

**DIGITAL PREDISTORTION OF NONLINEAR POWER AMPLIFIERS
USING ADAPTIVE FILTERING AND PARTICLE
SWARM OPTIMIZATION**

BY

Abubaker Hassan Babiker Abdelhafiz

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

ELECTRICAL ENGINEERING

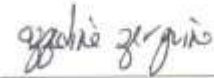
May 2013

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN- 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **ABUBAKER HASSAN ABDELHAFIZ** under the direction of his thesis advisor and approved by his thesis committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN ELECTRICAL ENGINEERING**.



Prof. Azzedine Zerguine
(Advisor)



Dr. Ali H. Al-Shaikhi
Department Chairman



Dr. Oualid Hammi
(Co-Advisor)

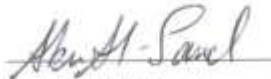


Dr. Salam A. Zummo
Dean of Graduate Studies



Dr. Adnan Landalousi
(Member)

5/5/13
Date



Dr. Wajih Abul-Saud
(Member)



Dr. Ali Al-Awami
(Member)

© Abubaker Hassan Abdelhafiz

2013

DEDICATION

To my family and friends, whose amusing interruptions have made the past few years
incredibly entertaining.

ACKNOWLEDGMENTS

First and foremost, I would like to thank Allah almighty for guiding me towards the correct paths in life, despite my talent for making the worst possible decisions at every turn. Truly, for one to make as many bad decisions as I have and yet somehow end up not failing miserably is a sign of God's infinite grace.

Next, I would like to offer my deepest gratitude to my family for being the most amusing collection of misfits one can find under one roof. Many thanks to them for being as understanding of my absence and hermit-like tendencies as they (hopefully) were and for putting up with me, generally-unpleasant and flippant person that I am.

Many thanks are due to my advisors, Prof. Azzedine Zerguine and Dr. Oualid Hammi, a veritable Dynamic Duo (who are every bit as amazing and entertaining as the referenced pair of avian-themed characters) who have exhibited superhuman levels of patience worthy of being immortalized in song and poem for decades to come. The mere fact that neither of them have yet to smack me upside the head despite my countless failings says it all, not to mention the incredible effort they put into guiding me throughout this work and arranging a research trip at the University of Calgary for a buffoon as bumbling as myself. Throughout it all, it was Dr.Zerguine's unflinching optimism and Dr.Oualid's infectious enthusiasm that always motivated me to do my best and march on.

Following that, I would like to thank all my friends for standing by my side and provide much-needed laughs, guidance and support. In particular, no amount of flowery prose on my part would suffice to thank Mr.Eyad Al-Sibai; KFUPM/KAUST alumnus, award-winning prodigy, Data Mining wizard at Aramco and connoisseur of all things food-

related for being such an amazing friend for the past 5 years (and counting!) and for succeeding at the Herculean task of retaining his sanity despite hanging around with my rambunctious self for half a decade. Another fine specimen of humanity who has earned his place in this section is my Mr. Tayyab Mujahid, for introducing me to the wonders of Hyderabadi cuisine and for his failed attempts at humor (which never fail to entertain nonetheless) , among his many other endearing qualities. Mr. Mujahid has been extremely supportive and always ready to lend an ear (or two) at a moment's notice.

No acknowledgement section would be complete without thanking KFUPM for all it has done for me. While this might come off as a bit overwrought (perhaps rightfully so), I cannot stress the impact KFUPM has had on my life and the role it had in changing it for the better. For someone such as I, enrollment at KFUPM was -without exaggeration- a life-changing event that allowed me to once again stand on my feet, find my bearing in life and take my life back into my hands. During the three years since my enrollment, I have gained an education, wonderful friendships, the means of personal enablement, countless opportunities, the chance to (very literally) travel the corners of the earth and a state of physical health I could only dream of for most of my life.

Among the many faculty and members of the KFUPM family whom I'd like to thank, special mention goes out to Dr. Maan Kousa, for his wonderful guidance and advice, Mr. Umar Johar for being so tolerant and helpful as he always was, Professor Asrar Sheikh for his enlightening and most entertaining lectures, Dr. Ali Al-Shaikhi for being so supportive in countless ways.

Special thanks go to Professor Fadhel Ghannouchi for inviting me for a research visit to the pioneering Intelligent Radio (iRadio) lab at the University of Calgary, Alberta, Canada and for generously hosting me for the duration of my visit. Thanks also go out to the many colorful members of the iRadio family who made this trip every bit as entertaining as it was educational, especially Mr. Andrew Kwan (resident lab wizard, nerd supreme and ace clueless-intern sitter) for putting up with my patently-silly questions and always being ready to lend a hand at a moment's notice. In every sense of the word, Mr. Kwan is what every engineer and researcher should strive to become: talented, patient, humble, hard-working and armed with unparalleled technical skill.

Another individual who has been an integral part of my story at KFUPM is Dr. Ahmad Masoud. Despite having neither taught me a course or supervised me during my research, my (many and rather lengthy) conversations with Dr. Masoud have had great influence on my personal and intellectual development over the years. Dr. Masoud's unique perspectives on life and his interesting take on current affairs have been as enlightening as they were entertaining.

In closing, it is only appropriate to thank He who is both Alpha and Omega; Allah Almighty, for bestowing upon me the gift of life and endowing me with the ability to experience as much of it as I have. In times of peace and those of turmoil, in day or in night, for richer or for poorer, it is, was and will always be He who illuminates my life and strengthens my resolve and it is He whom I could never thank enough.

Doesn't mean I can't try, though.

At the end, there remains one thing that must be said. From the bottom of my heart and with all the emotion my soul can muster, Alhamdullellah.

Abubaker Hassan Abdelhafiz

KFUPMer for life,

Tuesday, April 30th

, 2013

TABLE OF CONTENTS

DEDICATION.....	IV
ACKNOWLEDGMENTS.....	V
TABLE OF CONTENTS.....	IX
LIST OF TABLES.....	XIV
LIST OF FIGURES.....	XV
LIST OF ABBREVIATIONS.....	XVII
ABSTRACT.....	XIX
CHAPTER 1 INTRODUCTION.....	1
1.1 Power Amplifier Nonlinearities.....	2
1.1.1 Memory Effects.....	4
1.1.2 Impedance Variations.....	4
1.1.3 Frequency-Domain Nonlinearities.....	5
1.2 Digital Predistortion.....	6
1.3 Problem Statement and Formulation.....	7
1.4 Thesis Contributions.....	8
1.5 Thesis Outline and Organization.....	9
CHAPTER 2 BEHAVIORAL MODELING OF NONLINEAR POWER AMPLIFIERS....	10
2.1 Introduction to Behavioral Modeling.....	10
2.2 Memoryless Models.....	11
2.2.1 Look-Up-Table Model.....	11
2.2.2 Memoryless Polynomial Model.....	12

2.3	Models with Memory.....	12
2.3.1	Volterra Series Model.....	12
2.3.2	Wiener Model	14
2.3.3	Hammerstein Model.....	15
2.3.4	Wiener-Hammerstein Model.....	16
2.3.5	Memory Polynomial Model.....	17
2.3.6	The Orthogonal Memory Polynomial Model	18
2.3.7	The Two-Box Twin-Nonlinear Model	20
2.4	Choice of Model Used and Challenges Facing Parameter-Estimation	21
2.4.1	Ill-conditioning of the Data Matrix	22
2.4.2	Determination of Model Dimensions.....	26
2.5	Metrics Used for the Evaluation of Behavioral Models and Predistorters	29
2.5.1	Time-Domain Metrics	29
2.5.2	Frequency-Domain Metrics	30
2.6	Identification of Behavioral Model Parameters.....	31
2.7	Conclusions	31
CHAPTER 3 ADAPTIVE IDENTIFICATION OF NONLINEAR POWER AMPLIFIERS		
32		
3.1	General Definitions	33
3.2	Stochastic Gradient-Based Algorithms and their Variants	35
3.2.1	The Least Mean-Square (LMS) Algorithm	36
3.2.2	The Normalized Least Mean-Square (NLMS) Algorithm	36
3.2.3	The Sign-LMS Algorithm	37
3.2.4	The Leaky-LMS Algorithm.....	38
3.2.5	The Least Mean-Fourth (LMF) Algorithm	39

3.2.6	The Normalized Least Mean-Fourth (NLMF) Algorithm	40
3.2.7	The Least-Mean Mixed-Norm (LMMN) Algorithm	40
3.2.8	The Affine Projection Algorithm (APA)	41
3.3	The Least-Squares (LS) Family	42
3.3.1	The Recursive Least-Squares Algorithm (RLS)	43
3.3.2	The QR Decomposition-based Recursive Least-Squares Algorithm (QR-RLS)	44
3.4	Shortcomings of the Available Adaptive Filtering Algorithms and Some Solutions.....	46
3.4.1	Pre-processing Using Normalization and Data-centering	46
3.4.2	Use of Whitening Lattice	47
3.5	Performance Metrics for Adaptive Filters	51
3.5.1	Normalized Mean-Square Error (NMSE)	51
3.5.2	Convergence Speed	52
3.5.3	Computational Complexity	52
3.6	Comparative Study of Adaptive Identification of Nonlinear Power Amplifier Parameters	53
3.7	Results and Conclusions	59
 CHAPTER 4 DIGITAL PREDISTORTION USING PARTICLE SWARM OPTIMIZATION.....		61
4.1	Basic Structure of the PSO Algorithm	62
4.2	Some of the PSO variants in the literature	67
4.2.1	Constriction-Factor PSO.....	67
4.3	Shortcomings of available PSO techniques.....	71
4.4	Proposed PSO techniques	73
4.4.1	Cluster-based PSO (C-PSO)	73
4.4.2	l_0 -Penalized PSO	77
4.4.3	l_1 -Penalized PSO	83

4.5	Experimental Validation of Proposed Algorithms.....	84
4.6	Sensitivity analysis of the PSO algorithms.....	85
4.6.1	Sensitivity analysis for the l_0 -VCFPSO algorithm.....	86
4.6.2	Sensitivity analysis for the l_1 -VCFPSO algorithm.....	90
4.7	Simulation Results.....	92
4.7.1	Results for experiment A: Correctly-Sized Model.....	92
4.7.2	Results for experiment B : Oversized model with $L = 3, K = 8$	96
4.7.3	Results for experiment C : Oversized model with $L = 5, K = 8$	97
4.8	Summary of Results and Conclusions Reached.....	102
CHAPTER 5 THESIS CONCLUSIONS AND FUTURE WORK.....		104
5.1	Summary of Work Done and Conclusions.....	104
5.2	Future Work.....	104
APPENDIX A POWER AMPLIFIER BASICS.....		106
6.1	Classes and Types of Power Amplifiers.....	106
6.1.1	Class A Amplifiers.....	107
6.1.2	Class B Amplifiers.....	107
6.1.3	Class AB Amplifiers.....	108
6.1.4	Class C Amplifiers.....	109
6.1.5	Class D Amplifiers.....	109
6.1.6	The Doherty Amplifier.....	110
APPENDIX B SINGULAR VALUE DECOMPOSITION AND THE METHOD OF LEAST SQUARES.....		111
7.1	Introduction.....	111
7.2	Application to System Identification.....	111

7.3	Limitations of Using SVD	112
	REFERENCES.....	113
	VITAE.....	122

LIST OF TABLES

Table 2.1 Comparison of time required for the generation of MPM and OMPM models (in seconds)	20
Table 3.1 Effect of using the lattice with the MP model	50
Table 3.2 Effect of the number of stages on post-MPM lattice performance.....	50
Table 3.3 Computational cost of the various adaptive algorithms tested	53
Table 3.4 Comparison of the performance of the various adaptive identification algorithms	58
Table 4.1 Numerical results comparing adaptive algorithms and PSO	71
Table 4.2 Parameter Selection for the traditional PSO Algorithms.....	86
Table 4.3 Parameter Selection for the l_0 -VCFPSO Algorithm	86
Table 4.4 Parameter Selection for the l_1 -VCFPSO Algorithm.....	91
Table 4.5 Numerical results for the first identification experiment.....	95
Table 4.6 Numerical results for the second identification experiment	96
Table 4.7 Numerical results for the third identification experiment.....	100

LIST OF FIGURES

Figure 1.1	Typical OFDM signal displaying high PAPR	3
Figure 1.2	Example of Nonlinear amplifier characteristics[13]	4
Figure 1.3	Example of distortion effects in the frequency domain.....	5
Figure 1.4	Block diagram of the digital predistortion process using the Indirect Learning Architecture[18].....	6
Figure 1.5	Distortion-mitigation using DPD	7
Figure 1.6	Thesis flow	8
Figure 2.1	Sample diagram representing a third-order Volterra series model.....	13
Figure 2.2	Wiener Model.....	15
Figure 2.3	Wiener-Hammerstein Model.....	16
Figure 2.4	Block Diagram of the various TNTB arrangements [45]: a) Forward. b)Reverse. c)Parallel.....	21
Figure 2.5	Relationship between MPM dimensions and condition number.....	26
Figure 2.6	Structures of correctly- and over-sized MP model dimensions: (a) Correct dimensions. (b) Oversized K . (c) Oversized L . (c) Both parameters oversized.....	28
Figure 3.1	Block Diagram of an adaptive PA-identification setup.....	33
Figure 3.2.	Structure of the lattice filter.....	47
Figure 3.3.	Structure of a single stage of the lattice.....	48
Figure 3.4.	Possible arrangements for implementing the lattice whitener (a)pre-MPM (b)post-MPM.....	50
Figure 3.5	AM/AM Characteristics of the DUT	54
Figure 3.6	AM/PM Characteristics of the DUT.....	55
Figure 3.7	Convergence behavior of the different adaptive algorithms when estimating the parameters of an MPM-based predistorter.....	56
Figure 3.8	Predistortion performance of the various adaptive algorithms and SVD.....	59
Figure 4.1	Block Diagram illustrating the flow of the PSO algorithm.....	66
Figure 4.2	Comparison of PSO algorithms.....	69
Figure 4.3	DPD performance of RLS and PSO algorithms	70
Figure 4.4	Actual and PSO-estimated coefficients of an oversized model.....	72
Figure 4.5	Flow of the cluster-based PSO algorithm.....	75
Figure 4.6	Flow of the l_0 -PSO algorithm	81
Figure 4.7	Actual and l_1 PSO-estimated coefficients of an oversized model.....	84
Figure 4.8	Effect of the swarm size on the performance of l_0 -VCFPSO	87
Figure 4.9	Effect of the choice of the parameters b, c on the performance of l_0 -VCFPSO	88

Figure 4.10 Effect of the choice of the parameters k_{\min}, k_{\max} on the performance of l_0 -VCFPSO	89
Figure 4.11 Effect of the choice of the parameter a on the performance of l_0 -VCFPSO	90
Figure 4.12 Effect of the swarm size on the performance of l_1 -VCFPSO	91
Figure 4.13 Effect of the choice of the parameter a on the performance of l_1 -VCFPSO	92
Figure 4.14 Learning curves for the PSO algorithms when estimating a correctly-sized model.....	93
Figure 4.15 DPD performance of the PSO algorithms	94
Figure 4.16 Coefficients estimated by PSO and l_0 -VCFPSO	97
Figure 4.17 Learning curves for the adaptive algorithms and PSO variants for the oversized model	98
Figure 4.18 DPD performance of the PSO algorithms	99
Figure 4.19 Coefficients estimated by PSO and l_0 -VCFPSO	101
Figure 4.20 Actual and l_1 -VCFPSO-estimated coefficients of an oversized model	102
Figure 6.1 Typical Amplifier Circuit Configuration	106
Figure 6.2 Class A Amplifier Operation[83].	107
Figure 6.3 Class B Amplifier Operation[83].	108
Figure 6.4 Class AB Amplifier Operation[83].	108
Figure 6.5 Class C Amplifier Operation[83].	109
Figure 6.6 Class D Amplifier Operation[83].	109
Figure 6.7 Typical configuration of a Doherty PA[85].	110

LIST OF ABBREVIATIONS

PA	:	Power Amplifier.
DPD	:	Digital Predistortion.
PSO	:	Particle Swarm Optimization.
LUT	:	Look-Up Table.
MP	:	Memory Polynomial.
OMP	:	Orthogonal Memory Polynomial.
TNTB	:	Twin-Nonlinear Two-Box.
LMS	:	Least Mean Squares.
NLMS	:	Normalized Least Mean Squares.
LMF	:	Least Mean Fourth.
LMMN	:	Least Mean Mixed-Norm.
AP	:	Affine Projection.
RLS	:	Recursive Least Squares.
QR-RLS	:	QR-Decomposition Recursive Least Squares.
NMSE	:	Normalized Mean Square Error.

- NAMSE** : Normalized Absolute Mean Spectrum Error.
- VCF-PSO** : Variable Constriction Factor Particle Swarm Optimization.
- l_0 -**VCFPSO** : l_0 -Penalized Particle Swarm Optimization.
- l_1 -**VCFPSO**: l_1 -Penalized Variable Constriction Factor Particle Swarm Optimization.

ABSTRACT

Full Name : Abubaker Hassan Abdelhafiz

Thesis Title : Digital Predistortion of Nonlinear Power Amplifiers Using Adaptive Filtering and Particle Swarm Optimization

Major Field : Electrical Engineering

Date of Degree : May 2013

In this work, the pre-distortion of nonlinear power amplifiers (PAs) through the use of adaptive estimation and Particle Swarm Optimization (PSO) is investigated. After studying the techniques available in the literature and proposing the use of some techniques, it was found that adaptive filtering falls short of the desired performance targets and subsequently, PSO was used. Several variants of PSO were examined and to solve the dual problem of identifying the parameters of a nonlinear PA and estimating the dimensionality of its model, two new PSO algorithms are proposed: l_0 norm-Penalized PSO (l_0 -PSO) and l_1 norm-Penalized PSO (l_1 -PSO) are proposed and studied. The simulations performed indicate that the proposed PSO algorithms produce an estimate of the model's dimension while maintaining comparable performance to the available PSO variants.

Keywords: nonlinear power amplifiers, digital predistortion, memory polynomial mode, twin-nonlinear two-box model, adaptive identification, adaptive filtering, Particle Swarm Optimization (PSO), l_0 -norm, l_1 -norm..

ملخص الرسالة

الاسم الكامل: أبوبكر حسن بابكر عبد الحفيظ

عنوان الرسالة: التشويه الرقمي المسبق لمضخمات القدرة اللاخطية باستخدام الترشيح المتكيف و أمثلة حشود

الجزئيات

التخصص: الهندسة الكهربائية

تاريخ الدرجة العلمية: مايو 2013

في هذا البحث، أجريت دراسة شاملة على مجال التعرف على خواص مضخمات القدرة اللاخطية باستخدام خوارزميات التعرف المتكيفة. بعد دراسة هذه الخوارزميات وتقييم أدائها واقتراح بعض التقنيات لمعالجة القصور في أدائها ، تم التوصل لنتيجة مفادها أن هذه الخوارزميات تعني من قصور في الأداء. من ثم اقتراح استخدام خوارزمية أمثلة سرب الجزئيات وبعد دراسة الخوارزميات المتوفرة، تم استنباط مجموعة خوارزميات جديدة تعتمد على عقوبة مشتقة من معياري الصفر والواحد للتعرف على خواص المضخم وأبعاد النموذج المستخدم لتمثيله في آن واحد. وفقاً لنتائج المحاكاة التي تم إجراؤها، تم التحقق من جودة أداء الخوارزميات المقترحة ودقة تخمينها للأبعاد الصحيحة للنموذج المستخدم مقارنة بخوارزميات حشود الجزئيات المتاحة.

كلمات دلالية: مضخمات القدرة اللاخطية، التشويه الرقمي، نموذج كثير الحدود ذي الذاكرة، النموذج ذو الصندوقين مزدوج اللاخطية، التعرف التكييفي، أمثلة حشود الجزئيات، معيار الصفر، معيار الواحد.

CHAPTER 1

INTRODUCTION

The Power Amplifier (PA) is one of the most commonplace devices encountered in electrical engineering. Whether in audio and speech applications or in the front ends of communication transmitters, power amplifiers can be found in a large number of practical real-world circuits.

An amplifier performs its task of magnifying (scaling) an input signal by a constant factor fairly reliably until they are driven into their saturation regions; after which point their gains become limited, producing saturation and clipping.

Being such integral components of electronic systems, many attempts have been made to understand the behavior of power amplifiers. Many attempts have been made to model and compensate the behavior of such devices and identify their nonlinear characteristics using a variety of approaches to compensate for these nonlinearities using what is known as digital predistortion (DPD) [1], [2].

A DPD is a digitally-implemented inverse function designed to counteract an amplifier's nonlinear behavior, making the design of a DPD essentially an inverse-system identification problem. To design the best possible pre-distorter, reliable and accurate methods for identifying the nonlinear characteristics of a power amplifier are needed since identifying a pre-distorter is conceptually the same as identifying a PA, and it is this identification process with which this work is chiefly concerned.

In this work, the use of adaptive filtering algorithms for this purpose is thoroughly investigated, the shortcomings of this approach clearly identified and subsequently, the use of Particle Swarm Optimization (PSO) is proposed [3]-[7].

The performance of PSO in this context is studied extensively and a novel PSO family of algorithms is developed to solve the dual problem of estimating the coefficients of a nonlinear PA model and the correct dimension of the model, given an oversized estimate of the model's size.

1.1 Power Amplifier Nonlinearities

With the recent developments in high-rate communication systems and the corresponding increase in demand for high-speed services, multi-user systems such as Orthogonal Frequency Division Multiplexing (OFDM) and Long-Term Evolution (LTE) have emerged as attractive standards for modern communication systems. While OFDM and similar systems are attractive choices for their many features such as resilience to noise, they suffer from the well-studied issue of having high Peak-to-Average-Power-Ratios (PAPR) (Figure 1.1) [8],[9], i.e. they have rather high values of instantaneous peak-signals in comparison to their average values. This is a result of the time-domain signal being composed of the sum of multiple signals which are multiplexed and transmitted simultaneously; which leads to the high peaks when multiple peaks occur at the same time. The PAPR problem in multiplexed signals is well-documented and has been thoroughly examined in the literature [10], [11].

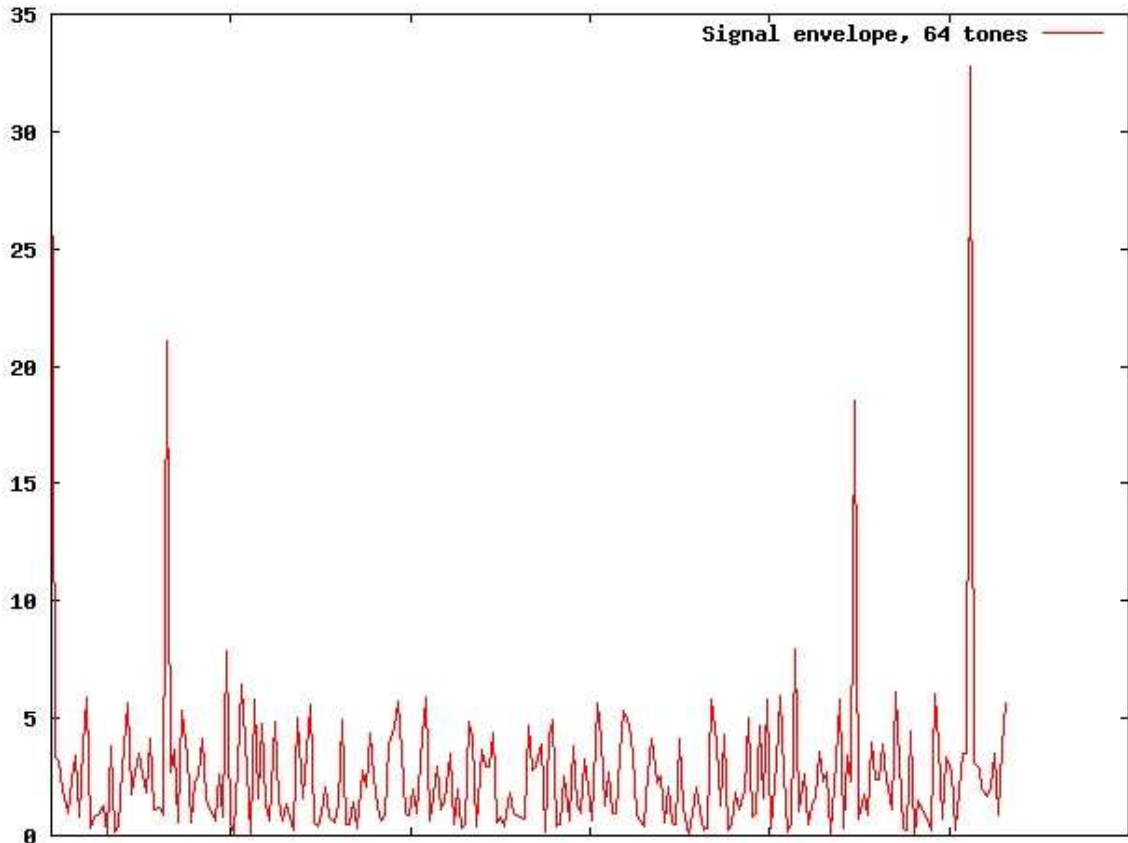


Figure 1.1 Typical OFDM signal displaying high PAPR

The implication of the PAPR issue is that the PAs used to amplify these signals will encounter large spikes in amplitudes which drive said PAs into their nonlinear regions of operation (Figure 1.2).

This leads us to consider applying input-level reduction or using less power-efficient PA classes (Appendix A) -which corresponds to a reduction of efficiency -, or investigating methods to compensate for the amplifier's nonlinear behavior through what is known as Digital Pre-Distortion (DPD) [1].

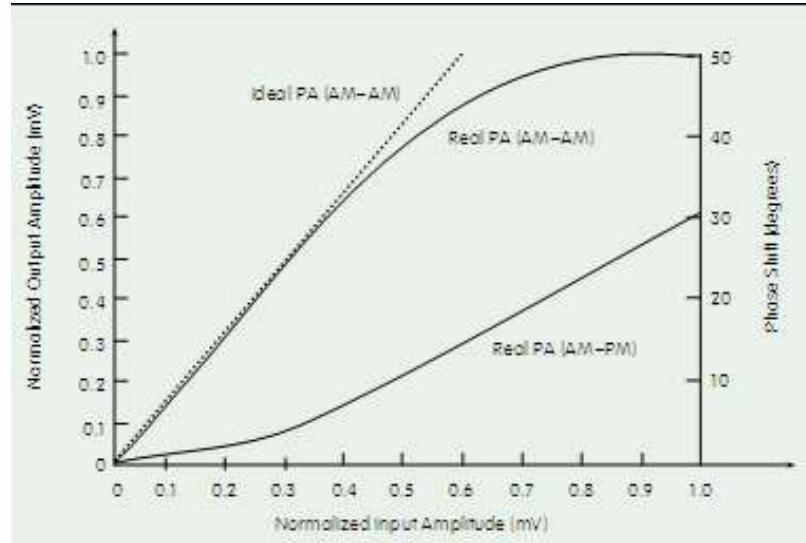


Figure 1.2 Example of Nonlinear amplifier characteristics[13]

It is noteworthy that causes behind the nonlinear behavior of power amplifiers include memory effects, and impedance variations, among others [13].

1.1.1 Memory Effects

Memory effects in a power amplifier can result from electro-thermal or electrical causes [13],[14]. Electro-thermal effects are usually borne from the heat generated by a transistor, whereas electrical memory is caused by biasing and termination imperfections [15],[16].

1.1.2 Impedance Variations

In an RF system, impedance matching is crucial to maximize the power transferred to the load . Usually, PAs are designed assuming a fixed load; which could potentially damage the amplifier's performance or even lead to physically damaging the device in some extreme cases, as reported in [17]. One situation in which this can be of concern is when the load is dynamic or time-varying.

1.1.3 Frequency-Domain Nonlinearities

When a signal is passed through a nonlinear PA, it suffers from distortions in its frequency content resulting from causes such as intermodulation effects, among others. To combat these nonlinearities, digital predistortion was developed [1].

An example of frequency-domain distortion is presented in Figure 1.3. This figure shows the effect a nonlinear Doherty PA has on the spectrum of a four-carrier (1001) 20MHz WCDMA signal. Note how there are significant spectrum components outside the bandwidth of the original signal.

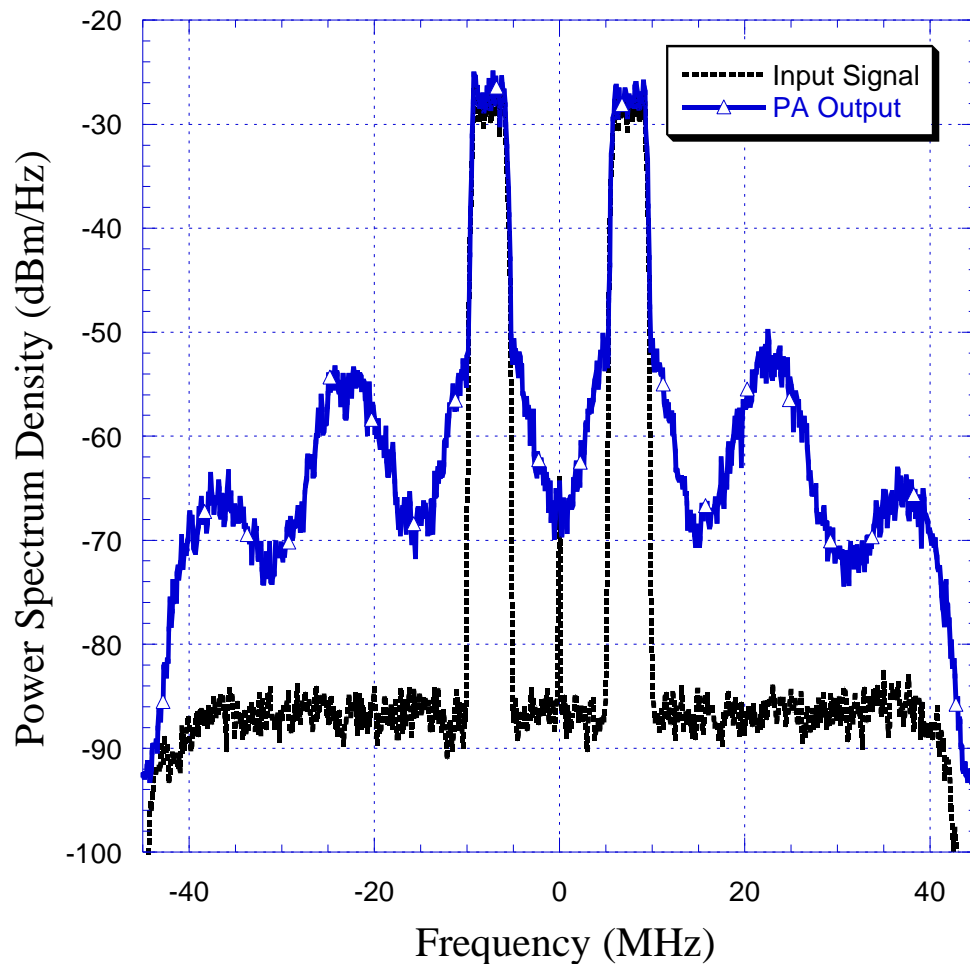


Figure 1.3 Example of distortion effects in the frequency domain

1.2 Digital Predistortion

To compensate for a PA's nonlinear behavior and produce an output that is as close to linear as possible, a technique known as Digital Predistortion (DPD) is employed [1],[2].

DPD is usually through what is known as the Indirect Learning Architecture (ILA) [18]. In this architecture (Figure 1.4), the parameters of a predistorter are identified using the output obtained from a PA as input to an estimator (after normalizing it using its small-signal gain) and the original signal $x(n)$ is used as the reference, or *desired* signal.

After the estimation process is completed, the parameters obtained are then copied to the pre-distorter which is then placed between the input signal and the PA, to obtain a linearized output signal.

Looked at in terms of system-identification, the problem of finding a pre-distorter is very similar to that of finding an inverse system and thus, the performance of the algorithms and techniques used to identify a PA's parameters would be analogous to that of identifying a DPD's parameters.

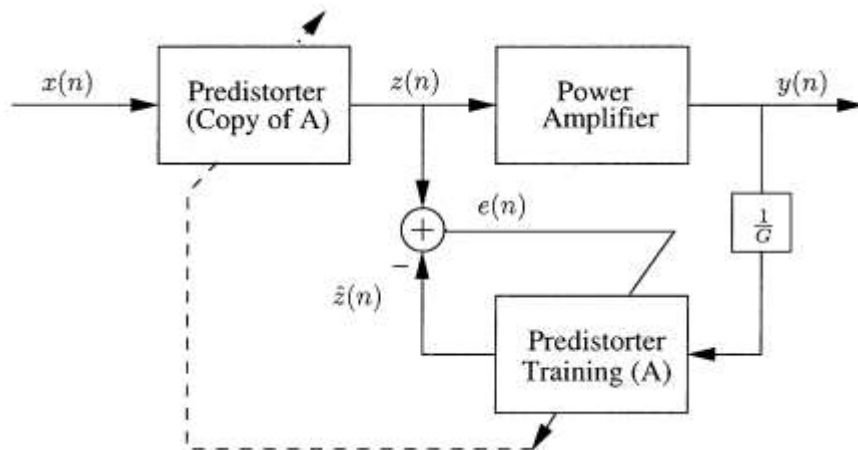


Figure 1.4 Block diagram of the digital predistortion process using the Indirect Learning Architecture[18]

Digital predistortion attempts to mitigate the effects of a PA's nonlinearities as much as possible to produce an output that is linear. An example of the effect of using a DPD in front of the same PA from Figure 1.3. From this figure, it can be observed that the out-of-band components have been suppressed significantly.

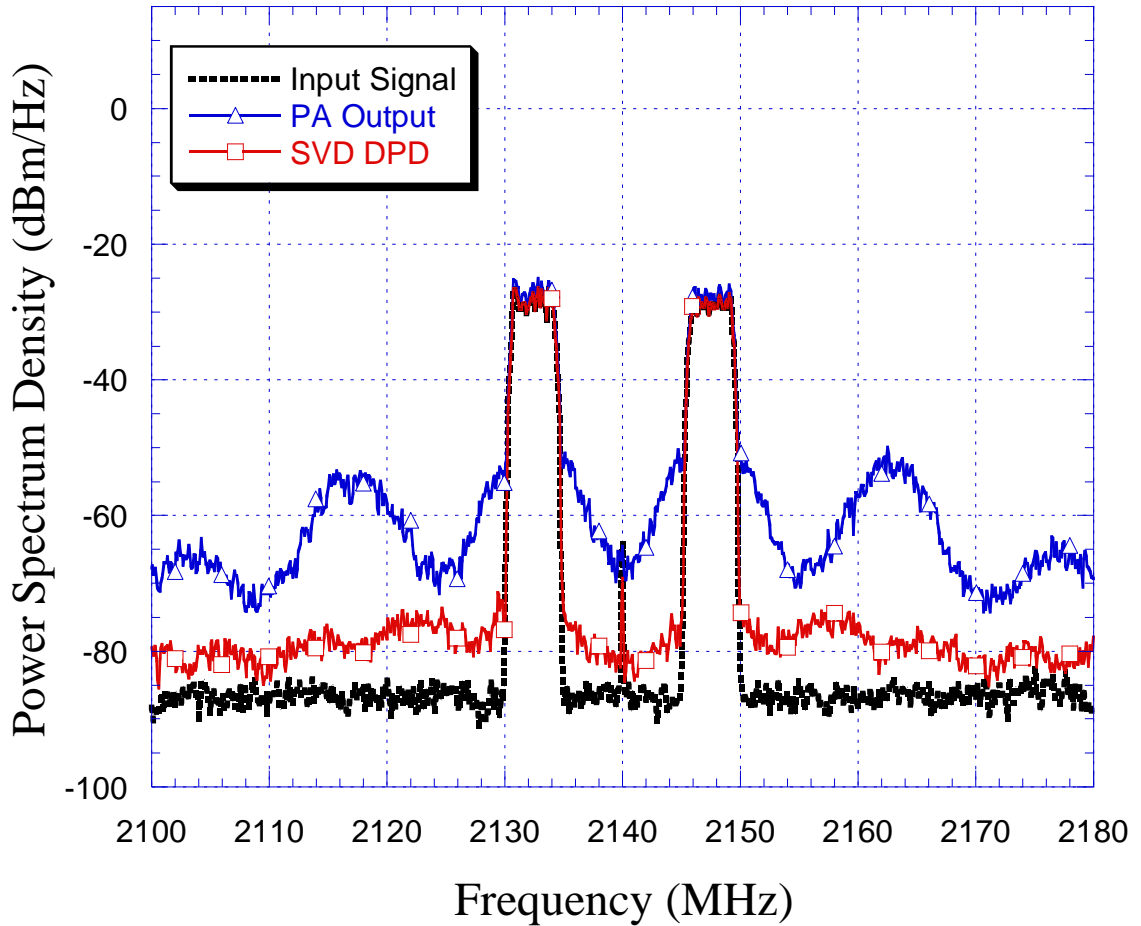


Figure 1.5 Distortion-mitigation using DPD

1.3 Problem Statement and Formulation

After giving a brief summary of the types of nonlinearities present in power amplifiers, we now state our problem: Given a nonlinear power amplifier, we would like to find an

appropriate mathematical model which describes the amplifier's behavior, then develop an estimation technique that can best extract its parameters.

To find the best technique, adaptive filtering techniques were thoroughly surveyed before settling on particle swarm optimization techniques.

After finding a technique that can best estimate the parameters of the PA models, this work attempts to develop techniques which can estimate a model's dimension. This process is illustrated in Figure 1.6 below.

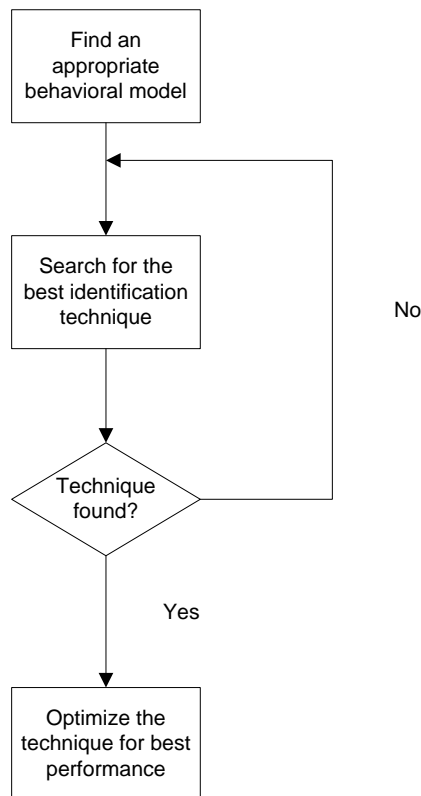


Figure 1.6 Thesis flow

1.4 Thesis Contributions

The main contributions of this work are:

1. Performing an extensive survey of the use of adaptive filters in the area of real-world PA modeling and identification and pre-distortion, then proposing some methods to improve their performance.
2. Designing and implementing a variety of novel accurate PSO estimators that are capable of estimating the true dimensions of a PA model.

1.5 Thesis Outline and Organization

This thesis is organized into five chapters, including this one: Chapter 2 conducts a survey of power amplifier behavioral modeling. Chapter 3 studies the use of adaptive filtering for the identification of nonlinear power amplifiers and identifies the shortcomings of these techniques. Chapter 4 deals with the use of Particle Swarm Optimization (PSO) to identify nonlinear power amplifier model parameters, proposes a new family of PSO algorithms that can estimate the dimension of a model along with its coefficients. and discusses the results obtained and their implications.

Finally, Chapter 5 summarizes the work done, outlines the main conclusions arrived at and offers some suggestions for future work.

CHAPTER 2

BEHAVIORAL MODELING OF NONLINEAR POWER AMPLIFIERS

After outlining the importance of modeling the behavior of nonlinear power amplifiers in the previous chapter, we discuss the mathematical models used to describe nonlinear power amplifiers and explain the approaches taken to identifying their parameters, concluding with the selection of a specific model for the remainder of this study.

2.1 Introduction to Behavioral Modeling

In its most general form, the output of a PA model can be written as

$$\mathbf{y} = f(\mathbf{x}) \tag{2.1}$$

where the input vector \mathbf{x} is mapped to an output vector \mathbf{y} through some function $f(\mathbf{x})$. The study of behavioral modeling is concerned with finding a function which best maps \mathbf{x} to \mathbf{y} in a manner that fits the operation of a real-world nonlinear PA.

The motivation behind this effort is that in order to better understand the behavior of power amplifiers, one needs to know about the models, mathematical or otherwise, used to describe them. From simple, experimentally-derived Look-Up Table (LUT)-based models to the highly complex and accurate Volterra Series, a wide variety of models that attempt to best describe amplifier behavior are available and in the initial part of this study, a number of these models were surveyed in order to select the most appropriate one. In this work, the various models are classified into two broad groups: Memoryless models and those with memory.

2.2 Memoryless Models

These models attempt to describe the nonlinear behavior of a power amplifier without taking the memory effects into account. They are usually simple to construct and analyze, but are limited in accuracy since memory effects can be significant in some situations. Among these models, the Look-Up Table (LUT) and memoryless polynomial models are discussed. Other models (such as the Saleh, Ghorbani and Rapp models) were investigated but are not included in this discussion.

2.2.1 Look-Up-Table Model

In this model, the experimentally-derived values of instantaneous gain of the amplifier is stored in a Look-Up-Table (LUT) [1] . Mathematically, the output of this model can be written as

$$y(n) = G(|x(n)|) \cdot |x(n)| , \quad (2.2)$$

where $G(|x(n)|)$ is the instantaneous complex gain associated with the particular value of the input magnitude.

In essence, this model is a direct mapping that associates the values of input magnitudes with their corresponding values of gain. A drawback of this model is that it is constructed based on our knowledge of the amplifier in question; meaning that its accuracy depends heavily on the designer's knowledge. Despite these weaknesses, the LUT is one of the most commonly-used models due to its simplicity and ease of formulation [20], [21].

2.2.2 Memoryless Polynomial Model

This model attempts to fit a polynomial to measured input and output signals. The polynomial used is of the form [22]

$$y(n) = \sum_{k=1}^K h_k |x(n)|^k \quad (2.3)$$

which can be written in vector format as

$$y(n) = \mathbf{x}(n)\mathbf{h}^T = \begin{bmatrix} |x(n)| & \cdots & |x(n)|^K \end{bmatrix} \begin{bmatrix} h_1 \\ \vdots \\ h_K \end{bmatrix} \quad (2.4)$$

where h_1 through h_K are the model coefficients. This model is simple to construct and estimate, but fails to account for the memory effects of a power amplifier, thus making it an unrealistic choice for modeling the PAs encountered in real life.

2.3 Models with Memory

In contrast to the simpler memoryless LUT- and polynomial-based models, these models attempt to account for the memory effects that affect the behavior of a power amplifier. These models fall under the class of nonlinear models with memory, which is a broad field of study studied by many researchers [23]–[30]. As one would expect, these models are more complex and (thus more expensive to compute) than the memoryless models discussed in the previous section.

2.3.1 Volterra Series Model

Named in honor of the Italian mathematician Vito Volterra, this model was first used in system theory by Norbert Wiener during the 1940's to describe the effect of noise in a

nonlinear receiver . Mathematically, the discrete-time version of this model is defined as [23]

$$y(n) = \sum_{k=1}^K \sum_{i_1=0}^M \cdots \sum_{i_p=0}^M h_p(i_1, \dots, i_p) \prod_{j=1}^k x(n-i_j) \quad .(2.5)$$

The above is a truncated version of the Volterra series where K and M have finite values. A way to intuitively understand the above expression is to view it as n -fold convolution of the input signal $x(n)$ with p filters, $h_p(i_1, \dots, i_p)$ – or as they are known in the literature, Volterra kernels [31] .

It can be seen from the above that for $k=1$, (2.5) becomes the familiar linear convolution operation performed using multiple filters (kernels).

For any $k > 1$, the kernels $h_p(i_1, \dots, i_p)$ are known as higher-order impulse responses and describe the nonlinearity of the system with K being the order of said nonlinearity. These kernels are known to be symmetric [23]. A block diagram illustrating the structure of this model is shown in Figure 2.1

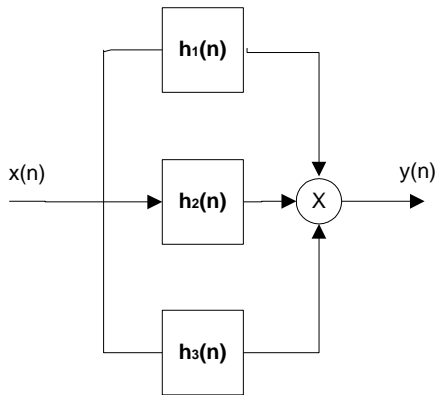


Figure 2.1 Sample diagram representing a third-order Volterra series model

Volterra series models are better suited to describing weakly nonlinear systems. These models have problems with convergence when the systems being modeled have large orders of nonlinearity, as the sum in (2.5) might not converge quickly for strongly nonlinear cases and as a result, very long computation times would be needed [31].

In return for the Volterra series' accuracy, the computational complexity it imposes is usually very large, prohibitively so at times. Due to this, various simplifications of this model have been attempted to differing degrees of success and the use of the Volterra series is usually not preferred [26].

2.3.2 Wiener Model

The Wiener model is a two-stage representation that consists of a Finite Impulse Response (FIR) filter followed by a nonlinear filter which has no memory, which can be represented by an LUT [23].

The output of a Wiener model can be written as

$$y(n) = G(|u(n)|) \bullet u(n) \quad (2.6)$$

where the gain $G(|u(n)|)$ is the result of the memoryless mapping of an input signal magnitude to its corresponding value of gain and $u(n)$ is the output of the FIR filter $H(n)$ of length M obtained through convolution

$$u(n) = \sum_{i=0}^M h(i)x(n-i) = h(n) \otimes x(n) . \quad (2.7)$$

While relatively simple, this model does not take all kinds of nonlinear behavior into account. Another weakness of this model is that it is not linear with respect to its coefficients, making identification of the model difficult [32]. To address some of these issues, the Augmented Wiener model was proposed in [33]. This model attempts to include nonlinear effects not described by the standard Wiener model by applying another FIR branch with a second-order nonlinear function, making the pre-LUT output $u(n)$ equal to

$$u(n) = \sum_{i=0}^{M_1} h_1(i)x(n-i) + \sum_{j=0}^{M_2} h_2(j)x(n-j)|x(n-j)| \quad (2.8)$$

where M_1 and M_2 are the lengths of the two FIR filters $h_1(n)$ and $h_2(n)$, respectively.

A block diagram representing the Wiener model is depicted in Figure 2.2.

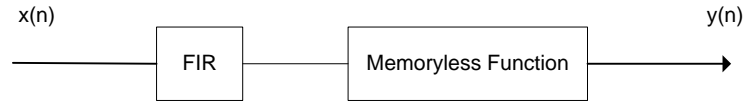


Figure 2.2 Wiener Model

2.3.3 Hammerstein Model

The Hammerstein model is an analogue of the Wiener model in which the LUT precedes the FIR filter [34]. It corresponds to a simplification of the Volterra series. The output is now expressed as

$$y(n) = \sum_{i=0}^M h(i)u(n-i) = h(n) \otimes u(n) \quad , \quad (2.9)$$

where

$$u(n) = G(|x(n)|) \cdot x(n) , \quad (2.10)$$

As can be seen from the above equations, this model can be shown to be linear in its coefficients [35]. Since it and the Wiener filter are analogues, they can function as each other's inverses if their respective FIR and nonlinear portions are inverses. As is the case with the Wiener model, an augmented version of this model exists as well, with the only difference in this version being that the LUT is applied in front of the FIR branches.

2.3.4 Wiener-Hammerstein Model

If we connect an additional FIR filter $g(n)$ after the output of the nonlinearity of the Wiener model, we obtain what is known as the Wiener-Hammerstein model [18], shown in Figure 2.3.

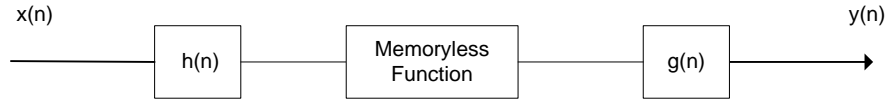


Figure 2.3 Wiener-Hammerstein Model

The output of this three-box model is expressed as

$$y(n) = \sum_{i=0}^M g(i) \sum_{k=1}^K w_k \times \left[\sum_{j=0}^{M-1} h(j) x(n-i-j) \right] , \quad (2.11)$$

The above is more general than either the Wiener and Hammerstein models, but is not linear in the set of coefficients; complicating its identification procedure and thus reducing its popularity among researchers.

2.3.5 Memory Polynomial Model

This model can be considered to be a special case of the parallel Wiener formulation.

This model has obtained good popularity in the field of PA modeling and pre-distortion.

This model was proposed in [18] and relates the output to the input through the equation

$$y(n) = \sum_{k=0}^{K-1} \sum_{l=0}^{L-1} h_{kl} x(n-l) |x(n-l)|^k . \quad (2.12)$$

The form of this model is a polynomial of order K spanning a memory depth of L time samples, hence its name. From the equation above, it can be seen that the output depends on the interaction of an input sample with polynomial versions of itself. The output can also be thought of as the output of K linear FIR filters, each preceded by a nonlinear polynomial function of order k where $0 \leq k \leq K$.

This model has the desirable characteristic of being linear in the set of parameters h_{kl} while achieving reasonable accuracy in its description of nonlinear behavior due to having two adjustable parameters K and L [32].

Another advantage of this model is its relative ease of modification, which has led to variations of it being developed [36]-[39] with each attempting to improve performance through either adding more interaction between the different memory terms or through using a different set of basis functions. One such variant is the Augmented Moving Average Model (AMA), which attempts to improve on the accuracy of the memory polynomial model (MPM) by introducing an additional summation term allowing for the cross-interaction of a sample with those of time indexes other than its own, resulting in a model that can be expressed as [40]

$$y(n) = \sum_{k=0}^{K-1} \sum_{l=0}^{L-1} a_{kl} x(n-l) |x(n-l)|^k + \sum_{m=0}^{M-1} \sum_{p=0}^{P-1} b_{mp} x(n) |x(n-m)|^k . \quad (2.13)$$

The first term of (2.13) is the standard MP model, while the second is the cross-product term introduced by this model. This modification results in a slight performance improvement of $-1\sim 2$ dB compared to the regular model in return for noticeable increase in processing time, which brings its viability into question.

An issue facing the MP model is the ill-conditioned nature of its data matrix, which has significant implications on the parameter-estimation process since the more conditioned a data matrix is, the more prone to errors the parameter-extraction process becomes [41], [42].

To combat this issue, a variant which uses orthogonal basis functions, known as the Orthogonal Memory Polynomial model, was developed in [43].

2.3.6 The Orthogonal Memory Polynomial Model

In order to reduce the amount of correlation between the samples of the MP data vector $\mathbf{u}(n)$, polynomial models that use basis functions orthogonal to one another were suggested by some researchers and subsequently investigated [43], [44]. The Orthogonal Memory Polynomial model (OMPM) is defined as

$$y(n) = \sum_{k=0}^{K-1} \sum_{l=0}^{L-1} h_{lk} \psi_k(x(n-l)) \quad (2.14)$$

which can then be written in matrix form as

$$y(n) = \mathbf{\Psi}(n) \mathbf{h}^T \quad (2.15)$$

where \mathbf{w} is the coefficient vector and $\Psi(n)$ is the $LK \times 1$ -sized vector

$$\Psi(n) = [\psi_0(x(n)) \cdots \psi_{K-1}(x(n))] . \quad (2.16)$$

Each of the elements $\psi_k(x(n))$ is constructed using the summation [43]

$$\psi_k(x(n)) = \sum_{j=1}^k (-1)^{j+k} \frac{(k+j)!}{(j-1)!(j+1)!(k-j)!} |x(n)|^{j-1} x(n) \quad (2.17)$$

Since the elements of the data vector $\Psi(n)$ are orthogonal -as proven in [43]-, the correlation displayed in the MPM data vector $\mathbf{u}(n)$ is no longer as major an issue leading us to expect that adaptive identification of the OMPM model to achieve better performance, as it has indeed been shown to result in an improvement of identification accuracy of around $\sim 3\text{dB}$ [43].

In return for this enhanced performance, OMPM requires a substantially greater amount of time and resources to generate as one could surmise from equations (2.16) and (2.17) above. Comparing this to the relatively simple structure of the MPM data vector defined in (2.12). The amount of time required for the generation of OMPM in contrast to MPM and the proposed model for various model sizes is catalogued in Table 2.1.

Table 2.1 Comparison of time required for the generation of MPM and OMPM models (in seconds)

Model	Number of Coefficients			
	15	21	27	39
MPM	.011	.0167	.0203	.0303
OMPM	2.796	3.97	5.11	7.3751

It can be seen from Table 2.1 above that the amount of time required to generate an OMP model is quite significant. For example, generating an OMP model with 15 coefficients takes 92 times the time needed to build an MP model of size 39. Due to its slow speed in return for a performance gain which usually does not exceed $-3\sim-5$ dB ([43],[44]), OMPM was not used for this study.

2.3.7 The Two-Box Twin-Nonlinear Model

This model, recently proposed by Hammi, ([45]) is comprised of two parts as its name indicates: a memoryless nonlinearity and a low-order polynomial with memory, with the memoryless part being modeled by either a Look-Up Table (LUT) or a polynomial function. Depending on how the two components of the model are arranged, the model can be referred to as forward, reverse or parallel-TNTB. In Forward TNTB, the LUT precedes the MP block whereas in the reverse case, the MP comes first and in the parallel version, the output of both blocks is summed to produce the output, as shown in Figure 2.4.

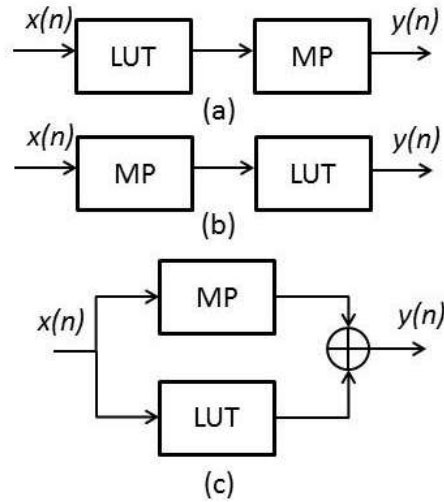


Figure 2.4 Block Diagram of the various TNTB arrangements [45]: a) Forward. b)Reverse. c)Parallel

The TNTB model has been shown to require, on average, 50% less coefficients to achieve the same accuracy as the traditional MP model which results in significant gains in algorithm performance and speed that more than make up for the minor overhead required by the two-step identification process, as shown in [46]. The performance of TNTB and the effect of its use on the identification process is investigated in the next chapter.

2.4 Choice of Model Used and Challenges Facing Parameter-Estimation

After considering various aspects of the models presented above, the memory polynomial model was found to provide the best balance between modeling accuracy and computational load and thus was chosen as the basis for this work.

The two main issues with the memory polynomial model is the correlated nature of its data vector and finding the correct size of the model.

2.4.1 Ill-conditioning of the Data Matrix

To see the role correlation plays in the identification and predistortion problem, we make a brief stop to investigate where it comes from. To do this, we take a look at the output of the memory polynomial model and rewrite it in vector form as follows:

Assuming a memory-polynomial model of known order K and memory depth L , we rewrite equation (2.12) in matrix form as follows:

$$y(n) = \mathbf{u}(n)\mathbf{h}^T \quad (2.18)$$

Where the the LK -length vectors \mathbf{w} and $\mathbf{u}(n)$ are defined as

$$\mathbf{h} = [h_{1,0}, \dots, h_{1,(K-1)}, \dots, h_{L,0}, \dots, h_{L,(K-1)}] \quad (2.19)$$

$$\mathbf{u}(n) = [\mathbf{u}_0(n), \dots, \mathbf{u}_{L-1}(n)] \quad (2.20)$$

where

$$\mathbf{u}_l(n) = [x(n-l) | x(n-l)|^0, \dots, x(n-l) | x(n-l)|^{K-1}] \quad (2.21)$$

Alternatively, $\mathbf{u}(n)$ can be written as

$$\mathbf{u}(n) = [x(n), x(n-1), \dots, x(n-L+1), \dots, x(n) | x(n)|^{K-1}, \dots, x(n-L+1) | x(n-L+1)|^{K-1}] \quad (2.22)$$

Rewriting (2.18) compactly gives us the input-output equation of the MP model in vector form

$$y(n) = \begin{bmatrix} x(n) & \cdots & x(n)|x(n)|^{K-1} & \cdots & x(n-L+1)|x(n-L+1)|^{K-1} \end{bmatrix} \begin{bmatrix} h_{0,0} \\ \vdots \\ h_{0,K-1} \\ \vdots \\ h_{L-1,K-1} \end{bmatrix} \quad (2.23)$$

Examining Equation (2.23) above, which defines the output of a memory polynomial model, $y(n)$ at time n ; we see that the data vector used as input, $\mathbf{u}(n)$ is of a particular structure where every data sample, $x(n)$, appears –in some form or other– K times. Even in the ideal case where each sample of $x(n)$ is completely uncorrelated with all others, this means that the input data vector used for computation and identification of the MP model is block-correlated and in the more realistic situation where there exists some correlation between $x(n)$ and its neighbors, the correlation becomes more prevalent. This leads to various issues relating to the identification of the model, especially for the more highly-nonlinear cases since a higher K means that a sample is repeated a larger number of times, since a correlated data vector means that the autocorrelation matrix will be *ill-conditioned*; resulting in a larger eigenvalue spread and subsequently worse identification performance overall [41],[42], as will be apparent when the simulation results are presented. To gain better insight into the inner workings of this issue, the autocorrelation matrix of an MPM data vector $\mathbf{u}(n)$ is examined.

Starting from the definition of autocorrelation

$$\mathbf{R}_{uu}(n) = E\{\mathbf{u}^H(n)\mathbf{u}(n)\} \quad (2.24)$$

Expanding the above

$$\mathbf{R}_{uu}(n) = \mathbb{E} \left\{ \begin{bmatrix} u_0 \\ \vdots \\ u_{L-1} \end{bmatrix}^* \begin{bmatrix} u_0 & \cdots & u_{L-1} \end{bmatrix} \right\}. \quad (2.25)$$

Inserting the complex conjugate operator into the column vector and multiplying, we have the autocorrelation matrix

$$\mathbf{R}_{uu}(n) = \begin{bmatrix} \mathbb{E}\{\mathbf{u}_0^* \mathbf{u}_0\} & \cdots & \mathbb{E}\{\mathbf{u}_0^* \mathbf{u}_{L-1}\} \\ \vdots & \ddots & \vdots \\ \mathbb{E}\{\mathbf{u}_{L-1} \mathbf{u}_0^*\} & \cdots & \mathbb{E}\{\mathbf{u}_{L-1}^* \mathbf{u}_{L-1}\} \end{bmatrix} \quad (2.26)$$

The above is a block-Toeplitz matrix. Inspecting a single block more closely

$$\mathbb{E}\{\mathbf{u}_j^*(n) \mathbf{u}_m(n)\} = \mathbb{E} \left\{ \begin{bmatrix} x(n-j)|x(n-j)|^0 \\ \vdots \\ x(n-j)|x(n-j)|^{K-1} \end{bmatrix}^* \begin{bmatrix} x(n-m)|x(n-m)|^0 & \cdots & x(n-m)|x(n-m)|^{K-1} \end{bmatrix} \right\} \quad (2.27)$$

$$= \mathbb{E} \left\{ \begin{bmatrix} x^*(n-j)x(n-m) & \cdots & x^*(n-j)x(n-m)|x(n-m)|^{K-1} \\ \vdots & \ddots & \vdots \\ x(n-j)^* |x(n-m)|^{K-1} & \cdots & x(n-j)^* |x(n-j)|^{K-1} |x(n-m)|^{K-1} \end{bmatrix} \right\} \quad (2.28)$$

For the special case of $j = m = p$, the autocorrelation matrix becomes

$$\mathbf{R}_{uu}(n) = \mathbb{E} \begin{bmatrix} |x(n-p)|^2 & \cdots & |x(n-p)|^{K+1} \\ \vdots & \ddots & \vdots \\ |x(n-p)|^{K+1} & \cdots & |x(n-p)|^{2K} \end{bmatrix}. \quad (2.29)$$

Using the higher moments of the absolute value of a uniform random variable $\in [0,1]$, the individual elements of (2.30) can be evaluated

$$E\left\{|x(n-p)|^{k+j}\right\} = \frac{1}{1+k+j} \quad (2.31)$$

Looking at (2.31) above, we can see that the autocorrelation matrix (2.26) and the submatrices that comprise it exhibit a degree of correlation; resulting in an increase of the condition number. To obtain an intuitive understanding of the behavior of the condition number, Figure 2.5 displays the relationship between the MPM model dimensions and the data matrix condition number in dB. Looking at the figure, we can see how even a moderately-sized MP model (e.g.: 3 branches and a nonlinearity order of 8), the value of the condition number would be rather high (108.1 dB).

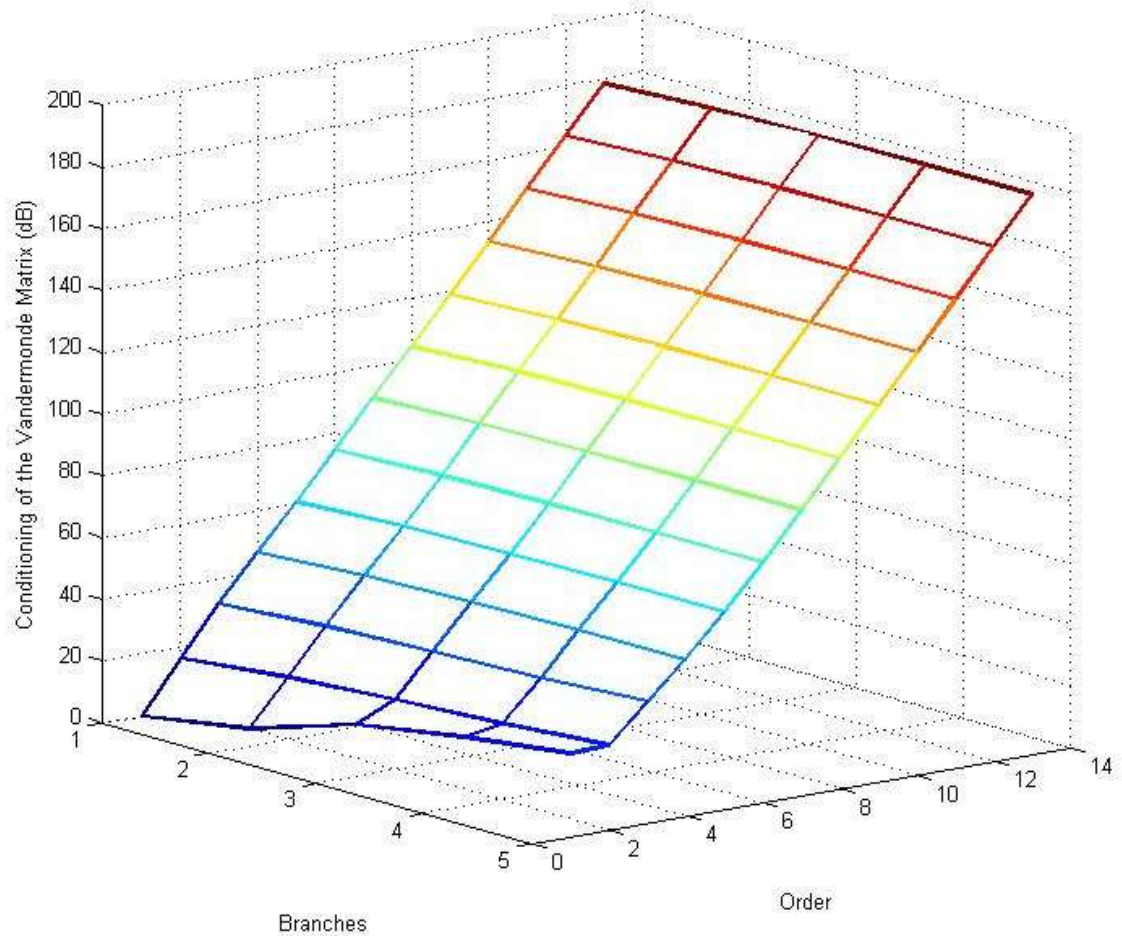


Figure 2.5 Relationship between MPM dimensions and condition number.

2.4.2 Determination of Model Dimensions

Examining the input-output relationship in (2.23) once again, it can be seen that to generate the model and subsequently estimate its parameters, a priori knowledge of the parameters K and L is assumed.

In practice, however, this information might not be available to the designer, who only has access to a PA and its practically-measured output signal and usually resorts to sweeping the model's parameters L and K until an appropriate pair is found, which can

be rather cumbersome. As a result, educated guesses about the model's dimensions are often made, leading to oversized estimates.

Since we have two parameters for this model, over-estimating the dimensionality of an MP model has three cases:

1. Overestimating L only.
2. Overestimating K only.
3. Overestimating both parameters.

Denoting the correct dimensions by (L, K) and the extra entries added as a result of over-sizing by (L_o, K_o) , each of the three scenarios –in addition to the correctly-sized case–is illustrated in Figure 2.6. In this figure, the shaded blocks indicate the entries resulting from over-sizing, with different shades being used for clarity. The data vector is constructed using the definition in (2.32).

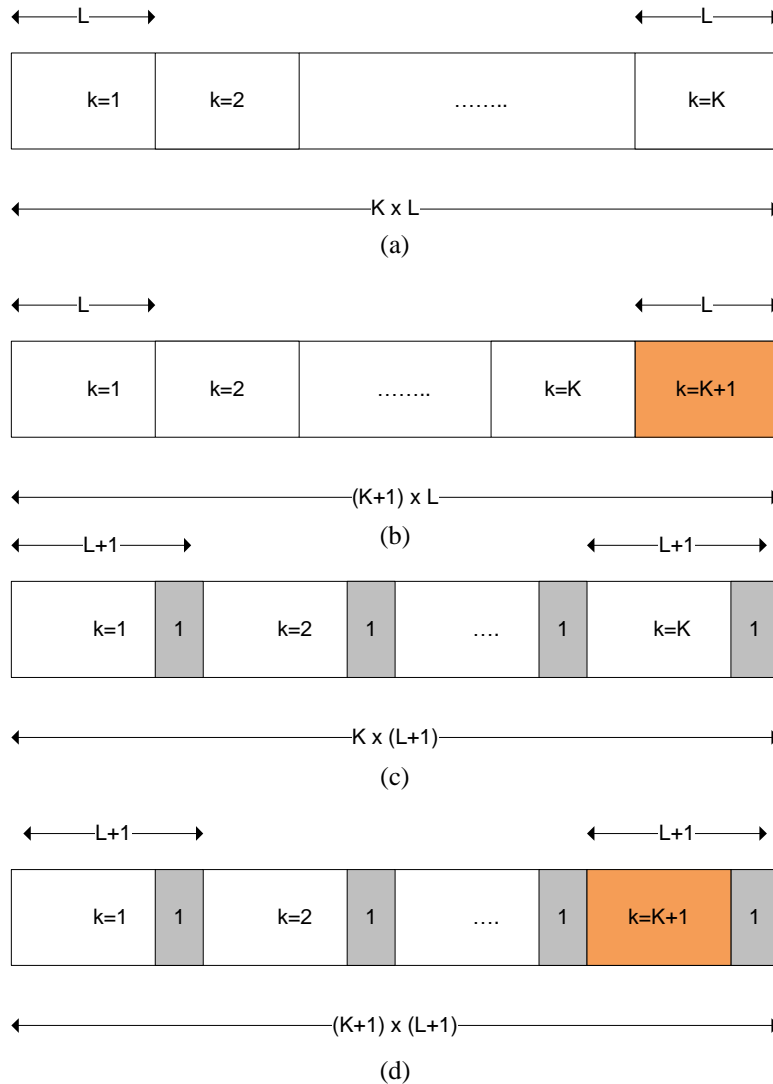


Figure 2.6 Structures of correctly- and over-sized MP model dimensions: (a) Correct dimensions. (b) Oversized K . (c) Oversized L . (d) Both parameters oversized

From the above figure, the implications of oversizing the model's dimensions even by 1 can be seen, as every additional memory block adds K additional entries periodically and overestimating the nonlinearity order by 1 results in L extra entries, and so on.

It should be noted that even when the 'correct' dimensions of the model are known, not all of the entries corresponding to the nonlinearity order K are actually required, as shown recently in [48]. This further supports the argument that the MP model is prone to having 'extra' coefficients which are not necessary, hence motivating the development of techniques tuned to handle such a case. Also, it should be noted that the value of K is usually set to be equal to or larger than L .

In light of the above, we would like to have techniques which are capable of estimating the correct dimensions of a model given an oversized initial estimate, motivating the development of the PSO techniques proposed in Chapter 4.

2.5 Metrics Used for the Evaluation of Behavioral Models and Predistorters

2.5.1 Time-Domain Metrics

The Normalized Mean Square Error (NMSE) is a time-domain metric that measures the difference between the measured and estimated output signals. Mathematically, NMSE is defined as

$$NMSE_{dB} = 10 \log_{10} \left(\frac{\|\mathbf{e}\|^2}{\|\mathbf{d}\|^2} \right) = 10 \log_{10} \left(\frac{\sum_{n=1}^N |e(n)|^2}{\sum_{n=1}^N |d(n)|^2} \right) \quad (2.33)$$

As NMSE is dominated by the in-band errors, it is not a sufficient indicator of the performance of a behavioral model or its DPD. Thus, it is used in conjunction with frequency-domain metrics or plots.

2.5.2 Frequency-Domain Metrics

There are many metrics used to evaluate the performance of behavioral models and DPDs in the frequency domain, among which are the Adjacent Channel Power Ratio (ACPR) and the Normalized Absolute Mean Spectrum Error (NAMSE) [49].

The ACPR is defined as the amount of power contained in the bands neighboring that of our signal. ACPR is subjective since it depends on the definition of spectrum ranges. Another weakness of ACPR is that its calculation relies on integrating the power over some frequency range; meaning that some components can dominate the metric.

NAMSE is analogous to the NMSE in the time domain and is defined by

$$NAMSE = 10 \log_{10} \left(\frac{\|\mathbf{Z}_{measured} - \mathbf{Z}_{estimated}\|^2}{\|\mathbf{Z}_{measured}\|^2} \right) \quad (2.34)$$

where $\mathbf{Z}_{measured}$ and $\mathbf{Z}_{estimated}$ are the power spectral densities of the measured output and the estimated out. In this study, NAMSE is modified to evaluate the performance of a DPD as follows

$$NAMSE_{DPD} = 10 \log_{10} \left(\frac{\|\mathbf{Z}_{measured} - \mathbf{Z}_{DPD}\|^2}{\|\mathbf{Z}_{measured}\|^2} \right) . \quad (2.35)$$

This metric is used in conjunction with spectrum plots to evaluate DPD performance.

2.6 Identification of Behavioral Model Parameters

The parameter-extraction of PA behavioral model parameters can be carried out through a variety of techniques, ranging from the demanding Singular Value Decomposition (SVD) (Appendix B) and Least-Square (LS) methods to adaptive filtering techniques, neural networks and Particle Swarm Optimization (PSO).

In this work, the use of adaptive filtering and Particle Swarm Optimization is investigated in depth as less-complex alternatives to SVD and LS.

2.7 Conclusions

After performing a comprehensive survey of a number of models, memoryless and otherwise, the memory polynomial was found to be the most appropriate for this study, due to achieving a good balance between model accuracy, complexity and the flexibility (i.e. the ability to be modified) and thus, it was selected for the work done in this thesis.

After presenting the model to be used, the identification of its parameters using adaptive filtering algorithms is discussed in the next chapter.

CHAPTER 3

ADAPTIVE IDENTIFICATION OF NONLINEAR POWER AMPLIFIERS

After studying the various behavioral models used, we would now like to develop techniques that estimate their parameters.

Ideally, the Least-Squares (LS) method ([50]) is used for parameter estimation due to its high accuracy, but due to the computational load required by this method, adaptive filtering approaches are preferred [50]-[52].

An adaptive filter is defined as a linear filter whose parameters are recursively adjusted according to a specific set of rules (i.e. algorithm) in order to fulfill a certain criterion, or performance goal, defined by the cost function from which said algorithm was derived. Figure 3.1 illustrates the basic concept of an adaptive filter used to identify a nonlinear system, where the filter's coefficients (weights) are updated to provide an output that most closely matches that of the unknown system whose coefficients we would like to identify.

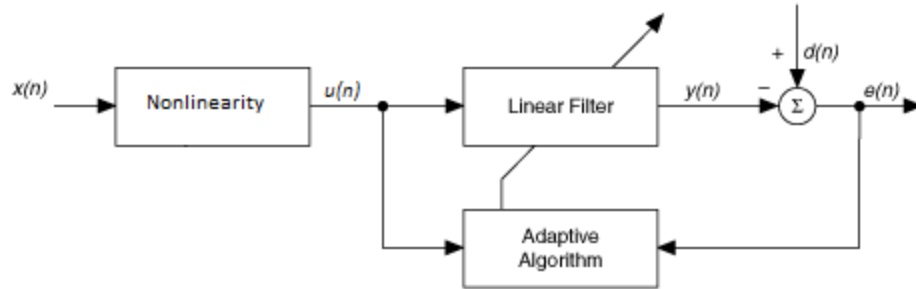


Figure 3.1 Block Diagram of an adaptive PA-identification setup

In this chapter, a quick introduction to the topic of adaptive filtering is given, followed by the introduction of a multitude of adaptive filtering algorithms and a comparison of them in terms of their performance where nonlinear systems are concerned. Each algorithm will be discussed in general, then its performance in the context of systems described by a memory polynomial models will be dissected, leading to the introduction of some methods which enhance performance.

3.1 General Definitions

An adaptive filter, as illustrated in Figure 3.1, is an iterative estimator that progressively updates the coefficients of an FIR filter in search of the set of optimum coefficients, \mathbf{h}_0 , which minimizes the difference between the filter output and some reference signal (also known as the *desired* signal) $d(n)$. The variables involved in a typical adaptive filtering setup, as shown above, are:

1. The input data $x(n)$, which in the nonlinear PA case is replaced by the appropriate nonlinear vector $\mathbf{u}(n)$ corresponding to the model of choice, in this case the MPM vector defined in (2.21).

2. The filter output $y(n) = \mathbf{u}(n)\mathbf{h}^T(n)$.
3. The reference signal $d(n)$ which is known beforehand. This signal is used to evaluate the quality of the fit by comparing the adaptive filter's output to the actual output of the PA.
4. The error $e(n)$, defined as

$$e(n) = d(n) - \mathbf{u}(n)\mathbf{h}^T(n) \quad (4.1)$$

5. The adaptively-updated set of filter coefficients $\mathbf{h}(n)$ whose values are adjusted through some set of equations depending on the algorithm used .

Usually –with some notable exceptions such as blind adaptive algorithms– adaptive algorithms use a reference output signal, often referred to as the desired signal in order to compare the output of the adaptive filter against it, and update the next set of filter coefficients to produce an output that most closely resembles the reference [51] .

A large number of adaptive algorithms are available in the literature from the simple LMS algorithms to the more complex RLS algorithms [50]–[57]. In this chapter, a quick introduction to the topic of adaptive filtering is given, followed by the introduction of a multitude of adaptive filtering algorithms and a comparison of them in terms of their performance where nonlinear systems are concerned. Each algorithm will be discussed in general, then its performance in the context of systems described by a memory polynomial model will be dissected.

Adaptive algorithms can be classified into two broad families: *Stochastic Gradient (GS)* and *Least-Squares (LS)* based adaptive algorithms [50]. Stochastic Gradients are derived by direct differentiation of some cost function – usually a power of the error function– and minimizing it to arrive at the optimal set of coefficients, whereas the Least Squares algorithms are based on iteratively solving the Least-Squares problem or some variant of it [51]. In general terms, the SG-based algorithms are not as resource-intensive as their LS counterparts but in return, suffer from slower convergence and some issues with their performance when nonlinear systems are to be identified, as will be demonstrated through the results presented later on in this chapter.

3.2 Stochastic Gradient-Based Algorithms and their Variants

This family of adaptive filtering algorithms is based on applying the stochastic-gradient method to a variety of cost functions [50], giving us a multitude of adaptive filtering algorithms each having distinct benefits and drawbacks. These algorithms replace the expectation of the cost function with the instantaneous values of the variables involved, leading to performance limitations in the form of steady-state error.

The cost function is usually of the form

$$J(n) = |e(n)|^L, \quad (4.1)$$

where different values of L result in different algorithms. Note that the above definition of the cost function does not involve expectations.

3.2.1 The Least Mean-Square (LMS) Algorithm

The LMS algorithm [57] is perhaps the most well-known adaptive filtering algorithm. By setting $L = 2$ in (4.1) and applying the stochastic-gradient method, we arrive at the LMS recursion for the weight vector update, given as,

$$\mathbf{h}(n+1) = \mathbf{h}(n) + \mu \mathbf{a}^H(n) e(n) , \quad (4.2)$$

which is the recursive update equation defining the LMS algorithm. We can see from (4.2) that the LMS algorithm is relatively simple; requiring only a low number of computations per iteration. However, the LMS algorithm and its performance depend on the statistical characteristics of the input signal; showing worse performance if the input signal is correlated or has statistics of non-white nature. Due to the structure of the input signals inherent to the MP model used, this results in the LMS performance being affected, as the results presented in this chapter indicate.

3.2.2 The Normalized Least Mean-Square (NLMS) Algorithm

In order to enhance the performance of LMS for correlated input signals, the normalized-LMS (NLMS) algorithm was proposed. This algorithm follows what is known as the minimal disturbance principle, which states that the variation of the weight vector between iterations should be kept to a minimum. Mathematically, this is stated as a constrained optimization problem of the form [51]

$$\min_{\mathbf{h}(n+1)} \|\mathbf{h}(n+1) - \mathbf{h}(n)\|^2 \quad (4.3)$$

subject to

$$\mathbf{u}(n)\mathbf{h}^T(n+1) = d(n) . \quad (4.4)$$

Using the method of Lagrange multipliers [58] , the problem can be solved to yield the NLMS update equation

$$\mathbf{h}(n+1) = \mathbf{h}(n) + \frac{\mu}{\|\mathbf{u}(n)\|^2} (\mathbf{u}(n))^H e(n) . \quad (4.5)$$

In order to avoid dividing by zero for an input vector of zero, a small coefficient δ is added to the denominator to give us the δ -NLMS algorithm

$$\mathbf{h}(n+1) = \mathbf{h}(n) + \frac{\mu}{\delta + \|\mathbf{u}(n)\|^2} (e(n))^H \mathbf{u}(n) . \quad (4.6)$$

The NLMS algorithm can be viewed as an implementation of LMS where the step-size is a variable quantity defined as

$$\mu(n) = \frac{\mu}{\|\mathbf{u}(n)\|^2} . \quad (4.7)$$

It has been shown that the NLMS algorithm converges better than LMS for correlated inputs [59], [60] but the improvement is not significant in the context of nonlinear system identification [23]. The results presented in this chapter support this.

3.2.3 The Sign-LMS Algorithm

The cost function used to derive this algorithm is

$$J(n) = |e(n)| \quad (4.8)$$

whose optimization yields the sign-error LMS update recursion

$$\mathbf{h}(n+1) = \mathbf{h}(n) + \mu \mathbf{u}(n) \text{sign}(e(n)) \quad (4.9)$$

where $\text{sgn}(x)$ is the complex signum function defined as

$$\text{csgn}(x) = \text{sign}(\text{real}(x)) + j \text{sign}(\text{imaginary}(x)) \quad (4.10)$$

with

$$\text{sign}(x) = \pm 1 \quad (4.11)$$

depending on the sign of the input [61]. In the tests performed, this algorithm performed rather poorly.

3.2.4 The Leaky-LMS Algorithm

Defining a different cost function

$$J(n) = \alpha \|\mathbf{h}(n)\|^2 + |e(n)|^2 \quad (4.12)$$

where α is a positive constant which controls the contribution the traditional LMS algorithm makes to the optimization process. By minimizing the above cost function, it is possible to get what is known as the Leaky LMS algorithm [50], [53]:

$$\mathbf{h}(n) = (1 - \mu\alpha)\mathbf{h}(n-1) + \mu \mathbf{u}^H(n)e(n) \quad (4.13)$$

In essence, Leaky LMS is a weight-controlled algorithm that attempts to combine traditional LMS and other algorithms. Setting the parameter α to zero would transform this algorithm into the regular LMS.

In [62], the Leaky-LMS algorithm's convergence behavior has been shown to depend greatly on the amount of correlation between the input samples, explaining its extremely poor performance in the simulations performed.

3.2.5 The Least Mean-Fourth (LMF) Algorithm

The Least-Mean Fourth (LMF) algorithm, as can be inferred from its name, is derived by minimizing the fourth power of the error, instead of the second power used in LMS, giving us the following cost function [63]

$$J(n) = |e(n)|^4 \quad (4.14)$$

Optimizing the above function, leads to the LMF recursion

$$\mathbf{h}(n) = \mathbf{h}(n-1) + \bar{\mu} \mathbf{u}^H(n) e(n) |e(n)|^2 \quad (4.15)$$

Note that the step-size parameter $\bar{\mu}$ is a different coefficient than the one present in LMS and usually has much smaller values.

It is well-known that the LMF algorithm outperforms the LMS algorithms in applications where the noise is non-Gaussian distributed or the systems involved are nonlinear [64], leading to expect that one might be better off using it instead of the LMS algorithm in this application. However, the simulation results presented later in this chapter suggest otherwise, as the LMF algorithm is more prone to exhibiting divergent behavior, as reported in [65]. In the literature, variants of the LMF algorithm have been proposed [66].

3.2.6 The Normalized Least Mean-Fourth (NLMF) Algorithm

Similarly to the LMS algorithm, one can develop a normalized version of the LMF algorithm, expressed as [66]

$$\mathbf{h}(n) = \mathbf{h}(n-1) + \mu \frac{\mathbf{u}^H(n)}{\|\mathbf{u}(n)\|^2} e(n) |e(n)|^2 \quad (4.16)$$

As one would expect, it shares the same input-whitening characteristic with its mean-square brethren, leading to somewhat better performance than the LMF algorithm both during convergence and in the steady-state when identifying PAs of a highly nonlinear nature [66].

3.2.7 The Least-Mean Mixed-Norm (LMMN) Algorithm

In this algorithm, the cost-function to be optimized with respect to the weight vector coefficients is chosen to be [67]

$$J(n) = \delta |e(n)|^2 + \frac{1}{2} (1 - \delta) |e(n)|^4 . \quad (4.17)$$

The cost function in this case is equal parts LMS and LMF, with the 'contribution' each algorithm makes being controlled by the parameter δ ; which takes values between 0 and 1. A value of 1 sets the algorithm to LMS whereas setting it to 0 gives us a pure LMF implementation. Use of this cost function leads to the development of what is known as the Least-Mean-Mixed-Norm (LMMN) algorithm, defined as [67]

$$\mathbf{h}(n) = \mathbf{h}(n-1) + \mu \mathbf{u}^H(n) e(n) \left[\delta + (1 - \delta) |e(n)|^2 \right] \quad (4.18)$$

Instead of manually setting this coefficient, one can allow it to change conditionally or recursively to better optimize the performance of the mixed-norm algorithm, as was done in [68]. The variable-coefficient LMMN algorithm has been found to be prone to instability when the identification of nonlinear PAs is concerned, so care should be exercised when implementing it within this context.

3.2.8 The Affine Projection Algorithm (APA)

If we chose a different approximation for the auto- and cross-correlation matrices present in some of the underling derivations, we reach at what are collectively known as Affine Projection (AP) algorithms. The AP algorithm can be considered as a batch-processing-based version of LMS that uses S data blocks instead of data samples as its input. Consequently, the definitions for the desired and input data variables are changed to reflect this batch-based nature [50]. Let $\mathbf{d}(n)$ and $\mathbf{U}(n)$ be defined, respectively, as:

$$\mathbf{d}(n) = \begin{bmatrix} d(n) \\ \vdots \\ d(n-S+1) \end{bmatrix} \quad (4.19)$$

$$\mathbf{U}(n) = \begin{bmatrix} \mathbf{u}(n) \\ \vdots \\ \mathbf{u}(n-S+1) \end{bmatrix} \quad (4.20)$$

Taking the above into account, the expression for the AP algorithm weight update can be written as

$$\mathbf{h}(n) = \mathbf{h}(n-1) + \mu \mathbf{U}^H(n) (\epsilon + \mathbf{U}(n) \mathbf{U}^H(n))^{-1} [\mathbf{d}(n) - \mathbf{U}(n) \mathbf{h}(n-1)] \quad (4.21)$$

The definition of the quantities $\mathbf{d}(n)$ and $\mathbf{U}(n)$ are what distinguish the AP algorithm from the rest of the stochastic-gradient algorithms as being a batch-processing algorithm where several blocks of data – the number of those controlled by the parameter S – are collected and then used by the algorithm in an attempt to estimate the optimum weights. The parameter ϵ is a small-valued variable used to prevent division by zero.

Since APA is a batch-based processing algorithm, one needs to account for the time delay needed to collect the required data blocks when evaluating AP algorithm's speed and such, it tends to be somewhat slower than the simpler algorithms as assessed in the simulation results section. APA-based identification of nonlinear PAs has been found to outperform the LMS algorithm and some of its variants when an impulsive perturbation is present, supporting the findings in [70]

3.3 The Least-Squares (LS) Family

These algorithms eschew the simpler stochastic-gradient approach in lieu of the least-squares approach. As a result, these algorithms have different formulations and behave differently from their SG-based counterparts. The most well-known member of this family of algorithms is the Recursive Least-Squares algorithm [50]-[52].

In contrast to the SG-based algorithms, Least-Squares-based algorithms such as the well-known Recursive Least Squares (RLS) and its derivative algorithm, the QR-RLS algorithm, are derived through attempting to solve the Least-Squares problem in a recursive manner rather than performing the matrix inversions one would usually need to do otherwise. These algorithms tend to outperform the members of the SG family, albeit at the cost of greater computational complexity [50].

3.3.1 The Recursive Least-Squares Algorithm (RLS)

The well-known Recursive Least Squares adaptive algorithm [50], based on the method of recursive least-squares, is defined by the following set of equations

$$\mathbf{Q}(0) = \delta \mathbf{I} \quad (4.22)$$

$$\pi(n) = \mathbf{Q}(n-1) \mathbf{u}(n) \quad (4.23)$$

$$\mathbf{k}(n) = \frac{\pi(n)}{(\gamma + \mathbf{u}^H(n) \pi(n))} \quad (4.24)$$

where $\mathbf{Q}(0)$ is the initial estimate of the autocorrelation matrix, δ is a parameter set by the algorithm designer which can take either small values for high-SNR signals and high values when the signal SNR is high and affects the behavior of the algorithm in its initial stages, \mathbf{I} is the identity matrix and γ is known as the *forgetting-factor* and also helps to avoid division by zero, usually taking small values below 1. Continuing,

$$\mathbf{e}(n) = d(n) - \mathbf{u}(n) \mathbf{h}^T(n) \quad (4.25)$$

$$\mathbf{h}(n+1) = \mathbf{h}(n) + \mathbf{k}(n) e^*(n) \quad (4.26)$$

$$\mathbf{Q}(n) = \gamma^{-1} [\mathbf{Q}(n-1) - \mathbf{k}(n) \mathbf{u}^H(n) \mathbf{Q}(n-1)] \quad (4.27)$$

This algorithm converges in a lower number of iterations than the SG-based algorithms and achieves a better steady-state error. The simulation results support this argument. The reason for this is that the LS-based methods are known to not be affected by the nature of the input vector as severely as their SG counterparts; explaining why they perform better

for nonlinear PA identification as the formulation used here is not dependent on the nature of the error surface or data vector as is the case with the SG algorithms. Finally, it should be noted that RLS can be susceptible to some numerical instabilities since it directly evaluates the inverse of the autocorrelation matrix; as it diverges if $\mathbf{Q}(n)$ loses its positive definitiveness, a problem which the QR-RLS algorithm, discussed next, attempts to solve [50].

3.3.2 The QR Decomposition-based Recursive Least-Squares Algorithm (QR-RLS)

A version of the RLS algorithm obtained through directly computing the product of the input vector with the autocorrelation matrix through use of the QR decomposition [50]. This algorithm is more numerically stable than RLS but differs from it in how it is implemented, as QR-RLS operates on blocks of data instead of working with samples. Additionally, QR-RLS has the advantage of being implementable using systolic arrays and enhanced numerical stability when operating under limited precision [52][56].

The QR-RLS algorithm's stability results from its use of an orthogonal similarity transform known to find the QR-decomposition with various methods for performing the decomposition being available, such as the Gram-Schmidt procedure and the Givens rotation. The Givens rotation is the preferable method since it manipulates the available data matrix, as opposed to creating a new one. Essentially, the Givens rotation attempts to extract the desirable part of some matrix by 'zeroing out' the parts which are not needed. This procedure is what equation (4.28) below describes. The QR-RLS algorithm is given as

$$\Phi^{\frac{1}{2}}(0) = \delta^{1/2} \mathbf{I} \quad (4.28)$$

$$\mathbf{p}(0) = 0 . \quad (4.29)$$

$$\mathbf{d}^T(n) = [d(1) \ \cdots \ d(n)] \quad (4.30)$$

$$\begin{bmatrix} \lambda^{\frac{1}{2}} \Phi^{\frac{1}{2}}(n-1) & \mathbf{u}(n) \\ \lambda^{\frac{1}{2}} \mathbf{p}^H(n-1) & \mathbf{d}(n) \\ 0^T & 1 \end{bmatrix} \Theta(n) = \begin{bmatrix} \Phi^{\frac{1}{2}}(n) & 0 \\ \mathbf{p}^H(n) & \mathbf{xx} \\ 0^T & \mathbf{xx} \end{bmatrix} \quad (4.31)$$

$$\mathbf{h}^H(n) = \mathbf{p}^H(n) \Phi^{-\frac{1}{2}}(n) \quad (4.32)$$

Where $\mathbf{u}(n)$ is the data vector defined earlier , δ is a regularization parameter, $\Theta(n)$ is the Givens matrix, $\mathbf{p}(n)$ is an intermediary vector used in to calculate the coefficient vector $\mathbf{w}(n)$, which is updated at each iteration according to (4.33). The constant λ is the exponential weighting parameter [71] and \mathbf{xx} represent elements whose values are disregarded.

QR-RLS is, essentially, a version of RLS that emphasizes numerical stability at the cost of speed and its performance does not stray far from that of its source of inspiration. In terms of computational cost, QR-RLS requires more operations (on the order of $(LK)^3$, as shown in Table 3.3), where LK is the number of coefficients) per iteration and even when faster versions of it are implemented, the cost remains high in comparison to RLS. QR-RLS's numerical stability comes from its use of the decompositions explained above.

3.4 Shortcomings of the Available Adaptive Filtering Algorithms and Some Solutions

As the simulation results presented in this chapter will confirm, traditional SG-based adaptive filtering techniques perform inadequately when used to estimate the parameters of a nonlinear memory polynomial model, due to the high degree of correlation inherent in its data vector $\mathbf{u}(n)$, as it is well-known that the performance of SG-based algorithms depends on the structure of the input data signal.

This suggests that one should use the LS-based algorithms but as these algorithms have high computational cost, we would like to develop some methods to improve the performance of SG-based estimators. Techniques investigated throughout this work include the widely-used data-centering and scaling techniques [40] and the proposed use of the lattice proposed in [72] to whiten the MP data vector $\mathbf{u}(n)$ before passing it to an adaptive filter.

3.4.1 Pre-processing Using Normalization and Data-centering

To reduce the conditioning of the MP model's data vector (Section 2.4.1 and Figure 2.5), it was proposed in [40] to remove the mean from the original input data $x(n)$ and normalize it by its standard deviation before passing it to the MPM generator, giving us a new input signal of

$$\dot{x}(n) = \frac{x(n) - \mu_x}{\sigma_x} \quad (4.34)$$

which is then used to construct the MP model's data vector. Use of this technique was found to reduce the condition number but has the drawback of requiring that the entire set

of training data be available to us, and that the statistical properties of the signal do not fluctuate with time. To attempt to address these shortcomings, use of a lattice filter was considered instead.

3.4.2 Use of Whitening Lattice

In [72], a nonlinear lattice structure (First introduced in [73]) was used to whiten a correlated input signal before passing it to a neural network-based decision-feedback equalizer (DFE), resulting in significant improvements in both NMSE and convergence speed. Here, the same concept is used to 'whiten' the MP model's data vector $\mathbf{u}(n)$ before passing it to adaptive estimators to enhance their performance.

In this arrangement, the signal $\mathbf{u}(n)$ is passed through the multi-stage lattice shown in Figure 3.2. The lattice output error $\mathbf{b}(n)$ is then used as the input to an adaptive filtering algorithm such as LMS, which is de-correlated due to the lattice's properties.

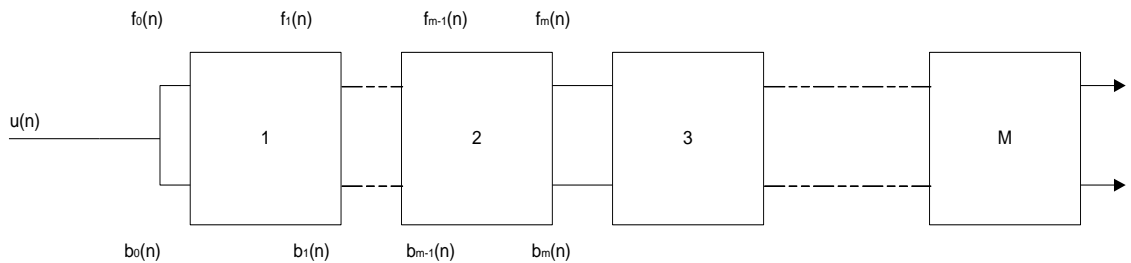


Figure 3.2. Structure of the lattice filter

The whitening effect comes from the well-known orthogonality property of lattice errors; which states that a lattice filter outputs signal components which are orthogonal to one another [51].

This lattice is composed of L stages [73], each having the structure in Figure 3.3.

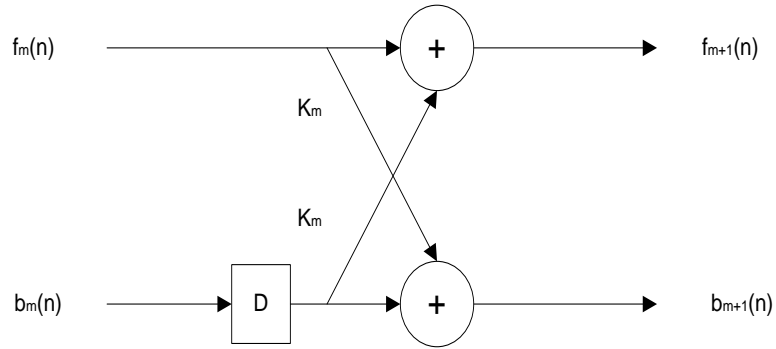


Figure 3.3. Structure of a single stage of the lattice

The variables involved in the lattice are as follows:

$x(n)$ is the input sample at time n .

$b_m(n)$ is known as the *backward prediction error*, which is obtained by estimating the value of a sample using its past values.

$f_m(n)$ is the *forward prediction error*, obtained by using the $M-1$ th most recent values of the input to estimate a past sample.

$k_m(n)$ is the *reflection coefficient* associated with each stage of the lattice. These coefficients can be obtained through a number of methods such as the Levinson-Durbin algorithm and the Schurr method [51][50].

The lattice can be combined with the memory polynomial model by placing it either before the model or afterwards, producing configurations termed the *pre-* and *post-* MPM lattices. The structure of each of these configurations is outlined next:

I. *Pre-MPM Lattice*

In this configuration, every L samples of the input signal $x(n)$ are filtered through the $L-1$ scalar stages of the lattice to generate $L-1$ samples which are as uncorrelated as possible. Subsequently, the outputs of the lattice are then used as inputs to the MP model generator. Upon implementation, this approach was found to produce lower improvements than the post-MPM lattice; which can be explained by the fact that most of the correlation in the model comes from the structure of the model itself, as opposed to the data signals used.

II. *Post-MPM Lattice*

In this approach, a vector lattice having $L-1$ stages and of size $K \times 1$ each is used to decorrelate each MPM sub-block $\mathbf{u}_k(n)$ individually. This approach is more costly to implement, due to the fact that the reflection coefficients of each stage is now a $K \times K$ matrix, as opposed to a scalar sample. As a result, the computational load of this version of the lattice combined with MPM was found to be slightly superior to OMPM for a lower implementation cost.

Table 3.1 and Table 3.2 show the effect of implementing the lattice on the conditioning and coefficient-dispersion of the MP model of dimensions $L=5, K=9$, respectively. From these tables, it can be seen that the post-MPM lattice has the superior performance at the cost of high computation time, and that L stages are required. The high amount of time required by the Post-MPM lattice can be explained by its need to find L reflection coefficient matrices of size $K \times K$ at each time step.

Table 3.1 Effect of using the lattice with the MP model

Modeling Method	Condition Number	Dispersion	CPU Time (Sec)
MPM	6.89×10^{12}	1.5×10^{10}	0.41
OMPM	8.81×10^9	5.60×10^4	47.70
Pre-MPM Lattice	2.70×10^{10}	2.5×10^9	3.81
Post-MPM Lattice	4.79×10^8	2.61×10^6	34.62

Table 3.2 Effect of the number of stages on post-MPM lattice performance

# of Stages	Condition Number	Coefficient Dispersion	CPU Time (Sec)
M= 2	6.89×10^{12}	4.68×10^9	24.01
M= 3	1.57×10^{10}	3.91×10^8	28.91
M=4	2.67×10^9	1.38×10^7	30.50
M= 5	4.79×10^8	2.61×10^6	34.62

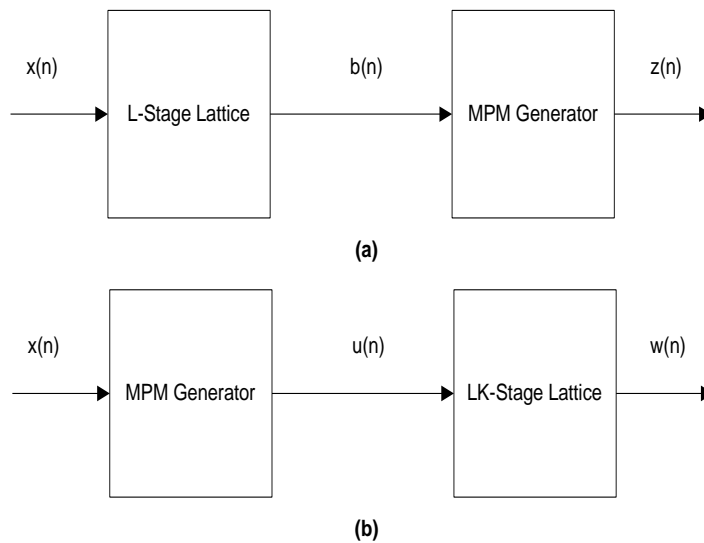


Figure 3.4. Possible arrangements for implementing the lattice whitener (a)pre-MPM (b)post-MPM

The lattice above has been modified implemented as a whitening device in conjunction with the LMS, NLMS and LMF algorithms and tested in section 3.6. Further details regarding the implementation of lattice filters can be found in [72] and [73].

As a result of implementing the post-lattice, improvements of almost -6dB in NMSE for LMS performance and better convergence were obtained in an amount of time slightly higher than that needed by RLS, with comparable improvements observed for the other SG algorithms, as the results in Section 3.7 show.

3.5 Performance Metrics for Adaptive Filters

In order to decide which algorithm to use for the identification of a particular model, one needs to have a set of criteria by which the performance of adaptive estimators can be evaluated and compared. The criteria used were the normalized mean-square error (NMSE) and the speed of convergence .

3.5.1 Normalized Mean-Square Error (NMSE)

The normalized Mean-Square Error is a direct measure of how closely the output of an adaptive filter matches the reference signal and usually, a value between $-30 \sim -40 \text{ dB}$ is indication of acceptable performance [1].

Recalling the definition in Chapter 2 Section 5, NMSE is mathematically defined as

$$NMSE_{dB} = 10 \log_{10} \left(\frac{\|\mathbf{e}\|^2}{\|\mathbf{d}\|^2} \right) = 10 \log_{10} \left(\frac{\sum_{n=1}^N |e(n)|^2}{\sum_{n=1}^N |d(n)|^2} \right) . \quad (4.35)$$

where N is the total length of the desired and error vectors.

3.5.2 Convergence Speed

The convergence speed of an adaptive filter is measured by the amount of time-steps (iterations) an algorithm takes to reach a value of NMSE close to the optimum and settles within its vicinity. In general, SG-based algorithms tend to have slower convergence in comparison to their RLS counterparts [50],[51].

3.5.3 Computational Complexity

the complexity of an adaptive estimator is often measured by the number of operations that need to be performed in every iteration. Taking into account that some operations are more costly than others, the number of multiplications per iteration is the metric used in this work to give a general idea of how complex an algorithm is. The importance of complexity comes into view when one wishes to move from simulations into an implementation using FPGAs or on-device predistortion chips; as power consumption becomes an issue of great significance in such situations.

Table 3.3 summarizes the different algorithms and their corresponding computational complexity for a data vector of length LK [50], keeping in mind that S is the number of data blocks used by the AP algorithm in each iteration.

From the expressions in this table and the numerical results presented in this chapter, it can be seen that the LS-based algorithms have a high computational cost, especially QR-RLS. Examining the table, we can see that the lattice pre-whitener is less demanding than QR-RLS and RLS as the vector size increases, which makes it quite suitable for estimating an MP model with a large number of parameters.

It can also be seen that SVD has the highest computational requirement in return for its superior performance.

Table 3.3 Computational cost of the various adaptive algorithms tested

Algorithm	Number of Multiplications Per Iterations
LMS	$8(LK) + 1$
NLMS	$8(LK) + 1$
Leaky-LMS	$10(LK) + 3$
LMF	$8(LK) + 5$
NLMF	$8(LK) + 5$
LMMN	$8(LK) + 6$
APA	$(S^2 + 2S)(LK) + S^3 + S$
RLS	$4(LK)^2 + 16(LK) + 1$
QR-RLS	$9(LK)^3 - 3(LK)^2 + 28(LK) - 1$
SVD	$4(LK) + 22(LK^3)$
Lattice-filtered LMS	$18(LK) + 39(LK)$

3.6 Comparative Study of Adaptive Identification of Nonlinear Power Amplifier Parameters

The Device Under Test (DUT) is a Symmetrical Doherty PA (Appendix A) built using Cree's 10W packaged GaN devices (CGH400010) with a central nominal frequency of operation of 2.425GHz. The carrier amplifier is biased for class AB operation with $I_{DS}=200\text{mA}$ and $V_{ds}=28\text{V}$. The peaking amplifier is biased for class C, with both carrier and peaking amplifiers being harmonically tuned. The input signal is a complex 4-carrier WCDMA signal (1001) with a total bandwidth of 20 MHz, sampled at 92.6MHz.

The measured AM-AM and AM-PM characteristics of the device under test are reported in Figure 3.5 and Figure 3.6.

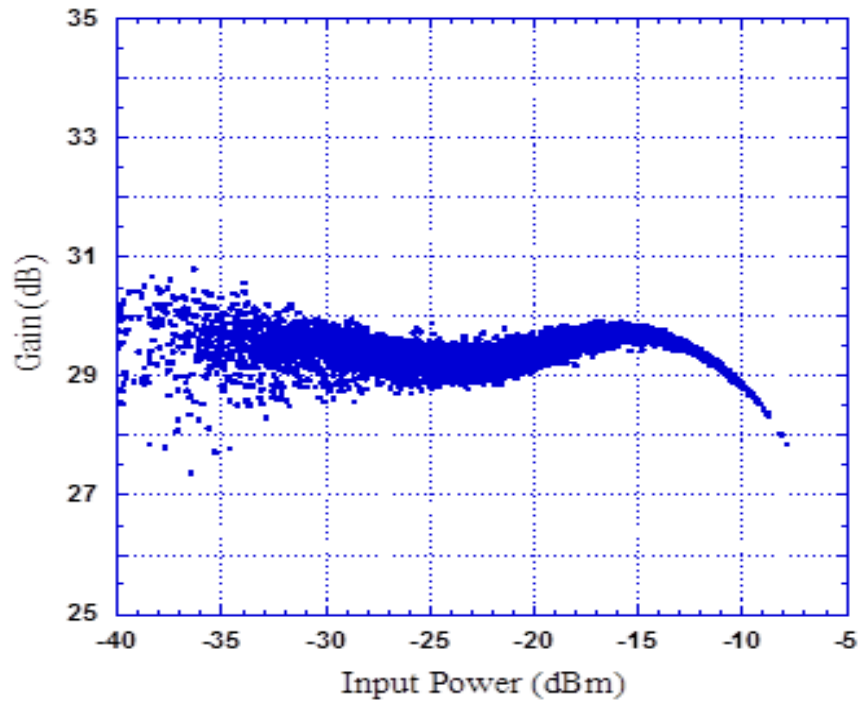


Figure 3.5 AM/AM Characteristics of the DUT

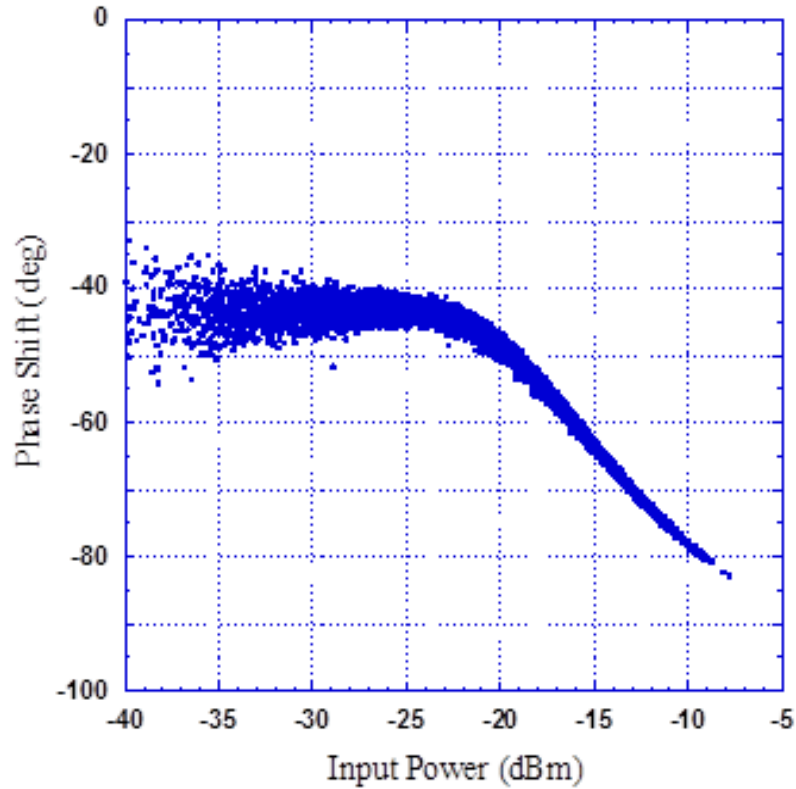


Figure 3.6 AM/PM Characteristics of the DUT

To validate the performance of adaptive filtering algorithms, the PA was modeled once using a memory polynomial with a memory depth of $L=3$ and a nonlinearity order of $K=6$, chosen after performing an extensive sweep analysis.

The nonlinear PA model was identified using a variety of adaptive algorithms under using the MATLAB R2012a software on an Intel Core i7 CPU, E8400 1.73GHz computer.

For simplicity, the learning curves of only the six most prominent algorithms (NLMS, Lattice LMS, APA, LMF, RLS, QR-RLS) were chosen in order to avoid cluttering, with the rest of the algorithms having their performance recorded in Table 3.4. Figure 3.7 shows the convergence behavior of the adaptive algorithms when estimating the

coefficients of a nonlinear MP model, with an observation period of 200 iterations. In these experiments, the algorithms were initialized with coefficients that are not close to the actual values to test for a realistic scenario where the designer has no prior information about the PA parameters.

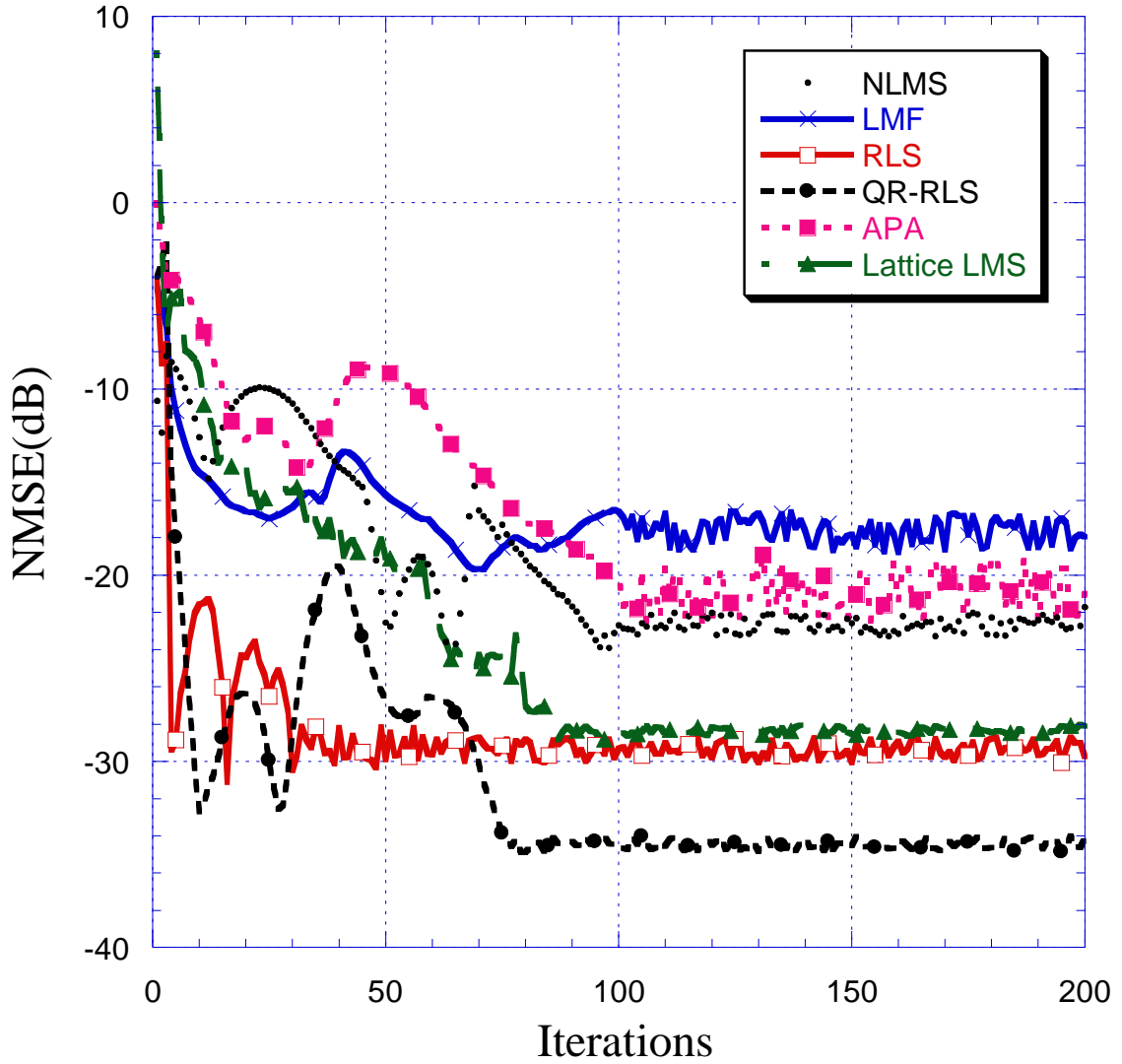


Figure 3.7 Convergence behavior of the different adaptive algorithms when estimating the parameters of an MPM-based predistorter

The results presented in the figures and table given show that among the SG-based algorithms, there wasn't much variation in terms of steady-state performance, convergence speed and complexity (with the notable exception of the batch-based AP algorithm); with all of them exhibiting generally poor performance in terms of estimation error, ranging from -17 to about -24 dB. The reason for this poor performance is the high level of correlation introduced into the input data vector $\mathbf{u}(n)$ by the MP model. Note that since SVD is implemented in a different fashion than adaptive filters, its performance is gauged in terms of NMSE and frequency-domain linearization.

When the lattice was applied to LMS, a gain in performance of almost -10 dB was obtained, achieving performance similar to RLS while requiring less than one third the CPU time and one tenth the time needed by QR-RLS, the best algorithm in terms of NMSE.

As for the LS-based algorithms, both the RLS and QR-RLS algorithms outperform their SG counterparts in terms of performance, with QR-RLS having achieved a better NMSE than RLS, at the cost of requiring a very large amount of time to implement and having a high computational cost. Of particular interest is the fact that in terms of the number of total operations required to achieve convergence, RLS actually requires less multiplications to converge than the SG algorithms, since it converges to a much better estimate much more quickly than all of them.

Looking at the frequency-domain plot given in Figure 3.8, the performance of DPDs implemented using adaptive algorithms can be inferred. Among the algorithms tested, NLMS was found to perform rather poorly in terms of error, linearization and sidelobe-

suppression. In contrast, the Lattice-filtered LMS and RLS algorithms had the best performance, noting that applying the lattice to NLMS and LMF did not provide smaller improvements than those obtained in the LMS case.

Overall, none of the adaptive algorithms were capable of competing with SVD and QR-RLS, performance-wise.

Table 3.4 Comparison of the performance of the various adaptive identification algorithms

Algorithm	NMSE(dB)	NAMSE(dB)
LMS	-20.76	-7.11
NLMS	-23.33	-9.88
LMMN	-23.90	-9.21
Leaky LMS	-16.18	-5.64
Sign-Error LMS	-15.35	-5.01
LMF	-18.81	-6.83
NLMF	-21.12	-7.92
APA	-22.03	-8.18
Lattice LMS	-26.11	-12.52
Lattice NLMS	-25.61	-12.22
Lattice LMF	-24.01	-11.87
RLS	-29.83	-12.82
QR-RLS	-34.02	-14.81
SVD	-35.82	-15.77

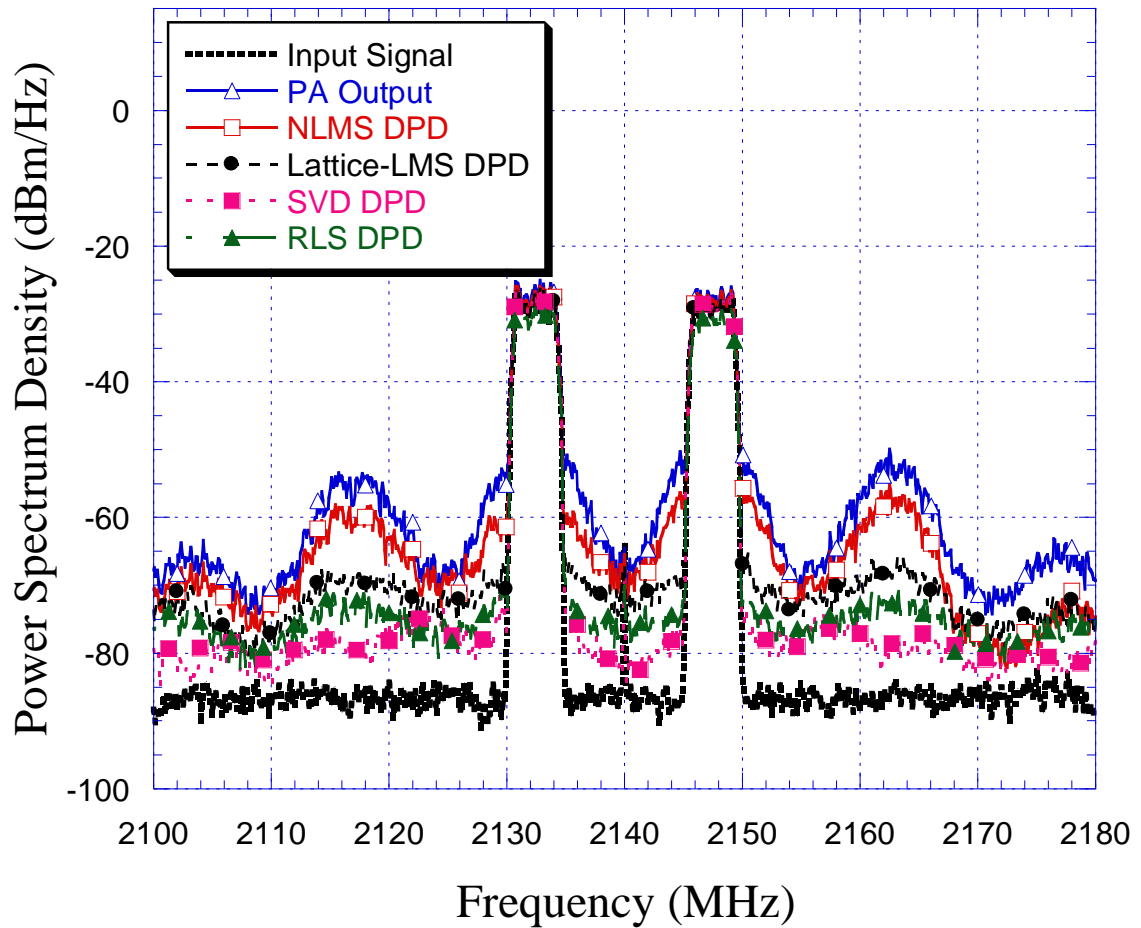


Figure 3.8 Predistortion performance of the various adaptive algorithms and SVD

3.7 Results and Conclusions

After introducing the various adaptive algorithms, this chapter illustrated the differences between them in terms of accuracy, convergence, behavior under perturbation, and speed of convergence. As a result of this study, the following observations were made:

- 1- Stochastic-Gradient-based adaptive estimators, in general, perform poorly when identifying highly-nonlinear models when compared to the LS-based algorithms by a margin of 8 to 10dB.

2- Among RLS and QR-RLS, QR-RLS performs better but takes a much longer amount of time to converge to its optimal coefficient estimate.

3- The use of a lattice enhances SG-based adaptive filtering (especially LMS) greatly in terms of convergence and error performance, giving an NMSE improvement of around 8dB. Additionally, the whitening lattice was found to be well-suited to estimating large models such as the model at hand.

4- Even with the use of various pre-processing techniques such as lattices and centering, the performance of adaptive filtering still leaves much to be desired when it comes to estimating the parameters of nonlinear amplifier models, as none of the algorithms was capable of approaching the performance of SVD, even ones as complex and demanding as QR-RLS.

As such, it was found that adaptive estimation of nonlinear PA models proves to be of less-than-desirable performance due to the many issues associated with its use; motivating the investigation of alternative methods such as Particle Swarm Optimization and its variants, which is the topic of the next chapter.

CHAPTER 4

DIGITAL PREDISTORTION USING PARTICLE

SWARM OPTIMIZATION

As we have seen in the previous chapter, the use of adaptive filtering for identifying nonlinear power amplifiers is faced with some serious shortcoming and challenges in return for its flexibility and relatively low computational cost in contrast to methods such as the method of least squares (LS) and singular-value decomposition. As a result of these issues in conjunction with recent developments in the field of parallel processing, consideration has been given to techniques such as Particle Swarm Optimization (PSO) [3], as viable means of implementing robust, efficient pre-distorters that have been found to perform reasonably well [74],[75].

However, most of the attempts to use PSO for the identification and pre-distortion of nonlinear PAs tends to either focus on PA models that are either memoryless [74] or use multiple-box models that describe weakly-nonlinear systems with a low number of parameters such as the Wiener model used in [75].

Another shortcoming of the aforementioned attempts is that they make the assumption that the correct dimension of the model to be estimated is known in advance whereas in reality, this information is often not available to a designer who only has access to input-output signal pairs. To deal with this issue, this work proposes a variant of the PSO

algorithm that can better estimate the coefficients of a nonlinear model with memory when there are zero-valued coefficients.

In light of the above, this work investigates the use of the PSO algorithm for the identification of a highly-nonlinear Doherty PA exhibiting memory effects modeled with an instance of the well-known Memory Polynomial (MP) model that uses a high number of parameters. After introducing the basic PSO and using it to identify the aforementioned PA, variants of the algorithm are introduced and their performance is compared with the available algorithms

4.1 Basic Structure of the PSO Algorithm

The biologically-inspired Particle Swarm Optimization (PSO) algorithm was first developed by Kennedy and Eberhart in their seminal 1995 paper [3].

PSO utilizes a group (swarm) of agents known as particles that cooperate with one another and share information to achieve some goal or other, similar to how a school of fish would move in a coordinated manner to find food or evade a threat. Using this concept, PSO can be used to solve a variety of optimization problems. In this study, PSO is used to minimize the NMSE.

The basic version of PSO progresses in two stages: initialization and computation. In PSO, each particle has a *position* \mathbf{p}_i which is updated at each iteration step n by adding the particle's current position to a velocity term \mathbf{v}_i whose definition depends on the algorithm in use. Each position vector represents a possible solution and an implementation of PSO can be thought of as having a group of agents scan a vector space to find the optimum solution that minimizes the cost function at hand.

1- Initialization Stage:

In the *initialization* stage, each particle is assigned a random starting position $\mathbf{p}_i(0)$ within the boundaries of the problem space as defined by the user. The velocities are similarly given initial values for each particle $\mathbf{v}_i(0)$, that lie within the velocity boundaries. Usually, the velocity boundary is taken to be half that of the position

$$\mathbf{p}_i(0) \in [\mathbf{p}_{min}, \mathbf{p}_{max}] \quad (5.1)$$

$$\mathbf{v}_i(0) \in [\mathbf{v}_{min}, \mathbf{v}_{max}] . \quad (5.2)$$

In this work, the boundaries were intentionally set to be large; since it is assumed that the user does not have much advance knowledge of the device to be tested. This has the effect of increasing the amount of time required for the initial convergence stage; since the particles are now searching a broader solution space. A large search space reflects the lack of a priori information about the problem at hand and results in slower convergence [3].

After randomly initializing the locations of the particles, a fitness (or *objective*) function is evaluated for each particle using its parameter vector to determine which of them has the best position corresponding to the lowest value for the fitness function – denoted as the *global best* –, which is then stored in its own vector. The objective function commonly used is the normalized mean squared error (NMSE), rewritten here for convenience

$$Fitness_{PSO} = 10 \log_{10} \left(\frac{\|e(n)\|^2}{\|d(n)\|^2} \right) = 10 \log_{10} \left(\frac{\|d(n) - \mathbf{u}(n)\mathbf{P}_i\|^2}{\|d(n)\|^2} \right) \quad (5.3)$$

noting that since PSO is a batch-based technique, the fitness function is evaluated for different data blocks and then averaged. In this work, 100 data blocks of size LK each were used.

Finally, the position of each particle by the end of the initialization process is designated as its 'local best' up to that point and saved.

2- Iterative computation stage :

Following the conclusion of the initialization phase, the algorithm then moves into the main iterative computation stage. In this stage, the position of each particle is updated according to the following equation

$$\mathbf{p}_i(n) = \mathbf{p}_i(n-1) + \mathbf{v}_i(n) \quad (5.4)$$

where the velocity of the i th particle, \mathbf{v}_i , is determined by:

$$\mathbf{v}_i(n) = \omega * \mathbf{v}_i(n-1) + b * (\mathbf{pbest}_i - \mathbf{p}_i(n)) + c * (\mathbf{gbest} - \mathbf{p}_i(n)) . \quad (5.5)$$

The variables involved in the above update equation are:

The inertia weight ω : which is usually assigned values that decrease with the passage of time. This facilitates fast exploration of the solution space initially and more steady convergence later on.

The cognitive coefficient b : which determines how much influence a particle's best position is allowed to have on updating the particle's new position.

The social acceleration constant c : this parameter fulfills the same role as b but for the global best estimate.

Since the performance of the PSO algorithm depends on the values chosen for its parameters, a sensitivity analysis must be conducted beforehand to choose the best parameters .

The next step in the PSO computational stage would be the evaluation of each particle's fitness function to determine whether the particle's current position results in a better fit than the previously recorded **pbest_{*i*}**, and if so, whether the current position produces a better fit than the swarm's global best position, **gbest** .If a particle's position gives a better fit than **pbest_{*i*}**, or **gbest** , it is used in their place.

This process is repeated until the PSO is run for the maximum number of allowable iterations or the PSO reaches or exceeds the performance threshold set in advance.

Figure 4.1 below describes the general flow of the PSO algorithm

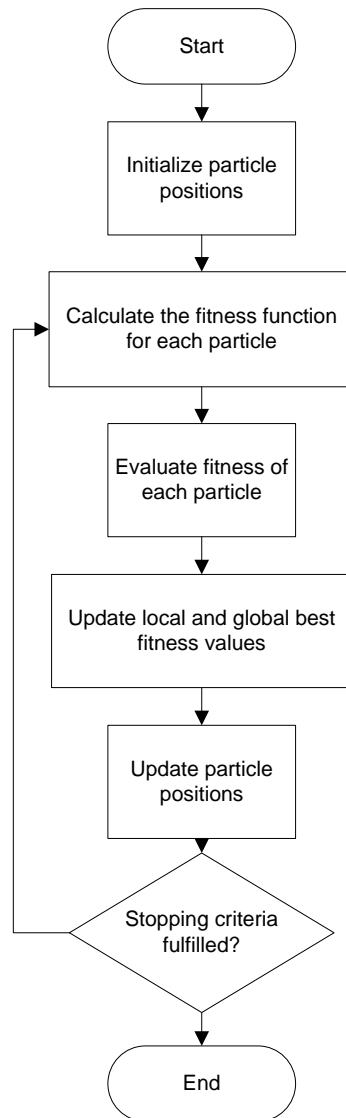


Figure 4.1 Block Diagram illustrating the flow of the PSO algorithm

The simulations performed in this work show that the PSO algorithm and its variants greatly outperform most of the adaptive algorithms tested in terms of performance and

To improve the performance of the basic PSO algorithm, variants of it algorithm were developed throughout the literature [76]-[78].

4.2 Some of the PSO variants in the literature

Since PSO is a flexible technique that is tailor-made to each specific problem, many variants of PSO were developed and investigated in the literature since its inception, each tailored to some specific application or other. In this section, some of the better-known PSO variants will be discussed.

4.2.1 Constriction-Factor PSO

In [78],[79], it was suggested and subsequently shown that the introduction of a constricting factor to the velocity update term could enhance convergence stability and improve performance. Taking this factor into account, the velocity term is now defined as

$$\mathbf{v}_i(n) = K * [\mathbf{v}_i(n-1) + b * (\mathbf{pbest}_i - \mathbf{p}_i(n)) + c * (\mathbf{gbest} - \mathbf{p}_i(n))]. \quad (5.1)$$

Where K is the constriction factor calculated using the following equation

$$K = \frac{k}{2 - (c + b) - \sqrt{(c + b)^2 - 4(c + b)}} \quad (5.2)$$

In [77], the constriction factor K was made to be time-varying; decreasing in value with every iteration. This was achieved through defining the numerator k in the following manner:

$$k(n) = k_{min} + (k_{max} - k_{min}) \frac{m - n}{m - 1} \quad (5.3)$$

Where m is the maximum number of iterations. The effect of this definition of K is further improvements in stability, leading to the use of this factor through the various PSO implementations studied in this work. In the same work, the variable-constriction

factor PSO was found to outperform both the basic and constant-constriction factor PSO algorithms and was thus chosen over the both the basic and constant constriction-factor PSO for this study.

In general, PSO and its variants outperform adaptive algorithms in terms of NMSE when used to identify nonlinear power amplifiers, as the results in Figure 4.2 show for the identifying the same model used in the previous chapter. From these results, we see that the PSO algorithms achieve an improvement in performance over RLS of about 2~3dB when run with a swarm of size 200. The detailed analysis and performance comparison which will be given in the next chapter support these results.

It should be noted that since PSO methods have the advantage of using multiple agents in parallel, a direct comparison with adaptive filtering based on NMSE alone might not be fair so the amount of time required should be looked at when making a comparison. Nevertheless, the advantage of PSO algorithms over adaptive techniques is evident in when it comes to both the estimation of DPD coefficients and reducing frequency-domain nonlinearities (Figure 4.2 and Figure 4.3.). Numerical results for this experiment are given in Table 4.1.

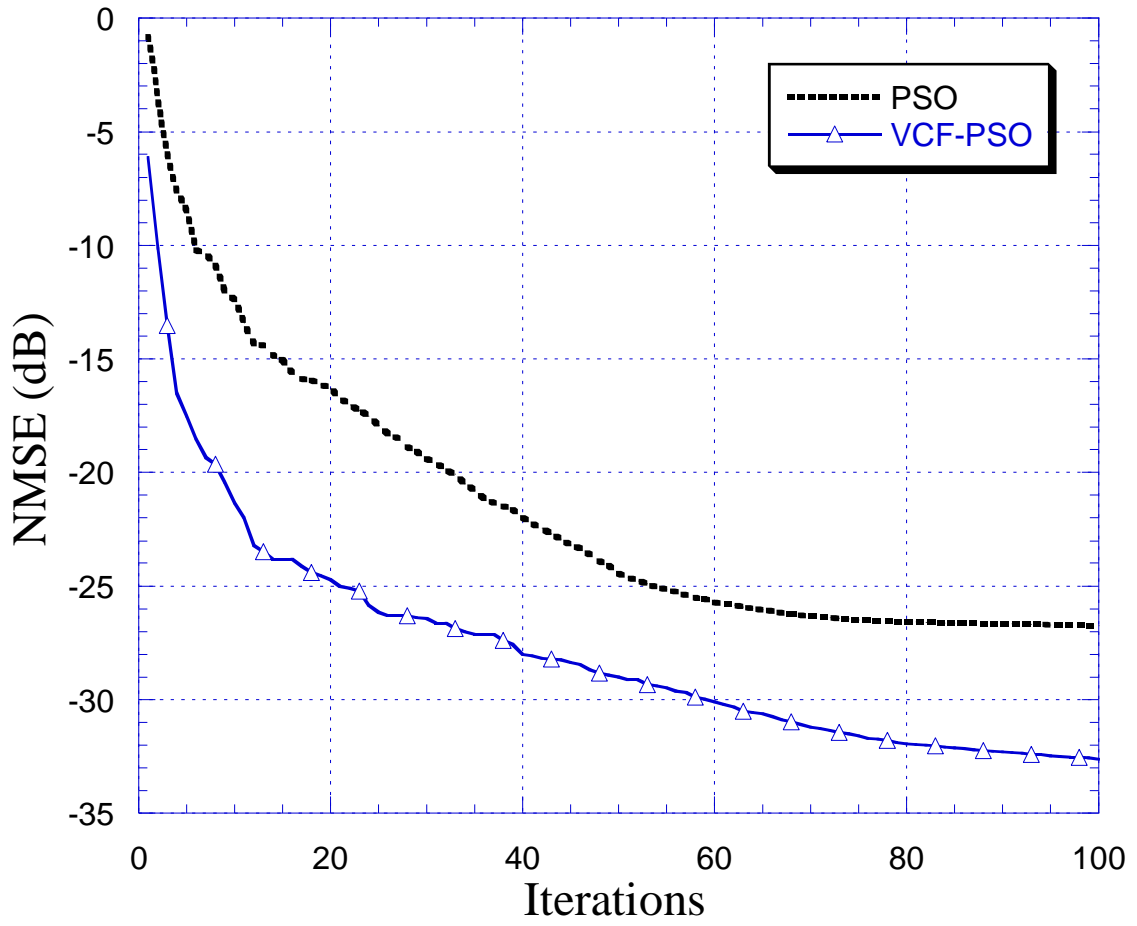


Figure 4.2 Comparison of PSO algorithms

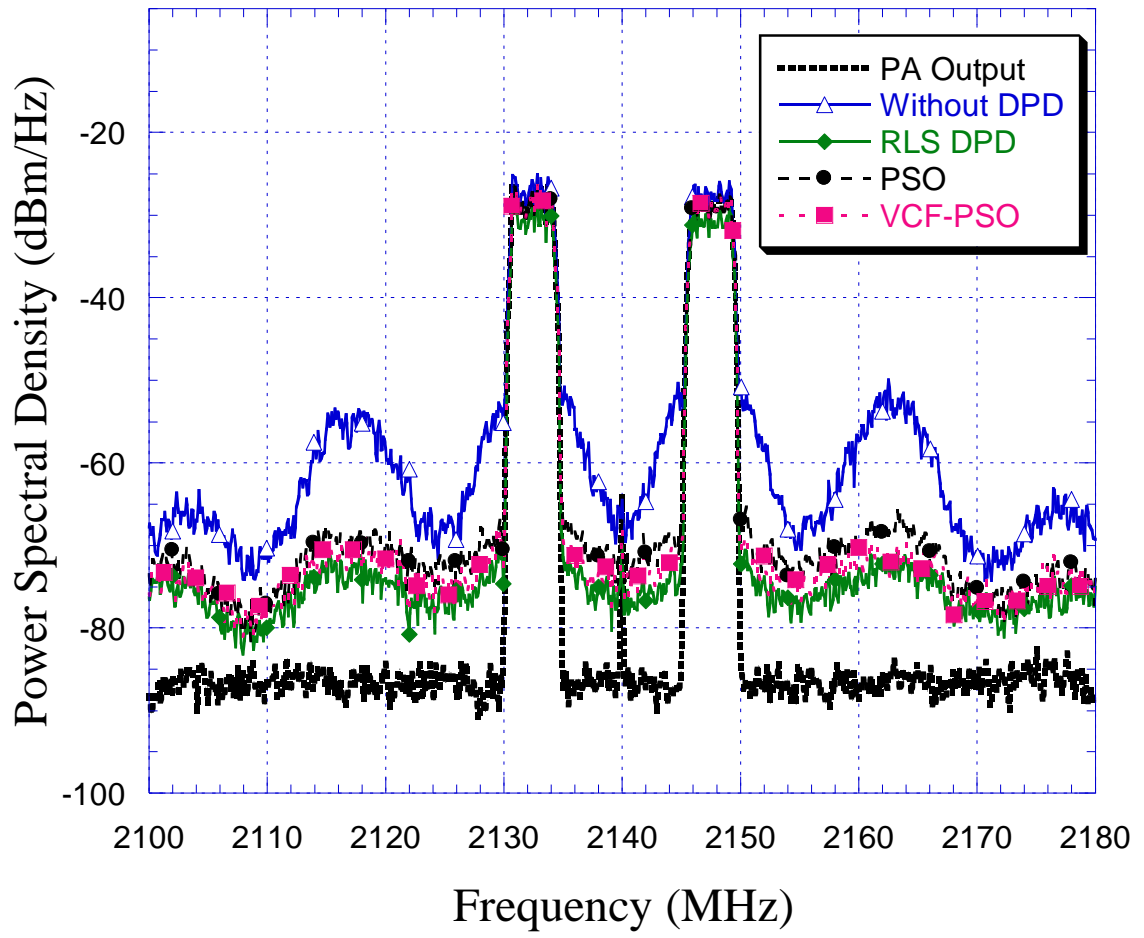


Figure 4.3 DPD performance of RLS and PSO algorithms

Table 4.1 Numerical results comparing adaptive algorithms and PSO

Algorithm	NMSE (dB)	NAMSE (dB)	Number of Multiplications/Iteration
NLMS	-22.70	-9.88	$8(LK + 1)$
RLS	-29.95	-12.82	$4(LK)^2 + 16(LK) + 1$
PSO	-27.01	-13.09	$3S(LK)$
VCF-PSO	-32.82	-13.28	$3S(LK)$

Where S here is the size of the swarm. It should be noted that since PSO is a parallel technique which can be implemented using parallel processing, the time required to run it is significantly less than what would be expected, furthering its advantage over RLS and QR-RLS, especially as the size of the vector LK increases. The results above show the potential of PSO techniques when used for developing DPDs.

4.3 Shortcomings of available PSO techniques

While they generally achieve good performance gains, as seen above, there remains some room for improvement in the implementation of PSO techniques to identifying DPD coefficients, since these methods are blind to the particular structure of the model used and simply try to populate all the entries of a weight vector without taking the actual dimensionality and nature of the model into consideration.

For example, consider the case of the MP model. Even if the 'correct' dimensions of an MP model are known in advance, some of the model's coefficients are un-necessary, more so if the model dimensions are overestimated (Chapter 2, Section 4). When

identifying such a model, the available PSO techniques simply ignore this structure and try to populate their estimated vector in a way that minimizes the cost function. This results in situations where PSO estimates an MP model of size LK and produces a fully-populated coefficient vector whereas the actual vector has zero entries in it.

To illustrate, such a situation is demonstrated in Figure 4.4; where PSO is used to estimate a coefficient vector having a length of 30 taps with only 15 nonzero entries. Note how PSO produces an estimate of the vector that has many samples of nonzero magnitude, even though the actual coefficient vector has zeroes in those positions.

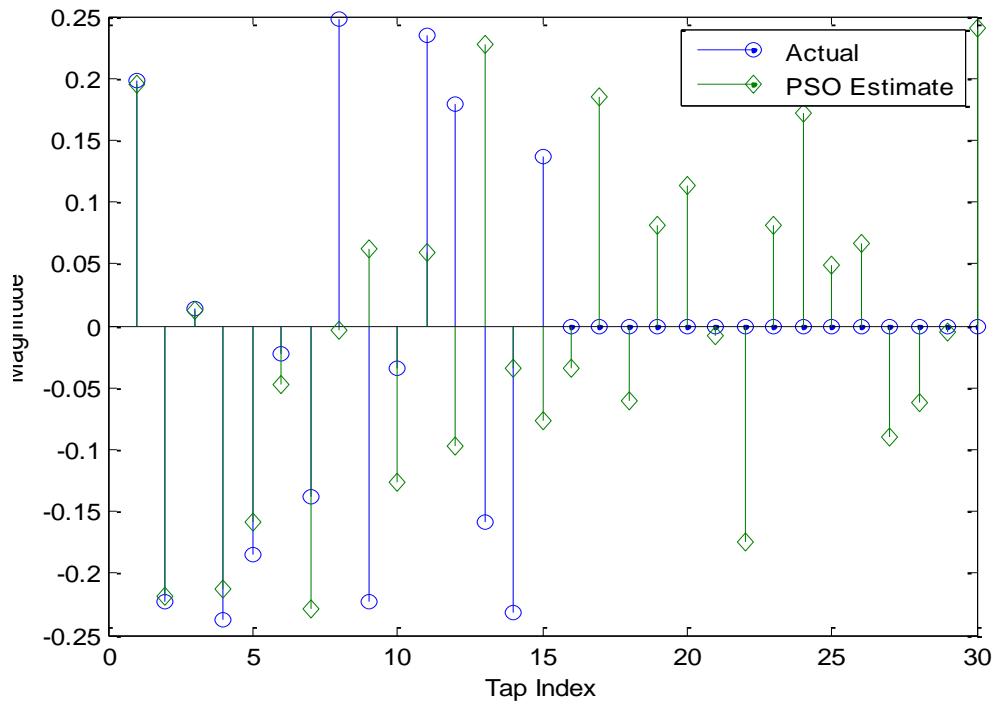


Figure 4.4 Actual and PSO-estimated coefficients of an oversized model

The above leads to a situation in which we obtain an estimated set of coefficients that produces an output signal which is very close to the desired one, but is longer than the actual coefficient vector. This has practical implications since we would ultimately like to implement our estimators on Field Programmable Gate Arrays (FPGAs) or use them to estimate digital pre-distorters (DPDs) to compensate for PA nonlinearities. In such applications, having a larger number of coefficients would require additional computational time and more operations to implement and thus we would like to have as short a coefficient vector as possible, motivating the investigation of PSO techniques that can perform the function of model-dimension estimation.

4.4 Proposed PSO techniques

4.4.1 Cluster-based PSO (C-PSO)

Since PSO is a population-based technique that uses a large number of particles to solve an optimization problem, this work investigated the feasibility of utilizing this feature to estimate the size of an oversized MP model by emulating the parameter-sweeping procedure a designer would use to find the correct dimension, albeit in a parallel manner. The objective in this approach is two-fold: finding the best NMSE performance using the smallest possible model.

In this approach, the swarm is divided into C groups (referred to as *clusters*) of particles and each cluster is made to test a different pair of (l, k) . This is carried out by forcing some of the entries of a cluster's position vectors to be zero and only updating those which correspond to the chosen (l, k) pair.

The updating of the positions of particles is done in the same manner as the traditional PSO algorithm with the **gbest** used being that of the cluster a particle belongs to, meaning that the behavior of a particle belonging to a cluster is isolated from that of particles in other clusters. The reason for such isolation is perhaps evident, as each cluster occupies its own 'corner' of the search space, defined by the (l, k) pair it is assigned to and hence, the particles of a cluster need to be confined within that part of the search space.

After assigning to each cluster an (l, k) pair, initializing the positions of its particles and computing their fitness functions, the algorithm then enters its iterative stage.

This stage proceeds similarly to that of the familiar PSO except for one point: here, a cluster is given a grace period of a few iterations in which it is left to operate normally. After this period, the cluster's **gbest** is then compared with those of the other clusters to decide whether it should be allowed to continue its search or whether it should be disbanded. To better improve the speed of convergence, the members of a disbanded cluster are distributed between the remaining clusters or can simply be removed from the swarm if they are not needed (e.g. if the targeted performance had already been achieved by another cluster).

The flow of this version of PSO is illustrated in Figure 4.5, with the index of the clusters being denoted by c .

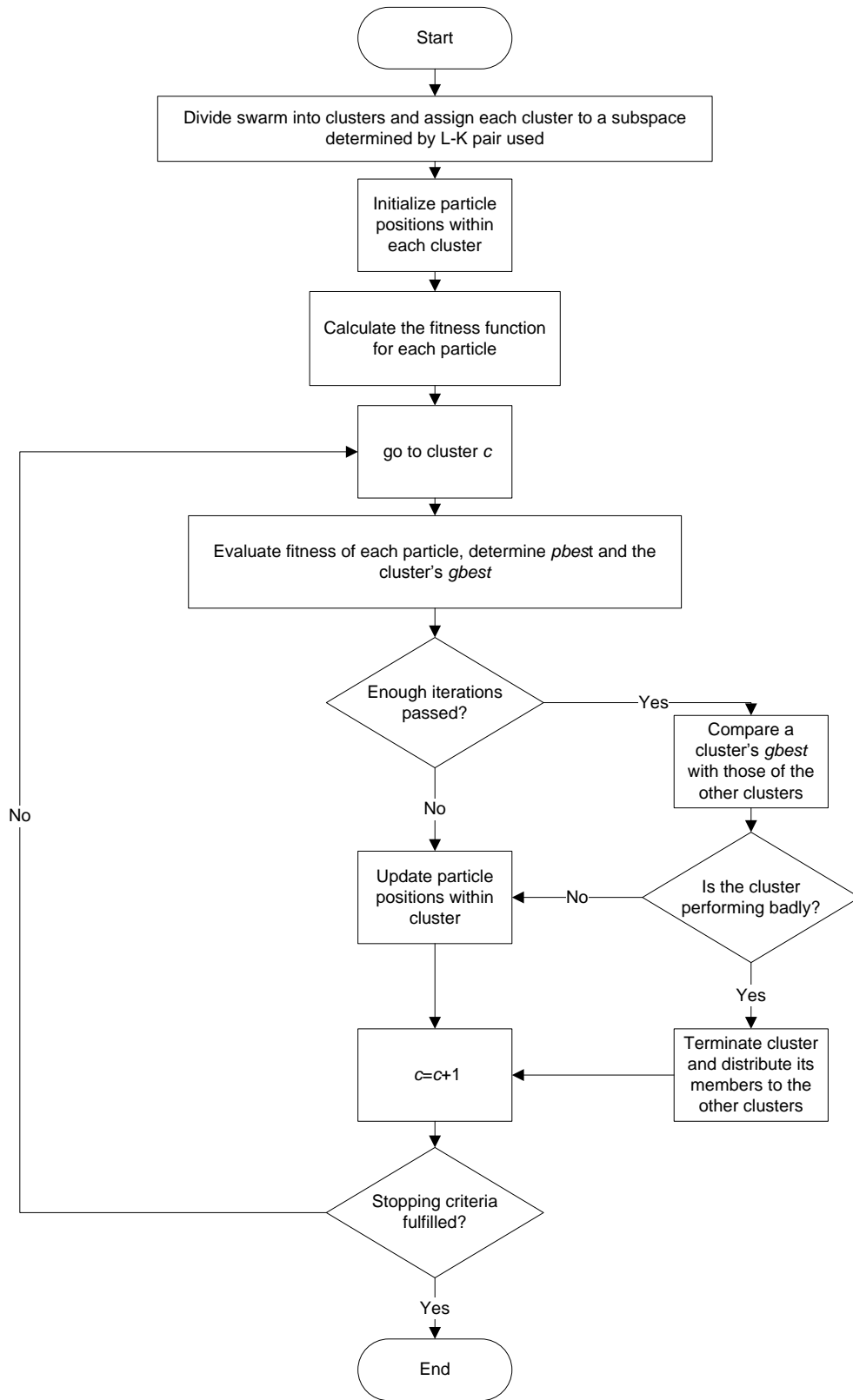


Figure 4.5 Flow of the cluster-based PSO algorithm

As can be inferred from the above discussion, C-PSO's performance depends on a number of factors in addition to those affecting a traditional PSO implementation. These factors are:

- i. Cluster size: The number of particles allocated to each cluster has an important effect on overall performance, since each cluster is essentially a PSO implementation of its own. One way to reduce the size of the clusters needed is to quickly disband swarms of unfavorable performance after a few iterations and distributing their members.
- ii. Grace period: the amount of time a cluster is allowed to operate before a decision is made on whether it should be kept or discarded directly affects convergence speed and overall performance since allowing clusters of subpar performance to continue functioning denies access to particles which could otherwise be better utilized by assigning them to other clusters instead.
- iii. Method of handling the members of disbanded swarms: the allocation can be either *blind*, *preferential* or *proportional*. In blind allocation, the particles freed up by the termination of their cluster are distributed evenly among the remaining clusters whereas preferential clustering assigns all of them to the best-performing cluster. Weighted clustering attempts to achieve a middle ground between the two by distributing particles in proportion to a cluster's performance. This can either be directly proportional to concentrate resources into improving the performance of 'good' clusters, or inversely proportional to give lagging clusters a second chance of sorts for a few iterations. The simulations performed indicate

that directly proportional allocation is the preferable approach, as would be expected.

The main shortcomings of this method are as follows:

- i. Large swarm size: Since each cluster is an independent implementation of PSO, implementation of this method would require using swarms of much larger size than what is usually required by the other methods. To somewhat mitigate this, a sub-optimal cluster size can be used initially with a shortened grace period to quickly eliminate 'bad' clusters and transfer their particles elsewhere, creating a 'dummy' initial phase, noting that the choice of the length of this phase can affect the algorithm's performance.
- ii. The need for additional tweaking: since there are more factors impacting the performance of this approach than in the case of the other PSO algorithms, more attention needs to be paid to the fine-tuning of this algorithm.

As direct implementation of this method was found to require swarms of extremely large sizes (> 700 particles), it was abandoned in favor of a simpler technique combining the clustering method with the use of the l_0 norm, as discussed next.

4.4.2 l_0 -Penalized PSO

In this group of PSO algorithms, the cost function to be optimized by PSO is modified to reward (or penalize) the particle based on its dimensionality, measured by the number of nonzero elements it has. The modified cost function is expressed as

$$J(n) = \frac{\|e(n)\|^2}{\|d(n)\|^2} + a(f(\|\mathbf{p}_i\|_0)) \quad (5.4)$$

where $f(\|\mathbf{p}_i\|_0)$ is some function of the zero-norm of a particle's position vector and a is some constant. The zero-norm is defined as

$$\|\mathbf{p}_i\|_0 = \sum_{l=1}^D |\mathbf{p}_i(l)|^0 \quad (5.5)$$

where D is the dimension of a particle's position vector. From this definition, we see that the zero norm essentially counts the number of non-zero elements in a vector. Using this feature, the fitness function is now modified through the second term to reward the particles of smaller dimensions by endowing their fitness functions with a 'bonus' that grows in value the more zeros a particle has in its position vector, with the size of this bonus being controlled by the designer-set parameter a .

In this study, the best performance was obtained when using a function $f(\|\mathbf{p}_i\|_0)$ of the form

$$f(\|\mathbf{p}_i\|_0) = 10 \log_{10} \left(\frac{\|\mathbf{p}_i\|_0}{D} \right) \quad (5.6)$$

The cost function in (5.4) now modifies the behavior of PSO by making the particles try to find the solution having the highest possible number of zeroes within the search space that fulfills the NMSE criteria. After finding the shortest possible solution among the available candidates and choosing said position to be the new **gbest**, the positions of the zeroes in that vector are checked to see if they have been 'off' for more than one iteration

before forcing the corresponding coordinates of the remaining particles to zero in subsequent evaluations and the algorithm continues its operation as normal. The implementation of the proposed PSO algorithm proceeds in two stages: The implementation of the proposed PSO algorithm is carried out over two stages: The initial exploration and clustering stage, and the dimension-finding stage.

1. Initial exploration and clustering stage:

In the initial exploration stage, the particles are randomly distributed in a dispersed manner and every particle is allowed to freely roam the search space, without it getting drawn to the global best position. This is accomplished by setting the parameters c and α to low values without picking them as zero, which eliminates the effect of the global best and the zero-norm condition on the algorithm behavior. This turns the algorithm into a parallel optimizer that does not utilize social interaction.

In this stage, the trends among particles are monitored and if any clustering is observed, the locations around which the particles converge are recorded.

Following the initial exploration stage, the social functions of the swarm are gradually restored by progressively increasing c for a few iterations without completely centralizing the swarm in order to allow for some diversity. In this phase, if any particle finds a solution that is slightly inferior to that of the global best particle, its position is saved. Additionally, promising search directions are considered and some particles are allocated to search over them. This stage continues for N_i iterations in total. In this study, 4 iterations were found to be sufficient for this phase.

2. Dimension-finding stage:

In this stage, the zero-norm condition is activated and the multiple competing solutions are sorted in terms of the error they produce. If a solution among that set is found to have a lower number of significant coefficients while being close enough in terms of NMSE, it is chosen as a viable candidate for being a member of the 'global' best group. Each particle then picks the global best it pursues from the group based on its distance from it and searches only in the coordinates not equal to zero. In [86], it was suggested that carrying out the picking process at random could provide better results while maintaining diversity in the search process. In this work, closest-neighbor based selection was found to improve the dimension-estimation capabilities of the algorithm.

An additional aspect of this stage is that if the swarm settles on a global best and exhibits stagnation, some particles are retained to search within the vicinity of this solution while the remaining members of the swarm are re-scattered throughout the search space to find a better solution if one exists. This modification ensures that local minima are avoided and that alternative solutions can be investigated.

The steps involved in the implementation of the proposed PSO algorithm are summarized in Figure 4.6.

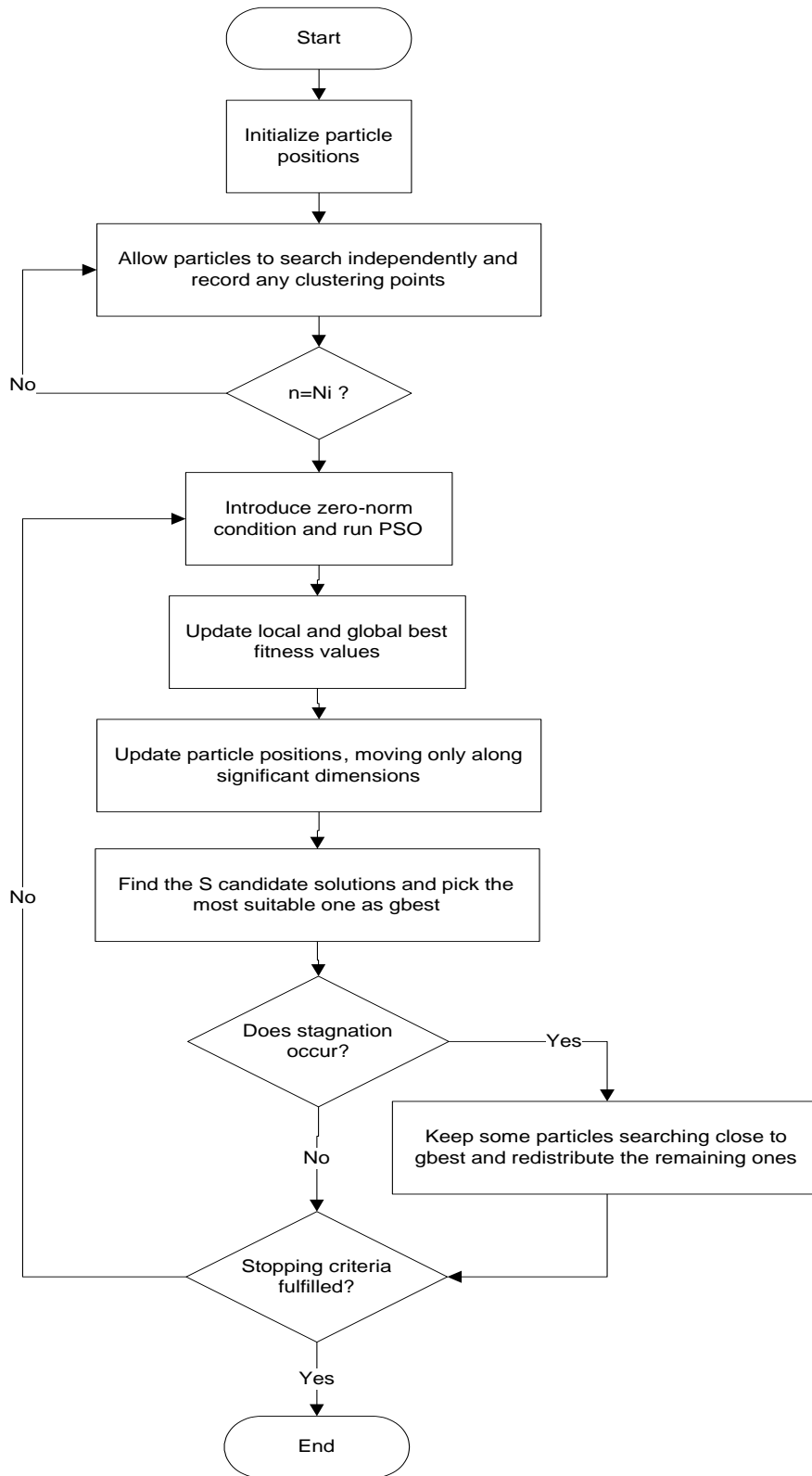


Figure 4.6 Flow of the l_0 -PSO algorithm

The balancing between the two requirements of minimizing the NMSE using the shortest possible vector is governed by the parameter a . In intuitive terms, one can think of a as an indicator of how much importance one places on the dimension of the coefficient vector compared to the accuracy of the estimation. The effect of changing the value of this parameter is studied further on

Effectively, the use of the l_0 norm creates a *zero-attractor* which 'draws' the values of the coefficients to zero; with its effect being more pronounced on taps of lower values. This idea of using the l_0 -norm has been implemented for adaptive filtering algorithms in [82] to improve their performance when estimating systems which are known to have zeroes.

The minimization of the l_0 -norm in (5.4) is challenging when attempted using traditional optimization techniques; as it represents what is known as a Non-Polynomial Hard problem. However since PSO techniques are independent of the cost function used, they can be used to solve this problem.

The use of the cost function in (5.4) was found to enable PSO to reduce the number of nonzero coefficients for an oversized system, as indicated by the results presented later in this chapter. The significance of this is that it allows us to know which coefficients are necessary and which aren't; which in turn saves us a large number of operations when carrying out an implementation.

To give consideration to alternative approaches, a PSO algorithm inspired by the recent developments in the area of Compressed Sensing [87][88] is proposed next.

4.4.3 l_1 -Penalized PSO

Building on the findings in compressed sensing theory that minimization of the l_1 norm can be utilized to improve system-identification when the models are oversized [89], a version of the PSO algorithm incorporating this norm into its cost function is developed. In this algorithm, the cost function becomes

$$J(n) = \frac{\|e(n)\|^2}{\|d(n)\|^2} + a(\|\mathbf{p}_i\|_1) , \quad (5.7)$$

where $\|\mathbf{p}_i\|_1$ is the l_1 norm of the vector \mathbf{p}_i defined as

$$\|\mathbf{p}_i\|_1 = \sum_{j=1}^D |p_i(j)| . \quad (5.9)$$

Minimizing the l_1 norm has the advantage of being simple to implement; since all we need to do is to include the norm into the cost function minimized by PSO and zero-forcing the least significant taps. In return for this simplicity, however, this version of PSO is incapable of estimating the actual dimensions of a model since it does not utilize the model structure in any way. This can be observed from Figure 4.7, which shows the coefficients found by this algorithm when estimating the same PA in earlier experiments and utilizing an oversized model of dimensions $L=5, K=6$ when the actual size is $L=3, K=5$. By examining the figure, it can be seen that this algorithm finds a coefficient vector which does not obey any particular structure. Similarly to the previous proposed algorithm, VCF-PSO is used as the basis for this algorithm.

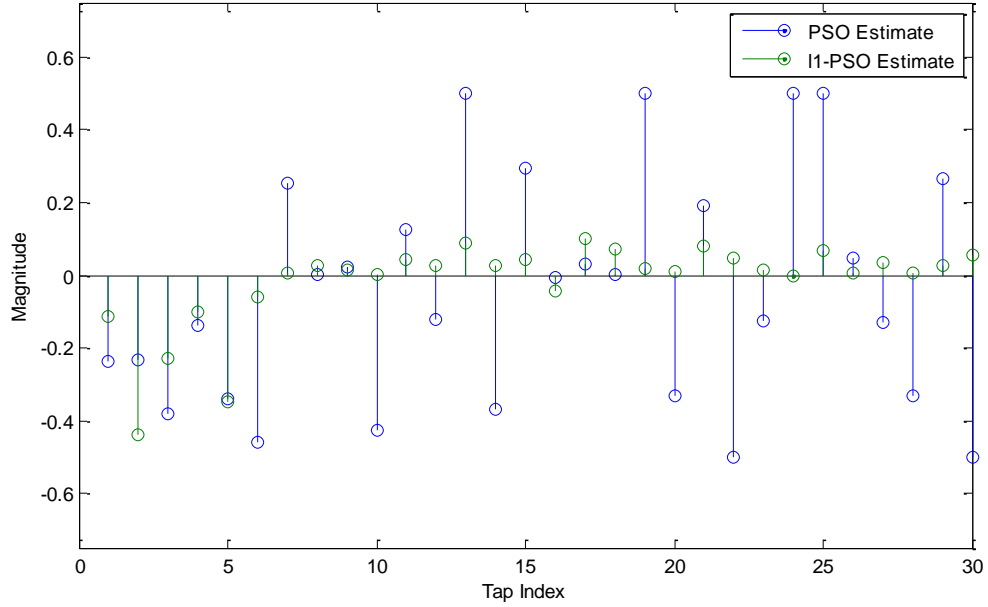


Figure 4.7 Actual and l_1 PSO-estimated coefficients of an oversized model

4.5 Experimental Validation of Proposed Algorithms

To validate the performance of the proposed predistorter, a nonlinear Doherty PA was modeled using an MP model known to have a 'correct' dimension of $L = 3, K = 5$. This device was then modeled using oversized MPM models of varying dimensions to account for the possible oversizing scenarios and the coefficients of the oversized models were subsequently extracted using the proposed PSO technique. Four scenarios were considered in this study:

1. Correctly-sized case: with $L = 3, K = 5$.
2. K overdetermined by 3: In this instance, K is chosen as 8 and L fixed at 3.
3. Both L and K overdetermined: As an extreme scenario, both L and K were overdetermined, leading to the choice of $L = 5$ and $K = 8$.

In all experiments, the swarm size was fixed to 200 particles and the identification experiments were run for 100 times independently on MATLAB R2010a using an Intel Core i7 CPU, 1.73GHz computer.

Prior to performing the experiments, however, an extensive sensitivity analysis was carried out to determine the effect of each of the parameters used in the proposed PSO algorithms on performance, with the results presented in the next section.

4.6 Sensitivity analysis of the PSO algorithms

To choose the best combinations of parameters for each algorithm, an extensive sensitivity analysis was performed and the results were documented in this section. The analysis consists of sweeping the swarm size, inertia weight, social and local parameters, in addition to the common parameter a and the parameters unique to each variant.

To avoid repetition, the results of the sensitivity analysis for the proposed algorithms were plotted, whereas the parameters selected for the traditional PSO algorithms are given numerically in Table 4.2.

For all of the algorithms tested (whether traditional or l_0 -penalized), a swarm size of 200 for the algorithms produced the best NMSE results. As for the number of iterations to be observed, it was found that 20 iterations were sufficient to study the behavior of the various algorithms, as all of them reached their steady-state within this window.

Table 4.2 Parameter Selection for the traditional PSO Algorithms

Algorithm	Swarm Size	Inertia Weight	b	c	k_{\min}, k_{\max}
PSO	200	1.1	4	2	N/A
VCF-PSO	200	N/A	4	4	4, 6

4.6.1 Sensitivity analysis for the l_0 -VCFPSO algorithm

A sensitivity analysis for the parameters of the l_0 -VCFPSO algorithm was carried out, with the results as given in Figure 4.8 through Figure 4.11 and numerically in Table 4.3.

Table 4.3 Parameter Selection for the l_0 -VCFPSO Algorithm

Swarm Size	b	c	k_{\min}, k_{\max}	a
200	4	4	4,6	0.75

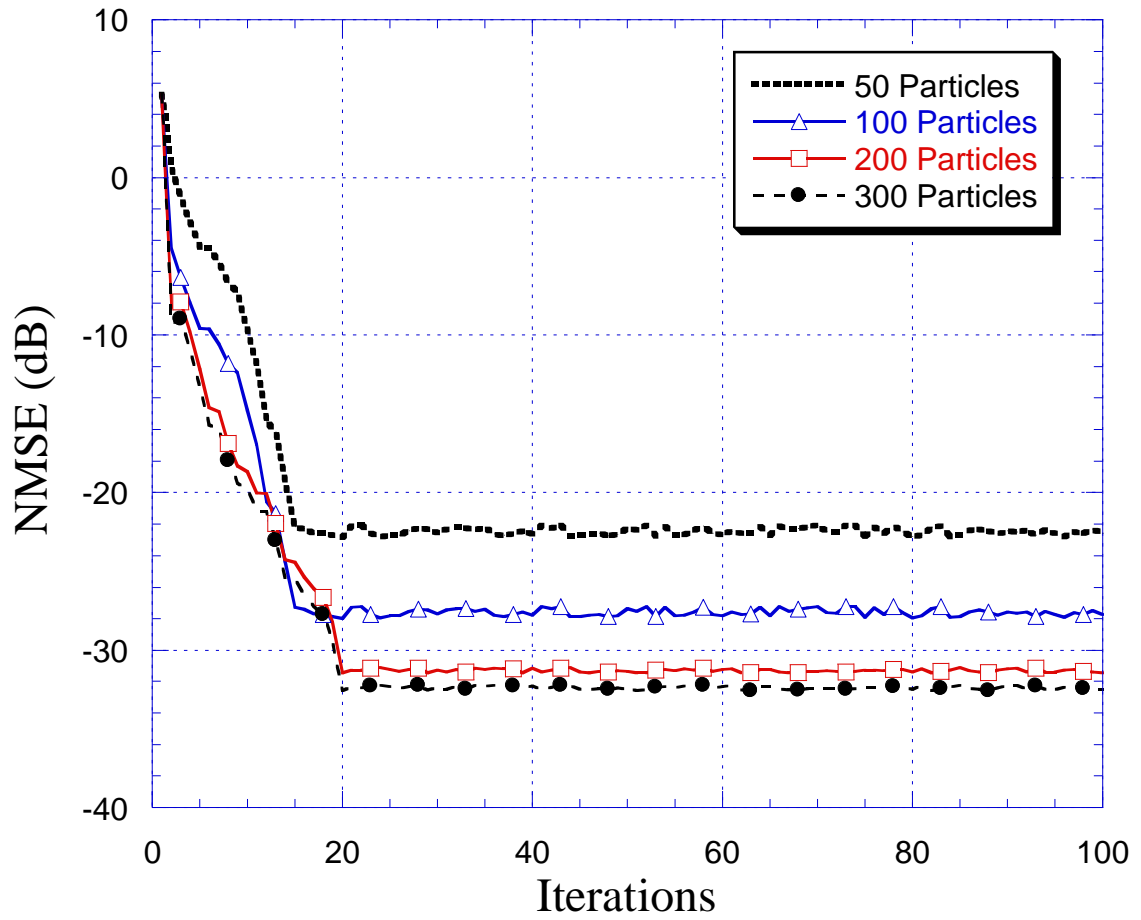


Figure 4.8 Effect of the swarm size on the performance of l_0 -VCFPSO

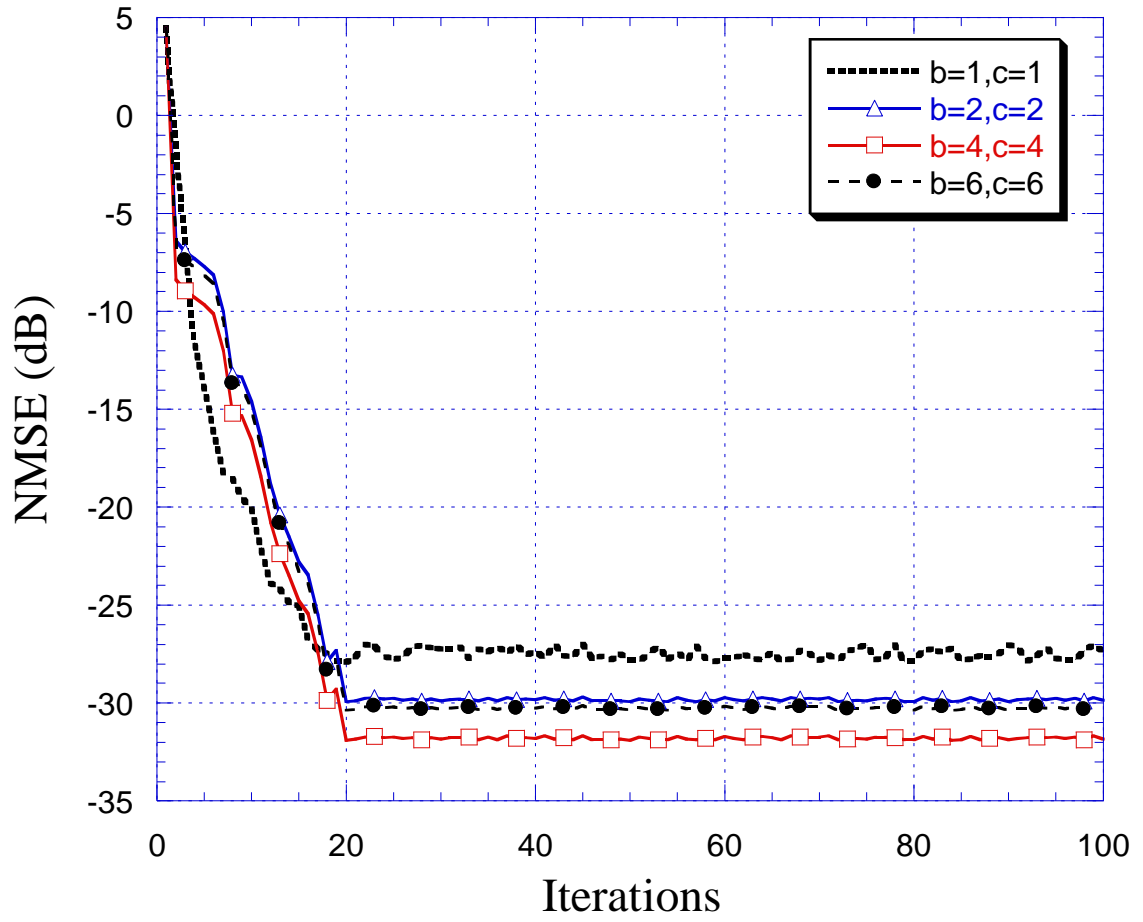


Figure 4.9 Effect of the choice of the parameters b, c on the performance of l_0 -VCFPSO

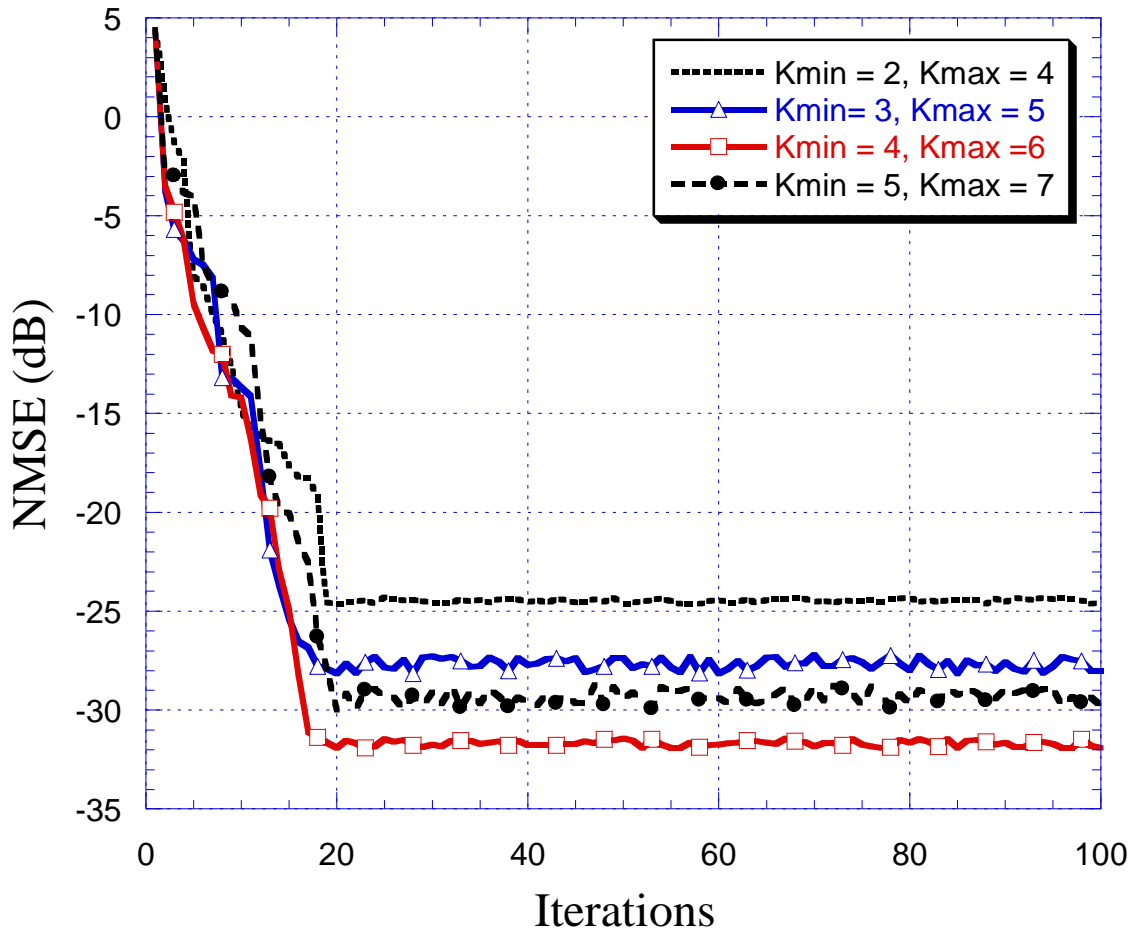


Figure 4.10 Effect of the choice of the parameters k_{\min}, k_{\max} on the performance of l_0 -VCFPSO

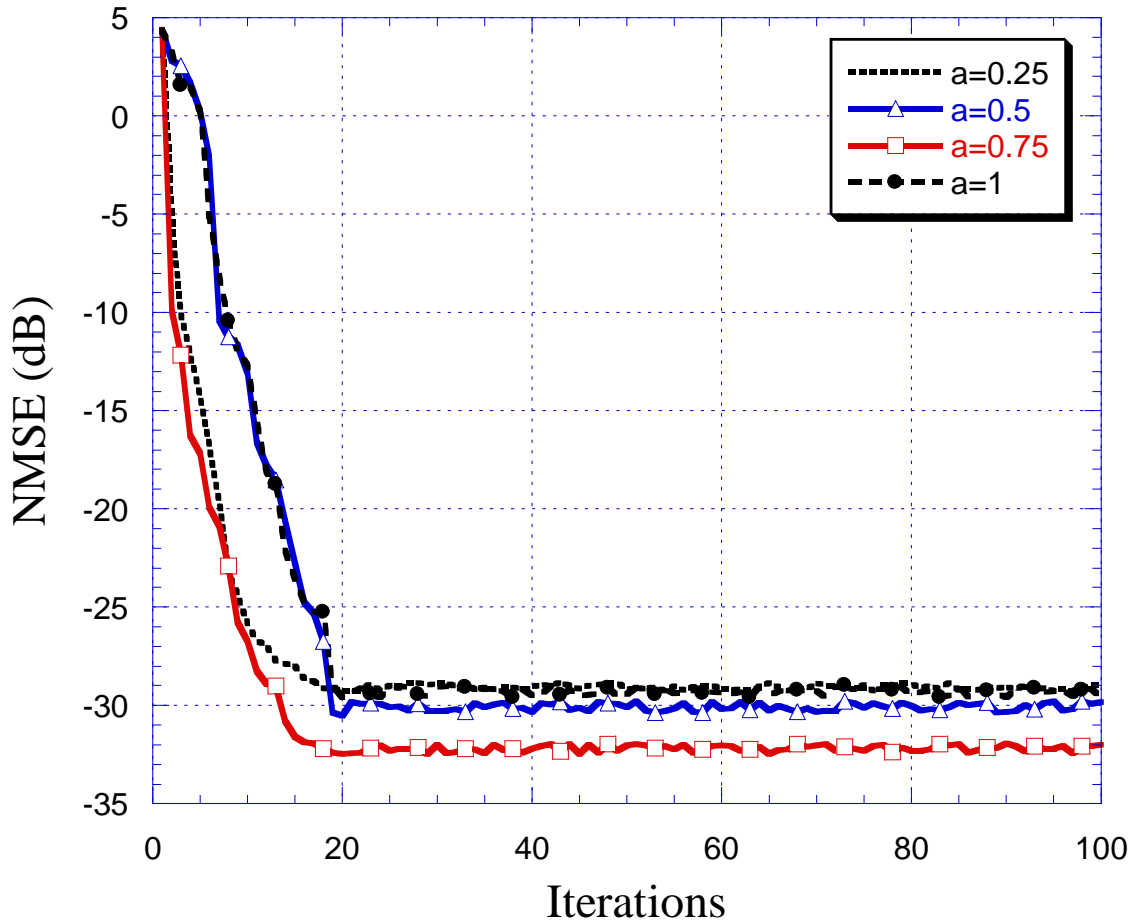


Figure 4.11 Effect of the choice of the parameter a on the performance of l_0 -VCFPSO

4.6.2 Sensitivity analysis for the l_1 -VCFPSO algorithm

Similarly, a sensitivity analysis for the parameters of the l_1 -VCFPSO algorithm was carried out, with the results as given in Table 4.4. As only the results for sweeping the swarm size and the penalty parameter are distinct from those of the previous algorithm, plots for these two parameters are given with the rest omitted for brevity.

Table 4.4 Parameter Selection for the l_1 -VCFPSO Algorithm

Swarm Size	b	c	k_{\min}, k_{\max}	a
200	4	4	4,6	0.5

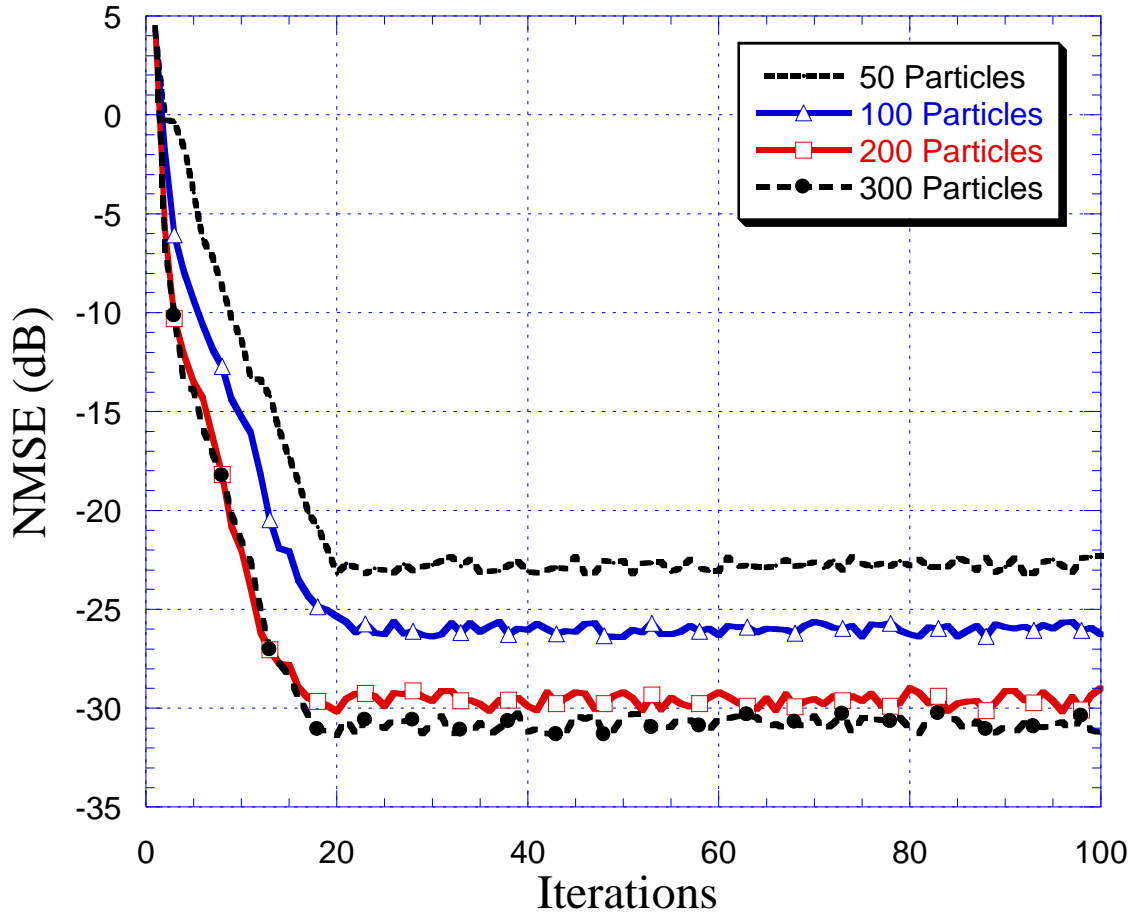


Figure 4.12 Effect of the swarm size on the performance of l_1 -VCFPSO

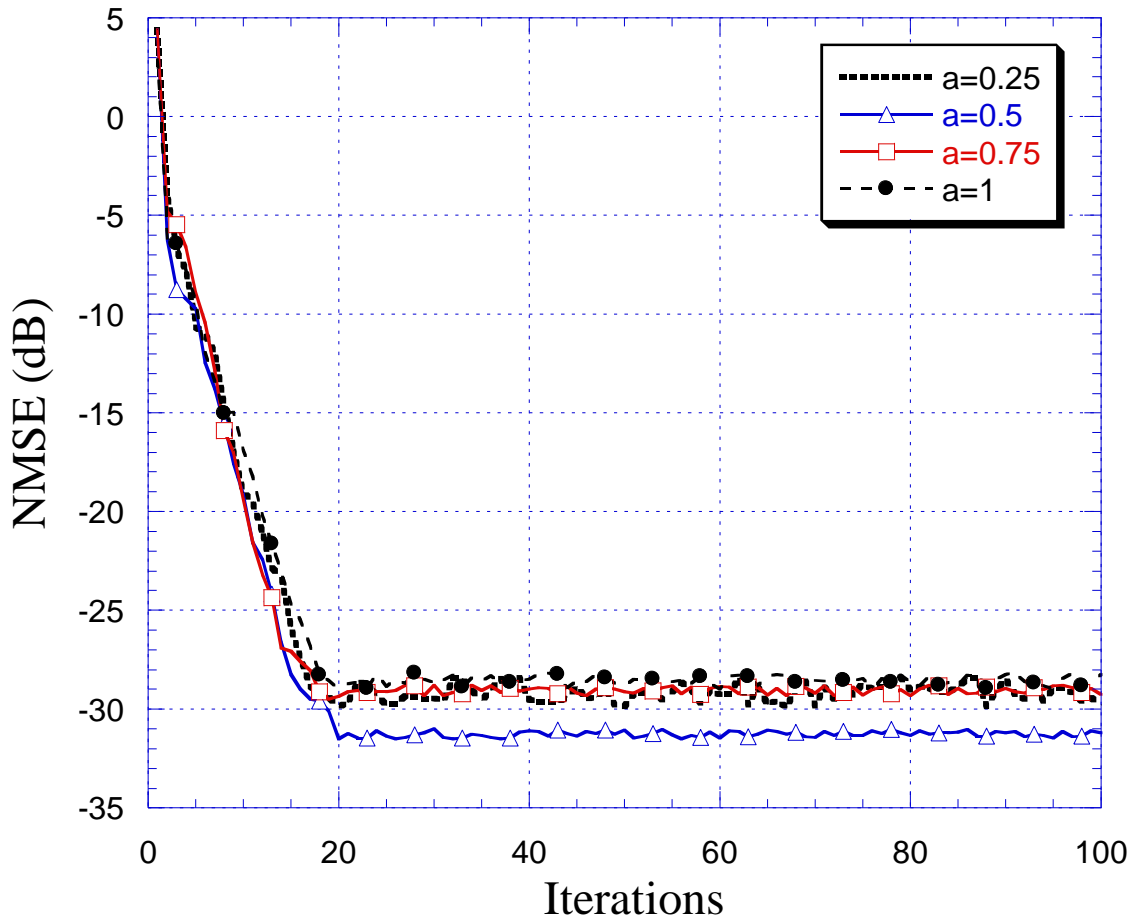


Figure 4.13 Effect of the choice of the parameter a on the performance of l_1 -VCFPSO

4.7 Simulation Results

The two PA identification experiments described earlier were performed using the traditional and proposed PSO algorithms and their results were collected and plotted.

4.7.1 Results for experiment A: Correctly-Sized Model

In this experiment, an MP model of dimensionality 15 was estimated by deploying the PSO algorithms to find a coefficient vector of the same length. The traditional and proposed norm-penalized PSO algorithms were compared in terms of NMSE, how fast they arrived at their best estimate, the amount of time they each require to complete the

computation process, the number of nonzero coefficients in their estimates and DPD performance.

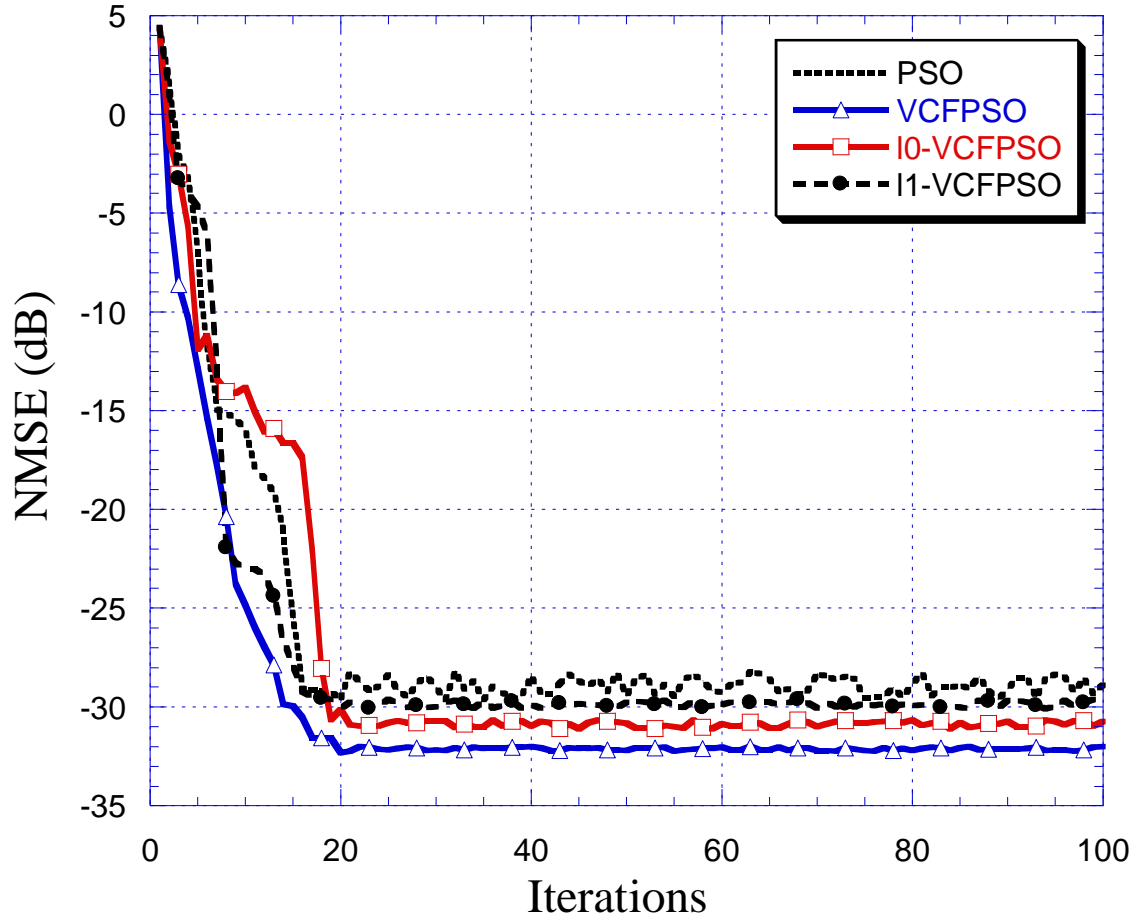


Figure 4.14 Learning curves for the PSO algorithms when estimating a correctly-sized model

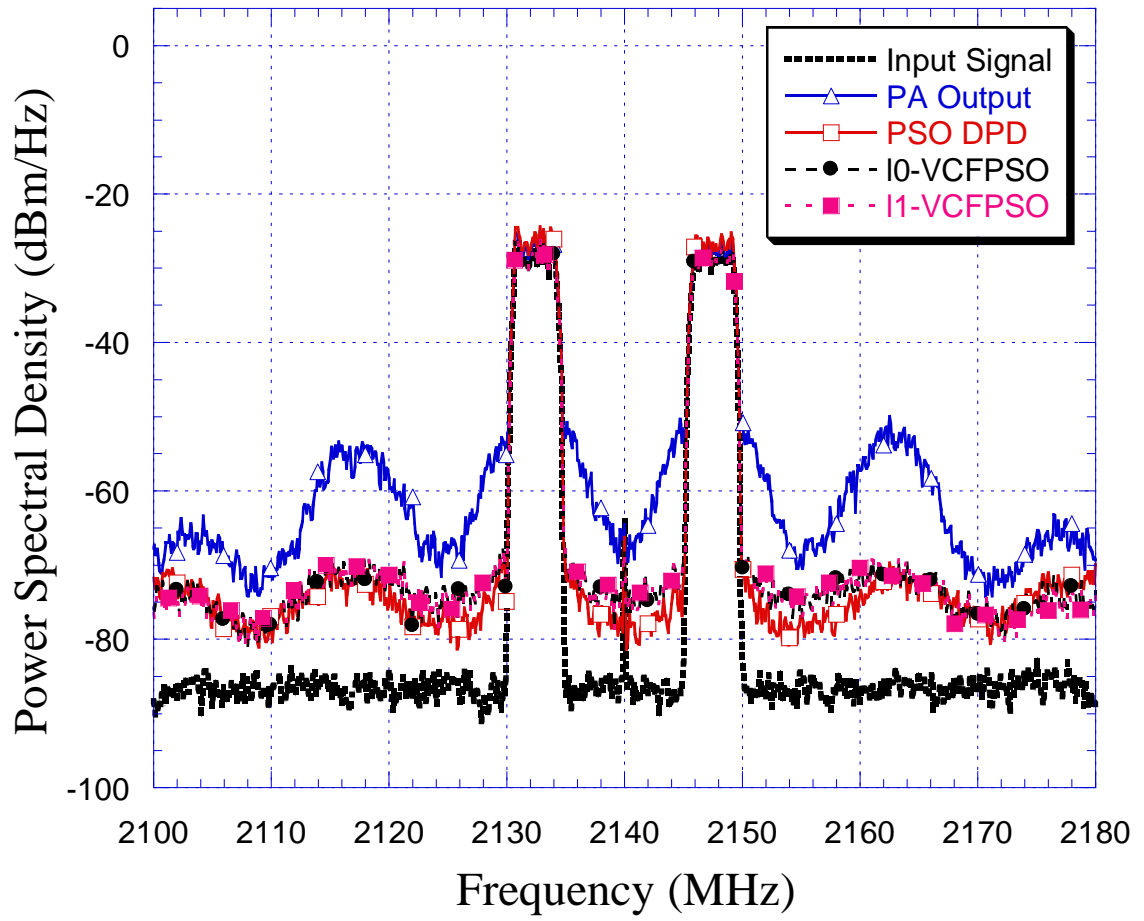


Figure 4.15 DPD performance of the PSO algorithms

Table 4.5 Numerical results for the first identification experiment

Algorithm	NMSE (dB)	NAMSE (dB)	Model Dimensions	
			L	K
PSO	-28.25	-13.09	3	5
VCF-PSO	-32.82	-13.28	3	5
l_0 -VCFPSO	-31.08	-13.09	3	5
l_1 -VCFPSO	-29.94	-13.80	3	5

Examining Figure 4.14 and Table 4.5 above, we see that the available and proposed algorithms are close in terms of performance. This is somewhat expected, as the proposed algorithms are designed to take advantage of the oversizing of the model. Similar conclusions can be made by examining the frequency-domain results in Figure 4.15.

4.7.2 Results for experiment B : Oversized model with $L=3, K=8$

In this instance, the PSO algorithms were used to identify an MP model with 24 coefficients ($L=3, K=8$). The numerical results of this experiment are recorded in Table 4.6 without the corresponding learning curves. Figure 4.16 displays the coefficients estimated by the traditional and proposed PSO algorithms. Examining this figure, it can be seen that l_0 -PSO finds a coefficient vector whose structure corresponds to a model of dimensions $L=3, K=5$; as the majority of the entries associated with $K > 5$ are populated with zero entries. Conversely, l_0 -PSO finds a coefficient vector that has coefficients of reduced magnitude without reducing the size of the model.

Table 4.6 Numerical results for the second identification experiment

Algorithm	NMSE (dB)	NAMSE (dB)	Model Dimensions	
			L	K
PSO	-27.05	-13.09	3	8
VCF-PSO	-31.31	-13.28	3	8
l_0 -VCFPSO	-31.66	-13.09	3	5
l_1 -VCFPSO	-33.94	-13.80	3	8

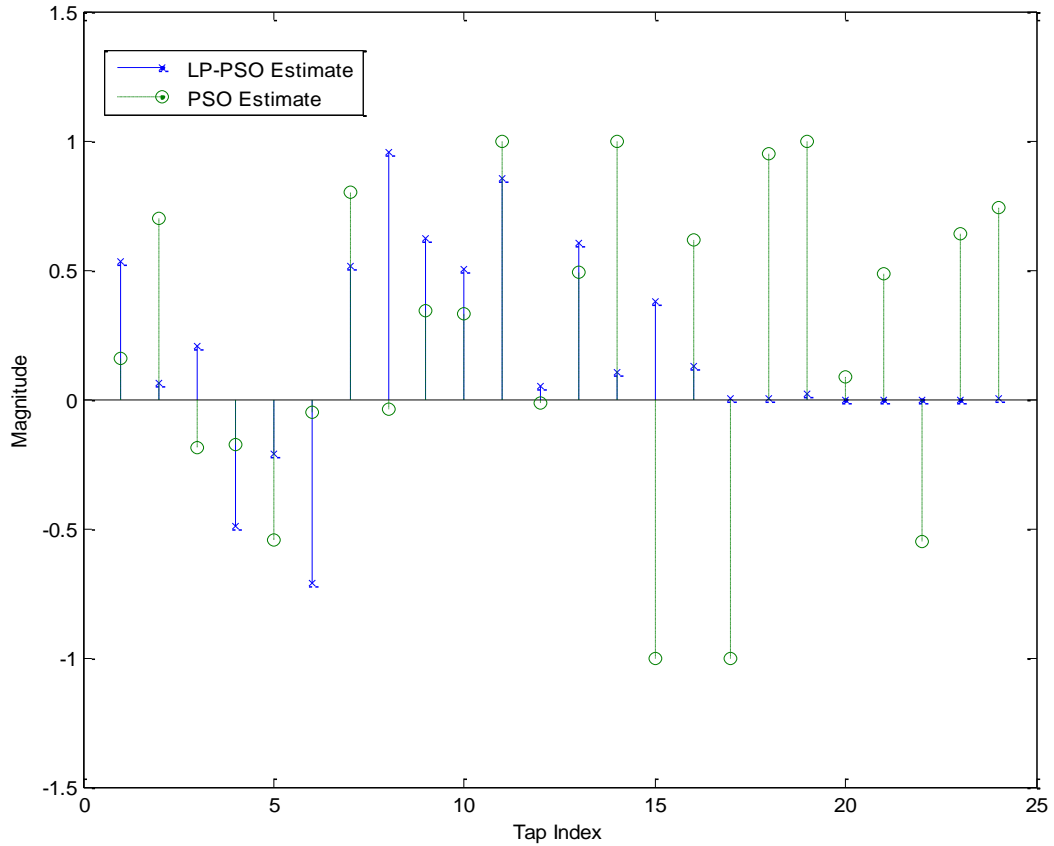


Figure 4.16 Coefficients estimated by PSO and l_0 -VCFPSO

4.7.3 Results for experiment C : Oversized model with $L=5, K=8$

In this case, the PSO algorithms were tasked with identifying an MP model of size 40 ($L=5, K=8$) used to describe the PA generating the same input-output pair of signals. From the learning curves plotted in Figure 4.17, the advantage of the proposed PSO algorithms in terms of NMSE and convergence behavior can now be seen.

In this case, the performance of the traditional PSO algorithms is degraded due to them effectively trying to estimate a 40-tap vector using swarms of 100 particles, which explains their behavior. The proposed algorithms, however, are not as sensitive to this

issue since they have a built-in capability to 'know' what the correct dimension is and are thus more robust in this situation.

Among the proposed algorithms, l_0 -VCFPSO had the best overall NMSE performance; achieving a steady-state NMSE of -31.94 in less time and a smaller number of iterations than the other algorithms. By comparison, l_1 -VCFPSO fails to achieve comparable performance, as its steady-state NMSE is -27.71 dB.

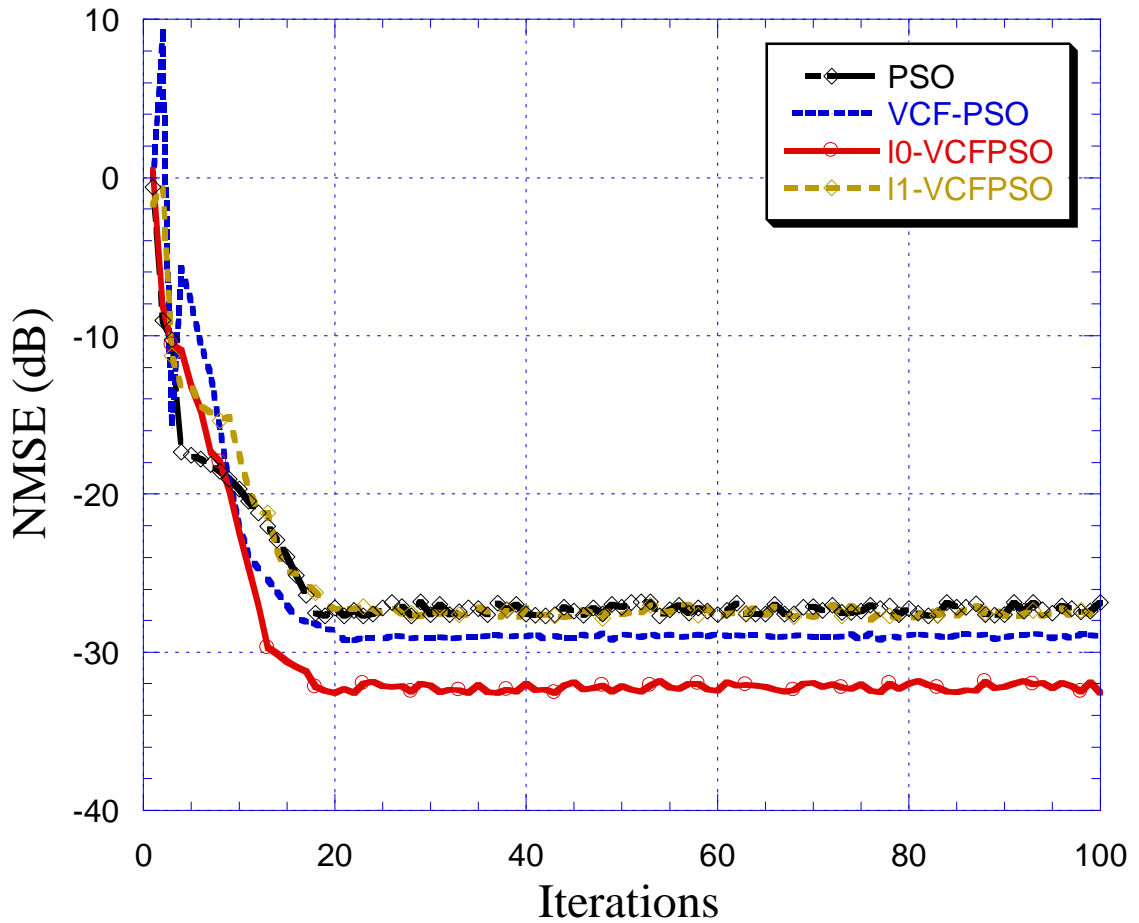


Figure 4.17 Learning curves for the adaptive algorithms and PSO variants for the oversized model

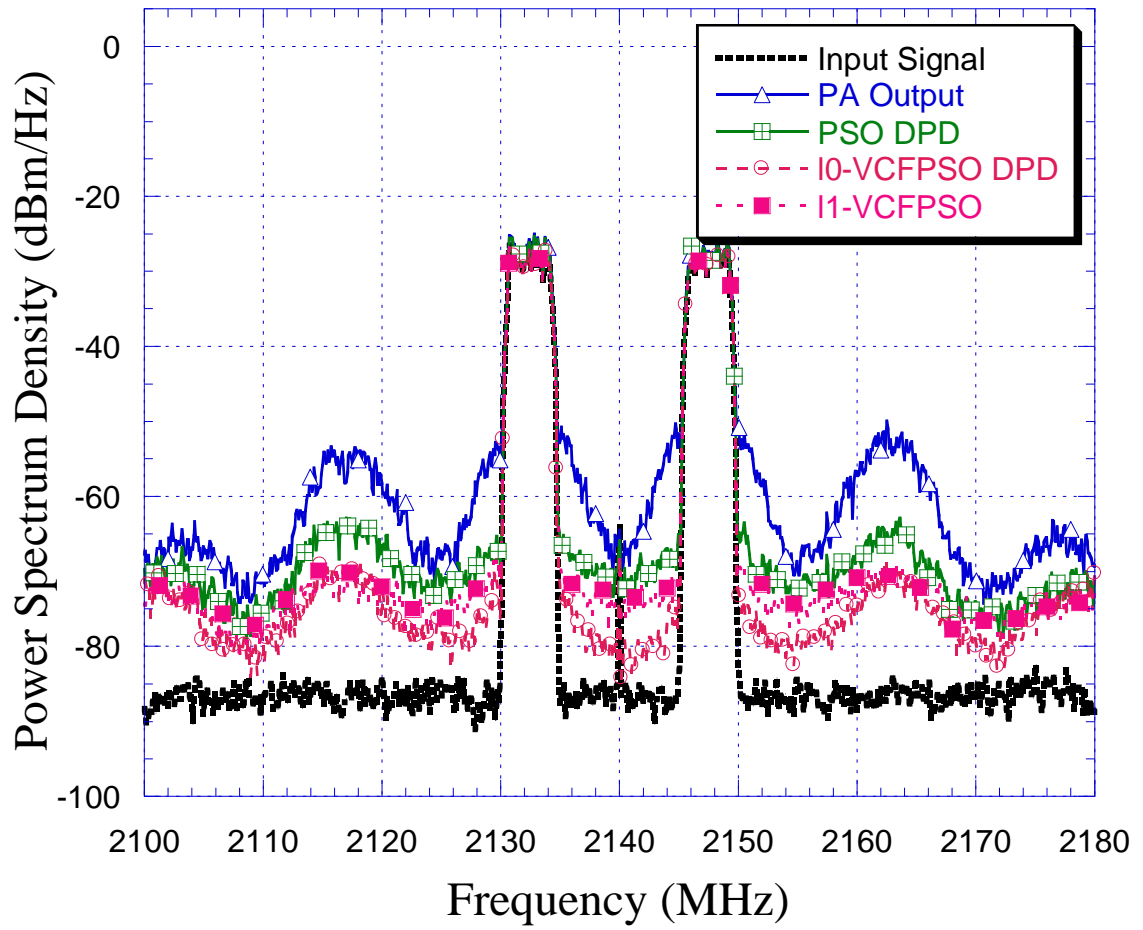


Figure 4.18 DPD performance of the PSO algorithms

Table 4.7 Numerical results for the third identification experiment

Algorithm	NMSE (dB)	NAMSE (dB)	Model Dimensions	
			L	K
PSO	-27.45	-12.59	5	8
PSO-VCF	-29.15	-13.38	5	8
l_0 -VCFPSO	-31.02	-13.99	4	6
l_1 -VCFPSO	-27.42	-14.19	3	8

When the PSO algorithms estimators were deployed to identify a heavily oversized MP , the value of the l_0 -PSO algorithms becomes more apparent. In this case, the l_0 -PSO algorithm outperforms the traditional algorithms by a margin of 2~3dB. This can be explained by the fact that l_0 -PSO 'guesses' the correct size of the coefficient vector and attempts to estimate a vector of that size, as opposed to using the full-length coefficient vector as the other algorithms do. This means that while the traditional PSO algorithms use 100 particles to find a solution of dimension 40, the proposed algorithms use the same number of particles to search a space of smaller size.

Figure 4.19 shows the coefficients estimated by the PSO and l_0 -PSO algorithms for the $L = 5, K = 8$ case. The results in this figure support the earlier findings that l_0 -PSO can closely guess the correct dimensions of an MP model, even when both parameters are oversized. In contrast, the l_1 -PSO performs quite poorly in terms of NMSE, suggesting that its tendency to uniformly 'compress' the magnitudes of all of the coefficient taps degrades its estimation accuracy.

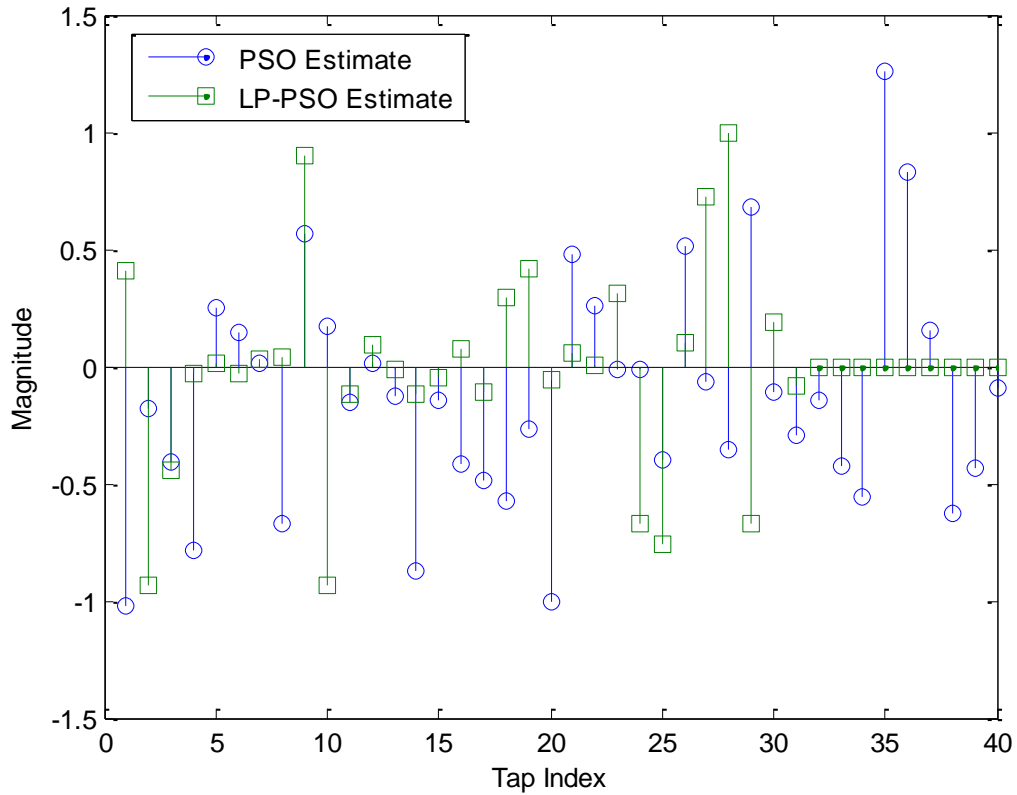


Figure 4.19 Coefficients estimated by PSO and l_0 -VCFPSO

Examining the pattern of the zero elements in Figure 4.19, it can be seen that l_0 -PSO estimates a model of size 4 by 5 as opposed to the regular PSO algorithm, which has a coefficient vector which is almost fully populated. In contrast, the coefficients estimated by l_1 -PSO are plotted in Figure 4.20. In this figure, it can be seen that the coefficients estimated here are not of a particular structure but also happen to have many low entries. However, it should be remembered that the use of this algorithm produces an estimation of lowered accuracy compared to the other algorithms (by a margin of 3~4 dB), so these coefficients do not represent a reliable estimate.

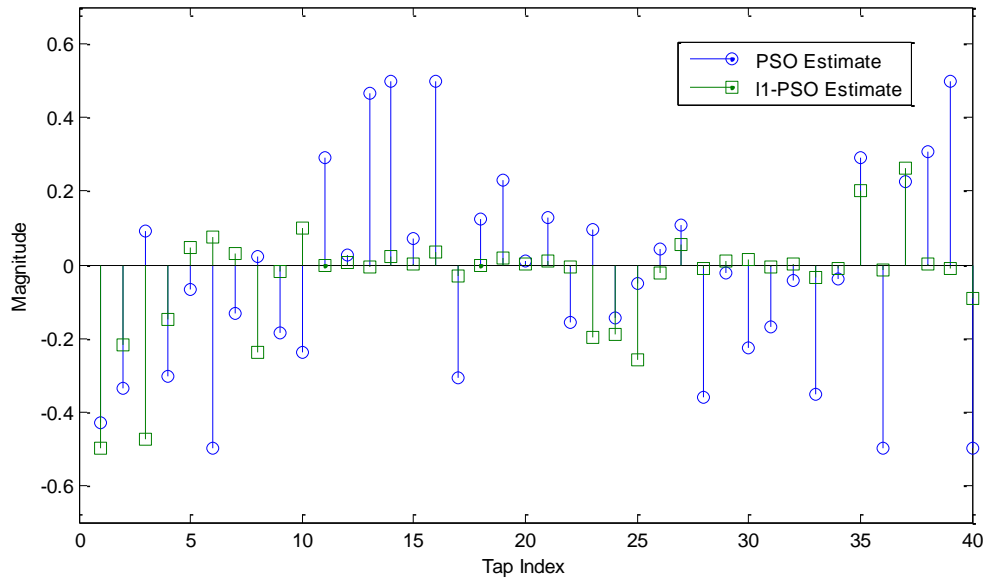


Figure 4.20 Actual and l_1 -VCFPSO-estimated coefficients of an oversized model

4.8 Summary of Results and Conclusions Reached

In this work, the use of particle swarm optimization in the area of nonlinear PA identification was investigated. After identifying the main issues facing the application of conventional adaptive filtering approaches, PSO and some of its variants were used and their performance was compared with that of the adaptive filters.

After obtaining initial results confirming that PSO techniques outperform adaptive filtering, their population-based nature was utilized to solve the problem of estimating a model's coefficients more efficiently when the model is oversized.

To solve this problem, a l_0 -penalized PSO algorithm was developed and its performance was extensively studied.

After performing sensitivity analyses on each of the algorithms used and extensively verifying the performance of the proposed algorithms in both estimating a correctly-sized model and an oversized one, it was found that the proposed techniques outperformed the algorithms available in the literature in both error performance and the number of significant coefficients, thus alleviating the burden of figuring out the correct dimension of the model and enabling a designer to focus on the design of the PA system itself.

CHAPTER 5

THESIS CONCLUSIONS AND FUTURE WORK

5.1 Summary of Work Done and Conclusions

In this thesis, the topic of identifying the parameters of nonlinear power amplifier behavioral models and predistorters using adaptive filtering and PSO techniques was thoroughly investigated; starting with surveying of behavioral modeling, then moving on to adaptive filtering and concluding with the study of the use of PSO techniques and proposing a novel PSO algorithms were designed for the efficient estimation of Memory Polynomial model coefficients.

Ultimately, the proposed PSO techniques were found to perform well when estimating the parameters of nonlinear PA models in the case of oversized models having zero entries, thus enabling a designer to begin with a rough estimate of model dimensions and obtain a good estimate of both the model's parameters and their its size.

5.2 Future Work

The results obtained for the proposed PSO algorithms suggest that there is potential for further research in the following areas:

- I. Utilizing information about the structure of nonlinear models in the development of enhanced l_0 -PSO variants by either including this information into the cost function used, or utilizing it as a constraint.
- II. Utilizing PSO to develop improved pre-distorters by solving multi-objective optimization problems, such as optimizing across the time- and frequency-domains.

In conclusion, the results obtained indicate that PSO and its variants have the potential to be of further value in the area of nonlinear PA design and predistortion due to their accuracy and ability to perform auxiliary functions such as guessing a model's dimensions.

APPENDIX A

POWER AMPLIFIER BASICS

6.1 Classes and Types of Power Amplifiers

Amplifiers are usually constructed with a complementing pair of transistors (e.g. CMOS), where each transistor operates in one half of an input signal while some amplifiers use the same transistor to pass both the positive and negative halves of an input signal (Figure 6.1). Power amplifiers are grouped into classes (A,B,..etc) based on the percentage of the time they operate in “on” mode.

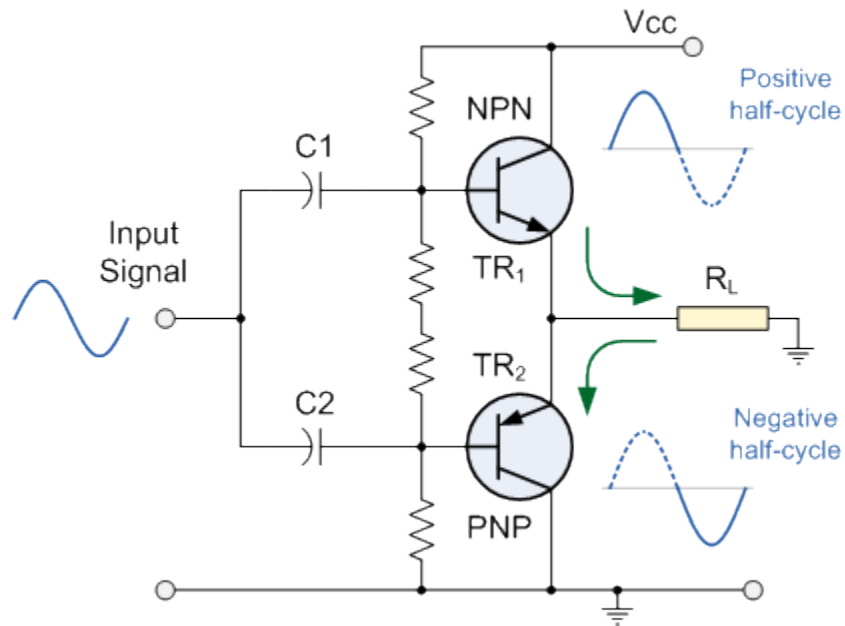


Figure 6.1 Typical Amplifier Circuit Configuration

6.1.1 Class A Amplifiers

A class-A amplifier is defined by always operating in “on” mode, continuously passing current. In this configuration, input signal magnitudes are kept small to avoid going into the cut-off region of the amplifier. Class A amplifiers are the most linear, yet the least efficient of the various classes, their efficiency being around 50% , limiting their usefulness.

The operating point of a class A amplifier is placed somewhere around the center of the load line, as can be seen in Figure 6.2 [13],[83].

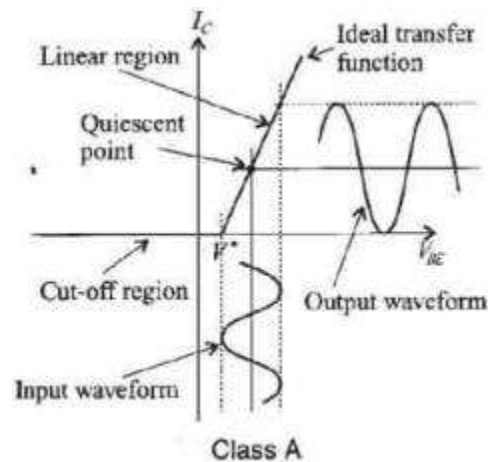


Figure 6.2 Class A Amplifier Operation[83].

6.1.2 Class B Amplifiers

Contrary to the class A amplifier, the class B is “on” for only half of the cycle of a sinusoidal input (i.e. it passes either the positive or negative part of the input signal). In return of the higher efficiencies (upwards of 75%) they provide, they produce harmonics and are much less linear in their behavior [83].

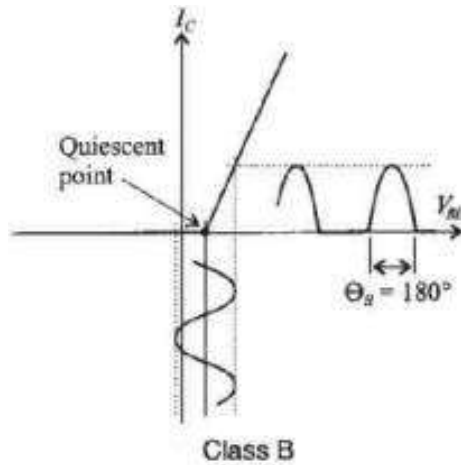


Figure 6.3 Class B Amplifier Operation[83].

6.1.3 Class AB Amplifiers

As its name indicates, amplifiers belonging to this class are a middle-of-the-road compromise between amplifiers of classes A and B [13].

The efficiency and conduction-time percentage of AB-class amplifiers is somewhere in between those of A and B-class amplifiers. The nonlinearities in this class are mainly due to saturation effects. Also, AB amplifiers have the most observable memory effects [83].

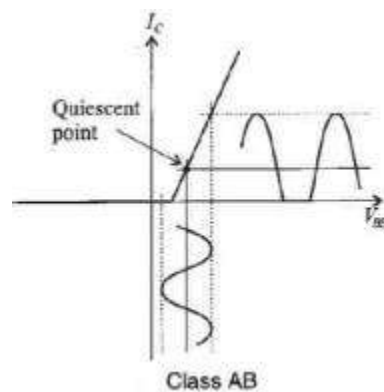


Figure 6.4 Class AB Amplifier Operation[83].

6.1.4 Class C Amplifiers

As the general trend thus far would lead us to expect, the Class C amplifier operates for under half the cycle time of an input signal, as shown in Figure 6.5 [13]. These amplifiers achieve high efficiencies at the cost of suffering from high distortion [83].

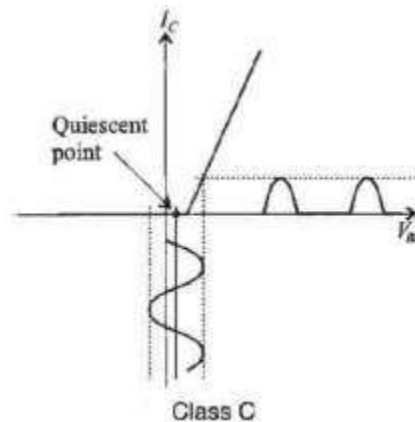


Figure 6.5 Class C Amplifier Operation[83].

6.1.5 Class D Amplifiers

Constructed from transistor-based switching circuits, a class-D amplifier can achieve an efficiency of 100% in theory but they remain unrealizable in real life, due to the existence of real-world effects such as switch resistances and parasitic capacitances [83].

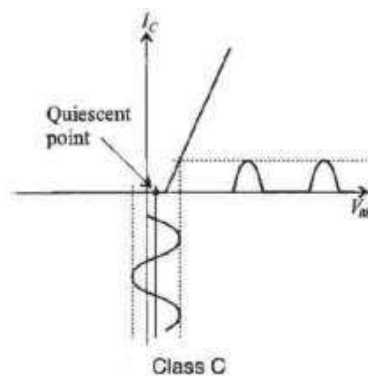


Figure 6.6 Class D Amplifier Operation[83].

6.1.6 The Doherty Amplifier

This popular amplifier, first invented by William H. Doherty in 1936, is built by modifying a class-B amplifier [84]. This amplifier is constructed by connecting two amplifiers as shown in Figure 6.7, with one of them being an AB-class amplifier and the other a class-C. The amplifiers are configured in such a way that the class-C amplifier operates when the input signal is high, and the AB amplifier comes into play when the input signal power falls below a certain threshold. This configuration results in high efficiency, making the Doherty amplifier a popular choice for most of the modern communication systems such as CDMA and LTE, with a number of variations on the design being available [85], [19]. The data sets used in this project were obtained from a Doherty amplifier.

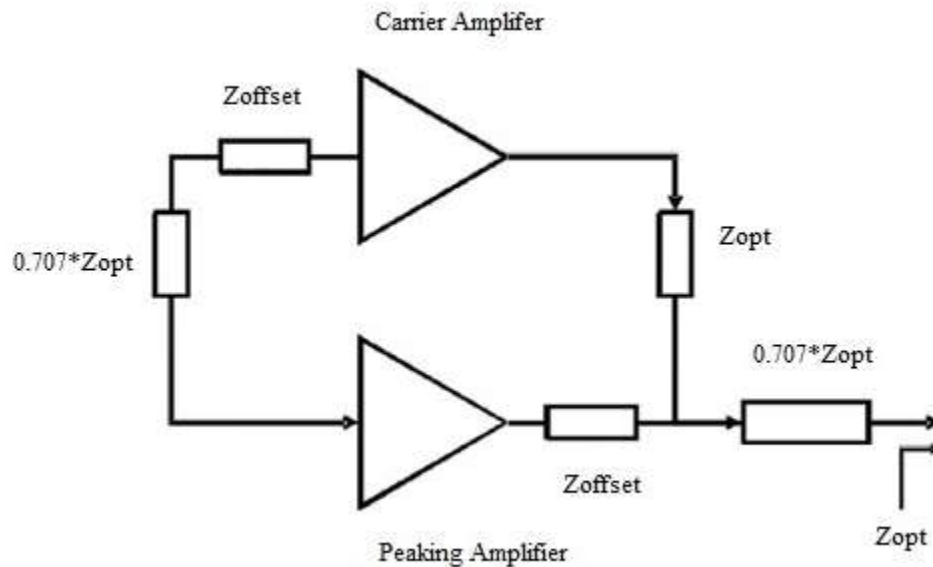


Figure 6.7 Typical configuration of a Doherty PA[85].

APPENDIX B

SINGULAR VALUE DECOMPOSITION AND THE METHOD OF LEAST SQUARES

7.1 Introduction

Singular value decomposition (SVD) is a well-known method which factorizes an $N \times M$ matrix \mathbf{A} in the following manner [50]

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* \quad (7.1)$$

\mathbf{U} is an $M \times M$ matrix, $\mathbf{\Sigma}$ is an $M \times N$ matrix with positive real numbers on the diagonal and \mathbf{V}^* is an $N \times N$ unitary matrix. It should be noted that for every \mathbf{A} , only $\mathbf{\Sigma}$ is unique. Another point to keep in mind is that \mathbf{U} and \mathbf{V}^* are composed of orthonormal columns

7.2 Application to System Identification

If we have a system whose input-output relationship can be written in the form

$$\mathbf{y} = \mathbf{A}\mathbf{h} \quad (7.2)$$

with \mathbf{A} being a non-square matrix of dimensions $N \times M$, the coefficient vector \mathbf{h} can be estimated by using the pseudo-inverse to compute them as follows [50]

$$\mathbf{h} = \mathbf{A}^+\mathbf{y} \quad (7.3)$$

The computation of the pseudo-inverse is carried out through SVD as follows

$$\mathbf{A}^+ = \mathbf{V}\Sigma^+\mathbf{U}^* \quad (7.4)$$

with Σ^+ being the pseudo-inverse of Σ . Since Σ is a rectangular diagonal matrix, its pseudo-inverse can be directly obtained by reciprocating the matrix and then transposing.

7.3 Limitations of Using SVD

The main limitation of using SVD in real time comes from the fact that SVD requires the full data matrix \mathbf{A} to be constructed before implementing it; meaning that SVD can only be used offline after the entire data set has been collected. This prevents its use in real-time predistortion systems where the identification procedure needs to be carried out often to track any changes in PA behavior.

References

- [1] F. M. Ghannouchi and O. Hammi, "Behavioral Modeling and Predistortion," *IEEE Microwave Magazine*, vol. 1527–3342, no. December, 2009.
- [2] G. K. Manne and T. Yao, "On the Predistortion Technique for Improving Transmission Linearity of OFDM System" in *Proc. 60th IEEE VTC*, Sept. 2004, pp. 3876–3879.
- [3] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Conf. Neural Networks*, Nov. 1995 pp. 1942–1948.
- [4] D. Bratton and J. Kennedy, "Defining a Standard for Particle Swarm Optimization," in *Proc. IEEE SIS*, Apr. 2007, pp. 120–127.
- [5] R. Poli, J. Kennedy, and T. Blackwell, "Particle Swarm Optimization," *Swarm Intelligence*, vol. 1, no. 1, pp. 33–57, Jan. 2007.
- [6] F. Bergh, "An Analysis of Particle Swarm Optimizers," PhD thesis, The University of Pretoria, 2001.
- [7] Z.-H. Zhan, J. Zhang, Y. Li, and H. S.-H. Chung, "Adaptive particle swarm optimization.," *IEEE Trans. on systems, man, and cybernetics.*, vol. 39, no. 6, pp. 1362–81, Dec. 2009.
- [8] A. H. Ballard, "A New Multiplex Technique for Communication Techniques," *IEEE Trans. on Power Apparatus and Systems*, vol. 85, no. 10, pp. 1054–1059, Oct. 1966
- [9] A. Behravan and T. Eriksson, "PAPR and other measures for OFDM systems with nonlinearity," in *Proc. IEEE 5th Intl. WPMC Symposium*, pp. 149–153, Oct. 2002.
- [10] Z. Wu, R. Qiu, and S. Zhu, "MIMO-OFDM PAPR Reduction by Space-Frequency Permutation and Inversion," in *Proc. of the IEEE 4th WiCOM*, 2008, no. 3, pp. 1–4, Oct. 2004.
- [11] E. B. Al-Safadi, "On Reducing the Complexity of Tone-Reservation Based PAPR Reduction Schemes by Compressive Sensing," in *Proc. IEEE GLOBECOM*, Dec. 2009.

- [12] T. A. Thomas and D. L. Jones, "PAPR Reduction For Beamforming OFDM Transmitters," in *Proc. of the IEEE ICASSP*, Vol.4, pp. 1–4, May 2006.
- [13] S. Cripps, *RF Power Amplifiers for Wireless Communications*, 2nd ed. Artech House Microwave Library, 2006.
- [14] S. Boumaiza and F. M. Ghannouchi, "Thermal memory effects modeling and compensation in RF power amplifiers and predistortion linearizers," *IEEE Trans. on Microw. Theory And Tech.*, vol. 51, no. 12, pp. 2427–2433, Dec. 2003.
- [15] T. Liu, S. Boumaiza, and F. M. Ghannouchi, "Identification and pre-compensation of the electrical memory effects in wireless transceivers," in *IEEE Radio and Wireless Symposium*, Jan. 2006, pp. 535–538.
- [16] W. Bosch and G. Gatti, "Measurement and simulation of memory effects in predistortion linearizers," *IEEE Trans. Microw. Theory and Tech.*, vol. 37, no. 12, pp. 1885–1890, Dec. 1989.
- [17] H. Murakami, Y. Yamashita, and T. Suetsugu, "Simulation analysis of transient behavior of RF class D power amplifier due to fluctuation in load impedance", in *Proc. IEEE TENCON*, Nov. 2010, pp. 1776–1779.
- [18] L. Ding, G. Zhou, and D. Morgan, "A robust digital baseband predistorter constructed using memory polynomials," *IEEE Trans. on Commun*, vol. 52, no. 1, pp. 159–165, Jan. 2004.
- [19] M. Ben Ayed and S. Boumaiza, "Experimental sensitivity analysis of multi-standard power amplifiers nonlinear characterization under modulated signals," in *Proc. IEEE ARFTG 75th*, May 2010, pp. 1–4.
- [20] Z. S. Z. Shi, J. Z. J. Zhou, H. Hayashi, and T. Kubo, "A coefficients extraction algorithm for LUT-based Hammerstein pre-distorters," in *Proc. IEEE PIMRC*, Sept. 2008, 1–4.
- [21] Z. He, W. Ye, and S. Feng, "Digital predistortion based on multidimensional look-up table storing polynomial coefficients," in *Proc. IEEE Electronics Letters*, vol. 48, no. 22, p. 1396, Oct. 2012.

- [22] J.T. Stonick, V.L. Stonick, J.M.F. Moura, R.S. Zborowski, "Memoryless Polynomial Adaptive Predistortion," in *Proc. IEEE ICASSP*, May 1995, Vol. 2, pp. 981–984.
- [23] T. Ogunfunmi, *Adaptive Nonlinear System Identification: The Volterra and Wiener Series Approaches*. Springer, 2007.
- [24] O.Hammi, F. M. Ghannouchi and B. Vassilakis, "A Compact Envelope-Memory Polynomial for RF Transmitters Modeling with Application to Baseband and RF-Digital Predistortion," in *Proc. IEEE Microw. Compon. Letters*, vol. 18, no. 5, pp. 359–361, May 2008.
- [25] C. Eun and E. J. Powers, "A New Volterra Predistorter Based on the Indirect Learning Architecture," *IEEE Trans. Signal Process.*, vol. 45, no. 1, pp. 223–227, Jan. 1997.
- [26] C. Crespo-Cadenas, J. Reina-Tosina, M. J. Madero-Ayora, and J. Munoz-Cruzado, "A New Approach to Pruning Volterra Models for Power Amplifiers," *IEEE Trans. on Signal Process.*, vol. 58, no. 4, pp. 2113–2120, Apr. 2010.
- [27] N. J. Bershad, S. S. McLaughlin, P. Celka, "Analysis of stochastic gradient identification of Wiener-Hammerstein systems for nonlinearities with Hermite polynomial expansions," *IEEE Trans. Signal Process.*, vol. 49, no. 5, pp. 1060–1072, May 2001.
- [28] N. J. Bershad, P. Celka, and J. Vesin, "Analysis of Stochastic Gradient Tracking of Time-Varying Polynomial Wiener Systems," *IEEE Trans. Signal. Process.* vol. 48, no. 6, pp. 1676–1686, Jun. 2000.
- [29] J. Liu, W.B. Xu and J. Sun, "Nonlinear System Identification of Hammerstien and Wiener Model Using Swarm Intelligence," in *Proc. IEEE Intl. Conf. on Inf. Acquisition*, pp. 1219–1223, Aug. 2006.
- [30] H. Lin, T. Liu, Y. Ye, X. Luo, G. Cao, and C. Sun, "Doherty power amplifier linearization with augmented Hammerstein model for LTE systems," in *Proc. IEEE ICMMT*, May 2012, vol. 4, pp. 1–4.
- [31] J. A. Cherry, "Distortion Analysis of Weakly Nonlinear Filters Using the Volterra Series," MSc thesis, Carleton University, 1995.

- [32] D. R. Morgan, Z. Ma, J. Kim, M. G. Zierdt, and J. Pastalan, "A Generalized Memory Polynomial Model for Digital Predistortion of RF Power Amplifiers," *IEEE Trans. on Signal Process.*, vol. 54, no. 10, pp. 3852–3860, Oct. 2006.
- [33] F. M. Ghannouchi, T. Liu, S. Boumaiza, "Augmented Hammerstein predistorter for linearization of broad-band wireless transmitters," *IEEE Trans. Microw. Theory Tech.*, vol. 54, no. 4, pp. 1340–1349, Jun. 2006.
- [34] G. M. P. Gilabert and E. Bertran, "On the Wiener and Hammerstein models for power amplifier predistortion," in *Proc. of the IEEE APMC, Dec. 2005*, vol. 2, pp. 1–4.
- [35] A. E. N. and L. H. Zetterberg, "Identification of certain time-varying nonlinear Wiener and Hammerstein systems," *IEEE Trans. Signal Processing*, vol. 49, no. 3, pp. 577–592, Mar. 2001.
- [36] M. Rawat, K. Rawat and F. M. Ghannouchi, "Three-Layered Biased Memory Polynomial for Dynamic Modeling and Predistortion of Transmitters With Memory," *IEEE Trans. on Circuits and Systems I*, vol. 60, no. 3, pp. 768–777, Mar. 2013.
- [37] S. Hong, Y. Y. Woo, J. Kim, J. Cha, I. Kim, J. Moon, J. Yi, and B. Kim, "Weighted Polynomial Digital Predistortion for Low Memory Effect Doherty Power Amplifier," *IEEE Trans. on Microw. Theory and Tech.*, vol. 55, no. 5, pp. 925–931, May 2007.
- [38] H. Zhou, G. Wan, and L. Chen, "A Nonlinear Memory Power Amplifier Behavior Modeling and Identification Based on Memory Polynomial Model in Soft-Defined Shortwave Transmitter," in *Proc. IEEE 6th WiCOM*, pp. 1–4, Sep. 2010.
- [39] K. Homayounfar and B. Rohani, "Digital Predistortion without Lookup Tables," in *IEICE Proc. SIP RCS*, 2009, pp. 1–5
- [40] O. Hammi, S. Boumaiza, and F. M. Ghannouchi, "On the Robustness of Digital Predistortion Function Synthesis and Average Power Tracking for Highly Nonlinear Power Amplifiers," *IEEE Trans. on Microw. Theory and Tech.*, vol. 55, no. 6, pp. 1382–1389, Jun. 2007.

- [41] W. Pan, Y. Liu and Y. Tang, "A Predistortion Algorithm Based on Accurately Solving the Reverse Function of the Memory Polynomial Model," *IEEE Wireless Commun. Lett.* , vol. 1, no. 4, pp. 384–387, Jun. 2012.
- [42] E. Eweda, "Transient performance degradation of the LMS, RLS, sign, signed regressor, and sign-sign algorithms with data correlation," *IEEE Trans. On Circuits And Systems II*, vol. 46, no. 8, pp. 1055–1062, Aug. 1999.
- [43] R. Raich, H. Qian, and G. T. Zhou, "Orthogonal Polynomials for Power Amplifier Modeling and Predistorter Design," *IEEE Trans. on Vehic. Tech.*, vol. 53, no. 5, pp. 1468–1479, Sept. 2004.
- [44] S. Ohmori, X. Guangsheng, O. Muta and Y. Akaiwa, "An Adaptive Predistortion Method Based on Orthogonal Polynomials," in *Proc. IEEE 18th PIMRC*, 2007, pp. 1–5, Sept. 2007.
- [45] O. Hammi and F. M. Ghannouchi, "Twin Nonlinear Two-Box Models for Power Amplifiers and Transmitters Exhibiting Memory Effects With Application to Digital Predistortion," *IEEE Microw. Wireless Compon. Lett.*, vol. 19, no. 8, pp. 530–532, Aug. 2009.
- [46] A. H. Abdelhafiz, O. Hammi, and A. Zerguine, "Efficient Adaptive Identification of Nonlinear Power Amplifiers Using The Twin-Nonlinear Two-Box Model," in *Proc. IEEE VTS APWCS*, Aug. 2012, pp.1–4 .
- [47] I. D. Coope, "Circle Fitting by Linear and Nonlinear Least Squares," *Journal of Optimization Theory and Applications* , vol. 76, no. 2, pp. 1–5, Feb. 1993.
- [48] O. Hammi, A. M. Kedir, F. M. Ghannouchi, "Nonuniform memory polynomial behavioral model for wireless transmitters and power amplifiers," in *Proc. IEEE APMC*, Dec. 2012, pp. 836–838.
- [49] O. Hammi, M. Younes and F.M Ghannouchi, "Metrics and Methods for Benchmarking of RF Transmitter Behavioral Models With Application to the Development of a Hybrid Memory Polynomial Model," *IEEE Trans. Broad.*, vol. 56, no. 3, pp. 350-357, Sept. 2010.
- [50] A. H. Sayed, *Fundamentals of Adaptive Filtering*. IEEE Press, 2003.
- [51] S. Haykin, *Adaptive Filter Theory*, 3rd ed. Prentice-Hall, 2001, pp. 10–61.

- [52] F. Boroujeny, *Adaptive Filters Theory and Applications*. Wiley, 1998.
- [53] K. Mayyas and T. Aboulnasr, "Leaky LMS algorithm: MSE analysis for Gaussian data," *IEEE Transactions on Signal Processing*, vol. 45, no. 4, pp. 927–934, Apr. 1997.
- [54] T. T. Aboulnasr and A. Zerguine, "Variable Weight Mixed-Norm LMS-LMF Adaptive Algorithm," in *Proc. IEEE 33rd ASILOMAR*, Oct. 1999, vol. 1, pp. 791–794.
- [55] S. J. Elliott, I. A. N. M. Stothers, and P. A. Nelson, "A Multiple Error LMS Algorithm and Its Application to the Active Control of Sound and Vibration," *IEEE Trans. on Acoustics, Speech and Signal Process.* vol.35, no. 10, pp. 1423–1434, Oct. 1987.
- [56] S. D. Muruganathan and A. B. Sesay, "A QRD-RLS-Based Predistortion Scheme for High-Power Amplifier Linearization," *IEEE. Trans. Circuits and Systems II*, vol. 53, no. 10, pp. 1108–1112, Oct. 2006.
- [57] B. Widrow and M. Hoff, "Adaptive Switching Circuits," in *Proc. IRE WESCON Conv.*, Aug. 1960, pp. 96–140.
- [58] Dimitri P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press. 1982.
- [59] N. Bershad, "Behavior of the ε -normalized LMS algorithm with Gaussian inputs," *IEEE Trans. on Acoustics, Speech and Signal Process.*, vol. 35, no. 5, pp. 636–644, May 1987.
- [60] D. T. M. Slock, "On the normalization of the step size in nonsymmetric stochastic gradient algorithms," in *Proc. IEEE 26th ASILOMAR*, Oct. 1992, vol.2, pp. 876–880.
- [61] S. Dasgupta, C. Johnson Jr., and A. Baksho, "Sign-sign LMS convergence with independent stochastic inputs," *IEEE Trans. on Inf. Theory*, vol. 36, no. 1, pp. 197–201, Jan. 1990.

- [62] V. H. Nascimento and A. H. Sayed, "Unbiased and stable leakage-based adaptive filters," *IEEE Trans. on Signal Process.*, vol. 47, no. 12, pp. 3261–3276, Dec. 1999.
- [63] 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.
- [64] S. H. Cho, S. D. Kim, and K. Y. Jeon, "Statistical convergence of the adaptive least mean fourth algorithm," in *Proc. IEEE 3rd ICSP*, Oct. 1996, vol. 1, pp. 610–613.
- [65] V. H. Nascimento and J. C. M. Bermudez, "When is the least-mean fourth algorithm mean-square stable?," in *Proc. ICASSP*, Mar. 2005, vol. 4, pp. 341–344.
- [66] E. Eweda and A. Zerguine, "A normalized least mean fourth algorithm with improved stability," in *Proc. IEEE 44th ASILOMAR*, 2010, pp. 1002–1005.
- [67] A. Zerguine, C. F. Cowan, and M. Bettayeb. "Adaptive echo cancellation using least mean mixed-norm algorithm," *IEEE Trans. on Signal Process.*, vol. 45, no. 5, pp. 1340-1343, May. 1997.
- [68] A. Zerguine and T. Aboulnasr, "A variable weight mixed-norm adaptive algorithm," in *Proc. IEEE ICASSP* , May 2002, vol. 2, pp. II–1401–II–1404.
- [69] S. H.-C. Shin and A. H. Sayed, "Transient behavior of affine projection algorithms," in *Proc. IEEE ICASSP*, April 2003, vol. 6, pp. VI–353.
- [70] D. Z. D. Zhou and V. E. DeBrunner, "Affine Projection Algorithm Based Direct Adaptations for Adaptive Nonlinear Predistorters," in *Proc. IEEE 40th ASILOMAR*, Oct. 2006, pp. 144–147.
- [71] R. Merched and A. H. Sayed, "Order-Recursive RLS Laguerre Adaptive Filtering," *IEEE Transc. Signal Process.* vol. 48, no. 11, pp. 3000–3010, 2000.
- [72] Azzedine Zerguine, Ahmar Shafi and Maamar Bettayeb, " Multilayer Perceptron-Based DFE with lattice structure," *IEEE Trans. Signal. Process.*, vol. 12, No.3, May 2001.

- [73] F. Ling and J. G. Proakis, "Adaptive lattice decision-feedback equalizers: Their performance and application to time-variant multipath channels," *IEEE Trans. Comm.*, vol. COM-33, pp. 348–356, April 1985.
- [74] P. Carro and P. Dúcar, "Nonlinear Distortion Cancellation Using Particle Swarm Optimization (PSO) based Predistortion in OFDM Systems," in *Proc 16th IST Mobile and Wireless Communications Summit*, Jul. 2007, pp. 1–5.
- [75] S. Chen, "An Efficient Predistorter Design for Compensating Nonlinear Memory High Power Amplifiers," *IEEE Trans.on Broadcasting*, vol. 57, no. 4, pp. 856–865, Dec. 2011.
- [76] X. H. X. Hu, Y. S. Y. Shi, and R. Eberhart, "Recent advances in particle swarm," in *Proc. IEEE CEC*, Jun. 2004, vol. 1, pp. 90–97.
- [77] A. T. Al-Awami, A. Zerguine, L. Cheded, A. Zidouri, and W. Saif, "A new modified particle swarm optimization algorithm for adaptive equalization," *Digital Signal Processing*, vol. 21, no. 2, pp. 195–207, Mar. 2011.
- [78] Y. De Valle, G. K. Venayagamoorthy, S. Mohagheghi, C. Jean-Carlos and R. G. Harley, "Particle Swarm Optimization: Basic Concepts , Variants and Applications in Power Systems," *IEEE Trans. Evol. Compu.* vol. 12, no. 2, pp. 171–195, April 008.
- [79] R. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proc. IEEE CEC*, 2000, vol. 1, pp. 84–88.
- [80] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: simpler, maybe better," *IEEE Trans. on Evol. Compu.*, vol. 8, no. 3, pp. 204–210, Jun. 2004.
- [81] J. Kennedy, "Social interaction is a powerful optimiser: The particle swarm," in *Proc. IEEE BICTA* , Sept. 2008, pp. 9–10.
- [82] Y. Gu, J. Jin, and S. Mei, "l₁₀ Norm constraint LMS algorithm for sparse system identification," *IEEE Signal Process. Lett.*, vol. 16, no. 9, pp. 774–777, Sep. 2009.
- [83] L. N. Robert L. Boylestad, *Electronic Devices and Circuit Theory*, 7th ed. Prentice Hall, 1999, pp. 53–71.

- [84] W. Doherty, "Amplifier," U.S. Patent US2210028 (A)1940.
- [85] P. Colantonio, F. Giannini, R. Giofre, and L. Piazzon, "The Doherty amplifier and its evolution for modern communication systems," in *Proc. IEEE MRRS*, Aug. 2011, pp. 51–54.
- [86] A. T. Al-Awami, E. V. Sortomme and M. A. El-Sharkawi, "Optimizing economic/environmental dispatch with wind and thermal units", in *Proc. IEEE/PES Gen. Meet.*, 2009
- [87] D. L. Donoho, "Compressed sensing," *IEEE Trans. Info. Theory*, vol.52, no. 4, pp. 1289–1306, Sept. 2006.
- [88] E. J. Candès, "Compressive sampling," in *Proc. International Congress of Mathematicians*, Madrid, Spain, 2006, vol. 3, pp. 1433–1452.
- [89] R. G. Baraniuk, "Compressive sensing," *IEEE Signal Processing Mag.*, vol. 24, no. 4, pp. 118–120, 124, July 2007.

VITAE

Name :Abubaker Hassan Babiker Abdelhafiz

Nationality :Sudanese

Date of Birth :6/1/1985

Email :abubakrh056@gmail.com

Address :KFUPM Campus, 31261, Dhahran. P.O. Box 8506

Academic Background :BSc. in Electrical Engineering, University of Khartoum.
MSc. in Electrical Engineering, King Fahd University of
Petroleum and Minerals.