp. 79-84, Jan.

[64] Rafaely B. (2000), "Spatial-temporal correlation of a diffuse sound field," *ISVR*, University of Southampton, U.K. Feb.

[65] Rafaely B. and Elliott S. J. (2000a), "Frequency-Domain Adaptation of Causal Digital Filters," *IEEE Trans. On Signal Processing*, Vol. 48, No.5, pp. 1354-1364, May 2000.

[66] Rafaely B. and Elliott S. J. (2000b), "A computationally efficient frequency-domain LMS algorithm with constraints on the adaptive filter," *IEEE Transactiona on Signal Processing,* Vol. 48, No. 6, pp. 1649-1655, June 2000.

[67] Radlovic B. D. and Williamson R. C. (2000), "Equalisation in an Acoustic Reverberant Environment: Robustness Results," *IEEE Transaction on Speech and Audio Processing*, Vol. 8, No. 3, May.

[68] Roberts, A. W. and Varberg, D. E. (1973). *Convex Functions.* Academic press. 1973.

[69] Rosen S. and Howell P. (1991). *Signals and systems for speech and hearing.* Academic Press. 1991.

[70] Saad Y. (1996). *Iterative Methods for Sparse Linear systems.* PWS Publishing Company. 1996.

[71] Santillan A. O. (2001), "Spatially extended sound equalization in rectangular rooms," *J. Acoust. Soc. Am,* 110 (4), pp.1989-1997, Oct. 2001.

[72] Schittkowski K. and Zillober Ch. (2002), "Nonlinear programming," *University of Bayreuth, Germany.*

[73] Shusina N. A. and Rafaely B. (2000), "Study of the Frequency-Domian LMS for Feedback Cancellation in Hearing Aids," in *Proceedings of the Fifth International Conference on Mathematics in Signal Processing*, Warwick, UK, Dec. 2000.

[74] Shynk J. J. (1992), "Frequency-Domain and Multirate Adaptive Filtering," *IEEE Signal Processing Mag.*, Vol. 9, pp. 14-37, Jan. 1992.

[75] Sommen P. C. W., Van Gerwen P. J., Kotmans H. J. and Janssen A. J. E. M. (1987), "Convergence Analysis of a Frequency-domain adaptive Filter with Exponential Power Averaging and Generalized Window Function," *IEEE Transactions on Circuits and Systems*, Vol. 34, No. 7, pp. 788-798, July 1987.

[76] Tam, P.S., Rafaely, B. (2001), "Efficient Computation of Very Long Optimal FIR Filters," *Fifth International Conference on Mathematics on Signal Processing*, Warwick, U.K., Dec. 2001.

[77] Tam P. S. and Rafaely B. (2002), "Adaptive Sound Equalisation using Constrained Frequency-domain Filters," *Active 2002*, ISVR Southampton, U.K., July 2002.

[78] Todd M. (2001), "Semidefinite Optimization", *Acta Numerica* 10(2001), pp. 515-560.

[79] Toh K. C., Todd M. J. and Tutuncu R. H., "SDPT3 – a Matlab software package for semidefinite programming," http://www.math.cmu.edu/~reha/eguide/guide.html, Oct. 1998.

[80] Vandenberghe L. and Boud S. (1999), "Applications of semidefinite programming," *Applied Numerical Mathematic*, Vol. 29, pp. 283-299.

[81] Walsh G. R. (1975). *Methods of optimization*. John Wiley & Sons.

[82] Widrow B. and Stearns S. D. (1985). *Adaptive Signal Processing*. Prentice Hall.

[83] Wilkinson J. H. (1965). *The algebraic eigenvalueproblem*. Oxford University Press.

[84] Wolkowicz H, Saigal R. and Vandenberghe L. (2000). "Handbook on Semidefinite Programming," *Kluwer*, 2000.

[85] Wolfe M. A.(1978). *Numerical methods for unconstrained optimization*. Van nostrand reinhold company 1978.

[86] Zhao Q., Karisch S. E., Rendl F. and Wolkowicz H. (1998) "Semidefinite Programming Relaxations for the Quadratic Assignment Problem," *Journal of Combinatorial Optimization*, 2(1): pp. 71-109, 1998.

Appendix:

Chapter 2: Theorem 2.1

Proof: [Greig, 1980 and Wolfe, 1978]

Let λ_i ($i = 0, \ldots, m$) be such that

$$\sum_{i=0}^{m} \lambda_i p_i = 0$$

then

$$\sum_{i=0}^{m} \lambda_i G p_i = 0$$

$$\Rightarrow \qquad \sum_{i=0}^{m} \lambda_i p_j^{\mathrm{T}} G p_i = 0 \qquad (j = 0, \ldots, m).$$

So by hypothesis (2)

$$\Rightarrow \qquad \lambda_i \boldsymbol{p}_i^{\mathrm{T}} G \boldsymbol{p}_i = 0 \qquad (i = 0, \ldots, m),$$

but by hypothesis (1),

$$\Rightarrow \qquad \boldsymbol{p}_i^{\mathrm{T}} \boldsymbol{G} \boldsymbol{p}_i > 0, \text{ since } \boldsymbol{G} \text{ is positive definite.}$$

Hence $\lambda_i = 0$, and $\boldsymbol{p}_i^{\mathrm{T}} \boldsymbol{G} \boldsymbol{p}_i \neq 0$, i.e. $\boldsymbol{p}_i$ are linearly independent for $i = 0, 1, \ldots, m$. //

Chapter 2: Theorem 2.2

Proof: [Dennis et al, 1983 and Fletcher, 1987]

By condition (a) and (b), we have

$$\boldsymbol{p}_i^{\mathrm{T}}\boldsymbol{g}_k = 0, \qquad \boldsymbol{p}_j^{\mathrm{T}}\boldsymbol{g}_{k+1} = 0$$

and also by condition (c), we get

$$\boldsymbol{p}_i^{\mathrm{T}}\boldsymbol{G}\boldsymbol{p}_j = 0$$

Since, from the quadratic function, we know that

$$\boldsymbol{g}_{k+1} - \boldsymbol{g}_k = \boldsymbol{G}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k),$$

and we have

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_j^* \boldsymbol{p}_j,$$

where α_j^* is the value of α_j which minimises $f(\boldsymbol{x}_k + \alpha_j \boldsymbol{p}_j)$. Hence,

$$\boldsymbol{p}_i^{\mathrm{T}}\boldsymbol{g}_{k+1} = \boldsymbol{p}_i^{\mathrm{T}}(\boldsymbol{g}_k + \alpha_j^*\boldsymbol{G}\boldsymbol{p}_j) = 0,$$

Hence $\boldsymbol{g}_{k+1}$ is orthogonal to all the preceding descent directions $\boldsymbol{p}_i$ $(i = 0, 1, \ldots, k)$. //

Chapter 2: Theorem 2.3

Proof: [Fletcher, 1987]

By the conjugate direction algorithm, we have α_k by estimating a minimiser of $f(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k)$. So, $\boldsymbol{x}_{k+1}$ can be written as

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_0 + \sum_{j=0}^{k} \alpha_j p_j \qquad (0 \le k < n) \tag{A.1}$$

By $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{Gx} + \boldsymbol{b}$, so we have

$$\boldsymbol{g}_{k+1} = \boldsymbol{Gx}_{k+1} + \boldsymbol{b}$$

$$\Rightarrow \qquad = \boldsymbol{Gx}_0 + \boldsymbol{b} + \sum_{j=0}^{k} \alpha_j G p_j$$

$$\Rightarrow \qquad \alpha_k = -\frac{\boldsymbol{p}_k^{\mathrm{T}} \boldsymbol{Gx}_0}{\boldsymbol{p}_k^{\mathrm{T}} \boldsymbol{Gp}_k} - \frac{\boldsymbol{p}_k^{\mathrm{T}} \boldsymbol{b}}{\boldsymbol{p}_k^{\mathrm{T}} \boldsymbol{Gp}_k} \qquad (0 \le k < n),$$

then by (A.1) with $k = n - 1$, we get

$$\boldsymbol{x}_n = \boldsymbol{x}_0 - \sum_{j=0}^{n-1} \frac{p_j^{\mathrm{T}} G x_0}{p_j^{\mathrm{T}} G p_j} p_j - \sum_{j=0}^{n-1} \frac{p_j^{\mathrm{T}} G(G^{-1} b)}{p_j^{\mathrm{T}} G p_j} p_j$$

$$\Rightarrow \qquad \boldsymbol{x}_n = \boldsymbol{x}_0 - \boldsymbol{x}_0 - \boldsymbol{G}^{-1}\boldsymbol{b}$$

$$\Rightarrow \qquad = -\boldsymbol{G}^{-1}\boldsymbol{b}$$

but by $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{Gx} + \boldsymbol{b}$ and $\boldsymbol{g}(\boldsymbol{x}^*) = 0$

$$\Rightarrow \qquad \boldsymbol{x}^* = -\boldsymbol{G}^{-1}\boldsymbol{b}.$$

Hence, $\boldsymbol{x}_n = \boldsymbol{x}^*$.

Hence, depending upon x_0, the minimizer $\boldsymbol{x}^*$ of f is attained in at most n iterations. //

Chapter 3: Testing for Convexity

A function *f*(*x*) is convex if for any two points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, we have

$$f(\boldsymbol{x}_2) - f(\boldsymbol{x}_1) \geq \nabla f^{\mathrm{T}}(\boldsymbol{x}_1)(\boldsymbol{x}_2 - \boldsymbol{x}_1).$$

Proof: [Fletcher, 1987][Roberts and Varberg, 1973]

If *f*(*x*) is convex, we have by definition

$$f(\theta\, \boldsymbol{x}_2 + (1-\theta)\boldsymbol{x}_1) \leq \theta\, f(\boldsymbol{x}_2) + (1-\theta) f(\boldsymbol{x}_1)$$

that is,

$$f(\boldsymbol{x}_1 + \theta(\boldsymbol{x}_2 - \boldsymbol{x}_1)) \leq f(\boldsymbol{x}_1) + \theta(f(\boldsymbol{x}_2) - f(\boldsymbol{x}_1))$$

Let $\boldsymbol{h} = \boldsymbol{x}_2 - \boldsymbol{x}_1$, then we have:

$$f(\boldsymbol{x}_1 + \theta\, h) \leq f(\boldsymbol{x}_1) + \theta(f(\boldsymbol{x}_2) - f(\boldsymbol{x}_1))$$

Now, using the Taylor Series:

$$\begin{aligned}
\theta(f(\boldsymbol{x}_2) - f(\boldsymbol{x}_1)) &\geq f(\boldsymbol{x}_1 + \theta\, \boldsymbol{h}) - f(\boldsymbol{x}_1) \\
&\geq f(\boldsymbol{x}_1) + \theta\, \boldsymbol{h} \nabla f(\boldsymbol{x}_1) + \theta^2 \boldsymbol{h}^{\mathrm{T}} \nabla^2 f(\boldsymbol{x}_1)\boldsymbol{h} - f(x_1) \\
&\geq \theta\, \boldsymbol{h} \nabla f(\boldsymbol{x}_1) \\
&\geq \theta\, \nabla f(\boldsymbol{x}_1)(\boldsymbol{x}_2 - \boldsymbol{x}_1)
\end{aligned}$$

Cancelling through by *θ*, we have $f(\boldsymbol{x}_2) - f(\boldsymbol{x}_1) \geq \nabla f^{\mathrm{T}}(\boldsymbol{x}_1)(\boldsymbol{x}_2 - \boldsymbol{x}_1)$. //

Chapter 3:

A function $f(x)$ is convex if the Hessian matrix $\nabla^2 f(x)$ is positive semidefinite.

Proof: [Fletcher, 1987][Roberts and Varberg, 1973]

From Taylor's theorem we have

$$f(\boldsymbol{x}^* + \boldsymbol{h}) = f(\boldsymbol{x}^*) + \sum_{i=1}^{n} h_i \frac{\partial f}{\partial x_i}(\boldsymbol{x}^*) + \frac{1}{2!}\sum_{i=1}^{n}\sum_{j=1}^{n} h_i h_j \left.\frac{\partial^2 f}{\partial x_i \partial x_j}\right|_{x=x^*+\theta h}$$

where $0 < \theta < 1$. By letting $\boldsymbol{x}^* = \boldsymbol{x}_1$, $\boldsymbol{x}^* + \boldsymbol{h} = \boldsymbol{x}_2$ and $\boldsymbol{h} = \boldsymbol{x}_2 - \boldsymbol{x}_1$,

$$\Rightarrow \quad f(\boldsymbol{x}_2) = f(\boldsymbol{x}_1) + \nabla f^{\mathrm{T}}(\boldsymbol{x}_1)(\boldsymbol{x}_2 - \boldsymbol{x}_1) + \frac{1}{2}(\boldsymbol{x}_2 - \boldsymbol{x}_1)^{\mathrm{T}} \nabla^2 f[\boldsymbol{x}_1 + \theta(\boldsymbol{x}_2 - \boldsymbol{x}_1)](\boldsymbol{x}_2 - \boldsymbol{x}_1).$$

It can be seen that inequality $f(\boldsymbol{x}_2) - f(\boldsymbol{x}_1) \geq \nabla f^{\mathrm{T}}(\boldsymbol{x}_1)(\boldsymbol{x}_2 - \boldsymbol{x}_1)$ is satisfied if $\nabla^2 f(\boldsymbol{x})$ is positive semidefinite.

Further, if $\nabla^2 f(x)$ is positive definite, the function $f(x)$ will be strictly convex. //

Chapter 7: Appendix 1

Example 1: Find the eigenvalues of the size 2 × 2 matrix A at frequency $k = 1$, and since A is $2N \times 2N$ matrix, thus $N = 1$ and $n = 0, 1$:

$$\det(A - \lambda I) = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = \begin{vmatrix} 1-\lambda & e^{-j\pi} \\ e^{j\pi} & 1-\lambda \end{vmatrix}$$

$$= (1-\lambda)(1-\lambda) - e^{-j\pi}e^{j\pi}$$
$$= 1 - 2\lambda + \lambda^2 - 1$$
$$= \lambda(\lambda - 2)$$
$$= \lambda^{2N-1}(\lambda - 2N)$$

Hence, the roots are 0, and 2.

Example 2: Find the eigenvalues of the size 4 × 4 matrix A at frequency $k = 1$. A is $2N \times 2N$ matrix, thus $N = 2$ and $n = 0, 1, 2, 3$:

$$\det(A - \lambda I) = \begin{vmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{vmatrix} = \begin{vmatrix} 1-\lambda & e^{-j\pi/2} & e^{-j\pi} & e^{-j3\pi/2} \\ e^{j\pi/2} & 1-\lambda & e^{-j\pi/2} & e^{-j\pi} \\ e^{j\pi} & e^{j\pi/2} & 1-\lambda & e^{-j\pi/2} \\ e^{j3\pi/2} & e^{j\pi} & e^{j\pi/2} & 1-\lambda \end{vmatrix}$$

$$\begin{aligned} &= a_{11}a_{22}a_{33}a_{44} - a_{11}a_{22}a_{34}a_{43} + a_{11}a_{23}a_{34}a_{42} \\ &\quad - a_{11}a_{23}a_{32}a_{44} + a_{11}a_{24}a_{32}a_{43} - a_{11}a_{24}a_{33}a_{42} \\ &\quad - a_{12}a_{23}a_{34}a_{41} + a_{12}a_{23}a_{31}a_{44} - a_{12}a_{24}a_{31}a_{43} \\ &\quad + a_{12}a_{24}a_{33}a_{41} - a_{12}a_{21}a_{33}a_{44} + a_{12}a_{21}a_{34}a_{43} \\ &\quad + a_{13}a_{24}a_{31}a_{42} - a_{13}a_{24}a_{32}a_{41} + a_{13}a_{21}a_{32}a_{44} \\ &\quad - a_{13}a_{21}a_{34}a_{42} + a_{13}a_{22}a_{34}a_{41} - a_{13}a_{22}a_{31}a_{44} \\ &\quad - a_{14}a_{21}a_{32}a_{43} + a_{14}a_{21}a_{33}a_{42} - a_{14}a_{22}a_{33}a_{41} \\ &\quad + a_{14}a_{22}a_{31}a_{43} - a_{14}a_{23}a_{31}a_{42} + a_{14}a_{23}a_{32}a_{41} \end{aligned}$$

$$\begin{aligned}
&= (1-\lambda)(1-\lambda)(1-\lambda)(1-\lambda) - (1-\lambda)(1-\lambda)(e^{-j\pi/2})(e^{j\pi/2}) + (1-\lambda)(e^{-j\pi/2})(e^{-j\pi/2})(e^{j\pi}) \\
&\quad - (1-\lambda)(e^{-j\pi/2})(e^{j\pi/2})(1-\lambda) + (1-\lambda)(e^{-j\pi})(e^{j\pi/2})(e^{j\pi/2}) - (1-\lambda)(e^{-j\pi})(1-\lambda)(e^{j\pi}) \\
&\quad - (e^{-j\pi/2})(e^{-j\pi/2})(e^{-j\pi/2})(e^{j3\pi/2}) + (e^{-j\pi/2})(e^{-j\pi/2})(e^{j\pi})(1-\lambda) - (e^{-j\pi/2})(e^{-j\pi})(e^{j\pi})(e^{j\pi/2}) \\
&\quad + (e^{-j\pi/2})(e^{-j\pi})(1-\lambda)(e^{j3\pi/2}) - (e^{-j\pi/2})(e^{j\pi/2})(1-\lambda)(1-\lambda) + (e^{-j\pi/2})(e^{j\pi/2})(e^{-j\pi/2})(e^{j\pi/2}) \\
&\quad + (e^{-j\pi})(e^{-j\pi})(e^{j\pi})(e^{j\pi}) - (e^{-j\pi})(e^{-j\pi})(e^{j\pi/2})(e^{j3\pi/2}) + (e^{-j\pi})(e^{j\pi/2})(e^{j\pi/2})(1-\lambda) \\
&\quad - (e^{-j\pi})(e^{j\pi/2})(e^{-j\pi/2})(e^{j\pi}) + (e^{-j\pi})(1-\lambda)(e^{-j\pi/2})(e^{j3\pi/2}) - (e^{-j\pi})(1-\lambda)(e^{j\pi})(1-\lambda) \\
&\quad - (e^{-j3\pi/2})(e^{j\pi/2})(e^{j\pi/2})(e^{j\pi/2}) + (e^{-j3\pi/2})(e^{j\pi/2})(1-\lambda)(e^{j\pi}) - (e^{-j3\pi/2})(1-\lambda)(1-\lambda)(e^{j3\pi/2}) \\
&\quad + (e^{-j3\pi/2})(1-\lambda)(e^{j\pi})(e^{j\pi/2}) - (e^{-j3\pi/2})(e^{-j\pi/2})(e^{j\pi})(e^{j\pi}) + (e^{-j3\pi/2})(e^{-j\pi/2})(e^{j\pi/2})(e^{j3\pi/2})
\end{aligned}$$

$$\begin{aligned}
&= (1-\lambda)(1-\lambda)(1-\lambda)(1-\lambda) - 3(1-\lambda)(1-\lambda) + 2(1-\lambda) \\
&\quad + 2(1-\lambda) - 1 - (1-\lambda)(1-\lambda) \\
&\quad + 2(1-\lambda) - 1 - (1-\lambda)(1-\lambda) \\
&\quad + 2(1-\lambda) - 1 - (1-\lambda)(1-\lambda)
\end{aligned}$$

$$\begin{aligned}
&= (1-\lambda)^4 - 6(1-\lambda)(1-\lambda) + 8(1-\lambda) - 3 \\
&= (\lambda^4 - 4\lambda^3 + 6\lambda^2 - 4\lambda + 1) - (6\lambda^2 - 12\lambda + 6) + (8 - 8\lambda) - 3 \\
&= \lambda^4 - 4\lambda^3 \\
&= \lambda^3(\lambda - 4) \\
&= \lambda^{2N-1}(\lambda - 2N) \qquad \text{when } N = 2.
\end{aligned}$$

Hence, the roots are 0, 0, 0 and 4.

Similarly, the eigenvalues of the size $2N \times 2N$ matrices $\boldsymbol{A}_i$ at frequency $k = 0, 1, 2, \ldots, N\text{-}1$, will be

$$|\boldsymbol{A}_i| = \lambda^{2N-1}(\lambda - 2N) \qquad \Rightarrow \qquad \lambda = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 2N \end{pmatrix} \tag{i}$$

During the process of finding the eigenvalues λ, the complex exponential elements of the Fourier transform matrix $\boldsymbol{A}_i$ would cancel out each other, and give the form of equation (i). //

Chapter 7: Appendix 2

Example: To find the eigenvalues of the summation of the size 4×4 matrices A_i.

When we consider the 4×4 matrices, this means the matrix is created from samples $n = 0, 1, 2, 3$ since $n = 0, 1, 2, \ldots, 2N\text{-}1$. It also means that in this case, $N = 2$, and hence A_1 and A_2 at frequencies $k = 1$ and $k = 2$ will be considered. Therefore the summation of 4×4 matrices A_i is:

$$\sum_{i=1}^{N} A_i = \sum_{i=1}^{2} A_i = A_1 + A_2 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} + \begin{pmatrix} 1 & e^{-j\pi/2} & e^{-j\pi} & e^{-j3\pi/2} \\ e^{j\pi/2} & 1 & e^{-j\pi/2} & e^{-j\pi} \\ e^{j\pi} & e^{j\pi/2} & 1 & e^{-j\pi/2} \\ e^{j3\pi/2} & e^{j\pi} & e^{j\pi/2} & 1 \end{pmatrix} \tag{1}$$

The sum of the matrices can also be defined by:

$$\sum_{i=1}^{N} A_i = \begin{pmatrix} N & \sum_{k=0}^{N-1} e^{-jk(\pi/N)} & 0 & \sum_{k=0}^{N-1} e^{-j3k(\pi/N)} \\ \sum_{k=0}^{N-1} e^{jk(\pi/N)} & N & \sum_{k=0}^{N-1} e^{-jk(\pi/N)} & 0 \\ 0 & \sum_{k=0}^{N-1} e^{jk(\pi/N)} & N & \sum_{k=0}^{N-1} e^{-jk(\pi/N)} \\ \sum_{k=0}^{N-1} e^{j3k(\pi/N)} & 0 & \sum_{k=0}^{N-1} e^{jk(\pi/N)} & N \end{pmatrix} \tag{2}$$

The above equation was defined in section 7.3, equation (7.56).

So from (1) or (2), $\Rightarrow$

$$\sum_{i=1}^{N} A_i = A_1 + A_2 = \begin{pmatrix} 2 & 1+e^{-j\pi/2} & 0 & 1+e^{-j3\pi/2} \\ 1+e^{j\pi/2} & 2 & 1+e^{-j\pi/2} & 0 \\ 0 & 1+e^{j\pi/2} & 2 & 1+e^{-j\pi/2} \\ 1+e^{j3\pi/2} & 0 & 1+e^{j\pi/2} & 2 \end{pmatrix} \tag{3}$$

Consider the determinant of $\sum_{i=1}^{2} A_i$,

$$\det\left(\left(\sum_{i=1}^{2} A_i\right) - \lambda I\right) = \begin{vmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{vmatrix} = 0 \tag{4}$$

$$\Rightarrow \quad \begin{vmatrix} 2-\lambda & 1+e^{-j\pi/2} & 0 & 1+e^{-j3\pi/2} \\ 1+e^{j\pi/2} & 2-\lambda & 1+e^{-j\pi/2} & 0 \\ 0 & 1+e^{j\pi/2} & 2-\lambda & 1+e^{-j\pi/2} \\ 1+e^{j3\pi/2} & 0 & 1+e^{j\pi/2} & 2-\lambda \end{vmatrix} = 0 \qquad (5)$$

Similar as appendix 1, using the equation (7.42) in section 7.3, we have

$$\begin{aligned} 0 = {} & A_{11}A_{22}A_{33}A_{44} - A_{11}A_{22}A_{34}A_{43} + A_{11}A_{23}A_{34}A_{42} \\ & - A_{11}A_{23}A_{32}A_{44} + A_{11}A_{24}A_{32}A_{43} - A_{11}A_{24}A_{33}A_{42} \\ & - A_{12}A_{23}A_{34}A_{41} + A_{12}A_{23}A_{31}A_{44} - A_{12}A_{24}A_{31}A_{43} \\ & + A_{12}A_{24}A_{33}A_{41} - A_{12}A_{21}A_{33}A_{44} + A_{12}A_{21}A_{34}A_{43} \\ & - A_{14}A_{21}A_{32}A_{43} + A_{14}A_{21}A_{33}A_{42} - A_{14}A_{22}A_{33}A_{41} \\ & + A_{14}A_{22}A_{31}A_{43} - A_{14}A_{23}A_{31}A_{42} + A_{14}A_{23}A_{32}A_{41} \end{aligned} \qquad (6)$$

As equation (5) shows that the element A_{13}, the element of first row and third column, contains zero. So the permutation of A_{13} can be negated. Also the elements of A_{24}, A_{31}, A_{42} are zeros, so equation (6) becomes,

$$\begin{aligned} 0 = {} & (2-\lambda)(2-\lambda)(2-\lambda)(2-\lambda) - (2-\lambda)(2-\lambda)(1+e^{-j\pi/2})(1+e^{j\pi/2}) \\ & -(2-\lambda)(1+e^{-j\pi/2})(1+e^{j\pi/2})(2-\lambda) \\ & -(1+e^{-j\pi/2})(1+e^{-j\pi/2})(1+e^{-j\pi/2})(1+e^{j3\pi/2}) \\ & -(1+e^{-j\pi/2})(1+e^{j\pi/2})(2-\lambda)(2-\lambda) + (1+e^{-j\pi/2})(1+e^{j\pi/2})(1+e^{-j\pi/2})(1+e^{j\pi/2}) \\ & -(1+e^{-j3\pi/2})(1+e^{j\pi/2})(1+e^{j\pi/2})(1+e^{j\pi/2}) - (1+e^{-j3\pi/2})(2-\lambda)(2-\lambda)(1+e^{j3\pi/2}) \\ & +(1+e^{-j3\pi/2})(1+e^{-j\pi/2})(1+e^{j\pi/2})(1+e^{-j3\pi/2}) \end{aligned} \qquad (7)$$

$$\begin{aligned} \Rightarrow \quad 0 = {} & (2-\lambda)^4 - [2(2-\lambda)(2-\lambda)] - [2(2-\lambda)(2-\lambda)] \\ & -(-4) - [2(2-\lambda)(2-\lambda)] + 4 \\ & -(-4) - [2(2-\lambda)(2-\lambda)] + 4 \end{aligned}$$

$$\Rightarrow \quad 0 = (2-\lambda)^4 - 4[2(2-\lambda)(2-\lambda)] + 16$$

$$\Rightarrow \quad 0 = [\lambda^4 - 8\lambda^3 + 24\lambda^2 - 32\lambda + 16] - [8\lambda^2 - 32\lambda + 32] + 16$$

$$\Rightarrow \quad 0 = \lambda^4 - 8\lambda^3 + 16\lambda^2$$

$$\Rightarrow \quad 0 = \lambda^2(\lambda^2 - 8\lambda + 16)$$

$$\Rightarrow \quad 0 = \lambda^2(\lambda - 4)^2$$

Hence, the roots are $\lambda_1 = \lambda_2 = 0$ and $\lambda_3 = \lambda_4 = 4$. It can be written as

$$\lambda(A_1 + A_2) = \lambda\left(\sum_{i=1}^{2} A_i\right) = \begin{pmatrix} 0 \\ 0 \\ 4 \\ 4 \end{pmatrix} \tag{8}$$

Consider the summation of the size $2N \times 2N$ matrices, it means the matrices are created from samples $n = 0, 1, 2, \ldots, 2N\text{-}1$. It also means that matrices $A_1, A_2, A_3, \ldots, A_N$ at frequencies $k = 0$, $k = 1, k = 2, \ldots, k = N\text{-}1$ respectively, will be considered. Therefore the summation of $2N \times 2N$ matrix $\sum_{i=1}^{N} A_i$ is:

$$\sum_{i=1}^{N} A_i = \begin{pmatrix} N & \sum_{k=0}^{N-1} e^{-jk(\pi/N)} & 0 & \cdots & \sum_{k=0}^{N-1} e^{-j(2N-1)k(\pi/N)} \\ \sum_{k=0}^{N-1} e^{jk(\pi/N)} & N & \sum_{k=0}^{N-1} e^{-jk(\pi/N)} & \cdots & 0 \\ 0 & \sum_{k=0}^{N-1} e^{jk(\pi/N)} & N & \cdots & \sum_{k=0}^{N-1} e^{-j(2N-3)k(\pi/N)} \\ \sum_{k=0}^{N-1} e^{j3k(\pi/N)} & 0 & \sum_{k=0}^{N-1} e^{jk(\pi/N)} & & 0 \\ 0 & \sum_{k=0}^{N-1} e^{j3k(\pi/N)} & 0 & & \sum_{k=0}^{N-1} e^{-j(2N-5)k(\pi/N)} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \sum_{k=0}^{N-1} e^{j(2N-1)k(\pi/N)} & 0 & \sum_{k=0}^{N-1} e^{j(2N-3)k(\pi/N)} & & N \end{pmatrix} \tag{9}$$

The determinant of the $2N \times 2N$ matrix $\sum_{i=1}^{N} A_i$ is similar to the 4×4 matrix $\sum_{i=1}^{2} A_i$. Therefore it can be seen that increasing the number of sample n, increases the number of the constraints. Although n is increased, the elements of the $2N \times 2N$ summation matrix have the same character as the 4×4 summation matrix. It has a periodical characteristic due to the elements are formed from complex Fourier exponentials. Therefore most of the elements can be cancelled out by each other. Then the determinant of the matrix (9) can be calculated in the same manner as equation (6) and (7).

Hence, the determinant of the matrix $A_1 + A_2 + A_3 + \ldots + A_N$, equation (9), is defined

$$\Rightarrow \qquad \lambda^N (\lambda - 2N)^N = 0$$

Therefore the eigenvalues are

$$\lambda_1 = \lambda_2 = \ldots = \lambda_N = 0 \qquad \text{and} \qquad \lambda_{N+1} = \lambda_{N+2} = \ldots = \lambda_{2N} = 2N.$$

It can be written in the form

$$\lambda(A_1 + A_2 + \ldots + A_N) = \lambda\left(\sum_{i=1}^{N} A_i\right) = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 2N \\ 2N \\ \vdots \\ 2N \end{pmatrix}$$

Thus, the eigenvalues of the matrix $(\boldsymbol{A}_1 + \boldsymbol{A}_2 + \boldsymbol{A}_3 + \ldots + \boldsymbol{A}_N)$ is defined. //