Optimum DOA algorithms for narrowband and wideband applications

Bao, Haoji

2005

Bao, H. (2005). Optimum DOA algorithms for narrowband and wideband applications. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/4602>

<https://doi.org/10.32657/10356/4602>

Nanyang Technological University

Downloaded on 26 Apr 2023 18:49:07 SGT

**Optimum DOA Algorithms
For
Narrowband and Wideband Applications**

Bao Haoji

**SCHOOL OF ELECTRICAL & ELECTRONIC ENGINEERING
NANYANG TECHNOLOGICAL UNIVERSITY**

A thesis submitted to Nanyang Technological University
in fulfillment of the requirement for the Degree of
Master of Engineering

2005

Acknowledgements

There are no words that can fully express my sincere gratitude to my supervisor, Assoc Prof. Ng Boon Poh. His inspiring encouragement, insightful suggestion and valuable advice have consistently guided me throughout my research work. Without his continuous support and guidance, the completion of this project would be impossible.

I would also like to thank Dr. Ma Changzheng, Dr. Feng Aigang and Dr. Yang Yixin for their generous time and patience. Their helpful opinions and friendly assistance are very much appreciated.

Last but not least, I would like to thank staff in Information System Research Lab for providing me with an excellent working environment and technical support.

Table of Contents

Acknowledgements	i
List of Figures	vii
List of Tables	x
Summary	xi
1 Introduction	1
1.1 Background	1
1.2 Motivation	4
1.3 Objectives	5
1.4 Major Contribution of the Thesis	5
1.5 Organization of the Thesis	6
2 Estimation of Directions Of Arrival (DOA) using Adaptive Array Processing Techniques	8
2.1 Principle of Adaptive Array Processing	8

TABLE OF CONTENTS

iii

2.2	Antenna Array Geometry Consideration	11
2.3	Fundamentals of DOA Estimation	12
2.3.1	Signals and Noise	12
2.3.2	Advantages using an Array	16
2.3.3	Eigenvector-Based Techniques	18
2.4	Conventional Method: Capon's Minimum Variance Estimator	23
2.5	Subspace Methods for DOA Estimation	25
2.5.1	MUSIC Algorithm	25
2.5.2	Minimum Norm Method	27
2.5.3	Unity Response Algorithm	29
3	An Averaged Approach to Phase Optimization in MN Algorithm & its Extension to UR Algorithm	31
3.1	Issue of Phase Optimization in MN Estimator	31
3.2	Proposition of the Averaged Method to MN Algorithm	33
3.3	Simulation and Observation	36
3.4	Averaged Method used in UR Algorithm	39
3.4.1	Equivalence of Phase Issues in MN and UR Templates	39
3.4.2	Averaged Method Extended to UR Algorithm	40
3.5	Statistical Performance Analysis of UR Algorithm	42
4	Phase Selection using a Constant Modulus Approach	49

TABLE OF CONTENTS

iv

4.1	Optimization via “Phase-synchronization”	49
4.2	Iterative Phase Selections using a Constant Modulus Technique . . .	52
4.3	Consideration of the Imposed Eigen-constraint	56
4.3.1	A Signal Subspace Matrix Approach	56
4.3.2	A Noise Subspace Matrix Approach	58
4.4	Numerical Results	59
5	DOA Estimation of Wideband Signal Sources	65
5.1	Unity-Response Approaches to DOA Estimation of Wideband Signals	65
5.2	A 2-D Search Approach	66
5.2.1	Formulation	66
5.2.2	Ways to Determine Number of Signal Sources	69
5.2.3	Simulation Results	72
5.2.4	Comments on the Proposed Method	77
5.3	A Taylor-Series Approach	78
5.3.1	Formulation	78
5.3.2	Simulation and Discussion	81
6	Implementation of Capon and UR Algorithms on DaqPC	85
6.1	Hardware System Description	85
6.1.1	Introduction of DaqPC	85

TABLE OF CONTENTS

v

6.1.2	Configurations of DaqPC 554/111	87
6.1.3	System Operation	91
6.2	Structure of the DSP Blade	95
6.3	Algorithm Coding and Experimental Results	99
7	Conclusion and Recommendation	105
7.1	Conclusion	105
7.2	Recommendations for Further Research	108
	Author's Publications	110
	Bibliography	111
	Appendices	119
A	Analysis of the Statistical Performance of UR Algorithm	119
A.1	Concept of Root Unity Response Method	119
A.2	Derivation of Mean Squared Error	121
B	The Closed Form Solution of Quadratic Optimization Problem in UR Algorithm	126
C	Source Code in C for Capon and UR Algorithms	129
C.1	Main Invoking Function	129
C.2	Invoking Function of Capon Estimator	132

TABLE OF CONTENTS

vi

C.3 Invoking Function of UR Estimator	137
D Pre-defined Functions in “ClientDSP.h”	145

List of Figures

2.1	Functional block diagram of an M-element adaptive array system . . .	9
2.2	Illustration of signals picked by a pair of sensor elements	15
2.3	A typical linear array scene	19
3.1	Distribution of “signal” zeros and “noise” zeros	34
3.2	Comparison between MN, MUSIC and Averaged MN estimators . . .	37
3.3	MSE in DOA estimation for UR and MUSIC estimators	46
3.4	MSE in DOA estimation for Averaged UR estimators	47
3.5	Examination of resolution of the 3 different estimators	47
4.1	UR responses plotted using different phase values	51
4.2	A comparison between before optimization and after	60
4.3	A magnified image of one of the peaks	61
4.4	Illustration of iteration convergence	61
4.5	A comparison between signal and noise sub-space approach	62
4.6	A comparison between using different initialization values	62

LIST OF FIGURES

viii

4.7	Plot of the average absolute value of phase difference per iteration . .	64
5.1	A double-ring array geometry	72
5.2	UR Estimator extended to wideband scenario using method A	74
5.3	UR Estimator extended to wideband scenario using method B	75
5.4	UR Estimator extended to wideband scenario using method C	76
5.5	UR extended for wideband application via a Taylor-series approach .	82
5.6	Phase optimized UR in broadband scenario using CM approach . . .	83
6.1	Picture I of the DaqPC system	88
6.2	Picture II of the DaqPC system	88
6.3	Block diagram of DaqPC 554/111	89
6.4	Schematic diagram of system configuration for DOA estimation . . .	92
6.5	An outlook of the dBm frequency converter	93
6.6	Transmit antenna (one antenna used to transmit signal)	94
6.7	Receive antenna array of 4 elements	94
6.8	DSP blade physical layout diagram	95
6.9	Internal structure of Mango Seagull Quad C64 DSP board	98
6.10	Flowchart of programming Capon algorithm	100
6.11	Flowchart of programming Unity-Response algorithm	101
6.12	Real-time data acquired and recorded by the Rx Blade	102
6.13	DOA estimation in real environment using Capon algorithm	103

LIST OF FIGURES

ix

6.14 DOA estimation in real environment using UR algorithm	103
6.15 Capon estimator using Matlab generated data	104
6.16 UR estimator using Matlab generated data	104

List of Tables

6.1	Acronyms of system parts	87
6.2	Description of functional blocks	90
6.3	Constituent boards of DSP blade	96

Summary

Direction-of-arrival (DOA) estimation is a major research topic in the field of array signal processing. In recent developments of DOA estimation, focus has been put on tailoring of existing algorithms for implementation purpose. However, there are still issues unsolved regarding the basic DOA theories, one of which is concerning phase determination/selection. Moreover, most of the existing DOA estimators were originally designed only for narrowband applications. Given the increasing need of solutions in wideband scenario, wideband DOA algorithms become important and necessary.

This project targets two critical issues, 1)to study existing algorithms and to optimize them by mitigation/elimination of performance degradation caused by phase uncertainties; 2)to extend narrowband DOA algorithm to wideband application and provide wideband solutions. In this thesis, the nature of phase uncertainties in Minimum-norm (MN) and Unity-response (UR) algorithms is studied and ways of optimization are proposed accordingly. An averaged method is designed for MN algorithm by exploiting its characteristics in the concept of polynomial rooting. And an iterative way of phase optimization is proposed for UR algorithm using a constant modulus approach. Subsequently, two different methods to extend narrowband UR estimator to wideband applications are discussed. We hope that, through the thesis, we are able to make the theory of DOA estimation more complete.

During the completion of this thesis, we also had the opportunity to implement DOA algorithms on a hardware system, i.e. DaqPC 554/111. A DaqPC is a high-end data acquisition and processing system that combines the low cost and familiarity of a standard PC environment with high performance PCI products from ICS (Interactive Circuits and Systems Ltd) and third party DSP and data storage hardware. It can simulate signal transmit and receive in a real environment. Various signal processing algorithms can be coded onto the system. In this project, DOA methods of Capon and Unity-Response algorithms are implemented and their effectiveness are verified with experimental results.

Chapter 1

Introduction

1.1 Background

The wireless communication industry has grown explosively over the last few years. Recent evolutions in the telecommunications arena indicate a clear trend towards enhanced services that are expected to flourish in the near future. The advent of rate-demanding services such as tele-commuting, home networking, video conference, fast wireless/mobile internet access and multimedia constitute only the first manifestation of the projected demand for broadband access to information sources of every kind. The need for ubiquitous coverage and connectivity in personal, local or wide area networks and the increasing demand for mobility, flexibility and simplicity of system deployment necessitate wireless access. This prompts a demanding increase of the capacity on the existing wireless communication infrastructure.

To increase the capacity of communication systems, four kinds of multiple access techniques have been invented, namely (1) Frequency Division Multiple Access (FDMA), (2) Time Division Multiple Access (TDMA), (3) Code Division Multiple Access (CDMA) and (4) Spatial Division Multiple Access (SDMA). In the past

years, the first three multiple access techniques have been thoroughly investigated and successfully deployed in various kinds of wireless access applications. Recently, researchers have started to pay more and more attention to SDMA approach with the hope of further increasing wireless system capacity in the spatial domain. Spatial Division Multiple Access techniques allow multiple users to transmit information in the same wireless radio channel at the same time. This is usually accomplished by deploying “Smart Antennas”. Smart antennas have the property of spatial filtering, which makes it possible to receive signal from a particular direction while simultaneously blocking it from other directions. This property makes smart antennas a very effective tool in detecting and locating a signal source of various kinds. Moreover, its ability to direct transmitting energy toward a desired direction makes smart antenna a very useful technique in many fields, such as medical diagnosis, underwater detection or jamming signal cancellation in a communications system.

The smart antenna field has been a very active area of research for over four decades. Many types of processors for smart antennas have been proposed and their performance has been studied. Techniques used in smart antennas fall into the area of array signal processing. Sensor arrays have been in use for several decades in many practical signal processing applications. Such an array consists of a set of sensors that are spatially distributed at known locations with reference to a common reference point. These sensors collect signals from sources in their field of view. Depending on the sensor characteristics and the propagation path, the source waveforms undergo deterministic and/or random modifications. The sensor outputs are composed of these source components and additive noise. Practical problems of interest in array signal processing are extracting the desired parameters such as directions of arrival, power levels and cross-correlations of the signals present in the scene from the available information including the measured data.

One of the interesting topics that have drawn most attention of researchers in the

field of array signal processing is estimating Directions Of Arrival of impinging signals, namely, DOA estimation. It is a special case of the general parameter estimation problem. DOA estimation is also referred to as source localization in sensor array signal processing. Its primary goal is to estimate the impinging direction of the source signal received by the sensor arrays. Many techniques for accurate and fast DOA estimation have been invented, among which eigenstructure-based algorithms are of great importance.

In the last decade, subspace processing techniques have found prominence in the area of DOA estimation, which makes eigenstructure-based techniques a popular subject of active research. Many eigenstructure-based algorithms, pioneered by Pisarenko and generalized by Schmidt and Bienvenu and Kopp, have been proposed [1]. Below is a list of some well-known DOA estimation algorithms,

- Capon's Minimum Variance Method
- Linear Predictive (LP) Method
- Multiple Signal Classification (MUSIC) Algorithm
- Minimum Norm (MN) Method
- Unity-Response (UR) Algorithm
- Estimation of Signal Parameters via Rotational Invariance Techniques (ES-PRIT)
- Matrix Pencil (MP) Method

Most of these algorithms have been investigated in great details and practically implemented in many applications. In most cases, they were proven effective, efficient and accurate. Nevertheless, despite the sophisticated nature of these algorithms,

there are still issues that remain open, which means there is room for them to be further improved. Moreover, most of these algorithms were initially developed for plane wave signals emitted by narrowband sources only. With the fast development of wireless communication, it is necessary to find out wideband solutions.

1.2 Motivation

Direction of arrival (DOA) estimation has been one of the active research topics in the field of array signal processing. Various algorithms have been proposed by researchers and indicated of great performance in different applications. However, in the history of DOA estimation, there are still issues unsolved, one of which is concerning the determination of phase quantity or position. It can be found in many widely-used theorems or algorithms such as Linear Predictive (LP) theorem, Minimum Norm (MN) algorithm and Unity-Response (UR) algorithm. In some cases, an improper decision made on the phase value can lead to a severe degradation in performance. Despite the dramatic effects of these phase issues, in the recent development of DOA estimation study, attention has mainly been focused on tailoring of existing algorithms for implementation purposes. To the best of our knowledge, there are no outstanding methods that can effectively eliminate or mitigate the performance degradation caused by this phase issue. Therefore, it is vital to find solutions to the issue of phase optimization.

On top of that, most of the DOA algorithms were initially designed for the narrowband signals only. However, with the rapid development of modern communications industry, wideband solutions for DOA estimation become more and more necessary. There are so far very few algorithms that have been successfully extended to wideband applications. Famous as MUSIC or Minimum Norm algorithms, very seldom were articles published on them being applied in wideband scenario.

1.3 Objectives

Motivated by the necessity for phase optimization in DOA algorithm and increasing need for wideband solutions, the objectives of this project are therefore:

1. To study and compare existing DOA algorithms and verify their effectiveness.
2. To find solutions to phase optimization issue that exists in Minimum Norm and Unity-Response algorithms.
3. To extend Unity-Response algorithm to wideband applications.

1.4 Major Contribution of the Thesis

The main contributions of this project are as follows,

1. An averaged method is proposed to Minimum Norm algorithm to mitigate the effect of phase error. This approach is proposed by exploiting the nature of Minimum Norm algorithm in terms of the concept of polynomial rooting. It is subsequently extended to Unity-Response algorithm due to its similarity to Minimum Norm algorithm. Moreover, statistical performance of Unity-Response algorithm is studied.
2. A new iterative approach is designed for Unity-Response algorithm, aiming to eliminate the need for phase optimization. The proposed approach adopts Constant Modulus technique. It ensures that the array weight vector is computed in such a way that it best satisfies the condition in which the phase values selected are equal to that of the array outputs, such that the limitation of the implied phase constraint in Unity-Response algorithm is eliminated.

3. Two techniques to extend Unity-Response algorithm for DOA estimation of multiple narrowband and wideband signal sources are proposed. One of the techniques proposed is essentially an extension of the original idea of Unity-Response algorithm in the frequency domain, which performs a 2-dimensional search over both frequency and spatial spectrum. The other technique utilizes Taylor-series to extend signal subspace in Unity-Response template, so that wideband signals consisting of different frequency components can be detected.
4. Capon and Unity-Response algorithms are programmed in C and implemented using DaqPC 554/111. The configurations of a DaqPC is studied and its ways of operation are handled. Simulation is conducted in a real environment and real-time data captured is processed using Capon or Unity-Response estimator. The obtained experimental results are compared with those generated by Matlab simulations, from which the effectiveness of these two DOA algorithms are verified.

1.5 Organization of the Thesis

This thesis consists of 7 chapters. In Chapter 2, it firstly introduces the fundamentals of adaptive array processing and explains the signal and noise model in details. Subsequently, it is provided a literature review on typical DOA estimation algorithms that have been studied, compared or improved in this project. Meanwhile, terms and notations encountered frequently in DOA estimation are introduced. In Chapter 3, an averaged method is applied in Minimum Norm algorithm to mitigate the effects of phase optimization. Furthermore, the proposed approach is extended to Unity-Response algorithm. Statistical performance of Unity-Response is also studied in this chapter. Chapter 4 includes the Constant Modulus approach proposed

to eliminate phase uncertainty embedded in Unity-Response template. Computer study is conducted to verify the analysis. Chapter 5 introduces two approaches proposed to extend Unity-Response algorithm to wideband applications. Pros and cons of them are discussed respectively. Moreover, the Constant Modulus method proposed in Chapter 4 is also adopted for wideband scenario. In Chapter 6, hardware implementation of Capon and Unity-Response algorithms is conducted on a hardware platform of a DaqPC 554/111. Internal structures of a DaqPC are described and experimental results are attached. Lastly, Chapter 7 concludes this research project and makes recommendation for future work.

Chapter 2

Estimation of Directions Of Arrival (DOA) using Adaptive Array Processing Techniques

2.1 Principle of Adaptive Array Processing

An adaptive array is a system consisting of an array of antenna elements and a real-time signal-processing unit that is able to automatically adjust the array beam sensitivity. It is adaptive in a way that array weight factors are obtained from the real-time signal received by the array. Adaptive array techniques depend on digital beamforming (BF) technology for their practical implementation. Digital beamforming refers to a processing structure that accepts digital signals from multiple sensors in an antenna array and then performs spatial processing on them. The functional block diagram of a typical adaptive array system is illustrated in Figure 2.1.

As in the diagram, the adaptive array system consists mainly of an antenna array,

Figure 2.1: Functional block diagram of an M-element adaptive array system

a beamforming network, and an adaptive beam control unit or adaptive processor that adjusts the variable weights in the beamforming network. The adaptive beam control unit may be divided into 2 parts: a signal processing block and an adaptive control block. The manner in which the system is actually implemented relies heavily on the propagation medium in which the array operates, the frequency spectrum of interest, and the user's knowledge of the operational signal environment. The array itself consists of antenna elements designed to receive (and/or transmit) signals. The antennas are arranged to give adequate coverage over a certain desired spatial region. The output of each antenna is sent to the beamforming network through receiving channels and subsequently multiplied by a complex weight factor with both amplitude and phase. Outputs from each antenna are then summed up to give the overall array output. It follows from Figure 2.1 that an expression for

the array output is given by

$$y(t) = \sum_{i=1}^M w_i^* x_i(t) = \mathbf{w}^H \mathbf{x}(t) \quad (2.1.1)$$

where it denotes the weights and signals of each antenna in a vector form,

$$\mathbf{w} = [w_1, w_2, \dots, w_M]^T \quad (2.1.2)$$

$$\mathbf{x}(t) = [x_1(t), x_2(t), \dots, x_M(t)]^T \quad (2.1.3)$$

Throughout this thesis, the superscript ‘ T ’ refers to the matrix transpose, the superscript ‘ H ’ refers to the complex-conjugate transpose, the superscript ‘ $*$ ’ refers to the complex-conjugate and the superscript ‘ -1 ’ will refer to matrix inverse. In addition, matrices and vectors are denoted by bold alphabets.

The weight values w_i ’s within the beamforming network then determine the overall array beam pattern. It is this ability to shape the overall beam pattern and adjust pattern gain that in turn determines how well the specified system requirements can be met for a given signal environment. With the weight control, interference suppression can be obtained by appropriately steering beam nulls and reducing side-lobe levels in the direction of interference sources, while desired signal reception is maintained or enhanced by steering mainlobe to the signal direction. Therefore, an adaptive array system depends largely on spatial characteristics to improve the output Signal-to-Noise Ratio (SNR) for spatial selection of desired signals.

2.2 Antenna Array Geometry Consideration

An antenna array, as the name may imply, is a collection of antennas located at different locations. The array itself may take on a variety of different geometries in light of applications of interest. Different array geometries can impose different impact on adaptive array system. In particular, the array geometry should be matched to the spatial characteristics of the radio environment. For example, in rural operational environments where angular spreading is low, and received signals tend to have well-defined directions of arrival. However, in the case of indoor environment, the angular spreading is high, thus, signals may not have well-defined directions of arrival. The choice of array topology (as well as the control algorithms) should reflect these differences in the signal environment. Hence, no single antenna array geometry is universally optimum for all environments.

In general, there are two types of geometry, i.e. linear arrays and planar arrays such as circular array, rectangular array and triangular array. After array geometry is determined, it is the array element-spacing that is the next thing of concern. Array element-spacing has great influence on array aperture and resolution. Generally, in a loose sense, as the element-spacing in an array increases, beamwidth of mainlobe in array beam pattern decreases and array resolution increases. High array resolution improves maximum output SNR when the angular separation between desired and undesired signals is small. However, on the other hand, large element-spacing may give rise to the occurrence of grating lobes and produce more sidelobes. Thus, there is an extent to which a particular array should stretch its element-spacing to keep high resolution, yet avoid grating lobes. For a uniform linear array (ULA) where array elements are equally spaced, it is shown in [2] that the maximum element-spacing is half of the wavelength of the received signals. It is also found, in a recent study of this issue, that for a ULA, element-spacing of the array can be extended

beyond half wavelength [3].

In this project, however, theoretical investigation and computer simulation have been conducted mainly on circular arrays or uniform linear arrays with half wavelength element-spacing. It can be seen in the following chapters that spectra of DOA estimations using different algorithms on the specified array geometry are analyzed.

2.3 Fundamentals of DOA Estimation

2.3.1 Signals and Noise

The Control Unit in Figure 2.1 is the crucial part of an adaptive array system. For DOA estimations in different scenarios, there are different control algorithms that can be applied. However, before proceeding to discuss these algorithms in detail, it is necessary to understand the nature of signals and noise the array is to receive. It is also helpful for a better understanding of using a sensor array over a single element in various aspects of detection and estimation.

At any instant, every signal has an amplitude component and a phase (with respect to a reference) component. At times, in addition to this in-phase component, it becomes necessary to generate its quadrature part for optimum processing [2]. Instead of carrying out all the computations separately in terms of their in-phase and quadrature phase components, it is often convenient and advantageous to represent them as the real and imaginary parts of a complex signal. The actual signals that appear in any physical system are real and in this representation they appear as the real part of the equivalent complex signal [4]. Thus, at a reference element if

$$u_r(t) = \sqrt{P} \cos(\omega_0 t + \phi(t) + \theta) \quad (2.3.4)$$

is the actual phase modulated carrier signal with a random phase factor θ , its complex representation is

$$\tilde{u}(t) = \sqrt{P}e^{j(\omega_0 t + \phi(t) + \theta)} \quad (2.3.5)$$

Clearly,

$$u_r(t) = \text{Re}[\tilde{u}(t)] \quad (2.3.6)$$

where $\text{Re}[\cdot]$ stands for the real part of $[\cdot]$ and

$$u(t) = \sqrt{P}e^{j(\phi(t) + \theta)} \quad (2.3.7)$$

is known as the baseband reduced complex envelop of the real signal $u_r(t)$. Thus, only the information carrying component, and not the carrier frequency of the modulated signal, appears in this description. In physical terms, if the signals have originated far away from the array, then they can be modelled as information carrying uniform plane waves as above. If these plane waves advance through a non-dispersive medium that only introduces propagation delays, the output of any other array element can be represented by a time-advanced or time-delayed version of the above complex envelop at the reference element [5]. Once again this avoids the carrier frequency description.

Signals such as above that can be adequately characterized by a single carrier frequency are known as narrowband sources. In contrast to this, signals that occupy a significant spectral band constitute wideband (broadband) sources. Further all of these signals can be uncorrelated, correlated or coherent with each other. In particular, for two wide-sense stationary signals $u_i(t), u_j(t)$, let ρ_{ij} represent their correlation coefficient. By definition,

$$\rho_{ij} = \frac{E[u_i(t)u_j^*(t)]}{\sqrt{E[|u_i(t)|^2]E[|u_j(t)|^2]}} \quad (2.3.8)$$

where $E[.]$ stands for the expectation of $[.]$.

Following from the Schwarz inequality, it is concluded $|\rho_{ij}| \leq 1$, thus,

$$\begin{aligned}\rho_{ij} = 0 &\longrightarrow u_i(t), u_j(t) \text{ are uncorrelated} \\ 0 < |\rho_{ij}| < 1 &\longrightarrow u_i(t), u_j(t) \text{ are correlated} \\ |\rho_{ij}| = 1 &\longrightarrow u_i(t), u_j(t) \text{ are coherent}\end{aligned}$$

It is not difficult to show that under coherent conditions,

$$u_j(t) = \alpha u_i(t) \quad (2.3.9)$$

where α is a nonrandom complex constant. In practice, the additive noise components are uncorrelated with the signal part. However, among themselves the inter-element noises may be correlated or uncorrelated with each other. One standard assumption is to consider them as uncorrelated (independent in case of Gaussian) and identical processes, i.e., if $n_i(t)$, $n_j(t)$ represent i^{th} and j^{th} sensor noise, then

$$E[n_i(t)n_j^*(t)] = \sigma^2\delta_{ij} \quad (2.3.10)$$

where σ^2 represents the noise power common to all sensors.

It was remarked earlier that when an advancing plane wave passes through a non-dispersive medium, the signal output at any sensor element immersed in that medium can be represented as a time-delayed/advanced version of its complex envelop at a reference element. To see this, let $\tilde{u}$ in (2.3.5) denote the complex signal representing a modulated plane wave at the reference element in Figure 2.2. The normal to the plane wavefront makes an angle θ with the line joining the sensors in the linear array. Further let $\tilde{d}$ denote the distance of the second sensor from the reference point in absolute units. The output at the second sensor is denoted by $\tilde{v}(t)$. It

Figure 2.2: Illustration of signals picked by a pair of sensor elements

can be seen that in Figure 2.2 that the time difference is determined by the time for the plane wave to propagate through $\tilde{d} \cos(\theta)$ and if c represents the velocity of propagation, then this time delay q is given by

$$q = \frac{\tilde{d} \cos(\theta)}{c} \quad (2.3.11)$$

Thus

$$\tilde{v}(t) = \tilde{u}(t + q) \quad (2.3.12)$$

If the carrier frequency is fairly large compared to the bandwidth of the modulating signal, then the modulating signal can be treated as quasi-static during time intervals of order q and in that case (2.3.12) reduces to

$$\tilde{v}(t) = \tilde{u}(t) e^{\frac{j\omega_0 \tilde{d} \cos \theta}{c}} = \tilde{u}(t) e^{\frac{j2\pi \tilde{d} \cos \theta}{\lambda}} \quad (2.3.13)$$

where λ is the associated carrier wavelength. In terms of the baseband reduced complex envelop representation, the reference signal $u(t)$ is given by (2.3.7) and from (2.3.13) the second sensor output is

$$v(t) = u(t)e^{j2\pi\tilde{d}\cos\theta/\lambda} \quad (2.3.14)$$

If we let d represent the normalized distance between the reference element and the second sensor with respect to half wavelength, i.e. $d = \tilde{d}/(\lambda/2)$, then (2.3.14) becomes

$$v(t) = u(t)e^{j\pi d\cos\theta} \quad (2.3.15)$$

Evidently, the time delay (difference) appears as a pure phase delay of the reference signal.

2.3.2 Advantages using an Array

The possibility of modifying the array outputs to enhance the desired signal reception and simultaneously suppress the undesired ones can be illustrated by considering a single source situation as in Figure 2.2 in the presence of M identical sensors. Let $d_1, d_2, \dots, d_M$ represent the normalized distances of these sensors with respect to a reference point and $u(t)$ the complex envelop of the signal at that point. Further let $n_1(t), n_2(t), \dots, n_M(t)$ represent the respective noise components that are assumed to be identically distributed as in (2.3.10). With $x_i(t)$ representing the complex envelop of the total received signal at the i^{th} sensor, and using (2.3.15) it is easy to see that

$$x_i(t) = u(t)e^{j\pi d_i \cos\theta} + n_i(t) \quad (2.3.16)$$

and the input SNR is

$$SNR_i = \frac{E[|u(t)|^2]}{E[|n_i(t)|^2]} = \frac{P}{\sigma^2} \quad (2.3.17)$$

where $P = E[|u(t)|^2]$ represents the signal power. From (2.3.16) the signal components can be coherently combined if the array outputs are deliberately phase shifted by $e^{-j\pi d_i \cos \theta}$; $i = 1, 2, \dots, M$ and the resulting signals are added up. This gives the output signal $y(t)$ to be

$$y(t) = \sum_{i=1}^M x_i(t) e^{-j\pi d_i \cos \theta} = Mu(t) + \sum_{i=1}^M n_i(t) e^{-j\pi d_i \cos \theta} = Mu(t) + n(t) \quad (2.3.18)$$

The output SNR in this case is given by

$$\begin{aligned} SNR_o &= \frac{E[|Mu(t)|^2]}{E[|n(t)|^2]} = \frac{M^2 P}{|\sum_i \sum_j E[n_i(t) n_j^*(t)]|} \\ &= \frac{M^2 P}{M \sigma^2} = M(SNR_i) \end{aligned} \quad (2.3.19)$$

Thus a simple phase shifting and adding operation among the sensor outputs results in an improvement in the signal-to-noise ratio by a factor equal to the number of sensors. Physically, through appropriate phase delays the desired signal has been coherently combined (in voltage), with noise adding up only incoherently (in power). This results in a gain factor for the overall output signal compared to the noise. Alternatively, the array has been “steered” to look along the direction θ in this process, which is also known as beamforming.

Traditionally the array output power is evaluated as a function of the arrival angle under various optimality considerations, and the peaks in the output power distribution are taken to correspond to the true directions of arrival of signals present in the scene. This is the conventional techniques used in classical DOA estimation algorithms, e.g. Capon or Linear Predictive method.

With notations defined in (2.1.1)~(2.1.3), the array output power can be expressed

$$P_o = E[|y(t)|^2] = \mathbf{w}^H E[\mathbf{x}(t)\mathbf{x}^H(t)]\mathbf{w} = \mathbf{w}^H \mathbf{R} \mathbf{w} \quad (2.3.20)$$

where

$$\mathbf{R} = E[\mathbf{x}(t)\mathbf{x}^H(t)] \quad (2.3.21)$$

represents the $M \times M$ array output covariance matrix. Notice that $\mathbf{R}$ is hermitian ($\mathbf{R} = \mathbf{R}^H$) and always nonnegative definite (i.e. $\mathbf{w}^H \mathbf{R} \mathbf{w} \geq 0$).

2.3.3 Eigenvector-Based Techniques

Eigenvector-based DOA algorithms contribute a significant part in the field of DOA estimation. In the last decade, eigenvector-based subspace techniques have increasingly become prominent and a major topic of research in different DOA estimation scenarios. Regardless of difference of these algorithms in terms of formulations and expressions, they all have the following important property in common.

Consider a linear array consisting of M identical sensors and receiving signals from K ($K \leq M$) uncorrelated narrowband signals $u_1(t), u_2(t), \dots, u_K(t)$ that arrive at the array from directions $\theta_1, \theta_2, \dots, \theta_K$ with respect to the line of array in Figure 2.3. Using superposition theorem of different signal contributions, the received signal $x_i(t)$ at the i^{th} sensor can be written as

$$x_i(t) = \sum_{k=1}^K u_k(t) e^{j\pi d_i \cos \theta_k} + n_i(t) \quad (2.3.22)$$

where d_i , as before, represents the i^{th} sensor element position normalized to half of the common wavelength, with respect to a common reference point, and $n_i(t)$ stands for the additive noise at the i^{th} sensor. It is assumed that noises are stationary,

Figure 2.3: A typical linear array scene

zero-mean, uncorrelated random processes, and further as in (2.3.10) the noises are assumed to be uncorrelated and identical between themselves with common variance σ^2 . Rewriting (2.3.22) in common vector notation, we have

$$\mathbf{x}(t) = \sum_{k=1}^K u_k(t) \mathbf{s}(\theta_k) + \mathbf{n}(t) \quad (2.3.23)$$

where $\mathbf{x}(t)$ is a $M \times 1$ vector as in (2.1.3), representing signals picked up by each array element. And $\mathbf{s}(\theta_k)$ is a direction vector associated with the arrival angle θ_k , defined as Steering Vector (SV) of angle θ_k , i.e.,

$$\mathbf{s}(\theta_k) = [e^{j\pi d_1 \cos \theta_k}, e^{j\pi d_2 \cos \theta_k}, \dots, e^{j\pi d_M \cos \theta_k}]^T \quad (2.3.24)$$

and

$$\mathbf{n}(t) = [n_1(t), n_2(t), \dots, n_M(t)]^T \quad (2.3.25)$$

Further, (2.3.23) can be re-written as

$$\mathbf{x}(t) = \mathbf{S}\mathbf{u}(t) + \mathbf{n}(t) \quad (2.3.26)$$

where

$$\mathbf{u}(t) = [u_1(t), u_2(t), \dots, u_K(t)]^T \quad (2.3.27)$$

$$\mathbf{S} = [\mathbf{s}(\theta_1), \mathbf{s}(\theta_2), \dots, \mathbf{s}(\theta_K)] \quad (2.3.28)$$

From our assumptions, it follows that the $M \times M$ array output covariance matrix defined in (2.3.21) has the form

$$\begin{aligned} \mathbf{R} &= E[\mathbf{x}(t)\mathbf{x}^H(t)] \\ &= \mathbf{S}E[\mathbf{u}(t)\mathbf{u}^H(t)]\mathbf{S}^H + E[\mathbf{n}(t)\mathbf{n}^H(t)] = \mathbf{S}\mathbf{R}_u\mathbf{S}^H + \sigma^2\mathbf{I} \end{aligned} \quad (2.3.29)$$

Notice that $\mathbf{R}$ is hermitian and positive definite and by definition

$$\mathbf{R}_u = E[\mathbf{u}(t)\mathbf{u}^H(t)] \quad (2.3.30)$$

represents the source covariance matrix of size $K \times K$ which remains nonsingular so long as there are no coherent signal sources present in the data. Therefore, in our case, $\mathbf{R}_u$ is of rank K and from the linear independence of the steering vectors $\mathbf{s}(\theta_1), \mathbf{s}(\theta_2), \dots, \mathbf{s}(\theta_K)$, it is evident $\mathbf{S}$ is also of rank K , which taken together implies that the nonnegative definite matrix $\mathbf{S}\mathbf{R}_u\mathbf{S}^H$ is also of rank K . Let $\mu_1 \geq \mu_2 \geq \dots \geq \mu_K$ denote its K nonzero eigenvalues. Then the M eigenvalues of $\mathbf{R}$ are given by

$$\lambda_k = \begin{cases} \mu_k + \sigma^2 & k = 1, 2, \dots, K \\ \sigma^2 & k = K + 1, K + 2, \dots, M \end{cases} \quad (2.3.31)$$

and further let $\beta_1, \beta_2, \dots, \beta_K, \beta_{K+1}, \dots, \beta_M$ represent their associated eigenvectors. Then

$$\mathbf{R} = \sum_{k=1}^M \lambda_k \beta_k \beta_k^H = \mathbf{B} \mathbf{\Lambda} \mathbf{B}^H \quad (2.3.32)$$

where

$$\mathbf{B} = [\beta_1, \beta_2, \dots, \beta_K, \beta_{K+1}, \dots, \beta_M] \quad (2.3.33)$$

$$\mathbf{\Lambda} = \text{diag}[\lambda_1, \lambda_2, \dots, \lambda_K, \sigma^2, \sigma^2, \dots, \sigma^2] \quad (2.3.34)$$

where $\text{diag}[\cdot]$ denotes a diagonal matrix with $[\cdot]$ elements on the main diagonal. Note also that eigenvectors, β_k , are orthonormal to each other, which results in

$$\mathbf{B} \mathbf{B}^H = \mathbf{I}_M \quad (2.3.35)$$

where $\mathbf{I}_M$ is an $M \times M$ identity matrix.

Moreover, for any $i > K$, from (2.3.31) and (2.3.32), we have

$$\mathbf{R} \beta_i = \lambda_i \beta_i = \sigma^2 \beta_i \quad (2.3.36)$$

But we also have

$$\mathbf{R} \beta_i = (\mathbf{S} \mathbf{R}_u \mathbf{S}^H + \sigma^2 \mathbf{I}) \beta_i \quad (2.3.37)$$

and together they imply

$$\mathbf{S} \mathbf{R}_u \mathbf{S}^H \beta_i = 0 \quad (2.3.38)$$

which further lead to the following conclusion [5]

$$\mathbf{S}^H \beta_i = 0 \iff \beta_i^H \mathbf{s}(\theta_k) = 0, \quad K+1 \leq i \leq M, 1 \leq k \leq K \quad (2.3.39)$$

The last step follows from the full rank property of $\mathbf{S}$ and $\mathbf{R}$. Stated in words, eigenvectors associated with the repeating lowest eigenvalue of $\mathbf{R}$ are orthogonal to the steering vectors corresponding to the actual directions of arrival. This remarkable observation forms the cornerstone for basically all the eigenvector-based subspace DOA algorithms.

To view this geometrically, received signals, $\mathbf{x}(t)$ in (2.1.3), can be viewed as an M dimensional Hilbert space H over the complex field C . Eigenvectors $\beta_{K+1}, \beta_{K+2}, \dots, \beta_M$ associated with the lowest eigenvalues together span an $M - K$ dimensional subspace N of H and every steering vector corresponding to an actual arrival direction is orthogonal to this subspace. Since the K signal steering vectors are themselves linearly independent, the K dimensional “signal” subspace S spanned by these actual signal vectors is orthogonal to the subspace N . Since H can always be written as the direct sum of any finite dimensional subspace and its orthogonal complement [6], we have

$$H = S \oplus N \quad (2.3.40)$$

where

$$S = \text{span} (\mathbf{s}(\theta_1), \mathbf{s}(\theta_2), \dots, \mathbf{s}(\theta_K)) = \text{span} (\beta_1, \beta_2, \dots, \beta_K) \quad (2.3.41)$$

$$N = \text{span} (\beta_{K+1}, \beta_{K+2}, \dots, \beta_M) \quad (2.3.42)$$

Note that orthonormal eigenvectors corresponding to the largest K eigenvalues span the same subspace as steering vectors associated with actual directions of arrival do. Thus, we name this subspace S “signal” subspace and since subspace N is the orthogonal complement of S , it may be fittingly designed as the “noise” subspace.

It is defined

$$\mathbf{E}_s = [\beta_1, \beta_2, \dots, \beta_K] \quad (2.3.43)$$

$$\mathbf{E}_n = [\beta_{K+1}, \beta_{K+2}, \dots, \beta_M] \quad (2.3.44)$$

where $\mathbf{E}_s$, $\mathbf{E}_n$ denote signal eigen matrix and noise eigen matrix, respectively.

2.4 Conventional Method: Capon's Minimum Variance Estimator

The next two sections will provide a detailed overview of the various algorithms available for estimation of directions of arrival of a radio signal using antenna array. The use of an adaptive array system is critical in position location applications [7], and given mandate, for instance, to require 125m accuracy on wireless emergency calls by the FCC [8] [9], there is an intense interest in determining the Directions of Arrival (DOA) of RF signals in wireless systems.

DOA estimation algorithms using antenna array can be broadly divided into two categories, i.e., conventional techniques and eigenvector-based subspace methods. Conventional methods for DOA estimation are based on the concept of beamforming and null-steering and do not exploit the the model of the received signal vector $\mathbf{x}(t)$ or the statistical model of the signals and noises, unlike eigenvector-based techniques. Given the knowledge of the array manifold, they electronically steer beams in all possible directions and look for peaks in the output power. Conventional methods include Delay-and-Sum method, Capon's Minimum Variance method, Linear Prediction method and so on. In this section, Capon's Minimum Variance method is introduced for its ease of implementation.

Capon's minimum variance technique attempts to form a beam in the desired look direction while simultaneously trying to form nulls in the directions of interfering signals. This technique minimizes the contribution of the undesired interferences by minimizing the output power while maintaining the gain along the look direction to be constant, usually unity. The expression for array output power in (2.3.20) plays a central role in Capon minimum variance estimator. The formulation of Capon algorithm is given in [10], i.e.,

$$\min_{\mathbf{w}} \quad \mathbf{w}^H \mathbf{R} \mathbf{w} \quad (2.4.45)$$

$$\text{subject to} \quad |\mathbf{w}^H \mathbf{s}(\theta)| = 1 \quad (2.4.46)$$

where $|\cdot|$ is the absolute sign of a scalar. The array weight vector, $\mathbf{w}$, obtained from the above is called the minimum variance distortionless response (MVDR) beamformer weight since, for a particular look direction, it minimizes the variance (average power) of the output signal while passing the signal arriving in the look direction without distortion (unity gain).

The Capon template given above represents a constrained optimization problem which can be solved using the method of Lagrange multipliers [11] [12]. This approach converts the constraint optimization problem into an unconstrained one, thereby allowing the use of Least Square (LS) techniques to determine the solution. Using a Lagrange Multiplier, the weight vector that satisfies (2.4.45) and (2.4.46) can be shown to be [11],

$$\mathbf{w} = \frac{\mathbf{R}^{-1} \mathbf{s}(\theta)}{\mathbf{s}^H(\theta) \mathbf{R}^{-1} \mathbf{s}(\theta)} \quad (2.4.47)$$

Now the array output power can be given, by Capon's spatial spectrum,

$$P_{\text{Capon}} = \mathbf{w}^H \mathbf{R} \mathbf{w} = \frac{1}{\mathbf{s}^H(\theta) \mathbf{R}^{-1} \mathbf{s}(\theta)} \quad (2.4.48)$$

By computing and plotting Capon's spectrum over the whole range of θ , i.e. scan or search over Field of View (FOV) of the array, the directions of arrivals of signals can be estimated by locating the peaks in the spectrum.

This procedure is sometimes mistakenly referred to as the "maximum likelihood method", which is another type of DOA estimator. This is because for any choice of θ , P_{Capon} is the maximum likelihood estimate of the power of a signal arriving from the direction θ in the presence of white Gaussian noise having arbitrary spatial characteristics.

Capon's method suffers from some disadvantages as well. First of all, it fails if other signals that are correlated with the signal-of-interest are present because the correlated components may be combined destructively in the process of minimizing the output power. Secondly, Capon's method requires the computation of a matrix inverse, which can be expensive for large scale arrays. Thirdly, although Capon's method has relatively higher resolution than other types of conventional DOA estimators, its resolution capability is normally still poor, especially compared to eigenvector-based subspace algorithms.

2.5 Subspace Methods for DOA Estimation

2.5.1 MUSIC Algorithm

Though conventional methods such as Capon's minimum variance method are often successfully and have been widely used, these methods have some fundamental limitations in resolution. Most of these limitations arise because they do not exploit the structure of the input data model given in (2.3.23). Schmidt [13] and Bienvenu and Kopp [14] were the first to exploit the structure of a more accurate data model

for the case of sensor arrays of arbitrary form. Schmidt derived a complete geometric solution to the DOA estimation problem in the absence of noise, and extended the geometric concepts to obtain a reasonable approximation to the solution in the presence of noise. The technique proposed by Schmidt is called the Multiple Signal Classification (MUSIC) algorithm, and it has been thoroughly investigated ever since its inception [13]- [20].

The MUSIC algorithm proposed by Schmidt in 1979 basically depends on the orthogonality between noise subspace and signal steering vectors. We have, in Section 2.3.3, derived this important property in details. From (2.3.28), (2.3.39) and (2.3.44), we have

$$\mathbf{s}(\theta_k)^H \mathbf{E}_n \mathbf{E}_n^H \mathbf{s}(\theta_k) = 0 \quad (2.5.49)$$

where θ_k 's correspond to true directions of desired signals.

Thus, the DOAs of the multiple incident signals can be estimated by locating the peaks of a MUSIC spatial spectrum given by

$$D_{MUSIC}(\theta) = \frac{1}{\mathbf{s}^H(\theta) \mathbf{E}_n \mathbf{E}_n^H \mathbf{s}(\theta)} = \frac{1}{\|\mathbf{E}_n^H \mathbf{s}(\theta)\|^2} \quad (2.5.50)$$

where D represents a direction finding function that is different from the power estimate function in the case of Capon method. Orthogonality between signal steering vector and noise subspace will minimize the denominator and hence will give rise to the peaks in the MUSIC spectrum to indicate the directions of impinging signals on the array. And the number of largest peaks ought to correspond to the number of signals present in the field.

The procedure of MUSIC algorithm may be summarized as follows,

1. Collect a number of input samples within a time frame and estimate array output covariance matrix.

2. Perform eigen-decomposition on the estimated output covariance matrix.
3. Estimate the number of signals by subtracting the number of smallest eigenvalues.
4. Compute the MUSIC spectrum given in (2.5.50) and search over array field of view.
5. Locate the largest peaks to obtain estimates of directions of arrival.

It should be noted that, unlike the conventional methods, the MUSIC spatial spectrum does not estimate the signal power associated with each arrival angle. Instead, when the ensemble average of the array input covariance matrix is known exactly, under uncorrelated and identical noise conditions, the peaks of D_{MUSIC} are guaranteed to correspond to the true directions of arrival. Since these peaks are distinct irrespective of the actual separation between arrival angles, in principle, with perfect array calibration, these estimators can distinguish and resolve arbitrarily closely spaced signals.

2.5.2 Minimum Norm Method

The geometric concept upon which MUSIC algorithm is founded forms the basis for a much broader class of eigenvector-based subspace algorithms [21]. Apart from MUSIC, the primary contributions to the subspace based algorithms also include the Minimum Norm (MN) method proposed by Kumaresan and Tufts [22].

The minimum-norm property and a better motivation for the algorithm was presented in detail in [22]. In Minimum Norm algorithm, Kumaresan and Tufts propose to use a single vector in the noise subspace to conduct DOA estimations. This vector,

serving as the array weight vector $\mathbf{w}$, can be computed via the following formulation

$$\min_{\mathbf{w}} \quad \mathbf{w}^H \mathbf{w} \quad (2.5.51)$$

$$\text{subject to} \quad \mathbf{E}_s^H \mathbf{w} = \mathbf{0} \quad (2.5.52)$$

$$\boldsymbol{\delta}^H \mathbf{w} = 1 \quad (2.5.53)$$

where $\mathbf{E}_s$ represents the signal subspace matrix defined in (2.3.43) and $\mathbf{0}$ represents a zero vector of proper dimension. Note $\boldsymbol{\delta}$ is a vector consisting of a “1” and zeros and its dimension is determined by the number of array sensor elements. Typically,

$$\boldsymbol{\delta} = [1, 0, 0, \dots]^T \quad (2.5.54)$$

The array weight vector can be obtained by solving the above quadratic equation and the solution is provided in [23]. Thereby, DOAs of impinging signals can be estimated by locating the peaks in the following spectrum

$$D_{MN}(\theta) = \frac{1}{|\mathbf{w}^H \mathbf{s}(\theta)|^2} \quad (2.5.55)$$

Minimum Norm algorithm can alternatively be interpreted as a beamformer that produces nulls at the directions of arrival. As it is applicable for a uniform linear array, the optimization problem to solve for the array weight vector may be transformed to a polynomial rooting problem, which will be elaborated in Chapter 3. It is also noted that in minimum norm algorithm, only one vector in noise subspace is utilized to estimate DOAs. This is different from MUSIC algorithm where the whole noise subspace is utilized. As a matter of fact, the solution to $\mathbf{w}$ is actually a weighted noise subspace, thereby, minimum norm algorithm is sometimes referred to in the literature as a weighted MUSIC algorithm [23]. Comparisons of MUSIC and Minimum Norm algorithms are contained in [24].

2.5.3 Unity Response Algorithm

Unity Response (UR) algorithm was proposed in [25]. A similar approach termed as “CLOSET” algorithm was also proposed at about the same time in [26]. The basic idea of UR algorithm is to minimize the mean-square value of the difference between the actual array response and the desired unity response in certain directional regions while imposing zero-value directional eigenvector constraints obtained from the correlation matrix. The mathematical expression for UR estimator template is given in [25]

$$\min_{\mathbf{w}} \int_{\theta} |H(\theta) - e^{j\tau(\theta)}|^2 d\theta \quad (2.5.56)$$

$$\text{subject to} \quad \mathbf{E}_s^H \mathbf{w} = \mathbf{0} \quad (2.5.57)$$

where θ in (2.5.56) indicates the integration is over field of view (FOV) and $H(\theta)$ is termed array response at angle θ , defined as

$$H(\theta) = \mathbf{w}^H \mathbf{s}(\theta) \quad (2.5.58)$$

It is noted that $\tau(\theta)$ in (2.5.56) is a quantity that can be optimized. However, the optimization criteria and technique are not certain. It is initially suggested in [25] that $\tau(\theta)$ may take the following forms

$$\tau(\theta) = \tau_0 \quad \text{for some constant } \tau_0 \quad (2.5.59)$$

$$\tau(\theta) = a \sin(\theta) \quad \text{for some real quantity } a \quad (2.5.60)$$

With the array weight vector, $\mathbf{w}$, computed from the above quadratic problem, the direction finding function is written below

$$D_{UR}(\theta) = \frac{1}{|H(\theta)|^2} = \frac{1}{|\mathbf{w}^H \mathbf{s}(\theta)|^2} \quad (2.5.61)$$

Unity Response method turns out to be an algorithm in between renowned MUSIC and MN algorithms. It results in SNR spectral-resolution thresholds close to Minimum-norm (MN) algorithm and for some array geometries, substantially lower; also, its peak location variance is observed to be comparable with that of MUSIC algorithm, and higher than MN's, therefore, with resolution close to MN's and location estimation variance comparable to MUSIC's, UR approach represents one of the significant contribution to element-space spatial-spectrum estimation [26]. Apart from the above, variation of the integration range in (2.5.56) can result in different performance behavior. Generally speaking, an integration over array field of view can bring the performance of UR estimator close to MN estimator, whereas an integration at each and every bearing angle can push its performance back to that of MUSIC estimator; moreover, the spatial regions of interest can be integrated repeatedly for more than one time so that emphasis is given to flatten the response over these regions [25]. Another advantage that UR estimator possess over MN estimator is its flexibility. Not only is it because UR estimator is applicable to arbitrary array geometry, but also it is because variables in UR template can be adjusted accordingly to different applications, as we can see in the next chapters.

Chapter 3

An Averaged Approach to Phase Optimization in MN Algorithm & its Extension to UR Algorithm

3.1 Issue of Phase Optimization in MN Estimator

Direction-of-arrival (DOA) estimation is one of the major research topics in the field of array signal processing. In recent developments of DOA estimations, attention has mainly been focused on tailoring existing algorithms for implementation purpose [27]- [38]. However, there still are some remaining issues for basic DOA estimation theories, one of which falls in the area of phase optimization. As in some DOA algorithms, there are uncertainties concerning selection of the optimum phase quantity whose solutions are not yet found. This means that the issue of phase optimization in DOA estimation is still a gap to be filled.

It is stated in [39] that for the case of Linear Predictive (LP), one of the issues concerns the selection of reference sensor whose output is predicted by merging all

other sensor's outputs. The choice of reference sensor can have dramatic effects on the spatial spectrum. No one yields consistently "best" results. A general guideline is also suggested in [39] that for linear arrays, two choices are evident: a sensor located at either end or one (if present) located at the array's phase center. In this case, one end reference is usually chosen. For other array geometries, an "end" may not be defined; thus, a sensor near the geometry center is usually selected as the reference. This guideline works well for some cases, but not for all. When this phase issue comes to Minimum-norm (MN) algorithm, it is reflected in the proper choice of δ in (2.5.54). It is indicated in [40] that in MN algorithm, constraint vectors having their elementary entry, 1, in different positions can result in different array responses for the same signal scenario, but the best choice is yet to be found. It is also noted that such need for phase optimization is not a problem in Capon [41] or MUSIC [23], since the phase terms in these algorithms offset each other and disappear eventually in the mathematical derivations. Not only in the classic DOA estimation algorithm exists the issue of phase optimization, but it is also present in the recently proposed DOA algorithms, one of which is Unity-response (UR) algorithm. As in the MN algorithms, different formations of $\tau(\theta)$ in (2.5.56) can lead to different array responses in UR algorithm.

These optimization issues are related, but a general solution is difficult to obtain due to the fact that the basic formulations where these phase quantities are embedded vary diversely. In this chapter, MN algorithm is investigated and an averaged method is proposed using the concept of polynomial rooting. Subsequently, the proposed method is extended to UR algorithm and statistical performance analysis is made. Computer simulations are conducted to verify the results.

3.2 Proposition of the Averaged Method to MN Algorithm

The issue of phase optimization in Minimum-norm algorithm had been stated in the previous section and array output in MN algorithm takes the same form as that of (2.5.58) in UR estimator. Consider in a case of a uniform linear array (ULA) of M elements, the first element of which is the reference sensor, array output of a MN estimator can be written as follows

$$H(\theta) = \mathbf{w}^H \mathbf{s}(\theta) = \sum_{i=1}^M w_i e^{j(i-1)\pi \cos(\theta)} \quad (3.2.1)$$

where w_i is the i^{th} element in array weight vector $\mathbf{w}$ and is essentially the weight of i^{th} array sensor. If further, define the following

$$z = e^{j\pi \cos(\theta)}, |z| = 1 \quad (3.2.2)$$

then,

$$H(\theta) = \sum_{i=1}^M w_i z^{i-1} \quad (3.2.3)$$

Therefore, the array output response can be interpreted as a polynomial and the task of finding direction-of-arrival is equivalent to solving for the roots of the polynomial. This explains the formulation of Minimum-norm algorithm from another angle and provide means by which phase issue in MN template can possibly be resolved.

Consider again that there are K uncorrelated signal sources present in the field. Kumaren and Tufts, inventors of Minimum-norm algorithm, hypothesized to use a single vector in noise subspace that can result in an accurate estimation [23]. This is different from MUSIC algorithm where the whole noise subspace is utilized. In MN algorithm, there is only one single vector in noise subspace used for estimation.

Figure 3.1: Distribution of “signal” zeros and “noise” zeros

Nevertheless, it is anticipated to produce accurately K polynomial roots with magnitude being unity, also termed “signal” zeros as in [42], for the signal impinging directions and in the mean time, to regulate the rest of $M - K$ polynomial roots, termed “noise” zeros or “extraneous” zeros, in a such way that it is less likely to generate spurious spikes. This concept is explained in detail in [42]. Also, one can refer to Appendix A.1 for the concept of signal and noise zeros. However, from the study of Minimum-norm algorithm, it can be seen that there are ripples commonly seen in the noise floor of the array output plot, especially in the presence of noise. Despite the fact it can correctly produce “signal” zeros to indicate directions-of-arrival using MN formulation, there are often spurious spikes produced too. The coarseness in noise floor is due to the effects of “noise” zeros.

In Figure 3.1, an example of zeros’ distribution is shown. Signal zeros are denoted

by cross, noise zeros in the absence of noise are denoted by black dots and noise zeros in the presence of noise are denoted by small circles.

It is cited in [42], that in the case noise is not considered, by the formulation of MN algorithm, K “signal” zeros can be found on the unity circle, provided that $M > K$; and more importantly, the remaining $M - K$ “noise” zeros fall within the unity circle independent of the K “signal” zeros locations and tend to be uniformly distributed, as the dotted circle in Figure 3.1.

However, the above is true under the assumption that signal sources are not corrupted by noise. If there is noise in the data, the “noise” zeros tend to fall closer or outside the unity circle as shown in Figure 3.1, which are usually observed as spurious spikes in the spectral spectrum estimation plot. This is because as “noise” zeros get closer to some points on the unity circle, a very small value can possibly be produced in the spatial directions corresponding to these points. As a result, these spatial directions are viewed as spurious spikes.

It is observed the distribution of “noise” zeros, when noise is taken into account, varies as vector $\boldsymbol{\delta}$ defined in (2.5.54) has its single 1 in different entries. The exact relation between “noise” zeros’ distribution and position of 1 in $\boldsymbol{\delta}$ is so far unknown. Assumption has herein been made that the averaged superposition of these distributions stick closely to the dotted circle in Figure 3.1, therefore, it is proposed the averaged method, in which responses of MN estimators of 1s in different entries in $\boldsymbol{\delta}$ are added and averaged. The proposed method turns out to be simple but effective and tested by tremendous simulation practice which indicates there is a much less probability that a common spurious spike can occur between differently modified MN estimators (1s in different entries of $\boldsymbol{\delta}$). In mathematical formulation, the spatial spectrum can be expressed as,

$$P_{avg}(\theta) = |H_{avg}(\theta)|^2 = \frac{1}{N} \sum_{i=1}^N |H_{MN_i}(\theta)|^2 = \frac{1}{N} \sum_{i=1}^N |\mathbf{w}_i^H \mathbf{s}(\theta)|^2 \quad (3.2.4)$$

where $P_{avg}(\theta)$ indicates the power spectrum is considered. N is the number of MN estimators considered and H_{MN_i} stands for the response of one MN estimator in which δ has its single 1 in i^{th} entry. The corresponding weight vector is denoted by $\mathbf{w}_i$.

The proposed averaged method can also be explained by referring to the formulation of Linear-prediction (LP) algorithm. In LP algorithm, one sensor's output is estimated using linear combinations of the remaining sensors' outputs and the mean-square error, i.e. the square of the error between the estimate and the actual output, is minimized [39]. Now in the case of Minimum-norm algorithm, responses of MN estimators with δ having 1 in different entries are summed and averaged, which is analogous to estimating each sensor's expected output in turn using LP estimator and the results are subsequently averaged. Apparently, the outcome is more accurate.

3.3 Simulation and Observation

A comparison between single MN estimator, MUSIC estimator and proposed averaged method is made for a signal scenario where there are two signals present in the field impinging from -20° and 40° directions, respectively. The simulation is carried out for a uniform linear array of 10 sensor elements with half wavelength spacing. SNR is 10dB and the number of snapshots taken is 100. The result is shown in Figure 3.2.

It is clearly shown in Figure 3.2, that the number of ripples and spikes in the noise floor using averaged MN estimators is much less than using single MN estimator alone. It is also observed that possibility for spurious spikes is greatly reduced so that the overall performance of averaged MN estimators is very close to that of MU-

Figure 3.2: Comparison between MN, MUSIC and Averaged MN estimators

SIC estimator. As a matter of fact, the more MN estimators considered in averaged method, the more likely its performance approaches MUSIC estimator, which can be mathematically explained as follows.

Rewrite (3.2.4) as shown below,

$$P_{avg}(\theta) = |H_{avg}(\theta)|^2 = \frac{1}{N} \sum_{i=1}^N |H_{MN_i}(\theta)|^2 = \frac{1}{N} \sum_{i=1}^N \mathbf{s}^H(\theta) \mathbf{w}_i \mathbf{w}_i^H \mathbf{s}(\theta) \quad (3.3.5)$$

In a matrix form, (3.3.5) can be expressed as,

$$P_{avg}(\theta) = |H_{avg}(\theta)|^2 = \frac{1}{N} (\mathbf{s}^H(\theta) \mathbf{W}_d \mathbf{W}_d^H \mathbf{s}(\theta)) \quad (3.3.6)$$

where $\mathbf{W}_d$ is a matrix whose columns are array weight vectors of single MN estimators,

$$\mathbf{W}_d = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_N] \quad (3.3.7)$$

Expanding $\mathbf{W}_d$ yields

$$\mathbf{W}_d = \begin{pmatrix} 1 & \epsilon_{12} & \epsilon_{13} & \cdots & \epsilon_{1N} \\ \epsilon_{21} & 1 & \epsilon_{23} & \cdots & \epsilon_{2N} \\ \epsilon_{31} & \epsilon_{32} & 1 & \cdots & \epsilon_{3N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \epsilon_{(M-1)1} & \epsilon_{(M-1)2} & \epsilon_{(M-1)3} & \cdots & 1 \\ \epsilon_{M1} & \epsilon_{M2} & \epsilon_{M3} & \cdots & \epsilon_{MN} \end{pmatrix} \quad (3.3.8)$$

where $N \leq M$ and ϵ_{ij} are numbers whose absolute value are much smaller than 1, i.e. $|\epsilon_{ij}| \ll 1$. This can be justified by the formulation of MN estimator in which norm of array weight vector, $\mathbf{w}^H \mathbf{w}$, is minimized as in (2.5.51). Consequently, unless in some extreme cases, e.g., two signals present in the field are extremely closely spaced, it will result in weight vectors consisting of a single 1 and other small values much less than 1 in most of the signal scenarios.

It is observed that all the 1s are on the main diagonal with the rest of the elements ϵ_{ij} sufficiently small. It can be approximated column vectors in $\mathbf{W}_d$ are independent to each other. Moreover, since each single column, $\mathbf{w}_i$, is a vector in noise subspace, $\mathbf{W}_d$ can therefore resemble the whole noise subspace, $\mathbf{E}_n$ as in (2.3.44), to a great extent. If $\mathbf{W}_d$ is replaced by $\mathbf{E}_n$, (3.3.6) is rewritten as,

$$P_{avg}(\theta) = |H_{avg}(\theta)|^2 = \frac{1}{N} (\mathbf{s}^H(\theta) \mathbf{E}_n \mathbf{E}_n^H \mathbf{s}(\theta)) \quad (3.3.9)$$

where the formulation in the parentheses is exactly the well-known MUSIC estimator.

3.4 Averaged Method used in UR Algorithm

3.4.1 Equivalence of Phase Issues in MN and UR Templates

It is stated that UR estimator can be considered as a generalization of MN estimator if the integration in UR template is over the whole array field of view [26]. It is also stated in [25] that UR estimator can yield the same unity beam pattern and more importantly the same spatial phase response in quiescent conditions where there is no signals present in the field. Thus, for both UR and MN estimators, when eigenvector constraints are added into the formulation, the estimators produce nulls at the signal direction and the at the same time, they try to maintain the quiescent response, in the minimum mean-square error sense, away from the signal directions [25]. The following example can elaborate the above concept.

Consider again a uniform linear array of M sensors with half-wavelength spacing and for convenience, the 1st array sensor is regarded as the reference sensor. It is known in this case that the steering vector at direction θ is,

$$\mathbf{s}(\theta) = [1, e^{j\pi \cos(\theta)}, e^{j2\pi \cos(\theta)}, \dots, e^{j(M-1)\pi \cos(\theta)}]^T \quad (3.4.10)$$

It is found that under quiescent conditions, if $\tau(\theta)$ takes the form of $m\pi \cos(\theta)$, where $m = 0, 1, 2, \dots, M-1$, the weight vectors and spatial responses obtained by using UR estimator are identical to those obtained by using MN estimator in which the single 1 in vector $\boldsymbol{\delta}$ in (2.5.54) takes m^{th} entry. Therefore, the phase issue of $\tau(\theta)$ optimization in UR estimator can equivalently be converted to that in MN estimator.

3.4.2 Averaged Method Extended to UR Algorithm

It has been explained previously that the study of phase quantity in UR estimator can be considered as the study of 1's position in δ in MN estimator. Thus, the averaged method initially proposed for MN algorithm is also applicable to UR estimator. Moreover, it is found that computation load of DOA estimation can be possibly reduced, if using UR estimator template. This is due to the flexibility of UR estimator over MN estimator, since the phase quantity in UR estimator can be “continuous” whereas 1's position in MN estimator is “discrete”.

Consider the array of M sensors whose steering vector $\mathbf{s}(\theta)$ is composed as follows,

$$\mathbf{s}(\theta) = [s_1(\theta), s_2(\theta), \dots, s_i(\theta), \dots, s_M(\theta)]^T \quad (3.4.11)$$

Using the proposed averaged method, the estimator is expressed as,

$$P_{avg}(\theta) = |H_{avg}(\theta)|^2 = \frac{1}{N} \sum_{i=1}^N |H_{UR_i}(\theta)|^2 = \frac{1}{N} [\mathbf{s}^H(\theta) (\sum_{i=1}^N \mathbf{w}_i \mathbf{w}_i^H) \mathbf{s}(\theta)] \quad (3.4.12)$$

where H_{UR_i} stands for the array response of a single UR estimator whose phase quantity, $\tau_i(\theta)$, is chosen in such a way that it satisfies the following,

$$e^{j\tau_i(\theta)} = s_i(\theta) \quad (3.4.13)$$

and $\mathbf{w}_i$ is the corresponding weight vector. N is the number of UR estimators taken into account.

The closed form solution for $\mathbf{w}_i$ is readily given in [25], where

$$\mathbf{w}_i = \mathbf{Q}^{-1} \mathbf{E}_s (\mathbf{E}_s^H \mathbf{Q}^{-1} \mathbf{E}_s)^{-1} \mathbf{E}_s^H \mathbf{Q}^{-1} \mathbf{P}_i - \mathbf{Q}^{-1} \mathbf{P}_i \quad (3.4.14)$$

where

$$\mathbf{Q} = \int_{\theta} \mathbf{s}(\theta) \mathbf{s}^H(\theta) \quad (3.4.15)$$

$$\mathbf{P}_i = \int_{\theta} \mathbf{s}(\theta) e^{-j\tau_i(\theta)} = \int_{\theta} \mathbf{s}(\theta) s_i^*(\theta) \quad (3.4.16)$$

(3.4.12) can be further simplified if the array manifold $\mathbf{S}_{array}$ is defined below,

$$\begin{aligned} \mathbf{S}_{array} &= [\mathbf{s}(\theta_1), \mathbf{s}(\theta_2), \dots, \mathbf{s}(\theta_l)] \\ &= \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \\ \mathbf{a}_M \end{bmatrix} \end{aligned} \quad (3.4.17)$$

where the number of columns of $\mathbf{S}_{array}$ is determined by the range of field of view and spatial resolution. And $\mathbf{a}_i$ is introduced herein to simplify the calculation, which essentially represents the i^{th} row vector in array manifold.

Further define matrix $\mathbf{F}$, where

$$\mathbf{F} = \mathbf{Q}^{-1} \mathbf{E}_s (\mathbf{E}_s^H \mathbf{Q}^{-1} \mathbf{E}_s)^{-1} \mathbf{E}_s^H \mathbf{Q}^{-1} - \mathbf{Q}^{-1} \quad (3.4.18)$$

Thus, (3.4.14)-(3.4.16) can be written into,

$$\mathbf{w}_i = \mathbf{F} \mathbf{P}_i \quad (3.4.19)$$

$$\mathbf{Q} = \mathbf{S}_{array} \mathbf{S}_{array}^H \quad (3.4.20)$$

$$\mathbf{P}_i = \mathbf{S}_{array} \mathbf{a}_i^H \quad (3.4.21)$$

Therefore,

$$\begin{aligned} \sum_{i=1}^N \mathbf{w}_i \mathbf{w}_i^H &= \sum_{i=1}^N \mathbf{F} \mathbf{S}_{array} (\mathbf{a}_i^H \mathbf{a}_i) \mathbf{S}_{array}^H \mathbf{F}^H \\ &= \mathbf{F} \mathbf{S}_{array} (\mathbf{A}^H \mathbf{A}) \mathbf{S}_{array}^H \mathbf{F}^H \end{aligned} \quad (3.4.22)$$

where $\mathbf{A}$ is a sub-matrix of array manifold, since $N \leq M$,

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \\ \mathbf{a}_M \end{bmatrix} \quad (3.4.23)$$

In (3.4.22), it is clear that $\mathbf{A}$ can be easily computed using offline processing. Array manifold, $\mathbf{S}$, is known to an array geometry. The only variable $\mathbf{F}$ is a function of signal subspace, $\mathbf{E}_s$. And the value of $\mathbf{F}$ requires to be calculated one time only, which is compared to the case of MN estimators in which N times calculations are necessary. Apparently, substituting (3.4.22) in (3.4.12) will lead to a reduction in terms of computation load. This reflects the advantage in flexibility using UR estimator over MN estimator.

3.5 Statistical Performance Analysis of UR Algorithm

In the literature, statistical performance analysis for MN and MUSIC algorithms have been exploited and compared by researchers. In this section, statistical performance analysis of single UR estimator is investigated and subsequently, mean-

squared error (MSE) in DOA estimation is derived and results are compared.

The statistical performance analysis for Minimum-norm algorithm was conducted in [43], following two main steps. Firstly, the mean-squared error in the roots of the polynomial is examined, since direction finding in MN can be interpreted into polynomial rooting as discussed in Section 4.2. Secondly, the mean-squared error in DOA estimation is derived. For Unity-response algorithm, the same procedure can be used to conduct the statistical performance analysis.

Before the mean-squared error of MN estimator can be given, the following equations need to be introduced for the case where a ULA of M sensors is considered in a scenario of K independent signals.

$$\mathbf{w}_{\text{MN}} = [1, \mathbf{g}^T]^T \quad (3.5.24)$$

$$\mathbf{E}_s = \begin{bmatrix} \mathbf{e}_1 \\ \mathbf{E}_2 \end{bmatrix} \quad (3.5.25)$$

$$\mathbf{s}_2(\theta_i) = [s_1(\theta), s_2(\theta), \dots, s_i(\theta), \dots, s_{M-1}(\theta)]^T \quad (3.5.26)$$

$$\mathbf{R} = \mathbf{B}\mathbf{A}\mathbf{B}^H = \mathbf{E}_s\mathbf{\Lambda}_s\mathbf{E}_s^H + \sigma_n^2\mathbf{I} \quad (3.5.27)$$

where $\mathbf{w}_{\text{MN}}$ is an array weight vector of an MN estimator. $\mathbf{g}$ is an $(M-1) \times 1$ vector extracted from $\mathbf{w}_{\text{MN}}$. And $\mathbf{e}_1$ represents the first row of $\mathbf{E}_s$. And $\mathbf{E}_2$ is a sub-matrix of dimension $(M-1) \times K$ from signal subspace. $\mathbf{s}_2(\theta_i)$ denotes a vector composed of the first $(M-1)$ elements of array steering vector corresponding to signal direction θ_i . $\mathbf{R}$ is covariance matrix with $\mathbf{\Lambda}$ and $\mathbf{\Lambda}_s$ being diagonal matrix with eigenvalues arranged on main diagonal in descending order. σ_n^2 denotes noise power. $\mathbf{I}$ is an $M \times M$ identity matrix.

Thus, the mean-squared error in the roots, $\overline{|\Delta z_i|_{MN}^2}$, derived in [43], is readily given below,

$$\overline{|\Delta z_i|_{MN}^2} = \frac{M}{M-1} S_{MN} \frac{\sigma_n^2}{T} (1 + \|\mathbf{g}\|^2) \left(\sum_{j=1}^K |\alpha_j|^2 \frac{\lambda_j}{(\lambda_j^s)^2} \right) \quad (3.5.28)$$

where

$$S_{MN} = \frac{M-1}{\prod_{j=1, j \neq i}^{M-1} |1 - z_j z_i^{-1}|^2} \quad (3.5.29)$$

$$\boldsymbol{\alpha} = \mathbf{E}_2^+ \mathbf{s}_2(\theta_i) \quad (3.5.30)$$

where S_{MN} is termed parameter sensitivity for MN estimator. z_i is the signal root corresponding to the signal direction and z_j 's denote the rest of $(M-1)$ polynomial roots; λ_j & λ_j^s are eigenvalues from $\mathbf{\Lambda}$ & $\mathbf{\Lambda}_s$, respectively; α_j s are elements of the vector $\boldsymbol{\alpha}$. T is the number of snapshots taken. The sign “+” denotes the Moore-Penrose pseudo-inverse. “ $\|\cdot\|$ ” denotes the norm of a vector and “ $\overline{|\cdot|^2}$ ” is the mean-squared operator.

With the mean-squared error in roots obtained, it was subsequently derived in [43] that the mean-squared error in DOA estimate is governed by the following equation

$$\overline{|\Delta\theta_i|_{MN}^2} = \left(\frac{1}{d\pi \cos(\theta_i)} \right)^2 \frac{\overline{|\Delta z_i|_{MN}^2}}{2} \quad (3.5.31)$$

where d is the normalized inter-sensor separation with respect to half-wavelength. The similar steps of derivation in [43] for MN algorithm can be applied to UR algorithm (as in Appendix A). It can result in the following expression of mean-squared error in roots,

$$\overline{|\Delta z_i|_{UR}^2} = S_{UR} \frac{\sigma_n^2}{T} \mathbf{w}_{UR}^H \mathbf{w}_{UR} \left(\sum_{j=1}^K |\gamma_j|^2 \frac{\lambda_j}{(\lambda_j^s)^2} \right) \quad (3.5.32)$$

where

$$S_{UR} = \frac{M}{\prod_{j=1, j \neq i}^{M-1} |1 - z_j z_i^{-1}|^2} \quad (3.5.33)$$

$$\boldsymbol{\gamma} = \mathbf{E}_s^H \mathbf{s}(\theta_i) \quad (3.5.34)$$

where S_{UR} is the sensitivity parameter. $\mathbf{w}_{MN}$ is the array weight vector computed, and γ_j 's are elements in $\boldsymbol{\gamma}$.

It is noted that in MN and UR estimators, the following approximation can be made,

$$1 + \|\mathbf{g}\|^2 \approx 1 \approx \mathbf{w}_{UR}^H \mathbf{w}_{UR} \quad (3.5.35)$$

$$\boldsymbol{\alpha} = \mathbf{E}_2^+ \mathbf{s}_2(\theta_i) \approx \mathbf{E}_2^H \mathbf{s}_2(\theta_i) \approx \mathbf{E}_s^H \mathbf{s}(\theta_i) = \boldsymbol{\gamma} \quad (3.5.36)$$

Comparing (3.5.28) with (3.5.32) via (3.5.29), (3.5.33), (3.5.35) and (3.5.36), it is easily observed that Minimum-norm and Unity-response algorithms have approximately the same statistical performance behavior in general case.

Lastly, substituting (3.5.32) into (3.5.31), it yields the mean-squared error in direction estimations, which is shown below,

$$\overline{|\Delta\theta_i|_{UR}^2} = \left(\frac{1}{d\pi \cos(\theta_i)} \right)^2 \frac{\overline{|\Delta z_i|_{UR}^2}}{2} \quad (3.5.37)$$

To test the validity of the above equation (3.5.37), a one-source situation is examined. For a single source case with signal power level P and noise power σ^2 , the number of sensor is M and the sensor spacing is half-wavelength, i.e. $d = 1$. If the number of snapshots taken is T , it can be simplified that in this case, MSE in direction estimation is,

$$\overline{|\Delta\theta_i|_{UR}^2} = \left(\frac{1}{\pi \cos(\theta_i)} \right)^2 \frac{2}{M} \left(\frac{\sigma^2}{MPT} \right) \quad (3.5.38)$$

In contrast, mean-squared error for root-MUSIC algorithm in the same signal scenario as stated above has been derived in [44], which is given in (3.5.39),

$$\overline{|\Delta\theta_i|_{MUSIC}^2} = \left(\frac{1}{\pi \cos(\theta_i)} \right)^2 \frac{6}{M(M+1)} \left(\frac{\sigma^2}{MPT} \right) \quad (3.5.39)$$

Figure 3.3: MSE in DOA estimation for UR and MUSIC estimators

Computer study has been conducted for the case where $M = 8$, $\theta_i = 18^\circ$ and $T = 100$. The various plots of MSE against SNR are summarized in the figures shown.

In Figure 3.3, the UR and root-MUSIC estimators are examined and compared for the single source case. It can be clearly observed that the simulated results adhere closely to the theoretical one. The theoretical derivation is in good agreement with computer simulations.

In Figure 3.4, computer study for averaged UR estimators are attached. It is indicated that averaged UR estimator behaves in between single UR and root-MUSIC, but superior over single UR estimator in the sense that it is more robust in combating noise. Moreover, as larger number of estimators are taken into account, its performance approaches that of MUSIC.

However, the enhanced robustness against noise is at the cost of reduced resolution capability. In terms of resolution, single MN or UR estimator has advantage over

Figure 3.4: MSE in DOA estimation for Averaged UR estimators

Figure 3.5: Examination of resolution of the 3 different estimators

the averaged method. Statistical study is attached in Figure 3.5.

In Figure 3.5, the plot is generated using the same ULA geometry as the previous cases. The signal scenario considered is as follows: there are two signals of unity power present in the field and they are symmetrically located with respect to the array broadside. In Figure 3.5, X-axis represents the angle separation between one of the signals and array broadside, whereas Y-axis represents the SNR value required to separate the two signals.

It can be seen when two signals are closely placed to each other, the required SNR value to resolve the two signals for averaged method is in between MUSIC and UR estimator, which implies its resolution capability is higher than MUSIC, but lower than single UR estimator. As the separation between signals become larger, the three curves start converging to the same value, which means any of the estimators can resolve the two signals with low SNR.

Chapter 4

Phase Selection using a Constant Modulus Approach

4.1 Optimization via “Phase-synchronization”

We have seen that Unity-Response algorithm, when integrated over field of view, has a very similar performance to that of Minimum-norm algorithm. In Chapter 4, an averaging approach is used to mitigate the phase effect in DOA estimations. In this chapter, we will propose, from the perspective of UR estimator, a constant modulus approach to further eliminate the effect of improper phase selection.

The idea behind Unity-Response algorithm is actually an extension of Notch filter applied in spatial domain. It targets to maintain array response as close to unity as possible over all the bearing directions, meanwhile, force zero response at signal impinging directions through the consideration of eigen-constraint. The template for UR estimator has been set out in (2.5.56)-(2.5.58), but for the ease of elaboration of this chapter, it is necessary to rewrite them into a summation form. Assuming in this chapter that the array Field of View (FOV) ranges from $[\theta_1, \theta_2, \dots, \theta_i, \dots, \theta_k]$,

the template for Unity-Response approach can be rewritten as

$$\min_{\mathbf{w}} \sum_{i=1}^k |H(\theta_i) - e^{j\tau(\theta_i)}|^2 \quad (4.1.1)$$

$$\text{subject to} \quad \mathbf{E}_s^H \mathbf{w} = \mathbf{0} \quad (4.1.2)$$

where k is the number of bearing angles counted in FOV. $\mathbf{E}_s$ is signal subspace. $\tau(\theta_i)$ is the phase quantity of interest to be studied. The phase term is a quantity that can be optimized, however, the criterion for its best choice has not been found [25]. The reason why UR algorithm is studied here is because $\tau(\theta_i)$ can be viewed as a continuous function and is more flexible than LP or MN algorithms where only discrete position number can be changed.

For simplicity of illustration, a linear array of 3 elements in a single signal scenario is considered and $\tau(\theta_i)$ takes the form of $a \cos(\theta_i)$, where a is a scalar quantity. The noiseless plot is generated in Figure 4.1 when a takes values of 0.5 and 1 and signal impinging from 120° . Evidently, one curve is not the translated version of the other. More plots can also be easily generated by using different values of a .

Two forms of $\tau(\theta_i)$ were recommended in [25], already given in (2.5.59) and (2.5.60), both of which are loosely suitable for general signal scenarios. However, in certain cases, the recommendation may lose its generality and result in undesired plots. In [26], $\tau(\theta_i)$ was described as the “desired response” and the issue of $\tau(\theta_i)$ selection was considered as “analogous to that of selecting a reasonable desired phase response for a FIR filter”. Likewise, undesired results can be generated in some signal scenarios. This is because the solutions proposed in both papers are pre-selected and independent of signal scenarios, such that value of $\tau(\theta_i)$ is not “adaptive” to the real signal environment. In this chapter, an iterative alternative solution is proposed, which is signal-dependent and able to have phase values adjusted based on the acquired signal information.

Figure 4.1: UR responses plotted using different phase values

From equation (4.1.1), it can be seen that array response, H , is constrained in two different ways, i.e., for $\tau(\theta_i)$ given certain form, firstly, the amplitude of H is designed to be as close to unity as possible despite the choice of $\tau(\theta_i)$ and secondly, the phase angle of H is implied to be as close to the value of $\tau(\theta_i)$ as possible. However, as far as DOA estimation is concerned, only the amplitude of H is critical to overall performance. The implied or “hidden” phase constraint can compromise the amplitude constraint and sometimes severely degrade the performance of Unity-Response algorithm. Therefore, a proper choice of $\tau(\theta_i)$ is important.

In this chapter, it is proposed that the concept of phase-synchronization in which it defines that value of the desired phase response, $\tau(\theta_i)$, is equal to the phase angle of the array response, $H(\theta_i)$, in which sense, $\tau(\theta_i)$ is “phase synchronized” to $H(\theta_i)$. Thus, (4.1.1) is reformulated as follows

$$\min_{\mathbf{w}} \sum_{i=1}^k |H(\theta_i) - e^{j\angle H(\theta_i)}|^2 \quad (4.1.3)$$

where $\angle\alpha$ refers to the phase angle of a complex quantity α .

From (4.1.3), Unity-Response algorithm can be interpreted in such a way that it targets to minimize over FOV the squared value of the difference between the actual array response and a unity circle on a complex plane while imposing zero-value directional eigenvector constraint. This is distinguished from the original intention of Unity-Response algorithm where the difference is made between array response and “desired response” which is essentially a predetermined signal-independent response. By designing $\tau(\theta_i)$ via the concept of phase-synchronization, it mitigates the effects of the implied phase constraint to its maximum extent. An obvious solution to (4.1.3) is thus far not available. However, since $\tau(\theta_i)$ is synchronized with $H(\theta_i)$, (4.1.3) can be converted to the following

$$\min_{\mathbf{w}} \sum_{i=1}^k ||H(\theta_i)| - 1|^2 \quad (4.1.4)$$

It is observed that (4.1.4) can be viewed as a typical Constant Modulus (CM) problem. Constant Modulus algorithm is a technique frequently used in blind adaptive signal processing and solutions can be derived. Nevertheless, eigen-constraint in (4.1.2) shall still be satisfied.

4.2 Iterative Phase Selections using a Constant Modulus Technique

A Least Square approach to CM problem was proposed by Agee using an extension of the method of nonlinear least squares, also known as Gauss’s Method [45]. Applying the extension of Gauss’s method in this case, if a cost function can be expressed in

the form,

$$F(\mathbf{w}) = \sum_{i=1}^k |g_i(\mathbf{w})|^2 = \|\mathbf{g}(\mathbf{w})\|^2 \quad (4.2.5)$$

where

$$\mathbf{g}(\mathbf{w}) = [g_1(\mathbf{w}), g_2(\mathbf{w}), \dots, g_k(\mathbf{w})]^T \quad (4.2.6)$$

then the cost function has a partial Taylor-series expansion with the sum-of-square form [10]

$$F(\mathbf{w} + \mathbf{d}) \approx \|\mathbf{g}(\mathbf{w}) + \mathbf{D}^H(\mathbf{w})\mathbf{d}\|^2 \quad (4.2.7)$$

where $\mathbf{d}$ is an offset vector, and

$$\mathbf{D}(\mathbf{w}) = [\nabla(g_1(\mathbf{w})), \nabla(g_2(\mathbf{w})), \dots, \nabla(g_k(\mathbf{w}))] \quad (4.2.8)$$

The gradient vector of $F(\mathbf{w} + \mathbf{d})$ with respect to $\mathbf{d}$ is as in [10]

$$\begin{aligned} \nabla_{\mathbf{d}}(F(\mathbf{w} + \mathbf{d})) &= 2 \frac{\partial F(\mathbf{w} + \mathbf{d})}{\partial \mathbf{d}^*} \\ &= 2 \frac{\partial [(\mathbf{g}(\mathbf{w}) + \mathbf{D}^H(\mathbf{w})\mathbf{d})^H (\mathbf{g}(\mathbf{w}) + \mathbf{D}^H(\mathbf{w})\mathbf{d})]}{\partial \mathbf{d}^*} \\ &= 2 \frac{\partial [\|\mathbf{g}(\mathbf{w})\|^2 + \mathbf{g}^H(\mathbf{w})\mathbf{D}^H(\mathbf{w})\mathbf{d} + \mathbf{d}^H\mathbf{D}(\mathbf{w})\mathbf{g}(\mathbf{w}) + \mathbf{d}^H\mathbf{D}(\mathbf{w})\mathbf{D}^H(\mathbf{w})\mathbf{d}]}{\partial \mathbf{d}^*} \\ &= 2[\mathbf{D}(\mathbf{w})\mathbf{g}(\mathbf{w}) + \mathbf{D}(\mathbf{w})\mathbf{D}^H(\mathbf{w})\mathbf{d}] \end{aligned} \quad (4.2.9)$$

Setting $\nabla_{\mathbf{d}}$ equal to zero, the offset that minimizes the cost function $F(\mathbf{w} + \mathbf{d})$ is

$$\mathbf{d} = -[\mathbf{D}(\mathbf{w})\mathbf{D}^H(\mathbf{w})]^{-1}\mathbf{D}(\mathbf{w})\mathbf{g}(\mathbf{w}) \quad (4.2.10)$$

Adding $\mathbf{d}$ to $\mathbf{w}$ results in a new weight vector that minimizes the cost function in (4.2.7). Therefore, the weight vector can be updated by [10]

$$\mathbf{w}(n+1) = \mathbf{w}(n) - [\mathbf{D}(\mathbf{w}(n))\mathbf{D}^H(\mathbf{w}(n))]^{-1}\mathbf{D}(\mathbf{w}(n))\mathbf{g}(\mathbf{w}(n)) \quad (4.2.11)$$

where n denotes the number of iteration. Using the result in (4.2.11), solutions to Constant Modulus problem expressed in (4.1.4) can be derived. Let,

$$\begin{aligned} F(\mathbf{w}) &= \sum_{i=1}^k ||H(\theta_i)| - 1|^2 \\ &= \sum_{i=1}^k ||\mathbf{w}^H \mathbf{s}(\theta_i)| - 1|^2 \end{aligned} \quad (4.2.12)$$

Comparing (4.2.12) with (4.2.5), it is seen that

$$g_i(\mathbf{w}) = |H(\theta_i)| - 1 = |\mathbf{w}^H \mathbf{s}(\theta_i)| - 1 \quad (4.2.13)$$

Substituting (4.2.13) into (4.2.6), it yields

$$\mathbf{g}(\mathbf{w}) = \begin{pmatrix} |H(\theta_1)| - 1 \\ |H(\theta_2)| - 1 \\ \vdots \\ |H(\theta_k)| - 1 \end{pmatrix} \quad (4.2.14)$$

The gradient of $g_i(\mathbf{w})$ is

$$\nabla(g_i(\mathbf{w})) = 2 \frac{\partial g_i(\mathbf{w})}{\partial \mathbf{w}^*} = \mathbf{s}(\theta_i) \frac{H^*(\theta_i)}{|H(\theta_i)|} = \mathbf{s}(\theta_i) e^{-j\tau(\theta_i)} \quad (4.2.15)$$

It is noted that the normalized array response, $\frac{H^*(\theta_i)}{|H(\theta_i)|}$, is equal to $e^{-j\tau(\theta_i)}$, since $\tau(\theta_i)$ is chosen in such a way that it is synchronized to the phase angle of array response. Substituting (4.2.15) into (4.2.8), $\mathbf{D}(\mathbf{w})$ can be written as,

$$\begin{aligned} \mathbf{D}(\mathbf{w}) &= [\mathbf{s}(\theta_1) e^{-j\tau(\theta_1)}, \mathbf{s}(\theta_2) e^{-j\tau(\theta_2)}, \dots, \mathbf{s}(\theta_k) e^{-j\tau(\theta_k)}] \\ &= \mathbf{S}_{array} \mathbf{R}_{diag} \end{aligned} \quad (4.2.16)$$

where $\mathbf{S}_{array}$ is the array manifold and equal to,

$$\mathbf{S}_{array} = [\mathbf{s}(\theta_1), \mathbf{s}(\theta_2), \dots, \mathbf{s}(\theta_k)] \quad (4.2.17)$$

And $\mathbf{R}_{diag}$ is a diagonal matrix whose diagonal entries are

$$\mathbf{r} = [e^{-j\tau(\theta_1)}, e^{-j\tau(\theta_2)}, \dots, e^{-j\tau(\theta_k)}]^T \quad (4.2.18)$$

Using (4.2.16) and (4.2.14), it yields,

$$\mathbf{D}(\mathbf{w})\mathbf{D}^H(\mathbf{w}) = \mathbf{S}_{array}\mathbf{R}_{diag}\mathbf{R}_{diag}^H\mathbf{S}_{array}^H = \mathbf{S}_{array}\mathbf{S}_{array}^H \quad (4.2.19)$$

and further, from (4.2.14)~(4.2.16),

$$\begin{aligned} \mathbf{D}(\mathbf{w})\mathbf{g}(\mathbf{w}) &= \mathbf{S}_{array}\mathbf{R}_{diag} \begin{pmatrix} |H(\theta_1)| - 1 \\ |H(\theta_2)| - 1 \\ \vdots \\ |H(\theta_k)| - 1 \end{pmatrix} \\ &= \mathbf{S}_{array} \begin{pmatrix} H^*(\theta_1) - e^{-j\tau(\theta_1)} \\ H^*(\theta_2) - e^{-j\tau(\theta_2)} \\ \vdots \\ H^*(\theta_k) - e^{-j\tau(\theta_k)} \end{pmatrix} \\ &= \mathbf{S}_{array}(\mathbf{y} - \mathbf{r}) \end{aligned} \quad (4.2.20)$$

where

$$\mathbf{y} = [H^*(\theta_1), H^*(\theta_2), \dots, H^*(\theta_k)]^T \quad (4.2.21)$$

Substituting (4.2.19) and (4.2.20) into (4.2.11), it yields,

$$\begin{aligned}
 \mathbf{w}(n+1) &= \mathbf{w}(n) - [\mathbf{S}_{array} \mathbf{S}_{array}^H]^{-1} \mathbf{S}_{array} (\mathbf{y}(n) - \mathbf{r}(n)) \\
 &= \mathbf{w}(n) - [\mathbf{S}_{array} \mathbf{S}_{array}^H]^{-1} \mathbf{S}_{array} \mathbf{S}_{array}^H \mathbf{w}(n) + [\mathbf{S}_{array} \mathbf{S}_{array}^H]^{-1} \mathbf{S}_{array} \mathbf{r}(n) \\
 &= [\mathbf{S}_{array} \mathbf{S}_{array}^H]^{-1} \mathbf{S}_{array} \mathbf{r}(n)
 \end{aligned} \tag{4.2.22}$$

Note in the above equation,

$$\mathbf{y}(n) = (\mathbf{w}^H(n) \mathbf{S}_{array})^H = \mathbf{S}_{array}^H \mathbf{w}(n) \tag{4.2.23}$$

$$\mathbf{r}(n) = [e^{-j\tau_n(\theta_1)}, e^{-j\tau_n(\theta_2)}, \dots, e^{-j\tau_n(\theta_k)}]^T = e^{j\angle \mathbf{y}(n)} \tag{4.2.24}$$

where $\tau_n(\theta_i)$ denotes the value of $\tau(\theta_i)$ at n^{th} iteration. The above is the iterative solution to (4.1.4) in Section 4.1. However, the weight obtained shall also meet the eigen-constraint that it is orthogonal to signal sub-space, i.e.(4.1.2) must still be satisfied.

4.3 Consideration of the Imposed Eigen-constraint

4.3.1 A Signal Subspace Matrix Approach

In this section, two methods are presented to solve the eigen-constraint that is additionally imposed on CM problem. There are different approaches using signal subspace matrix, $\mathbf{E}_s$, and noise subspace matrix, $\mathbf{E}_n$, respectively.

For any iterated result, $\mathbf{w}_0$, the vector closest to it and orthogonal to signal subspace

matrix, $\hat{\mathbf{w}}$, can be computed by the following,

$$\min_{\hat{\mathbf{w}}} \quad \|\hat{\mathbf{w}} - \mathbf{w}_0\|^2 \quad (4.3.25)$$

$$\text{subject to} \quad \mathbf{E}_s^H \hat{\mathbf{w}} = \mathbf{0} \quad (4.3.26)$$

The above are quadratic optimization equations, the solution of which can be referred to in Appendix B, the answer is provided below,

$$\hat{\mathbf{w}} = \mathbf{w}_0 - \mathbf{E}_s \mathbf{E}_s^H \mathbf{w}_0 = (\mathbf{I} - \mathbf{E}_s \mathbf{E}_s^H) \mathbf{w}_0 \quad (4.3.27)$$

where $\mathbf{I}$ is an identity matrix of proper dimension. In (4.3.27), it is noted that $\hat{\mathbf{w}}$ is always orthogonal to signal sub-space, given $\mathbf{w}_0$ is a non-zero vector. Thus, it is suggested array response in (4.3.27) be rewritten as,

$$H(\theta_i) = \hat{\mathbf{w}}^H \mathbf{s}(\theta_i) = \mathbf{w}_0^H (\mathbf{I} - \mathbf{E}_s \mathbf{E}_s^H) \mathbf{s}(\theta_i) = \mathbf{w}_0^H \mathbf{x}(\theta_i) \quad (4.3.28)$$

where

$$\mathbf{x}(\theta_i) = (\mathbf{I} - \mathbf{E}_s \mathbf{E}_s^H) \mathbf{s}(\theta_i) \quad (4.3.29)$$

With steering vector $\mathbf{s}(\theta_i)$ replaced by $\mathbf{x}(\theta_i)$, the exact same derivation in (4.2.12)-(4.2.24) in Section 4.2 can be repeated and similar results can be obtained as shown,

$$\mathbf{w}(n+1) = (\mathbf{X}_{array} \mathbf{X}_{array}^H)^{-1} \mathbf{X}_{array} \mathbf{r}(n) \quad (4.3.30)$$

where

$$\mathbf{X}_{array} = (\mathbf{I} - \mathbf{E}_s \mathbf{E}_s^H) \mathbf{S}_{array} \quad (4.3.31)$$

and the array weight vector obtained after n^{th} iteration now is,

$$\mathbf{w}_{final} = (\mathbf{I} - \mathbf{E}_s \mathbf{E}_s^H) \mathbf{w}(n+1) \quad (4.3.32)$$

By this means, the eigen-constraint is taken into account and array weight vector can be computed. Moreover, computation load of the above-mentioned approach can be reduced if the following noise sub-space approach is used.

4.3.2 A Noise Subspace Matrix Approach

In this part, an alternative noise sub-space approach is presented. Noise subspace is the complementary subspace of signal subspace and is always orthogonal to signal subspace, i.e.,

$$\mathbf{E}_s^H \mathbf{E}_n = \mathbf{0} \quad (4.3.33)$$

Based on (4.3.33), we deliberately compose an array weight vector, $\hat{\mathbf{w}}$, which is expressed in the following form instead of (4.3.27),

$$\hat{\mathbf{w}} = \mathbf{E}_n \mathbf{v} \quad (4.3.34)$$

where $\mathbf{v}$ is a non-zero vector of proper dimension such that $\hat{\mathbf{w}}$ is guaranteed to be completely in noise subspace. Thus, array response can be rewritten as

$$H(\theta_i) = \hat{\mathbf{w}}^H \mathbf{s}(\theta_i) = \mathbf{v}^H \mathbf{E}_n^H \mathbf{s}(\theta_i) = \mathbf{v}^H \mathbf{f}(\theta_i) \quad (4.3.35)$$

where

$$\mathbf{f}(\theta_i) = \mathbf{E}_n^H \mathbf{s}(\theta_i) \quad (4.3.36)$$

As in Section 4.3.1, the derivation in (4.2.12)-(4.2.24) is again repeated and results obtained are shown below,

$$\mathbf{v}(n+1) = (\mathbf{F}_{array} \mathbf{F}_{array}^H)^{-1} \mathbf{F}_{array} \mathbf{r}(n) \quad (4.3.37)$$

where

$$\mathbf{F}_{array} = \mathbf{E}_n^H \mathbf{S}_{array} \quad (4.3.38)$$

And the final result of array weight vector in this case is,

$$\mathbf{w}_{final} = \mathbf{E}_n \mathbf{v}(n+1) \quad (4.3.39)$$

In both approaches, the convergence of (4.3.30) or (4.3.37) is achieved by selecting proper initial values to start iterations. The convergence behavior using Constant Modulus algorithm is studied in [46]. In this case, the difference using either signal sub-space approach or noise sub-space approach is not very critical. They essentially converge to the same solution. Moreover, the speeds of convergence are approximately the same and relevant to the initial value chosen before iteration gets started, as can be seen in the next section.

4.4 Numerical Results

In the computer studies, a basic structure of uniform linear array is simulated. It consists of 9 sensor elements with half wavelength spacing. And for the signal scenario considered, it is assumed that there are 2 uncorrelated signal sources of unity-power present in the field. One is impinging from 40° and the other from 120° with respect to the array axis. The additive noise power is assumed to be 10dB below the zero reference and the number of snapshots taken to form a data covariance matrix was 100. Simulation results before and after optimization of $\tau(\theta_i)$ were generated and compared in Figure 4.2. The solid line is generated when $\tau(\theta_i)$ is given the form of $5 \cos(\theta_i)$ according to what is suggested in [25]. And the dashed line is plotted after 8 iterations.

Figure 4.2: A comparison between before optimization and after

It can be seen that in both cases, the two global minima indicating impinging directions are about the same value, as in Figure 4.3, because this is mainly determined by the eigen-constraint and SNR value. However, using the iteration procedure, the dashed line can basically maintain a unity-response with less ripples all over the directions where no impinging signals are present. It is robust and less sensitive to signal scenarios and less likely to produce spurious spikes.

The convergence of the iterative procedure for the same signal scenario is shown in Figure 4.4. The circle-solid and circle-dashed lines are by noise subspace and signal subspace approaches, respectively. As can be observed, the square-root value of the cost function is reduced and gradually converges to a constant. Simulation results match theoretical analysis. Moreover, it can be seen that the overall performances via using different sub-space approaches are almost the same. In fact, the difference between sub-space approaches are very trivial and is relevant to the initial value chosen for $\tau(\theta_i)$ to start the iteration. In Figure 4.5, a comparison is made between

Figure 4.3: A magnified image of one of the peaks

Figure 4.4: Illustration of iteration convergence

Figure 4.5: A comparison between signal and noise sub-space approach

Figure 4.6: A comparison between using different initialization values

these two approaches in the same scenario, but with randomly selected initial value. It is observed that the two plots differ more than in the previous figure using a different set of initial values. However, as far as overall performance is concerned, the difference is still very slight. Because they basically converge to the same value after the same number of iteration and the convergence speed is approximately the same, although it is true that within the iteration process, it sometimes converges faster if using signal sub-space and vice versa.

It shall be noted that the adaptive CM algorithm applied in this case is only locally optimum. There may be different performance using different initialization values to start an iteration. In general, a simple phase function has already been able to give us a good starting point. The local minimum problem is unlikely to cause serious problems when SNR is adequate. To guarantee the convergence more effectively, it is suggested one first obtain an array weight vector, $\mathbf{w}_u$, with an arbitrary phase value in UR template, for instance, $\tau(\theta_i) = 0$. Subsequently, one can calculate the array response over FOV and select the computed phase values to start the iteration, i.e. $\mathbf{r}_o(0) = e^{j\angle \mathbf{S}_{array}^H \mathbf{w}_u}$. A comparison between random initialization and suggested method has been made in Figure 4.6. It is evident that circle-dashed line converges faster than the circle-solid line. Eventually, both of them approximately reach the same convergence. However, it does not necessarily mean it is more computationally efficient to start with $\mathbf{r}_o(0)$, since it is already the 1st iteration to obtain $\mathbf{r}_o(0)$.

In addition, the value of $\tau(\theta_i)$ obtained at n^{th} iteration can alternatively be substituted into (4.1.1), but the same results will be obtained, following the closed-form solution for (4.1.1) and (4.1.2) derived in [25]. This is because the solution of every iteration in this paper is essentially the same as the closed-form solution. This can be justified from the fact that in (4.2.22), on the right side of the equation is exactly the quiescent solution of (4.1.1), thereafter, imposing identical eigen-constraint on this quiescent solution will result in the same answer.

Figure 4.7: Plot of the average absolute value of phase difference per iteration

Figure 4.7 is obtained via direct substitution of $\tau(\theta_i)$ into (4.1.1) and (4.1.2) for the same signal scenario and with the same number of iterations. It illustrates $\tau(\theta_i)$ is gradually phase-synchronized with array response as iteration number increases. The absolute phase difference is made between $\tau(\theta_i)$ and the phase angle of array response, $\angle H(\theta_i)$. It is then summed up over FOV and averaged afterwards.

Chapter 5

DOA Estimation of Wideband Signal Sources

5.1 Unity-Response Approaches to DOA Estimation of Wideband Signals

Most of DOA algorithms invented were initially invented for plane wave signals emitted by narrowband sources. To the best of our knowledge, there are so far very few algorithms that have been extended to wideband applications. Classic as MUSIC or Minimum-norm algorithms, seldom were articles published on them being extended to wideband scenario. Motivated by the increasing need of solutions for DOA estimation in wideband scenario, it is proposed in this chapter two approaches that successfully extend Unity-response (UR) algorithm to wideband applications. The reason why Unity-Response algorithm is chosen to be studied is because of its flexibility. It can be seen later that wideband signal parameters can be easily incorporated into template of UR estimator and it is applicable to arbitrary array

geometry. In the case of wideband signals, additional modification or reformulation is usually needed to guarantee the effectiveness of UR estimator.

Two methods to apply UR estimator to wideband applications are proposed and illustrated with numerical results in this chapter. In the first method proposed, UR estimator is modified and constructed to perform a 2-dimensional search over spatial domain as well as frequency domain in the presence of wideband signals, provided that the overall frequency range is known. Its basic idea is to search among every possible combination of frequency band and bearing angle to find null responses produced at signal direction and frequency band. This method is discussed and commented in Section 5.2. In Section 5.3, an alternative method is discussed. In the second method presented, UR estimator is reformulated by essentially manipulating signal subspace with Taylor-series expansion and as a result, only 1-dimensional search over spatial domain is needed. It utilizes the property of Taylor-series such that it is able to bring the null responses of a wideband signal at different frequency bands back to reference frequency. Consequently, only a 1-dimensional search over bearing angles is necessary. Numerical results and comments are attached in Section 5.3.

5.2 A 2-D Search Approach

5.2.1 Formulation

Attempts on extending eigen-structure based high-resolution algorithms to wideband applications have been made in the literature [47]- [56]. For narrowband signals, the time delay between array sensor elements appear as a pure phase delay of the reference signal, moreover, this phase delay depends only on the spacing between the sensors in question and the arrival angle of the plane wave, and is independent

of the time variable, as derived in (2.3.11)-(2.3.15). However this is not the case in wideband situation. The difference is that the phase delay is frequency sensitive in the case of wideband signals.

Most of the narrowband eigen-structure DOA algorithms are essentially constructed from the eigen-values and eigen-vectors introduced in Chapter 2. Now consider in the wideband case, a wideband signal can be regarded as the sum of several narrowband signals. And the superposition theorem still holds for different frequency bands. Therefore, equation (2.3.23) is still valid except that $\mathbf{s}(\theta_k)$ is frequency dependent in this case. One way to estimate DOAs of wideband signals is to perform UR algorithm separately in each frequency band. Thus, there will be one $\mathbf{E}_s$ and one $\mathbf{w}$ for every estimation, in which sense, $\mathbf{E}_s$ and $\mathbf{w}$ are both functions of frequency. This is presumably a feasible approach for DOA estimation of wideband signals. However, the computation load of this approach could be N times that of a narrowband UR estimator, where N here is the number of frequency bands. In practice, eigen-decomposition can be very computationally expensive. Moreover, we would like to preserve the array synthesis nature of UR algorithm where only one constant array weight vector is used throughout one entire estimation. According to this, a 2-D search method of modified UR algorithm is proposed with the following formulations

$$\min_{\mathbf{w}} \quad \int_f \int_{\theta} |H(f, \theta) - e^{j\tau(\theta)}|^2 d\theta df \quad (5.2.1)$$

$$\text{subject to} \quad \mathbf{E}_s^H \mathbf{w} = \mathbf{0} \quad (5.2.2)$$

The solution for the above quadratic problem (cf. Appendix B) would be, assuming in this chapter $\tau(\theta) = 0$,

$$\mathbf{w} = \mathbf{Q}^{-1} \mathbf{E}_s (\mathbf{E}_s^H \mathbf{Q}^{-1} \mathbf{E}_s)^{-1} \mathbf{E}_s^H \mathbf{Q}^{-1} \mathbf{P} - \mathbf{Q}^{-1} \mathbf{P} \quad (5.2.3)$$

where

$$\mathbf{Q} = \int_f \int_\theta \mathbf{s}(f, \theta) \mathbf{s}^H(f, \theta) d\theta df \quad (5.2.4)$$

$$\mathbf{P} = \int_f \int_\theta \mathbf{s}(f, \theta) d\theta df \quad (5.2.5)$$

It is noted in the above equations, array response, $H(f, \theta)$, and steer vector, $\mathbf{s}(f, \theta)$, are functions of both frequency and bearing angles, as contrasted to the narrowband scenario where only bearing angles are the variables.

Subsequently, the direction finding function is

$$D_{2-D}(f, \theta) = \frac{1}{|\mathbf{w}^H \mathbf{s}(f, \theta)|^2} \quad (5.2.6)$$

It shall be noted that $\mathbf{E}_s$ in (5.2.2) is obtained by eigen-decomposition of the received wideband signals. It is not a function of frequency, and hence, nor is $\mathbf{w}$. One may question that in a wideband scenario, $\mathbf{E}_s$ obtained this way can no longer fully characterize the model of the signals received. This is correct, but the extent to which $\mathbf{E}_s$ is able to reflect the signal model can be improved under certain assumptions.

In general, Steering Vector for a impinging signal at f_i in the direction θ_i can be represented by $e^{j\mathbf{a}(\theta_i)b(f_i)}$, where $\mathbf{a}(\theta_i)$ is a vector determined by array geometry and $b(f_i)$ is usually a 1st order function of frequency. However in a wideband scenario, for the same signal received, it may be generated by one or more different combinations of θ and f , other than θ_i and f_i . We would name these kinds of combinations spurious sets. Thus, frequency and angle directions become indistinguishable or ambiguous. Fortunately, this kind of ambiguity can be minimized in certain array geometry, for instance, ring geometry. This is why the first assumption made for this approach is that the proposed 2-D search method is applicable to certain array

geometries only, e.g. ring/double-ring geometry, with sufficient number of array sensors. Wideband structure is necessary if one applies this method to linear array. Secondly, the proposed method is effective with a limited bandwidth (not larger than 40% of the center frequency), since the wider the bandwidth, the more likely spurious sets would occur. Thirdly, for a given bandwidth and array geometry, the proposed method is more advantageous to signals with wider bandwidth than those with narrower bandwidth or narrowband signals. As one can see in Section 5.2.3, due to noise or disturbances, there are spurious peaks/nulls scattered in the spectra plots using the 2-D search method, one ought to look for the consecutive peaks/nulls in order to find the right combinations of frequencies and DOAs. Presumably, a signal with wider bandwidth can result in a longer continuity of peaks/nulls and thus, it is more likely to be identified.

For the proposed method, the idea behind is indeed an extension of UR algorithm. UR can be interpreted as an algorithm that tries to maintain the array response to be unity (or flat) over the field of view, and in the mean time zero-value directional eigenvector constraints are imposed. Extending the idea to wideband scenario, a 2-D scan approach is proposed to maintain the array response to be unity not only over field of view but also over frequency spectrum with the eigenvector constraints still imposed.

5.2.2 Ways to Determine Number of Signal Sources

Theoretically, the above-mentioned algorithm can be used to estimate DOA as well as signal frequency, but problems arise when it is implemented. The most controversial one would probably be how to determine the number of signals in a wideband scenario, which is critical to the determination of signal subspace $\mathbf{E}_s$.

Many eigen-structure-based algorithms require the exact determination of the effective rank of the covariance matrix. However, in a signal environment containing both narrowband and wideband sources, the effective rank cannot be determined accurately even if the signals are not correlated [49]. Also, it is quoted from [53], one of the difficulties in wideband study using eigenstructure-based approach is that one is unable to deduce the number of signal sources existing in the signal field based on the multiplicity of the smallest eigenvalues. In this chapter, three ways are suggested and simulation results obtained are compared.

Method A.

It is suggested that instead of finding signal subspace, one may choose to utilize the whole set of eigenvectors, however, weighted by its corresponding eigenvalues, which is equivalent to the following

$$\mathbf{E}_s = \mathbf{B} \left(\frac{\mathbf{\Lambda}}{\lambda_M} \right)^C \quad (5.2.7)$$

where $\mathbf{B}$ and $\mathbf{\Lambda}$ are as introduced in (2.3.33) and (2.3.34). λ_M represents the smallest eigenvalue obtained. Here C is an adjustable positive number. The idea behind this method is to weight the eigenvectors in $\mathbf{B}$ by normalized eigenvalues in $\mathbf{\Lambda}$ in such a way that the effect of those eigenvectors corresponding to signals is amplified, whereas the effect of eigenvectors corresponding to noise subspace is overwhelmed. The normalization of eigenvalues is achieved with respect to the smallest eigenvalue.

Method B.

The number of signals present in the field is determined by the number of eigenvalues, the ratio of whose sum over trace of covariance matrix exceeds specified threshold. The spatial covariance matrix, $\mathbf{R}$, can be computed. Thus, its trace, $tr(\mathbf{R})$, can also be calculated. If the specified threshold is denoted by ρ , the suggested method

is equivalent to the following

$$\frac{\sum_{i=1}^x \lambda_i}{tr(\mathbf{R})} \geq \rho \quad (5.2.8)$$

where $tr(.)$ denotes the trace of a matrix and λ_i 's are eigenvalues arranged in an descending order. x is the number of signal sources to be computed to satisfy (5.2.8). It is commented that the trace of a matrix is equal to the sum of all its eigenvalues. Unlike the narrowband scenario, it is difficult to determine the dimension of signal subspace by comparing the absolute values of eigenvalues. Thus, we may define the percentage ρ of signal eigenvalues among all of the eigenvalues. Once this percentage is exceeded, we may assume the accumulated eigenvalues correspond to signals. However, the percentage threshold, ρ , is usually an empirical value.

Method C.

The last method introduced is probably the simplest. It is suggested that the eigenvector corresponding to the minimum eigenvalue is taken out to compose noise subspace and the remaining set of eigenvectors make up of signal subspace. This last method introduced is a rather pragmatic approach to compose signal subspace in practice. Due to the complicated signal models in wideband scenario, the eigenvalues obtained are often close to each other. Therefore, in method C, the smallest eigenvalue is discarded and the rest of the eigenvalues are considered to correspond to signals received.

Figure 5.1: A double-ring array geometry

5.2.3 Simulation Results

In the simulation conducted, a double-ring array pattern of 10 elements is adopted (as in Figure 5.1). The signal scenario in which simulations were conducted is described. There are two wideband signals. One is from 40 degrees and the other 120 degrees. The signal from 40 degrees is assumed to have a flat spectrum covering a frequency range from 0.8 to 1.2 with respect to reference frequency (a percentage bandwidth of 40%) and the signal from 120 degrees possesses frequency coverage from 1.0 to 1.1 (a percentage bandwidth of 10%). SNR is 10dB and number of snapshot of 100 is taken to form the covariance matrix. Using the three methods described in Section 5.2.2, the simulation results are attached in order on the next

page. It is suggested in method A, power factor C can empirically be assigned to be 7. And in method B, ρ takes 99%.

In Figures 5.2, 5.3 and 5.4, on the upper side are those plotted with mesh function in Matlab and on the bottom are modified versions of the left mesh plot. They are plotted by averaging over all the frequency responses at every angle being scanned. Observing from the simulation results, it can be seen that methods B and C have very similar performance. As a matter of fact, method B can eventually turn to be method C by lifting up the value of the threshold ρ .

5.2.1:

5.2.2:

Figure 5.2: UR Estimator extended to wideband scenario using method A

5.3.1:

5.3.2:

Figure 5.3: UR Estimator extended to wideband scenario using method B

5.4.1:

5.4.2:

Figure 5.4: UR Estimator extended to wideband scenario using method C

5.2.4 Comments on the Proposed Method

As it can be seen, the proposed 2-D scan approach is able to not only estimate DOA of wideband signals but also give estimation of the frequency band in which signals are present. Moreover, it utilizes only one weight vector applicable to any combination of frequency and direction, which is superior over many other algorithms, e.g. Capon or MUSIC in which weight vectors change with different sets of frequency or directions. However, there are two main drawbacks for the proposed approach.

First of all, as the name implies, it has to adopt a 2-D scan procedure, i.e. either scan the whole frequency spectrum for every angle or scan over the whole FOV at every frequency band. It is obvious that the computation is complicated and the load is tremendous.

Secondly, it can be observed, from the 3-D plots in Figures 5.2.1, 5.3.1 and 5.4.1, that there are spurious peaks scattered at some frequencies and directions that are very misleading. Fortunately, they can be eliminated by adding up the responses of the whole frequency spectrum at every angle, as in the bottom plots in Figure 5.2.2, 5.3.2 and 5.4.2. However, doing so is essentially in favor of the signals with broader frequency coverage. It is in less favor of signals with narrower band, especially in the scenario both narrowband and wideband signals are present, it may result in the total loss of narrowband signal. And this also explains why in the figures above the peak at 120 degrees is not as high as that in 40 degrees.

5.3 A Taylor-Series Approach

5.3.1 Formulation

The biggest disadvantage of the 2-D search algorithm is its tremendous computation load. In applications where only signal DOA is of interest, the 2-D search approach appears to be inefficient. With the wish to reduce computation load and yet not to degrade system performance, an alternative algorithm is proposed in this section, namely, a Taylor-series approach.

Assuming there is a signal with operating frequency f_i present in the field impinging from direction θ , then its steering vector $\mathbf{s}(f_i, \theta)$ can be expanded using Taylor series as

$$\mathbf{s}(f_i, \theta) = \mathbf{s}(f_0, \theta) + \Delta f \frac{\partial \mathbf{s}(f, \theta)}{\partial f} \Big|_{f=f_0} + \frac{\Delta f^2}{2!} \frac{\partial^2 \mathbf{s}(f, \theta)}{\partial f^2} \Big|_{f=f_0} + \cdots \quad (5.3.9)$$

where f_0 is the reference (nominal) frequency and $\Delta f = f_i - f_0$.

If a null response is expected at frequency f_i and direction θ , i.e.

$$\mathbf{w}^H \mathbf{s}(f_i, \theta) = 0 \quad (5.3.10)$$

Substitute (5.3.9) into (5.3.10), it can be derived that sufficient conditions for (5.3.10) to be valid are

$$\mathbf{w}^H \mathbf{s}(f_0, \theta) = 0 \quad (5.3.11)$$

$$\mathbf{w}^H \frac{\partial \mathbf{s}(f, \theta)}{\partial f} \Big|_{f=f_0} = 0 \quad (5.3.12)$$

$$\mathbf{w}^H \frac{\partial^2 \mathbf{s}(f, \theta)}{\partial f^2} \Big|_{f=f_0} = 0 \quad (5.3.13)$$

$$\vdots$$

$$\mathbf{w}^H \frac{\partial^n \mathbf{s}(f, \theta)}{\partial f^n} \Big|_{f=f_0} = 0 \quad (5.3.14)$$

From the observation above, it is proposed to reconstruct the signal subspace matrix, $\mathbf{E}_s$, as $\mathbf{E}_d$, where

$$\mathbf{E}_d = [\mathbf{E}_s, \frac{\partial \mathbf{s}(f, \theta)}{\partial f}|_{f=f_0}, \frac{\partial^2 \mathbf{s}(f, \theta)}{\partial f^2}|_{f=f_0}, \dots, \frac{\partial^n \mathbf{s}(f, \theta)}{\partial f^n}|_{f=f_0}] \quad (5.3.15)$$

By replacing $\mathbf{E}_s$ with $\mathbf{E}_d$ in UR template, the sufficient conditions in (5.3.11)-(5.3.14) are all met. Subsequently, a null response, expected in (5.3.10), can be produced at proper frequency band and angle direction.

It is noted that (5.3.11) is not necessary because of the constraint included in UR template that weight vector, $\mathbf{w}$, must be orthogonal to signal subspace, $\mathbf{E}_s$. It is also noted that n in (15) shall be an integer selected according to the bandwidth required. The larger the bandwidth, the larger value n shall take. Moreover, n shall be selected in such a way that the dimension of $\mathbf{E}_d$ cannot exceed the number of array elements.

As mentioned, $\mathbf{E}_s$ is difficult to find due to the fact that in case both narrowband and wideband signals are present, it is unable to deduce the number of signal sources existing in the field. In [51], a simple eigen-weighting method is adopted to eliminate the need of estimating the number of signal sources. In actual implementation, $\mathbf{E}_d$ can be resembled by $\mathbf{E}_b$, where

$$\mathbf{E}_b = [\mathbf{B}(\frac{\lambda_1}{\lambda_M})^p, (\frac{\lambda_1}{\lambda_M})^{p-q} \frac{\partial \mathbf{s}(f, \theta)}{\partial f}|_{f=f_0}, (\frac{\lambda_1}{\lambda_M})^{p-q} \frac{\partial^2 \mathbf{s}(f, \theta)}{\partial f^2}|_{f=f_0}, \dots, (\frac{\lambda_1}{\lambda_M})^{p-q} \frac{\partial^n \mathbf{s}(f, \theta)}{\partial f^n}|_{f=f_0}] \quad (5.3.16)$$

where $\mathbf{B}$, as usual, denotes the whole set of eigenvectors. p and q are non-negative real numbers selected accordingly.

Thus, UR algorithm can be applied to wideband signal scenario by replacing signal subspace matrix $\mathbf{E}_s$ with $\mathbf{E}_b$ in UR template. However, the computation load still

would be heavy due to the fact that close-form solution of UR template is involved and $\mathbf{E}_b$ is a θ -dependent vector. To save computation load, some off-line preprocessing technique is necessary. It is proposed to construct a new complementary subspace matrix, $\mathbf{E}_c$.

$$\mathbf{E}_c = \left[\frac{\partial \mathbf{s}(f, \theta)}{\partial f} \Big|_{f=f_0}, \frac{\partial^2 \mathbf{s}(f, \theta)}{\partial f^2} \Big|_{f=f_0}, \dots, \frac{\partial^n \mathbf{s}(f, \theta)}{\partial f^n} \Big|_{f=f_0} \right] \quad (5.3.17)$$

Thus, UR estimator can be reformulated into two parts to function in the presence of wideband signals. Firstly,

$$\min_{\mathbf{y}} \int_{\theta} |G(f_0, \theta) - e^{j\tau(\theta)}|^2 d\theta \quad (5.3.18)$$

$$\text{subject to} \quad \mathbf{E}_c^H \mathbf{y} = \mathbf{0} \quad (5.3.19)$$

where $G(f_0, \theta)$ and $\mathbf{y}$ are hypothesized array response and hypothesized array weight vector, respectively. And

$$G(f_0, \theta) = \mathbf{y}^H \mathbf{s}(f_0, \theta) \quad (5.3.20)$$

The solution is given as follows, assuming $\tau(\theta) = 0$.

$$\mathbf{y} = \mathbf{Q}^{-1} \mathbf{E}_c (\mathbf{E}_c^H \mathbf{Q}^{-1} \mathbf{E}_c)^{-1} \mathbf{E}_c^H \mathbf{Q}^{-1} \mathbf{P} - \mathbf{Q}^{-1} \mathbf{P} \quad (5.3.21)$$

where

$$\mathbf{Q} = \int_{\theta} \mathbf{s}(f_0, \theta) \mathbf{s}^H(f_0, \theta) d\theta \quad (5.3.22)$$

$$\mathbf{P} = \int_{\theta} \mathbf{s}(f_0, \theta) d\theta \quad (5.3.23)$$

Note that $\mathbf{E}_c$ is a matrix dependant on θ . Hence, for every particular angle θ_i concerned, there is a corresponding weight vector $\mathbf{y}_i$ computed. Note again that all the calculation carried above is in the absence of signals in the real environment.

That is why $G(f_0, \theta)$ and $\mathbf{y}$ are “hypothesized”. And because of that, the value of $\mathbf{y}_i$ corresponding to different bearing angle can be pre-computed before real-time implementation and stored. Secondly,

$$\min_{\mathbf{w}} \quad (\mathbf{w} - \mathbf{y})^H (\mathbf{w} - \mathbf{y}) \quad (5.3.24)$$

$$\text{subject to} \quad \mathbf{E}_s^H \mathbf{w} = \mathbf{0} \quad (5.3.25)$$

As a result,

$$\mathbf{w} = \mathbf{y} - \mathbf{E}_s \mathbf{E}_s^H \mathbf{y} \quad (5.3.26)$$

In this 2nd part of formulation, $\mathbf{E}_s$ is unique and obtained from the spatial covariance matrix measured, but due to the fact that $\mathbf{y}$ shall take different values of $\mathbf{y}_i$ ’s for different directions when search over bearing angles, consequently, $\mathbf{w}$ is not a constant vector but varies over spatial domain. As a result, for every angle θ_i concerned, a weight vector $\mathbf{w}_i$ shall be computed. The actual array response is

$$H(f_0, \theta_i) = \mathbf{w}_i^H \mathbf{s}(f_0, \theta_i) \quad (5.3.27)$$

5.3.2 Simulation and Discussion

Again, the array geometry in Figure 5.1 is adopted for comparison purpose. The signal scenario existing in the field is there are 3 signal sources. One signal from 40° is assumed to have a flat spectrum covering from a frequency range from 0.8-1.2 with respect to reference (nominal) frequency, i.e. a 40% bandwidth. The signal from 60° has a flat frequency spectrum from 1.0-1.1, i.e. a 10% bandwidth. the third signal from 120° is a narrowband signal at nominal frequency. SNR is set to be 10dB and the number of snapshots is 100. $\mathbf{E}_s$ in (5.3.25) is represented by $\mathbf{B}(\frac{\Lambda}{\lambda_M})^p$ where p takes the value of 5. And n in (5.3.17) is assigned 3. The simulation result for

Figure 5.5: UR extended for wideband application via a Taylor-series approach

the proposed method in the specified scenario is attached in Figure 5.5. Compared with the 2-D search approach in Section 5.2, the Taylor-series approach utilizes 1-D search only. With the reconfigured 2-step method, the estimator is constrained to have null response at all signal directions and signal frequencies f_i . Therefore, if the flat response at f_0 is extended to the frequencies f_i at the signal directions, the effect of null response at the signal directions will in turn be brought back and reflected on the spatial spectrum at frequency f_0 . Thus, only a 1-D search over FOV using nominal frequency is necessary. Moreover, since all the “hypothesized” terms can be pre-computed, the computation load is largely reduced and much less involved. However, its weight vector is no longer a fixed vector applicable to all the directions. It indeed changes as the direction being searched. This is due to the fact that for each direction, there is a different $\mathbf{E}_c$.

It shall be noted the phase term τ in the UR estimator is set to 0 throughout this chapter. In Figure 5.6, the plot is generated for the exactly same signal scenario as

Figure 5.6: Phase optimized UR in broadband scenario using CM approach

above, however, with τ optimized using the Constant Modulus approach proposed in Chapter 4. After comparison with Figure 5.5, it is observed that it results in a noise floor with less number of ripples or spurious spikes using Constant Modulus approach and hence, system is more noise-tolerant.

Lastly, before we can conclude the Taylor-Series approach in this section, there is one more comment to make. It shall be noted that the sufficient condition in (5.3.14) may result in a situation that certain impinging signals escape detection. This likely happens when, for instance, signal frequency f_i is much larger than f_0 and the value of n is too small. As a matter of fact, n in (5.3.14) is a positive integer number and the value assigned to it shall be determined by the frequency bandwidth of interest. A general guideline is the larger is the frequency bandwidth considered, the larger is the value assigned to n . Therefore, when signal frequency f_i falls in the vicinity of f_0 , a small integer number, e.g. 2 or 3, would be sufficient, however, when f_i falls in a range far away from f_0 , a larger value of n shall be considered. In practical cases

where the frequency bandwidth of interest is not very large with respect to f_0 , yet the value of n is sufficiently large, it is unlikely to happen that certain signals may escape detection.

Chapter 6

Implementation of Capon and UR Algorithms on DaqPC

6.1 Hardware System Description

6.1.1 Introduction of DaqPC

The DaqPC is a high-end data acquisition and processing system that combines the low cost and familiarity of a standard PC environment with high performance PCI products from ICS (Interactive Circuits and Systems Ltd) and third party DSP and data storage hardware. The DaqPC uses WinNT, Windows 2000 or Windows 2000 server operating systems and adopts the popular LabView software for GUI based control and display. With the recent introduction of a family of highly advanced PCI and PMC bus boards and associated software, a wide range of high speed and high accuracy data acquisition, storage and demodulation applications can now be easily configured. Relevant abbreviations can be referred to in Table 6.1.

The PC hardware used by the DaqPC comes in a standard INTEL industrial rack

mount unit. The rack-mount version of the DaqPC can be delivered in a wide variety of configurations, with high-end processor, memory, monitor, peripherals, etc. An SCSI based RAID is included for high-speed real-time storage.

For acquisition of analog signals, there are a number of different choices in terms of off-the-shelf PCI and PMC boards. These boards are very high performance products and support simultaneous sampling irrespective of the number of boards used. The acquired data is streamed at high speed to a NTFS RAID. The RAID based configuration allows for data acquisition and storage to occur at over 200MB/s, with storage capacity up to 1.4TB. All DaqPC's use a Dual Xeon motherboard, and have all of the typical PC options such as dual Gigabit Ethernet, COM and parallel ports, a LS-120 floppy and DVD R/W drive.

Acronym	Complete Name
A/D	Analog to Digital (converter)
ADC	Analog to Digital Converter
CD	Compact Disk
D/A	Digital to Analog (converter)
DAC	Digital to Analog Converter
DDC	Digital Down Converter
DMA	Direct Memory Access
EDMA	Enhanced Direct Memory Access
FFT	Fast Fourier Transform
FIFO	First In First Out (memory)
FPDP	Front Panel Data Port (a high speed bus)
FPGA	Field-programmable Gate Array
GUI	Graphical User Interface
HDK	Hardware Development Kit
P14	A user-definable connector on a PMC card
PCI	Peripheral Component Interconnect
PMC	PCI Mezzanine Card
RAID	Redundant Array of Independent (or Inexpensive) Disks
Rx	Receive
SBC	Single Board Computer
SDK	Software Development Kit
Tx	Transmit

Table 6.1: Acronyms of system parts

6.1.2 Configurations of DaqPC 554/111

The hardware platform used in this project is a customized DaqPC 554/111. A snapshot of the system is taken in Figure 6.1 and Figure 6.2. In this section, system configurations and internal structures are introduced and functions are illustrated with block diagrams. Acronyms that are helpful to the illustration have been summarized in Table 6.1. In Figure 6.3, it contains a block diagram inside the DaqPC 554/111.

Figure 6.1: Picture I of the DaqPC system

Figure 6.2: Picture II of the DaqPC system

Figure 6.3: Block diagram of DaqPC 554/111

The Rx subsystem is contained in the upper Single Board Computer (SBC) and the Tx subsystem is contained in the lower SBC. In Figure 6.3, the different boards perform the different functions described in Table 6.2.

Board	Description
ICS-554C-4-MN	The ICS-554 is a 4-channel 14-bit A/D module in PCI MezzanineCard (PMC) format with sample rates of up to 105 MHz/channel(100 MHz if using the on-board DDC's). It also includes a 16-channel multi-narrow bandwidth DDC.
ICS-564A	The ICS-564A is a 4-channel, 200MHz, 14-bit PMC Module DAC card that offers a wide range of operating modes and features.
Transtech FPGA	PMC module with a Xilinx 3 or 6 Million Gate Virtex II FPGA. These boards are used to implement an FPDP interface, a PCI interface and an interface via the P14 connector to the D/A, A/D DSP boards. Much of the processing capability of the Xilinx FPGAs is currently unused. The user may implement additional data processing on these boards by modifying the ICS-supplied firmware.
Data ADC-PMC2	A dual 64-bit PCI carrier card.
Blade-PCI-554-F06M/007	The ICS-554 and 6-Million-gate FPGA cards, mounted on a carrier are referred to as the Blade-PCI-554-F06M/007 (henceforth referred to as the Rx Blade).
Blade-PCI-564-F06M/002	The ICS-564 and 6-Million-gate FPGA cards, mounted on a carrier are referred to as the Blade-PCI-564-F06M/002 (henceforth referred to as the Tx Blade).
Blade-PCI-DP01-F03M/001	The DSP and 3-Million-gate FPGA cards, mounted on a carrier are referred to as the Blade-PCI-564-F06M/002 (henceforth referred to as the DSP Blade).

Table 6.2: Description of functional blocks

6.1.3 System Operation

In this project, the DaqPC 554/111 system is configured in the following manner for the task of DOA estimation:

- Both Blade-PCI-564 (Tx Blade) and Blade-PCI-554 (Rx Blade) are connected to the dBm frequency converter, as shown in Figure 6.5.
- A Transmit Antenna is connected to Tx Blade through the dBm frequency up converter and a Receive Antenna array of four elements is connected to Rx Blade through the dBm frequency down converter as well.
- A digital baseband signal is generated by the system and up-converted to IF analog signal at the output of Tx Blade.
- The IF analog signal goes through dBm frequency up converter and is further up-converted to radio frequency and transmitted via the transmit antenna.
- Radio frequency signal picked up by the receive antenna array is down-converted to IF analog signal by dBm frequency down converter.
- IF signal is further down-converted to baseband digital signal through Rx Blade and passed on the DSP Blade.
- DOA algorithms are loaded on the DSP Blade and signals are processed to obtain DOA estimation.

The above-stated system configuration is depicted in Figure 6.4.

Figure 6.4: Schematic diagram of system configuration for DOA estimation

In Figure 6.4, the DaqPC system has been shown in Figure 6.1 and Figure 6.2. The dBm frequency converter is a Up and Down Frequency converter system. The Up converter is housed in one chassis and the Down converter is housed in a sperated chassis. Each chassis contains its own frequency synthesizer which is programmed via the IEEE-488 bus and is used as a local oscillator to tune the converter. A snapshot of it is taken in Figure 6.5. The transmit antenna is fabricated by our own project team and is shown in Figure 6.6. It is noted that only one element in Figure 6.6 is connected to signal source, because we intend to simulate the scenario of a point signal source. The four-element receive antenna array is pictured in Figure 6.7.

Figure 6.5: An outlook of the dBm frequency converter

Figure 6.6: Transmit antenna (one antenna used to transmit signal)

Figure 6.7: Receive antenna array of 4 elements

6.2 Structure of the DSP Blade

Different algorithms to be implemented can be developed on the DSP blade. It is the core processing unit in this case. The DSP Blade (Blade-PCI-DP01-F03M/001) is intended for high-performance data processing of FPDP data. It may be configured to process received FPDP data, or generate FPDP transmit data. Data may be received or transmitted at up to 200MB/s. Figure 6.8 shows a physical layout diagram of the DSP blade. The DSP blade consists of a Mango Seagull DSP and a 3-Million-gate FPGA PMC board, mounted on a carrier. The details of the constituent boards are described in Table 6.3.

Figure 6.8: DSP blade physical layout diagram

Board	Description
Mango Seagull Quad C64 DSP	This is a Mango Seagull PMC module with 4 TI C6415 DSP processors, and an Altera FPGA. This board contains a custom interface done on an Altera chip to provide data I/O over the P14 connector. The PCI interface on this module is 32-bit and 33MHz.
Transtech FPGA	PMC module with a Xilinx 3 Million gate Virtex II FPGA. This board is used to implement an FPDP-to-P14 interface to the DSP board. Much of the processing capability of the Xilinx FPGA is currently unused. The user may implement additional data processing on this board by modifying the ICS-supplied firmware. This module includes a daughtercard to provide a standard FPDP interface, which can be configured to receive or transmit.
PMC to PCI carrier card	This is an Alpha Data ADC-PMC2A, dual 64-bit PMC to PCI carrier card.

Table 6.3: Constituent boards of DSP blade

The DSP blade can be software configurable to both receive and transmit modes. However, as far as DOA estimation is concerned, only the receive mode matters. The receive applications function in the following manner:

- The Transtech FPGA is configured as an FPDP receiver. The on-board FPGA is programmed to pass through the required signals from the FPDP module to its Pn4 connector (which is in turn connected to the DSP board). Once this module has been configured via the PCI bus, no further interaction with this module is required.
- The DSP module FPGA is programmed to receive and buffer (with FIFOs) the FPDP data stream. It is hard-coded to receive 4 FPDP channels (each 32-bit channel containing a 16-bit in-phase and 16-bit quadrature pair). The FPGA sends
 1. channel 1 data to DSP1;
 2. channel 2 data to DSP2;

3. channel 3 data to DSP3;
 4. channel 4 data to DSP4.
- The internal structure of DSP module is shown in Figure 6.9 , the core of which consists of 4 DSPs. The DSP module runs software that
 1. configure each DSP chip where end-users can develop custom algorithms;
 2. uses an Enhanced Direct Memory Access (EDMA) to read data from the FPGA into each DSP;
 3. has been configured to expect a particular FPDP byte pattern. Each DSP compares one channel of received FPDP data to the data expected and keeps a running sum of all errors detected.
 - A host PC is connected to the DSP board through P11 & P12 as in Figure 6.8. A program running on the host PC loads the FPGA code and software executable to the DSP module. It then controls the various stages of the DSP algorithms. It can optionally transfer data to and from the DSPs while algorithms are running on the DSPs.

Figure 6.9: Internal structure of Mango Seagull Quad C64 DSP board

6.3 Algorithm Coding and Experimental Results

The coding of Capon and Unity-Response algorithms is conducted using C programming language. The experiment is conducted with the mobile transmit antenna terminal connected to Blade-PCI-564 (Tx Blade) where a signal is generated, modulated and transmitted. The transmitted signal goes under the real environment and are subsequently picked up by a four-element receive antenna array with half wavelength spacing. The four antennas of the array are each connected to one channel of Blade-PCI-554 (Rx Blade) through which received signals are demodulated, down-converted and passed on to the Blade-PCI-DP01 (DSP Blade). We are then able to implement our algorithms to obtain DOA estimates.

Figure 6.10 and Figure 6.11 are displaying the flowcharts of coding Capon and UR algorithms, respectively. In Figure 6.11, **P**, **Q** and **Es** have all been introduced in Chapter 2 and Chapter 3. The detailed source codes are attached in Appendix C, meanwhile in Appendix D, source codes for pre-defined functions are included.

Figure 6.10: Flowchart of programming Capon algorithm

Figure 6.11: Flowchart of programming Unity-Response algorithm

Figure 6.12: Real-time data acquired and recorded by the Rx Blade

In Figure 6.12, the actual data is captured by Rx Blade and its information is displayed using Labview. The results of a typical experiment are attached in Figure 6.13 and Figure 6.14. In this case, the antenna terminal is located at 80 degrees with respect to the array axis. A 2.45GHz high frequency sinusoidal signal with -24dBm power level is transmitted in an indoor environment. The received signal is first down-converted to the intermediate frequency at 70MHz , and further down-converted to baseband signal. Capon and Unity-Response algorithms are implemented, by which signal DOA is correctly estimated. 200 snapshots are taken. The results are compared with the theoretical results by Matlab simulations in Figure 6.15 and Figure 6.16. The coarseness in the floor in Figure 6.13 and Figure 6.14 can be justified by the reflection, deflection and movements of the indoor environment. It is also observed that mainlobe width of UR estimator is normally narrower than that of Capon estimator, which implies the fact that UR algorithm has high resolution capacity over Capon algorithm.

Figure 6.13: DOA estimation in real environment using Capon algorithm

Figure 6.14: DOA estimation in real environment using UR algorithm

Figure 6.15: Capon estimator using Matlab generated data

Figure 6.16: UR estimator using Matlab generated data

Chapter 7

Conclusion and Recommendation

7.1 Conclusion

With the explosive growth of mobile communications and increasing need of ubiquitous coverage and connectivity, people are struggling for capacity enhancement on wireless communication systems. “Smart antennas” is one way to improve wireless system capacity in the spatial domain. Yet, it allows multiple users to transmit information over the same radio channel. The central technologies of smart antennas come from array signal processing.

In the subject of array signal processing, Directions-of-arrival (DOA) estimation has been a major topic that captivates researchers in this field. In the history of DOA estimation, various algorithms have been proposed. Nevertheless, there are still issues left unsolved. Optimization of phase values is one of the controversial issues that has existed in some algorithms for a long time. It can be found in many widely-used theorems or algorithms, such as Linear Predictive (LP) theorem, Minimum Norm (MN) algorithm and Unity-Response (UR) algorithm. In some cases, an improper decision made on the phase quantity can lead to a severe degradation

in performance. Therefore, it is important to find solutions to resolving phase optimization. Moreover, most of the DOA algorithms were initially designed for the narrowband signals only. With the rapid development of modern communications industry, it is critical to extend DOA algorithms to wideband and provide wideband solutions.

With the hope to obtain above-stated solutions, in this thesis, techniques of phase optimization have been proposed to different DOA algorithms. Furthermore, Unity-Response estimator has been reformulated for wideband applications. In Chapter 2, necessary background was provided in the field of array signal processing. It has introduced the array concept and explained the signal and noise model a sensor array was expected to receive in detail. The advantages in using an array were illustrated for a single source case. More importantly, the underlying property for eigenvector-based DOA algorithms has been derived thoroughly. Besides the above, Chapter 2 also has served to provide a review of some typical DOA estimation algorithms of different kinds. Those algorithms studied and modified in this project were elaborated in detail. In the mean time, terms and notations encountered often in DOA estimation were introduced or defined.

With all the knowledge readily given in the previous two chapters, we have proposed an averaged method in Chapter 3 for Minimum-norm algorithm to optimize its phase position. The method was proposed using the concept of polynomial rooting in which MN estimator was transformed into a polynomial and the mission to find DOAs was equivalent to solving for the roots of the polynomial. The proposed method was subsequently applied in Unity-Response algorithm. The relation between MN and UR estimators was discussed and we have concluded that, the issues of phase optimization in both estimators could be equally treated. Moreover, it turns out that computation load could be furthered reduced if applying the averaged method to UR estimator than MN estimator. Lastly, statistical performance

analysis of UR estimator has been given and compared with that of MN and MUSIC estimators. Computer simulations have been conducted to support the results.

In Chapter 4, the nature of phase selection in Unity-Response algorithm was exploited. Criteria and solution other than the averaged method have been proposed. By reformulating UR estimator template, the problem of phase selection was transformed into a Constant Modulus problem, however, with additionally imposed eigen-constraint. A least square approach was applied to solve the Constant Modulus problem and two different subspace approaches were subsequently presented to incorporate the eigen-constraint. Numerical results have also showed the proposed approach was correct and effective in achieving robust DOA estimation using optimized Unity-response algorithm.

In Chapter 5, we have proposed two methods to extend Unity-response algorithm to wideband scenarios, namely a 2-Dimensional search approach and a Taylor-series approach. For the first 2-D search approach, it could be interpreted as an extension of the idea of UR estimator in the frequency domain, which enabled UR algorithm to estimate not only DOA but also frequency band of the impinging signals. Tremendous computation load was its main disadvantage. For the second approach discussed, it provided an solution with much less computation load by using property of Taylor-series. And only 1-dimensional search was necessary. Issues regarding number of signal sources present in the field were discussed and ways to determine them were introduced. Additionally, the Constant Modulus approach proposed in Chapter 4 for phase optimization was also adopted for wideband signal scenario. Simulations results were provided to verify the analysis.

In Chapter 6, we have implemented Capon algorithm and Unity-Response algorithms on a DaqPC 554/111 platform. A DaqPC is a high-end data acquisition and processing system that can simulate the signal transmit and receive in a real environment. Algorithms of different categories can be programmed and loaded on

its DSP blade to perform real-time processing. In this project, we have familiarized with internal structures of DaqPC 554/111 and configured the system in a way to achieve the mission of DOA estimations. Subsequently, Capon and Unity-Response estimators were coded and loaded to perform DOA estimation. The experimental results were provided to prove the effectiveness of these two DOA algorithms.

7.2 Recommendations for Further Research

Besides all the contents included above, there are still remaining topics that can be possibly researched from this point on. However, due to limited time and energy, these topics are not completed yet and will contribute to some of my recommendations for future research.

It shall be noted that all the algorithms or optimization techniques we discussed thus far are only applicable to signal scenarios where there are no correlation between signal sources. When it comes to correlated or even coherent signal scenarios, spatial smoothing techniques shall be adopted. It is a question mark whether the proposed methods can be combined with spatial smoothing techniques.

Moreover, both of the phase optimization methods proposed previously are invented for narrowband signal sources, whereas in Chapter 5, Unity-Response algorithm has been extended for wideband applications. It is reasonable to think of a way to extend the phase optimization techniques to wideband as well. Some of the simulations have been conducted, e.g. Figure 5.6, to indicate the possibility of achieving so. However, rigorous mathematical derivation is yet to be done.

One more research topic to recommend is to extend the existing DOA algorithms to Ultra Wideband (UWB) applications. Ultra Wideband is a term for a classification of signals that occupy a substantial bandwidth relative to their centre frequencies.

UWB signals consist of very short pulses of energy separated in time but an amount much larger than the length of the pulse. Essentially, this means that the duty cycle is very low. Typical UWB signals are composed of pulses of only a few cycles in length and occupy a very large frequency bandwidth. Due to the low duty cycle of UWB, the average transmitted power is very low, resulting in low power consumption. UWB wireless system allow low probability of intercept and detection (LPI/D), multipath immunity, high data throughput, precision ranging and localization. Therefore, it is advantageous to have DOA solutions for UWB applications. Lastly, in terms of hardware implementation of theoretical algorithms, the DaqPC platform used in this project is a powerful and efficient tool for the verification of algorithms' effectiveness. It is recommended newly developed algorithms can be coded and loaded using the DaqPC system to perform proper functions.

Author's Publications

1. Haoji Bao, Boon Poh Ng, "A Constant Modulus Approach to Phase Optimization of Unity-Response Algorithm in Direction-of-Arrival Estimation", submitted to IEE Proc. Radar, Sonar & Navigation.
2. Haoji Bao, Boon Poh Ng, "An Averaged Method to Phase Optimization of Minimum-norm and Unity-response Algorithms in Direction-of-arrival Estimation", submitted to the Institute of Electronics, Information and Communication Engineers (IEICE).
3. Haoji Bao, Boon Poh Ng, "Estimation of Direction-of-arrival of Wideband Signal Sources using Unity-Response (UR) Approach", submitted to the Institute of Electronics, Information and Communication Engineers (IEICE).
4. Changzheng Ma, Boon Poh Ng and Haoji Bao, "Robust Adaptive Beamforming For Large Steering Angle Error", IEEE International Symposium on Circuits and Systems (ISCAS 2004), Vancouver, Canada, May 2004.

Bibliography

- [1] K.R.Srinivas and V.U.Reddy, "Finite Data Performance of MUSIC and Minimum Norm Methods," IEEE Trans., Vol. 30, January 1994.
- [2] R.A.Monzingo and T.W Miller, "Introduction to Adaptive Arrays," Wiley, 1980.
- [3] B.P.Ng, M.H.Er and Alex C.C.Kot, "A New Limit on Inter-element Spacing for a Linear Array," IEEE Antennas and Propagation Magazine, Vol. 40, No. 4, pp. 22 - 29, August 1998.
- [4] R.T.Compton, "Adaptive Antennas," Englewood Cliffs, NJ: Prentice Hall, 1988.
- [5] S.Unnikrishna Pillai, "Array Signal Processing," New York: Springer-Verlag, 1989.
- [6] C.N. Dorny, "A Vector Space Approach to Models and Optimization," New York: John Wiley and Sons, 1975.
- [7] J.P.Kennedy and S.W.Ellingson, "Smart Antenna Testbed for Mobile Wireless Systems," Proc. Sixth Va. Tech Symposium on Wireless Personal Comm., Blacksburg, VA, 1996.

- [8] J.H.Reed, K.J.Krizman, B.D.Woerner and T.S.Rappaport, "An Overview of the Challenges and Progress in Meeting the E-911 Requirement for Location Service," IEEE Personal Comm. Magazine, Vol. 5, No. 3, pp. 30-37, April 1998.
- [9] T.S.Rappaport, J.H.Reed, and B.D.Woerner, "Position Location Using Wireless Communications on Highways of the Future," Vol. 34, No. 10, pp. 31-41, October 1996.
- [10] Joseph C. Liberti, JR. and Theodore S. Rappaport, "Smart Antennas for Wireless Communications: IS-95 and third generation CDMA applications," Prentice Hall PTR, 1999.
- [11] S.Haykin, "Adaptive Filter Theory," Prentice Hall, NJ, 1991.
- [12] Yan Guozheng, Wang Hongjie, Ding Guoqing and Lin Liangming, "Augmented Lagrange Multiplier based Fuzzy Evolutionary Algorithm and Application for Constrained Optimization," Proceedings of the 4th World Congress on Intelligent Control and Automation, Vo. 3, pp. 1774-1778, June 2002.
- [13] R.O. Schmidt, "Multiple Emitter Location and Signal Parameter Estimation," Proc. of RADC Spectrum Estimation Workshop, Griffiss AFB, NY, pp. 243-258, 1979.
- [14] Bienvenu, W.J., Kopp, "Principle de la Goniometrie Passive Adaptive", Proc. 7^{eme} Colloque GRESIT. Nice, Vol. 106, pp. 1-10, 1979.
- [15] S.U. Pillai and B.H. Kwon, "Performance Analysis of MUSIC-Type High Resolution Estimator for Direction Finding in Correlated and Coherent Scenes," IEEE Trans. on Acoustics, Speech, and Signal Processing, Vol. 37, No. 8, pp. 1176-1789, August 1989.

- [16] Tsakalides, P. and Nikias, C.L, “The Robust Covariation-based MUSIC (ROC-MUSIC) Algorithm for Bearing Estimation in Impulsive Noise Environments,” IEEE Transactions on Signal Processing, Vol. 44, No. 7, pp. 1623-1633, July 1996.
- [17] Ostman, T., Parkvall, S. and Ottersten, B., “An Improved MUSIC Algorithm for Estimation of Time Delays in Asynchronous DS-CDMA Systems,” IEEE Transactions on Communications, Vol. 47, No. 11, pp. 1628-1631, November 1999.
- [18] Astely, D. and Ottersten, B., “The Effects of Local Scattering on Direction of Arrival Estimation with MUSIC,” IEEE Transactions on Signal Processing, Vol. 47, No. 12, pp. 3220-3234, December 1999.
- [19] Yung-Yi Wang; Jiunn-Tsair Chen; Wen-Hsien Fang, “Joint estimation of DOA and delay using TST-MUSIC in a wireless channel,” IEEE Signal Processing Letters, Vol. 8, No. 2, pp. 58-60, February 2001.
- [20] Hirata, A., Morimoto, T. and Kawasaki, Z., “DOA Estimation of Ultra-wideband EM waves with MUSIC and Interferometry,” Antennas and Wireless Propagation Letters, IEEE, Vol. 2, No. 13, pp. 190-193, 2003.
- [21] A. Paulraj, “Subspace Methods for Direction-of-Arrival Estimation,” Handbook of Statistics 10 - Signal Processing and its Applications, Ed. by N.K. Bose and C.R.Rao Chapter 16, pp.693–640, 1993.
- [22] R.Kumaresan and D.W.Tufts, “Estimating the Angles of Arrival of Multiple Plane Waves,” IEEE Trans. on Aerospace and Electronic Systems, Vol. AES-19, No. 1, pp. 134-139, January 1983.
- [23] Harry L. Van Tree, “Optimum Array Processing,” John Wiley & Sons, Inc., 2002.

- [24] W. Xu and M. Kaveh, "Comparative Study of the biased MUSIC-like Estimator," *Signal Process.*, Vol. 50, pp. 39-55, 1996.
- [25] Boon Poh Ng, "Bearing Estimation using Unity Response Beamforming Approach," *IEE Signal Proceeding*, Vol. 138, No. 6, December 1991.
- [26] Kevin M. Buckley and Xiao-Liang Xu, "Spatial-spectrum Estimation in a Location Sector," *IEEE Trans*, Vol. 38, No. 11, November 1990.
- [27] Tapan K. Sarkar, Jinwan Koh, "A pragmatic approach to adaptive antennas," *IEEE Antennas and Propagation Magazine*, Vol. 42, No.2, April 2000.
- [28] Angel Lozano, Farrokh R. Farrokhi, and Reinaldo A. Valenzuela, Bell Laboratories Lucent Technologies, "Lifting the limits on high-speed wireless data access using antenna arrays," *IEEE Communications Magazine*, September 2001.
- [29] Zhengyuan Xu, "Perturbation analysis for subspace decomposition with applications in subspace-based algorithms," *IEEE Transactions on signal processing*, vol. 50, No.11, November 2002.
- [30] Zhi Tian, Kristine L. Bell, Harry L. Van Trees, "A recursive least squares implementation for LCMP beamforming under quadratic constraint," *IEEE Transaction on signal processing*, vol.49,No.6, June 2001.
- [31] Hassanien A., Shahbazpanahi S., Gershman A.B., "A generalized capon estimator for localization of multiple spread sources," *IEEE Transactions on signal processing*, Vol.52, No. 1, January 2004.
- [32] In Ig Kim, Gyu Tae Park, Kyun Kyung Lee, "Computationally efficient high resolution DOA estimation algorithm," *Electronics Letters* , Vol.37 , No.12 , 7 Jun 2001.

-
- [33] Morrison A., Sharif B.S., Sali S., Hinton O.R., "An iterative DOA algorithm for a space-time DS-CDMA RAKE receiver," 3G Mobile Communication Technologies, IEE Conf. Publ. No. 471, March 2000.
 - [34] Ching-Tai Chiang, Ann-Chen Chang, "DOA estimation in the asynchronous DS-CDMA system," IEEE Transactions on Antennas and Propagation, Vol. 51, No. 1, Jan. 2003.
 - [35] Wei Yang, Junshi Chen, Zhenhui Tan, "An ESPRIT based DOA estimation for CDMA frequency-selective fading channels," 14th IEEE Proceedings on Personal, Indoor and Mobile Radio Communications, Vol.3, September 2003.
 - [36] Kuchar A., Tangemann M., Bonek E., "A real-time DOA-based smart antenna processor," IEEE Transactions on Vehicular Technology, Vol. 51, No. 6 , November 2002.
 - [37] Dandekar K.R., Hao Ling, Guanghan Xu, "Experimental study of mutual coupling compensation in smart antenna applications," IEEE Transactions on Wireless Communications, Vol.1, No.3, July 2002.
 - [38] Shiann-Shiun Jeng, Chia-Yuan Huang, Chih-Yang Lai, Tswen-Jiann Huang, "Performance evaluation of DOA based beamforming in W-CDMA system," Vehicular Technology Conference, 2001. VTC 2001 Fall. IEEE VTS 54th , Volume: 4 , 7-11 Oct. 2001.
 - [39] Johnson, D.H. & Dudgeon, Dan E., "Array signal processing:concepts and techniques," Englewood Cliffs, NJ : P T R Prentice Hall, c1993.
 - [40] Johnson, D.H., "The application of spectral estimation methods to bearing estimation problems," IEEE Proc., 70, (9), pp. 1018-1028,1982.
 - [41] Godara, Lal Chand, "Smart antennas," CRC Press, 2004.
-

-
- [42] Ramdas Kumaresan, "On the zeros of prediction-error filter for deterministic signals," IEEE Trans, vol. Assp-31, No. 1, February 1983.
- [43] B.D. Rao and K.V.S. Hari, "Statistical performance analysis of the minimum-norm method," IEE Proceedings, Vol. 136, June 1989.
- [44] B.D. Rao and K.V.S. Hari, "Performance analysis of root-MUSIC," IEEE Trans, ASSP, Decemeber 1989.
- [45] Z. Rong, "Simulation of adaptive array algorithm for CDMA systems," Master's thesis MPRG-TR-96-31, Mobile & Portable Radio Research Group, Virginia Tech, Blacksburg, VA, September 1996.
- [46] M.G. Larimore and J.R. Treichler, "Convergence behavior of the constant modulus algorithm," IEEE. int. Conf. on acoustics, speech and signal processing, Boston, MA, 1983.
- [47] Chen J.C., Hudson R.E., Kung Yao, "Maximum-likelihood source localization and unknown sensor location estimation for wideband signals in the near-field," IEEE Trans, Vol.50, No. 8, August 2002.
- [48] Tung, T.L., Yao, K., Chen, D., Hudson, R.E. and Reed, C.W., "Source localization and spatial filtering using wideband MUSIC and maximum power beamforming for multimedia applications," Signal Processing systems, 1999. SiPS 99. 1999 IEEE Workshops on, 20-22 Oct. 1999.
- [49] K.K. Buckley, L.J. Griffiths, "Broad-band signal-subspace spatial-spectrum (BASS-ALE) estimation," IEEE Trans. Vol.36, No. 7, July 1988.
- [50] Agrawal M., Prasad S., "Broadband DOA estimation using "Spatial-only" modeling of array data," IEEE Trans., Vol. 48, No. 3, March 2000.

-
- [51] B.P. Ng, M.H. Er and C. Kot, "A MUSIC approach for estimation of directions of arrival of multiple narrowband and broadband sources," *IEE Proc. Signal Processing* 40, 319-323, 1994.
- [52] Ning Ma, Joo Thiam Goh, "Broadband DOA estimation by virtual scaling array," *MTS/IEEE Conference and Exhibition*, Vol. 3, November 2001.
- [53] H. Wang, M. Kaveh, "Coherent signal-subspace processing for the detection and estimation of angles of arrival of multiple wide-band sources," *IEEE Trans.*, 1985.
- [54] Aigang Feng, Zheng Zhao, Qinye Yin, "Wideband direction-of-arrival estimation using fast chirplet-based adaptive signal decomposition algorithm," *Vehicle Technology Conference*, 2001. *VTC 2001 Fall. IEEE VTS 54th*, Vol. 3, October 2001.
- [55] Tuan Do-Hong and Russer P., "An analysis of wideband direction-of-arrival estimation for closely-spaced sources in the presence of array model errors," *IEEE Microwave and Wireless Components Letters*, Vol. 13, No. 8, August 2003.
- [56] Di Claudio E.D., Parisi R., "WAVES: weighted average of signal subspaces for robust wideband direction finding," *IEEE Trans.*, Vol. 49, No. 10, October 2001.
- [57] Brillinger D. R., "Time series: data analysis and theory," Holt, Rhinehart and Winston, New York, 1975.
- [58] Oppenheim A.V. and Schafer R.W., "Digital Signal Processing," Prentice Hall, 1975.

-
- [59] Rao B.D., "Perturbation analysis of a SVD based linear prediction method for estimating the frequencies of multiple sinusoids in noise," IEEE Transaction, ASSP-36 (7), pp. 1026-1035, 1988.
- [60] Stewart G.W., "The perturbation of pseudo-inverses, projections and linear least squares problems," Siam. Rev.,19,(4), pp. 634-662, 1977.

Appendix A

Analysis of the Statistical Performance of UR Algorithm

A.1 Concept of Root Unity Response Method

In the Unity Response algorithm, the DOAs are found by locating the peaks of $D(\omega)$, where

$$D(\omega) = \frac{1}{|H(\omega)|^2} \quad (\text{A.1.1})$$

where $H(\omega)$, termed array response, is defined as

$$H(\omega) = \mathbf{w}^H \mathbf{s}(\omega) \quad (\text{A.1.2})$$

with $\mathbf{w}$ and $\mathbf{s}(\omega)$ being weight vector and normalized steering vector of an M -element linear array, respectively. Where

$$\mathbf{w} = [w_1, w_2, \dots, w_M]^T \quad (\text{A.1.3})$$

$$\mathbf{s}(\omega) = \frac{1}{\sqrt{M}} [1, e^{j\omega}, \dots, e^{j(M-1)\omega}]^T \quad (\text{A.1.4})$$

Note that

$$\omega = \pi d \cos(\theta) \quad (\text{A.1.5})$$

where d is normalized inter-element spacing with respect to half wavelength.

$\mathbf{w}$ is a vector which is completely in the noise subspace and hence,

$$\mathbf{E}_s^H \mathbf{w} = \mathbf{0} \quad (\text{A.1.6})$$

where $\mathbf{E}_s$ spans the signal subspace. Consider the signal scenario where there are K signal sources present in the field, further, if we denote

$$z = e^{j\omega} \quad (\text{A.1.7})$$

It can be seen that the K signal DOAs $e^{j\omega_i}, i = 1, 2, \dots, K$ are roots of the polynomial

$$G(z) = w_1 + w_2 z^{-1} + w_3 z^{-2} + \dots + w_M z^{-(M-1)} \quad (\text{A.1.8})$$

Since

$$H(\omega) = \frac{1}{\sqrt{M}} G(e^{j\omega})^* \quad (\text{A.1.9})$$

The K signal zeros of the polynomial $G(z)$ will result in the peaks in $D(\omega)$. Consequently, instead of examining the peaks of $D(\omega)$, another variation of the approach is to find the signal zeros by rooting $G(z)$, and then extracting the signal frequencies ω_i and subsequently directions of arrival θ_i . This approach, more aptly called the root unity response method, is used here to perform statistical analysis. One thing to note is that rooting Equation (A.1.8) can result in $M - 1$ zeros, among which only K zeros on the unity circle can be used to extract DOA information. Because the following equality must hold

$$|z_i| = |e^{j\omega_i}| = 1 \quad (\text{A.1.10})$$

We name the K zeros on the unity circle Signal Zeros and the other non-unity zeros Extraneous Zeros [42] or simply Noise Zeros.

To conduct statistical analysis, the error in the estimates of the signal zeros z_i is first determined, and then related to the error in the DOA. It is clear that the error in the signal zeros is directly related to the peaks in $D(\omega)$ and should be indicative of the behavior of the original spectrum-based unity response method.

A.2 Derivation of Mean Squared Error

The effect of using an estimated covariance matrix is analyzed in this section. Usually an estimate of the covariance matrix, $\hat{\mathbf{R}}$, is obtained by time averaging T independent snapshots, i.e.

$$\hat{\mathbf{R}} = \frac{1}{T} \sum_{k=1}^T \mathbf{y}_k \mathbf{y}_k^H = \hat{\mathbf{B}} \hat{\mathbf{\Lambda}} \hat{\mathbf{B}}^H \quad (\text{A.2.11})$$

where $\hat{\cdot}$ denotes estimated value, $\mathbf{y}_k$ is the k th spatial sample of the received signal and

$$\hat{\mathbf{B}} = [\hat{\beta}_1, \hat{\beta}_2, \dots, \hat{\beta}_M] \quad (\text{A.2.12})$$

$$\hat{\mathbf{\Lambda}} = \text{diag}(\hat{\lambda}_k) \quad (\text{A.2.13})$$

From the above, estimated signal subspace matrix $\hat{\mathbf{E}}_s$ and noise subspace matrix $\hat{\mathbf{E}}_n$ are as below

$$\hat{\mathbf{E}}_s = [\hat{\beta}_1, \hat{\beta}_2, \dots, \hat{\beta}_K] \quad (\text{A.2.14})$$

$$\hat{\mathbf{E}}_n = [\hat{\beta}_{K+1}, \hat{\beta}_{K+2}, \dots, \hat{\beta}_M] \quad (\text{A.2.15})$$

Let

$$\hat{\boldsymbol{\beta}}_{\mathbf{k}} = \boldsymbol{\beta}_{\mathbf{k}} + \boldsymbol{\eta}_{\mathbf{k}} \quad (\text{A.2.16})$$

$$\hat{\lambda}_k = \lambda_k + \epsilon_k \quad (\text{A.2.17})$$

The analysis makes use of the asymptotic properties of the errors λ_k and ϵ_k . Asymptotically the errors are jointly Gaussian random variables with the errors in the eigenvectors being independent of that in the eigenvalues. Thus, the following property of the errors in the signal eigenvectors are given in [57], i.e.

$$\overline{\boldsymbol{\eta}_{\mathbf{k}} \boldsymbol{\eta}_{\mathbf{j}}^H} \approx \frac{\lambda_k}{T} \sum_{r=1, r \neq k}^M \frac{\lambda_r}{(\lambda_k - \lambda_r)^2} \boldsymbol{\beta}_r \boldsymbol{\beta}_r^H \delta_{kj} \quad (\text{A.2.18})$$

where δ_{kj} is the Kronecker delta and the approximations are $o(T^{-1})$, i.e. the term drops and approaches zero when multiplied by T , and T tends to infinity.

It has been shown that signal DOAs can be found by rooting the polynomial in (A.1.8). The effect of errors in the coefficients w_k on the zeros of $G(z)$ is provided in [58] [59], i.e.

$$\frac{\partial z_i}{\partial w_k} = \frac{-z_i^{-(k-1)}}{\prod_{j=1, j \neq i}^{M-1} (1 - z_j z_i^{-1})} \quad (\text{A.2.19})$$

Using (A.2.19), the error in z_i is given by

$$\begin{aligned} \Delta z_i &\approx \sum_{k=1}^M \frac{\partial z_i}{\partial w_k} \Delta w_k \\ &= \frac{-1}{\prod_{j=1, j \neq i}^{M-1} (1 - z_j z_i^{-1})} [1, z_i^{-1}, z_i^{-2}, \dots, z_i^{-(M-1)}] \Delta \mathbf{w} \\ &= \frac{-\sqrt{M}}{\prod_{j=1, j \neq i}^{M-1} (1 - z_j z_i^{-1})} \mathbf{s}^H(\omega_i) \Delta \mathbf{w} \end{aligned} \quad (\text{A.2.20})$$

The bias in the error and hence the estimate is given by

$$\overline{(\Delta z_i)_{UR}} = \frac{-\sqrt{M}}{\prod_{j=1, j \neq i}^{M-1} (1 - z_j z_i^{-1})} \mathbf{s}^H(\omega_i) \overline{\Delta \mathbf{w}} \quad (\text{A.2.21})$$

and the mean squared error is as follows

$$\begin{aligned} \overline{|\Delta z_i|_{UR}^2} &= \frac{M}{\prod_{j=1, j \neq i}^{M-1} |1 - z_j z_i^{-1}|^2} \mathbf{s}^H(\omega_i) \overline{\Delta \mathbf{w} \Delta \mathbf{w}^H} \mathbf{s}(\omega_i) \\ &= S_{UR} X(\omega_i) \end{aligned} \quad (\text{A.2.22})$$

where

$$S_{UR} = \frac{M}{\prod_{j=1, j \neq i}^{M-1} |1 - z_j z_i^{-1}|^2} \quad (\text{A.2.23})$$

$$X(\omega_i) = \mathbf{s}^H(\omega_i) \overline{\Delta \mathbf{w} \Delta \mathbf{w}^H} \mathbf{s}(\omega_i) \quad (\text{A.2.24})$$

In the above, S_{UR} denotes the sensitivity parameter of UR estimator and $X(\omega_i)$ is used to represent the quantity to be solved.

To determine the mean squared error, an expression for $\mathbf{s}^H(\omega_i) \Delta \mathbf{w}$ is required. From (A.1.6), we have

$$\Delta(\mathbf{E}_s^H \mathbf{w}) = 0 \quad (\text{A.2.25})$$

Hence

$$\Delta \mathbf{E}_s^H \mathbf{w} + \mathbf{E}_s^H \Delta \mathbf{w} = 0 \quad (\text{A.2.26})$$

or

$$\Delta \mathbf{w} = -(\mathbf{E}_s^H)^+ \Delta \mathbf{E}_s^H \mathbf{w} \quad (\text{A.2.27})$$

Therefore

$$\mathbf{s}^H(\omega_i) \Delta \mathbf{w} = -\mathbf{s}^H(\omega_i) (\mathbf{E}_s^H)^+ \Delta \mathbf{E}_s^H \mathbf{w} \quad (\text{A.2.28})$$

This expression can also be derived more rigorously using the results in [60]. To obtain an estimate of the mean squared error, we need an expression for $X(\omega_i)$.

With (A.2.28), it is easily seen

$$\begin{aligned}
 X(\omega_i) &= \mathbf{s}^H(\omega_i) \overline{\Delta \mathbf{w} \Delta \mathbf{w}^H} \mathbf{s}(\omega_i) \\
 &= \mathbf{s}^H(\omega_i) (\mathbf{E}_s^H)^+ \overline{\Delta \mathbf{E}_s^H \mathbf{w} \mathbf{w}^H \Delta \mathbf{E}_s} (\mathbf{E}_s)^+ \mathbf{s}(\omega_i) \\
 &= \mathbf{s}^H(\omega_i) (\mathbf{E}_s^H)^+ \mathbf{C} (\mathbf{E}_s)^+ \mathbf{s}(\omega_i)
 \end{aligned} \tag{A.2.29}$$

where

$$\mathbf{C} = \overline{\Delta \mathbf{E}_s^H \mathbf{w} \mathbf{w}^H \Delta \mathbf{E}_s} \tag{A.2.30}$$

Examining C_{jk} , the jk th element of the $K \times K$ matrix $\mathbf{C}$, we have

$$\begin{aligned}
 C_{jk} &= \overline{\boldsymbol{\eta}_j^H \mathbf{w} \mathbf{w}^H \boldsymbol{\eta}_k} \\
 &= \mathbf{w}^H \overline{\boldsymbol{\eta}_k \boldsymbol{\eta}_j^H} \mathbf{w} \\
 &= \mathbf{w}^H \frac{\lambda_k}{T} \sum_{r=1, r \neq k}^M \frac{\lambda_r}{(\lambda_k - \lambda_r)^2} \boldsymbol{\beta}_r \boldsymbol{\beta}_r^H \delta_{kj} \mathbf{w}
 \end{aligned} \tag{A.2.31}$$

Note the last equality is obtained by substitution of (A.2.18).

Using the fact that $\mathbf{w}$ is in the noise subspace and orthogonal to the signal subspace, we have

$$\begin{aligned}
 C_{jk} &= \mathbf{w}^H \frac{\lambda_k}{T} \sum_{r=K+1}^M \frac{\lambda_r}{(\lambda_k - \lambda_r)^2} \boldsymbol{\beta}_r \boldsymbol{\beta}_r^H \delta_{kj} \mathbf{w} \\
 &= \frac{\lambda_k}{T} \frac{\sigma_n^2}{(\lambda_k^s)^2} \mathbf{w}^H \mathbf{P} \mathbf{w} \delta_{kj} \\
 &= \frac{\sigma_n^2}{T} \frac{\lambda_k}{(\lambda_k^s)^2} \mathbf{w}^H \mathbf{w} \delta_{kj}
 \end{aligned} \tag{A.2.32}$$

The above derivation implies that $\mathbf{C}$ is a diagonal matrix. Note that

$$\mathbf{P} = \sum_{r=K+1}^M \beta_r \beta_r^H \quad (\text{A.2.33})$$

denotes the orthogonal projection operator onto the noise subspace.

It is well known that $\lambda_{K+1} = \lambda_{K+2} = \dots = \lambda_M = \sigma_n^2$, where σ_n^2 is noise power. We also have $\lambda_k^s = \lambda_k - \lambda_r$, for $r = K+1, K+2, \dots, M$, where λ_k^s is signal power.

Substituting (A.2.32) into (A.2.29), it yields

$$\begin{aligned} X(\omega_i) &= \mathbf{s}^H(\omega_i) \overline{\Delta \mathbf{w} \Delta \mathbf{w}^H} \mathbf{s}(\omega_i) \\ &= \frac{\sigma_n^2}{T} \mathbf{w}^H \mathbf{w} \sum_{k=1}^K |\gamma_k|^2 \frac{\lambda_k}{(\lambda_k^s)^2} \end{aligned} \quad (\text{A.2.34})$$

where γ_k 's are elements of the vector $\boldsymbol{\gamma}$, where

$$\boldsymbol{\gamma} = \mathbf{E}_s^+ \mathbf{s}(\omega_i) \approx \mathbf{E}_s^H \mathbf{s}(\omega_i) \quad (\text{A.2.35})$$

Substituting (A.2.34) into (A.2.22), we have

$$\overline{|\Delta z_i|_{UR}^2} = S_{UR} \frac{\sigma_n^2}{T} \mathbf{w}_{UR}^H \mathbf{w}_{UR} \left(\sum_{k=1}^K |\gamma_k|^2 \frac{\lambda_k}{(\lambda_k^s)^2} \right) \quad (\text{A.2.36})$$

where $\mathbf{w}_{UR}$ denotes weight vector obtained from UR algorithm.

So far we have derived an expression for the mean squared error in the estimate of signal zero. The next step is to find an expression for the mean squared error in the DOAs, $\overline{|\Delta \theta_i|_{UR}^2}$, in terms of $\overline{|\Delta z_i|_{UR}^2}$. The work regarding this part has been done in [43] and the answer is readily given

$$\overline{|\Delta \theta_i|_{UR}^2} = \left(\frac{1}{d \pi \cos(\theta_i)} \right)^2 \frac{\overline{|\Delta z_i|_{UR}^2}}{2} \quad (\text{A.2.37})$$

Appendix B

The Closed Form Solution of Quadratic Optimization Problem in UR Algorithm

A typical formulation of quadratic programming problem would be as follows

$$\min_{\mathbf{w}} \quad \mathbf{w}^H \mathbf{R}_x \mathbf{w} \quad (\text{B.0.1})$$

$$\text{subject to} \quad \mathbf{C}^H \mathbf{w} = \mathbf{f} \quad (\text{B.0.2})$$

where $\mathbf{w}$ is a $M \times 1$ weight vector to be determined. $\mathbf{R}_x$ is a $M \times M$ data covariance matrix that is usually obtained by time averaging spatial samples. $\mathbf{C}$ and $\mathbf{f}$ are constraint matrix or vectors of proper dimensions. The above quadratic optimization problem can be solved using method of Lagrange Multipliers [11] [12]. In [30], the solution to (B.0.1) and (B.0.2) is provided

$$\mathbf{w} = \mathbf{R}_x^{-1} \mathbf{C} (\mathbf{C}^H \mathbf{R}_x^{-1} \mathbf{C})^{-1} \mathbf{f} \quad (\text{B.0.3})$$

The Closed Form Solution of Quadratic Optimization Problem in UR Algorithm 27

In unity response algorithm, the quadratic template is again written below

$$\min_{\mathbf{w}} \int_{\theta} |\mathbf{w}_{\text{UR}}^H \mathbf{s}(\theta) - a(\theta)|^2 d\theta \quad (\text{B.0.4})$$

$$\text{subject to} \quad \mathbf{E}_s^H \mathbf{w}_{\text{UR}} = \mathbf{0} \quad (\text{B.0.5})$$

where $a(\theta)$ is a scalar quantity. (B.0.4) can further be expanded

$$\begin{aligned} \int_{\theta} |\mathbf{w}_{\text{UR}}^H \mathbf{s}(\theta) - a(\theta)|^2 d\theta &= \int_{\theta} ((\mathbf{w}_{\text{UR}}^H \mathbf{s}(\theta) - a(\theta))(\mathbf{s}^H(\theta) \mathbf{w}_{\text{UR}} - a(\theta)^*)) d\theta \\ &= \int_{\theta} (\mathbf{w}_{\text{UR}}^H \mathbf{s}(\theta) \mathbf{s}^H(\theta) \mathbf{w}_{\text{UR}} - \mathbf{w}_{\text{UR}}^H \mathbf{s}(\theta) a(\theta)^* \\ &\quad - a(\theta) \mathbf{s}^H(\theta) \mathbf{w}_{\text{UR}} + |a(\theta)|^2) d\theta \\ &= \mathbf{w}_{\text{UR}}^H \mathbf{Q} \mathbf{w}_{\text{UR}} - \mathbf{w}_{\text{UR}}^H \mathbf{P} - \mathbf{P}^H \mathbf{w}_{\text{UR}} + A \end{aligned} \quad (\text{B.0.6})$$

where

$$\mathbf{Q} = \int_{\theta} (\mathbf{s}(\theta) \mathbf{s}^H(\theta)) d\theta \quad (\text{B.0.7})$$

$$\mathbf{P} = \int_{\theta} (a(\theta)^* \mathbf{s}(\theta)) d\theta \quad (\text{B.0.8})$$

$$A = \int_{\theta} |a(\theta)|^2 d\theta \quad (\text{B.0.9})$$

Now we denote

$$\hat{\mathbf{w}} = \mathbf{w}_{\text{UR}} - \mathbf{w}_0 \quad (\text{B.0.10})$$

where $\mathbf{w}_0$ is a constant vector. Thus, we have the following

$$\begin{aligned} \hat{\mathbf{w}}^H \mathbf{Q} \hat{\mathbf{w}} &= (\mathbf{w}_{\text{UR}}^H - \mathbf{w}_0^H) \mathbf{Q} (\mathbf{w}_{\text{UR}} - \mathbf{w}_0) \\ &= \mathbf{w}_{\text{UR}}^H \mathbf{Q} \mathbf{w}_{\text{UR}} - \mathbf{w}_{\text{UR}}^H \mathbf{Q} \mathbf{w}_0 - \mathbf{w}_0^H \mathbf{Q} \mathbf{w}_{\text{UR}} + D \end{aligned} \quad (\text{B.0.11})$$

where

$$D = \mathbf{w}_0^H \mathbf{Q} \mathbf{w}_0 \quad (\text{B.0.12})$$

The Closed Form Solution of Quadratic Optimization Problem in UR Algorithm 128

From the above, if $\mathbf{w}_0$ satisfies $\mathbf{Q}\mathbf{w}_0 = \mathbf{P}$, i.e.

$$\mathbf{w}_0 = \mathbf{Q}^{-1}\mathbf{P} \quad (\text{B.0.13})$$

Thus, it is equivalent to minimize (B.0.6) or (B.0.11). Note A and D are both constants and have no effect on the results. However, the solution to the following quadratic optimization problem can be found

$$\min_{\hat{\mathbf{w}}} \quad \hat{\mathbf{w}}^H \mathbf{Q} \hat{\mathbf{w}} \quad (\text{B.0.14})$$

$$\text{subject to} \quad \mathbf{E}_s^H \hat{\mathbf{w}} = \mathbf{E}_s^H (\mathbf{w}_{\text{UR}} - \mathbf{w}_0) = -\mathbf{E}_s^H \mathbf{w}_0 \quad (\text{B.0.15})$$

Note the last equality holds in (B.0.15) is because $\mathbf{w}_{\text{UR}}$ is in noise subspace and orthogonal to $\mathbf{E}_s$. Using the result in (B.0.3), it yields

$$\hat{\mathbf{w}} = -\mathbf{Q}^{-1}\mathbf{E}_s(\mathbf{E}_s^H \mathbf{Q}^{-1} \mathbf{E}_s)^{-1} \mathbf{E}_s^H \mathbf{w}_0 \quad (\text{B.0.16})$$

Lastly, substituting (B.0.10) and (B.0.13) into (B.0.16), it results in

$$\mathbf{w}_{\text{UR}} = \mathbf{Q}^{-1}\mathbf{P} - \mathbf{Q}^{-1}\mathbf{E}_s(\mathbf{E}_s^H \mathbf{Q}^{-1} \mathbf{E}_s)^{-1} \mathbf{E}_s^H \mathbf{Q}^{-1}\mathbf{P} \quad (\text{B.0.17})$$

Appendix C

Source Code in C for Capon and UR Algorithms

C.1 Main Invoking Function

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <iostream.h>
#include "ClientDSP.h"

void main(void)
{
    FILE* pDatafile;

    int i=0;

    int j=0;

    int temp[8*LEN]={0}; //LEN denotes the snapshot taken
```

```
complex Data[DIM] [LEN]; //DIM represents number of array sensor
                        //DIM and LEN are defined in "ClientDSP.h"

PTheta *pPT, * pMinPT;

double mmin=0;

int    kmin=0;

//open file for read
if((pDatafile=fopen("C:\\data_store.txt", "r"))==NULL)
{
    printf("wrong:cannot open data file\n");
}
else
{
    for(i=0; i<1600; i++)
    {
        fscanf(pDatafile, "%d", &temp[i]);
    }
    fclose(pDatafile);

    //put the captured real data into the matrix Data[] []
    i=0;
    while (i<LEN)
    {
        Data[0][i].re=temp[8*i];
        Data[0][i].im=temp[8*i+1];
        i++;
    }
    i=0;
    while (i<LEN)
```



```

    {
        Data[1][i].re=temp[8*i+2];
        Data[1][i].im=temp[8*i+3];
        i++;
    }
    i=0;
    while (i<LEN)
    {
        Data[2][i].re=temp[8*i+4];
        Data[2][i].im=temp[8*i+5];
        i++;
    }
    i=0;
    while (i<LEN)
    {
        Data[3][i].re=temp[8*i+6];
        Data[3][i].im=temp[8*i+7];
        i++;
    }

    //It is an user-specified option here which algorithm can be used.
    pPT=Capon(Data, DIM);    //DOA estimation using Capon Algorithm
    pPT=UniRes(Data, DIM);   //DOA estimation using unity-response algorithm
    pMinPT=minvalue(pPT, 181); //Locating the peak location
    printf("DOA result: Power=%2.3f   theta=%d\n", pMinPT->P, pMinPT->theta);
}
}

```

C.2 Invoking Function of Capon Estimator

```

PTheta* Capon(complx const data[][LEN], int M)
{

    complx R[4][4],X[4][4]; //R denote covariance matrix, X is the inverse of it.
    PTheta PT[181];          // PT used to store array responses over FOV
    int theta;                //[-90,90]
    double fi=0.0;
    complx a[4][181];         // used to store array steering vector
    int k=0;
    complx pk[181];           //
    complx px[4];             //temp variables
    complx temp2;             //
    //double modu[181]; //store the mouduluses

    int i=0;
    int j=0;
    //initialize the variables
    for (i=0; i<181; i++)
    {
        pk[i].re=0;
        pk[i].im=0;

        for (j=0; j<4; j++)
        {
            a[j][i].re=0;

```

```

        a[j][i].im=0;
    }
}
for (i=0; i<4; i++)
{
    px[i].re=0;
    px[i].im=0;

    for(j=0; j<4; j++)
    {
        R[i][j].re=0;
        R[i][j].im=0;

        X[i][j].re=0;
        X[i][j].im=0;
    }
}

//print for reference
printf("Data-----\n\n");
for (i=0; i<M; i++)
{
    for (j=0; j<LEN; j++)
    {
        printf(" %2.5f+j%2.5f  ", data[i][j].re, data[i][j].im);
    }
    printf("\n\n end %d \n\n", i);
}

```

```

}

covmtrx(data, R);          //R=Data*Data'

//print for reference
printf("R-----\n\n");
for (i=0; i<M; i++)
{
    for (j=0; j<M; j++)
    {
        R[i][j].re=R[i][j].re/LEN;
        R[i][j].im=R[i][j].im/LEN;
        printf("%2.2f+j%2.2f  ", R[i][j].re, R[i][j].im);
    }
    printf("\n");
}
printf("end of R-----\n\n");

//get the inverse matrix of R=> X
inverse (R,X);

//print for reference
printf("X-----\n\n");
for (i=0; i<M; i++)
{
    for (j=0; j<M; j++)
    {

```

```

        printf("%.2e+j%.2e  ", X[i][j].re, X[i][j].im);
    }

    printf("\n");
}

printf("end of X----- \n\n");

//build the steering vector for every theta (-90 to 90)
for (k=0; k<M; k++)
{
    for (theta=-90; theta<=90; theta++)
    {
        fi=PI*sin(theta*(PI/180))*k;
        a[k][theta+90].re=cos(fi);
        a[k][theta+90].im=sin(fi);
    }
}

//for k=1:length(theta) P(k)=1./(a(:,k)'*RINV*a(:,k));
for (k=0; k<181; k++)
{
    for (j=0; j<M; j++)
    {
        px[j].re=0;
        px[j].im=0;
        for (i=0; i<M; i++)
        {
            cmpy(getcjug(a[i][k]),X[i][j], &temp2);

```

```

        cadd(px[j], temp2, &px[j]);
    }
}

for (j=0; j<M; j++)
{
    cmpy(a[j][k], px[j], &temp2);
    cadd(pk[k], temp2, &pk[k]);
}

//modu[k]=getmodul(pk[k]);
PT[k].P=getmodul(pk[k]);
PT[k].theta=k-90;
}

printf("power value -----\\n\\n");
for (i=0; i<181; i++)
{
    printf("%f ", PT[i].P);
}

printf("\\n");

printf("end of power-----\\n\\n");

return PT;
}

```

C.3 Invoking Function of UR Estimator

```

PTheta* UniRes(complx const data[][LEN], int arraynum)
{
    complx R[4][4];    // data covariance matrix
    complx Q[4][4];    // parametric matrix Q
    complx IQ[4][4];   // inverse of Q
    complx P[4];       // parametric vector P
    complx W[4];       // array weight vector
    complx Es[4];      // largest eigen vector
    PTheta PT[181];
    int theta;         // theta range [-90,90]
    double fi=0.0;
    complx a[4][181]; // store array manifold information

    int k=0;
    complx pk[181];
    complx IQP[4];      //IQP=IQ*P
    complx IQEs[4];     //IQEs=IQ*Es
    complx b[181][4];   //b=a';
    complx temp1, temp2, temp3; //temp1=1/(Es'IQEs), temp2=Es'IQP,
                                //temp3=temp1*temp2;

    complx tempresult;

    //initialization
    int i=0;
    int j=0;

```

```

for (i=0; i<181; i++)
{
    pk[i].re=0;
    pk[i].im=0;
    for (j=0; j<arraynum; j++)
    {
        a[j][i].re=0;           //inititalize arraymanifold
        a[j][i].im=0;
    }
}
for (i=0; i<arraynum; i++)
{
    P[i].re=0;                  //inititalize vector P
    P[i].im=0;

    Es[i].re=0;                 //inititalize signal subspace
    Es[i].im=0;

    W[i].re=0;                  //initialize array weight vector
    W[i].im=0;
    for(j=0; j<arraynum; j++)
    {
        R[i][j].re=0;           //inititalize covariance matrix
        R[i][j].im=0;

        Q[i][j].re=0;           //inititalize matrix Q
        Q[i][j].im=0;
    }
}

```

```

        IQ[i][j].re=0;        //inititalize inverse matrix Q
        IQ[i][j].im=0;
    }
}

//print data matrix for reference
printf("Data-----\n\n");
for (i=0; i<arraynum; i++)
{
    for (j=0; j<LEN; j++)
    {
        printf(" %2.5f+j%2.5f  ", data[i][j].re, data[i][j].im);
    }
    printf("\n\n end %d \n\n", i);
}

covmtrx(data, R);        //R=Data*Data'

//print covariance matrix R for reference
printf("R-----\n\n");
for (i=0; i<arraynum; i++)
{
    for (j=0; j<arraynum; j++)
    {
        R[i][j].re=R[i][j].re/LEN;
        R[i][j].im=R[i][j].im/LEN;
        printf("%2.2f+j%2.2f  ", R[i][j].re, R[i][j].im);
    }
}

```

```

    }

    printf("\n");
}

printf("end of R-----\n\n");

//store the steering vector in a[] [] for every theta (-90 to 90)
for (k=0; k<arraynum; k++)
{
    for (theta=-90; theta<=90; theta++)
    {
        fi=PI*sin(theta*(PI/180))*k;
        a[k][theta+90].re=cos(fi);
        a[k][theta+90].im=sin(fi);
    }
}

//Computation of Q and P
//using Arraymanifold stored in a[i][j]
//Q = a x a';
// complx b[181][4];          //b=a';
for (i=0; i<181; i++)
{
    for (j=0; j<4; j++)
    {
        b[i][j].re=a[j][i].re;
        b[i][j].im=-a[j][i].im;
    }
}

```

```

}

mxmpytwo(a, b, Q);

//print matrix Q for reference
printf("Q-----\n\n");
for (i=0; i<arraynum; i++)
{
    for (j=0; j<arraynum; j++)
    {
        printf("%2.2e+j%2.2e  ", Q[i][j].re, Q[i][j].im);
    }
    printf("\n");
}
printf("end of Q----- \n\n");

inverse(Q,IQ); //acquire inverse matrix of Q

//print inverse of matrix Q for reference
printf("IQ-----\n\n");
for (i=0; i<arraynum; i++)
{
    for (j=0; j<arraynum; j++)
    {
        printf("%2.2e+j%2.2e  ", IQ[i][j].re, IQ[i][j].im);
    }
    printf("\n");
}
printf("end of IQ----- \n\n");

```

```

//P is a 4x1 vector that results from
//summation of column vector of arraymanifold
for (i=0; i<4; i++)
{
    for (j=0; j<181; j++)
    {
        P[i].re=P[i].re+a[i][j].re;
        P[i].im=P[i].im+a[i][j].im;
    }
}

// print vector P for reference
printf("vector P-----\n\n");
for (i=0; i<4; i++)
{
    printf("%2.2e+j%2.2e ", P[i].re, P[i].im);
}
printf("\n");
printf("end of vector P-----\n\n");

//Eigendecomposition of covariance matrix R
//and obtain signal subspace Es
eigende(R,Es);

// print signal subspace Es for reference
printf("vector Es-----\n\n");
for (i=0; i<4; i++)

```

```

{
    printf("%2.2e+j%2.2e ", Es[i].re, Es[i].im);
}

printf("\n");

printf("end of vector Es-----\n\n");

//Calculation of array weight W

mxmpythree(IQ, P, IQP);
mxmpythree(IQ, Es, IQEs);
temp1=cdotc(IQEs, Es, 4);
temp1=creciprocal(temp1);
temp2=cdotc(IQP, Es, 4);
cmpy(temp1,temp2,&temp3);

for (i=0; i<4; i++)
{
    cmpy(IQEs[i], temp3, &W[i]);
    csub(IQP[i], W[i], &W[i]);
}

// print array weight W for reference

printf("array weight W-----\n\n");
for (i=0; i<4; i++)
{
    printf("%2.2e+j%2.2e ", W[i].re, W[i].im);
}

printf("\n");

```

```

printf("end of array weight W-----\n\n");

//calculation of array response over field of view

for (i=0; i<181; i++)
{
    for (j=0; j<4; j++)
    {
        cmpy(getcjug(W[j]),a[j][i], &tempresult);
        cadd(pk[i], tempresult, &pk[i]);
    }
    PT[i].P=getmodul(pk[i]);
    PT[i].theta=i-90;
}

// print power value corresponding to each angle
printf("power value ----- \n\n");
for (i=0; i<181; i++)
{
    printf("%e ", PT[i].P);
}

printf("\n");

printf("end of power-----\n\n");

return PT;
}

```

Appendix D

Pre-defined Functions in “ClientDSP.h”

```
#pragma once

#define TRUE    1
#define FALSE   0
#define DIM     4
#define LEN     200
#define PI      3.1415926

//define complex data structure
typedef struct complx
{
    double    re;
    double    im;
}complx;
```

```
//complex multiplier  a*b->c
void cmpy(complx a, complx b, complx *c)
{
    c->re=a.re*b.re-a.im*b.im;
    c->im=a.im*b.re+a.re*b.im;
}

// complex number addition
void cadd(complx a, complx b, complx *c)
{
    c->re =a.re+b.re;
    c->im =a.im+b.im;
}

//complex number subtraction
void csub(complx a, complx b, complx *c)
{
    c->re =a.re-b.re;
    c->im =a.im-b.im;
}

// get the conjugate complex
complx getcjug(complx a)
{
    complx b;
```



```
    b.re=a.re;

    b.im=-a.im;

    return b;
}

//complex vector dot comply  a[]*b[]=
complex cdotc(complex a[], complex b[], int num)
{
    complex temp, c;
    int i;

    c.re=0;
    c.im=0;

    for (i=0; i<num; i++)
    {
        cmpy(a[i], getcjug(b[i]), &temp);
        cadd(temp, c, &c);
    }
    return c;
}

//compute the modulus of the complex number
double getmodul(complex a)
{
```

```
double m=0;

m=pow(a.re, 2)+pow(a.im, 2);

m=sqrt(m);

return m;
}

//print one row of complex matrix
void cprint(complx a[], int num)
{
    int i;
    for(i=0; i<num; i++)
    {
        printf("%f+j%f \n", a[i].re,a[i].im);
    }
}

//computation of the inverse matrix of a 4x4 square matrix
void inverse(complx const a[4][4], complx b[4][4])
{
    complx t1, result,t2;

    // b[0][0]=
    cmpy(a[1][1], a[2][2], &t1);
    cmpy(a[3][3],t1,&result);
```

```
cmpy(a[1][1], a[2][3], &t2);
```

```
cmpy(t2, a[3][2], &t1);
```

```
csub(result, t1, &result);
```

```
cmpy(a[2][1], a[1][2], &t2);
```

```
cmpy(t2, a[3][3], &t1);
```

```
csub(result, t1, &result);
```

```
cmpy(a[2][1], a[1][3], &t2);
```

```
cmpy(t2, a[3][2], &t1);
```

```
cadd(result, t1, &result);
```

```
cmpy(a[3][1], a[1][2], &t2);
```

```
cmpy(t2, a[2][3], &t1);
```

```
cadd(result, t1, &result);
```

```
cmpy(a[3][1], a[1][3], &t2);
```

```
cmpy(t2, a[2][2], &t1);
```

```
csub(result, t1, &result);
```

```
b[0][0] = result;
```

```
// b[0][1] = -(
```

```
cmpy(a[0][1], a[2][2], &t2);
```

```
cmpy(t2, a[3][3], &result);
```

```
cmpy(a[0][1], a[2][3], &t2);
```

```

    cmpy(t2,a[3][2],&t1);
    csub(result,t1,&result);

    cmpy(a[2][1],a[0][2],&t2);
    cmpy(t2,a[3][3],&t1);
    csub(result,t1,&result);

    cmpy(a[2][1],a[0][3],&t2);
    cmpy(t2,a[3][2],&t1);
    cadd(result,t1,&result);

    cmpy(a[3][1],a[0][2],&t2);
    cmpy(t2,a[2][3],&t1);
    cadd(result,t1,&result);

    cmpy(a[3][1],a[0][3],&t2);
    cmpy(t2,a[2][2],&t1);
    csub(result,t1,&result);

    b[0][1].re = (-1)*result.re;
    b[0][1].im = (-1)*result.im;

    // b[0][2]=
    cmpy(a[0][1],a[1][2],&t2);
    cmpy(t2,a[3][3],&result);

    cmpy(a[0][1],a[1][3],&t2);

```

```

    cmpy(t2,a[3][2],&t1);
    csub(result,t1,&result);

    cmpy(a[1][1],a[0][2],&t2);
    cmpy(t2,a[3][3],&t1);
    csub(result,t1,&result);

    cmpy(a[1][1],a[0][3],&t2);
    cmpy(t2,a[3][2],&t1);
    cadd(result,t1,&result);

    cmpy(a[3][1],a[0][2],&t2);
    cmpy(t2,a[1][3],&t1);
    cadd(result,t1,&result);

    cmpy(a[3][1],a[0][3],&t2);
    cmpy(t2,a[1][2],&t1);
    csub(result,t1,&result);
    b[0][2] = result;

    // b[0][3]= -(
    cmpy(a[0][1],a[1][2],&t2);
    cmpy(t2,a[2][3],&result);

    cmpy(a[0][1],a[1][3],&t2);
    cmpy(t2,a[2][2],&t1);
    csub(result,t1,&result);

```

```

    cmpy(a[1][1],a[0][2],&t2);
    cmpy(t2,a[2][3],&t1);
    csub(result,t1,&result);

    cmpy(a[1][1],a[0][3],&t2);
    cmpy(t2,a[2][2],&t1);
    cadd(result,t1,&result);

    cmpy(a[2][1],a[0][2],&t2);
    cmpy(t2,a[1][3],&t1);
    cadd(result,t1,&result);

    cmpy(a[2][1],a[0][3],&t2);
    cmpy(t2,a[1][2],&t1);
    csub(result,t1,&result);

    b[0][3].re = (-1) * result.re;
    b[0][3].im = (-1) * result.im;

    // b[1][0]=
    cmpy(a[1][0],a[2][2],&t2);
    cmpy(t2,a[3][3],&result);
    result.re = (-1)*result.re;
    result.im = (-1)*result.im;

    cmpy(a[1][0],a[2][3],&t2);

```

```
    cmpy(t2,a[3][2],&t1);
    cadd(result,t1,&result);

    cmpy(a[2][0],a[1][2],&t2);
    cmpy(t2,a[3][3],&t1);
    cadd(result,t1,&result);

    cmpy(a[2][0],a[1][3],&t2);
    cmpy(t2,a[3][2],&t1);
    csub(result,t1,&result);

    cmpy(a[3][0],a[1][2],&t2);
    cmpy(t2,a[2][3],&t1);
    csub(result,t1,&result);

    cmpy(a[3][0],a[1][3],&t2);
    cmpy(t2,a[2][2],&t1);
    cadd(result,t1,&result);
    b[1][0] = result;

    // b[1][1]=
    cmpy(a[0][0],a[2][2],&t2);
    cmpy(t2,a[3][3],&result);

    cmpy(a[0][0],a[2][3],&t2);
    cmpy(t2,a[3][2],&t1);
    csub(result,t1,&result);
```

```
    cmpy(a[2][0],a[0][2],&t2);
    cmpy(t2,a[3][3],&t1);
    csub(result,t1,&result);

    cmpy(a[2][0],a[0][3],&t2);
    cmpy(t2,a[3][2],&t1);
    cadd(result,t1,&result);

    cmpy(a[3][0],a[0][2],&t2);
    cmpy(t2,a[2][3],&t1);
    cadd(result,t1,&result);

    cmpy(a[3][0],a[0][3],&t2);
    cmpy(t2,a[2][2],&t1);
    csub(result,t1,&result);
    b[1][1] = result;

    // b[1][2]=
    cmpy(a[0][0],a[1][2],&t2);
    cmpy(t2,a[3][3],&result);
    result.re = (-1)*result.re;
    result.im = (-1)*result.im;

    cmpy(a[0][0],a[1][3],&t2);
    cmpy(t2,a[3][2],&t1);
    cadd(result,t1,&result);
```



```

    cmpy(a[1][0],a[0][2],&t2);
    cmpy(t2,a[3][3],&t1);
    cadd(result,t1,&result);

    cmpy(a[1][0],a[0][3],&t2);
    cmpy(t2,a[3][2],&t1);
    csub(result,t1,&result);

    cmpy(a[3][0],a[0][2],&t2);
    cmpy(t2,a[1][3],&t1);
    csub(result,t1,&result);

    cmpy(a[3][0],a[0][3],&t2);
    cmpy(t2,a[1][2],&t1);
    cadd(result,t1,&result);

    b[1][2] = result;

    // b[1][3]= -(
    cmpy(a[0][0],a[1][2],&t2);
    cmpy(t2,a[2][3],&result);
    result.re = (-1)*result.re;
    result.im = (-1)*result.im;

    cmpy(a[0][0],a[1][3],&t2);
    cmpy(t2,a[2][2],&t1);

```

```
cadd(result,t1,&result);
```

```
cmpy(a[1][0],a[0][2],&t2);
```

```
cmpy(t2,a[2][3],&t1);
```

```
cadd(result,t1,&result);
```

```
cmpy(a[1][0],a[0][3],&t2);
```

```
cmpy(t2,a[2][2],&t1);
```

```
csub(result,t1,&result);
```

```
cmpy(a[2][0],a[0][2],&t2);
```

```
cmpy(t2,a[1][3],&t1);
```

```
csub(result,t1,&result);
```

```
cmpy(a[2][0],a[0][3],&t2);
```

```
cmpy(t2,a[1][2],&t1);
```

```
cadd(result,t1,&result);
```

```
b[1][3].re = (-1)*result.re;
```

```
b[1][3].im = (-1)*result.im;
```

```
//b[2][0]= -(
```

```
cmpy(a[1][0],a[2][1],&t2);
```

```
cmpy(t2,a[3][3],&result);
```

```
result.re = (-1)*result.re;
```

```
result.im = (-1)*result.im;
```

```

    cmpy(a[1][0],a[2][3],&t2);
    cmpy(t2,a[3][1],&t1);
    cadd(result,t1,&result);

    cmpy(a[2][0],a[1][1],&t2);
    cmpy(t2,a[3][3],&t1);
    cadd(result,t1,&result);

    cmpy(a[2][0],a[1][3],&t2);
    cmpy(t2,a[3][1],&t1);
    csub(result,t1,&result);

    cmpy(a[3][0],a[1][1],&t2);
    cmpy(t2,a[2][3],&t1);
    csub(result,t1,&result);

    cmpy(a[3][0],a[1][3],&t2);
    cmpy(t2,a[2][1],&t1);
    cadd(result,t1,&result);
    b[2][0].re = (-1)*result.re;
    b[2][0].im = (-1)*result.im;

    // b[2][1]=
    cmpy(a[0][0],a[2][1],&t2);
    cmpy(t2,a[3][3],&result);
    result.re = (-1)*result.re;
    result.im = (-1)*result.im;

```

```
    cmpy(a[0][0],a[2][3],&t2);
    cmpy(t2,a[3][1],&t1);
    cadd(result,t1,&result);

    cmpy(a[2][0],a[0][1],&t2);
    cmpy(t2,a[3][3],&t1);
    cadd(result,t1,&result);

    cmpy(a[2][0],a[0][3],&t2);
    cmpy(t2,a[3][1],&t1);
    csub(result,t1,&result);

    cmpy(a[3][0],a[0][1],&t2);
    cmpy(t2,a[2][3],&t1);
    csub(result,t1,&result);

    cmpy(a[3][0],a[0][3],&t2);
    cmpy(t2,a[2][1],&t1);
    cadd(result,t1,&result);
    b[2][1] = result;

    // b[2][2]= -(
    cmpy(a[0][0],a[1][1],&t2);
    cmpy(t2,a[3][3],&result);
    result.re = (-1)*result.re;
    result.im = (-1)*result.im;
```

```
    cmpy(a[0][0],a[1][3],&t2);
    cmpy(t2,a[3][1],&t1);
    cadd(result,t1,&result);

    cmpy(a[1][0],a[0][1],&t2);
    cmpy(t2,a[3][3],&t1);
    cadd(result,t1,&result);

    cmpy(a[1][0],a[0][3],&t2);
    cmpy(t2,a[3][1],&t1);
    csub(result,t1,&result);

    cmpy(a[3][0],a[0][1],&t2);
    cmpy(t2,a[1][3],&t1);
    csub(result,t1,&result);

    cmpy(a[3][0],a[0][3],&t2);
    cmpy(t2,a[1][1],&t1);
    cadd(result,t1,&result);
    b[2][2].re = (-1)*result.re;
    b[2][2].im = (-1)*result.im;

    // b[2][3]=
    cmpy(a[0][0],a[1][1],&t2);
    cmpy(t2,a[2][3],&result);
    result.re = (-1)*result.re;
```

```
result.im = (-1)*result.im;
```

```
cmpy(a[0][0],a[1][3],&t2);
```

```
cmpy(t2,a[2][1],&t1);
```

```
cadd(result,t1,&result);
```

```
cmpy(a[1][0],a[0][1],&t2);
```

```
cmpy(t2,a[2][3],&t1);
```

```
cadd(result,t1,&result);
```

```
cmpy(a[1][0],a[0][3],&t2);
```

```
cmpy(t2,a[2][1],&t1);
```

```
csub(result,t1,&result);
```

```
cmpy(a[2][0],a[0][1],&t2);
```

```
cmpy(t2,a[1][3],&t1);
```

```
csub(result,t1,&result);
```

```
cmpy(a[2][0],a[0][3],&t2);
```

```
cmpy(t2,a[1][1],&t1);
```

```
cadd(result,t1,&result);
```

```
b[2][3] = result;
```

```
//b[3][0]=
```

```
cmpy(a[1][0],a[2][1],&t2);
```

```
cmpy(t2,a[3][2],&result);
```

```
result.re = (-1)*result.re;
```

```
result.im = (-1)*result.im;
```

```
cpy(a[1][0],a[2][2],&t2);
```

```
cpy(t2,a[3][1],&t1);
```

```
cadd(result,t1,&result);
```

```
cpy(a[2][0],a[1][1],&t2);
```

```
cpy(t2,a[3][2],&t1);
```

```
cadd(result,t1,&result);
```

```
cpy(a[2][0],a[1][2],&t2);
```

```
cpy(t2,a[3][1],&t1);
```

```
csub(result,t1,&result);
```

```
cpy(a[3][0],a[1][1],&t2);
```

```
cpy(t2,a[2][2],&t1);
```

```
csub(result,t1,&result);
```

```
cpy(a[3][0],a[1][2],&t2);
```

```
cpy(t2,a[2][1],&t1);
```

```
cadd(result,t1,&result);
```

```
b[3][0] = result;
```

```
// b[3][1]= -(
```

```
cpy(a[0][0],a[2][1],&t2);
```

```
cpy(t2,a[3][2],&result);
```

```
result.re = (-1)*result.re;
```

```

    result.im = (-1)*result.im;

    cmpy(a[0][0],a[2][2],&t2);
    cmpy(t2,a[3][1],&t1);
    cadd(result,t1,&result);

    cmpy(a[2][0],a[0][1],&t2);
    cmpy(t2,a[3][2],&t1);
    cadd(result,t1,&result);

    cmpy(a[2][0],a[0][2],&t2);
    cmpy(t2,a[3][1],&t1);
    csub(result,t1,&result);

    cmpy(a[3][0],a[0][1],&t2);
    cmpy(t2,a[2][2],&t1);
    csub(result,t1,&result);

    cmpy(a[3][0],a[0][2],&t2);
    cmpy(t2,a[2][1],&t1);
    cadd(result,t1,&result);
    b[3][1].re = (-1)*result.re;
    b[3][1].im = (-1)*result.im;

    // b[3][2]=
    cmpy(a[0][0],a[1][1],&t2);
    cmpy(t2,a[3][2],&result);

```



```

    cmpy(a[0][0],a[1][2],&t2);
    cmpy(t2,a[3][1],&t1);
    csub(result,t1,&result);

    cmpy(a[1][0],a[0][1],&t2);
    cmpy(t2,a[3][2],&t1);
    csub(result,t1,&result);

    cmpy(a[1][0],a[0][2],&t2);
    cmpy(t2,a[3][1],&t1);
    cadd(result,t1,&result);

    cmpy(a[3][0],a[0][1],&t2);
    cmpy(t2,a[1][2],&t1);
    cadd(result,t1,&result);

    cmpy(a[3][0],a[0][2],&t2);
    cmpy(t2,a[1][1],&t1);
    csub(result,t1,&result);
    b[3][2].re = (-1)*result.re;
    b[3][2].im = (-1)*result.im;

    // b[3][3]= -(
    cmpy(a[0][0],a[1][1],&t2);
    cmpy(t2,a[2][2],&result);
    result.re = (-1)*result.re;

```

```

    result.im = (-1)*result.im;

    cmpy(a[0][0],a[1][2],&t2);
    cmpy(t2,a[2][1],&t1);
    cadd(result,t1,&result);

    cmpy(a[1][0],a[0][1],&t2);
    cmpy(t2,a[2][2],&t1);
    cadd(result,t1,&result);

    cmpy(a[1][0],a[0][2],&t2);
    cmpy(t2,a[2][1],&t1);
    csub(result,t1,&result);

    cmpy(a[2][0],a[0][1],&t2);
    cmpy(t2,a[1][2],&t1);
    csub(result,t1,&result);

    cmpy(a[2][0],a[0][2],&t2);
    cmpy(t2,a[1][1],&t1);
    cadd(result,t1,&result);
    b[3][3].re = (-1)*result.re;
    b[3][3].im = (-1)*result.im;
    return;
}

//compute covariance matrix from data matrix

```

```
void covmtrx(complx const a[4][200], complx c[4][4])
{
    int i=0;
    int j=0;
    complx temp;

    for (i=0; i<4; i++)
    {
        for (j=0; j<LEN; j++)
        {
            cmpy(a[i][j], getcjug(a[0][j]), &temp);
            cadd(temp, c[i][0], &c[i][0]);

            cmpy(a[i][j], getcjug(a[1][j]), &temp);
            cadd(temp, c[i][1], &c[i][1]);

            cmpy(a[i][j], getcjug(a[2][j]), &temp);
            cadd(temp, c[i][2], &c[i][2]);

            cmpy(a[i][j], getcjug(a[3][j]), &temp);
            cadd(temp, c[i][3], &c[i][3]);
        }
    }
}

//define data structure
typedef struct PTheta
```

```

{
    double P;                //power on each theta
    int    theta;            //theta 0~180

}PTheta;

//Function to get the minimum value of the array responses
PTheta* minvalue(PTheta a[], int num)
{
    int i, tempi;
    double mmoud;

    mmoud=a[0].P;
    tempi=0;

    for (i=1; i<num; i++)
    {
        if (mmoud>a[i].P)
        {
            mmoud=a[i].P;
            tempi=i;
        }
    }
    return &a[tempi];
}

// take the reciprocal of a complex number

```

```
complex creciprocal(complex a)
{
    double m=0;
    complex b;
    m=pow(a.re, 2)+pow(a.im, 2);
    b.re=a.re/m;
    b.im=-a.im/m;
    return b;
}

// multiplication of 2 4x4 square matrix
void mxmpyone(complex a[4][4], complex b[4][4], complex c[4][4])
{
    int j=0;
    int i=0;
    int q=0;
    complex temp;

    for(i=0; i<4; i++)    //initialization
    {
        for(j=0; j<4; j++)
        {
            c[i][j].re=0;
            c[i][j].im=0;
        }
    }
}
```

```
    for (i=0; i<4; i++)    // matrix operation
    {
        for (q=0; q<4; q++)
        {
            for (j=0; j<4; j++)
            {
                cpy(a[i][j], b[j][q], &temp);
                cadd(temp, c[i][q], &c[i][q]);
            }
        }
    }

}

// 4x181 matrix multiplication with a 181x4 matrix
void mxmpytwo(complx a[4][181], complx b[181][4], complx c[4][4] )
{
    int j=0;
    int i=0;
    int q=0;
    complx temp;

    for(i=0; i<4; i++)    //initialization to 0
    {
        for(j=0; j<4; j++)
```

```
        {
            c[i][j].re=0;
            c[i][j].im=0;
        }
    }

    for (i=0; i<4; i++)    // matrix operation
    {
        for (q=0; q<4; q++)
        {
            for (j=0; j<181; j++)
            {
                cmpy(a[i][j], b[j][q], &temp);
                cadd(temp, c[i][q], &c[i][q]);
            }
        }
    }
}

// 4x4 matrix multiplied with 4x1 matrix
void mxmpythree(complx a[4][4], complx b[4], complx c[4])
{
    int j=0;
    int i=0;
    complx temp;
```

```
    for(i=0; i<4; i++)    //initialization to 0
    {
        c[i].re=0;
        c[i].im=0;
    }

    for (i=0; i<4; i++)    // matrix operation
    {
        for (j=0; j<4; j++)
        {
            cmpy(a[i][j], b[j], &temp);
            cadd(temp, c[i], &c[i]);
        }
    }

}

// function to find the complex number with the max modulus in a vector
double maxmodu(complx a[], int num)
{
    int i, tempi;
    double modu;

    modu=getmodul(a[0]);
    tempi=0;

    for (i=1; i<num; i++)
```



```

    {
        if (modu<getmodul(a[i]))
        {
            modu=getmodul(a[i]);
            tempi=i;
        }
    }
    return modu;
}

//find eigenvector corresponding to the largest eigenvalue
//of a 4x4 square matrix using an iterative approaches named power method
void eigende(complx a[4][4], complx Es[4])
{
    complx x[4];
    double maxx;
    int i, k;

    //initialize x0
    x[0].re=1; x[0].im=1; x[1].re=1; x[1].im=1;
    x[2].re=1; x[2].im=1; x[3].re=1; x[3].im=1;

    for (k=0; k<100; k++)
    {
        maxx=maxmodu(x, 4);
        for (i=0; i<4; i++)
        {

```

```
        Es[i].re=x[i].re/maxx;

        Es[i].im=x[i].im/maxx;
    }

    mxmpythree(a, Es, x);
}

// find the determinant of a 2x2 matrix
complex deter2(complex s[2][2])
{
    complex t1;
    complex t2;
    complex m;
    m.re=0;
    m.im=0;

    cmpy(s[0][0], s[1][1], &t1);
    cmpy(s[0][1], s[1][0], &t2);
    csub(t1, t2, &m);

    return m;
}
```