NEW ADAPTIVE ALGORITHMS FOR LINEAR FILTERING AND NONLINEAR PREDICTION

by

Yasin Yılmaz

A Thesis Submitted to the
Graduate School of Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Electrical & Computer Engineering

Koç University

July, 2010

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Yasin Yılmaz

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

Asst. Prof. S. Serdar Kozat

Assoc. Prof. Alper T. Erdoğan

Assoc. Prof. Metin Türkay

Date:

To my family

ABSTRACT

In this thesis, we consider two adaptive filtering tasks: *Linear Adaptive Filtering* and *Nonlinear Adaptive Prediction*. We handle *system identification* and *sequential (online) nonlinear prediction* problems for these tasks respectively. 3 novel adaptive algorithms (2 in linear filtering and 1 in nonlinear prediction) are presented in this thesis.

For linear adaptive filtering, *Least Mean Squares (LMS)* is a fundamental, simple yet not fast enough converging algorithm. Proportionate update idea that is proposed by Duttweiler in [1] achieves a significant development in the convergence speed of LMS for sparse systems. We develop the *Proportionate Normalized Least Mean Fourth (PNLMF)* algorithm by minimizing mean fourth error (MFE) in the same way as [2] produces the *Least Mean Fourth (LMF)* algorithm from LMS. Yukawa's implementation of a Krylov subspace projection technique into the problem extends the use of proportionate update idea to non-sparse systems. We exploit the same Krylov subspace projection technique and introduce the *Krylov-Proportionate Normalized Least Mean Fourth (KPNLMF)* algorithm by again minimizing MFE. The Krylov-Proportionate Normalized Least Mean Squares (KPNLMS) algorithm minimizes mean square error (MSE) and our introduced KPNLMF algorithm is the MFE counterpart of KPNLMS. We observe the same relation between KPNLMS and KPNLMF as the one between LMS and LMF that is presented in [2]. Simulations show that KPNLMF attains a much lower mismatch (filter weight error power) than KPNLMS when the system noise has a probability density function among certain types that are of practical importance. It is also shown in the simulations that KPNLMF converges faster than the Normalized LMF (NLMF) algorithm. Another contribution of this thesis is that the *steady-state MSE analysis* is performed both for KPNLMS and KPNLMF. They are both shown theoretically to converge to the desired solution according to the steady-state MSE criterion.

In the second main part of the thesis, we deal with the sequential nonlinear prediction of an arbitrary, deterministic and bounded signal from its noise-corrupted past samples under square error loss. We present a novel randomized sequential prediction algorithm and name it *Competitive Randomized Noisy Nonlinear Predictor (CRNNP).* The main contribution of this thesis in this topic is that the CRNNP algorithm achieves a high prediction performance under additive noise. Since our CRNNP algorithm works for an arbitrary deterministic signal, we introduce a competitive framework in order to define a meaningful performance measure. CRNNP works in a competition class of algorithms that hypothetically work in parallel. We show that CRNNP achieves the performance of the best algorithm that can both select the best partition of the past observations space and the affine model parameters based on the desired clean signal in hindsight. The competition class is the class of certain nonlinear models, i.e. piecewise affine models represented on a context-tree. So, we employ a context-tree structure to model the nonlinearity that exists in the problem. The CRNNP algorithm serves for sequential decision problem and it makes its decision by choosing a strategy from several number of strategies at each time in a randomized fashion. Randomization weights are determined according to the prediction performances of the strategies.

ÖZETÇE

Bu tezde iki uyarlanır süzgeçleme işini ele alıyoruz: *Doğrusal Uyarlanır Süzgeçleme* ve *Doğrusal Olmayan Uyarlanır Öngörü.* Bu işleri sırasıyla *sistem tanımlama* ve *ardışık doğrusal olmayan öngörü* problemleri üzerinde düşünüyoruz. Bu tezde 3 adet yeni uyarlanır algoritma (2 adet doğrusal süzgeçleme için ve 1 adet doğrusal olmayan öngörü için) sunuluyor.

Doğrusal süzgeçlemede, *LMS (Least Mean Squares)* algoritması temel bir algoritma olup basit bir çalışma prensibi vardır ancak yeterince hızlı yakınsama yapmaz. [1]'de Duttweiler tarafından önerilen orantılı güncelleme fikri LMS algoritmasının yakınsama hızında seyreltik sistemler için önemli bir gelişme sağlar. Biz bu tezde [2]'nin LMS algoritmasından *LMF (Least Mean Fourth)* algoritmasını ürettiği yolu takip edip hatanın dördüncü kuvvetinin ortalamasını küçülterek *PNLMF (Proportionate Normalized LMF)* algoritmasını üretiyoruz. Yukawa'nın [3]'de probleme bir Krylov altuzayı izdüşüm tekniğini dahil etmesi orantılı güncelleme fikrini syreltik olmayan ayırgan sistemler için de kullanılabilir hale geitrir. Bu tezde, aynı Krylov altuzay izdüşüm tekniğini kullanıp yine hatanın dördüncü kuvvetinin ortalamasını küçülterek *KPNLMF (Krylov-Proportionate NLMF)* algoritmasını sunuyoruz. Burda, [2]'deki LMS ile LMF arasındaki ilişkinin aynısını KPNLMS ile KPNLMF arasında da gözlemliyoruz. Benzetimler, KPNLMF algoritmasının tatbiki önem içeren farklı olasılık yoğunluk fonksiyonlarına sahip gürültüler altında KPNLMS algoritmasından daha iyi çalıştığını gösteriyor. Benzetimlerde ayrıca KPNLMF algoritmasının başarımının NLMF algoritmasının başarımından üstün olduğu gösteriliyor. Bu tezin bir diğer katkısı KPNLMS ve KPNLMF algoritmaları için yatışkın durum ortalama karesel hata analizi gerçekleştirmesi. İki algoritmanın da kuramsal olarak yatışkın durum ortalama karesel hata kıstasına göre istenilen sonuca yakınsama yaptıkları kanıtlanıyor.

Tezin ikinci kısmında herhangi bir sınırlı, gerçek değerli ve belirlenimci sinyalin gürültüyle bozulmuş geçmiş örneklerinden karesel hata kayıp fonksiyonuyla ardışık doğrusal olmayan öngörülmesi ele alınıyor. *CRNNP* adında yeni bir rasgeleleştirilmiş ardışık öngörücü algoritma sunuyoruz. Bu tezin bu konudaki ana katkısı toplanır gürültü altında yüksek öngörü

başarımına ulaşmasıdır. CRNNP algoritması herhangi bir belirlenimci sinyalle çalışabildiği için anlamlı bir başarım ölçüsü tanımlamak adına yarışmacı bir çerçeve sunuyoruz. CRNNP, içindeki algoritmalar varsayımsal olarak paralel çalışan bir yarışma sınıfında işlem görüyor. CRNNP ulaşılmak istenen temiz sinayli kullanarak hem geçmiş gözlem uzayındaki en iyi bölümlemeyi hem de ilgin model parametrelerini seçebilen, yarışma sınıfındaki en iyi algoritmanın başarımına erişiyor. Yarışma sınıfı, bir bağlam ağacı yapısında temsil edilen parçalı ilgin modellerdir. Bu yüzden problemin doğasında varolan doğrusal olmamayı modellemek için bağlam ağacı yapısı kullanıyoruz. CRNNP algoritması ardışık karar verme problemine hizmet ediyor ve kararlarını her zaman belli sayıdaki yöntemden birini rasgeleleştirilmiş bir şekilde seçerek veriyor. Rasgeleleştirme ağırlıkları yöntemlerin öngörü başarımına dayalı olarak belirleniyor.

ACKNOWLEDGMENTS

Firstly, I would like thank my advisor Prof. Serdar Kozat who has been a great source of inspiration and provided the right balance of suggestions, criticism, and freedom.

I am grateful to Prof. Alper Erdoğan and Prof. Metin Türkay who are the other members of my thesis committee and Prof. Alper Demir for their valuable comments.

I thank The Scientific and Technological Research Council of Turkey (TÜBİTAK) for financial support.

I would like thank my housemates Yunus Emre Has, who has been with me for 14 years, Kutal Gökçe and Can Yağlı, and my officemates Mehmet Ali Yatbaz and Nergiz Kalantarova for their invaluable friendships.

Finally, my special thanks go to my family. I am thankful to my father İsmail, my mother Hava, my sisters Elif and Gülsüm, my nephews Ömer, Ahmet, Furkan and Kerim, and at last to my beloved wife Filiz for providing me morale support and motivation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

MSE	Mean Square Error
MFE	Mean Fourth Error
LMS	Least Mean Square
LMF	Least Mean Fourth
NLMS	Normalized Least Mean Square
NLMF	Normalized Least Mean Fourth
PNLMS	Proportionate Normalized Least Mean Square
PNLMF	Proportionate Normalized Least Mean Fourth
IPNLMS	Improved Proportionate Normalized Least Mean Square
KPNLMS	Krylov Proportionate Normalized Least Mean Square
KPNLMF	Krylov Proportionate Normalized Least Mean Fourth
GCR	Generalized Conjugate Residual
KPNLMMN	Krylov Proportionate Normalized Least Mean Mixed Norm
CRNNP	Competitive Randomized Noisy Nonlinear Predictor

Chapter 1

INTRODUCTION

Filtering, in general, means applying a device which can be a piece of physical hardware or software to a set of noisy data in order to extract information about a quantity of interest. The noise may arise from different sources like noisy sensors, a communication channel, etc. Under any circumstances, filtering may help us to perform three basic information-processing tasks [4]:

- *Filtering*: Extraction of information about a quantity of interest at time t by using data measured up to and including time t.

- *Smoothing*: Data measured after time t can be used to obtain information about the quantity of interest at time t. So, there is a delay in producing the result of interest in the case of smoothing.

- *Prediction*: Forecasting side of information processing is present here. The goal of prediction is to derive information about a future value of the quantity of interest at some time like $t + \tau$, where $\tau > 0$, by exploiting data measured up to and including time t.

If the output quantity of the filter is a linear function of the observations applied to the filter input, then the filter is said to be *linear*. Otherwise, it is *nonlinear*. In this thesis, two of the above tasks are studied. Novel algorithms for *linear filtering* and *nonlinear prediction* are proposed and detailed explanations are supplied in the following chapters.

From the statistical viewpoint to the solution of the linear filtering problem, certain statistical parameters such as mean and correlation functions of the useful signal and unwanted additive noise are assumed to be available. The goal is to minimize the effects of noise at the filter output according to some statistical criterion. Minimizing the mean-square value of the error signal, which is defined as difference between some desired response and the actual filter output, is a useful and common approach. The resulting solution for stationary inputs is known as the *Wiener filter* which is optimum in the mean-square sense. When dealing with situations in which nonstationarity exists, the optimum filter has to assume a time-varying form. *Kalman filter* exhibits a highly successful solution to this more difficult problem [5].

The statistics of the data to be processed are required for the design of the Wiener filter. However, for most real-world applications the statistics of the input data are not attainable. An efficient method to overcome this problem is to use an *adaptive filter*. An adaptive filter is a device that is self-designing in that it relies for its operation on a recursive algorithm. The algorithm starts from some predetermined set of initial conditions. These initial conditions express whatever we know about the environment. If working in a stationary environment, the algorithm converges to the optimum Wiener solution in some statistical sense after successive iterations. In a nonstationary environment, the algorithm offers a tracking capability. So, it can track time variations in the statistics of the input data, provided that the variations are slow enough.

In the analysis of linear adaptive filters, the following factors are commonly considered:

- *Rate of convergence*: In response to the stationary inputs, it is defined as the number of iterations required for the algorithm to converge to the optimum Wiener solution in the mean-square sense. An algorithm with a fast rate of convergence adapts rapidly to a stationary environment of unknown statistics.

- *Misadjustment*: It provides a quantitative measure of the amount by which the final value of the mean-squared error, averaged over an ensemble of adaptive filters, deviates from the minimum mean-squared error that is produced by the Wiener filter.

- *Tracking*: An adaptive algorithm is required to track the statistical variations in the environment when operating in a nonstationary environment.

- *Computational requirements*: Here, the issues of concern include the number of operations required to make one complete iteration of the algorithm, the size of the memory locations required to store the data and the program, and the investment required to program the algorithm on a computer. However, we basically deal with *computational complexity* in this thesis.

- *Robustness*: Disturbances with small energy should only result in small estimation errors. The disturbances may arise from a variety of internal or external factors.

In this thesis, we analyze our novel linear adaptive filtering algorithms in terms of the first four factors listed above. These factors, in their own ways, enter into the design of nonlinear adaptive filters. Nevertheless, we no longer have a well-defined frame of reference in the form of a Wiener filter. Rather, we talk about rate of convergence by looking at convergence curves and computational complexity of the algorithm.

1.1 Linear Adaptive Filtering

There is no unique solution to the linear adaptive filtering problem. Rather, we have some tools represented by a variety of adaptive algorithms, each of which offers desirable features of its own. Firstly, the user of adaptive filters should understand the capabilities and limitations of various adaptive filtering algorithms. Then, this understanding should be used in the selection of the appropriate algorithm for the application at hand.

Basically, we may identify two different approaches, namely *stochastic gradient approach* and *least-squares estimation*, for deriving recursive algorithms for the operation of linear adaptive filters [4].

Stochastic gradient approach uses the mean-squared error as the cost function for the case of stationary inputs. The dependence of the mean-squared error on the unknown filter coefficients may be viewed to be in the form of a multidimensional paraboloid with a uniquely defined minimum point. We refer to this paraboloid as the error-performance surface. The filter coefficients corresponding to the minimum point of the surface define the optimum Wiener solution. We proceed in a two-stage manner in order to develop a recursive algorithm for updating the filter coefficients of the adaptive filter. We first use the *method of steepest descent*, a well-known optimization technique, to modify the system of *Wiener-Hopf equations* (i.e., the matrix equation defining the optimum Wiener solution). This modification requires the use of a gradient vector. The value of a gradient vector depends on two parameters: the *correlation matrix* of the inputs in the filter, and the *cross-*

correlation vector between the desired response and the same inputs. In general, these two parameters are not available to us, that is why we use adaptive filtering algorithms instead of using the optimum Wiener filter. So, to derive an estimate for the gradient vector, we next use instantaneous values for these correlations. The resulting algorithm is widely known as the *least-mean-square (LMS) algorithm.* The essence of the LMS algorithm can be described in words as follows for the case of a transversal filter operating on real-valued data:

$$\left(\begin{array}{c}\text{updated value of}\\ \text{filter coefficient}\\ \text{vector}\end{array}\right)=\left(\begin{array}{c}\text{old value of}\\ \text{filter coefficient}\\ \text{vector}\end{array}\right)+\left(\begin{array}{c}\text{learning-}\\ \text{rate}\\ \text{parameter}\end{array}\right)\left(\begin{array}{c}\text{input}\\ \text{vector}\end{array}\right)\left(\begin{array}{c}\text{error}\\ \text{signal}\end{array}\right)$$

where the error signal is defined as the difference between some desired response and the actual filter response. Although, the LMS is simple, it is capable of achieving satisfactory performance under the right conditions. Its major limitations are a relatively slow rate of convergence and a sensitivity to variations in the condition number of the correlation matrix of the input vector. Within a nonstationary environment, the orientation of the error-performance surface varies continuously with time. In this case, the LMS algorithm has the additional task of continually tracking the bottom of the error-performance surface as long as the input data vary slowly compared to the learning rate of the LMS algorithm.

The LMS algorithm was devised by Widrow and Hoff in 1959 during their study of a pattern recognition scheme. The LMS algorithm is closely related to the concept of *stochastic approximation* developed by Robbins and Monro (1951) in statistics for solving certain sequential parameter estimation problems. The primary difference between them is that the LMS algorithm uses a fixed learning-rate parameter to update each filter coefficient, whereas in stochastic approximation methods the learning-rate parameter is made inversely proportional to time t or to a power of t. *Gradient adaptive lattice (GAL) algorithm* is another stochastic gradient algorithm that is closely related to the LMS algorithm (Griffiths, 1977,1978). The difference between them is structural in that the GAL algorithm is lattice-based, whereas the LMS algorithm uses a transversal filter. In 1981, Zames introduced the H^∞ *norm* (or minimax criterion) as a robust index of performance for solving problems in estimation and control. In this context, it is particularly noteworthy that Hassibi et al. (1996) have shown that the LMS algorithm is indeed optimal under the H^∞ criterion [6].

Thus, for the first time, theoretical evidence was presented for the robust performance of the LMS algorithm.

Least-squares estimation is the second approach to the development of linear adaptive filtering algorithms. It is based on the **method of least squares**. Here, we minimize a cost function that is defined as the *sum of weighted error squares. Recursive least-squares (RLS) estimation* is the most popular way to formulate the method of least-squares. The RLS estimation may be viewed as a special case of *Kalman filtering.* A distinguishing feature of the Kalman filtering is the notion of *state*, which provides a measure of all the inputs applied to the filter up to a specific instant of time. So, we may describe in words the recursion that the Kalman filtering has as follows:

$$\begin{pmatrix} \text{updated value} \\ \text{of the} \\ \text{state} \end{pmatrix} = \begin{pmatrix} \text{old value} \\ \text{of the} \\ \text{state} \end{pmatrix} + \begin{pmatrix} \text{Kalman} \\ \text{gain} \end{pmatrix} \begin{pmatrix} \text{innovation} \\ \text{vector} \end{pmatrix}$$

where the *innovation vector* represents new information put into the filtering process at the time of the computation. There is indeed one-to-one correspondence between the Kalman variables and RLS variables [5]. This correspondence means that we can use the vast literature on Kalman filters for the design of linear adaptive filters based on recursive least-squares estimation. We may classify the RLS family of linear adaptive filtering algorithms into three distinct categories: *Standard RLS algorithm, Square-root RLS algorithms, Fast RLS algorithms.* This classification depends on the approach taken. However, we will not go into the details of these different approaches to the RLS algorithm since our attention is on the LMS family of linear adaptive filtering algorithms.

The original paper on the *standard RLS algorithm* belongs to Plackett (1950). On the other hand, it must be said that many other investigators have derived and rederived the RLS algorithm. In 1974, Godard used Kalman filter theory to derive a variant of the RLS algorithm, which is also referred to in the literature as the *Godard algorithm.* Then, Sayed and Kailath (1994) published published an enlightening paper, in which the exact relationship between the RLS algorithm and Kalman filter theory was described for the first time [7]. Therefore, this publication of Sayed et al. presented the groundwork for how to exploit the vast literature on Kalman filters for solving linear adaptive filtering problems.

1.2 Nonlinear Adaptive Prediction

Linear prediction and linear predictive models have long been central themes within the signal processing literature [8]. More recently, nonlinear models based on piecewise linear [9] and locally linear [10] approximations have gained significant attention. Nonlinear adaptive prediction is a task that is carried out by a nonlinear adaptive filter. A nonlinear filter is a signal processing device whose output is not a linear function of its input as we described earlier. According to Haykin [4], there are fundamentally two types of nonlinear adaptive filters, namely Volterra-based nonlinear adaptive filters and neural networks.

The first type of nonlinear adaptive filters mentioned here relies on the use of a *Volterra series* that provides an attractive method for describing the input-output relationship of nonlinear device with a memory. This special form of series derives its name from Vito Volterra who studied it first in 1880 as a generalization of the Taylor series of a function. Norbert Wiener was the first to use the Volterra series to model the input-output relationship of a nonlinear system in 1958. Schetzen's book discusses the Volterra series in detail [11]. In 1989, Rayner and Lynch also studied nonlinear adaptive filter based on Volterra series [12], [13].

An *artificial neural network* or a *neural network* as it is commonly called, is a collection of a large number of interconnected nonlinear processing units called neurons. In other words, the nonlinearity is distributed throughout the network. The way the human brain performs its operations motivated the development of neural networks. [14] is an example of the usage of neural networks for nonlinear adaptive prediction.

In addition to the Haykin's classification, tree-structured nonlinear prediction can be seen as the third type of nonlinear adaptive filters. [9] is an example paper which studies tree-structured nonlinear adaptive prediction. Kozat et al. uses piecewise linearity via context trees to model the nonlinearity for the nonlinear prediction problem [15]. In the computational learning theory literature, the related problem of prediction as well as the best pruning of a decision tree has been considered in which data structures and algorithms similar to context trees have been used [16], [17], [18]. Our new nonlinear prediction algorithm is also a member of the third type. We exploit piecewise linear models to do nonlinear prediction under additive noise.

1.3 Contributions

The contributions of this thesis are in two different topics. We propose in total 3 novel adaptive algorithms: 2 for *linear filtering* and 1 for *nonlinear prediction.*

Contributions to linear filtering:

- 2 new fast-converging algorithms in the Least-Mean-Fourth (LMF) family of algorithms

- Superior performance in uniformly, sine wave type and square wave type distributed system noise cases

- Steady-state MSE analysis for the Krylov-Proportionate NLMS and the Krylov-Proportionate NLMF algorithms.

Contributions to nonlinear prediction:

- A novel sequential nonlinear predictor that makes no stochastic assumptions and that works for sequential decision problems

- High prediction performance under additive noise

1.4 Outline

We present here an outline of the thesis. In Chapter 2, two novel adaptive algorithms namely, *Proportionate Normalized Least-Mean-Fourth (PNLMF)* and *Krylov-proportionate Normalized Least-Mean-Fourth (KPNLMF)* algorithms for linear filtering problem are introduced. Derivation of the introduced algorithms, steady-state mean square error (MSE) analysis for the algorithms and the simulation results are provided in the sections of the Chapter 2.

Chapter 3 concentrates on the nonlinear prediction problem and it suggests another adaptive algorithm, namely *Competitive Randomized Noisy Nonlinear Predictor (CRNNP)* in a competitive framework. Its sections point out the algorithm description and the simulation results.

Finally, Chapter 4 summarizes the work that has been done and refer to the possible future directions.

1.5 Notation

In this thesis, all vectors are column vectors represented by boldface lowercase letters. Matrices are represented with boldface capital letters. We reserve letters that are not boldface to random variables. Time index appears as an argument within brackets, e.g. $d[t]$. For a time-invariant vector $\boldsymbol{x}$, $\boldsymbol{x}(i)$ is the ith entry of the vector. For a vector with time index $\boldsymbol{x}[t]$, $\boldsymbol{x}[t]^{(i)}$ is the ith entry. For a random variable x (or vector $\boldsymbol{x}$), $E\left[x\right]$ (or $E\left[\boldsymbol{x}\right]$) is the expectation. $(\cdot)^T$ is the transpose operation, $\|\cdot\|$ is the l_2-norm and $|\cdot|$ is the absolute value.

Chapter 2

LINEAR ADAPTIVE FILTERING TECHNIQUES BASED ON KRYLOV SUBSPACE PROJECTION METHOD

In this chapter, we introduce two novel linear adaptive filtering algorithms and investigate their performances for the system identification task. System identification task is chosen for the clearness of the problem statement. The first algorithm we introduce is the proportionate normalized least-mean-fourth (PNLMF) algorithm. While developing the PNLMF algorithm, we are inspired by the proportionate normalized least-mean-square (PNLMS) algorithm presented by Duttweiler in [1]. The PNLMF algorithm is the mean fourth error version of the PNLMS algorithm. The PNLMS algorithm has been proposed for the identification of sparse systems. It is known to exhibit faster convergence than the standard NLMS in certain setups [1]. We derive the PNLMF algorithm from the PNLMS algorithm in the same way as to obtain the LMF algorithm from the LMS algorithm. We note that for the derivation of the PNLMF algorithm, the improved PNLMS (IPNLMS) algorithm [19] is going to be used instead of the direct PNLMS algorithm of [1].

The second algorithm presented in this chapter is the Krylov-proportionate normalized least-mean-fourth (KPNLMF) algorithm. Here, Krylov subspace projection technique is incorporated within the framework of the PNLMF algorithm. The Krylov-proportionate normalized least-mean-square (KPNLMS) algorithm introduced in [3] extends the use of the PNLMS algorithm to the identification of dispersive systems by benefiting from the Krylov subspace projection technique. The KPNLMF algorithm inherits the advantageous features of KPNLMS for dispersive systems [3]. In addition, the KPNLMF algorithm is shown to outperform the KPNLMS algorithm in certain setups. An early version of this work was presented at a conference [20].

Steady-state performances of both KPNLMS and KPNLMF are analyzed in terms of MSE criterion in Section 2.3. We deal with stochastic signals, i.e., our input, output signals and noise in the system are modeled as stochastic signals and in realistic applications the statistics of the input and output signals are not known exactly. Therefore, we approximate the method of steepest descent [4] which employs deterministic gradient method by using stochastic gradient algorithms. All adaptive algorithms discussed in this chapter (LMF, NLMS, NLMF, PNLMS, IPNLMS, PNLMF, KPNLMS, KPNLMF) are members of the family of stochastic gradient algorithms as they are derived from the Least Mean Squares (LMS) algorithm which is the most popular member of the stochastic gradient family [4]. In our stochastic gradient assumptions, exact covariance and cross-correlation quantities are replaced by instantaneous estimates resulting in gradient noise. As a consequence, stochastic gradient algorithms perform worse than the original steepest descent method. However, the degradation in the performance of LMS is not significant when a suitable step-size is used [4]. Since our adaptive algorithms studied in this chapter are children of LMS, we also expect an insignificant degradation in their performances. Results of Section 2.3 coincide with what we expect here. KPNLMS and KPNLMF are shown to have small excess MSEs proving that their performances are very similar to the performance of the original steepest descent algorithm.

The system identification problem is studied throughout the chapter. We particularly investigate system identification framework since signal processing problems like noise canceling, echo canceling and channel equalization share the same system setup with system identification. In this framework, an unknown system is modeled adaptively by minimizing a certain statistical measure of the error between the output of the system to be identified and the output of the model system. We emphasize that although minimizing the mean square error (MSE) is the most widely known and used technique because of its tractability and simple analysis, there are other ways to minimize the estimation error. Expected value of the fourth power of the error is a popular alternative to minimize in linear adaptive filtering. Modified steepest descent algorithm for the mean fourth error case is studied and the least mean forth (LMF) algorithm is proposed as an adaptive filtering technique in [2]. The PNLMF and KPNLMF algorithms are derived starting from the LMF algorithm in Section 2.2. In Section 2.3, steady-state MSE analysis of KPNLMS and KPNLMF are carried out. Section 2.4 contains the simulation results for the sample cases, followed by the conclusion in Section 2.5.

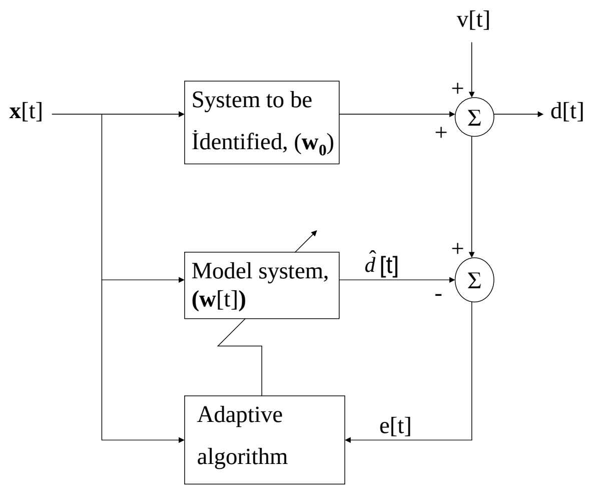


Figure 2.1: Block diagram of system identification

2.1 System Description

We consider the system identification task presented in Fig. 2.1. In this figure, $\boldsymbol{x}[t] \in \mathbb{R}^m$ is the input regressor with zero mean and covariance matrix $\boldsymbol{R} = E\left[\boldsymbol{x}[t]\boldsymbol{x}[t]^T\right]$. With this input regressor, the output of the desired unknown system is given by

$$d[t] = \boldsymbol{w}_o^T \boldsymbol{x}[t] + v[t],\ t \in \mathbb{N}, \tag{2.1}$$

where $\boldsymbol{w}_0 \in \mathbb{R}^m$ is the coefficient vector of the unknown system to be identified. Here, $v[t]$ is the i.i.d. noise with zero mean and variance σ_v^2. We assume that the input regressor and the noise signal are uncorrelated. The input regressor $\boldsymbol{x}[t]$ and the output signal $d[t]$ are available to estimate the filter coefficients of the unknown system. Given the input regressor, the estimate of the desired signal is given by

$$\hat{d}[t] = \boldsymbol{w}[t]^T \boldsymbol{x}[t],\ t \in \mathbb{N}, \tag{2.2}$$

where $\boldsymbol{w}[t] = [\boldsymbol{w}[t]^{(1)}, \boldsymbol{w}[t]^{(2)}, ..., \boldsymbol{w}[t]^{(m)}]^T$ is the adaptive weight vector to estimate $\boldsymbol{w}_o$.

In this framework, our aim is to minimize a specific statistical measure of the error between the desired signal $d[t]$ and the estimate produced by an adaptive algorithm $\hat{d}[t]$, i.e., $e[t] = d[t] - \hat{d}[t]$. In this chapter, we minimize the mean fourth power of the estimation error. The mean square error (MSE) and the mean fourth error (MFE) are given by

$$\begin{aligned} \text{MSE} &= E\left[e[t]^2\right] \\ \text{MFE} &= E\left[e[t]^4\right], \end{aligned} \tag{2.3}$$

respectively. Given this setup, in the next section we introduce two different adaptive algorithms that are constructed based on the MFE criteria using proportionate update idea and Krylov subspace projections.

Here, $\boldsymbol{R} = E\left[\boldsymbol{x}[t]\boldsymbol{x}[t]^T\right]$ is the autocorrelation matrix of the input regressor $\boldsymbol{x}[t]$ and $\boldsymbol{p} = E\left[d[t]\boldsymbol{x}[t]\right]$ is the cross-correlation vector between the input regressor $\boldsymbol{x}[t]$ and the output $d[t]$ at time t. Both $\boldsymbol{R}$ and $\boldsymbol{p}$, which are statistical measures of the input and the output, can be thought as known parameters for the exactness of the algorithm while developing it. Otherwise, they can be approximated at the beginning of the algorithm. We assume they are known beforehand for the sake of simplicity in the next section. The fact that we really do not know exact $\boldsymbol{R}$ and $\boldsymbol{p}$ is considered in Section 2.4 while simulating the algorithms.

2.2 Proposed Adaptive Filtering Algorithm

In this section, we first derive the update procedure for the filter coefficients of the PNLMF algorithm inspired from the IPNLMS algorithm. Then, Krylov subspace projection technique is incorporated within the PNLMF algorithm framework to yield the KPNLMF algorithm. We note that this order of constructing the corresponding algorithms is for the clarity of the presentation. This order of derivation is also the chronological order of the already developed algorithms following [1,3]. However, the logical flow of the proposed algorithms are in the reverse direction. The unknown system is thought to be projected onto the Krylov subspace in order to attain sparseness, then the PNLMF algorithm is applied to the resulting system. A better intuition is tried to be given in the following subsections.

2.2.1 Derivation of the PNLMF algorithm

The standard LMS algorithm updates the filter coefficients as

$$\boldsymbol{w}[t+1] = \boldsymbol{w}[t] + \mu e[t]\boldsymbol{x}[t], \tag{2.4}$$

where $e[t]\boldsymbol{x}[t]$ approximates the gradient $\nabla_{\boldsymbol{w}[t]} E\left[e[t]^2\right]$. Here, μ is the step size of the update, usually constrained to be $0 < \mu < \frac{2}{mE[x[t]^2]}$, where m is the filter length. The normalized LMS (NLMS) algorithm is introduced in order to alleviate the dependence of the standard LMS algorithm on the statistics of the input data. The NLMS algorithm uses the update equation

$$\boldsymbol{w}[t+1] = \boldsymbol{w}[t] + \mu e[t] \frac{\boldsymbol{x}[t]}{\|\boldsymbol{x}[t]\|^2 + \epsilon}, \tag{2.5}$$

where ϵ is a small regularization factor. In [2], instead of using the approximate gradient of $E\left[e[t]^2\right]$, the approximate gradient of $E\left[e[t]^4\right]$ is used to construct the update for the LMF algorithm as

$$\boldsymbol{w}[t+1] = \boldsymbol{w}[t] + 2\mu e[t]^3 \boldsymbol{x}[t], \tag{2.6}$$

where μ is the same step size as in the LMS algorithm. Following this, the NLMF algorithm readily turns out to be

$$\boldsymbol{w}[t+1] = \boldsymbol{w}[t] + 2\mu e[t]^3 \frac{\boldsymbol{x}[t]}{\|\boldsymbol{x}[t]\|^2 + \epsilon} \tag{2.7}$$

as stated in [21]. We emphasize that using the fourth power of the error signal in the update, the LMF algorithm is shown to yield faster convergence in the start of the adaptation.

We next incorporate the proportional update rule [19] into the NLMF framework. The motivation behind the IPNLMS algorithm is to update each filter coefficient individually with different step sizes. Each filter coefficient is updated according to the absolute value of the current filter coefficient in a proportional manner, where the name *proportional* stems from. In this sense, by using an update proportional to the absolute value of the current filter coefficients, the IPNLMS algorithm distinguishes between frequently used, rarely used or unused coefficients and updates them separately as follows:

$$\begin{aligned}
\boldsymbol{w}[t+1] &= \boldsymbol{w}[t] + \mu e[t] \frac{\boldsymbol{G}[t]\boldsymbol{x}[t]}{\boldsymbol{x}[t]^T \boldsymbol{G}[t] \boldsymbol{x}[t] + \epsilon} \\
\boldsymbol{G}[t] &= \operatorname{diag}(\boldsymbol{\phi}[t]^{(1)}, \boldsymbol{\phi}[t]^{(2)}, ..., \boldsymbol{\phi}[t]^{(m)}) \\
\boldsymbol{\phi}[t]^{(k)} &= (1-\gamma)\frac{1}{m} + \gamma \frac{|\boldsymbol{w}[t]^{(k)}|}{\|\boldsymbol{w}[t]\|_1 + \chi}, t \in \mathbb{N}, k \in [1, ..., m],
\end{aligned} \tag{2.8}$$

where γ is the proportionality factor and χ is a small regularization constant [19]. This idea of using a proportional update is applied to the NLMF algorithm following the IPNLMS algorithm. For the PNLMF algorithm, the matrix multiplying the individual filter coefficients, i.e., $\boldsymbol{G}[t]$, remains unchanged while only the first equation in (2.8) is changed to:

$$\boldsymbol{w}[t+1] = \boldsymbol{w}[t] + 2\mu e[t]^3 \frac{\boldsymbol{G}[t]\boldsymbol{x}[t]}{\boldsymbol{x}[t]^T\boldsymbol{G}[t]\boldsymbol{x}[t] + \epsilon}. \tag{2.9}$$

In the next subsection, we extend the PNMLF algorithm using Krylov subspaces.

2.2.2 Projection over Krylov subspace

It has been shown in [19] through simulations that the IPNLMS algorithm achieves superior performance than the NLMS algorithm in the system identification task when the underlying unknown channel is sparse. We project the unknown system, which is potentially dispersive, to the Krylov subspace to get benefit of the nice feature of the IPNLMS algorithm upon sparse systems. In [3], the author demonstrates that projecting the impulse response of the unknown system into the Krylov subspace yields a sparse representation if the input regressor is nearly white, i.e., $E\left[\boldsymbol{x}[t]\boldsymbol{x}[t]^T\right] \approx \boldsymbol{I}$. To be more precise, if the autocorrelation matrix $\boldsymbol{R} = E\left[\boldsymbol{x}[t]\boldsymbol{x}[t]^T\right]$ of the input regressor $\boldsymbol{x}[t]$ has clustered eigenvalues or the autocorrelation matrix has a condition number close to one, then any unknown system will have a sparse representation at the new Krylov subspace coordinates. In case the input is not white, a preconditioning process can be applied to the input regressor before applying the algorithm. We explain in the next subsection how projecting an unknown system to the Krylov subspace yields a sparse system in a different and more illustrative way than that of [3]. Some preconditioning methods are also described in the next subsection.

Since we work at the new Krylov subspace coordinates where our unknown system has a sparse structure, the PNLMF algorithm is altered to work in the new coordinates. This algorithm will be called the Krylov PNLMF (KPNLMF) and has the update

$$\boldsymbol{w}[t+1] = \boldsymbol{w}[t] + 2\mu e[t]^3 \frac{\boldsymbol{Q}\boldsymbol{G}[t]\boldsymbol{Q}^T\boldsymbol{x}[t]}{\boldsymbol{x}[t]^T\boldsymbol{Q}\boldsymbol{G}[t]\boldsymbol{Q}^T\boldsymbol{x}[t] + \epsilon} \tag{2.10}$$

where the orthogonal matrix $\boldsymbol{Q}$ represents the Krylov subspace coordinates. The columns of the matrix $\boldsymbol{Q}$ form a set of orthonormal basis vectors for the Krylov subspace which is spanned by the Krylov vectors

$$\boldsymbol{p}, \boldsymbol{R}\boldsymbol{p}, \boldsymbol{R}^2\boldsymbol{p}, ..., \boldsymbol{R}^{m-1}\boldsymbol{p}. \tag{2.11}$$

The orthogonal matrix $\boldsymbol{Q}$ is formed by orthonormalizing the Krylov vectors in (2.11). This orthonormalization process can be performed via *Arnoldi's method*. Arnoldi's method is an effective way of orthonormalizing Krylov vectors since it does not generate Krylov vectors explicitly. The explicit generation of Krylov vectors is an ill-conditioned numerical operation. The well known *Gram-Schmidt method* does not help here as it first generates the Krylov vectors and then orthonormalizes them. We form $\boldsymbol{Q}$ once in the whole process of the algorithm. So, it does not bring much computational burden. Note that since we now work in the space spanned by the columns of $\boldsymbol{Q}$, the step size scaling matrix, $\boldsymbol{G}[t]$ is updated accordingly. Defining the projected weight vector $\acute{\boldsymbol{w}}[t] = \boldsymbol{Q}^T\boldsymbol{w}[t]$, the new scaling matrix is defined as:

$$\begin{aligned}
\boldsymbol{G}[t] &= \text{diag}(\boldsymbol{\phi}[t]^{(1)}, \boldsymbol{\phi}[t]^{(2)}, ..., \boldsymbol{\phi}[t]^{(m)}) \\
\boldsymbol{\phi}[t]^{(k)} &= (1-\gamma)\frac{1}{m} + \gamma\frac{|\acute{\boldsymbol{w}}[t]^{(k)}|}{\|\acute{\boldsymbol{w}}[t]\|_1 + \chi}, \qquad t \in \mathbb{N}, \qquad k \in [1, ..., m],
\end{aligned} \tag{2.12}$$

Obtaining full $\boldsymbol{G}[t]^{m\times m}$ and forming $\boldsymbol{Q}\boldsymbol{G}[t]\boldsymbol{Q}^T$ with full $\boldsymbol{Q}^{m\times m}$ at every iteration is computationally expensive by using the formula for $\boldsymbol{G}[t]$ in (2.12). This iteration has a computational complexity of $O(m^2)$. However, fortunately we succeed to attain linear computational complexity per iteration. Before showing how we achieve the linear computational complexity, it will be beneficial to define new matrices $\boldsymbol{\Omega}[t] \triangleq \boldsymbol{Q}\boldsymbol{G}[t]\boldsymbol{Q}^T$, $\boldsymbol{Q}_\lambda^{m\times\lambda}$ as the first $\lambda(\ll m)$ columns of $\boldsymbol{Q}$ and $\boldsymbol{G}_\lambda[t]^{\lambda\times\lambda} = diag(\boldsymbol{\phi}[t]^{(1)}, ..., \boldsymbol{\phi}[t]^{(\lambda)})$. So, $\boldsymbol{Q} = [\boldsymbol{Q}_\lambda^{m\times\lambda}\boldsymbol{Q}_{m-\lambda}^{m\times(m-\lambda)}]$ and (2.10) becomes

$$\boldsymbol{w}[t+1] = \boldsymbol{w}[t] + 2\mu e[t]^3\frac{\boldsymbol{\Omega}[t]\boldsymbol{x}[t]}{\boldsymbol{x}[t]^T\boldsymbol{\Omega}[t]\boldsymbol{x}[t] + \epsilon}. \tag{2.13}$$

Assigning a pre-specified small constant value to the remaining $(m-\lambda)\boldsymbol{\phi}[t]^{(k)}$ values does not affect the convergence speed significantly since they correspond to the rarely used coefficients of the weight vector. So, if we assign a small constant value, ψ to the remaining $(m-\lambda)\boldsymbol{\phi}[t]^{(k)}$ values, the scaling matrix $\boldsymbol{G}[t]$ becomes

$$\tilde{\boldsymbol{G}}[t] = diag(\boldsymbol{\phi}[t]^{(1)}, ..., \boldsymbol{\phi}[t]^{(\lambda)}, \psi, ..., \psi) \tag{2.14}$$

If we look at the computation of the part $\boldsymbol{\Omega}[t]\boldsymbol{x}[t]$ in (2.13) replacing $\boldsymbol{G}[t]$ with the new scaling matrix $\tilde{\boldsymbol{G}}[t]$, we see that

$$\begin{aligned}
\tilde{\boldsymbol{\Omega}}[t]\boldsymbol{x}[t] &= [\boldsymbol{Q}_\lambda \quad \boldsymbol{Q}_{m-\lambda}] \begin{bmatrix} \boldsymbol{G}_\lambda[t] & \boldsymbol{0} \\ \boldsymbol{0} & \psi\boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{Q}_\lambda^T \\ \boldsymbol{Q}_{m-\lambda}^T \end{bmatrix} \boldsymbol{x}[t] \\
&= [\boldsymbol{Q}_\lambda\boldsymbol{G}_\lambda[t] \quad \psi\boldsymbol{Q}_{m-\lambda}] \begin{bmatrix} \boldsymbol{Q}_\lambda^T\boldsymbol{x}[t] \\ \boldsymbol{Q}_{m-\lambda}^T\boldsymbol{x}[t] \end{bmatrix} \\
&= [\boldsymbol{Q}_\lambda\boldsymbol{G}_\lambda[t]\boldsymbol{Q}_\lambda^T\boldsymbol{x}[t]]^{m\times 1} + [\psi\boldsymbol{Q}_{m-\lambda}\boldsymbol{Q}_{m-\lambda}^T\boldsymbol{x}[t]]^{m\times 1} \\
&= \boldsymbol{Q}_\lambda\boldsymbol{G}_\lambda[t]\boldsymbol{Q}_\lambda^T\boldsymbol{x}[t] - \psi\boldsymbol{Q}_\lambda\boldsymbol{Q}_\lambda^T\boldsymbol{x}[t] + \psi\boldsymbol{Q}_\lambda\boldsymbol{Q}_\lambda^T\boldsymbol{x}[t] + \psi\boldsymbol{Q}_{m-\lambda}\boldsymbol{Q}_{m-\lambda}^T\boldsymbol{x}[t] \\
&= \boldsymbol{Q}_\lambda(\boldsymbol{G}_\lambda[t]\boldsymbol{Q}_\lambda^T\boldsymbol{x}[t] - \psi\boldsymbol{Q}_\lambda^T\boldsymbol{x}[t]) + \psi\underbrace{(\boldsymbol{Q}_\lambda\boldsymbol{Q}_\lambda^T + \boldsymbol{Q}_{m-\lambda}\boldsymbol{Q}_{m-\lambda}^T)}_{\boldsymbol{I}}\boldsymbol{x}[t] \\
&= \boldsymbol{Q}_\lambda(\boldsymbol{G}_\lambda[t]\boldsymbol{Q}_\lambda^T\boldsymbol{x}[t] - \psi\boldsymbol{Q}_\lambda^T\boldsymbol{x}[t]) + \psi\boldsymbol{x}[t] \qquad (2.15)
\end{aligned}$$

Note that $\boldsymbol{Q}_{m-\lambda}$ is not used anywhere except in the computations of $\boldsymbol{\phi}[t]^{(k)}$ values. In (2.12), we need to compute $\acute{\boldsymbol{w}}[t] = \boldsymbol{Q}^T\boldsymbol{w}[t]$. However, only first λ entries of $\acute{\boldsymbol{w}}[t]$ are needed since only first λ components of $\boldsymbol{\phi}[t]^{(k)}$ are computed in our simplified algorithm. Subvector $\acute{\boldsymbol{w}}_\lambda[t]$ can be updated as follows.

$$\acute{\boldsymbol{w}}_\lambda[t+1] = \acute{\boldsymbol{w}}_\lambda[t] + 2\mu e[t]^3 \frac{\boldsymbol{G}_\lambda[t]\boldsymbol{Q}_\lambda^T\boldsymbol{x}[t]}{\boldsymbol{x}[t]^T\tilde{\boldsymbol{\Omega}}[t]\boldsymbol{x}[t] + \epsilon}. \qquad (2.16)$$

So, we do not need $\boldsymbol{Q}_{m-\lambda}$ anywhere. By computing only first λ components of $\boldsymbol{\phi}[t]^{(k)}$ values used in matrix $\tilde{\boldsymbol{G}}[t]$ and first λ columns of the orthogonal matrix $\boldsymbol{Q}$ we can attain linear computational complexity per iteration as showed in [3]. Then, we use the following formula in order to update filter coefficients in KPNLMF.

$$\boldsymbol{w}[t+1] = \boldsymbol{w}[t] + 2\mu e[t]^3 \frac{\tilde{\boldsymbol{\Omega}}[t]\boldsymbol{x}[t]}{\boldsymbol{x}[t]^T\tilde{\boldsymbol{\Omega}}[t]\boldsymbol{x}[t] + \epsilon} \qquad (2.17)$$

Here, computation of $\tilde{\boldsymbol{\Omega}}[t]\boldsymbol{x}[t]$ is given in (2.15), $\boldsymbol{Q}_\lambda$ is formed by orthonormalizing first λ Krylov vectors via *Arnoldi's method*, $\boldsymbol{G}_\lambda[t] = diag(\boldsymbol{\phi}[t]^{(1)}, ..., \boldsymbol{\phi}[t]^{(\lambda)})$ and ψ is a constant.

2.2.3 Sparsity obtained by Krylov subspace projection technique

In this subsection, we try to see how sparsity is obtained as a result of Krylov subspace projection technique. We consider solving the linear *Wiener-Hopf equation*, $\boldsymbol{R}\boldsymbol{w} = \boldsymbol{p}$ by using the *Generalized Conjugate Residual (GCR)* method which is a Krylov subspace technique. Since the autocorrelation matrix, $\boldsymbol{R}$ is symmetrical, the subspace spanned by the Krylov vectors equals to the subspace spanned by the residuals in the algorithm as shown next in 2.18.

$$span\{\boldsymbol{p}, \boldsymbol{Rp}, ..., \boldsymbol{R}^{m-1}\boldsymbol{p}\} = span\{\boldsymbol{r}_0, \boldsymbol{r}_1, ..., \boldsymbol{r}_{m-1}\} \tag{2.18}$$

where m is the filter length, i.e. length of the filter coefficient vector $\boldsymbol{w}$. Residuals have the following properties:

$$\begin{aligned} \boldsymbol{r}_0 &= \boldsymbol{p} && (2.19) \\ \boldsymbol{r}_k &= \boldsymbol{r}_0 - \sum_{i=1}^{k-1} \alpha_i \boldsymbol{R}\boldsymbol{r}_i && (2.20) \end{aligned}$$

since at each iteration system is projected to the next Krylov vector. So, residual decreases gradually. Then,

$$\boldsymbol{w}_k = \boldsymbol{\xi}_k(\boldsymbol{R})\boldsymbol{p} \tag{2.21}$$

where $\boldsymbol{\xi}_k(\boldsymbol{R})$ is the kth order polynomial which minimizes $\|\boldsymbol{r}_{k+1}\|^2$.

$$\begin{aligned} \boldsymbol{r}_{k+1} &= \boldsymbol{p} - \boldsymbol{R}\boldsymbol{w}_k && (2.22) \\ &= \boldsymbol{p} - \boldsymbol{R}\boldsymbol{\xi}_k(\boldsymbol{R})\boldsymbol{p} && (2.23) \\ &= (\boldsymbol{I} - \boldsymbol{R}\boldsymbol{\xi}_k(\boldsymbol{R}))\boldsymbol{p} && (2.24) \\ &= \gamma_{k+1}(\boldsymbol{R})\boldsymbol{p} && (2.25) \end{aligned}$$

Here, $\boldsymbol{\gamma}_{k+1}(\boldsymbol{R})$ is the k+1th order polynomial which minimizes $\|\boldsymbol{r}_{k+1}\|^2$ subject to $\boldsymbol{\gamma}_{k+1}(0) = 1$.

$$\begin{aligned} \|\boldsymbol{r}_{k+1}\| &\leq \|\gamma_{k+1}(\boldsymbol{R})\|\|\boldsymbol{p}\| && (2.26) \\ \frac{\|\boldsymbol{r}_{k+1}\|}{\|\boldsymbol{p}\|} &\leq \|\gamma_{k+1}(\boldsymbol{R})\| && (2.27) \end{aligned}$$

Noting that $\boldsymbol{p} = \boldsymbol{r}_0$ we get an upper bound for the percentage of the residual that is left after kth iteration in 2.27. So, we continue investigating this upper bound polynomial. If

$$\boldsymbol{R} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1},$$

then

$$
\begin{aligned}
\|\gamma_{k+1}(\boldsymbol{R})\| &= \|\boldsymbol{V}\gamma_{k+1}(\boldsymbol{\Lambda})\boldsymbol{V}^{-1}\| && (2.28)\\
&\leq \underbrace{\|\boldsymbol{V}\|\|\boldsymbol{V}^{-1}\|}_{cond(\boldsymbol{V})}\|\gamma_{k+1}(\boldsymbol{\Lambda})\| && (2.29)\\
&\leq cond(\boldsymbol{V})\left\|\begin{bmatrix}\gamma_{k+1}(\lambda_1) & & & \\ & \cdot & & \\ & & \cdot & \\ & & & \gamma_{k+1}(\lambda_m)\end{bmatrix}\right\|. && (2.30)
\end{aligned}
$$

Here, $\boldsymbol{V}$ is the eigenvector matrix and $\boldsymbol{\Lambda}$ is the eigenvalue matrix of $\boldsymbol{R}$. 2.28 follows from the *spectral mapping theorem* which states that

$$
spectrum(f(\boldsymbol{R})) = f(spectrum(\boldsymbol{R})) \qquad (2.31)
$$

because

$$
\begin{aligned}
\boldsymbol{RR} &= \boldsymbol{V\Lambda V}^{-1}\boldsymbol{V\Lambda V}^{-1} = \boldsymbol{V\Lambda}^2\boldsymbol{V}^{-1}\\
\boldsymbol{R}^n &= \boldsymbol{V\Lambda}^n\boldsymbol{V}^{-1}\\
f(\boldsymbol{R}) &= a_0\boldsymbol{VV}^{-1} + a_1\boldsymbol{V\Lambda V}^{-1} + ... + a_n\boldsymbol{V\Lambda}^n\boldsymbol{V}^{-1}\\
&= \boldsymbol{V}\left(a_0\boldsymbol{I} + a_1\boldsymbol{\Lambda} + ... + a_n\boldsymbol{\Lambda}^n\right)\boldsymbol{V}^{-1}\\
&= \boldsymbol{V}f(\boldsymbol{\Lambda})\boldsymbol{V}^{-1}. && (2.32)
\end{aligned}
$$

$cond(\boldsymbol{V})$ is the conditional number of $\boldsymbol{V}$. In our case because $\boldsymbol{R}$ is symmetrical, $\boldsymbol{V}$ has orthonormal columns ($cond(\boldsymbol{V}) = 1$) and $\boldsymbol{R}$ has real eigenvalues. Actually $\lambda_i \geq 0$ since $\boldsymbol{R}$ is positive semi-definite.

Now, we know that the upper bound for the residual after kth iteration is

$$
\begin{aligned}
\|\gamma_k(\boldsymbol{R})\| &\leq cond(\boldsymbol{V})\max_i|\gamma_k(\lambda_i)|\\
&\leq \max_i|\gamma_k(\lambda_i)| && (2.33)
\end{aligned}
$$

So, we try to minimize the maximum response of the $\boldsymbol{\gamma}$ function to the eigenvalues of $\boldsymbol{R}$. $\boldsymbol{\gamma}_k$ is any kth order polynomial such that $\boldsymbol{\gamma}_k(0) = 1$. So, our task turns into fitting a polynomial $\boldsymbol{\gamma}_k$ at each iteration (k:iteration number) to the eigenvalues $\lambda_i,\ i = 1, ..., m$ of the autocorrelation matrix $\boldsymbol{R}$ that are on the positive real axis. Fig. 2.2 illustrates this polynomial fitting task.

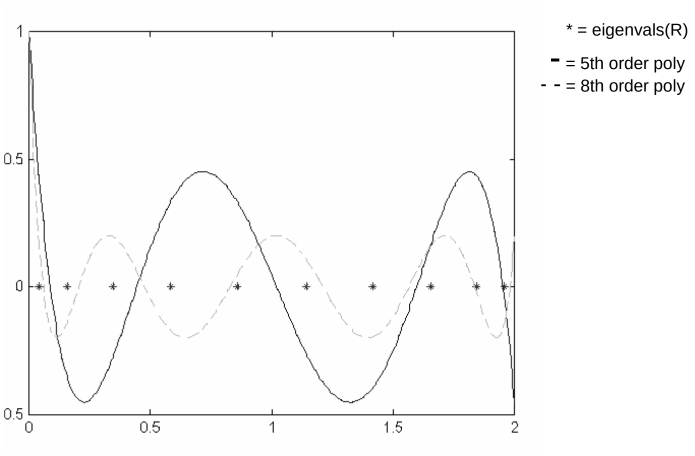


Figure 2.2: Polynomial fitting of GCR. GCR fits a higher order polynomial to the eigenvalues of $\boldsymbol{R}$ in order to minimize the maximum response to the eigenvalues.

It is obviously easy to fit a low order polynomial to clustered eigenvalues. In case that $cond(\boldsymbol{R}) = \frac{\lambda_{max}}{\lambda_{min}} \approx 1$, eigenvalues are very close to each other on the real positive axis which means again that a low order polynomial can easily fit them. Therefore, for either cases in a few iterations a good approximation to $\boldsymbol{w}$ can be obtained since residual will be small with a small upper bound as a result of good polynomial fit. If $\boldsymbol{R}$ has clustered eigenvalues or has condition number close to 1, then the representation in the new Krylov coordinates $\acute{w} = \boldsymbol{Q}^T\boldsymbol{w}$ of the system impulse response $\boldsymbol{w}$ is sparse because only first few columns of $\boldsymbol{Q}$ are enough to have a good approximation, i.e. in a few iteration of Krylov subspace projection residual becomes sufficiently small and a good approximation is obtained.

In case that $\boldsymbol{R}$ has neither clustered eigenvalues nor condition number close to 1, one can use a preconditioner to whiten the input regressor $\boldsymbol{x}[t]$. Preconditioning helps $\boldsymbol{R}$ to have clustered eigenvalues. Some methods for preconditioning are provided next. For more preconditioning techniques, see [22] and [23].

If $\boldsymbol{R}$ is a diagonal matrix,

$$\boldsymbol{R} = \begin{bmatrix} 1 & & & & & \\ & 2 & & & & \\ & & \cdot & & & \\ & & & \cdot & & \\ & & & & \cdot & \\ & & & & & N \end{bmatrix}$$

then $\boldsymbol{R}$ has N distinct eigenvalues and we need a preconditioner for $\boldsymbol{R}$ to have clustered eigenvalues. Using a preconditioner,

$$\boldsymbol{P} = \begin{bmatrix} 1 & & & & & \\ & \frac{1}{2} & & & & \\ & & \cdot & & & \\ & & & \cdot & & \\ & & & & \cdot & \\ & & & & & \frac{1}{N} \end{bmatrix} \tag{2.34}$$

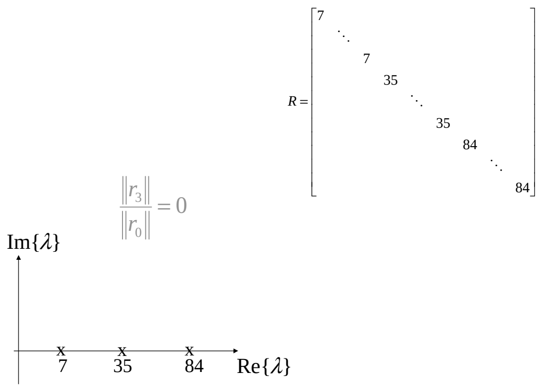
we have the equation $\boldsymbol{PRw} = \boldsymbol{Pp}$ where $\boldsymbol{PR} = \boldsymbol{I}$ has all eigenvalues equal to 1. For $\boldsymbol{R}$ being a diagonally dominant matrix, we can implement a similar procedure for preconditioning. Let $\boldsymbol{R} = \boldsymbol{D} + \boldsymbol{R}_{nd}$ where $\boldsymbol{D}$ is the diagonal part and $\boldsymbol{R}_{nd}$ is the nondiagonal part. $\boldsymbol{R}_{nd}$ has very small entries compared to diagonal entries. Applying $\boldsymbol{D}^{-1}$ as the preconditioner we get

$$\left(\boldsymbol{D}^{-1}\boldsymbol{R}\right)\boldsymbol{w} = \left(\boldsymbol{I} + \boldsymbol{D}^{-1}\boldsymbol{R}_{nd}\right)\boldsymbol{w} = \boldsymbol{D}^{-1}\boldsymbol{p}$$

Using the inverse of the diagonal part as the preconditioner usually improves the convergence and the inverse of a diagonal is cheap to compute. Diagonal and diagonally dominant cases are demonstrated in Figures 2.3 and 2.4.

One should choose a preconditioner $\boldsymbol{P}$ such that

- $\tilde{\boldsymbol{R}} \approx \boldsymbol{R}$
- $\tilde{\boldsymbol{R}}$ is easy to invert or factor
- $\boldsymbol{P} = \tilde{\boldsymbol{R}}^{-1}$



$R = \mathrm{diag}(7, \ldots, 7, 35, \ldots, 35, 84, \ldots, 84)$
$\frac{\|r_3\|}{\|r_0\|} = 0$
Im{λ}
Re{λ}
7
35
84

Figure 2.3: $\boldsymbol{R}$ is diagonal and has 3 distinct eigenvalues. At most 3 iterations a polynomial is fitted perfectly to the eigenvalues and residual becomes zero.

Let $\boldsymbol{R} \approx \boldsymbol{LU}$. Then, by using this approximate *LU factorization* as preconditioner we get,

$$\left((\boldsymbol{LU})^{-1}\boldsymbol{R}\right)\boldsymbol{w} = (\boldsymbol{LU})^{-1}\boldsymbol{p} \tag{2.35}$$

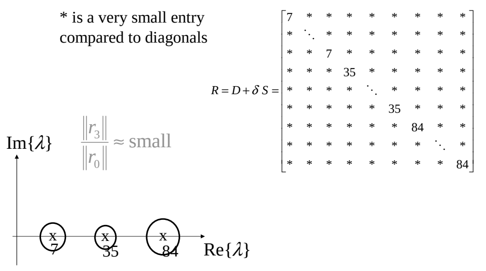
If we apply a *right preconditioner*, we should find first $\bar{\boldsymbol{w}}$, then $\boldsymbol{w}$ in $\boldsymbol{RP}\bar{\boldsymbol{w}} = \boldsymbol{p} \Rightarrow \boldsymbol{w} = \boldsymbol{P}\bar{\boldsymbol{w}}$

2.3 Steady-State MSE Analysis

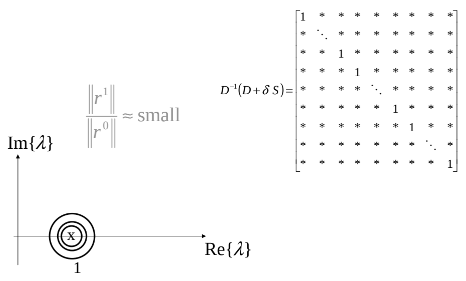
This section proceeds with the same order as in [24]. First, an energy conservation relation is derived. Then, a variance relation is developed from that energy conservation relation since we are interested in evaluating the steady-state variance of the estimation error. Afterwards, excess MSE expressions at steady- state are found for KPNLMS and KPNLMF. Full matrix representations of $\boldsymbol{\Omega}$, $\boldsymbol{Q}$ and $\boldsymbol{G}$ are used for clarity since MSE performances of the algorithms are not affected by the simplification made to attain linear computational complexity.

2.3.1 Energy Conservation Relation

$$\tilde{\boldsymbol{w}}[t] = \tilde{\boldsymbol{w}}[t-1] - \mu\boldsymbol{\Omega}[t]\boldsymbol{x}[t]g\left[e[t]\right] \tag{2.36}$$



(a)



(b)

Figure 2.4: (a)Eigenvalues of $\boldsymbol{R}$ are clustered around 3 points and in 3 iterations a small residual is achieved without preconditioning. (b)By using the inverse of the diagonal part as the preconditioner, we can achieve a small residual in only one iteration.

where $\tilde{\boldsymbol{w}}[t] = \boldsymbol{w}_0 - \boldsymbol{w}[t]$ is the weight-error vector, $0 < \mu < 2$, $\boldsymbol{\Omega}[t] = \boldsymbol{Q}\boldsymbol{G}[t]\boldsymbol{Q}^T$ and

$$\begin{aligned} g\left[e[t]\right] &= \frac{e[t]}{\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]} + \epsilon} \\ &= \frac{e_a[t] + v[t]}{\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]} + \epsilon} \end{aligned} \tag{2.37}$$

for KPNLMS

$$\begin{aligned} g\left[e[t]\right] &= \frac{2e[t]^3}{\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]} + \epsilon} \\ &= \frac{2(e_a[t] + v[t])^3}{\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]} + \epsilon} \end{aligned} \tag{2.38}$$

for KPNLMF Here, $e[t] = e_a[t] + v[t]$ is the total error composed of the estimation error $e_a[t]$ and the i.i.d. noise $v[t]$ with zero mean and variance σ_v^2. If we multiply (2.36) by $\boldsymbol{x}[t]^T$ from left, we get

$$e_p[t] = e_a[t] - \mu\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]} g\left[e[t]\right] \tag{2.39}$$

Here, $e_p[t] = \boldsymbol{x}[t]^T\tilde{\boldsymbol{w}}[t]$ and $e_a[t] = \boldsymbol{x}[t]^T\tilde{\boldsymbol{w}}[t-1]$ are the aposteriori and apriori estimation errors respectively. Then, from (2.39)

$$g\left[e[t]\right] = \frac{e_a[t] - e_p[t]}{\mu\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}} \tag{2.40}$$

Substitute (2.40) in (2.36).

$$\begin{aligned}
\tilde{\boldsymbol{w}}[t] &= \tilde{\boldsymbol{w}}[t-1] - \mu\boldsymbol{\Omega}[t]\boldsymbol{x}[t]\frac{e_a[t]-e_p[t]}{\mu\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}} \\
\tilde{\boldsymbol{w}}[t] + \mu\frac{\boldsymbol{\Omega}[t]\boldsymbol{x}[t]}{\mu\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}}e_a[t] &= \tilde{\boldsymbol{w}}[t-1] + \mu\frac{\boldsymbol{\Omega}[t]\boldsymbol{x}[t]}{\mu\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}}e_p[t] \\
\tilde{\boldsymbol{w}}[t] + \frac{\boldsymbol{\Omega}[t]\boldsymbol{x}[t]}{\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}}e_a[t] &= \tilde{\boldsymbol{w}}[t-1] + \frac{\boldsymbol{\Omega}[t]\boldsymbol{x}[t]}{\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}}e_p[t] \qquad (2.41)
\end{aligned}$$

Next, we equate the squared weighted norms of both sides of (2.41) using $\boldsymbol{\Omega}[t]^{-1}$ as weighting matrix.

$$\begin{aligned}
&[\tilde{\boldsymbol{w}}[t] + \bar{\mu}[t]\boldsymbol{\Omega}[t]\boldsymbol{x}[t]e_a[t]]^T\,\boldsymbol{\Omega}[t]^{-1}\,[\tilde{\boldsymbol{w}}[t] + \bar{\mu}[t]\boldsymbol{\Omega}[t]\boldsymbol{x}[t]e_a[t]] = \qquad (2.42) \\
&[\tilde{\boldsymbol{w}}[t-1] + \bar{\mu}[t]\boldsymbol{\Omega}[t]\boldsymbol{x}[t]e_p[t]]^T\boldsymbol{\Omega}[t]^{-1}[\tilde{\boldsymbol{w}}[t-1] + \bar{\mu}[t]\boldsymbol{\Omega}[t]\boldsymbol{x}[t]e_p[t]]
\end{aligned}$$

Here, $\bar{\mu}$ is defined as,

$$\bar{\mu}[t] \triangleq \begin{cases} \frac{1}{\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}} & \text{if } \boldsymbol{x}[t] \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

Noting that $\boldsymbol{\Omega}^T = \boldsymbol{\Omega} = \boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T$ since $\boldsymbol{G}$ is a diagonal matrix, (2.42) becomes,

$$\begin{aligned}
\|\tilde{\boldsymbol{w}}[t]\|^2_{\boldsymbol{\Omega}[t]^{-1}} + \bar{\mu}[t]\tilde{\boldsymbol{w}}[t]^T\boldsymbol{\Omega}[t]^{-1}\boldsymbol{\Omega}[t]\boldsymbol{x}[t]e_a[t] &+ \qquad (2.43) \\
\bar{\mu}[t]\boldsymbol{x}[t]^T\boldsymbol{\Omega}[t]\boldsymbol{\Omega}[t]^{-1}e_a[t]\tilde{\boldsymbol{w}}[t] + \bar{\mu}[t]^2\boldsymbol{x}[t]^T\boldsymbol{\Omega}[t]e_a[t]\boldsymbol{\Omega}[t]^{-1}\boldsymbol{\Omega}[t]\boldsymbol{x}[t]e_a[t] &= \\
\|\tilde{\boldsymbol{w}}[t-1]\|^2_{\boldsymbol{\Omega}[t]^{-1}} + \bar{\mu}[t]\tilde{\boldsymbol{w}}[t-1]^T\boldsymbol{\Omega}[t]^{-1}\boldsymbol{\Omega}[t]\boldsymbol{x}[t]e_p[t] &+ \\
\bar{\mu}[t]\boldsymbol{x}[t]^T\boldsymbol{\Omega}[t]\boldsymbol{\Omega}[t]^{-1}e_p[t]\tilde{\boldsymbol{w}}[t-1] + \bar{\mu}[t]^2\boldsymbol{x}[t]^T\boldsymbol{\Omega}[t]e_p[t]\boldsymbol{\Omega}[t]^{-1}\boldsymbol{\Omega}[t]\boldsymbol{x}[t]e_p[t] &
\end{aligned}$$

Substituting $e_p[t] = \boldsymbol{x}[t]^T\tilde{\boldsymbol{w}}[t]$ and $e_a[t] = \boldsymbol{x}[t]^T\tilde{\boldsymbol{w}}[t-1]$ we get,

$$\begin{aligned}
\|\tilde{\boldsymbol{w}}[t]\|^2_{\boldsymbol{\Omega}[t]^{-1}} + \bar{\mu}[t]\langle\tilde{\boldsymbol{w}}[t], \tilde{\boldsymbol{w}}[t-1]\rangle_{\boldsymbol{R}_{\boldsymbol{x}[t]}} + \bar{\mu}[t]e_a[t]e_p[t] + \bar{\mu}[t]^2\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}|e_a[t]|^2 &= \\
\|\tilde{\boldsymbol{w}}[t-1]\|^2_{\boldsymbol{\Omega}[t]^{-1}} + \bar{\mu}[t]\langle\tilde{\boldsymbol{w}}[t], \tilde{\boldsymbol{w}}[t-1]\rangle_{\boldsymbol{R}_{\boldsymbol{x}[t]}} + \bar{\mu}[t]e_a[t]e_p[t] + \bar{\mu}[t]^2\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}|e_p[t]|^2 & \qquad (2.44)
\end{aligned}$$

Here, $\langle\cdot,\cdot\rangle$ is the inner product operator, $\boldsymbol{R}_{\boldsymbol{x}[t]}$ is the covariance matrix of the input regressor which is used as weighting matrix for the inner product. Common terms on both side are canceled and since

$$\bar{\mu}[t]^2\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]} = \frac{\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}}{\|\boldsymbol{x}[t]\|^4_{\boldsymbol{\Omega}[t]}}$$

equation (2.44) reduces to

$$\|\tilde{\boldsymbol{w}}[t]\|^2_{\boldsymbol{\Omega}[t]^{-1}} + \bar{\mu}[t]|e_a[t]|^2 = \|\tilde{\boldsymbol{w}}[t-1]\|^2_{\boldsymbol{\Omega}[t]^{-1}} + \bar{\mu}[t]|e_p[t]|^2. \tag{2.45}$$

Equation (2.45) states the **Energy Conservation Relation**.

2.3.2 Variance Relation

Steady-state filter operation, [24]:

$$\begin{aligned}
E\left[\tilde{\boldsymbol{w}}[t]\right] &= E\left[\tilde{\boldsymbol{w}}[t-1]\right] = s \qquad \text{as t} \to \infty \qquad (\text{usually } s = 0) && (2.46)\\
E\left[\tilde{\boldsymbol{w}}[t]\tilde{\boldsymbol{w}}[t]^T\right] &= E\left[\tilde{\boldsymbol{w}}[t-1]\tilde{\boldsymbol{w}}[t-1]^T\right] = \boldsymbol{R}_{\tilde{\boldsymbol{w}}[t]} \qquad \text{as t} \to \infty && (2.47)\\
& \quad (\boldsymbol{R}_{\tilde{\boldsymbol{w}}[t]}\text{:covariance matrix of the weight error vector})\\
E\left[\|\tilde{\boldsymbol{w}}[t]\|^2\right] &= E\left[\|\tilde{\boldsymbol{w}}[t-1]\|^2\right] = Tr(\boldsymbol{R}_{\tilde{\boldsymbol{w}}[t]}) \qquad \text{as t} \to \infty && (2.48)
\end{aligned}$$

In our case,

$$\begin{aligned}
E\left[\|\tilde{\boldsymbol{w}}[t]\|^2_{\boldsymbol{\Omega}[t]^{-1}}\right] &= E\left[\tilde{\boldsymbol{w}}[t]^T\boldsymbol{\Omega}[t]^{-1}\tilde{\boldsymbol{w}}[t]\right] && (2.49)\\
E\left[\|\tilde{\boldsymbol{w}}[t-1]\|^2_{\boldsymbol{\Omega}[t]^{-1}}\right] &= E\left[\tilde{\boldsymbol{w}}[t-1]^T\boldsymbol{\Omega}[t]^{-1}\tilde{\boldsymbol{w}}[t-1]\right]. && (2.50)
\end{aligned}$$

If we define $\hat{\boldsymbol{w}}[t] \overset{\triangle}{=} \boldsymbol{Q}^T\tilde{\boldsymbol{w}}[t]$, then (2.49) and (2.50) become

$$\begin{aligned}
E\left[\|\tilde{\boldsymbol{w}}[t]\|^2_{\boldsymbol{\Omega}[t]^{-1}}\right] &= E\left[\underbrace{\tilde{\boldsymbol{w}}[t]^T\boldsymbol{Q}}_{\hat{\boldsymbol{w}}[t]^T}\boldsymbol{G}[t]^{-1}\underbrace{\boldsymbol{Q}^T\tilde{\boldsymbol{w}}[t]}_{\hat{\boldsymbol{w}}[t]}\right]\\
&= E\left[\hat{\boldsymbol{w}}[t]^T\boldsymbol{G}[t]^{-1}\hat{\boldsymbol{w}}[t]\right]\\
&= Tr(\boldsymbol{R}_{\hat{\boldsymbol{w}}[t]}\boldsymbol{G}[t]^{-1}) && (2.51)\\
E\left[\|\tilde{\boldsymbol{w}}[t-1]\|^2_{\boldsymbol{\Omega}[t]^{-1}}\right] &= E\left[\hat{\boldsymbol{w}}[t-1]^T\boldsymbol{G}[t]^{-1}\hat{\boldsymbol{w}}[t-1]\right]\\
&= Tr(\boldsymbol{R}_{\hat{\boldsymbol{w}}[t-1]}\boldsymbol{G}[t]^{-1}). && (2.52)
\end{aligned}$$

We know that $\boldsymbol{G}[t]$ is a diagonal matrix, so $\boldsymbol{G}[t]^{-1}$ is also a diagonal matrix. Next, we show that $\boldsymbol{R}_{\hat{\boldsymbol{w}}[t]} = \boldsymbol{R}_{\hat{\boldsymbol{w}}[t-1]}$.

$$\begin{aligned}
\boldsymbol{R}_{\hat{\boldsymbol{w}}[t]} &= E\left[\hat{\boldsymbol{w}}[t]\hat{\boldsymbol{w}}[t]^T\right]\\
&= \boldsymbol{Q}^T E\left[\tilde{\boldsymbol{w}}[t]\tilde{\boldsymbol{w}}[t]^T\right]\boldsymbol{Q}\\
&= \boldsymbol{Q}^T\boldsymbol{R}_{\tilde{\boldsymbol{w}}[t]}\boldsymbol{Q} && (2.53)\\
\boldsymbol{R}_{\hat{\boldsymbol{w}}[t-1]} &= \boldsymbol{Q}^T E\left[\tilde{\boldsymbol{w}}[t-1]\tilde{\boldsymbol{w}}[t-1]^T\right]\boldsymbol{Q}\\
&= \boldsymbol{Q}^T\boldsymbol{R}_{\tilde{\boldsymbol{w}}[t]}\boldsymbol{Q} && (2.54)
\end{aligned}$$

(2.53) and (2.54) follow from the fact that $E\left[\tilde{\boldsymbol{w}}[t]\tilde{\boldsymbol{w}}[t]^T\right] = E\left[\tilde{\boldsymbol{w}}[t-1]\tilde{\boldsymbol{w}}[t-1]^T\right] = \boldsymbol{R}_{\tilde{\boldsymbol{w}}[t]}$ in (2.47). In (2.53) and (2.54), it is shown that $\boldsymbol{R}_{\hat{\boldsymbol{w}}[t]} = \boldsymbol{R}_{\hat{\boldsymbol{w}}[t-1]}$. So, from (2.51) and (2.52) we can conclude that

$$E\left[\|\tilde{\boldsymbol{w}}[t]\|^2_{\boldsymbol{\Omega}[t]^{-1}}\right] = E\left[\|\tilde{\boldsymbol{w}}[t-1]\|^2_{\boldsymbol{\Omega}[t]^{-1}}\right] = Tr(\boldsymbol{R}_{\hat{\boldsymbol{w}}[t]}\boldsymbol{G}[t]^{-1}) \tag{2.55}$$

Energy conservation relation:

Returning to energy conservation relation we remember from (2.45) that

$$\|\tilde{\boldsymbol{w}}[t]\|^2_{\boldsymbol{\Omega}[t]^{-1}} + \bar{\mu}[t]|e_a[t]|^2 = \|\tilde{\boldsymbol{w}}[t-1]\|^2_{\boldsymbol{\Omega}[t]^{-1}} + \bar{\mu}[t]|e_p[t]|^2$$

and from (2.55) we know that $E\left[\|\tilde{\boldsymbol{w}}[t]\|^2_{\boldsymbol{\Omega}[t]^{-1}}\right] = E\left[\|\tilde{\boldsymbol{w}}[t-1]\|^2_{\boldsymbol{\Omega}[t]^{-1}}\right]$. Therefore, taking expectation of both sides our energy conservation relation turns into

$$E\left[\bar{\mu}[t]|e_a[t]|^2\right] = E\left[\bar{\mu}[t]|e_p[t]|^2\right] \tag{2.56}$$

Using the fact that $e_p[t] = e_a[t] - \mu\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}g[e[t]]$ in (2.39) we get

$$E\left[\bar{\mu}[t]|e_a[t]|^2\right] = E\left[\bar{\mu}[t]|e_a[t] - \mu\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}g[e[t]]|^2\right] \tag{2.57}$$

If we expand the inside of the expectation on the right hand side of (2.57), we get

$$\bar{\mu}[t]|e_a[t]|^2 + \mu^2\bar{\mu}[t]\|\boldsymbol{x}[t]\|^4_{\boldsymbol{\Omega}[t]}|g|^2 - \mu\bar{\mu}[t]\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}e_a[t]g^* - \mu\bar{\mu}[t]\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}e_a[t]^*g. \tag{2.58}$$

The argument of $g[e[t]]$ is not written while developing equations for clarity. We know that $\bar{\mu}[t]\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]} = 1$ for all $\boldsymbol{x}[t]$ except the trivial case $\boldsymbol{x}[t] = 0$ for which the product is zero. So,

$$E\left[\bar{\mu}[t]\|\boldsymbol{x}[t]\|^4_{\boldsymbol{\Omega}[t]}|g|^2\right] = E\left[\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}|g|^2\right] \tag{2.59}$$

$$E\left[\bar{\mu}[t]\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}e_a[t]g\right] = E\left[e_a[t]g\right] \tag{2.60}$$

Next, we substitute (2.59) and (2.60) in (2.58) and continue with (2.57).

$$E\left[\bar{\mu}[t]|e_a[t]|^2\right] = E\left[\bar{\mu}[t]|e_a[t]|^2\right] + \mu^2 E\left[\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}|g|^2\right] - \mu E\left[e_a[t]g^*\right] - \mu E\left[e_a[t]^*g\right]$$

$E\left[\bar{\mu}[t]|e_a[t]|^2\right]$ terms on each side cancel each other and we get,

$$\begin{aligned} \mu E\left[\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}|g|^2\right] &= E\left[e_a[t]g^* + e_a[t]^*g\right] \quad \text{as t} \to \infty \\ \mu E\left[\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}|g|^2\right] &= 2Re\{E\left[e_a[t]^*g\right]\} \quad \text{as t} \to \infty \qquad (2.61) \\ \mu E\left[\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}g^2\right] &= 2E\left[e_a[t]g\right] \quad \text{as t} \to \infty \qquad (2.62) \end{aligned}$$

(2.61) and (2.62) represent the **Variance Relation** for complex-valued and real-valued data respectively.

2.3.3 MSE of KPNLMS

For KPNLMS,

$$g[e[t]] = \frac{e_a[t] + v[t]}{\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]} + \epsilon} \tag{2.63}$$

Define a new variable $\hat{\mu}[t] \overset{\triangle}{=} \frac{1}{\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}+\epsilon}$ and substitute(2.63) in the variance relation (2.61).

$$\mu E\left[\hat{\mu}[t]^2\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}|e_a[t]+v[t]|^2\right] = 2Re\{E\left[\hat{\mu}[t]e_a[t]^*(e_a[t]+v[t])\right]\} \tag{2.64}$$

Separation Principle, [24]:
At steady-state, $\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}$ is independent of $e_a[t]$.

Lemma, [24]:
$v[t]$ is independent of $e_a[t]$, $\boldsymbol{x}[t]$ and $\boldsymbol{w}[j]$, $\tilde{\boldsymbol{w}}[j]$ where $j < t \quad (e.g., j = t-1)$.

So, expectations including cross terms of $e_a[t]$ and $v[t]$ are separated and since $v[t]$ is zero mean they are canceled.

$$\mu E\left[\hat{\mu}[t]^2\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}|e_a[t]|^2\right] + \mu\sigma_v^2 E\left[\hat{\mu}[t]^2\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}\right] = 2E\left[\hat{\mu}[t]|e_a[t]|^2\right] \tag{2.65}$$

where $\sigma_v^2 = E\left[|v[t]|^2\right] = J_{min}$. $\hat{\mu}$ is a function of $\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}$ and if we apply separation principle, (2.65) reduces to

$$\mu E\left[\hat{\mu}[t]^2\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}\right]\ E\left[|e_a[t]|^2\right] + \mu\sigma_v^2 E\left[\hat{\mu}[t]^2\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}\right] = 2E\left[\hat{\mu}[t]\right]\ E\left[|e_a[t]|^2\right] \tag{2.66}$$

Define new variables,

$$\alpha_{\boldsymbol{x}[t]} \overset{\triangle}{=} E\left[\hat{\mu}[t]^2\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}\right], \qquad \eta_{\boldsymbol{x}[t]} \overset{\triangle}{=} E\left[\hat{\mu}[t]\right]$$

Then, (2.66) becomes

$$\begin{aligned}(2\eta_{\boldsymbol{x}[t]} - \mu\alpha_{\boldsymbol{x}[t]})E\left[|e_a[t]|^2\right] &= \mu\sigma_v^2\alpha_{\boldsymbol{x}[t]} \qquad \text{t} \to \infty \\ lim_{t\to\infty}E\left[|e_a[t]|^2\right] &= \xi^{KPNLMS} = \frac{\mu\sigma_v^2\alpha_{\boldsymbol{x}[t]}}{2\eta_{\boldsymbol{x}[t]} - \mu\alpha_{\boldsymbol{x}[t]}}\end{aligned} \tag{2.67}$$

(2.67) is the **Excess MSE of KPNLMS**. We can simplify (2.67) by assuming that ϵ is sufficiently small, $\epsilon \approx 0$, which is usually the case. Using this assumption in two different places we can approximate the excess MSE of KPNLMS in two different ways.

First approximation:

Effect of ϵ can be ignored in the definitions of $\alpha_{\boldsymbol{x}[t]}$ and $\eta_{\boldsymbol{x}[t]}$ $\quad(\hat{\mu}[t] \approx \frac{1}{\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}})$.

$$\begin{aligned}
\alpha_{\boldsymbol{x}[t]} &= E\left[\hat{\mu}[t]^2\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}\right], \qquad \eta_{\boldsymbol{x}[t]} = E\left[\hat{\mu}[t]\right] \\
&= E\left[\frac{1}{\|\boldsymbol{x}[t]\|^4_{\boldsymbol{\Omega}[t]}}\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}\right], \qquad \eta_{\boldsymbol{x}[t]} = E\left[\frac{1}{\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}}\right] \\
\alpha_{\boldsymbol{x}[t]} &= \eta_{\boldsymbol{x}[t]} = E\left[\frac{1}{\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}}\right] \qquad (2.68)
\end{aligned}$$

Accordingly (2.67) reduces to,

$$\begin{aligned}
\xi^{KPNLMS} &= \frac{\mu\sigma_v^2\alpha_{\boldsymbol{x}[t]}}{(2-\mu)\alpha_{\boldsymbol{x}[t]}} \\
\xi^{KPNLMS} &= \frac{\mu\sigma_v^2}{2-\mu} \qquad \textit{(when } \epsilon \textit{ is small)} \qquad (2.69)
\end{aligned}$$

(2.69) is the first approximation to the excess MSE of KPNLMS.

Second approximation:

Use $\epsilon \approx 0$ in (2.65) before applying the *separation principle* $\quad(\hat{\mu}[t] \approx \frac{1}{\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}})$.

$$\begin{aligned}
\mu E\left[\frac{1}{\|\boldsymbol{x}[t]\|^4_{\boldsymbol{\Omega}[t]}}\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}|e_a[t]|^2\right] + \mu\sigma_v^2 E\left[\frac{1}{\|\boldsymbol{x}[t]\|^4_{\boldsymbol{\Omega}[t]}}\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}\right] &= 2E\left[\frac{1}{\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}}|e_a[t]|^2\right] \\
\mu E\left[\frac{|e_a[t]|^2}{\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}}\right] + \mu\sigma_v^2 E\left[\frac{1}{\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}}\right] &= 2E\left[\frac{|e_a[t]|^2}{\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}}\right] \quad (2.70)
\end{aligned}$$

Then, implement the following steady-state approximation instead of separation principle in (2.70).

$$E\left[\frac{|e_a[t]|^2}{\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}}\right] \approx \frac{E\left[|e_a[t]|^2\right]}{E\left[\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}\right]} = \frac{E\left[|e_a[t]|^2\right]}{Tr(\boldsymbol{R}_{\hat{\boldsymbol{x}}[t]}\boldsymbol{G}[t])} \quad \text{as } t \to \infty \qquad (2.71)$$

$$\hat{\boldsymbol{x}}[t] = \boldsymbol{Q}^T\boldsymbol{x}[t]$$

$$\begin{aligned}
\frac{2-\mu}{Tr(\boldsymbol{R}_{\hat{\boldsymbol{x}}[t]}\boldsymbol{G}[t])}E\left[|e_a[t]|^2\right] &= \mu\sigma_v^2 E\left[\frac{1}{\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}}\right] \\
\xi^{KPNLMS} &= \frac{\mu\sigma_v^2}{2-\mu}Tr(\boldsymbol{R}_{\hat{\boldsymbol{x}}[t]}\boldsymbol{G}[t])E\left[\frac{1}{\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}}\right] \qquad (2.72) \\
&\quad \textit{(when } \epsilon \textit{ is small)}
\end{aligned}$$

(2.72) is the second approximation to the excess MSE of KPNLMS. In (2.71), $E\left[\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]}\right] = E\left[\boldsymbol{x}[t]^T\boldsymbol{Q}\boldsymbol{G}[t]\boldsymbol{Q}^T\boldsymbol{x}[t]\right] = E\left[\hat{\boldsymbol{x}}[t]^T\boldsymbol{G}[t]\hat{\boldsymbol{x}}[t]\right] = Tr(\boldsymbol{R}_{\hat{\boldsymbol{x}}[t]}\boldsymbol{G}[t])$.

Both approximations resulted small numbers for the excess MSE of KPNLMS meaning that the KPNLMS algorithm converges to the desired filter coefficients.

2.3.4 MSE of KPNLMF/KPNLMMN

For KPNLMMN (Krylov Proportionate Normalized Least Mean Mixed Norm),

$$g[e[t]] = \frac{e[t]\left[\delta + (1-\delta)2|e[t]|^2\right]}{\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]} + \epsilon} \tag{2.73}$$

KPNLMMN is a generic algorithm in which both KPNLMS and KPNLMF are included as special cases. δ arranges the ratio of the usages of KPNLMS and KPNLMF. Larger δ is, KPNLMMN is more like KPNLMS and smaller δ is, it is more like KPNLMF. KPNLMF is the specific case of KPNLMMN where $\delta = 0$ and also $\delta = 1$ yields KPNLMS. So we will continue the derivation for the general case KPNLMMN and afterwards the result for KPNLMF will be obtained easily. We can also check the result we found for the excess MSE of KPNLMS in the previous part when we reach the result for KPNLMMN.
$g[e[t]]$ for KPNLMF is given by the equation,

$$g[e[t]] = \frac{2[e_a[t] + v[t]]^3}{\|\boldsymbol{x}[t]\|^2_{\boldsymbol{\Omega}[t]} + \epsilon} \tag{2.74}$$

where $e_a[t] + v[t] = e[t]$.

Real-valued Data

Define $\bar{\delta} \overset{\triangle}{=} 1 - \delta$. Write (2.73) again by using the new variable $\bar{\delta}$ and $\hat{\mu}[t]$. Time index, t will be ignored while developing equations for the sake of clarity.

$$g(e) = \hat{\mu}\delta(e_a + v) + \hat{\mu}2\bar{\delta}(e_a + v)(e_a^2 + v^2 + 2e_a v) \tag{2.75}$$

In order to use the variance relation stated in (2.62) first we compute $E[e_a g(e)]$. The argument of g(e) will be dropped again for the sake of clarity. From (2.75),

$$E[e_a g] = (\delta + 6\bar{\delta}\sigma_v^2)\, E\left[\hat{\mu}e_a^2\right] + 2\bar{\delta}\, E\left[\hat{\mu}e_a^4\right] \tag{2.76}$$

since e_a and v are independent. We ignore the third and higher order powers of e_a since the estimation error $e_a[t]$ becomes small in steady-state.

$$E[e_a g] \approx (\delta + 6\bar{\delta}\sigma_v^2)\, E\left[\hat{\mu} e_a^2\right] = b\, E\left[\hat{\mu} e_a^2\right] \tag{2.77}$$

where we defined a new variable

$$b \overset{\triangle}{=} (\delta + 6\bar{\delta}\sigma_v^2). \tag{2.78}$$

Next, we compute $E\left[\|\boldsymbol{x}\|^2_{\boldsymbol{\Omega}} g^2\right]$. Start by finding g^2.

$$\begin{aligned} g^2 &= \hat{\mu}^2\delta^2 e^2 + \hat{\mu}^2 4\bar{\delta}^2 e^6 + 4\hat{\mu}^2\delta\bar{\delta}e^4 && (2.79)\\ e^2 &= e_a^2 + 2e_a v + v^2 && (2.80)\\ e^4 &= e_a^4 + 6e_a^2v^2 + 4e_a^3 v + 4e_a v^3 + v^4 && (2.81)\\ e^6 &= e_a^6 + 6e_a^5 v + 6e_a v^5 + 15e_a^4v^2 + 15e_a^2v^4 + 20e_a^3v^3 + v^6 && (2.82) \end{aligned}$$

We do the following tasks: substitute (2.80),(2.81) and (2.82) in (2.79); multiply (2.79) by $\|\boldsymbol{x}\|^2_{\boldsymbol{\Omega}}$ from left; take the expectation of both sides; use the fact that v is independent of both $\boldsymbol{x}$ and e_a; ignore third and higher order terms in e_a. The result is,

$$\begin{aligned} E\left[\|\boldsymbol{x}\|^2_{\boldsymbol{\Omega}} g^2\right] &\approx a\, E\left[\|\boldsymbol{x}\|^2_{\boldsymbol{\Omega}}\hat{\mu}^2\right] + c\, E\left[\hat{\mu}^2\|\boldsymbol{x}\|^2_{\boldsymbol{\Omega}} e_a^2\right] + && (2.83)\\ & 16\delta\bar{\delta}\, E\left[\hat{\mu}^2\|\boldsymbol{x}\|^2_{\boldsymbol{\Omega}} e_a\right] E[v3] + 24\bar{\delta}^2 E\left[\|\boldsymbol{x}\|^2_{\boldsymbol{\Omega}} e_a\right] E[v^5] \end{aligned}$$

where

$$\begin{aligned} a &\overset{\triangle}{=} \delta^2\sigma_v^2 + 4\delta\bar{\delta}\tau_v^4 + 4\bar{\delta}^2\tau_v^6 && (2.84)\\ c &\overset{\triangle}{=} \delta^2 + 24\delta\bar{\delta}\sigma_v^2 + 60\bar{\delta}^2\tau_v^4 && (2.85) \end{aligned}$$

We defined $\tau_v^4 \overset{\triangle}{=} E\left[|v^4|\right]$ and $\tau_v^6 \overset{\triangle}{=} E\left[|v^6|\right]$. Remembering that $\alpha_{\boldsymbol{x}} = E\left[\|\boldsymbol{x}\|^2_{\boldsymbol{\Omega}}\hat{\mu}^2\right]$, and combining (2.77) and (2.83) we obtain the variance relation for KPNLMMN.

$$2b\, E\left[\hat{\mu} e_a^2\right] = \mu a \alpha_{\boldsymbol{x}} + \mu c\, E\left[\hat{\mu}^2\|\boldsymbol{x}\|^2_{\boldsymbol{\Omega}} e_a^2\right] + 16\mu\delta\bar{\delta}\, E\left[\hat{\mu}^2\|\boldsymbol{x}\|^2_{\boldsymbol{\Omega}} e_a\right]\tau_v^3 + 24\mu\bar{\delta}^2\, E\left[\|\boldsymbol{x}\|^2_{\boldsymbol{\Omega}} e_a\right]\tau_v^5 \tag{2.86}$$

as t $\rightarrow \infty$. We can simplify (2.86) in two ways depending on the step-size μ.

Sufficiently small μ:

Small step-size yields small $E\left[|e_a|^2\right]$ in steady-state and consequently small e_a. So, the last three terms on the right hand side of the equation (2.86) can be neglected. As a result,

$$E\left[\hat{\mu}e_a^2\right] = \frac{\mu a \alpha_{\boldsymbol{x}}}{2b} \tag{2.87}$$

If we apply separation principle,

$$\begin{aligned} \underbrace{E[\hat{\mu}]}_{\eta_{\boldsymbol{x}}}\, E\left[e_a^2\right] &= \frac{\mu a \alpha_{\boldsymbol{x}}}{2b} \\ E\left[e_a^2\right] = \xi^{KPNLMMN} &= \frac{\mu a \alpha_{\boldsymbol{x}}}{2b\eta_{\boldsymbol{x}}} \end{aligned} \tag{2.88}$$

Small ϵ provides that $\alpha_{\boldsymbol{x}} = \eta_{\boldsymbol{x}}$ and accordingly we have,

$$\xi^{KPNLMMN} = \frac{\mu a}{2b} \tag{2.89}$$

For KPNLMF, $\delta = 0$ and this leads to $a = 4\tau_v^6$ from (2.84), $b = 6\sigma_v^2$ from (2.78). So,

$$\xi^{KPNLMF} = \frac{\mu \tau_v^6}{3\sigma_v^2} \tag{2.90}$$

For sufficiently small μ, ϵ and applying separation principle we get the excess MSE formula for KPNLMF in (2.90).

For KPNLMS, $\delta = 1$ and accordingly $a = \sigma_v^2$, $b = 1$. So,

$$\xi^{KPNLMS} = \frac{\mu \sigma_v^2}{2} \tag{2.91}$$

The result found here for KPNLMS is similar to the one in (2.69). In (2.69), there is an extra $-\mu$ at the denominator. Actually, the result in (2.91) is the same result with the assumption of small μ. If $\mu \ll 2$, then it can be neglected. So, two results for the excess MSE of KPNLMS are consistent.

Large μ:

If we apply the separation principle, terms including τ_v^3 and τ_v^5 in (2.86) are zero since e_a has zero mean. Then, (2.86) turns into

$$\begin{aligned} 2b\eta_{\boldsymbol{x}}\, E\left[e_a^2\right] &= \mu a \alpha_{\boldsymbol{x}} + \mu c \alpha_{\boldsymbol{x}}\, E\left[e_a^2\right] \\ \xi^{KPNLMMN} &= \frac{\mu a \alpha_{\boldsymbol{x}}}{2b\eta_{\boldsymbol{x}} - \mu c \alpha_{\boldsymbol{x}}} \\ \xi^{KPNLMMN} &= \frac{\mu a}{2b - \mu c} \qquad (small\ \epsilon \quad (\alpha_{\boldsymbol{x}} = \eta_{\boldsymbol{x}})) \end{aligned} \tag{2.92}$$

For KPNLMF, $c = 60\tau_v^4$ from (2.85) and this results in,

$$\xi^{KPNLMF} = \frac{\mu\tau_v^6}{3\sigma_v^2 - 15\mu\tau_v^4} \tag{2.93}$$

In case of large μ, small ϵ and separation principle applied we have the excess MSE formula for KPNLMF given in (2.93). Excess MSE for KPNLMF, ξ^{KPNLMF} in both cases is a small number which depends on the step-size μ and the statistics of the noise v. Latter excess MSE (valid when step-size is large) is a little larger than the former one (valid when step-size is small) as expected.

In order to compare the result for KPNLMS ($\delta = 1$) with the previously obtained one in (2.69) we substitute $a = \sigma_v^2$, $b = 1$ and $c = 1$in (2.92). The result is

$$\xi^{KPNLMS} = \frac{\mu\sigma_v^2}{2-\mu} \tag{2.94}$$

which is same as the one in (2.69) as expected because we do not make small μ assumption here.

Complex-valued Data

$$\begin{aligned}
g &= \hat{\mu}\delta e + 2\hat{\mu}\bar{\delta}e|e|^2 && (2.95)\\
&= \hat{\mu}\delta(e_a+v) + 2\hat{\mu}\bar{\delta}(e_a+v)(e_a+v)(e_a^*+v^*) && (2.96)\\
e_a^*g &= \delta\hat{\mu}|e_a|^2 + \delta\hat{\mu}e_a^*v + 2\bar{\delta}\hat{\mu}(|e_a|^2+e_a^*v)(|e_a|^2+e_av^*+ve_a^*+|v|^2) && (2.97)\\
E[e_a^*g] &= \delta\, E\left[\hat{\mu}|e_a|^2\right] + \delta\, E[\hat{\mu}e_a^*]\underbrace{E[v]}_{0} + 2\bar{\delta}\{E\left[\hat{\mu}|e_a|^4\right] + E\left[\hat{\mu}|e_a|^2e_a\right]\underbrace{E[v^*]}_{0} + \\
&\quad E\left[\hat{\mu}|e_a|^2e_a^*\right]\underbrace{E[v]}_{0} + E\left[\hat{\mu}|e_a|^2\right]\sigma_v^2 + E\left[\hat{\mu}|e_a|^2e_a^*\right]\underbrace{E[v]}_{0} + E\left[\hat{\mu}|e_a|^2\right]\sigma_v^2 + \\
&\quad E\left[\hat{\mu}(e_a^*)^2\right]\underbrace{E\left[v^2\right]}_{0} + E[\hat{\mu}e_a^*]\,\sigma_v^2\underbrace{E[v]}_{0}\} && (2.98)\\
&= (\delta + 4\bar{\delta}\sigma_v^2)\, E\left[\hat{\mu}|e_a|^2\right] + 2\bar{\delta}\, E\left[\hat{\mu}|e_a|^4\right] && (2.99)\\
&\approx (\delta + 4\bar{\delta}\sigma_v^2)\, E\left[\hat{\mu}|e_a|^2\right] && (2.100)\\
&\approx \acute{b}\, E\left[\hat{\mu}|e_a|^2\right] && (2.101)
\end{aligned}$$

From equations (2.98) to (2.101), we used the following features:

- $v[t]$ is independent of $\boldsymbol{x}[t]$ and $e_a[t]$

- $E[v] = E[v^*] = 0$ and $E\left[|v|^2\right] = \sigma_v^2$
- $v[i]$ is assumed to be circular, i.e. $E\left[v^2\right] = 0$
- third and higher order terms of e_a are neglected since e_a is small at steady-state
- a new variable is defined,

$$\acute{b} \triangleq \delta + 4\bar{\delta}\sigma_v^2 \tag{2.102}$$

Following the same order as in the real-valued data subsection we proceed now by computing $E\left[\|\boldsymbol{x}\|^2_{\boldsymbol{\Omega}}|g|^2\right]$

$$\begin{aligned}
|g|^2 &= \hat{\mu}^2\delta^2|e|^2 + 4\hat{\mu}^2\delta\bar{\delta}|e|^4 + 4\hat{\mu}^2\bar{\delta}^2|e|^6 && (2.103)\\
|e|^2 &= |e_a|^2 + e_a^* + e_a v^* + |v|^2 && (2.104)\\
|e|^4 &= |e_a|^4 + 4|e_a|^2|v|^2 + 2|e_a|^2[e_a v^* + e_a^* v] + 2|v|^2[e_a v^* + e_a^* v] + v^2(e_a^*)^2 + \\
&\quad (v^*)^2 e_a^2 + |v|^4 && (2.105)\\
|e|^6 &= |e_a|^6 + |v|^6 + 3|e_a|^4[e_a v^* + e_a^* v] + 3|v|^4[e_a v^* + e_a^* v] + |e_a|^2 e_a^* v[3e_a^* v + 2e_a v^*] + \\
&\quad |e_a|^2 e_a v^*[3e_a v^* + 2e_a^* v] + 5|v|^2|e_a|^4 + 5|e_a|^2|v|^4 + |v|^2 e_a^* v[3e_a^* v + 2e_a v^*] + \\
&\quad |v|^2 e_a v^*[3e_a v^* + 2e_a^* v] + 9|v|^2|e_a|^2[e_a v^* + e_a^* v] + (e_a^*)^3 v^3 + e_a^3(v^*)^3 && (2.106)
\end{aligned}$$

Substitute (2.104),(2.105) and (2.106) in (2.103) and use the same features listed while computing $E[e_a^* g]$.

$$\begin{aligned}
E\left[\|\boldsymbol{x}\|^2_{\boldsymbol{\Omega}}|g|^2\right] &\approx \acute{a}\,E\left[\hat{\mu}^2\|\boldsymbol{x}\|^2_{\boldsymbol{\Omega}}\right] + \acute{c}\,E\left[\hat{\mu}^2\|\boldsymbol{x}\|^2_{\boldsymbol{\Omega}}|e_a|^2\right] + 8\delta\bar{\delta}\,E\left[\|\boldsymbol{x}\|^2_{\boldsymbol{\Omega}}|v|^2(e_a v^* + e_a^* v)\right] + \\
&\quad 12\bar{\delta}^2\,E\left[\|\boldsymbol{x}\|^2_{\boldsymbol{\Omega}}|v|^4(e_a v^* + e_a^* v)\right] && (2.107)\\
\acute{a} &= \delta^2\sigma_v^2 + 4\delta\bar{\delta}\tau_v^4 + 4\bar{\delta}\tau_v^6 && (2.108)\\
\acute{c} &= \delta^2 + 16\delta\bar{\delta}\sigma_v^2 + 36\bar{\delta}^2\tau_v^4 && (2.109)
\end{aligned}$$

We are now ready to utilize the variance relation in (2.61) by combining (2.101) and (2.107).

$$\begin{aligned}
2\acute{b}\,E\left[\hat{\mu}|e_a|^2\right] &= \mu\acute{a}\alpha_{\boldsymbol{x}} + \mu\acute{c}\,E\left[\hat{\mu}^2\|\boldsymbol{x}\|^2_{\boldsymbol{\Omega}}|e_a|^2\right] + 8\mu\delta\bar{\delta}\,E\left[\|\boldsymbol{x}\|^2_{\boldsymbol{\Omega}}|v|^2(e_a v^* + e_a^* v)\right] + \\
&\quad 12\mu\bar{\delta}^2\,E\left[\|\boldsymbol{x}\|^2_{\boldsymbol{\Omega}}|v|^4(e_a v^* + e_a^* v)\right] && (2.110)
\end{aligned}$$

We will again investigate two conditions on the step-size μ.

Sufficiently small μ:

We can ignore again last three terms in (2.110) because they contain e_a, which is small at steady-state when μ is small.

$$E\left[\hat{\mu}|e_a|^2\right] = \frac{\mu\acute{a}\alpha_{\boldsymbol{x}}}{2\acute{b}} \tag{2.111}$$

$$\xi^{KPNLMMN} = \frac{\mu\acute{a}\alpha_{\boldsymbol{x}}}{2\acute{b}\eta_{\boldsymbol{x}}} \qquad (\textit{separation principle}) \tag{2.112}$$

$$\xi^{KPNLMMN} = \frac{\mu\acute{a}}{2\acute{b}} \qquad (\textit{small } \epsilon \quad (\alpha_{\boldsymbol{x}} = \eta_{\boldsymbol{x}})) \tag{2.113}$$

$$\begin{aligned} \xi^{KPNLMF} &= \frac{\mu}{2}\frac{4\tau_v^6}{4\sigma_v^2} \\ &= \frac{\mu\tau_v^6}{2\sigma_v^2} \end{aligned} \tag{2.114}$$

For KPNLMF, $\delta = 0$ and this leads to $\acute{a} = 4\tau_v^6$ from (2.108), $\acute{b} = 4\sigma_v^2$ from (2.102) used in (2.114).

For KPNLMS, $\delta = 1$ and consequently $\acute{a} = \sigma_v^2, \acute{b} = 1$. So,

$$\xi^{KPNLMS} = \frac{\mu\sigma_v^2}{2} \tag{2.115}$$

which is the expected result with the assumption of small μ. This confirms our unified investigation of excess MSE for KPNLMS in real-valued and complex-valued data. We see here that real-valued and complex-valued data gives the same excess MSE expression for KPNLMS.

Large μ:

Last two terms in (2.110) cancel because they result zero like in real-valued data subsection. We assume small ϵ and use separation principle.

$$2\acute{b}\,E\left[\hat{\mu}|e_a|^2\right] = \mu\acute{a}\alpha_{\boldsymbol{x}} + \mu\acute{c}\,E\left[\hat{\mu}^2\|\boldsymbol{x}\|^2_{\boldsymbol{\Omega}}|e_a|^2\right] \tag{2.116}$$

$$2\acute{b}\eta_{\boldsymbol{x}}\,E\left[|e_a|^2\right] = \mu\acute{a}\alpha_{\boldsymbol{x}} + \mu\acute{c}\alpha_{\boldsymbol{x}}\,E\left[|e_a|^2\right] \qquad (\textit{separation principle}) \tag{2.117}$$

$$E\left[|e_a|^2\right] = \frac{\mu\acute{a}\alpha_{\boldsymbol{x}}}{2\acute{b}\eta_{\boldsymbol{x}} - \mu\acute{c}\alpha_{\boldsymbol{x}}} \tag{2.118}$$

$$\xi^{KPNLMMN} = \frac{\mu\acute{a}}{2\acute{b} - \mu\acute{c}} \qquad (\textit{small } \epsilon \quad (\alpha_{\boldsymbol{x}} = \eta_{\boldsymbol{x}})) \tag{2.119}$$

$$\xi^{KPNLMF} = \frac{4\mu\tau_v^6}{8\sigma_v^2 - 36\mu\tau_v^4} \qquad (\acute{c} = 36\tau_v^4 \textit{ for KPNLMF}) \tag{2.120}$$

$$\xi^{KPNLMF} = \frac{\mu\tau_v^6}{2\sigma_v^2 - 9\mu\tau_v^4} \tag{2.121}$$

Both small μ and large μ produced a small excess MSE for KPNLMF. Large μ results in slightly larger excess MSE as expected.

The results in real-valued data (2.90),(2.93) and complex-valued data (2.114),(2.121) prove that the KPNLMF algorithm converges to the desired solution with a small excess MSE.

$\acute{c} = 1$ for KPNLMS ($\delta = 1$). Hence, $\xi^{KPNLMS} = \frac{\mu\sigma_v^2}{2-\mu}$ approving the result in (2.69).

2.4 Simulation Results

This section contains numerical examples. Performance of the KPNLMF algorithm is compared to the performances of KPNLMS and NLMF algorithms in the following subsections.

2.4.1 KPNLMS-KPNLMF

In this subsection, we compare the performances of the KPNLMS and the KPNLMF algorithms. In [2], Walach and Widrow studied the LMF algorithm and compared the performance of the LMF algorithms to the performance of the LMS algorithm. Since we work with the Krylov-variants of the LMS and the LMF algorithms, we can expect the same comparison results with the ones in [2]. Our Krylov-proportionate LMS and LMF algorithms implement LMS and LMF like in [2], but the normalized versions in the Krylov subspace coordinates. Walach and Widrow looked at the performances of the LMS and the LMF algorithms at different noise contexts. Probability density functions (PDFs) of the investigated noise types are depicted in Fig. 2.5. Gaussian, uniform, sinusoidal and square wave probability densities are chosen because of their practical importance.

Here, we define a parameter $\alpha = \frac{M_{LMS}}{M_{LMF}}$ to measure the ratio between the mismatches of LMS and LMF where M_{LMS} is the mismatch of LMS and M_{LMF} is the one of LMF. Table 2.1 shows the α values in 4 different noisy environments [2].

	Gaussian	Uniform	Sine Wave	Square Wave
α	0.6	2.3	3.6	9

Table 2.1: Values of the mismatch ratio between LMS and LMF, $\alpha = \frac{M_{LMS}}{M_{LMF}}$ under different noise types

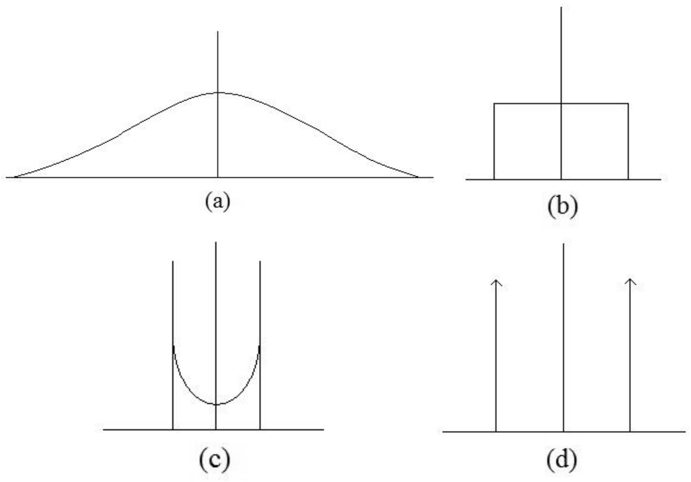
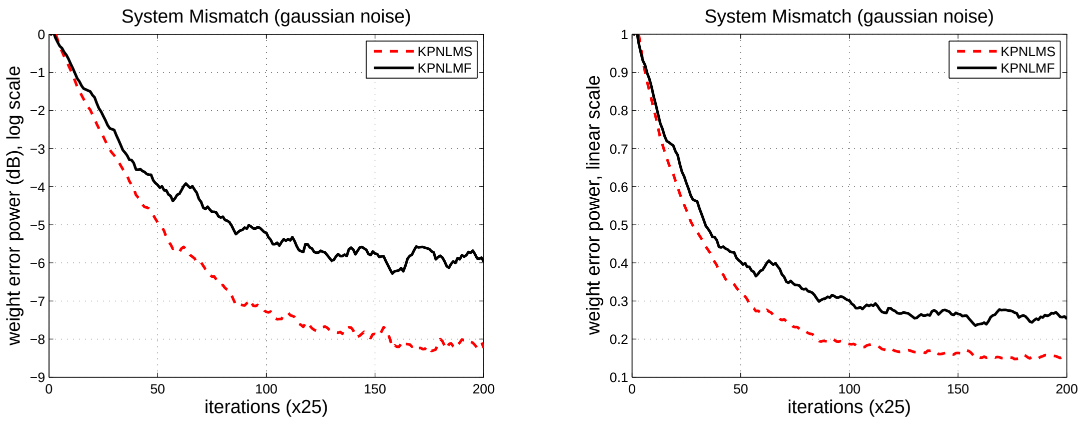


Figure 2.5: PDFs of the investigated noise types. (a)Gaussian density, (b)Uniform density, (c)Sinusoidal density, (d)Square wave density

According to the Table 2.1, KPNLMF should have a mismatch value which is $10log_{10}9 \approx 9.5$ dB less than the mismatch of KPNLMS under square wave type noise ($v[t] = \pm a$) for the same speed of convergence since we expect the same results for α with our Krylov-proportionate LMS and LMF algorithms. Similarly, under uniformly distributed noise M_{LMF}(dB) $\approx M_{LMS}$(dB)$-$3.6dB and under sine wave type noise M_{LMF}(dB) $\approx M_{LMS}$(dB)$-$5.6dB. However, under Gaussian type noise M_{LMF}(dB) $\approx M_{LMS}$(dB) + 2.2dB which means KPNLMS performs better than KPNLMF under Gaussian noise. On the other hand, KPNLMF beats KPNLMS considerably in terms of *weight error power* (mismatch) when the system noise has one of the distributions among uniform, sinusoidal and square wave. Our simulation results depicted in Fig. 2.6 through Fig. 2.9 agree this statement.

The examples in [2] are implemented here. In Fig. 2.6 through 2.9, we try to estimate the system with an impulse response given by $w^T = [0.1, 0.2, 0.3, 0.4, 0.5, 0.4, 0.3, 0.2, 0.1]$ where $m = 9$. A random input signal which is white and of unit power is used. Step sizes $\mu_{KPNLMS} = 1.8 \times 10^{-3}$ and $\mu_{KPNLMF} = 0.3 \times 10^{-5}$ are used to yield the same speed of convergence. For parameters that only belong to our Krylov-proportionate algorithms, we use $\lambda = 2$, $\gamma = 0.5$ in our experiments. $\epsilon = 0.0001$. Weight error powers, i.e. mismatches, are averaged over 200 independent trials. The experiments are carried out in low SNR conditions. Gaussian and uniform noises are simulated as white random processes of power 100. Sinusoidal noise is generated according to the model $v[t] = a\cos(wt)$ where $a = 10\sqrt{2}$ and $w = \pi/4$. Square wave noise is randomly modeled as $v[t] = \pm 10$. Weight error powers are delineated after averaging over periods that are 25 iterations-long.



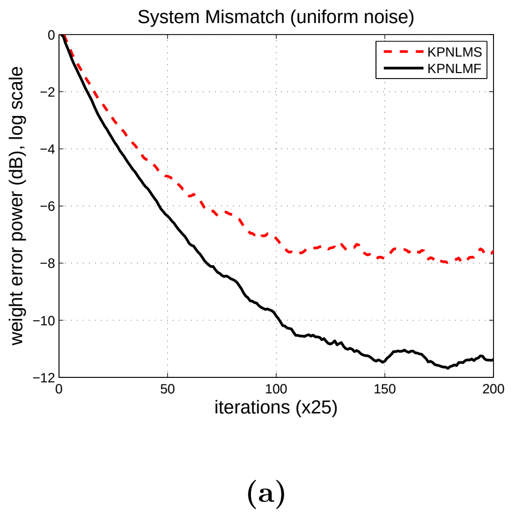
(a) **(b)**

Figure 2.6: Mismatch graphs for Gaussian noise case. (a)Log scale graph. At steady-state, $M_{KPNLMS} \approx -8.1$dB and $M_{KPNLMF} \approx -5.9$dB. So, $M_{KPNLMF} \approx M_{KPNLMS} + 2.2$dB which is a perfect match with the theoretical result obtained in [2]. (b)Linear scale graph.

Our experimental results are in a good harmony with the theoretical ones that are obtained in [2]. We provide the log scale graphs in Fig. 2.6 through 2.9 to illustrate this strong consistency. We also provide linear scale graphs for all types of noisy environments studied here to show that mismatch values are in steady-state.

2.4.2 NLMF-KPNLMF

System to be identified has a random dispersive impulse response with length m where $m = 50$ in the comparison between the NLMF and KPNLMF algorithms. Elements of the impulse response vector sum up to 1. Here, $\lambda = 4$ ($\ll m = 50$) and 200 independent trials



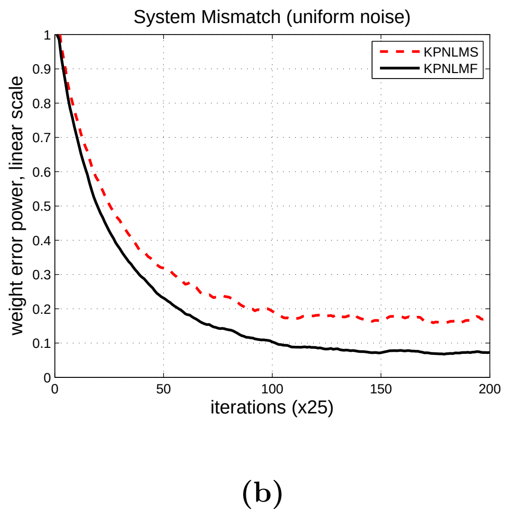


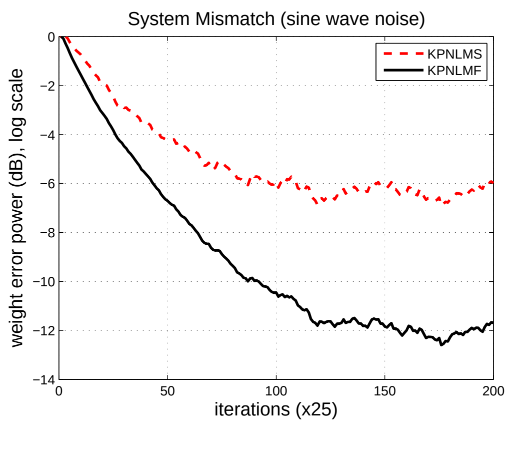
Figure 2.7: Mismatch graphs for uniform noise case. (a)Log scale graph. At steady-state, $M_{KPNLMS} \approx -7.9$dB and $M_{KPNLMF} \approx -11.5$dB. So, $M_{KPNLMF} \approx M_{KPNLMS} - 3.6$dB which is a perfect match with the theoretical result obtained in [2]. (b)Linear scale graph.

are performed with uniformly distributed, zero mean noise process and white inputs. We set signal-to-noise ratio, SNR $= 10\log_{10}\frac{E[x[t]^2]}{E[v[t]^2]} \approx 22dB$. The mean square error and the system mismatch (normalized weight error power) averaged over 200 trials are plotted as performance measures. The figures are in dB scale. As the regularization constant, we use $\epsilon = 0.0001$ for all algorithms.

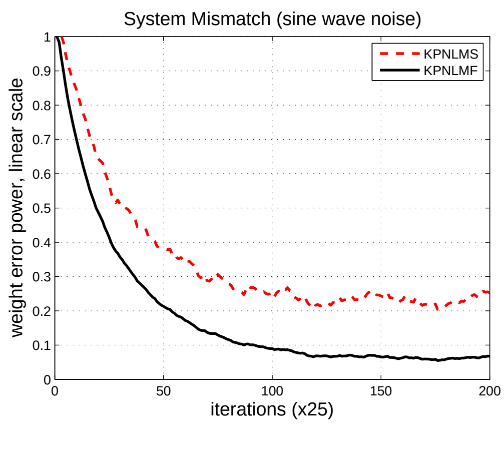
Fig. 2.10 compares the KPNLMF and the NLMF algorithms in terms of the MSE. The KPNLMF algorithm surpasses the NLMF algorithm in terms of the MSE performance. Two methods are also compared in the system mismatch setup and again the KPNLMF algorithm yields considerably superior performance with respect to the NLMF algorithm. The system mismatch performances for both algorithms are shown in Fig. 2.11. KPNLMF enjoys the superior performance related to the sparse structure of the unknown impulse response as a result of the Krylov projection technique.

2.5 Conclusion

The *proportionate* and *Krylov-proportionate normalized least mean fourth algorithms* are proposed in this chapter. It is shown through simulations that the *KPNLMF* algorithm



(a)

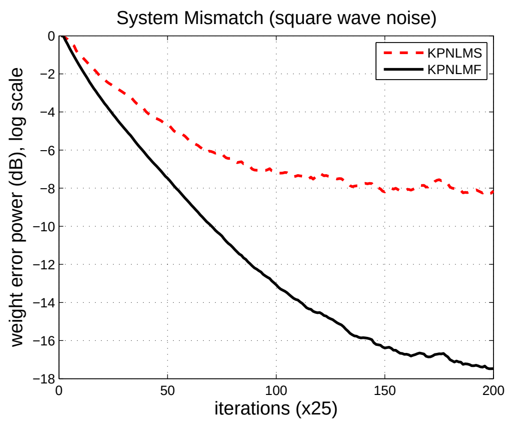


(b)

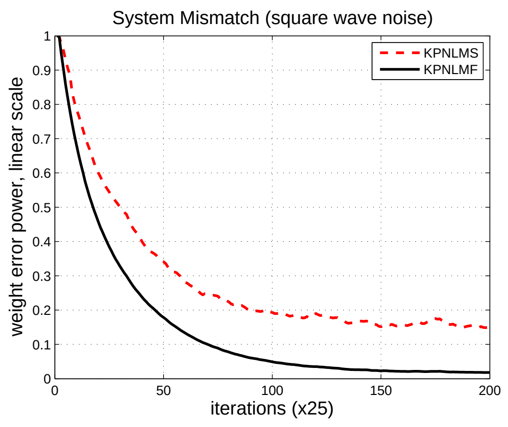
Figure 2.8: Mismatch graphs for sine wave noise case. (a)Log scale graph. At steady-state, $M_{KPNLMS} \approx -6.4$dB and $M_{KPNLMF} \approx -12$dB. So, $M_{KPNLMF} \approx M_{KPNLMS} - 5.6$dB which is a perfect match with the theoretical result obtained in [2]. (b)Linear scale graph.

outperforms the *NLMF* algorithm in convergence while having only linear computational complexity. The *KPNLMF* algorithm performs better than the *NLMF* algorithm since it benefits from both the sparse structure of the Krylov-projected unknown system and the proportional update technique. The system mismatch performance of *KPNLMF* is also compared to the performance of *KPNLMS* in several types of noisy environments that have practical importance. Although the *KPNLMF* algorithm is similar with the *KPNLMS*, simulation results demonstrated that the *KPNLMF* algorithm attains smaller amounts of weight error power, i.e. mismatch, in case the system noise has a probability distribution among uniform, sinusoidal and square wave distributions. *KPNLMF* achieves this superior performance against *KPNLMS* when they converge at the same speed. For the Gaussian noise case *KPNLMS* has smaller mismatch values than *KPNLMF*.

Steady-state MSE performances of KPNLMS and KPNLMF are investigated and excess MSE of both algorithms are found to be small. We showed that both algorithms converge to the desired solution according to the steady-state MSE criterion.



(a)



(b)

Figure 2.9: Mismatch graphs for uniform noise case. (a)Log scale graph. At steady-state, $M_{KPNLMS} \approx -8$dB and $M_{KPNLMF} \approx -17.5$dB. So, $M_{KPNLMF} \approx M_{KPNLMS} - 9.5$dB which is a perfect match with the theoretical result obtained in [2]. (b)Linear scale graph.

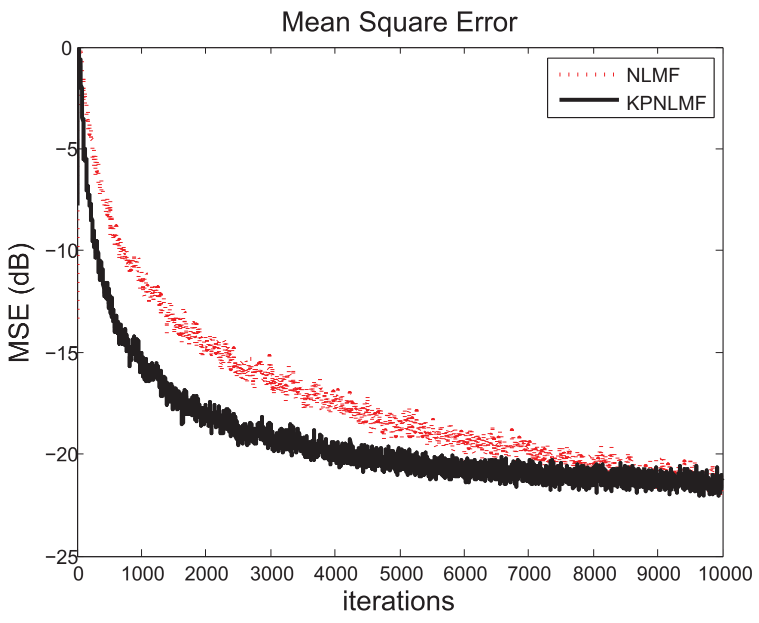


Figure 2.10: MSE comparison between NLMF and KPNLMF

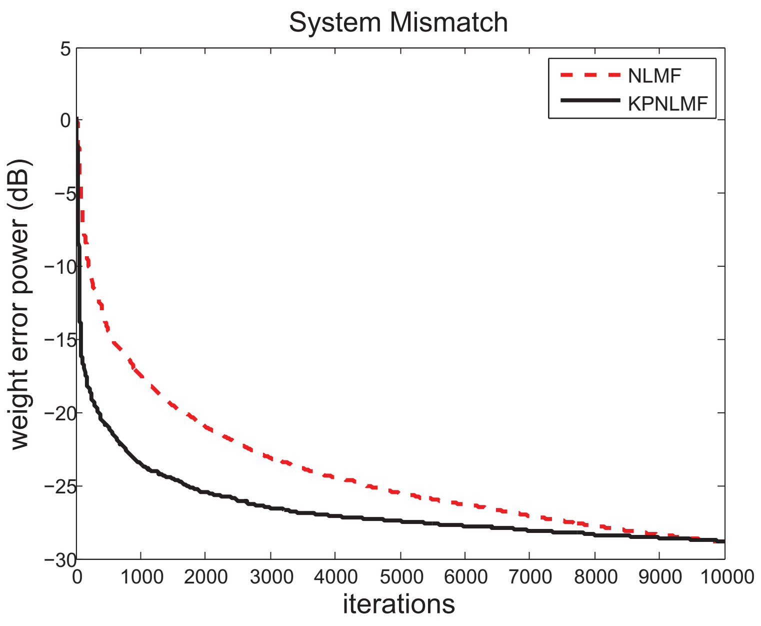


Figure 2.11: System mismatch comparison between NLMF and KPNLMF

Chapter 3

COMPETITIVE NONLINEAR ADAPTIVE PREDICTION UNDER ADDITIVE NOISE

In this chapter, we consider sequential (online) nonlinear prediction of an arbitrary, deterministic and bounded signal from its noise-corrupted past samples under square error loss and we propose a novel algorithm called *Competitive Randomized Noisy Nonlinear Predictor (CRNNP)*.

In the next section, the system that is studied throughout the chapter is described and the frameworks in which the described system is handled are explained. Later in Section 3.2, we first summarize the notion of context-trees. We then introduce CRNNP which is a randomized predictor constructed using context-trees, competes against all piecewise affine models defined on the context-tree and requires a computational complexity only linear in the depth of the context-tree per prediction. Simulations are performed to illustrate the performance of the introduced algorithm using chaotic signals. We finally conclude the chapter in Section 3.4.

3.1 System Description

In this fundamental signal processing problem [25], a bounded deterministic signal $\{x[t]\}_{t\geq 1}$, $|x[t]| \leq b_x$, $x[t] \in \mathbb{R}$, $0 < b_x < \infty$ is observed through an additive noise channel, $Y[t] = x[t] + N[t]$, where $\{N[t]\}_{t\geq 1}$ is an i.i.d., bounded noise process with variance σ_n^2 such that $|N[t]| \leq b_n$, $N[t] \in \mathbb{R}$, $0 < b_n < \infty$ with probability 1. Hence, $|Y[t]| \leq b_y$, $b_y \overset{\triangle}{=} b_x + b_n$ with probability 1. Then, the underlying signal $x[t]$ is predicted using the noise-corrupted past samples $\{Y[1], \ldots, Y[t-1]\}$ at each time t.

We emphasize that although we desire to predict the underlying signal $\{x[t]\}_{t\geq 1}$ and the performance measure including the loss function is defined with respect to $\{x[t]\}_{t\geq 1}$, the desired clean signal $\{x[t]\}_{t\geq 1}$ is not available for prediction or training. In this sense, this noisy framework differs from common classical adaptive signal processing approaches [4], where the desired clean signal, certain statistics or past samples are usually available for training or constructing predictions. This noisy framework is explained in the following subsection.

In this chapter, we refrain from making stochastic assumptions on the desired signal $\{x[t]\}_{t\geq 1}$ and require uniformly good performance for any deterministic signal $\{x[t]\}_{t\geq 1}$. If the desired signal $x[t]$ and the noise $N[t]$ are random processes, conditional mean $E\left[x[t]|y_1^{t-1}\right]$, $y_1^{t-1} \triangleq \{y[1], ..., y[t-1]\}$ is the optimal predictor of $x[t]$ that minimizes the mean-square error (MSE) between the desired signal and the prediction [4]. Since we make no such stochastic assumptions on the desired signal, we introduce a competitive framework, which is detailed in Subsection 3.1.2, in order to define a meaningful performance measure.

Another framework we consider in this chapter is the one where the underlying competition class is the class of certain nonlinear models, i.e., piecewise affine models represented on a context-tree. Although we discuss only affine models, as shown in the next section, one can assign arbitrary predictors (or regressors) to each region. The affine models are specifically used to yield smoothly varying arbitrary nonlinear models. This nonlinear framework which is based on context-tree structure is represented in Subsection 3.1.3.

Randomized selection framework is the last framework that is employed in this thesis. Sequential decision problem is the subject that is handled in this framework. A decision should be made among several choices in this problem. Our algorithm works in a randomized fashion and it uses randomization weights to select one of the possible choices as the output. Subsection 3.1.4 is dedicated to this framework.

To this end, we introduce a novel randomized prediction algorithm, CRNNP based on context-trees that uses only the past noisy samples of a desired signal and has computational complexity only linear in the depth of the context-tree per prediction. Without ever observing the desired clean signal, this algorithm will achieve the performance of the best piecewise affine predictor that can both choose its partition of the past observations space (from a class of a doubly exponential number of possible partitions) and tune the parameters of the affine models for these piecewise regions using the clean desired signal. We consider the square error loss function, however, our results can be generalized to several different loss functions, such as those considered in [26].

3.1.1 Noisy Framework

The framework in which additive noise exists on an individual deterministic sequence is introduced in [27] for binary prediction. The results in [27] are extended to the filtering problem in [28]. We are inspired by [27] and [28] to extend the results presented in [29], [30] and [15] to the noise-corrupted nonlinear prediction problem. This work is the nonlinear version of the noise-corrupted prediction problem in [25]. Context-tree structure is used to incorporate the nonlinearity. [31] also deals with the same problem.

3.1.2 Competitive Framework

This competitive approach has extensive roots in machine learning [32], [33], [34], adaptive signal processing [15] and information theory [35], [36] literature. Much of these works build on the Hannan's influential work on prediction of individual sequences [37]. Such competitive framework for sequential prediction of deterministic sequences was examined in [26] and [29] against a finite number of predictors, in [38] against the class of fixed-order linear models, and finally in [30] and [15] against switching linear and certain nonlinear models. However, in these past approaches [26], [38], [30] and [15] there is no consideration for noise, i.e. the desired clean signal is available.

In this competitive framework, we have, say m sequential (online) prediction algorithms as the competition class producing outputs $\{\hat{X}_k[t]\}_{t\geq 1}$, $k = 1, \ldots, p$, that "hypothetically" work in parallel to predict the underlying signal $\{x[t]\}_{t\geq 1}$. At each time t, each sequential algorithm suffers the loss $(x[t] - \hat{X}_k[t])^2$ (which is not available to us since $\{x[t]\}_{t\geq 1}$ is not observable). Our goal is then to find a sequential predictor that asymptotically achieves the performance of even the best algorithm in this class uniformly for any deterministic, bounded and arbitrary signal. Specifically, we seek a sequential predictor, say $\hat{X}[t]$, that has access to only noisy past samples $\{Y[1], \ldots, Y[t-1]\}$, predictions of the constituent algorithms $\{\hat{X}_k[1], \ldots, \hat{X}_k[t]\}$, $k = 1, \ldots, p$, never observes $\{x[t]\}_{t\geq 1}$ and satisfies

$$\lim_{n\to\infty} \frac{1}{n} E\left[\sum_{t=1}^{n}(x[t] - \hat{X}[t])^2 - \sum_{t=1}^{n}(x[t] - \hat{X}_k[t])^2\right] = 0, \tag{3.1}$$

for all k and n when it is used to predict any $\{x[t]\}_{t\geq 1}$. Here, the expectation is with respect to the noise process and the randomization of the introduced algorithm.

3.1.3 Context-tree Framework

As an example for the usage of context-tree structure, look at Fig. 3.1. We divide the space of the most recent observation space $[-b_y, b_y]$ (which $Y[t-1]$ belongs to) into four disjoint regions $\Delta_1, \ldots, \Delta_4$ such that $[-b_y, b_y] = \bigcup_{i=1}^{4} \Delta_i$ and assign each region an affine model say $(w_i Y[t-1] + c_i)$, $w_i, c_i \in \mathbb{R}$, $i = 1, \ldots, 4$. These assignments define a piecewise affine predictor where the prediction at each time t is given by $(w_i Y[t-1] + c_i)$ if $Y[t-1] \in \Delta_i$. For all $w_i, c_i \in \mathbb{R}$, one can define similar piecewise affine predictors yielding a competition class that has a continuum of predictors. Although one can approximate smoothly varying nonlinear functions by increasing the number of regions and the number of past samples used in prediction in this piecewise affine model, the boundaries of the piecewise regions are fixed. To make the boundaries of the piecewise regions also a design parameter, we will use the notion of context-trees to represent a doubly exponential number different partitions of the past observations space in the next section.

3.1.4 Randomized Selection Framework

In this framework, we serve for certain signal processing problems where one is expected to choose a particular strategy from a class of strategies at each time, instead of producing a new outcome based on the outcomes of the constituent algorithms. A well-studied problem that fits this framework is tracking a finite class of finite-delay scalar quantizers [39], [40].

The functional forms of the randomization weights of the introduced algorithm are similar to the weights used in [15] to construct weighted predictions. However, note that the algorithm of [15] trains on and uses the past observations of the clean desired signal $\{x[t]\}_{t\geq 1}$, which is unavailable here. Furthermore, we use these weights to define a randomized algorithm instead of using convex combination ideas as in [15].

3.2 Algorithm Description

In this section, as an illustrative example for the notion of context-tree structure, we present a binary context-tree to partition the space of only the most recent past observation, i.e., $[-b_y, b_y]$ where $Y[t-1]$ belongs to. As shown in Fig. 3.1, a depth-D binary context-tree ($D = 2$ in this figure) has 2^D leaves and $2^{D+1} - 1$ nodes. Each node on the context-tree, if

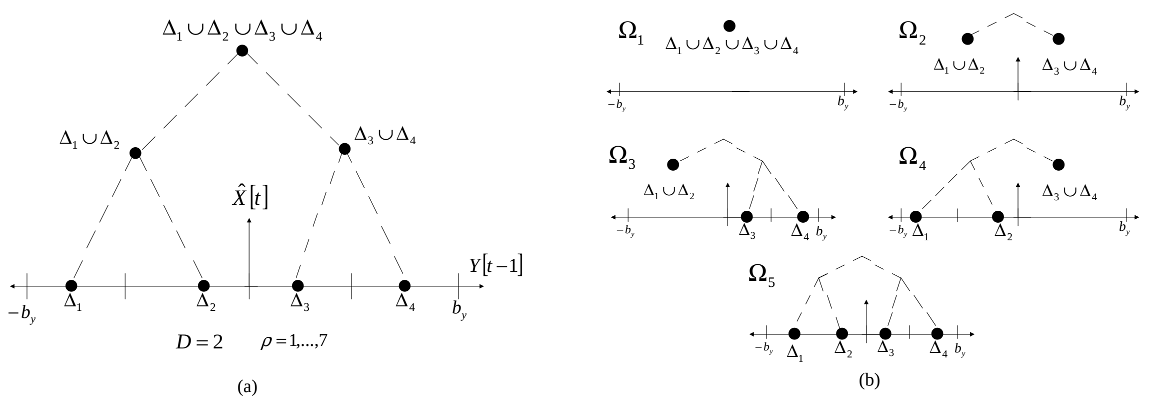


Figure 3.1: (a) A binary context-tree that partitions $[-b_y, b_y]$. This tree has depth $D = 2$ with 4 leaves and 7 nodes. Each node is assigned a region as the union of the regions assigned to its children. (b) The complete sub-trees along with the partitions they define. A depth-D context-tree defines $m \approx 1.5^{2^D}$ such sub-trees or partitions.

it is not a leaf node, has two children: the left hand side child and the right hand side child. We use this context-tree to define different partitions of $[-b_y, b_y]$ as follows. We first assign each leaf of the context-tree a different region of $[-b_y, b_y]$ as seen in Fig. 3.1. Each node on the context-tree is then assigned the region which is constructed as the union of the regions assigned to its children. On this context-tree, one can define a doubly exponential number, $m \approx 1.5^{2^D}$ [15], of different prunings or "complete" sub-trees. As an example, for a depth-2 context-tree, we provide 5 different sub-trees in Fig. 3.1. We call these sub-trees "complete" since the union of the regions assigned to the leaves of a sub-tree (which are the nodes or the leaves of the original context-tree) yields $[-b_y, b_y]$. Hence, a sub-tree along with the regions assigned to its leaves defines a partition of $[-b_y, b_y]$. Given a depth-D binary context-tree, we get a doubly exponential number $m \approx 1.5^{2^D}$ of such partitions, say Ω_k, $k = 1, \ldots, m$. For each partition Ω_k, we represent the constituent regions as $\Omega_k \overset{\triangle}{=} \{\Lambda_{k,1}, \ldots, \Lambda_{k,K_k}\}$ such that $[-b_y, b_y] = \bigcup_i \Lambda_{k,i}$, K_k is the number of the leaves in the partition and $\Lambda_{k,i}$ are the regions assigned to the leaves of the partition, e.g., for Ω_2,we have $\Lambda_{2,1} = \Delta_1 \cup \Delta_2$, $\Lambda_{2,2} = \Delta_3 \cup \Delta_4$ and $K_2 = 2$.

Suppose, given this context-tree, we assign each node ρ, $\rho = 1, \ldots, 2^{D+1} - 1$, a sequential predictor $\hat{X}_\rho[t]$ and define sequential predictors $\hat{X}_{\Omega_k}[t]$, $k = 1, \ldots, m$, corresponding to each partition Ω_k as: $\hat{X}_{\Omega_k}[t] = \hat{X}_\rho[t]$ if $Y[t-1] \in \Lambda_{k,j}$ (for some j) and $\Delta_\rho = \Lambda_{k,j}$. We initially define these m sequential predictors as our competition class. Later, by selecting $\hat{X}_\rho[t]$'s as certain sequential affine predictors, we will demonstrate that our algorithm asymptotically achieves the performance of the best piecewise affine model which can tune both its partitioning of the real line as well as the affine models in each region to $\{x[t]\}_{t\geq 1}$.

To this end, we introduce the randomized algorithm, CRNNP in Fig. 3.2, i.e., $\hat{X}[t]$, that is constructed using context-tree weighting method [41]. This randomized algorithm hypothetically constructs all $\hat{X}_{\Omega_k}[t]$, $k = 1, \ldots, m$, and run these in parallel. At each time t, the final estimation $\hat{X}[t]$ selects one of the outputs $\hat{X}_{\Omega_k}[t]$ to repeat it as its prediction, where the selection weights are calculated proportional to the performance of each $\hat{X}_{\Omega_k}[t]$ on the past data. However, note that, although there are m different piecewise competing algorithms, at each time t, each $\hat{X}_{\Omega_k}[t]$ is equal to one of the $D+1$ node predictions that $Y[t-1]$ belongs to. Hence, as shown in [15], at each time t, for the nodes that $Y[t-1]$ belongs to (these nodes are stored in vector $\boldsymbol{v}$ in Fig. 3.2), all the weights assigned to $\hat{X}_{\Omega_k}[t]$, $k = 1, \ldots, m$ can be merged using certain functions of node performance. These functions are represented as $F_\rho[t]$ and $\Gamma_\rho[t]$ in Fig. 3.2, and updated recursively in (line C), (line D) and (line E) in Fig. 3.2 with computational complexity only linear in depth of the context-tree. At each time t, these functions that reflect the combined prediction performance of that node on the past data are used to construct the probabilities $\boldsymbol{\eta}$ that are used for randomization. The CRNNP algorithm introduced in Fig. 3.2 satisfies:

Theorem: Let $Y[t] \stackrel{\triangle}{=} x[t] + N[t]$ represents the observation sequence such that $\{x[t]\}_{t\geq 1}$ is the desired deterministic signal with $|x[t]| \leq b_x$ and $N[t]$ is i.i.d. with variance σ_n^2, $|N[t]| \leq b_n$, i.e., $|Y[t]| \leq b_y = b_x + b_n$ with probability 1. The sequential randomized prediction algorithm presented in Fig. 3.2, which uses only the past noisy observations $\{Y[1], \ldots, Y[t-1]\}$, $\{\hat{X}_\rho[1], \ldots, \hat{X}_\rho[t]\}$, $\rho = 1, \ldots, 2^{D+1} - 1$ and never observes $\{x[t]\}_{t\geq 1}$ or prediction errors $(x[t] - \hat{X}[t])$, $(x[t] - \hat{X}_\rho[t])$, achieves

$$\frac{1}{n} E\left[\sum_{t=1}^{n} (x[t] - \hat{X}[t])^2 - \sum_{t=1}^{n} (x[t] - \hat{X}_{\Omega_k}[t])^2\right] \leq O\left(\frac{2^{\frac{D+1}{2}}}{\sqrt{n}}\right) \qquad (3.2)$$

for all n and any partition Ω_k, when it is applied to predict any $\{x[t]\}_{t\geq 1}$. Here, the expectation is with respect to the noise process and randomization.

```
A Pseudo-code of the CRNNP Algorithm:
% Initialize
For $\rho = 1, \ldots, 2^{D+1} - 1$: $F_\rho[0] = \epsilon_1$, $\Gamma_\rho[0] = \epsilon_2$, where $\epsilon_1$ and $\epsilon_2$ are small positive constants.   (line A)
% Proceed
For $t = 1, \ldots$
    Find nodes such that $Y[t-1] \in \Delta_\rho$ and store them in the vector $\boldsymbol{v}$ starting from the top node to the leaf node.
    % Form the prediction.
    $\boldsymbol{z}(1) = 1/2$.
    For $i = 2 : D+1$: $\boldsymbol{z}(i) = \frac{1}{2} F_{(\boldsymbol{v}(i),s)}[t-1]\boldsymbol{z}(i-1)$   (line B)
    (where $(\boldsymbol{v}(i), s)$ is the sibling node of $\boldsymbol{v}(i)$, i.e., they are the children of the same node).
    Construct a weight vector $\boldsymbol{\eta}$ of size $D+1$ as $\boldsymbol{\eta}(i) = (\boldsymbol{z}(i)\Gamma_{\boldsymbol{v}(i)}[t-1])/\left(F_{\boldsymbol{v}(1)}[t-1]\right)$.
    Select randomly one of the entries of $\boldsymbol{v}$ by using the weights in $\boldsymbol{\eta}$, i.e., $\boldsymbol{v}(i)$ is selected with probability $\boldsymbol{\eta}(i)$.
    Set $\hat{X}[t] = \hat{X}_{\boldsymbol{v}(i)}[t]$ if $i$ is the selected entry.
    % Update after observing $Y[t]$. All the other nodes remain the same.
    For $i = D+1, \ldots, 1$,
        $\Gamma_{\boldsymbol{v}(i)}[t] = \Gamma_{\boldsymbol{v}(i)}[t-1] \exp(-a(Y[t] - \hat{X}_{\boldsymbol{v}(i)}[t])^2)$   (line C)
        If $i == D+1$, then $F_{\boldsymbol{v}(i)}[t] = \Gamma_{\boldsymbol{v}(i)}[t]$   (line D)
        If $i \neq D+1$, then $F_{\boldsymbol{v}(i)}[t] = \frac{1}{2} F_{(\boldsymbol{v}(i),l)}[t] F_{(\boldsymbol{v}(i),r)}[t] + \frac{1}{2}\Gamma_{\boldsymbol{v}(i)}[t]$.   (line E)
        Update $\hat{X}_{\boldsymbol{v}(i)}[t+1]$ from $\hat{X}_{\boldsymbol{v}(i)}[t]$
```

Figure 3.2: A randomized sequential prediction algorithm using context-trees. Finding the nodes that $Y[t-1]$ belongs to require $O(D+1)$ operations since one only needs to find the leaf node that $Y[t-1]$ belongs and proceeds to the top. At each time t, the algorithm combines and updates the parameters of only $D+1$ node predictors with computational complexity $O(D+1)$.

To get the upper bound in (3.2), we need to set $a = \sqrt{8\ 2^{D+1} \ln 2/n}$ in Fig. 3.2. Note that although a is optimized over n, this need for *a priori* knowledge of n can be readily surpassed by applying the algorithm over exponentially increasing segments of $\{Y[t]\}_{t\geq 1}$. To achieve the performance of the best affine model with the best partition, we assign each node a special affine predictor studied in [25], which uses only the past samples $\{Y[1], \ldots, Y[t-1]\}$ that belong to that node as $\hat{X}_\rho[t] \stackrel{\triangle}{=} \tilde{w}_\rho[t-1]Y[t-1] + \tilde{c}_\rho[t-1]$, where

$$\left[\begin{array}{cc} \tilde{w}_\rho[t] & \tilde{c}_\rho[t] \end{array}\right]^T = \boldsymbol{R}^{-1}[t-1]\boldsymbol{p}[t-1],\ \boldsymbol{R}[t-1] \stackrel{\triangle}{=} \left(\sum_{l=1}^{t} \boldsymbol{Y}[l-1]\boldsymbol{Y}[l-1]^T s_\rho[l] + \delta I\right) \quad (3.3)$$

and $\boldsymbol{p}[t-1] \stackrel{\triangle}{=} \sum_{l=1}^{t-1} Y[l]\boldsymbol{Y}[l-1]s_\rho[l]$, where $\boldsymbol{Y}[l] = \left[\begin{array}{cc} Y[l] & 1 \end{array}\right]^T$, $s_\rho[l]$ is the indicator variable for node ρ, i.e., $s_\rho[l] = 1$ if $Y[l-1] \in \Delta_\rho$ otherwise $s_\rho[l] = 0$. The affine predictor in (3.3) is a least squares predictor that trains only on the observed data $\{Y[t]\}_{t\geq 1}$ that belongs to that node, i.e., that falls into the region Δ_ρ. Note that the update in (3.3) can be implemented with $O(D)$ computations using the matrix inversion lemma [4]. For the randomized predictor in Fig. 3.2 using these least squares predictors in each node, we have the following result:

Corollary: The sequential randomized prediction algorithm of Fig. 3.2, with the affine predictors (3.3) at each node ρ that only depends on the past noisy observations $\{Y[1], \ldots, Y[t-1]\}$ and never observes $\{x[t]\}_{t\geq 1}$, achieves

$$\frac{1}{n} E\left[\sum_{t=1}^{n}(x[t]-\hat{X}[t])^2 - \sum_{t=1}^{n}(x[t]-w_{k,d_k[t]}Y[t-1]+c_{k,d_k[t]})^2\right] \leq O\left(\sqrt{\frac{\ln n}{n}}\right)+O\left(\frac{2^{\frac{D+1}{2}}}{\sqrt{n}}\right) \tag{3.4}$$

for all n, $w_{k,j} \in \mathbb{R}$, $c_{k,j} \in \mathbb{R}$, $j = 1, \ldots, K_k$, for all partitions Ω_k, when it is applied to predict any $\{x[t]\}_{t\geq 1}$. Here, $d_k[t]$ is the selection variable for partition Ω_k such that if $Y[t-1] \in \Lambda_{k,j}$, $d_k[t] = j$.

Note that one can use the binary context-tree to partition the space of $\{Y[t-1], Y[t-2], \ldots, Y[t-r]\}$ for some r or use dth order affine predictors in each node that use $\{Y[t-1], \ldots, Y[t-d]\}$ as input regressor for some d. To use dth order regressors for affine prediction, one needs to update only (3.3). To partition $[-b_y, b_y]^r$, one needs to change the line in Fig. 3.2 that explains how to find the leaf node that $\{Y[t-1], \ldots, Y[t-r]\}$ belongs to. As explained in the caption of Fig. 3.2, the randomized algorithm has $O(D+1)$ computational complexity at each time t since finding the nodes used for prediction as well as updating the node predictors require only $O(D+1)$ additions and multiplications. The algorithm also has $O(2^{D+1})$ storage complexity to store weights and predictors corresponding to all nodes.

Remark 1: Note that the corollary holds for any $w_{k,j} \in \mathbb{R}$, $c_{k,j} \in \mathbb{R}$, $i = 1, \ldots, K_k$, even with the ones that are tuned by observing the whole $\{x[t]\}_{t\geq 1}$ and $\{Y[t]\}_{t\geq 1}$, in hindsight, for all n, before we even start predicting $\{x[t]\}_{t\geq 1}$. Hence, the algorithm of corollary asymptotically achieves the performance of the best piecewise affine predictor that can choose both its partitions as well as the prediction coefficients for that partition based on $\{x[t]\}_{t\geq 1}$ and $\{Y[t]\}_{t\geq 1}$ in hindsight.

Remark 2: We emphasize that the best piecewise model with the optimal weights tuned using all $\{x[t]\}_{t\geq 1}$ corresponds to the finest partition, i.e., the 5th partition shown in Fig. 3.1. Hence, at first sight, one is tempted to use the sequential predictor corresponding to the finest partition, i.e., $\hat{X}_{\Omega_5}[t]$ in Fig. 3.1, with the sequential algorithms from (3.3). However, note that this sequential predictor needs to learn the corresponding optimal weights in each region sequentially, hence, it may not be the best sequential algorithm as shown in the simulations section. Furthermore the bound in (3.4) holds for all partitions and the regret of our algorithm with respect to the best piecewise affine model corresponding to the finest partition is the largest, however, still $o(n)$.

Outline of the Proofs of the Theorem and the Corollary: For each sequential predictor corresponding to a partition, we define a function of its prediction loss on $\{Y[t]\}_{t\geq 1}$ as $F_{\Omega_k}[n] \overset{\triangle}{=} \exp\left(-a\sum_{t=1}^{n}(Y[t]-\hat{X}_{\Omega_k}[t])^2\right)$, $a \in \mathrm{R}^+$, at each time n, that only depends on noisy observations. We further define a weighted sum of these functions

$$F[n] = \sum_{k=1}^{m} 2^{-C(\Omega_k)} F_{\Omega_k}[n], \tag{3.5}$$

where $C(\Omega_k)$ are certain weights introduced in [41] satisfying $\sum_{k=1}^{m} 2^{-C(\Omega_k)} = 1$ and $C(\Omega_k) \leq 2K_k - 1 \leq 2\ 2^D - 1$. The weights $C(\Omega_k)$ are introduced for proof purposes and are not explicitly used in the final algorithm. Clearly, $F[n]$ is as large as any $2^{-C(\Omega_k)}F_{\Omega_k}[n]$. Our initial goal is to show that for some randomized predictor, say $\hat{X}[t]$, $F_{\hat{X}}[n]$ is as large as $F[n]$, i.e., the performance of $\hat{X}[t]$ for predicting $\{Y[t]\}_{t\geq 1}$ is as good as any $\hat{X}_{\Omega_k}[t]$. We will then use this predictor for prediction of $\{x[t]\}_{t\geq 1}$. To accomplish the first step, we observe that $F[n] = \prod_{t=1}^{n}(F[t]/F[t-1])$ by telescoping ($F[0] = 1$). However for each term in this product, we have

$$\begin{aligned} \frac{F[t]}{F[t-1]} &= \sum_{k=1}^{m} \frac{2^{-C(\Omega_k)}F_{\Omega_k}[t-1]}{F[t-1]} \exp\left(-a(Y[t]-\hat{X}_{\Omega_k}[t])^2\right) \\ &\leq \exp\left\{-aE\left[(Y[t]-\hat{X}_{\Omega_k}[t])^2\right] + \frac{a^2}{8}\right\}, \end{aligned} \tag{3.6}$$

where the expectation in the last inequality is with respect to the probabilities $\frac{2^{-C(\Omega_k)}F_{\Omega_k}[t-1]}{F[t-1]}$ and the inequality follows from Hoeffding's inequality [42]. Hence, if we construct a randomized predictor, say $\hat{X}[t]$, that outputs $\hat{X}_{\Omega_k}[t]$ as its prediction with probability $\frac{2^{-C(\Omega_k)}F_{\Omega_k}[t-1]}{F[t-1]}$ at each time t, then by (3.6), the accumulated loss of this algorithm will satisfy

$$\begin{aligned} \ln F_{\Omega_k}[n] - C(\Omega_k)\ln 2 &= -a\sum_{t=1}^{n}(Y[t]-\hat{X}_{\Omega_k}[t])^2 - C(\Omega_k)\ln 2 \\ &\leq \ln F[n] \\ &\leq -a\sum_{t=1}^{n} E[(Y[t]-\hat{X}[t])^2] + \frac{na^2}{8} \end{aligned} \tag{3.7}$$

for all k. Since $C(\Omega_k) < 2^{D+1}$, setting $a = \sqrt{8\ 2^{D+1}\ln 2/n}$, yields an upper bound $\sqrt{n2^{D+1}\ln 2/8}$ on the accumulated loss of $\hat{X}[t]$. This upper bound yields the upper bound in the theorem after normalization with n. Hence the randomized predictor $\hat{X}[t]$ is the desired predictor if the goal was to predict $\{Y[t]\}_{t\geq 1}$. However, even in this case, this randomized predictor $\hat{X}[t]$, at each time t, needs to calculate and update a doubly exponential number, i.e., m, of predictions, which is clearly an impossible feat even for modest m. However, note that, in $\hat{X}[t]$, at each time t, only $D+1$ node predictions $\hat{X}_\rho[t]$ that $Y[t-1]$ belongs to are used such that all the weights with same node predictions can be merged. It can be shown as in [15] that if one defines certain functions of performance for each node as $F_\rho[t]$, $\Gamma_\rho[t]$ which are initialized in (line A) and updated in (line C), (line D) and (line E) of Fig. 3.2, then the corresponding $F[t]$ can be written as

$$F[t] = \sum_{i=1}^{D+1} \boldsymbol{z}(i) \exp\left(-a \sum_{j=1}^{t} \left[(Y[j] - \hat{X}_{\boldsymbol{v}(i)}[j])^2 s_{\boldsymbol{v}(i)}[j]\right]\right), \tag{3.8}$$

where $\boldsymbol{v}$ is the vector of nodes that $Y[t-1]$ belongs to, the recursion for $\boldsymbol{z}$ is given in (line B) of Fig. 3.2 and $\boldsymbol{v}(i)$ is the ith entry of the vector $\boldsymbol{v}$. Hence $\hat{X}[t]$ is defined as the randomized predictor using probabilities

$$\boldsymbol{\eta}(i) \triangleq \frac{\sum_{i=1}^{D+1} \boldsymbol{z}(i) \exp\left(-c \sum_{j=1}^{t-1} \left[(Y[j] - \hat{X}_{\boldsymbol{v}(i)}[j])^2 s_{\boldsymbol{v}(i)}[j]\right]\right)}{F[t-1]}. \tag{3.9}$$

Note that all these performance bounds are with respect to the prediction of $\{Y[t]\}_{t\geq 1}$ not with respect to the prediction of the desired signal $\{x[t]\}_{t\geq 1}$. However, we observe that

$$\begin{aligned} E[(Y[t] - \hat{X}_{\Omega_k}[t])^2] &= E[(x[t] + N[t] - \hat{X}_{\Omega_k}[t])^2] \\ &= E[(x[t] - \hat{X}_{\Omega_k}[t])^2] + \sigma_n^2 \end{aligned} \tag{3.10}$$

and

$$\begin{aligned} E[(Y[t] - \hat{X}[t])^2] &= E[(x[t] + N[t] - \hat{X}[t])^2] \\ &= E[(x[t] - \hat{X}[t])^2] + \sigma_n^2, \end{aligned} \tag{3.11}$$

since $N[t]$ is i.i.d. and independent from both $\hat{X}_{\Omega_k}[t]$ and $\hat{X}[t]$. Note that $\hat{X}_{\Omega_k}[t]$ and $\hat{X}[t]$ are sequential and do not use $Y[t]$. Hence, using these equations yields the result in (3.2). This concludes the proof of the theorem. $\square$

To get the corollary, it has been shown in [25] that the affine predictor $\hat{X}_\rho[t]$ in (3.3) achieves

$$E\left[\sum_{t=1}^{n}(x[t] - \hat{X}_\rho[t])^2 s_\rho[t] - \sum_{t=1}^{n}(x[t] - wY[t-1] - c)^2 s_\rho[t]\right] \leq 2\ln(n_\rho) + O(1) \tag{3.12}$$

for all $w, c \in \mathbb{R}$ and n, where n_ρ is the number of times node ρ is used in prediction, i.e., $Y[t-1] \in \Delta_\rho$. Applying this result to any $\hat{X}_{\Omega_k}[t]$ that uses these affine predictors in each node yields,

$$E\left[\sum_{t=1}^{n}(x[t] - \hat{X}_{\Omega_k}[t])^2 - \sum_{t=1}^{n}(x[t] - wY[t-1] - c)^2\right] \leq 2\ln(n) + O(1) \qquad (3.13)$$

after maximization over n_ρ's. Combining this bound with (3.2) and selecting an appropriate value for a yields the result in the corollary. This completes the outline of the proof of the corollary. $\square$

3.3 Simulation Results

In this section, we illustrate the performance of the introduced algorithm, CRNNP, when it is used to predict noise-corrupted chaotic signals generated using the Duffing map described by $x[t] = c_1 x[t-2] + c_2 x[t-1] - (x[t-1])^3$. The Duffing map demonstrates chaotic behavior when $c_1 = -0.2$ and $c_2 = 2.75$. For these values of c_1 and c_2, we plot in Fig 3.3, a sample function generated using the Duffing map. Note that although the sample function is completely predictable from the governing dynamical equations using only the last two samples, it exhibits rather erratic behavior, and is in fact known to exhibit chaotic behavior for this set of coefficients. We also plot the corresponding attractor for the Duffing map in Fig. 3.4 showing its highly nonlinear nature. The desired signal $x[t]$ is then corrupted by an additive noise $N[t]$ with standard deviation 0.05. In Fig 3.5, we plot the accumulated and normalized MSE of different algorithms averaged over 200 random iterations of $\{x[t]\}_{t\geq 1}$ and $\{N[t]\}_{t\geq 1}$. In this figure, the context-tree based algorithms use a context-tree of depth-6, $a = 1$ and first order sequential linear predictors in each node. In Fig. 3.5, we have the CRNNP algorithm from Fig. 3.2 "alg"; the sequential algorithm corresponding to the finest partition (discussed in Remark 2) on this tree "finest"; the context-tree algorithm that trains on the clean signal $\{x[t]\}_{t\geq 1}$, however, still uses $Y[t-1]$ as the regressor "clean alg". Since at each time the introduced algorithm requires $O(7)$ computations, we also simulate a 7th order linear least squares algorithm using $\{Y[t-1], \ldots, Y[t-7]\}$ as its input regressor [4] "RLS". Note that this RLS algorithm provides significantly worse performance since it tries to approximate the nonlinear terms in the Duffing map by linear combinations. In order to model the nonlinear terms in the Duffing Map, we also implement a 4th order linear

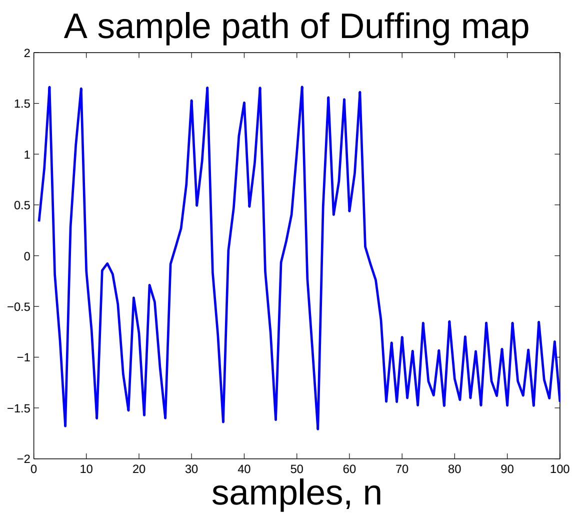


Figure 3.3: A sample function of the Duffing map

least squares algorithm using $\{Y[t-1],(Y[t-1])^2,\ldots,(Y[t-1])^4\}$ as the input regressor "NRLS". We observe in these simulations that the "alg" algorithm is able to outperform the "finest" algorithm in the initial samples since the finest partition needs to learn all the affine models used in the leaves. CRNNP is able to exploit the smaller sub-trees (or coarser models) which have less parameters to train, hence it provides better performance in the start of the simulations against all algorithms. As the data length grows, when the "”finest" algorithm has enough data to train on, both algorithms provide similar performance. The performance of CRNNP trained on noisy samples is nearly the same as the performance of the "clean alg" algorithm, i.e., the curves are nearly the same. This result was expected as shown in the proof of the introduced algorithm. For these chaotic signals, the introduced CRNNP algorithm outperforms all other algorithms that also use only the noise-corrupted samples.

3.4 Conclusion

In this chapter, we introduced a novel randomized sequential prediction algorithm (*CRNNP*) that only uses noise-corrupted past samples of a deterministic desired signal to predict the clean desired signal. This algorithm is shown to achieve asymptotically the performance of the best piecewise affine model that can both select the best partition of the space of past regressor (from a doubly exponential number of possible partitions) and the affine model parameters based on the clean desired signal. We demonstrated the performance of the introduced CRNNP algorithm when it is used to predict chaotic signals generated using the Duffing map.

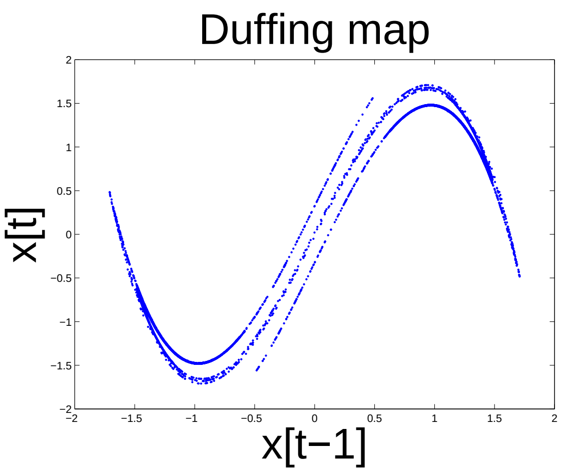


Figure 3.4: The attractor of the Duffing map

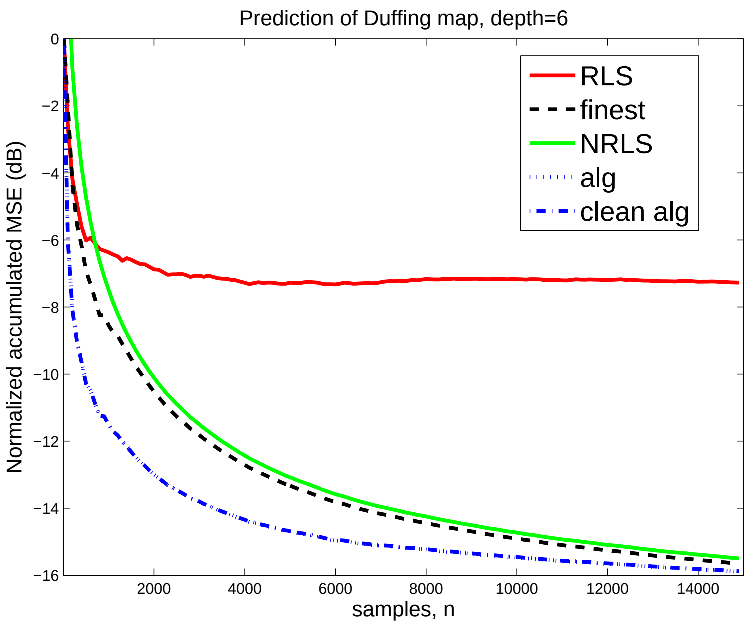


Figure 3.5: Normalized MSE for the algorithms described in the text

Chapter 4

CONCLUSIONS

In this thesis, we handle linear adaptive filtering and nonlinear adaptive prediction tasks. Novel adaptive algorithms for these tasks are presented throughout the sections of this thesis. Chapter 2 deals with the linear adaptive filtering task and proposes two new algorithms. Chapter 3 is dedicated to the nonlinear adaptive prediction task.

Proportionate update idea of [1] leads us to develop the *Proportionate Normalized Least Mean Fourth (PNLMF)* and *Krylov-Proportionate Normalized Least Mean Fourth (KPNLMF)* algorithms. PNLMF is built up in the same way [2] produces LMF from LMS. We implement Krylov subspace projection technique to develop the KPNLMF algorithm. KPNLMF is similar to KPNLMS in [3] except that our KPNLMF algorithm minimizes the mean fourth error. KPNLMF exploits the fact that projecting an unknown potentially dispersive system to a Krylov subspace yields a sparse system. This fact is proven in Subsection 2.2.3 of Chapter 2. Obtaining a sparse system whatever the input system is, enables us to employ the PNLMF algorithm which is derived previously.

The KPNLMF algorithm is investigated for the system identification problem and simulation results are provided in the Section 2.4 of the Chapter 2. Performance of KPNLMF is compared to the performances of NLMF and KPNLMS. It is shown that KPNLMF converges faster than NLMF both in mean-square-error sense and system mismatch, i.e. weight error power, sense. Faster convergence of KPNLMF when compared to NLMF is due to the Krylov subspace projection technique that is involved in the KPNLMF algorithm. KPNLMF is also shown to have superior performance than KPNLMS in certain practical cases where the noise probability distribution is one of the probability density functions (pdf) among uniform distribution, sine wave distribution and square wave distribution. KPNLMF has a system mismatch value that is much smaller than that of KPNLMS in these cases when they both converge at the same speed. However, KPNLMS performs better than KPNLMF if the noise has a Gaussian pdf.

Another contribution of this thesis is the steady-state Mean Square Error (MSE) analysis that are carried out for KPNLMS and KPNLMF. They are both shown to converge to the desired solution with sufficiently small excess MSEs.

We consider sequential nonlinear prediction of a real-valued, bounded and deterministic signal from its noise-corrupted past samples in Chapter 3. We introduce a randomized algorithm, which is called *Competitive Randomized Noisy Nonlinear Predictor (CRNNP)*, based on context-trees in a competitive framework. The CRNNP algorithm works in a framework that is combination of four different frameworks. The main contribution of CRNNP is that it produces high prediction performance under additive noise, so it includes a noisy framework. Secondly, since the CRNNP algorithm is for all bounded, real-valued and deterministic signals, we define a competitive framework to measure its quality of performance. We incorporate a competitive framework because we do not make any stochastic assumptions over the input signals. Thirdly, CRNNP involves a nonlinear framework which is based on context-trees. Context-trees are used to build piecewise affine predictors to model the nonlinearity. And lastly, our algorithm uses a randomized selection framework. The CRNNP algorithm needs this randomized selection framework as it serves for the kind of problems that need to choose a strategy from several number of strategies at each time. Those kind of problems are named sequential decision problems and we follow a randomized way to deal with such kind of problems. Randomization weights are used to make a decision at each time.

The CRNNP algorithm performs in the combination of these four frameworks and it is shown to beat all other algorithms that perform under the same conditions. CRNNP has a performance that is nearly the same as the one of the algorithm working on the clean signal. This algorithm working on the clean signal assumes the desired clean input signal is available, i.e. assumes no additive noise in the channel. Our algorithm achieves nearly the same performance without any increase in the computational complexity. The computational complexity of CRNNP is linear in the depth of the context-tree used per prediction. The proposed CRNNP algorithm is tested with chaotic signals that are created via Duffing map in Section 3.3 of Chapter 3.

It is also proven in Section 3.2 of Chapter 3 that without ever observing the desired clean signal, the CRNNP algorithm achieves the performance of the best piecewise affine predictor that can both choose its partition of the past observations space (from a class of a doubly exponential number of possible partitions) and tune the parameters of the affine models for these piecewise regions using the clean desired signal.

4.1 Future Work

Some possible directions for the future research in the fields that are studied in this thesis are pointed here.

- Direct usage of Krylov subspace techniques in linear adaptive filtering can be investigated. Both the KPNLMF algorithm in this thesis and the KPNLMS algorithm in [3] exploit the Krylov subspace projection technique in order to make the unknown input system sparse. However, if we can proceed in such a direct way that a Krylov subspace technique is used to solve the Wiener-Hopf equation, then a much faster converging linear adaptive filtering algorithm can be obtained. The way the LMS and the LMF algorithms employ to converge to the optimal solution are named *Richardson iteration* in numerical methods literature. There are mainly two groups of iterative methods for solving a system of linear equations and Richardson iteration is member of the stationary iterative methods group. The other group for solving a system of linear equations is the Krylov subspace methods. The proportionate update idea for the sparse systems that is presented first time in [1] is an improvement on the Richardson iteration. KPNLMS and KPNLMF extend this improvement to the non-sparse systems, but they are still members of the stationary iterative methods group. Developing an efficient algorithm which is a member of the Krylov subspace methods for the solution of the Wiener-Hopf equation has a great chance to outperform the present linear adaptive filtering algorithms.

- Exploiting context-graphs instead of context-trees to model the nonlinearity may improve the prediction performance of the CRNNP algorithm. Context-graphs are a generalization of context-trees and the usage of them for the universal prediction task is demonstrated in [43]. Context-graphs mitigate the limitations of the context-trees on the modeling power of the class of models by allowing a more general graphical structure. If we use a context-graph instead of a context-tree to model the nonlinearity that is present at the problem, we may end up with an algorithm that is more powerful in terms of modeling than CRNNP introduced in this thesis.

BIBLIOGRAPHY

[1] D. L.Duttweiler, "Proportionate normalized least-mean-squares adaptation in echo cancelers," *IEEE Trans. Speech Audio Process.*, vol. 8, no. 5, pp. 508–518, Sept. 2000.

[2] E. Walach and B. Widrow, "The least mean fourth (lmf) adaptive algorithm and its family," *IEEE Trans. Inf. Theory*, vol. 30, no. 2, Mar. 1984.

[3] M. Yukawa, "Krylov-proportionate adaptive filtering techniques not limited to sparse systems," *IEEE Trans. Signal Process.*, vol. 57, no. 3, Mar. 2009.

[4] Simon Haykin, *Adaptive Filter Theory, 3rd Edition*, Prentice Hall, Upper Saddle River, New Jersey, 1996.

[5] Thomas Kailath, Ali H. Sayed, and Babak Hassibi, *Linear Estimation*, Prentice Hall, Upper Saddle River, New Jersey, 2000.

[6] Babak Hassibi, Ali H. Sayed, and Thomas Kailath, "h^{∞} optimality of the lms algorithm," *IEEE Transactions on Signal Processing*, vol. 44, no. 2, pp. 267–280, 1996.

[7] A. H. Sayed and T. Kailath, "A state-space approach to adaptive rls filtering," *IEEE Signal Processing Magazine*, vol. 11, no. 3, pp. 18–60, 1994.

[8] J. Makhoul, "Linear prediction: A tutorial review," *Proc. IEEE*, vol. 63, pp. 561–580, April 1975.

[9] O.J.J. Michel, A.O.III Hero, and A.E. Badel, "Tree-structured nonlinear signal modeling and prediction," *IEEE Transactions on Signal Processing*, vol. 47, no. 11, pp. 3027–3041, 1999.

[10] A.C. Singer, G.W. Wornell, and A.V. Oppenheim, "Nonlinear autoregressive modeling and estimation in the presence of noise," *Digital Signal Processing*, vol. 4, no. 4, pp. 207–221, October 1994.

[11] Martin Schetzen, *The Volterra and Wiener Theories of Nonlinear Systems*, John Wiley, New York, 1980.

[12] P.J.W. Rayner and M.R. Lynch, "A new connectionist model based on a non-linear adaptive filter," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-89)*, 1989, pp. 1191–1194.

[13] M.R. Lynch and P.J.W. Rayner, "The properties and implementation of the nonlinear vector space connectionist model," in *First IEEE International Conference on Artificial Neural Networks*, 1989, pp. 186–190.

[14] A.R. Sadeghian, "Nonlinear neuro-fuzzy prediction: methodology, design and applications," in *The 10th IEEE International Conference on Fuzzy Systems*, 2001, pp. 1022–1026.

[15] S.S. Kozat, A.C. Singer, and G. Zeitler, "Universal piecewise linear prediction via context trees," *IEEE Transactions on Signal Processing*, vol. 55, no. 7, pp. 3730–3745, 2007.

[16] R. Zamir and M. Feder, "On universal quantization by randomized uniform/lattice quantizers," *IEEE Transactions on Information Theory*, vol. 38, no. 2:1, pp. 428–36, March 1992.

[17] T.M. Cover, "Universal gambling schemes and the complexity measures of kolmogorov and chaitin," *Tech. Rep. 12, Dept. Statist., Stanford Univ., Stanford, CA*, 1974.

[18] J. Rissanen, "Universal coding, information, prediction, and estimation," *IEEE Trans. Info. Theory*, vol. IT-30, pp. 629–636, 1984.

[19] J. Benesty and S.L. Gay, "An improved pnlms algorithm," in *Proc. IEEEInt. Conf. on Acoustic, Speech, Signal Process (ICASSP)*, 2002, pp. 1881–1884.

[20] Y. Yilmaz and S.S. Kozat, "An extended version of the nlmf algorithm based on proportionate krylov subspace projections," in *IEEE International Conf. on Machine Learning and Applications (ICMLA)*, 2009, pp. 404–408.

[21] A. Zerguine, "Convergence behavior of the normalized least mean fourth algorithm," in *Thirty-Fourth Asilomar Conference on Signals, Systems and Computers*, 2000.

[22] Yousef Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, 2003.

[23] Youcef Saad, *Numerical Methods for Large Eigenvalue Problems*, Manchester University Press, 1992.

[24] Ali H. Sayed, *Fundamentals of Adaptive Filtering*, Wiley-IEEE Press, Wiley, NJ, 2003.

[25] S. S. Kozat and A. C. Singer, "Competitive prediction under additive noise," *IEEE Transactions on Signal Processing*, vol. 57, September 2009.

[26] N. Bianchi and G. Lugosi, *Prediction, Learning and Games*, Cambridge University Press, 2006.

[27] T. Weissman and N. Merhav, "Universal prediction of individual binary sequences in the presense of noise," *IEEE Trans. Info. Theory*, vol. 47, no. 6, pp. 2151–2173, 2001.

[28] T. Moon and T. Weissman, "Competitive online linear fir mmse filtering," in *Proceedings of ISIT*, 2007, pp. 1126–1130.

[29] A.C. Singer and M. Feder, "Universal linear prediction by model order weighting," *IEEE Transactions on Signal Processing*, pp. 2685–2699, October 1999.

[30] S. S Kozat and A. C. Singer, "Universal switching linear least squares prediction," *IEEE Transactions on Signal Processing*, vol. 56, pp. 189–204, Jan. 2008.

[31] Y. Yilmaz and S.S. Kozat, "Competitive randomized nonlinear prediction under additive noise," *IEEE Signal Processing Letters*, vol. 17, no. 4, pp. 335–339, 2010.

[32] V. Vovk, "Competitive on-line linear regression," *Advances in Neural Information Processing Systems*, pp. 364–370, 1998.

[33] N. Cesa-Bianchi, Y. Freund, D. Helmbold, D. Haussler, R. Schapire, and M. Warmuth, "How to use expert advice," *Annual ACM Symposium on Theory of Computing*, pp. 382–391, 1993.

[34] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," *Information and Computation*, vol. 108, no. 2, pp. 212–261, 1994.

[35] T. Cover and E. Ordentlich, "Universal portfolios with side-information," *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 348–363, 1996.

[36] T. Linder and G. Lugosi, "A zero-delay sequential scheme for lossy coding of individual sequences," *IEEE Transactions on Information Theory*, vol. 46, no. 6, pp. 190–207, 2001.

[37] J. Hannan, *Approximation to Bayes risk in repeated play*, vol. III of *Contributions to the Theory of Games*, pp. 97–139, Princeton, NJ, 1957.

[38] A. C. Singer, S. S. Kozat, and M. Feder, "Universal linear least squares prediction: upper and lower bounds," *IEEE Trans. Information Theory*, vol. 48, no. 8, pp. 2354–2362, Aug 2002.

[39] A. Gyorgy, T. Linder, and G. Lugosi, "Tracking the best quantizer," *IEEE Trans. Info. Theo.*, vol. 54, pp. 1604–1625, April, 2008.

[40] N. Merhav, "On the minimum description length principle for sources with piecewise constant parameters," *IEEE Transactions on Information Theory*, vol. 41, pp. 1962–1967, 1993.

[41] F. Willems, Y. Shtarkov, and T. Tjalkens, "The context-tree weighting method: basic properties," *IEEE Trans. Info. Theory*, vol. IT-41, pp. 653–664, May 1995.

[42] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of Amer. Statist. Assoc*, vol. 58, pp. 13–30, 1963.

[43] R. J. Drost and A. C. Singer, "Constrained complexity generalized context-tree algorithms," in *IEEE/SP 14th Workshop on Statistical Signal Processing*, 2007, pp. 131–135.