
Evaluation of Selected Subspace Tracking Algorithms for Direction Finding

Ryan MITCHLEY

Thesis presented in partial fulfilment of the requirements for the degree of Master of
Science in Engineering (MScEng) at the University of Stellenbosch



UNIVERSITEIT • STELLENBOSCH • UNIVERSITY

Supervised by
Professor Johan LOURENS

March 2007

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signed:

.....

Date:

.....



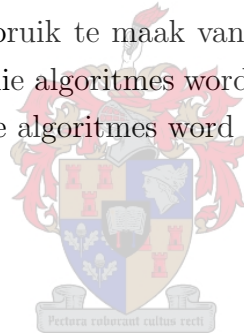
Abstract

This thesis examines three reduced complexity algorithms for subspace tracking in the context of radio direction finding. Projection Approximation Subspace Tracking (Yang), the Fast Data Projection Method (Doukopoulos and Moustakides) and OPERA (MacInnes) are presented and compared in terms of performance and efficiency. The algorithms' performances are contrasted using step changes in direction of arrival, sensitivity to noise and minimum angular discrimination. Their relative efficiencies are examined by comparing their theoretical complexities and by assessing benchmark results. The construction of a PC based signal simulator and direction finding client using the MUSIC algorithm are described. The results of a field assessment of the algorithms are presented, and finally, conclusions regarding the relative merits of the algorithms are drawn.



Abstrak

Hierdie tesis ondersoek drie gereduseerde komplekse algoritmes vir “subspace” volging met betrekking tot radio rigting peiling. Projection Approximation Subspace Tracking (Yang), die Fast Data Projection Method (Doukopoulos and Moustakides) en OPERA (MacInnes) word hier vertoon en vergelyk in terme van prestasie en effektiwiteit. Die algoritme prestasies word vergelyk met die gebruik van stap veranderings in rigting van aankoms, met betrekking tot sensitiviteit teen ruis en minimum hoek diskriminasie. Hulle relatiewe effektiwiteit word ondersoek deur vergelyking van teoretiese kompleksiteite en om ”benchmark” resultate te beskou. Die bou van ’n PC gebaseerde sein simuleerder en rigting peiler klient deur gebruik te maak van MUSIC algoritmes word beskryf. Die resultate van ’n veld toets van die algoritmes word voorgestel en die finale gevolgtrekking tot die relatiewe meriete van die algoritmes word geskets.



Acknowledgments

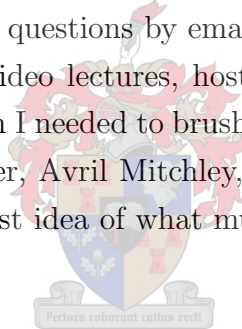
I would firstly like to thank Professor Johan Lourens for his support, guidance and patience during the completion of this thesis.

Peralex Electronics, in particular Trevor Bartleet, Alex Bassios and Per Karlsen, provided me with the time and resources required to guide this project to completion. Special thanks go to Hannes Coetzee and Leon M. Van Niekerk at Grintek Ewation in Pretoria, who helped greatly with the field tests, transporting diesel generators, radio transceivers and broadcast antennas around Pretoria!

I also wish to thank Dr Craig MacInnes and Professor G. Moustakides who patiently answered my (sometimes naïve) questions by email.

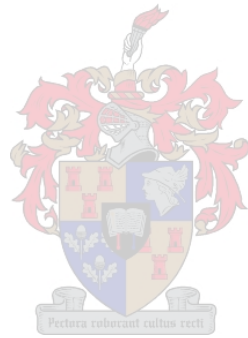
Professor Gilbert Strang's video lectures, hosted on the MIT OpenCourseWare web pages [18], were very useful when I needed to brush up on certain aspects of linear algebra.

Thanks also go to my mother, Avril Mitchley, who proofread the spelling and grammar (without having the slightest idea of what much of it meant!).



Soli Deo Gloria

*Two TV aerials met on a roof - fell in love - and got married.
The ceremony was rubbish, but the reception was brilliant!*



Contents

1	Introduction and Objectives	1
2	Subspace methods for direction finding	3
2.1	Spatial sampling by an antenna array	3
2.2	The array covariance matrix	7
2.2.1	Estimation of the array covariance matrix	8
2.2.2	The form of a typical array covariance matrix	9
2.3	Eigen decomposition of the array covariance matrix	12
2.4	The MUSIC spectrum and Direction-Of-Arrival	16
2.5	Reduced complexity algorithms for subspace tracking	20
2.5.1	Projection Approximation Subspace Tracking	21
2.5.2	Fast Data Projection Method (FDPM)	24
2.5.3	Operator Restriction Analysis (OPERA)	27
3	Performance and efficiency of subspace tracking algorithms	31
3.1	Creation of a software signal simulator	31
3.2	Creation of a Superresolution Direction Finding client	34
3.3	Response to a step change in Direction-Of-Arrival	36
3.3.1	Step change in DOA of one signal	36
3.3.2	Step change in Direction-Of-Arrival of two signals	44
3.3.3	Sensitivity to noise	50
3.3.4	Minimum angular separation	53
3.3.5	Conclusions regarding performance of the selected algorithms	55
3.4	Efficiency of the selected subspace tracking algorithms	57
3.4.1	The benchmark program	58
3.4.2	Complexity analysis	68
3.4.3	Conclusions regarding efficiency of the selected algorithms	69
4	Performance of subspace tracking algorithms using real-world data	70
4.1	Equipment and test setup	70

4.2	Calibration of the narrowband results	78
4.3	CW (constant sinusoidal) signal measurements	80
4.3.1	Location 1	82
4.3.2	Location 2	86
4.3.3	Location 3	90
4.3.4	Location 4	94
4.3.5	Location 5	98
4.3.6	Simulation of the field tests	102
4.3.7	Conclusions regarding the field tests	114
5	Conclusions	115
A	MATLAB code for the selected subspace tracking methods	120
A.1	PASTd	120
A.2	FDPM	121
A.3	EVD-OPERA	122
B	Some notes regarding the software simulator	124



List of Figures

2.1	Plot of AIC and MDL criteria vs. signal subspace dimension	15
2.2	Direction-Of-Arrival power spectrum (signal subspace)	17
2.3	MUSIC Direction-Of-Arrival power spectrum (noise subspace)	18
2.4	3D MUSIC DOA power spectrum, a function of azimuth and elevation. 90° elevation is at the centre. 0° elevation is toward the extremes of the unit circle. Azimuth is 0° at $x = 0, y = 1$ and 90° at $x = 1, y = 0$	19
3.1	Screenshot of the antenna array signal simulator	32
3.2	Screenshot of the SRDF client	34
3.3	Azimuth step response (one signal) using the eigen decomposition and signal subspace	37
3.4	Azimuth step response (one signal) using the eigen decomposition and signal subspace	38
3.5	Azimuth step response (one signal) using the eigen decomposition and signal subspace	39
3.6	Azimuth step response (one signal) using PASTd and the signal subspace .	40
3.7	Azimuth step response (one signal) using FDPM and the signal subspace .	41
3.8	Azimuth step response (one signal) using FDPM and the noise subspace .	42
3.9	Azimuth step response (one signal) using EVD-OPERA and the signal subspace	43
3.10	Azimuth step response (two signals) using the eigen decomposition and signal subspace	45
3.11	Azimuth step response (two signals) using PASTd and the signal subspace	46
3.12	Azimuth step response (two signals) using FDPM algorithm and the signal subspace	47
3.13	Azimuth step response (two signals) using FDPM algorithm and the noise subspace	48
3.14	Azimuth step response (two signals) using OPERA algorithm and the noise subspace	49
3.15	Average error in azimuth versus Signal-to-Noise-Ratio	51
3.16	Minimum angular separation as a function of frequency	54

3.17	Execution time of tracking algorithms versus number of elements	63
3.17	Execution time of tracking algorithms versus number of elements	64
3.17	Execution time of tracking algorithms versus number of elements (cont'd) .	65
3.17	Execution time of tracking algorithms versus number of elements (cont'd) .	66
3.18	Execution time of tracking algorithms compared to the eigen decomposition	67
4.1	Subspace tracking DF tests: antenna array layout	71
4.2	Profile of each antenna element (MRA2004), shown upside-down	72
4.3	A picture of two of the monopole antenna elements in the field	73
4.4	Base of a monopole element, showing the ground straps used	74
4.5	Aerial view of the field in which the antenna array was set up	75
4.6	Aerial view of the locations used for test measurements	76
4.7	MRD5000W 10-channel receiver and MRX3000W10 Data Processing Unit	77
4.8	Calibration correction tables	79
4.9	Location 1 (327.7°): DOA spectra	83
4.9	Location 1 (327.7°): DOA spectra (cont'd)	84
4.9	Location 1 (327.7°): DOA spectra (cont'd)	85
4.10	Location 2 (222.4°): DOA spectra	87
4.10	Location 2 (222.4°): DOA spectra (cont'd)	88
4.10	Location 2 (222.4°): DOA spectra (cont'd)	89
4.11	Location 3 (207.3°): DOA spectra	91
4.11	Location 3 (207.3°): DOA spectra (cont'd)	92
4.11	Location 3 (207.3°): DOA spectra (cont'd)	93
4.12	Location 4 (112.1°): DOA spectra	95
4.12	Location 4 (112.1°): DOA spectra (cont'd)	96
4.12	Location 4 (112.1°): DOA spectra (cont'd)	97
4.13	Location 5 (38.3°): DOA spectra	99
4.13	Location 5 (38.3°): DOA spectra (cont'd)	100
4.13	Location 5 (38.3°): DOA spectra (cont'd)	101
4.14	Location 1 (327.7°): DOA spectra	103
4.14	Location 1 (327.7°): DOA spectra (cont'd)	104
4.15	Location 2 (222.44°): DOA spectra	105
4.15	Location 2 (222.44°): DOA spectra (cont'd)	106
4.16	Location 3 (207.25°): DOA spectra	107
4.16	Location 3 (207.25°): DOA spectra (cont'd)	108
4.17	Location 4 (112.13°): DOA spectra	109
4.17	Location 4 (112.13°): DOA spectra (cont'd)	110
4.18	Location 5 (38.25°): DOA spectra	111
4.18	Location 5 (38.25°): DOA spectra (cont'd)	112

4.19 Location 1 (327.7°): DOA spectra 113

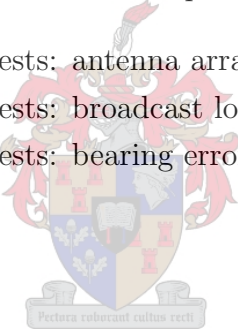


List of Tables

- 2.1 Ratio of Δ_d to $\frac{c}{B}$ for several typical signals and inter-element spacings . . . 4

- 3.1 Subspace tracking benchmarks: target platform 58
- 3.2 Time [s] to calculate one million subspace tracking updates: one signal . . 60
- 3.3 Time [s] to calculate one million subspace tracking updates: two signals . . 61
- 3.4 Time [s] to calculate one million subspace tracking updates: three signals . 61
- 3.5 Time [s] to calculate one million subspace tracking updates: four signals . . 62
- 3.6 Time [s] to calculate one million subspace tracking updates: five signals . . 62

- 4.1 Subspace tracking DF tests: antenna array coordinates [metres] 70
- 4.2 Subspace tracking DF tests: broadcast locations 76
- 4.3 Subspace tracking DF tests: bearing errors 101



Chapter 1

Introduction and Objectives

In 1923, Robert Watson Watt constructed the first equipment to perform direction finding on radio waves [15]. His original brief, given by the U.K. Meteorological Office, was to locate the radiation generated by lightning, and thus provide the ability to warn pilots of approaching thunderstorms. Since that time, the ability to locate the sources of radio emissions has seen applications in fields as diverse as aviation, electronic warfare, astronomy and telecommunications.

When two or more antenna elements receive the signal emitted by a single source, the differences in phase, amplitude and time of reception may be used to estimate the origin of the source. Early direction finding methods utilised direct calculation of trigonometric and amplitude differences to estimate the directional-of-arrival. More recent methods exploit the statistical relationships between antenna elements to build up a spatial picture of the signals received by the array. The array correlation matrix is a central theme of these statistical algorithms.

Eigen decomposition allows the correlation matrix to be decomposed into basis vectors that “explain” the correlations between elements. The eigenvectors may be separated into orthogonal signal and noise subspaces. Given known parameters of signals received by the array (such as frequency), the task of subspace direction finding methods is to estimate the remaining unknown parameters - in particular, the origin of the source. Using mathematical models of signal propagation, it is possible to determine the unknown signal parameters from either the signal or noise subspace. The MUSIC algorithm is a widely used spectral method that can determine Direction-Of-Arrival from the noise subspace.

Traditional methods to find the eigen decomposition of a correlation matrix (such as the Singular Value Decomposition or the QR algorithm) are, unfortunately, extremely computationally expensive. The computational load increases quadratically with the number of antenna elements. As signals appear and disappear, and as their parameters change, there is also a need to update (track) the decomposition continuously. There has thus been an enormous amount of interest in algorithms for subspace tracking that trade some aspects of accuracy for increased computational performance.

This thesis examines three reduced complexity algorithms for subspace tracking that were selected as promising candidates, following a literature study. One (PAST, outlined by Bin Yang in 1995 [24]) has become a well regarded reference method. Two other methods are more recent: OPERA (Craig MacInnes, 1998) [11] and FDPM (Doukopoulos and Moustakides, 2005) [2]. All three algorithms are characterised in terms of their computational efficiencies and abilities to resolve the bearings of signals when coupled to the MUSIC algorithm.

A PC-based program was developed during the course of this thesis for the simulation of spatially sampled signals. The output of this simulator is used as input to the three subspace tracking algorithms. Each algorithm has also been implemented as part of a PC program developed for this thesis, and timing information derived from this program is used to characterise the relative computational complexity of each algorithm.

Chapter 2 starts by formulating a mathematical model of spatial reception by an antenna array. It proceeds to describe how subspace methods may be used for Direction-Of-Arrival estimation. Finally, the three selected algorithms for subspace tracking are presented.

Chapter 3 begins by outlining the construction of the PC-based program for the simulation of spatial signals. The PAST, FDPM and OPERA methods for subspace tracking are then compared in terms of performance and efficiency. The theoretical complexity of each tracking algorithm is also examined.

Chapter 4 reports the results of an experiment using real-world signals, when applying the selected subspace tracking methods.

Finally, conclusions regarding the relative merits of the subspace tracking algorithms are drawn in chapter 5. It will be shown that all three methods may be used for direction-finding, although relative performance depends on factors such as Signal-to-Noise-Ratio and angular separation. A tradeoff between robustness and complexity will be demonstrated.

Chapter 2

Subspace methods for direction finding

Subspace methods for direction finding depend on statistical relationships between the elements of an antenna array. It is thus necessary to develop models that approximate the response of individual elements to a signal source, in order that these relationships may be determined.

2.1 Spatial sampling by an antenna array

By Fourier's theorem, we know that all real-world signals may be approximated by a linear combination of sinusoidal signals. We begin by determining the response of an antenna array to a sinusoidal generator.

A sinusoidal source of radio energy may be modelled as a complex exponential variation of the electric field as a function of time, $y(t)$:

$$y(t) = ae^{j\omega t} \quad [\text{V}\cdot\text{m}^{-1}] \quad (2.1)$$

where a is the (complex) amplitude of the oscillation and ω represents its angular frequency in $\text{rad}\cdot\text{s}^{-1}$.

The electromagnetic field excites an antenna element m at a distance d_m in the plane of polarisation, inducing a voltage $x_m(t)$:

$$x_m(t) = \frac{K_m}{d_m} e^{j\omega(t - \frac{d_m}{c})} = \frac{K_m}{d_m} e^{-j2\pi f \frac{d_m}{c}} e^{j2\pi f t} \quad [\text{V}] \quad (2.2)$$

where K_m is a constant depending on the directivity and power of the transmitter and other constant parameters of the receiving antenna element, c is the speed of propagation in the medium ($c \approx 299792458 \text{ m}\cdot\text{s}^{-1}$ in air) and $\omega = 2\pi f$. The amplitude of the induced voltage varies with the inverse of distance ($\frac{1}{d_m}$).

If the transmitting source is modulated by a complex modulation function, $m(t)$ equation 2.1 becomes:

$$y(t) = m(t)e^{j\omega t} \quad [\text{V}\cdot\text{m}^{-1}] \quad (2.3)$$

and the received signal may be represented as follows:

$$x_m(t) = \frac{K_m}{d_m} m\left(t - \frac{d_m}{c}\right) e^{-j2\pi f \frac{d_m}{c}} e^{j2\pi f t} \quad [\text{V}] \quad (2.4)$$

The modulation coherence between any two elements in the array may be defined as the measure (magnitude) of the cross correlation between them. If the modulation function is constant, i.e. $m(t) = a$, the propagating wave is purely sinusoidal and the field is completely coherent across all elements. If the modulation function is rapidly varying, the propagating signal occupies a bandwidth B . In this case, the field is coherent between two elements only if they are separated by a sufficiently small distance, Δ_d , where

$$\Delta_d \ll \frac{c}{B} \quad [\text{m}] \quad (2.5)$$

This constraint is termed the **narrowband assumption**. Narrowband array signals may typically be processed using simple phase shift algorithms (or, in general, multiplication by single complex coefficients). As a rule of thumb, a signal may be considered narrowband with respect to an antenna array if the ratio of Δ_d to $\frac{c}{B}$ is about 1/100 or less. The value of this ratio is tabulated for several types of signals and inter-element spacings:

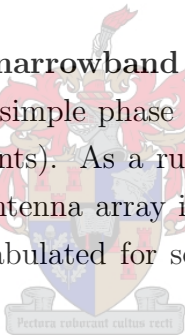


Table 2.1: Ratio of Δ_d to $\frac{c}{B}$ for several typical signals and inter-element spacings

Signal type	Bandwidth	Array length	$\Delta_d \frac{B}{c}$
HF speech	3.2 kHz	50 m	5.3×10^{-4}
Commercial FM	75 kHz	1 m	2.5×10^{-4}
PAL colour television	6 MHz	50 cm	0.01
Wi-Fi 802.11 g	22 MHz	50 cm	0.037
Radar	10 MHz	5 m	0.16

If the signal occupies a bandwidth too wide to be considered narrowband, simple phase shift processing may be inadequate for the application. In this case, pre-steering delays or other time-domain processing (such as FIR filtering) may be necessary. Alternately, the signal may be divided into frequency sub-bands that are processed as independent narrowband signals before subsequent recombination. Methods to decompose into sub-bands include the overlap-add (or overlap-save) FFT algorithm and quadrature mirror filtering.

If the bandwidth of the modulated signal is sufficiently narrow, the modulation term $m(t - \frac{d_m}{c})$ in equation 2.4 may be replaced by a simplification:

$$x_m(t) = \frac{K_m}{d_m} m(t - T_{avg}) e^{-j2\pi f \frac{d_m}{c}} e^{j2\pi ft} \quad [\text{V}] \quad (2.6)$$

where T_{avg} is the average time delay between the signal source and the receiving antenna array.

In most cases, the signal source is located far from the receiving antenna array and the relative differences between the spherical spreading coefficients ($\frac{1}{d_m}$) are negligible. This is called the **far field** assumption. In addition, the phase differences due to spherical spreading (termed **wavefront distortion**) may be ignored at large distances and the wavefront may be considered planar. The maximum value of the wavefront distortion in degrees, Θ_{max} , may be calculated as follows:

$$\Theta_{max} = \frac{360f(r - \sqrt{r^2 - d_{max}^2})}{c} \quad [^\circ] \quad (2.7)$$

where r is the distance in metres between the signal source and antenna array and d_{max} is the maximum distance between any two elements.

For the purposes of additional simplification, reflections (multipath propagation) and mutual coupling between antenna elements may be ignored. By assuming an isotropic, omnidirectional response pattern for each antenna element, the coefficients K_m may be discarded. The assumption of a planar wavefront also allows the phase response to be reformulated in terms of azimuth ϕ and elevation θ , when that is convenient:

$$x_m(t) = e^{-j2\pi f \frac{d_m}{c}} e^{j2\pi ft} = e^{-j2\pi f \frac{(x_m \cos \phi + y_m \sin \phi) \cos \theta + z_m \sin \theta}{c}} e^{j2\pi ft} \quad [\text{V}] \quad (2.8)$$

where the vector $\begin{bmatrix} x_m & y_m & z_m \end{bmatrix}$ represents the element's coordinates. Coordinates are usually referenced relative to the phase centre of the array or the location of the first element.

Notice that the term $e^{-j2\pi f \frac{d_m}{c}} = e^{-j2\pi f \frac{(x \cos \phi + y \sin \phi) \cos \theta + z \sin \theta}{c}}$ is independent of t and indicates a frequency dependent phase shift at each element. It may also be understood as a delay of $\frac{d_m}{c}$ seconds in the time domain, since Fourier Analysis tells us that $e^{j\omega \frac{d_m}{c}} \Leftrightarrow \delta(t - \frac{d_m}{c})$.

Generally, the M signals from the array elements are sampled by M ADC's (possibly following filtering and downconversion to reduce computation and data rates). The sampled data may be represent by an M by N matrix \mathbf{X} , where N is the number of sampling instants. Each column of \mathbf{X} represents a **snapshot** $\mathbf{x}(n)$ of the state of the array at some time instant $t = nT$, where T is the sampling period of the system. Each row consists of a series of samples corresponding to a particular antenna element.

The sample matrix \mathbf{X} obtained when an antenna array receives a far-field source may be rewritten using more compact vector notation:

$$\mathbf{X} = \begin{bmatrix} x_1(n) \\ x_2(n) \\ \vdots \\ x_M(n) \end{bmatrix} = \begin{bmatrix} e^{-j2\pi f \frac{(x_1 \cos \phi + y_1 \sin \phi) \cos \theta + z_1 \sin \theta}{c}} \\ e^{-j2\pi f \frac{(x_2 \cos \phi + y_2 \sin \phi) \cos \theta + z_2 \sin \theta}{c}} \\ \vdots \\ e^{-j2\pi f \frac{(x_M \cos \phi + y_M \sin \phi) \cos \theta + z_M \sin \theta}{c}} \end{bmatrix} e^{j2\pi f n T}$$

$$= \begin{bmatrix} s_1(\phi, \theta, f) \\ s_2(\phi, \theta, f) \\ \vdots \\ s_M(\phi, \theta, f) \end{bmatrix} \mathbf{y}(n) \quad (2.9)$$

$$= \mathbf{s}(\phi, \theta, f) \mathbf{y}(n) \quad (2.10)$$

where the antenna array consists of M elements. $\mathbf{x}(n)$ and $\mathbf{y}(n)$ are the discrete-time (sampled) representations of $x(t)$ and $y(t)$ respectively. The column vector $\mathbf{s}(\phi, \theta, f)$ has received several names in the literature, including the (array) response vector, space vector, propagation vector and transfer vector.

A beamformer takes the inputs from an antenna array and produces a single output that has maximum sensitivity at a given look (steering) direction, $\begin{bmatrix} \phi & \theta \end{bmatrix}$. The simplest kind of beamformer may be constructed by weighting the signal from each element by the conjugate of the corresponding element of the response vector before summation:

$$\mathbf{b} = \mathbf{s}(\phi, \theta, f)^H \mathbf{X} = \mathbf{w}^H \mathbf{X} \quad (2.11)$$

\mathbf{w} is termed the **weight vector** (a column vector with M entries). \mathbf{b} represents the sequence of beamformer output samples. Intuitively, this may be thought of as a kind of spatial matched filter: The conjugate multiplications equalise the phases across all antenna elements and maximise the amplitudes of the elements that are strongly illuminated, to give a focused summation.

2.2 The array covariance matrix

Covariance, a measure of the dependence of two variables, is a basic concept of statistics. The covariance between any two variables is defined in terms of the expectation of their product, with the sample means removed:

$$\text{cov}(x_i, x_j) = E\{(x_i - \mu_i)(x_j - \mu_j)\} \quad (2.12)$$

where μ_i and μ_j indicate the means of the random variables x_i and x_j respectively. E indicates the expectation operator.

When an array of sensor elements is sampled, the covariance between the sampled signals from each pair of elements may be completely expressed by the **covariance matrix**.

Given a matrix \mathbf{X} consisting of N array snapshots from M elements, the array covariance matrix \mathbf{R} is an M by M matrix, defined as follows:

$$\mathbf{R} = E\{\mathbf{X}\mathbf{X}^H\} \quad (2.13)$$

$$= \begin{bmatrix} \text{var}(x_1) & \text{cov}(x_1, x_2) & \text{cov}(x_1, x_3) & \cdots & \text{cov}(x_1, x_m) \\ \text{cov}(x_2, x_1) & \text{var}(x_2) & \text{cov}(x_2, x_3) & \cdots & \text{cov}(x_2, x_m) \\ \text{cov}(x_3, x_1) & \text{cov}(x_3, x_2) & \text{var}(x_3) & \cdots & \text{cov}(x_3, x_m) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \text{cov}(x_m, x_1) & \text{cov}(x_m, x_2) & \text{cov}(x_m, x_3) & \cdots & \text{var}(x_m) \end{bmatrix} \quad (2.14)$$

The variances of each elemental signal appear along the main diagonal, since $\text{cov}(x, x) = E\{(x - \mu)^2\} = \text{var}(x)$. Because $\text{cov}(x_i, x_j) = \text{cov}(x_j, x_i)$, the covariance matrix is symmetric. It can also be shown that the covariance matrix is positive definite.

If each element of \mathbf{R} is normalised by the product of the standard deviations of the pairwise elements, i.e.

$$\mathbf{R}_{i,j} = \frac{\text{cov}(x_i, x_j)}{\sigma_i \sigma_j} \quad (2.15)$$

the matrix is called the **correlation matrix**.

2.2.1 Estimation of the array covariance matrix

In practice, \mathbf{R} may be replaced by its estimate, $\hat{\mathbf{R}}$:

$$\hat{\mathbf{R}} = \frac{1}{N} \mathbf{X} \mathbf{X}^H = \frac{1}{N} \sum_{n=1}^N \mathbf{x}(n) \mathbf{x}(n)^H \approx \mathbf{R} \quad (2.16)$$

$\hat{\mathbf{R}}$ is an unbiased estimator of \mathbf{R} (i.e. it tends to its true value) if N is large and the $\mathbf{x}(n)$ are stationary zero-mean Gaussian processes. Positive definite matrix random variables that meet these criteria are said to follow the **Wishart distribution**.

The estimate of the covariance matrix may be updated recursively by a rank-one update as new samples arrive, to minimise computation:

$$\hat{\mathbf{R}}(n+1) = \frac{1}{n+1} \left(n \hat{\mathbf{R}}(n) + \mathbf{x}(n+1) \mathbf{x}(n+1)^H \right) \quad (2.17)$$

In practice, the signals received by the antenna array change characteristics slowly over time (although it is assumed that they are pseudostationary over short time intervals). It is thus necessary to update the estimate of the covariance matrix periodically so that older samples are deweighted (also called **downdating**). This may be achieved by reformulating equation 2.17 as a first-order moving average (similar in structure to a lowpass IIR filter):

$$\hat{\mathbf{R}}(n+1) = \beta \hat{\mathbf{R}}(n) + (1 - \beta) \mathbf{x}(n+1) \mathbf{x}(n+1)^H \quad (2.18)$$

where β is a **forgetting factor** between 0 and 1. β is typically chosen to be close to 1 if the signals received by the array are varying slowly, relative to the sample period.

Some applications require calculation of the covariance matrix inverse (for example, Sample Matrix Inversion beamforming). By the Sherman-Morrison formula, equation 2.18 may be rewritten to provide a downdated estimate of the inverse of the covariance matrix:

$$\hat{\mathbf{R}}(n+1)^{-1} = \beta \hat{\mathbf{R}}(n)^{-1} - \frac{(1 - \beta) \hat{\mathbf{R}}(n)^{-1} \mathbf{x}(n+1) \mathbf{x}(n+1)^H \hat{\mathbf{R}}(n)^{-1}}{1 + (1 - \beta) \mathbf{x}(n+1)^H \hat{\mathbf{R}}(n)^{-1} \mathbf{x}(n+1)} \quad (2.19)$$

$\hat{\mathbf{R}}$ is initialised as:

$$\hat{\mathbf{R}}(0) = \frac{1}{\varepsilon_0} \mathbf{I} \quad (\varepsilon_0 > 0) \quad (2.20)$$

2.2.2 The form of a typical array covariance matrix

Discrete point sources

Suppose that a signal $y(t)$ from a single source impinges on an antenna array, in the absence of reflections or interfering noise. The sampled array output is given by equation 2.9 as

$$\mathbf{X} = \mathbf{s}(\bar{\psi})\mathbf{y}(n)$$

where $\bar{\psi}$ is a vector representation of the parameters ϕ , θ and f .

The covariance matrix may then be calculated by equation 2.13,:

$$\mathbf{R}_S = E\{\mathbf{s}(\bar{\psi})\mathbf{y}(n) (\mathbf{s}(\bar{\psi})\mathbf{y}(n))^H\} \quad (2.21)$$

$$= E\{\mathbf{s}(\bar{\psi})\mathbf{y}(n)\mathbf{y}(n)^H\mathbf{s}(\bar{\psi})^H\} \quad (2.22)$$

$$= E\{\mathbf{s}(\bar{\psi})\rho^2\mathbf{s}(\bar{\psi})^H\} \quad (2.23)$$

$$= E\{\rho^2\}E\{\mathbf{s}(\bar{\psi})\mathbf{s}(\bar{\psi})^H\} \quad (2.24)$$

$$= \text{var}(\mathbf{y}(n))\mathbf{s}(\bar{\psi})\mathbf{s}(\bar{\psi})^H \quad (2.25)$$

The term $\mathbf{y}(n)\mathbf{y}(n)^H$ represents the dot product of the row vector $\mathbf{y}(n)$ with itself and evaluates to a single scalar ρ^2 , equal to the squared magnitude of $\mathbf{y}(n)$. Assuming that $\mathbf{y}(n)$ is a zero mean process, $E\{\rho^2\}$ represents the variance (or power) of $\mathbf{y}(n)$. Note that $E\{\mathbf{s}(\bar{\psi})\mathbf{s}^H\} = \mathbf{s}(\bar{\psi})\mathbf{s}(\bar{\psi})^H$, because the expected value of the matrix $\mathbf{s}(\bar{\psi})\mathbf{s}(\bar{\psi})^H$ is defined as the matrix of expected values, and $\mathbf{s}(\bar{\psi})\mathbf{s}(\bar{\psi})^H$ is constant when the signal is stationary. \mathbf{R}_S is Toeplitz and positive definite.

Uncorrelated Gaussian white noise

Real world signals from an antenna array are also subject to incoherent random noise, produced, for example, in the preamplifier circuitry of receiver channels. This Gaussian white noise is totally uncorrelated between channels and produces a covariance matrix of the form:

$$\mathbf{R}_U = \sigma^2\mathbf{I} \quad (2.26)$$

$$= \begin{bmatrix} \sigma^2 & 0 & 0 & \cdots & 0 \\ 0 & \sigma^2 & 0 & \cdots & 0 \\ 0 & 0 & \sigma^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \vdots & \sigma^2 \end{bmatrix} \quad (2.27)$$

where σ^2 is the power (variance) of the noise and \mathbf{I} is the identity matrix. The covariance matrix indicates quite clearly that the signal from each element is correlated only with itself (where the correlation is the noise power on the main diagonal).

If K **incoherent** signals impinge on the array, and Gaussian white noise is present, the total covariance matrix takes the form:

$$\mathbf{R} = \sigma^2 \mathbf{I} + \rho_1^2 \mathbf{s}(\overline{\psi_1}) \mathbf{s}(\overline{\psi_1})^H + \rho_2^2 \mathbf{s}(\overline{\psi_2}) \mathbf{s}(\overline{\psi_2})^H + \dots + \rho_K^2 \mathbf{s}(\overline{\psi_K}) \mathbf{s}(\overline{\psi_K})^H \quad (2.28)$$

$$= \sigma^2 \mathbf{I} + \sum_{k=1}^K \rho_k^2 \mathbf{s}(\overline{\psi_k}) \mathbf{s}(\overline{\psi_k})^H \quad (2.29)$$

The restriction that the signals are incoherent implies that their response vectors $\mathbf{s}(\overline{\psi_k})$ are linearly independent. This will be the case as long as the directions of arrival differ and the array geometry does not have direction ambiguities at the signal frequencies.

Spherically isotropic Gaussian white noise

There are many sources of radio frequency energy that contribute to the signal received by an antenna array. These sources include the sun, lightning, charged particles in the ionosphere, airborne radar, power generation and distribution facilities, cellphones, satellites, Radio and TV transmissions, Wi-Fi networks and interfering noise produced by electrical devices.

When received by an antenna array, these noise sources produce individual contributions, as in the discrete case. There are obviously too many sources to allow individual analysis. However, it is possible to develop combined (integral) models that account for these noise sources.

One model is that of **spherically isotropic** noise (isotropic means “the same in all directions”). This model assumes that a large number of uncorrelated noise sources are distributed evenly on the surface of a sphere. It also assumes that the antenna array is at the centre of the sphere, and that radius of the sphere is large compared to the spacing between antenna elements. This type of noise is sometimes referred to as **sky noise**.

If two antenna elements, i and j , are separated by a distance d , their covariance in the presence of spherically isotropic Gaussian white noise may be calculated as follows:

$$cov_{ij}(k, d) = \sigma^2 \frac{\sin(kd)}{kd} \quad (2.30)$$

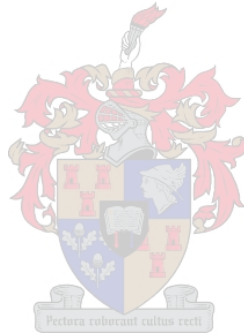
where $k = \frac{2\pi f}{c}$ is the wavenumber (the model is thus a narrowband approximation) and σ^2 is the variance [6, page 57]. The complete covariance matrix may be constructed, given this element-wise description.

Cylindrically isotropic Gaussian white noise

The majority of interfering noise sources arise from man-made sources, particularly in urban or industrial environments. These noise sources generally originate from locations on the ground, in the same plane as the antenna array. A suitable model is that of a ring, with elemental noise sources distributed evenly around the ring, and the antenna array at the centre. The radius is assumed to be large compared to the inter-element spacing. The covariance between any pair of elements under these conditions may be approximated as follows:

$$\text{cov}_{ij}(k, d) = \sigma^2 J_0(kd) \quad (2.31)$$

where J_0 is the zero-order Bessel function of the first kind, σ^2 is the variance and k is the wavenumber [6, page 57].



2.3 Eigen decomposition of the array covariance matrix

The array covariance matrix \mathbf{R} in equation 2.28 may be decomposed into M eigenvectors, $\mathbf{v}_1 \dots \mathbf{v}_M$, and their corresponding eigenvalues, $\lambda_1 \dots \lambda_M$, such that

$$\hat{\mathbf{R}}\mathbf{v}_k = \lambda_k \mathbf{v}_k \quad (\text{the definition of an eigenvector}) \quad (2.32)$$

and

$$\hat{\mathbf{R}} = \sum_{k=1}^M \lambda_k \mathbf{v}_k \mathbf{v}_k^H \quad (\text{because } \hat{\mathbf{R}} \text{ is Hermitian}) \quad (2.33)$$

If the signal environment consists of one signal only and uncorrelated noise, the eigenvector with the largest eigenvalue is equal to a scaled version of the response vector:

$$\mathbf{v}_1 = \frac{1}{\sqrt{M}} \mathbf{s}(\bar{\psi}) \quad (2.34)$$

and the corresponding eigenvalue is

$$\lambda_1 = \sigma^2 + M\rho_1^2 \quad (2.35)$$

where σ^2 is the power of the uncorrelated noise and ρ_1^2 is the power of the signal. The remaining $M - 1$ eigenvectors form some orthonormal set with eigenvalue σ^2 , and each one is orthogonal to the signal eigenvectors.

In practice, usually more than one signal impinges on the antenna array. If K **incoherent** signals are present, the K eigenvectors with the largest eigenvalues do *not* correspond directly to scaled response vectors. (Intuitively, this is obvious because eigenvectors are defined to be orthogonal, whereas response vectors are, in general, *not* orthogonal). However, these K eigenvectors do form a **basis** for the set of response vectors. In other words, the matrix with the signal response vectors as columns is equal to the matrix of “signal” eigenvectors, multiplied by some unknown transformation matrix \mathbf{T} :

$$\mathbf{S}(\bar{\psi}) = \begin{bmatrix} \mathbf{s}_1(\bar{\psi}) & \mathbf{s}_2(\bar{\psi}) & \dots & \mathbf{s}_K(\bar{\psi}) \end{bmatrix} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_K \end{bmatrix} \mathbf{T} = \mathbf{V}_S \mathbf{T} \quad (2.36)$$

Notice that the “signal” eigenvectors are entirely independent of the uncorrelated noise (although the “signal” eigenvalues have a contribution σ^2 from the noise). The $M - K$ “noise” eigenvalues all have the value σ^2 , if the noise is spectrally and spatially white.

The set of eigenvectors $[\mathbf{v}_1 \dots \mathbf{v}_K]$ with the largest corresponding eigenvalues is referred to as the **signal subspace**, \mathbf{V}_S , while the remaining $M - K$ eigenvectors are termed the **noise subspace**, \mathbf{V}_N :

$$\mathbf{V} = [\mathbf{v}_1 \mathbf{v}_2 \dots \mathbf{v}_K \quad \mathbf{v}_{K+1} \mathbf{v}_{K+2} \dots \mathbf{v}_M] \quad (2.37)$$

$$= [\mathbf{V}_S \quad \mathbf{V}_N] \quad (2.38)$$

The complete eigen decomposition of $\hat{\mathbf{R}}$ may be shown more pictorially in terms of the signal and noise subspaces as follows:

$$\hat{\mathbf{R}} = [\mathbf{V}_S \quad \mathbf{V}_N] \begin{bmatrix} \mathbf{\Lambda}_S & \mathbf{0} \\ \mathbf{0} & \mathbf{\Lambda}_N \end{bmatrix} [\mathbf{V}_S \quad \mathbf{V}_N]^H \quad (2.39)$$

$$= [\mathbf{V}_S \quad \mathbf{V}_N] \begin{bmatrix} \begin{bmatrix} \rho_1 & 0 & \cdots & 0 \\ 0 & \rho_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \rho_K \end{bmatrix} + \sigma^2 \mathbf{I}_K & \mathbf{0} \\ \mathbf{0} & \sigma^2 \mathbf{I}_{M-K} \end{bmatrix} [\mathbf{V}_S \quad \mathbf{V}_N]^H \quad (2.40)$$

If more than M uncorrelated signals are present, there is no useful relationship between the signal subspace and the set of response vectors. We will thus assume that the number of antenna elements exceeds the number of (strong) signals.

If the signals received by the array are correlated (coherent), the above analysis fails. In the most general case, equation 2.28 must be modified so that the array covariance matrix is decomposed into a linear combination of the coherent response vectors:

$$\mathbf{R} = \sigma^2 \mathbf{I} + \sum_{k=1}^K \rho_k^2 (\epsilon_1 \mathbf{s}(\psi_1) + \epsilon_2 \mathbf{s}(\psi_2) \dots \epsilon_K \mathbf{s}(\psi_K)) (\epsilon_1 \mathbf{s}(\psi_1) + \epsilon_2 \mathbf{s}(\psi_2) \dots \epsilon_K \mathbf{s}(\psi_K))^H \quad (2.41)$$

where the ϵ_k are the weights of the combination. When received signals are coherent, the covariance matrix tends to become rank-deficient and singular. The eigen decomposition may contain fewer eigenpairs than there are signals. It is extremely difficult to determine the unique response vectors, given an eigenpair arising from more than one correlated signal. Some possible solution to this problem are:

1. Pre-whiten the data received by the antenna array. For example, if the noise correlation matrix is \mathbf{R}_N , the output of the antenna elements are multiplied by $\mathbf{R}_N^{-1/2}$ (where $\mathbf{R}_N^{-1/2}$ is the Hermitian square-root factor of \mathbf{R}_N^{-1}).
2. Add synthetic noise to the diagonal of the correlation matrix, or add a fraction of the identity matrix:

$$\hat{\mathbf{R}} = \alpha \mathbf{I} + \frac{1}{N} \sum_{n=1}^N \mathbf{x}(n) \mathbf{x}(n)^H \quad (\alpha \in \mathbb{R}) \quad (2.42)$$

3. Group antenna elements into subarrays and perform **spatial smoothing** [23].

Estimation of the number of sources

The eigenvalues associated with the noise subspace all have the value σ^2 , if the noise is spectrally and spatially white (uncorrelated). The remaining “signal” eigenvalues are all somewhat larger, proportional to the powers of the signals received by the antenna array. It is thus possible to estimate the number of signals present by subdividing the eigenvalues into two groups, based on their magnitudes.

Two widely used methods for subdividing the eigenvalues are the Akaike Information Criterion (AIC) [1] and the Minimum Description Length (MDL) criterion [17]. Both of these methods have been used in the context of estimating the size of smallest model that accounts for a given set of data. When dealing with array signals, the “model” refers to the signal subspace, i.e. the smallest set of eigenvectors that explains the correlations between antenna elements.

These criteria arise out of the assumption that the eigenvalues of the noise subspace will be roughly equal. When a series of values are approximately equal, their arithmetic and geometric means are also nearly equal. If the values tend to be unequal, the arithmetic mean will *always* exceed the geometric mean. It is thus possible to define the best estimate of the noise subspace as the set of eigenvalues with the smallest ratio of arithmetic mean to geometric mean. The actual definition relies on the smallest logarithm of the ratio of geometric mean to arithmetic mean, because of considerations arising from Information Theory.

If the estimated dimension of the signal subspace is N_S , the value of the AIC criterion is defined as follows:

$$AIC(N_S) = -L(M - N_S) \ln \left(\frac{\left(\prod_{k=N_S+1}^M \lambda_k \right)^{\frac{1}{M-N_S}}}{\frac{1}{M - N_S} \sum_{k=N_S+1}^M \lambda_k} \right) + N_S (2M - N_S + 1) \quad (2.43)$$

where L is the number of samples used to create the covariance matrix, M is the number of its rows and the λ_k are it's eigenvalues.

The MDL criterion is defined similarly as:

$$MDL(N_S) = -L(M - N_S) \ln \left(\frac{\left(\prod_{k=N_S+1}^M \lambda_k \right)^{\frac{1}{M-N_S}}}{\frac{1}{M - N_S} \sum_{k=N_S+1}^M \lambda_k} \right) + \frac{1}{2} N_S (2M - N_S + 1) \ln L \quad (2.44)$$

where $\left(\prod_{k=N_S+1}^M \lambda_k\right)^{\frac{1}{M-N_S}}$ is the geometric mean and $\frac{1}{M-N_S} \sum_{k=N_S+1}^M \lambda_k$ is the arithmetic mean.

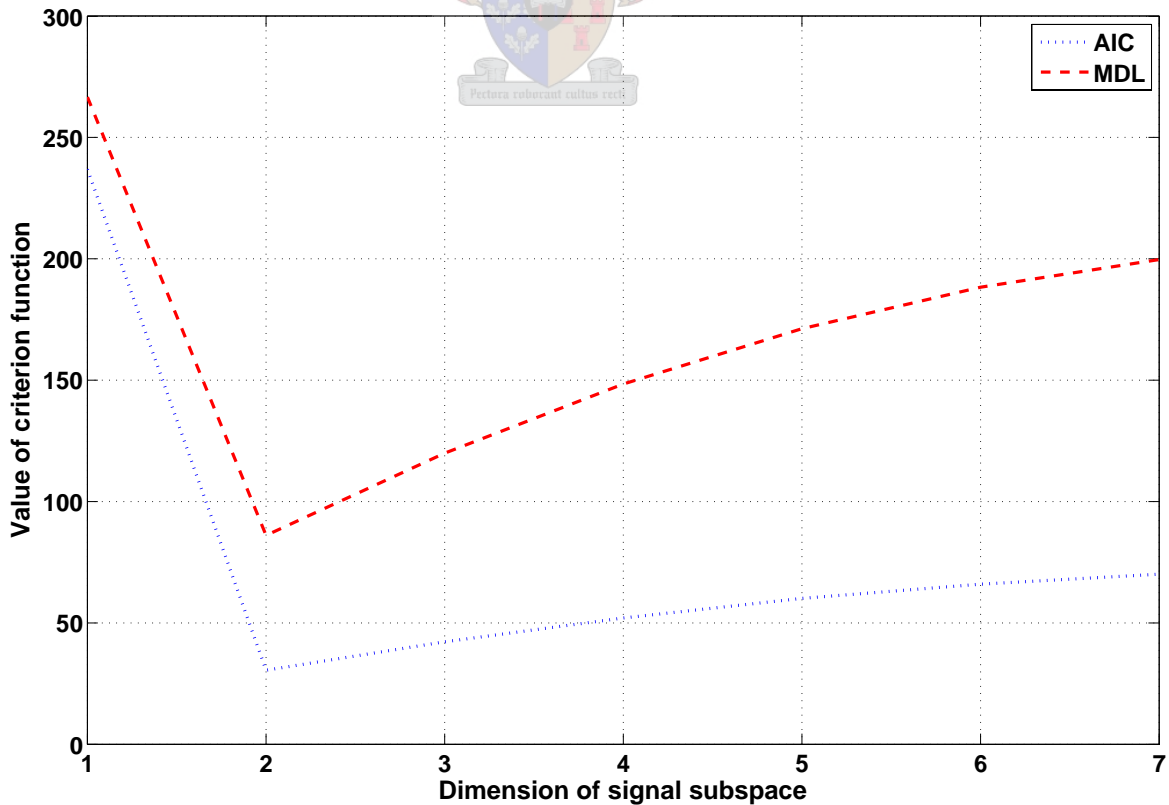
The smallest dimension of the signal subspace, i.e. the estimated number of signals, may be found by calculating the value of N_S that minimizes either the AIC or MDL criterion functions.

Figure 2.1 shows the AIC and MDL criteria for the following set of eigenvalues:

$$5.6248, 5.6248, 1.8429, 1.8120, 1.7948, 1.7699, 1.7580, 1.7479, 1.7260$$

This set was generated by simulating data from a 9-element antenna array, with two signals at azimuths of 150° and 200° respectively, each with amplitude -10 dBm. The noise floor of uncorrelated Gaussian white noise was set at -70 dBm. 1000 array snapshots were used. The minimum of each graph corresponds to the estimated minimum dimension of the signal subspace. Both the MDL and AIC criteria estimate this value as 2 (which correctly equals the number of signals present).

Figure 2.1: Plot of AIC and MDL criteria vs. signal subspace dimension



2.4 The MUSIC spectrum and Direction-Of-Arrival

The Fourier Transform response of a filter is a set of values that represent the amount by which sinusoidal input signals are scaled when they appear at the output. Sinusoidal basis functions are chosen because of their orthogonal properties (i.e. they have zero mutual dependence, calculated by a long term dot product). These sinusoidal basis functions may be thought of as **eigenvectors** of the filter, and the corresponding scalar values of the spectrum could be thought of as **eigenvalues**. Resonant signal frequencies are identified by peaks in the magnitude of the frequency response (referred to as the **power spectrum**).

The set of possible response vectors of an antenna array may similarly be used to generate a spectrum for a particular subspace of the covariance matrix. The response vectors are also orthogonal to each other (unless the array has ambiguities or the signals are correlated). This “spatial spectrum” is a function of Direction-Of-Arrival (DOA) and frequency, although the frequency is generally known.

The DOA power spectrum may be calculated from either the signal or noise subspaces. The direction finding problem may thus be framed in two ways:

1. finding “resonant” peaks in the power spectrum of the signal subspace, where the locations of the peaks correspond to the DOA
2. finding nulls in the power spectrum of the noise subspace, where the locations of the nulls correspond to the DOA

Nulls of the noise subspace power spectrum are found by searching its **inverse** for peaks. This works because the noise subspace is defined to be the complement of the signal subspace. Nullspace methods amount to identifying where signals *are* based on where we know they are *not*.

The power spectrum of the signal subspace may be calculated as follows:

$$P_{signal}(\bar{\psi}) = |\mathbf{V}_S^H \mathbf{s}(\bar{\psi})|^2 \quad (2.45)$$

and the power spectrum of the noise subspace may be calculated similarly:

$$P_{MUSIC}(\bar{\psi}) = \frac{1}{|\mathbf{s}(\bar{\psi}) \mathbf{V}_N|^2} \quad (2.46)$$

Equation 2.46 describes the well-known MUSIC (Multiple Signal Classification) spectrum. The MUSIC spectrum does not utilise the eigenvalues of the noise subspace, but assumes that they all have an equal value. This amounts to **whitening** of the noise subspace. (The alternative method, which makes use of the eigenvalues, is called the **eigenvector method**. It will not be considered here.)

Figure 2.2 shows the azimuth power spectrum for two simulated 10 MHz signals in the presence of Gaussian white noise. The spectrum is generated from the two columns of the signal subspace. The azimuths of the two signals are 75° and 225° respectively. The noise floor is simulated as -40 dBm and the amplitudes of the signals are -10 dBm each. The array used is a 9-element circular array with a radius of 32.5m.

Figure 2.2: *Direction-Of-Arrival power spectrum (signal subspace)*

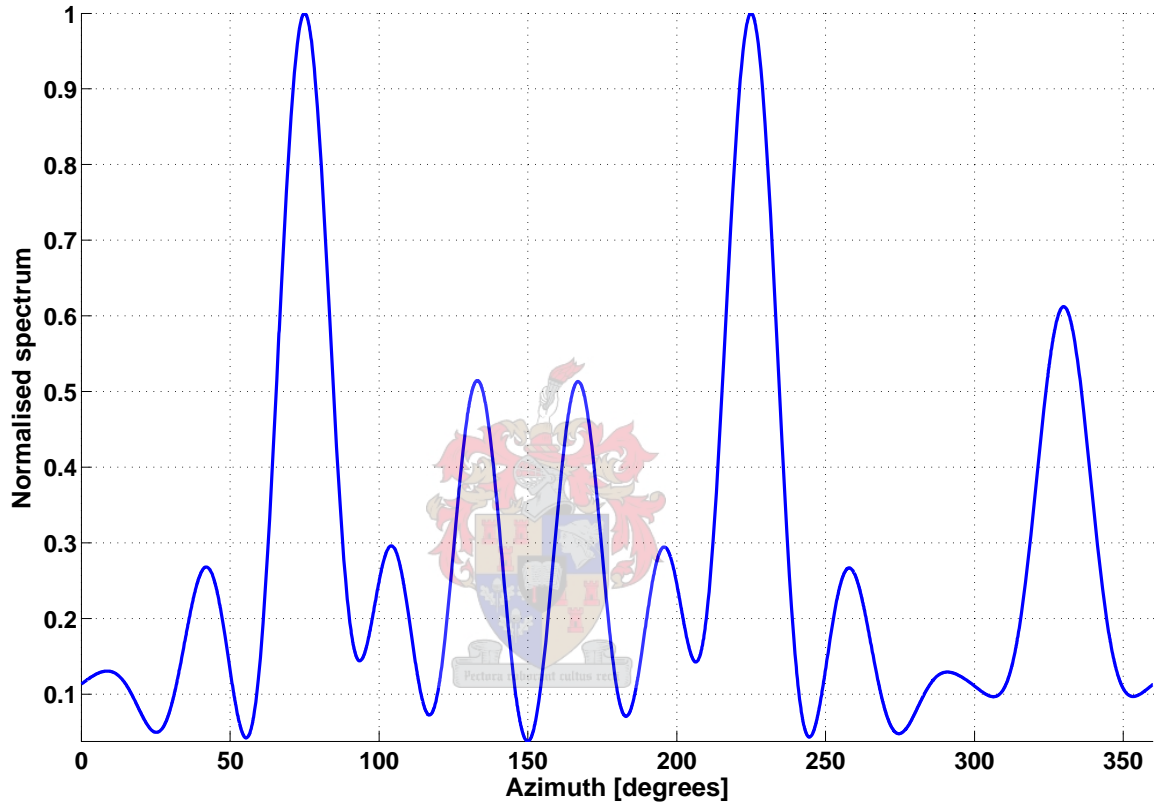
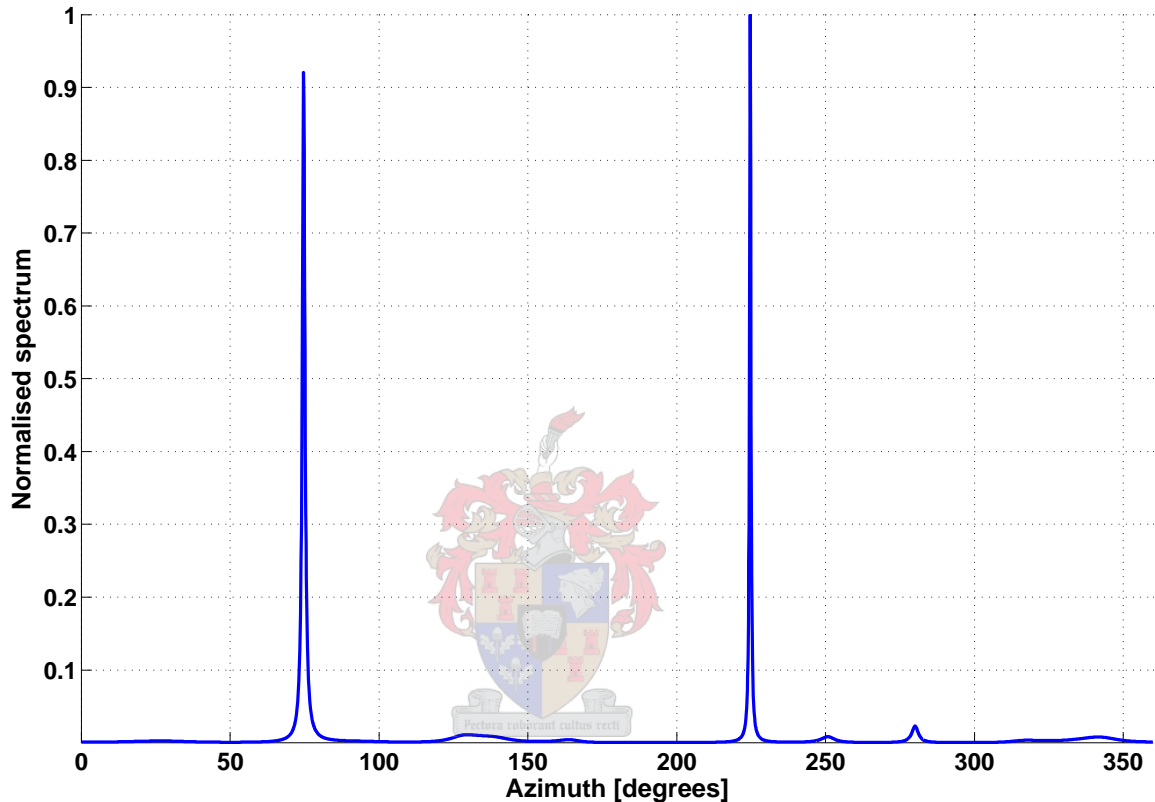


Figure 2.3 shows the MUSIC power spectrum for the same simulated parameters as figure 2.2. The noise subspace has 7 columns ($M = 9$, with 2 signals present). The amplitudes of the peaks do not correspond to the amplitudes of the signals - they merely indicate a “goodness-of-fit” for the particular DOA.

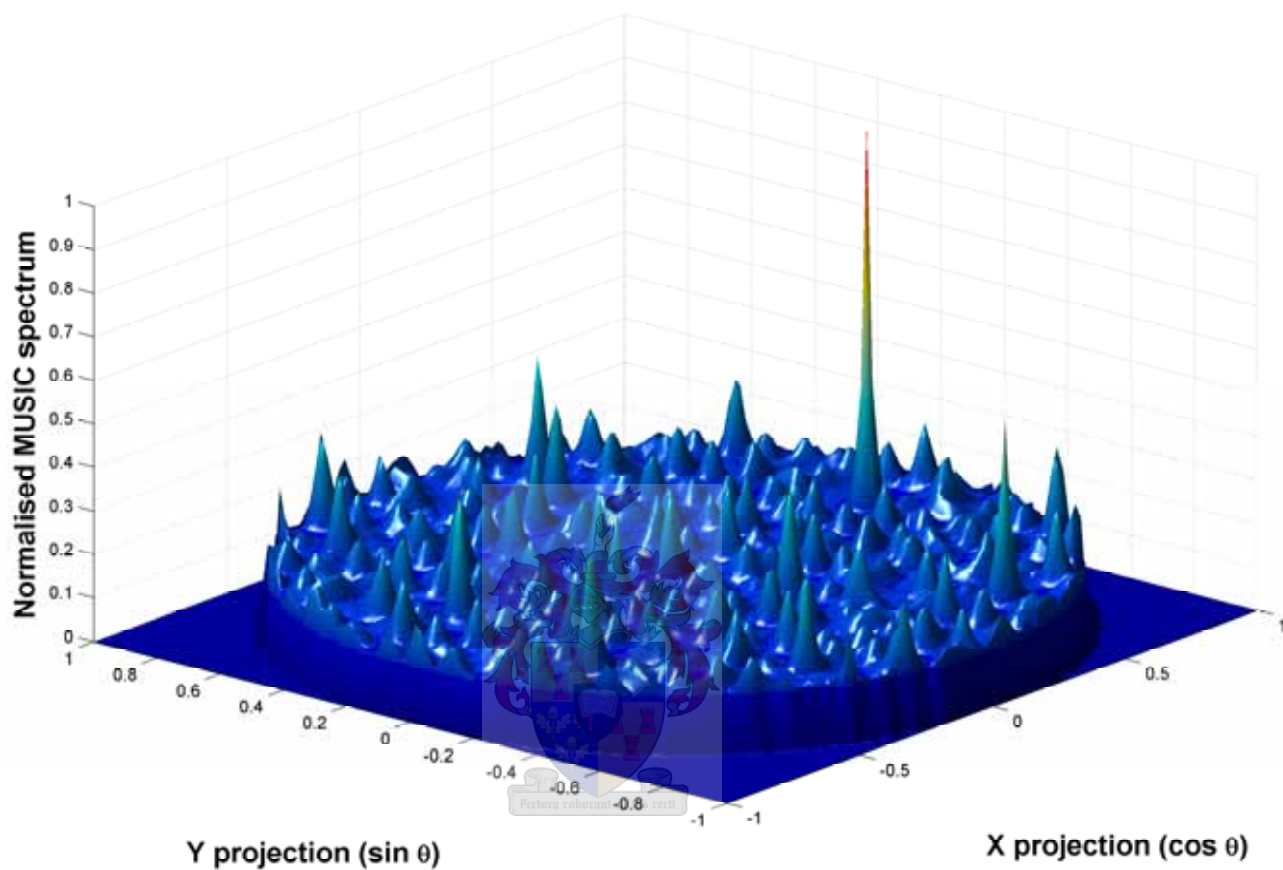
Figure 2.3: *MUSIC Direction-Of-Arrival power spectrum (noise subspace)*



Figures 2.2 and 2.3 demonstrate how much sharper the peaks of the noise spectrum are than the signal spectrum. In effect, nulls are being steered towards the signal arrival directions, and the (inverse) depths of the nulls appear as peaks in the spectrum. This sharpness of the peaks is referred to as **superresolution**. More precisely, superresolution refers to the ability to resolve signals spaced more closely than the Rayleigh limit (half a wavelength). The noise subspace is generally used in practice because of these properties.

When estimating the DOA of a signal, the MUSIC spectrum is usually a 3-dimensional surface, parameterised by both azimuth and elevation (see figure 2.4). Efficient MUSIC implementations may perform a coarse “grid search” for peaks and then progressively refine these peaks using search procedures such as the Broyden-Fletcher-Goldfarb-Shanno algorithm.

Figure 2.4: 3D MUSIC DOA power spectrum, a function of azimuth and elevation. 90° elevation is at the centre. 0° elevation is toward the extremes of the unit circle. Azimuth is 0° at $x = 0, y = 1$ and 90° at $x = 1, y = 0$.



2.5 Reduced complexity algorithms for subspace tracking

Calculation of the DOA using the signal or noise subspaces requires some method of decomposing the correlation matrix into eigenvalues and eigenvectors. Two immediately obvious solutions are the eigen decomposition of the correlation matrix and the Singular Value Decomposition of the data matrix.

The eigen decomposition of \mathbf{R} (where \mathbf{R} is square and symmetric) represents a factorisation:

$$\mathbf{R} = \mathbf{U}\mathbf{D}\mathbf{U}^H \quad (2.47)$$

where \mathbf{D} is a diagonal matrix with the eigenvalues of \mathbf{R} on the diagonal. \mathbf{U} contains the eigenvectors of \mathbf{R} as columns.

Given a (rectangular) data matrix \mathbf{X} , the Singular Value Decomposition (SVD) provides a method to obtain the factorisation:

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^H \quad (2.48)$$

where \mathbf{D} is a matrix with the **singular values** on the main diagonal and zeros elsewhere, \mathbf{U} (m by m) is an orthogonal matrix of “left” basis vectors and \mathbf{V} (n by n) is an orthogonal matrix of “right” basis vectors for \mathbf{R} . The columns of \mathbf{U} are the eigenvectors of $\mathbf{X}\mathbf{X}^H$, while the columns of \mathbf{V} are the eigenvectors of $\mathbf{X}^H\mathbf{X}$. Intuitively, the SVD shows that any matrix may be factorised into a rotation (\mathbf{U}), followed by a scaling (\mathbf{D}), followed by a rotation (\mathbf{V}^H). \mathbf{U} and \mathbf{V}^H are clearly rotation matrices, since their columns are mutually orthogonal and have unit length (by definition). The SVD can also be interpreted as a representation of the original data in a new coordinate system with a purely diagonal covariance matrix (\mathbf{D}). The new coordinate system has maximal variance along each axis, where each axis is represented by a singular vector. Notice that the SVD provides a method to obtain the subspace decomposition directly from the data matrix, without calculation of the covariance matrix (unlike the eigen decomposition).

In practice, the SVD is often computed through a series of QR iterations. The algorithm typically has a complexity of $\mathcal{O}(N^3)$. More efficient approximate methods are usually necessary in practice.

2.5.1 Projection Approximation Subspace Tracking

Suppose that we have an estimate of the dominant (signal) subspace, \mathbf{W} (where each column is an eigenvector). The **linear Principal Component Analysis criterion** measures to what extent this estimate is a “good” explanation of the correlation between observed data snapshots, $\mathbf{x}(n)$:

$$J(\mathbf{W}(n)) = E\{\|\mathbf{x}(n) - \mathbf{W}(n)\mathbf{W}(n)^H\mathbf{x}(n)\|^2\} \quad (2.49)$$

$\mathbf{W}(n)\mathbf{W}(n)^H\mathbf{x}(n)$ indicates a projection of $\mathbf{x}(n)$ into the subspace $\mathbf{W}(n)$. $J(\mathbf{W}(n))$ has a global minimum when $\mathbf{W}(n)$ is equal to a basis for the signal subspace, i.e. when $\mathbf{x}(n)$ and its projection into the signal subspace are most similar (measured by the Frobenius norm of their difference). In other words, the signal subspace may be estimated by minimising equation 2.49.

Note that $\mathbf{W}(n)$ is not equal to the signal subspace itself, but merely provides a possible **basis** for it. However, if $\mathbf{W}(n)$ is a single column vector, it does indeed become equal to the Principal Component (dominant eigenvector) under minimisation.

The expectation operator in equation 2.49 may be replaced by an exponentially weighted sum over n samples, to give the estimate:

$$\hat{J}(\mathbf{W}(n)) = \sum_{t=1}^n \beta^{n-t} \|\mathbf{x}(t) - \mathbf{W}(n)\mathbf{W}(n)^H\mathbf{x}(t)\|^2 \quad (2.50)$$

where β is the forgetting factor ($0 < \beta < 1$). The forgetting factor allows the subspace estimation to track nonstationary signals over time. The estimate is calculated for time n .

The term $\mathbf{W}^H(n)\mathbf{x}(t)$ may be replaced by an estimation $\mathbf{x}(t)' = \mathbf{W}^H(t-1)\mathbf{x}(t)$, using the previous value of $\mathbf{W}(n)^H$ in the iteration, giving:

$$\hat{J}'(\mathbf{W}(n)) = \sum_{t=1}^n \beta^{n-t} \|\mathbf{x}(t) - \mathbf{W}(n)\mathbf{x}'(t)\|^2 \quad (2.51)$$

Equation 2.51 resembles the cost function used to define a Recursive Least Squares (RLS) filter:

$$C(\mathbf{w}_n) = \sum_{t=1}^n \beta^{n-t} |e(t)|^2 \quad (2.52)$$

where $e(t)$ is the error signal. In this case, the error signal is the difference between $\mathbf{x}(t)$ (the “desired” signal) and its projection into the subspace, $\mathbf{W}(n)\mathbf{x}'(t)$.

After manipulation, the minimisation of equation 2.51 may be transformed into a series of recursive update equations that form the estimate of $\mathbf{W}(n)$ [24]:

for $n=1, 2, \dots$

$$\mathbf{x}'(n) = \mathbf{W}^H(n-1)\mathbf{x}(n) \quad (2.53a)$$

$$\mathbf{h}(n) = \mathbf{P}(n-1)\mathbf{x}'(n) \quad (2.53b)$$

$$\mathbf{g}(n) = \frac{\mathbf{h}(n)}{(\beta + \mathbf{x}'(n))^H \mathbf{h}(n)} \quad (2.53c)$$

$$\mathbf{P}(n) = \frac{1}{\beta} \text{tri}\{\mathbf{P}(n-1) - \mathbf{g}(n)\mathbf{h}^H(n)\} \quad (2.53d)$$

$$\mathbf{e}(n) = \mathbf{x}(n) - \mathbf{W}(n-1)\mathbf{x}'(n) \quad (2.53e)$$

$$\mathbf{W}(n) = \mathbf{W}(n-1) + \mathbf{e}(n)\mathbf{g}^H(n) \quad (2.53f)$$

end

The operator $\text{tri}\{\}$ indicates that only the upper triangular portion of the matrix need be calculated. The result for the upper triangular portion is then copied into the lower triangular portion. This forces the resulting matrix to be Hermitian.

Projection Approximation Subspace Tracking with Deflation (PASTd)

As mentioned above, the PAST algorithm provides a method to estimate only a **basis** for the dominant subspace. The exact eigenvectors (singular vectors) are not calculated, unless $\mathbf{W}(n)$ is a column vector - in which case only the dominant eigenvector (principal component) is estimated. However, the Karhunen-Loève expansion

$$\mathbf{x}(n) = \sum_{i=1}^N \mathbf{w}_i^H \mathbf{x}(n) \mathbf{w}_i \quad (2.54)$$

suggests a method to subtract out the dominant eigenvector \mathbf{w}_1 from the input data, once it is known:

$$\hat{\mathbf{x}}(n) = \mathbf{x}(n) - \mathbf{w}_1(n)\mathbf{w}_1(n)^H \mathbf{x}(n) \quad (2.55)$$

where $\hat{\mathbf{x}}(n)$ represents the n -th data snapshot, with the influence of the dominant eigenvector removed.

(The Karhunen-Loève expansion tells us that any $\mathbf{x}(n)$ may be expressed as a linear combination of the eigenvectors of the correlation matrix.)

By successively removing the influence of each dominant eigenvector, the complete set of signal eigenvectors spanning the subspace may be estimated from the data. This iterative process is called **deflation**, and the modified algorithm is called PASTd (Projection Approximation Subspace Tracking with Deflation).

After some re-arrangement, the complete PASTd algorithm may be summarised as follows:

for $n=1, 2, \dots, N$

$$\mathbf{x}_1(n) = \mathbf{x}(n) \quad (2.56a)$$

for $i=1, 2, \dots, K$

$$x'_i(n) = \mathbf{w}_i^H(n-1)\mathbf{x}_i(n) \quad (2.56b)$$

$$d_i(n) = \beta d_i(n-1) + |x'_i(n)|^2 \quad (2.56c)$$

$$\mathbf{e}_i(n) = \mathbf{x}_i(n) - \mathbf{w}_i(n-1)x'_i(n) \quad (2.56d)$$

$$\mathbf{w}_i(n) = \mathbf{w}_i(n-1) + \frac{\mathbf{e}_i(n)x_i'^*}{d_i(n)} \quad (2.56e)$$

end

end

where estimates are made of the K eigenvectors with the largest eigenvalues. Since $K \ll M$ in practice (it would be unusual to have nearly as many signals as antenna elements), this indicates an important optimisation compared to the eigendecomposition or SVD (which necessarily calculate all eigenvectors).

The eigenvector projection estimates \mathbf{w}_i are initialised to the columns of some non-zero orthogonal matrix, typically the first K columns of the identity matrix. The d_i are initialised to arbitrary non-zero constants (usually 1.0).

As the algorithm converges, the \mathbf{w}_i contain estimates of the R most dominant eigenvectors of the covariance matrix. The corresponding eigenvalues may be calculated by multiplying the d_i by $\frac{1-\beta}{\beta}$.

The “eigenvectors” calculated by the PAST and PASTd algorithms are generally not perfectly orthonormal, although they are good enough approximations to serve as input to the MUSIC algorithm.

If exact eigenvectors are needed, it is possible to modify either algorithm to include an orthonormalisation step, so that at each iteration:

$$\mathbf{W}(n) = \mathbf{W}(n) (\mathbf{W}^H(n)\mathbf{W}(n))^{-1/2} \quad (2.57)$$

where the inverse matrix square root is typically calculated using an efficient estimate.

The complexity of the PASTd algorithm is $\mathcal{O}(NK)$ (clearly visible from the nested “for” loops over N and K).

2.5.2 Fast Data Projection Method (FDPM)

The power method is a simple algorithm to calculate the dominant eigenvector of a matrix: Given a non-zero vector \mathbf{w}_1 and a matrix \mathbf{A} , the iteration

$$\begin{aligned} &\text{for } i = 1, 2, \dots, N \\ &\quad \mathbf{w}_1(i) = \frac{\mathbf{A}\mathbf{w}_1(i-1)}{\|\mathbf{A}\mathbf{w}_1(i-1)\|} \\ &\text{end} \end{aligned} \tag{2.58}$$

will converge to the dominant eigenvector of \mathbf{A} (provided that the M eigenvectors of \mathbf{A} are linearly independent, and there is a clearly dominant eigenvalue, $\lambda_1 > \lambda_2 > \dots > \lambda_M$).

Simultaneous iteration is the application of the power method to several vectors at once:

$$\begin{aligned} &\text{for } i = 1, 2, \dots, N \\ &\quad \mathbf{W}(i) = \frac{\mathbf{A}\mathbf{W}(i-1)}{\|\mathbf{A}\mathbf{W}(i-1)\|} \\ &\text{end} \end{aligned} \tag{2.59}$$

where \mathbf{W} has K columns that form a basis for the dominant subspace (i.e. they span the same subspace as the top K eigenvectors of \mathbf{A}).

Although these basis vectors are independent, they each tend to approach the dominant eigenvector of \mathbf{A} (they become “minimally” independent). Also, because of the multiplication by powers of \mathbf{A} at each successive iteration, the magnitudes of the vectors in \mathbf{W} tend to become very small. By orthogonalising \mathbf{W} and normalising the columns at each iteration, both of these problems may be eliminated. One method of orthonormalisation is the QR decomposition, leading to the **orthogonal iteration** algorithm:

$$\text{for } i = 1, 2, \dots, N \tag{2.60}$$

$$\mathbf{Q}(i)\mathbf{R}(i) = \mathbf{W}(i-1) \quad (\text{QR decomposition}) \tag{2.61}$$

$$\mathbf{W}(i) = \mathbf{A}\mathbf{Q}(i) \tag{2.62}$$

$$\text{end} \tag{2.63}$$

Methods for obtaining the QR decomposition include the Gram-Schmidt process, Householder transformations and successive Givens rotations. Unfortunately, the QR decomposition has a worst-case complexity of $\mathcal{O}(M^3)$ - which is no better than a full SVD, although the idea can form as a basis for practical methods.

It is possible to modify the orthogonal iteration algorithm so that a step size is intro-

duced. This leads to the **adaptive orthogonal iteration** algorithm:

$$\begin{aligned} &\text{for } i = 1, 2, \dots, N \\ &\quad \mathbf{W}(i) = \text{orthonormalise}\{(\mathbf{I} \pm \mu \mathbf{A}) \mathbf{W}(i-1)\} \\ &\text{end} \end{aligned} \tag{2.64}$$

where μ is a scalar that modifies the “step-size” at each iteration. Small values of μ indicate that $\mathbf{W}(n-1)$ is multiplied by an approximation to the identity matrix, leading to very slow adaptation. Large values of μ , conversely, indicate very rapid (and possibly unstable) adaptation. The “ \pm ” sign indicates that either the “+” or “-” operator should be chosen. In the case of “+”, the algorithm generates estimates for the dominant (signal) subspace, whereas “-” leads to estimates of the minor (noise) subspace (compare the “Inverse Iteration” algorithm in Chapter 8 of Golub & Van Loan [4]). This ability to estimate either the signal or noise subspace is potentially a major advantage of the FDPM algorithm (section 2.4 illustrates the increased sharpness of spectral peaks when using the noise subspace).

If \mathbf{A} is a covariance matrix (represented by \mathbf{R} in preceding sections), equation 2.64 may be modified to incorporate the instantaneous estimate of the covariance matrix:

$$\begin{aligned} &\text{for } i = 1, 2, \dots, N \\ &\quad \mathbf{W}(i) = \text{orthonormalise}\{(\mathbf{I} \pm \mu \mathbf{x}(n) \mathbf{x}^H(n)) \mathbf{W}(i-1)\} \\ &\text{end} \end{aligned} \tag{2.65}$$

Yang and Kaveh [25] proposed an iterative solution based on 2.64 that relies on the Gram-Schmidt method for orthonormalisation. The resulting algorithm is referred to as the **Data Projection Method**, and it has a complexity of $\mathcal{O}(K^2N)$. Doukopoulos and Moustakides [2] reformulated this algorithm by:

1. introducing a normalised step size (where steps are normalised by the magnitudes of the input data)
2. replacing the Gram-Schmidt orthonormalisation by an approximate orthonormalisation, based on the Householder transformation

They named this modified algorithm the **Fast Data Projection Method** (FDPM), and went on to prove that the algorithm produces stable subspace estimates that converge exponentially towards orthonormality.

The complete FDPM algorithm may be summarised as follows:

for $n=1, 2, \dots, N$

$$\mathbf{r}(n) = \mathbf{W}^H(n-1)\mathbf{x}(n) \quad (2.66a)$$

$$\mathbf{T}(n) = \mathbf{W}(n-1) \pm \frac{\mu}{\|\mathbf{y}(n)\|^2} \mathbf{y}(n) \mathbf{r}^H(n) \quad (2.66b)$$

$$\mathbf{a}(n) = \mathbf{r}(n) - \|\mathbf{r}(n)\| \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix}^t \quad (2.66c)$$

$$\mathbf{Z}(n) = \mathbf{T}(n) - \frac{2}{\|\mathbf{a}(n)\|^2} (\mathbf{T}(n)\mathbf{a}(n)) \mathbf{a}^H(n) \quad (2.66d)$$

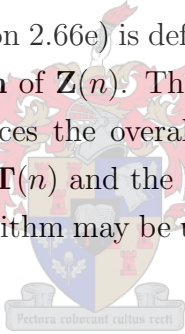
$$\mathbf{W}(n) = \text{normalise}\{\mathbf{Z}(n)\} \quad (2.66e)$$

end

$\mathbf{W}(0)$ is initialised to have M rows (corresponding to the M elements) and K columns, where K is the number of signal (or noise) eigenvectors required ($K \ll M$). For fastest convergence, it should be initialised to some orthonormal matrix (typically the first K columns of the identity matrix).

Note that “normalise” (equation 2.66e) is defined by Doukopoulos and Moustakides as the normalisation of each **column** of $\mathbf{Z}(n)$. This simplification still ensures convergence towards orthonormality and reduces the overall complexity from $\mathcal{O}(K^2N)$ to $\mathcal{O}(KN)$. Equation 2.66d is the product of $\mathbf{T}(n)$ and the Householder transformation of $\mathbf{a}(n)$.

Unlike PAST, the FDPM algorithm may be used to form estimates of either the signal or noise subspaces.



2.5.3 Operator Restriction Analysis (OPERA)

The final method for efficient subspace decomposition, outlined by MacInnes [11], approaches the problem in an entirely different way: OPERA (Operator Restriction Analysis) finds a projection of the input data vectors (with M elements) on to a smaller subspace, with the same dimension, K , as the signal subspace ($K < M$). In this smaller subspace, eigen decomposition of the updated data is a much simpler problem. The remaining task is then to project the updated eigen decomposition back on to the original coordinate system. The vectors that form a basis for the signal subspace are then used as input to a direction-finding algorithm such as MUSIC.

If \mathbf{V} is an **invariant subspace** of \mathbf{R} , any operation of \mathbf{R} on a vector in \mathbf{V} stays within \mathbf{V} (“in \mathbf{V} ” means “in the column space of \mathbf{V} ”).

For example, if \mathbf{R} represents clockwise rotation by an angle θ about the z -axis in \mathbb{R}^3

$$\mathbf{R} = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.67)$$

any linear combination of vectors in the x - y -plane will remain within that plane. The x - and y -axes (or any two independent vectors in the plane) thus make up an **invariant subspace** of the operator \mathbf{R} . Within the x - y subspace, the operation of \mathbf{R} may be described entirely by a smaller 2×2 matrix, \mathbf{A} :

$$\mathbf{A} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \quad (2.68)$$

We make the further restriction that the vectors $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_K$ comprising \mathbf{V} are **eigenvectors** of \mathbf{R} , i.e.

$$\begin{aligned} \mathbf{R}\mathbf{v}_1 &= \lambda_1 \mathbf{v}_1 \\ \mathbf{R}\mathbf{v}_2 &= \lambda_1 \mathbf{v}_2 \\ &\vdots \\ \mathbf{R}\mathbf{v}_K &= \lambda_1 \mathbf{v}_K \end{aligned} \quad (2.69)$$

In this case, \mathbf{V} is a K -dimensional subspace. Suppose that \mathbf{R} operates on a vector \mathbf{x} . There exists some projection matrix, \mathbf{Q} , that projects \mathbf{x} on to \mathbf{V} , giving $\hat{\mathbf{x}}$:

$$\hat{\mathbf{x}} = \mathbf{Q}\mathbf{x} \quad (2.70)$$

(Note that \mathbf{V}^H itself represents the required projection). Because $\hat{\mathbf{x}}$ is in \mathbf{V} , it can be represented by some linear combination of the eigenvectors in \mathbf{V} :

$$\hat{\mathbf{x}} = \mathbf{V}\mathbf{y} \quad (2.71)$$

Here, \mathbf{y} is K -dimensional and contains the “weights” of the columns of \mathbf{V} in the combination. In this new coordinate system (with the columns of \mathbf{V} as axes), $\hat{\mathbf{x}}$ may be completely described by the reduced dimension vector \mathbf{y} .

Equation 2.69 tells us that the operation of \mathbf{R} on the eigenvectors in \mathbf{V} is simply scaling each eigenvector by an eigenvalue. Since \mathbf{y} represents the weights of each column of \mathbf{V} , the operation of \mathbf{R} on \mathbf{y} is equivalent to scaling each element in \mathbf{y} by the corresponding eigenvalue. This operation may be represented as a diagonal matrix, \mathbf{A} , with the eigenvalues of $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_K$ on the diagonal:

$$\mathbf{A} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_K \end{bmatrix} \quad (2.72)$$

In other words, $\mathbf{A}\mathbf{y}$ is equivalent to $\mathbf{R}\hat{\mathbf{x}}$ in the new coordinate system.

Each new data snapshot sampled by the antenna array amounts to a rank-one update of the covariance matrix (equation 2.17). MacInnes [11] showed that a rank-one update of \mathbf{R} , $\mathbf{R} = \mathbf{R} + \mathbf{x}\mathbf{x}^H$, amounts to a rank-one update of \mathbf{A} , using this new coordinate system:

$$\begin{bmatrix} \lambda_1 & 0 & \cdots & 0 & 0 \\ 0 & \lambda_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \lambda_K & 0 \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix} + \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_K \\ \|(\mathbf{I} - \mathbf{Q}) \mathbf{x} \| \end{bmatrix} \begin{bmatrix} y_1 & y_2 & \cdots & y_K & \|(\mathbf{I} - \mathbf{Q}) \mathbf{x} \| \end{bmatrix} \quad (2.73)$$

Note that the update is not simply $\mathbf{A} = \mathbf{A} + \mathbf{y}\mathbf{y}^T$. This is because, in general, the product $\mathbf{x}\mathbf{x}^H$ may lie outside the column space of \mathbf{V} . This could occur if, for example, an additional signal were to appear. To deal with this most general case, \mathbf{A} 's rank is expanded by one, and the projected data snapshot is expanded in length by one. The extra item appended to \mathbf{y} is the norm of the portion of \mathbf{x} falling into the **complement** of \mathbf{V} .

The OPERA algorithm for signal subspace updating is thus as follows:

1. Calculate the projection of the input data snapshot on to the new coordinate system, with the invariant subspace as a basis.
2. Calculate the eigen decomposition of the rank-one update in this system.
3. Project the updated signal eigenvectors back on to the original coordinate system.
4. Eliminate any columns of the subspace that do not correspond to significant eigenvalues (i.e. retain signal eigenvectors only)

The key to OPERA's performance is that the eigen decomposition of the correlation matrix in the new coordinate system is much less computationally expensive than in the original coordinate system ($\mathcal{O}(K^3) \ll \mathcal{O}(M^3)$), since $K \ll M$.

MacInnes also showed that the OPERA algorithm may be used in conjunction with the SVD, instead of the eigen decomposition of the covariance matrix. In this case:

1. Separate left and right subspaces (\mathbf{U} and \mathbf{V}) must be maintained and updated.
2. The rank-one update in the new coordinate system is not necessarily Hermitian.

The two versions of OPERA are generally referred to as EVD-OPERA (eigen decomposition) and SVD-OPERA (Singular Value Decomposition).

A complete description of EVD-OPERA in pseudocode is as follows:

1. Given a new input data snapshot, \mathbf{y} and an estimate of the invariant subspace, calculate:

$$\mathbf{y} = \begin{bmatrix} y_1 & y_2 & \cdots & y_K & y_{K+1} \end{bmatrix}^T, \quad (2.74)$$

where $\begin{bmatrix} y_1 & y_2 & \cdots & y_K \end{bmatrix}^T = \mathbf{V}^H \mathbf{x}$
 and $y_{K+1} = \|x - \mathbf{V}(\mathbf{V}^H \mathbf{x})\|$

\mathbf{y} is the projection of \mathbf{x} into the new coordinate system, with the dimension incremented as in 2.73.

2. Update the covariance matrix in the transformed coordinate system:

$$\mathbf{A} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 & 0 \\ 0 & \lambda_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \lambda_K & 0 \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix} + \mathbf{y}\mathbf{y}^T \quad (2.75)$$

3. Take the eigen decomposition of the update in the transformed coordinate system:

$$\mathbf{A} = \mathbf{W}\mathbf{D}\mathbf{W}^H \quad (2.76)$$

4. Project the eigen decomposition back on to the original coordinate system

$$\mathbf{V}' = \mathbf{V}_x \mathbf{W} \quad (2.77)$$

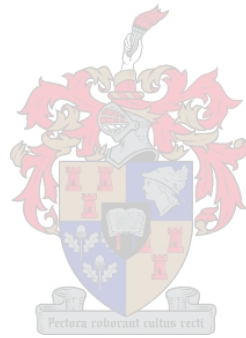
where the projection into the original coordinate system is

$$\mathbf{V}_x = \begin{bmatrix} \mathbf{V} & \frac{x - \mathbf{V}\mathbf{y}}{\|x - \mathbf{V}\mathbf{y}\|} \end{bmatrix} \quad (2.78)$$

$$= \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_K & \frac{x - \mathbf{V}\mathbf{y}}{\|x - \mathbf{V}\mathbf{y}\|} \end{bmatrix} \quad (2.79)$$

5. Set \mathbf{V} equal to \mathbf{V}' , with the least significant eigenvector of \mathbf{V}' removed.
6. Go to 1 and repeat for every new input snapshot received.

MacInnes [11] calculated the overall complexity of OPERA as $\mathcal{O}(MK^2)$.



Chapter 3

Performance and efficiency of subspace tracking algorithms

3.1 Creation of a software signal simulator

A signal simulator was created to provide data that could be used to evaluate the selected subspace tracking algorithms. The simulator was originally created to run under the MATLAB environment, but was subsequently rewritten in C++ as a stand-alone PC program.

The simulator was created in accordance with the following requirements:

1. IF (Intermediate Frequency) signals may be created with arbitrary bandwidth, IF and simulated RF frequencies.
2. Antenna geometries with an arbitrary number of elements (channels) may be simulated. Antenna modelling shall assume omnidirectional response patterns with no mutual coupling.
3. Azimuth, elevation and distance from the antenna array may be set for each signal.
4. Simulated signals may “orbit” (rotate about the origin of the antenna array) at user specified velocities.
5. Simulated signals may include the following types:
 - (a) Sinusoidal
 - (b) Narrowband Gaussian white noise with a user-defined bandwidth
 - (c) Wideband Gaussian white noise with a user-defined bandwidth (where phases are accurate across the entire bandwidth relative to the array geometry)
 - (d) Modulated audio signals with a user-defined bandwidth

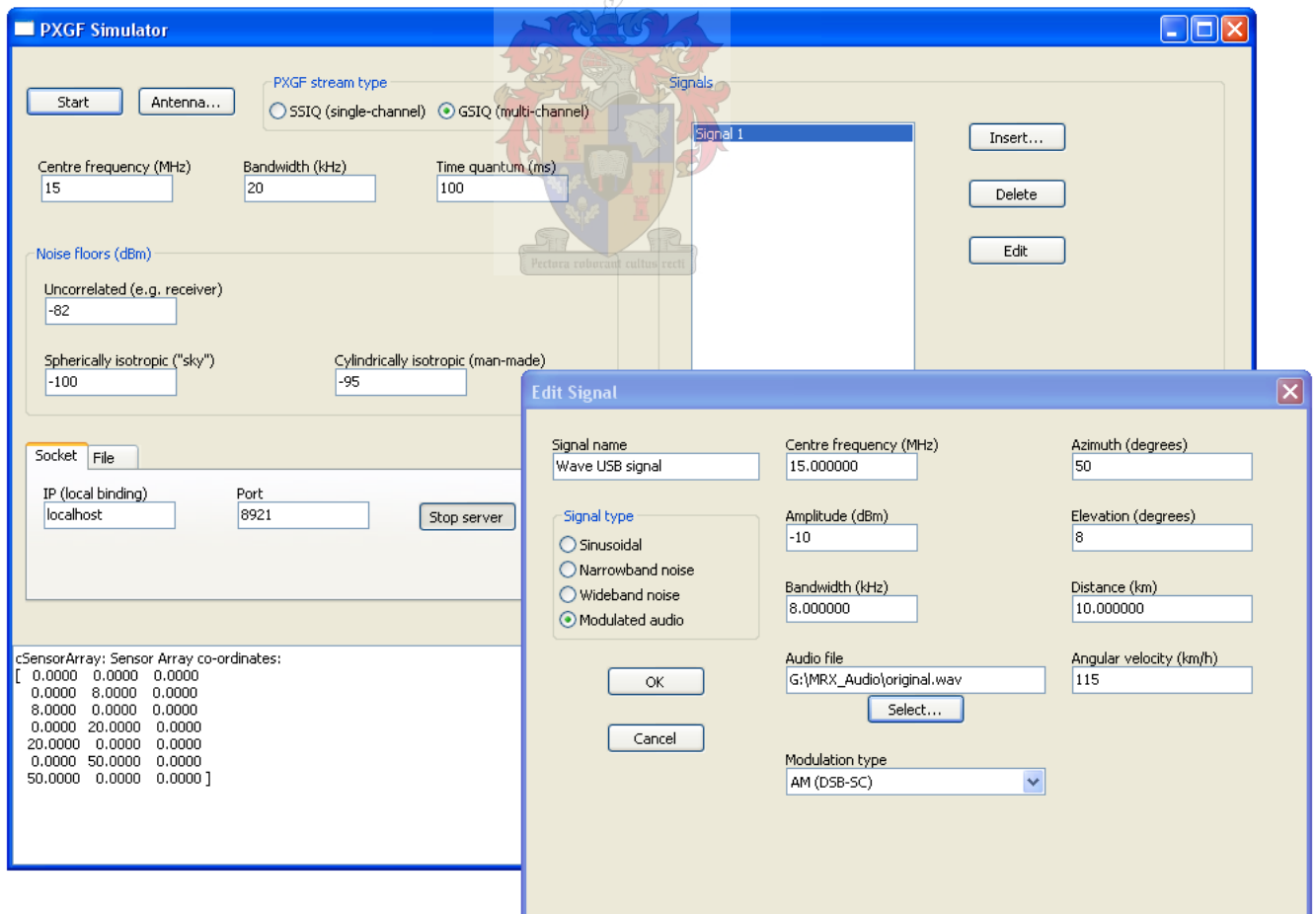
6. Gaussian white noise may be added to the simulated data with the following spatial distributions:
 - (a) Uncorrelated (e.g. noise arising in receiver preamplifier circuitry)
 - (b) Spherically isotropic (e.g. natural sources of interference in the atmosphere)
 - (c) Cylindrically isotropic (e.g. man-made sources originating in the same plane as the antenna array)

The noise floor of each type of noise may be independently set.

7. Simulated IF data may be written to a file or streamed from a server socket. IF samples are formatted as 16-bit complex integers and are transmitted using the PXGF protocol (PXGF is a standard for streaming IF data created by Peralex Electronics).

Figure 3.1 is a screenshot of the Signal Simulator.

Figure 3.1: Screenshot of the antenna array signal simulator



Generation of noise with a specified covariance matrix

Section 2.2.2 described the formulae for creating covariance matrices resulting from uncorrelated, spherically isotropic and cylindrically isotropic noise. Note that the superposition (sum) of such signals results in the sum of their respective covariance matrices, since the expectation operator (equation 2.12) is linear. In other words, the total noise covariance matrix takes the form

$$\mathbf{R}_N = \mathbf{R}_U + \mathbf{R}_S + \mathbf{R}_C \quad (3.1)$$

where \mathbf{R}_U , \mathbf{R}_S and \mathbf{R}_C are the contributions from the uncorrelated, cylindrical and spherical noise respectively.

Given an M by N matrix \mathbf{A} of uncorrelated noise with variance 1, it is possible to “shape” \mathbf{A} to have the specified covariance matrix \mathbf{R}_N by applying an operator \mathbf{L} :

$$\mathbf{X}_N = \mathbf{L}\mathbf{A} \quad (3.2)$$

\mathbf{L} may be found by the Cholesky decomposition of \mathbf{R}_N , since \mathbf{R}_N is symmetric positive-definite:

$$\mathbf{R}_N = \mathbf{L}\mathbf{L}^H \quad (\mathbf{L} \text{ is lower triangular}) \quad (3.3)$$

Then,

$$E\{\mathbf{X}_N\mathbf{X}_N^H\} = E\{(\mathbf{L}\mathbf{A})(\mathbf{L}\mathbf{A})^H\} = E\{\mathbf{L}\mathbf{A}\mathbf{A}^H\mathbf{L}^H\} \quad (3.4)$$

$$= \mathbf{L}E\{\mathbf{A}\mathbf{A}^H\}\mathbf{L}^H \quad (3.5)$$

$$= \mathbf{L}\mathbf{L}^H \quad (\mathbf{A} \text{ is uncorrelated with variance 1}) \quad (3.6)$$

$$= \mathbf{R}_N \quad (3.7)$$

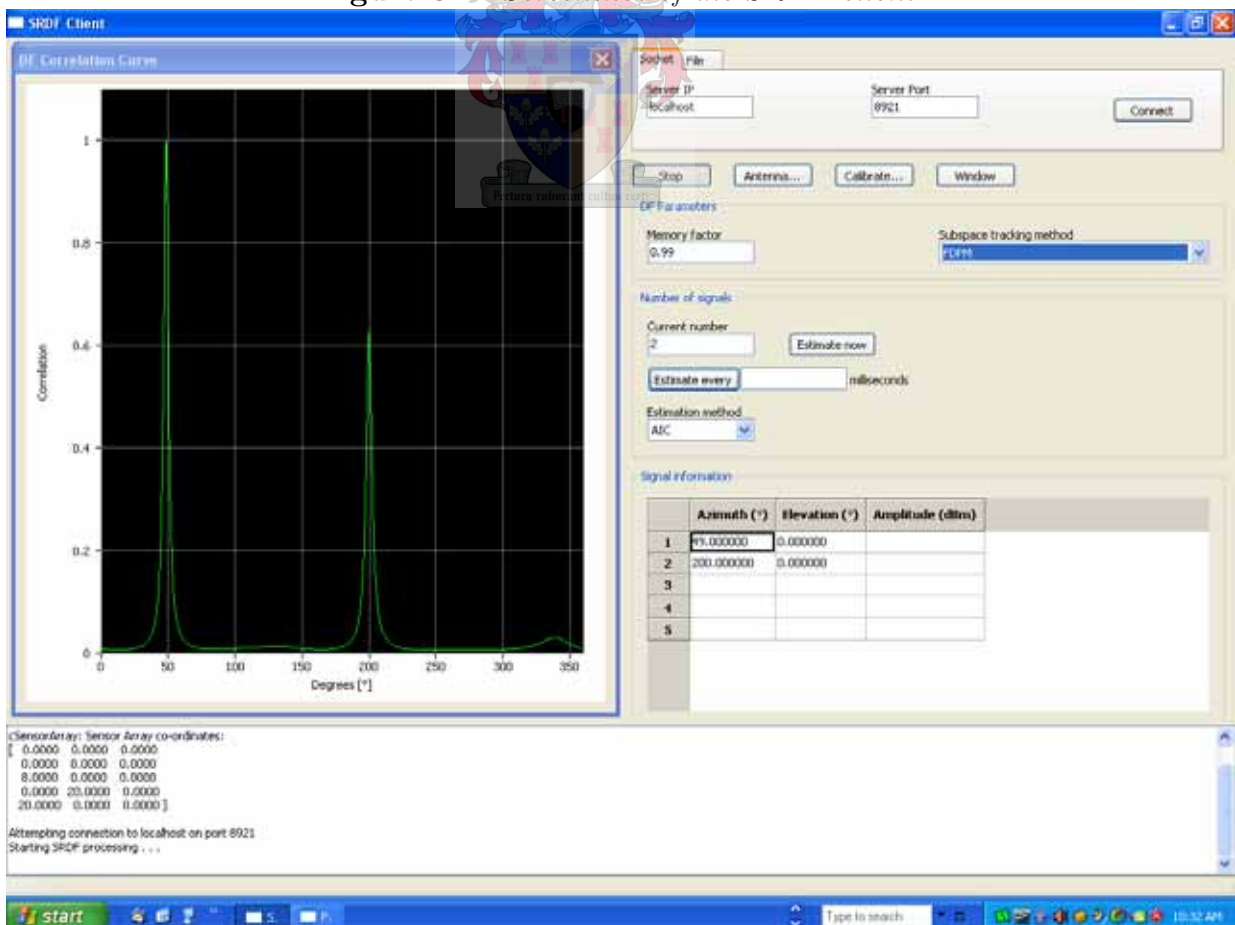
3.2 Creation of a Superresolution Direction Finding client

Evaluation of the selected subspace tracking methods was performed using a client that operates on data generated by the Signal Simulator. The client implemented the PASTd, FDPM and OPERA methods for subspace tracking and was prototyped using the MATLAB environment. The SRDF (Superresolution Direction Finding) client was then rewritten as a GUI-based C++ program.

Matrix and vector calculations relied on a purpose-written set of C++ classes that wrapped BLAS and LAPACK implementations. BLAS (Basic Linear Algebra Subprograms) and LAPACK (Linear Algebra PACKage) are well known optimised linear algebra libraries that were originally written in Fortran 77 and are maintained in the Netlib repository [12]. In this case, the specific implementation used was the Math Kernel Library supplied by Intel®.

Figure 3.2 is a screen shot of the SRDF client detecting narrowband noise signals at 50° and 200° azimuths (at the same frequency of 15 MHz).

Figure 3.2: Screenshot of the SRDF client



CHAPTER 3. PERFORMANCE AND EFFICIENCY OF SUBSPACE TRACKING ALGORITHMS

The performance results in this chapter were based on the output of the SRDF client. Some of the visualisations were created by importing data into MATLAB or using the MATLAB prototype client.



3.3 Response to a step change in Direction-Of-Arrival

From the theory of linear systems we know that many of a system's characteristics are clearly visible in the step response. The response to a Dirac impulse completely describes any Linear Time-Invariant system.

A subspace direction finding system is causal and time-invariant, although it is not linear (e.g. the search for maxima in the MUSIC spectrum is a non-linear process).

However, the response to a step change in direction is still a very useful tool for analysing how a DOA tracking system may operate in practice. It is less useful for memoryless DF methods (such as the Watson Watt and correlative DF algorithms).

The following step responses are based on simulated input signals that stay at one angle-of-arrival for L samples, before changing to another angle-of-arrival for the remaining $N - L$ samples. Although there are no real-world signal sources that can change locations instantaneously, this represents a "worst-case" scenario that illustrates how a particular subspace tracker might adapt.

3.3.1 Step change in DOA of one signal

Although subspace direction finding methods are well suited to an environment consisting of many signals, the ability to track a single signal is of considerable importance.

For the first set of comparisons, we consider a 10-element circular antenna array with a radius of 15 metres. Elements are spaced equally around the circumference of the circular. Circular geometry was chosen because of its isotropic azimuth response (i.e. the response pattern is not a significant function of azimuth).

A Gaussian noise signal with a bandwidth of 10 kHz arrives at an azimuth of 40° (elevation of 0°). After 120 samples, the azimuth is changed instantly to 120° . The noise environment consists of uncorrelated noise with a noise floor of -40 dBm, spherically isotropic noise with a noise floor of -50 dBm and cylindrically isotropic noise with a noise floor of -50 dBm. The total Signal-to-Noise-Ratio is roughly 50 dB over the 10 kHz bandwidth.

As a reference, we start by examining the DOA spectrogram when using the (inefficient) eigen decomposition for tracking. Eigen decomposition is performed on rank-one updates to the correlation matrix using the LAPACK function `cheev`. The first set of results (figure 3.3) uses the signal subspace and plots the DOA spectrogram for different values of β (equation 2.18). As β is increased, the transition between the two azimuths becomes less defined and tends to lag increasingly behind the input step transition ($n = 120$).

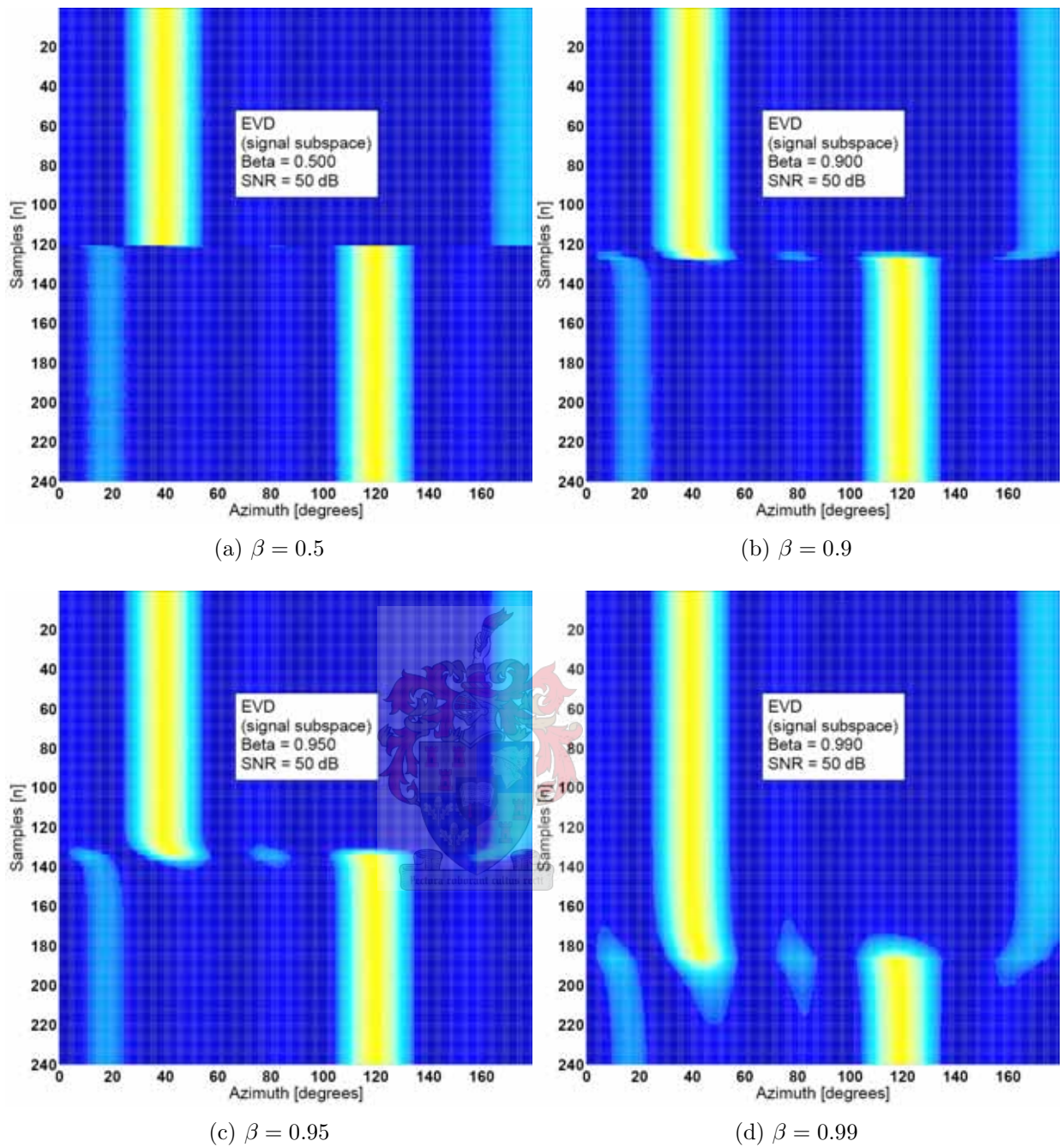


Figure 3.3: Azimuth step response (one signal) using the eigen decomposition and signal subspace

The step response may be plotted similarly for the noise subspace when performing the complete eigen decomposition. Figure 3.4 shows the results. Notice that the peaks corresponding to the DOA are much narrower (the **superresolution** property). Very high values of β again result in less definition at the transition and increased lag.

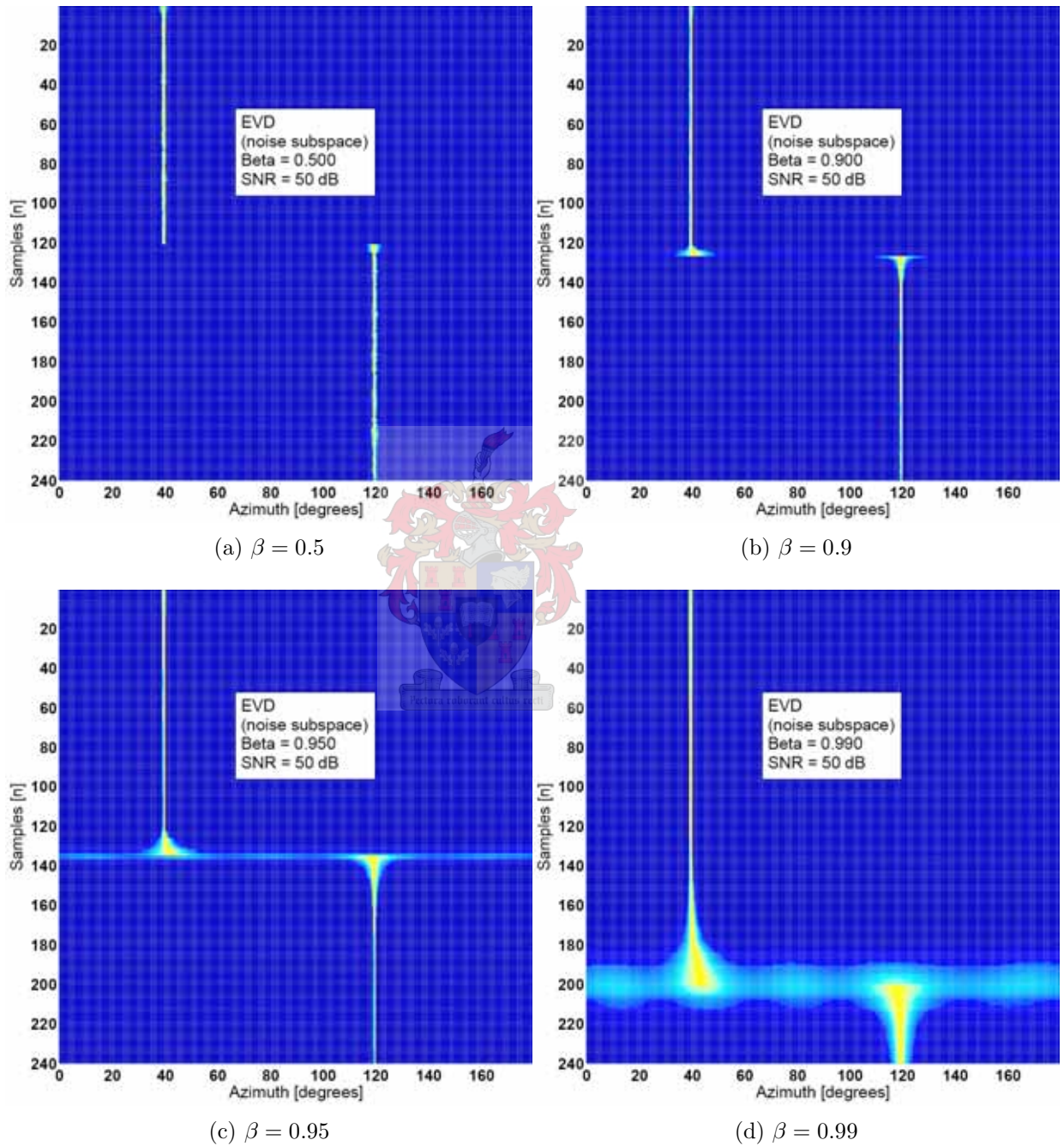


Figure 3.4: Azimuth step response (one signal) using the eigen decomposition and signal subspace

Influence of frequency

As the signal frequency changes (given constant array geometry), the width of the main lobe in the DOA spectrum is altered. Lower frequencies result in a wider main lobe, while higher frequencies result in a narrower main lobe. At even higher frequencies, the array becomes ambiguous (i.e. it is impossible to distinguish reliably which peak corresponds to the actual DOA). At 80 MHz, there are many spurious peaks in the DOA spectrum.

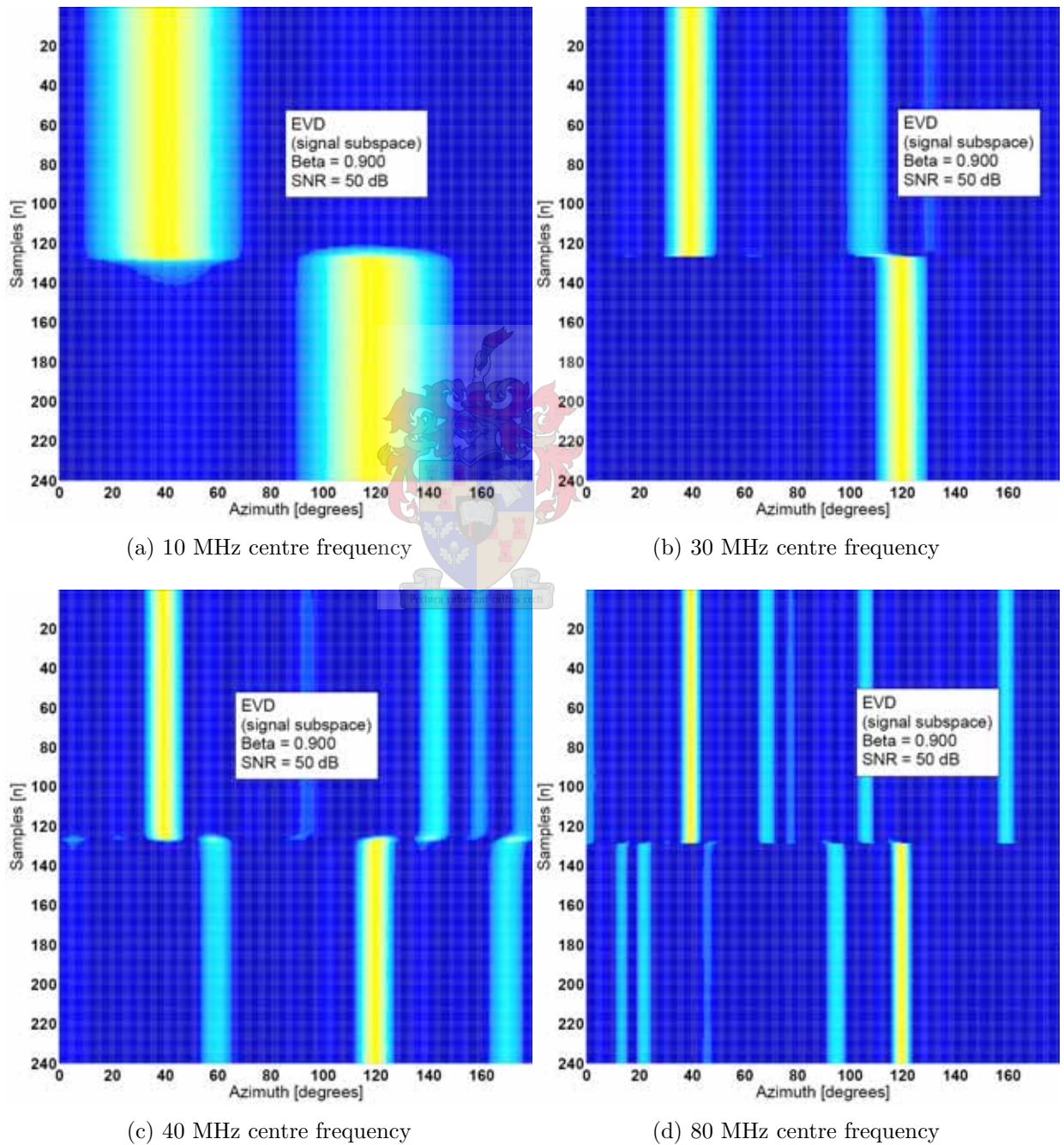


Figure 3.5: Azimuth step response (one signal) using the eigen decomposition and signal subspace

PASTd azimuth step response for one signal

Figure 3.6 indicates the response of the PASTd algorithm to a step change in direction, using one signal. The azimuths appear to have the peaks centred correctly at 40° and 120° , and the initial convergence seems very fast (no ambiguous directions are observed near $n = 0$). However, the lag at equivalent values of β seems to be much greater, compared to the eigen decomposition.

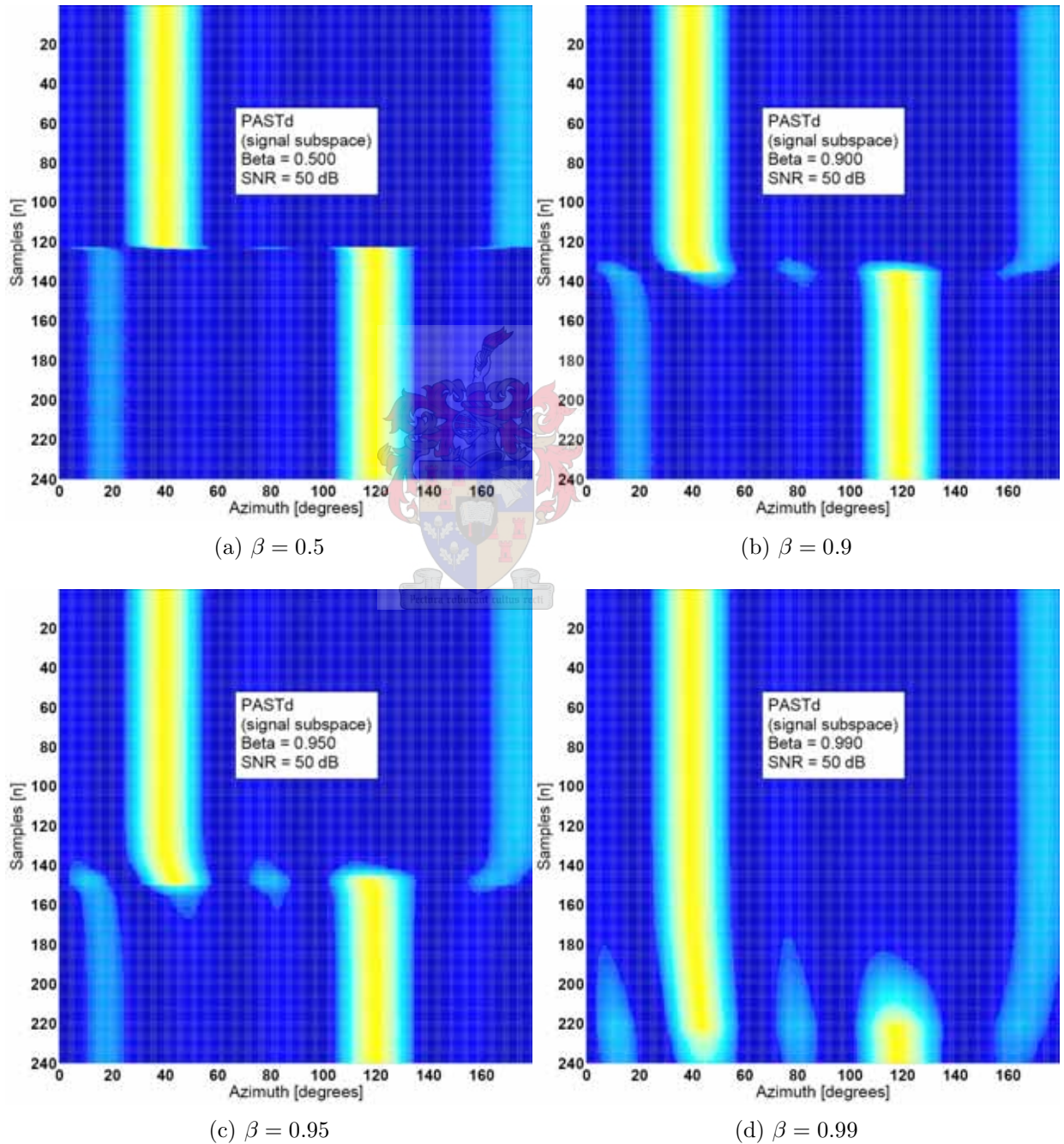


Figure 3.6: Azimuth step response (one signal) using PASTd and the signal subspace

FDPM azimuth step response for one signal (signal subspace)

The FDPM response using the signal subspace is depicted in figure 3.7. The initial convergence speed does not seem to be as fast as PASTd (initial ambiguities are particularly bad for $\beta = 0.99$). The lag is even longer than PASTd for equal values of β , indicating that smaller values of β should generally be chosen.

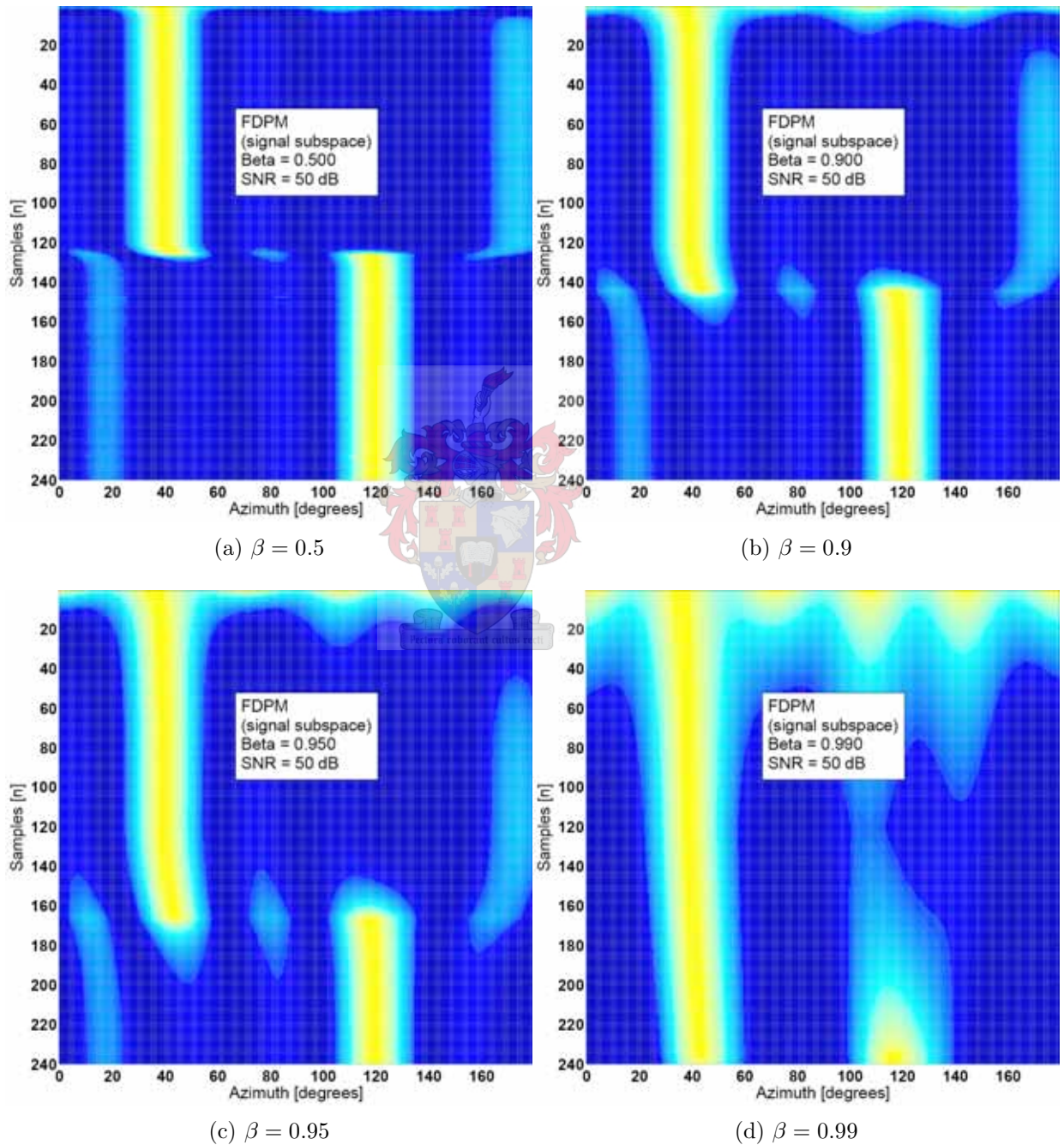


Figure 3.7: Azimuth step response (one signal) using FDPM and the signal subspace

FDPM azimuth step response for one signal (noise subspace)

The step response using the FDPM algorithm and noise subspace is depicted in figure 3.7. The azimuth peak is much sharper than when using the signal subspace. The initial convergence seems to require at least 50 samples. The azimuth results when $\beta = 0.5$ are more influenced by noise than the case when $\beta = 0.9$. For $\beta = 0.99$, the azimuth results do not ever seem to converge properly, indicating that smaller values of β should be chosen.

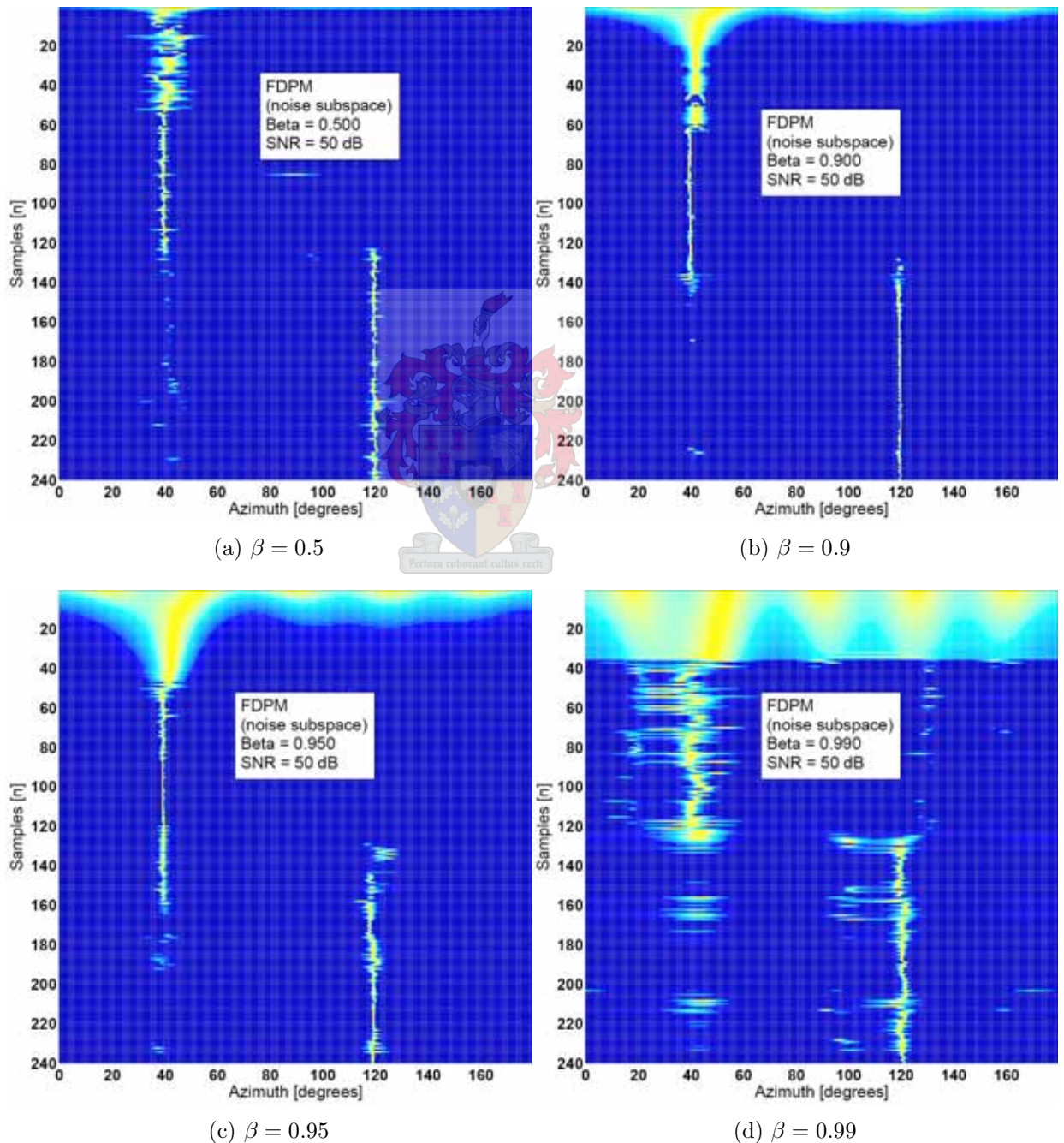


Figure 3.8: Azimuth step response (one signal) using FDPM and the noise subspace

OPERA azimuth step response for one signal (signal subspace)

The response to step change in azimuth when using the EVD-OPERA algorithm is shown in figure 3.9. The initial convergence is fast (at least as fast as PASTd), although it displays more lag than PASTd for equivalent values of β .

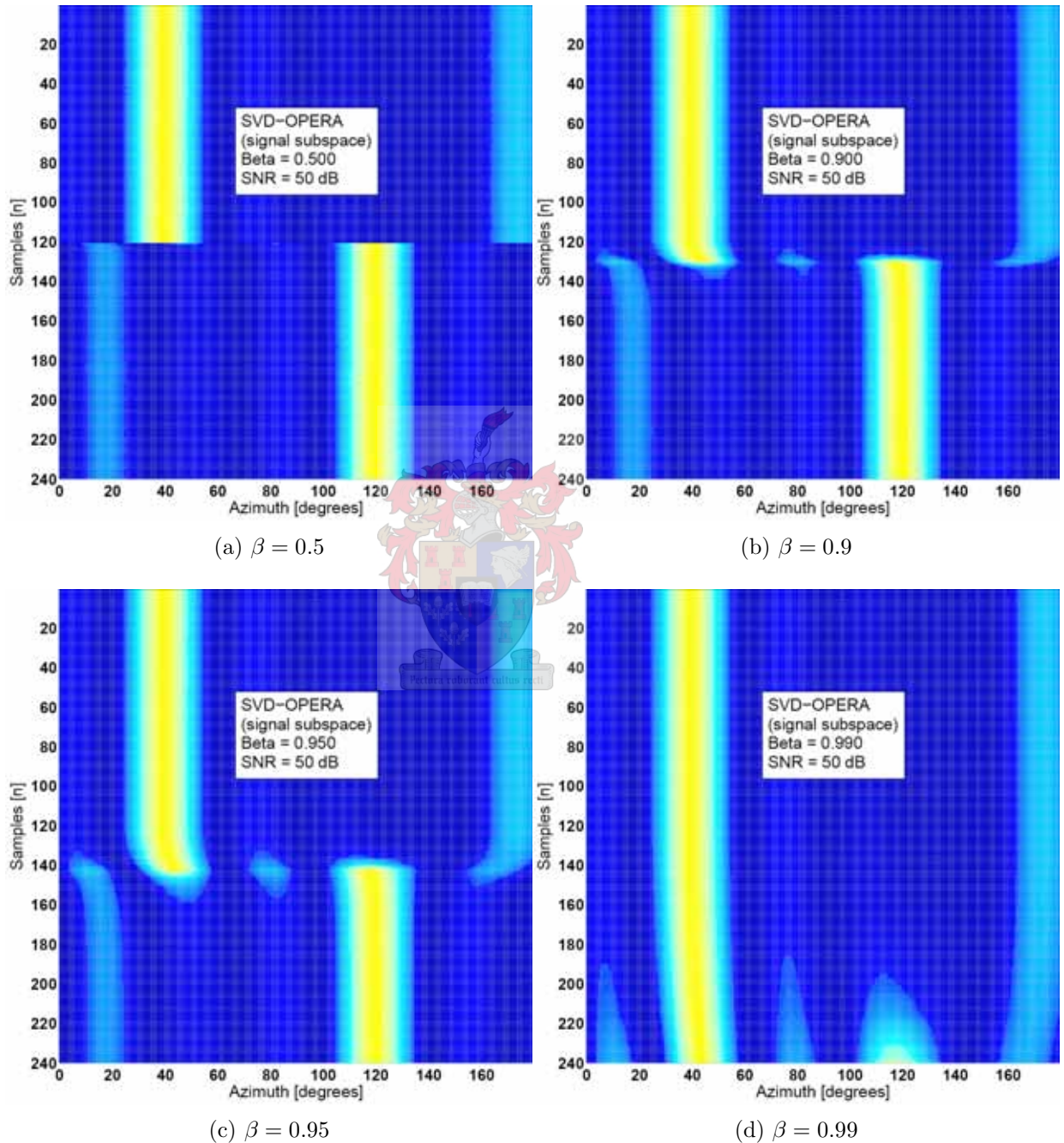


Figure 3.9: Azimuth step response (one signal) using EVD-OPERA and the signal subspace

3.3.2 Step change in Direction-Of-Arrival of two signals

The ability to track more than one signal at a given frequency is a major advantage of subspace direction finding methods. Simpler DF methods (such as the Watson Watt algorithm and correlative interferometer) are not able to display independent DOA results for multiple signals at the same frequency. The next set of tests display each algorithm's ability to track two signals at the same frequency. The two signals each undergo step changes in direction-of-arrival. The first signal starts at an azimuth of 100° and then changes instantly to an azimuth of 300° after 150 samples. The second signal starts at an azimuth of 200° and changes to an azimuth of 50° after 250 samples. The location of the peak in the MUSIC spectrum is used to identify the azimuth of each signal (an optimised search procedure with thresholding was used to perform the tests). The signal-to-noise ratio was set at 20 dB for all of the tests.

Eigen decomposition azimuth step response for two signals (signal subspace)

Figure 3.10 shows the step responses when using the eigen decomposition and signal subspace. The simplistic search procedure is not able to identify which signal changes azimuth and which remains constant at each transition, although the results are effectively the same (the correct azimuths are displayed, although the colours may be wrong). The actual trajectory of the first signal is shown using a solid red line, while the actual trajectory of the second signal is shown using a dashed red line. The azimuth results are shown using blue and green lines.

More noise is evident in the DOA results when $\beta = 0.5$, owing to the fact that older samples are highly deweighted and not much integration is performed. At higher values of β , the lag in azimuth transition becomes more pronounced, although the azimuth is tracked more stably. There appears to be a small amount of “pre-transition” mistracking (mainly evident in the blue trajectory).

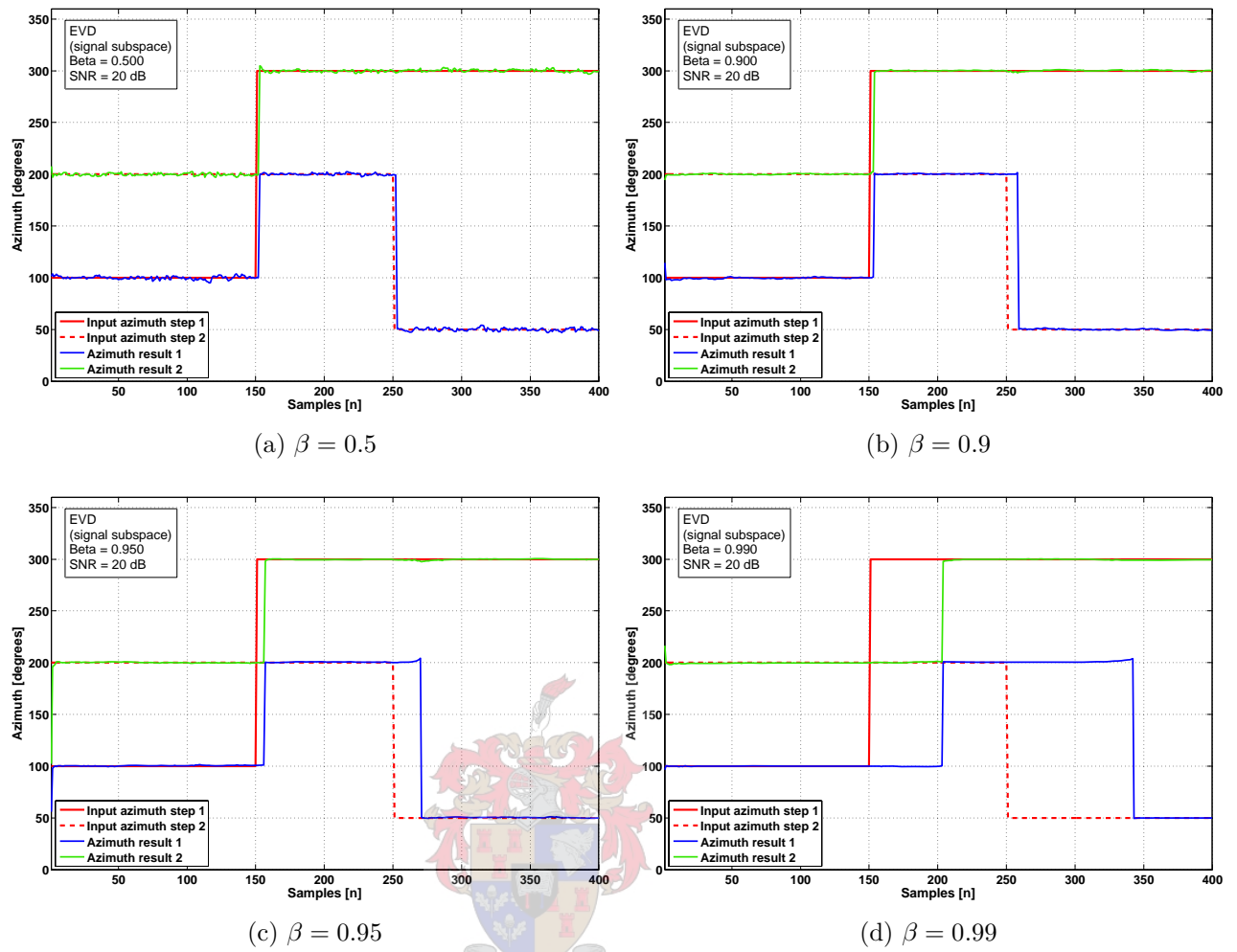


Figure 3.10: Azimuth step response (two signals) using the eigen decomposition and signal subspace

PASTd azimuth step response for two signals (signal subspace)

The step responses when using the PASTd algorithm are shown in figure 3.11. When $\beta = 0.5$, the algorithm appears to be more susceptible to noise than the eigen decomposition. When $\beta = 0.95$, there is very little visible noise, although there is some initial mistracking. The azimuth is tracked initially, but doesn't adapt correctly to the step changes when $\beta = 0.99$. Predictably, lag increases at higher values of β .

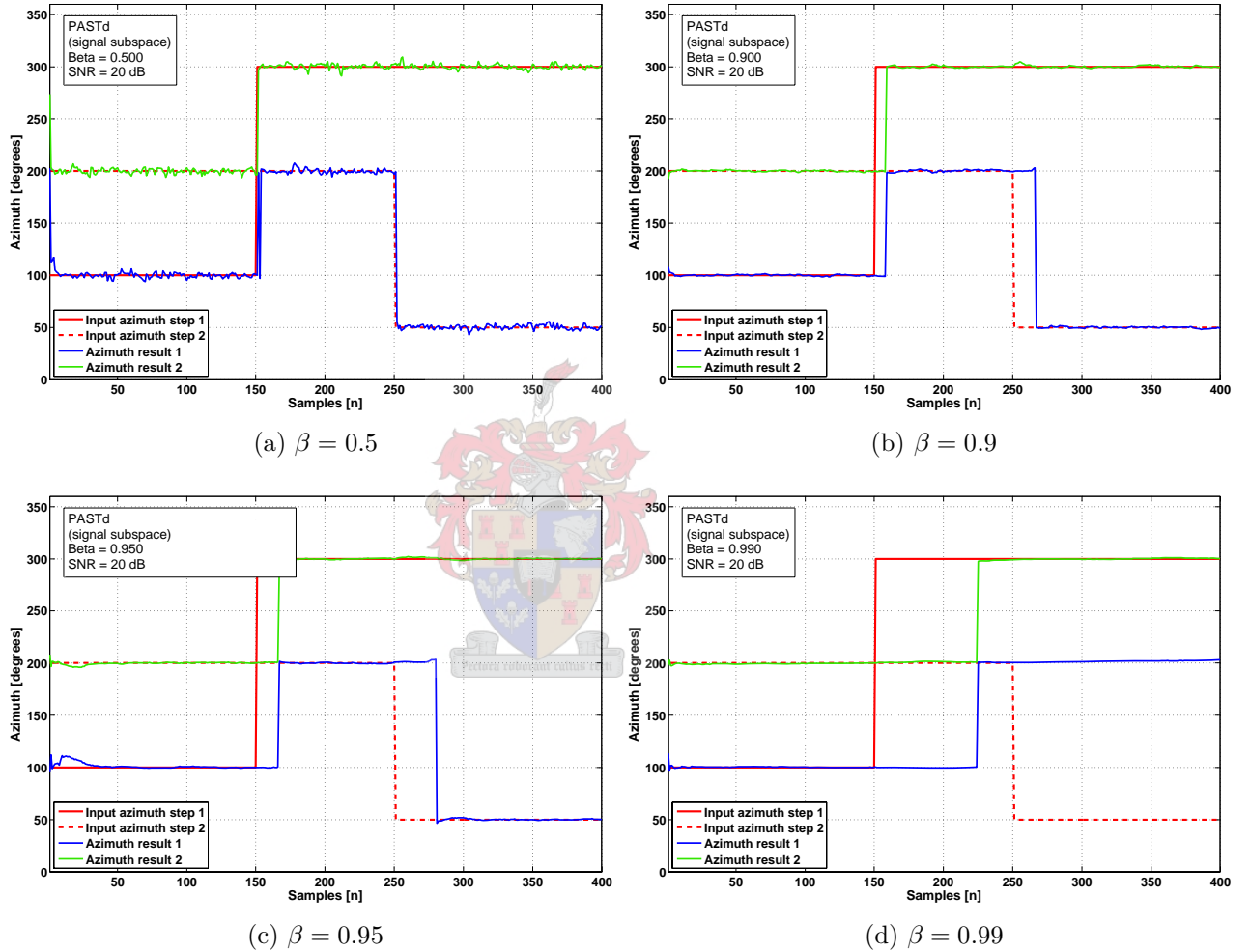


Figure 3.11: Azimuth step response (two signals) using PASTd and the signal subspace

FDPM azimuth step response for two signals (signal subspace)

Figure 3.12 shows the step response when using the FDPM algorithm for subspace tracking. The algorithm seems more sensitive to noise than PASTd when $\beta = 0.5$, although significantly less noise is visible when $\beta = 0.9$ and $\beta = 0.95$. Some initial mistracking is visible in each plot, and the azimuth results are not stable for at least 50 - 80 samples. A large amount of lag is evident, and it appears that $\beta = 0.9$ provides the best trade-offs between lag and susceptibility to noise. When $\beta = 0.99$, the azimuth results do not converge correctly.

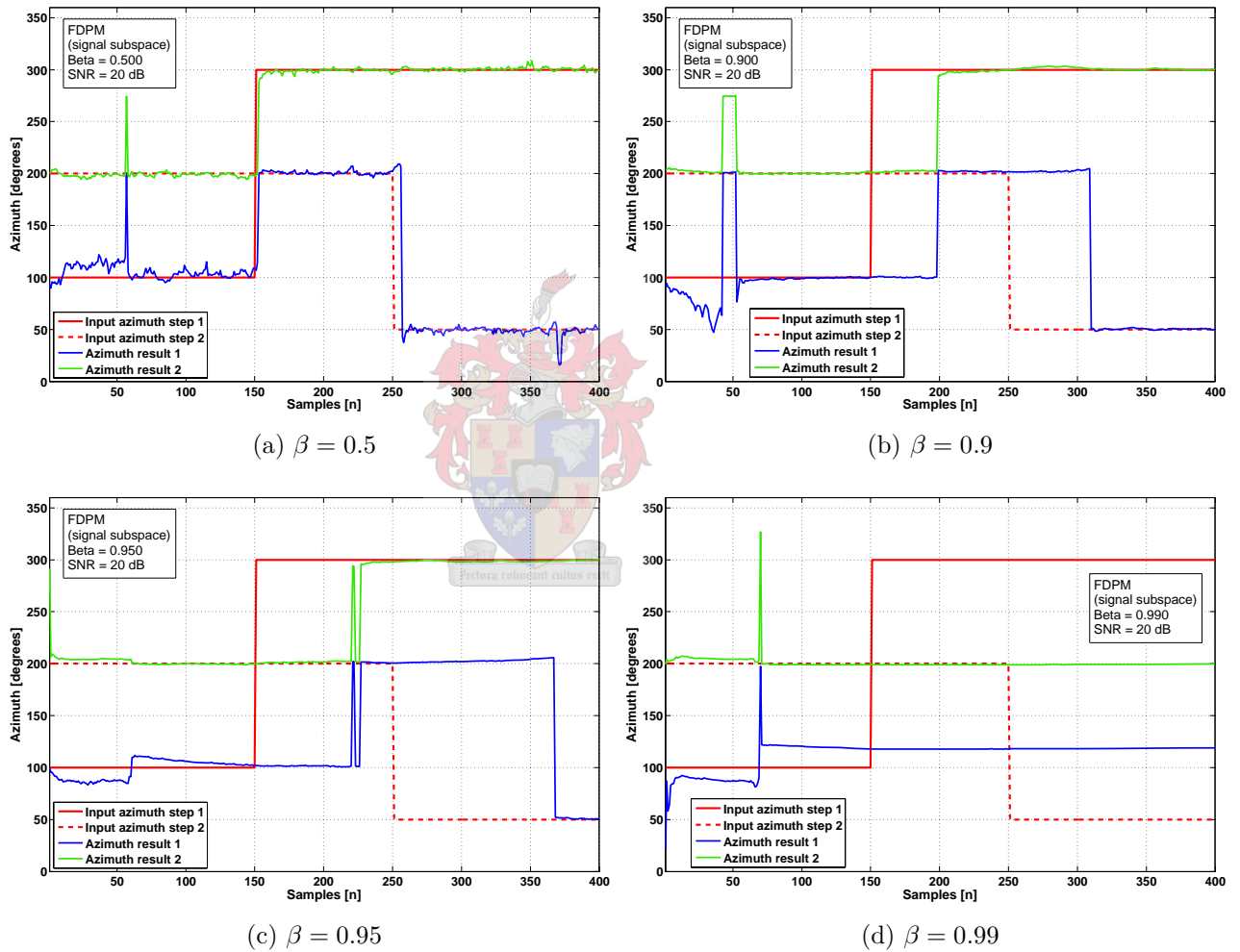


Figure 3.12: Azimuth step response (two signals) using FDPM algorithm and the signal subspace

FDPM azimuth step response for two signals (noise subspace)

The FDPM algorithm may also be used to estimate the noise subspace, producing the step response shown in figure 3.13. The plots are overall a lot more noisy, suggesting that this may be a tradeoff of the increased resolution when using the noise subspace. The azimuth results are tracked correctly when $\beta = 0.9$ and $\beta = 0.95$, although lots of noisy oscillation is visible. Other values of β do not produce acceptable tracking.

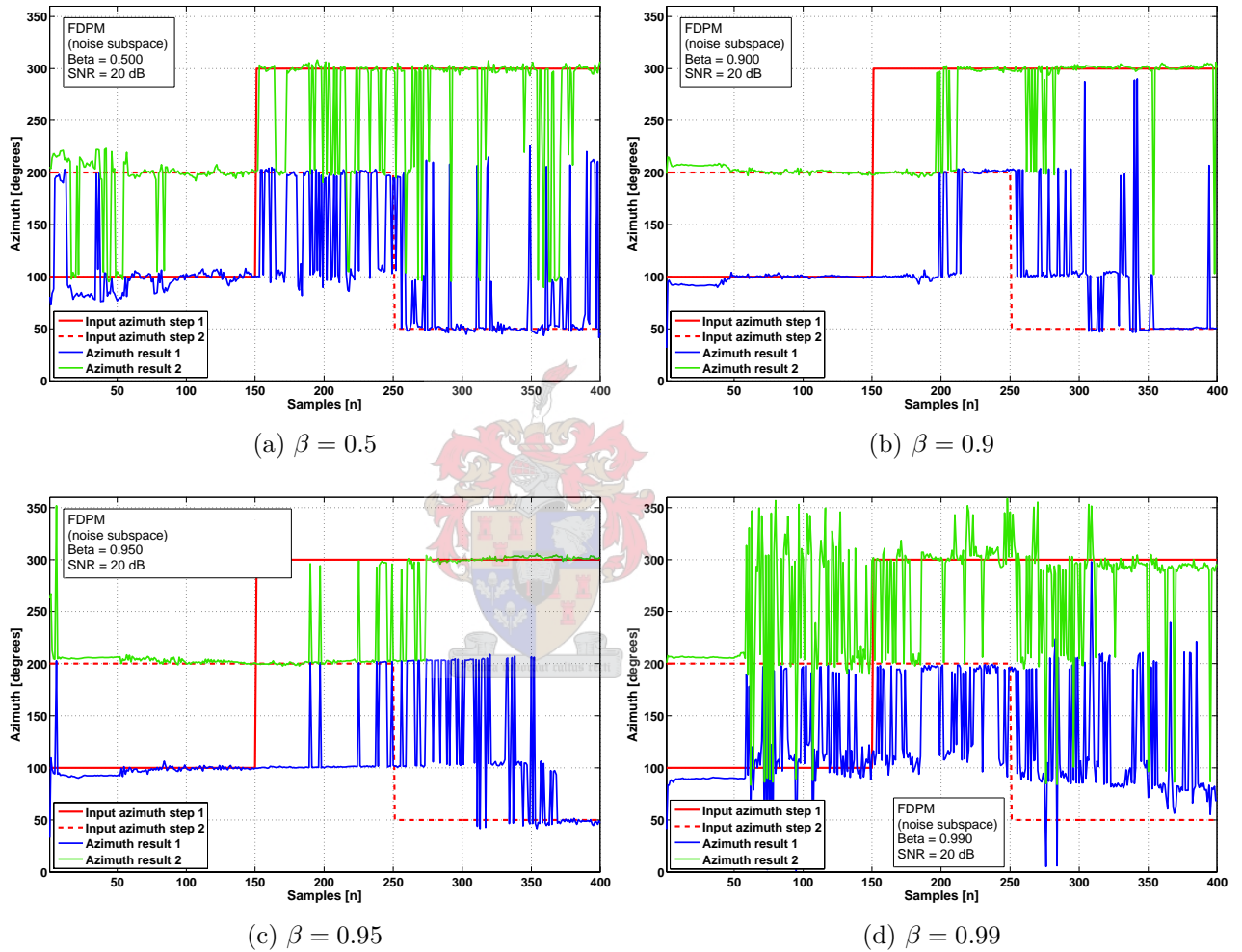


Figure 3.13: Azimuth step response (two signals) using FDPM algorithm and the noise subspace

OPERA azimuth step response for two signals (signal subspace)

The OPERA step responses (figure 3.14) show a close resemblance to those produced by the eigen decomposition. This is possibly not surprising, as the OPERA algorithm employs an eigen decomposition (in a subspace of reduced size) at its core. Noise is worst when $\beta = 0.5$, although the algorithm outperforms PASTd and FDPM in this regard. A very long lag time of more than 100 samples is evident when $\beta = 0.99$.

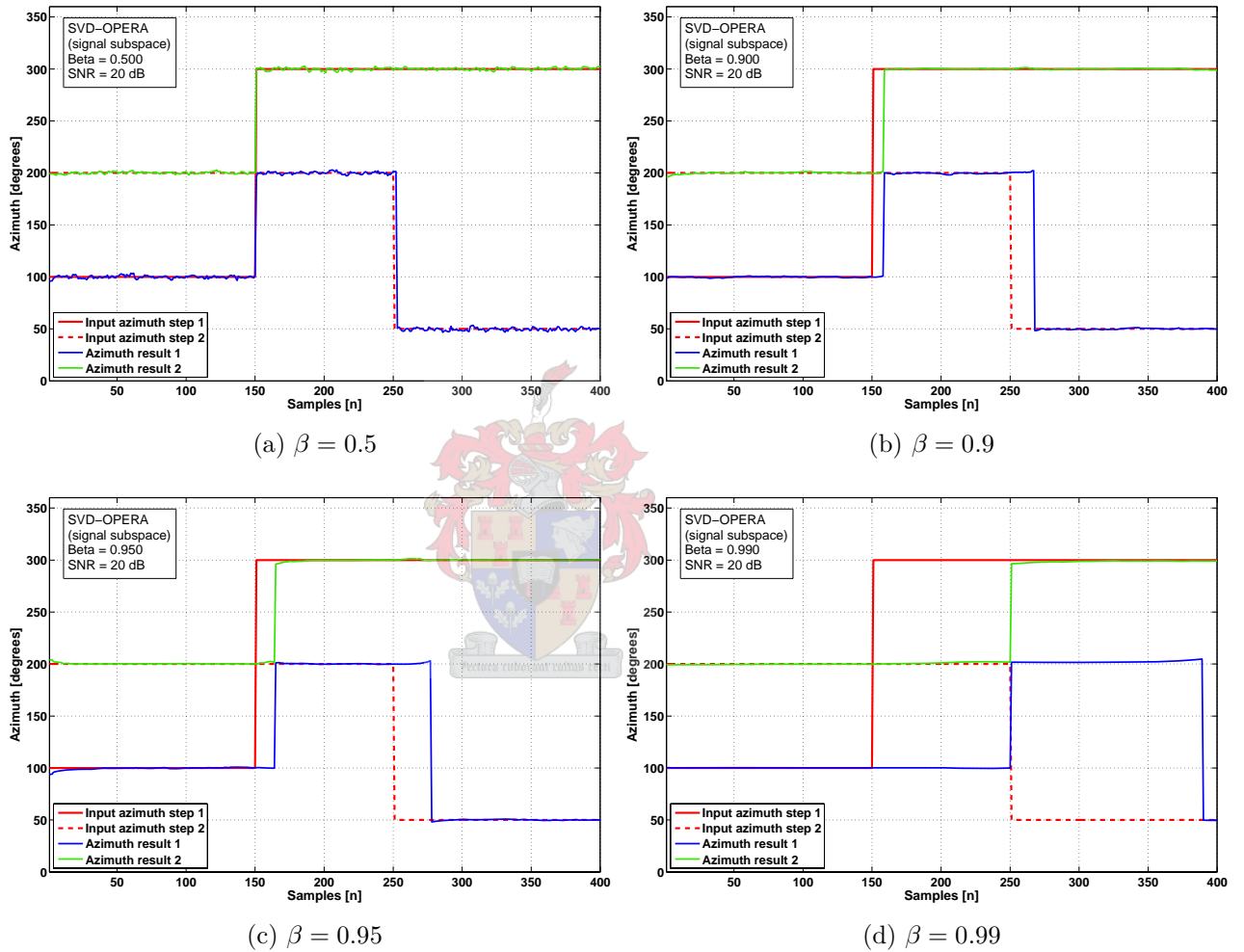


Figure 3.14: Azimuth step response (two signals) using OPERA algorithm and the noise subspace

3.3.3 Sensitivity to noise

As the Signal-to-Noise-Ratio (SNR) decreases, it becomes impossible to determine the Direction-Of-Arrival of a given signal. Even with large integration (memory) factors, at very poor Signal-to-Noise-Ratios, the systematic correlations due to signals become swamped by the random correlations produced by uncorrelated noise. In radio surveillance applications, it is frequently necessary to perform direction-finding on very weak or distant signals that have low Signal-to-Noise-Ratios. It is thus useful to compare how the selected subspace tracking algorithms degrade as the SNR decreases.

For the following tests, two signals were simulated with 20° of separation in azimuth. An 8-element circular array with a 40 metre radius was used. The simulated SNR was varied from -30 dB (the signals were well below the noise floor) to +40 dB, in 2 dB increments. At each value particular SNR, 100 trials were performed. The MUSIC spectrum was calculated for each trial, using 200 snapshots and a value of $\beta = 0.95$ (The results obtained in section 3.3.2 indicates that even the slowest algorithm converges after approximately 150 samples using this memory factor). The average error in azimuth was calculated to produce a plot of azimuth error versus SNR (figure 3.15).

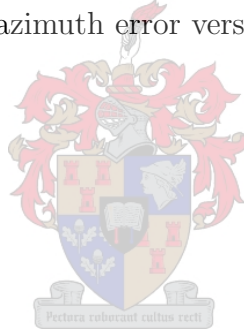
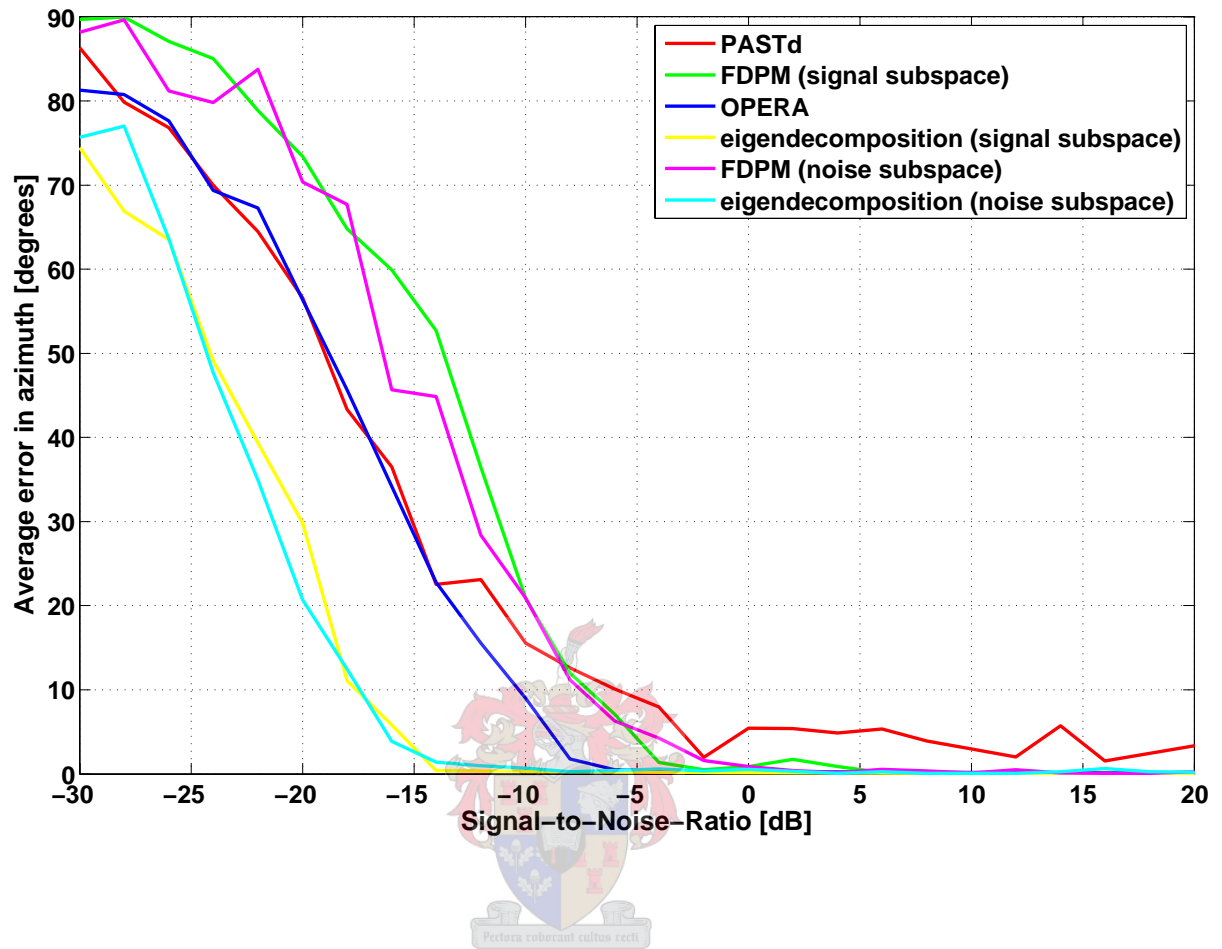


Figure 3.15: Average error in azimuth versus Signal-to-Noise-Ratio



The eigen decomposition clearly has the best performance under low SNR conditions. Both the noise and signal subspace variants of the eigen decomposition perform similarly. The OPERA tracking method appears to have the next best performance when the SNR is low. PASTd performs similarly to OPERA when the SNR is very low, although it does not seem to improve proportionally as the SNR increases. This may possibly be because PASTd does not enforce strict orthonormality of the estimated eigenvectors. The noise and signal subspace variants of the FDPM algorithm perform similarly. Both FDPM algorithms are somewhat worse than all of the other algorithms under low SNR conditions, although they outperform PASTd as the SNR improves.

Note that all of the subspace methods are capable of performing DF even when the SNR is negative (i.e. when the signal power is less than the noise power). Simple phase based DF methods (such as the Watson Watt and interferometric DF methods) do not have this property.



3.3.4 Minimum angular separation

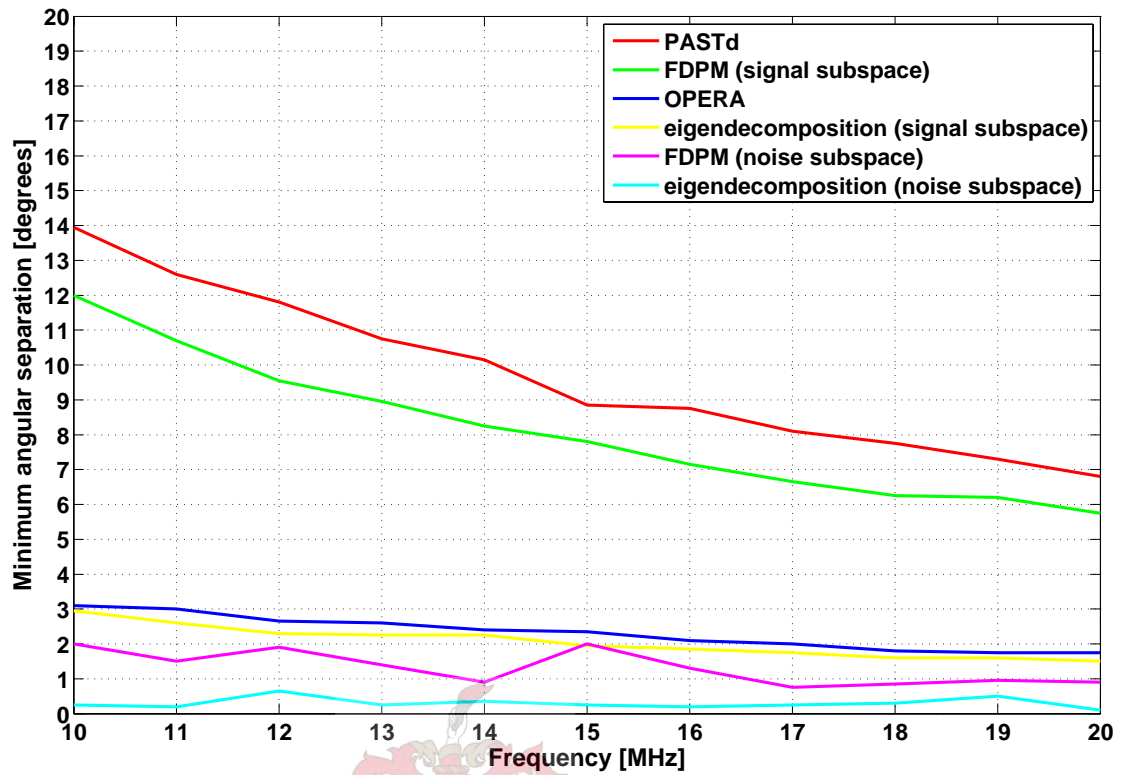
As two signals move closer together (in both azimuth and elevation), their corresponding eigenvectors in the covariance matrix become increasingly similar and tend to collapse into a single eigenvector. Such signals are closely correlated, leading to a rank deficient covariance matrix with only one dominant eigenvalue. One possible test of a subspace tracking method is how well it deals with correlated signals. There are modifications to the subspace tracking method that deal better with correlated signals (such as using sub-arrays and spatial smoothing), although they are beyond the scope of this thesis. However, some decomposition methods are intrinsically better at dealing with correlated sources. Note that highly correlated signals generally imply an ill-conditioned or (in the worst case) singular covariance matrix.

The following tests keep a fixed antenna geometry and run repeated simulation tests to determine the minimum angular separation that can be resolved. Following generation of the MUSIC azimuth spectrum, a search procedure locates the two largest turning points, corresponding to the directions-of-arrival. The average error of the two peaks in degrees is then determined. Each such trial is repeated 40 times, to calculate an overall average error in degrees. Successful resolution of the two directions-of-arrival is defined as an average error of less than 1° . Each trial utilises 150 snapshots and the memory factor β is set to 0.95. The antenna configuration used for the first set of results is a circular array of five elements, with a radius of 40m.

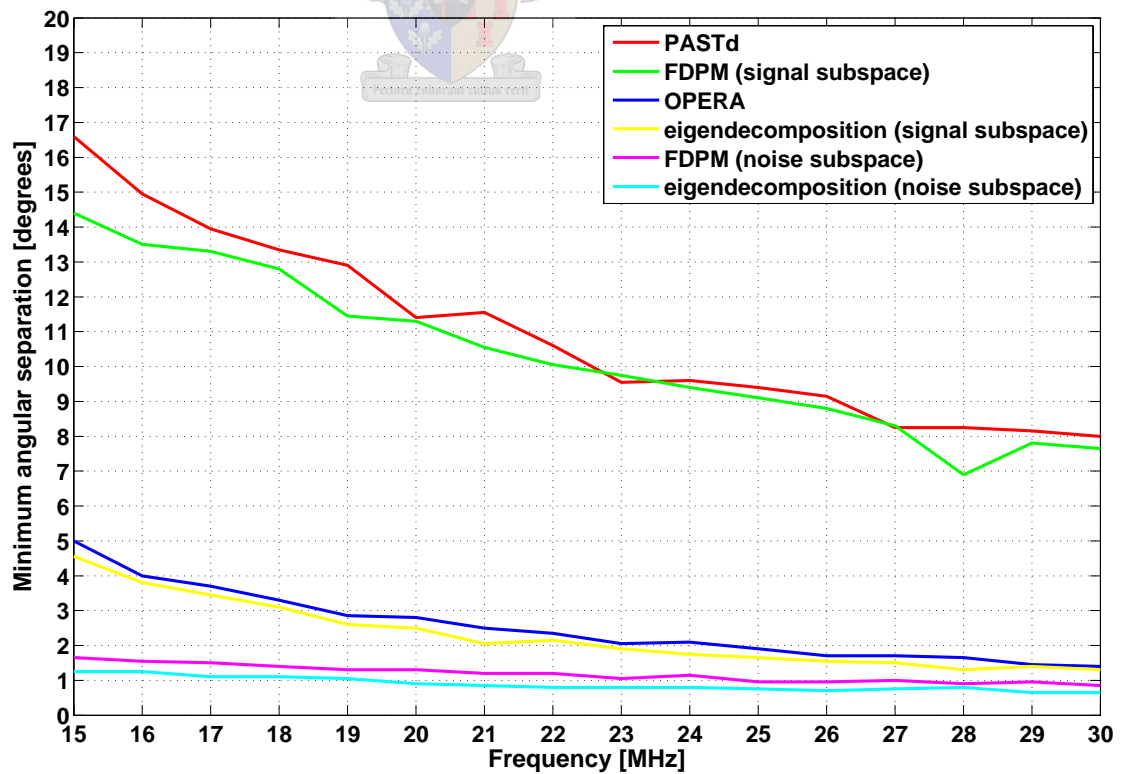
The results for the selected subspace tracking methods are plotted in figure 3.16(a). The eigen decomposition is capable of the finest angular resolution, with the ability to separate signals spaced by only 2° at 20 MHz. The results for EVD-OPERA are not much worse, with the minimum angular separation increasing by less than a degree across the frequency range. PASTd is clearly the worst at dealing with correlated signals, with a minimum angular separation of 8° at 20 MHz. The FDPM algorithm is typically capable of resolving signals with nearly 2° better angular resolution than PASTd.

The noise subspace variant of the FDPM algorithm is capable of much finer angular discrimination than any of the signal subspace algorithms. However, it does not outperform the results produced using the eigen decomposition and noise subspace.

To prove a pattern, the simulated antenna geometry was changed to that of a logarithmically spaced L-array (with elements at 8, 20 and 50 metres from the origin). The frequency range of operation was also changed to between 15 and 30 MHz. The results are plotted in figure 3.16(b). No significantly different trends are visible in comparison to figure 3.16(a), although the difference between the FDPM and PASTd algorithms is not quite as large.



(a) 5-element circular array



(b) 7-element logarithmically spaced L-array

Figure 3.16: Minimum angular separation as a function of frequency

3.3.5 Conclusions regarding performance of the selected algorithms

The signal subspace methods present similar results when tracking a signal source that makes a step change in direction-of-arrival. However, the OPERA algorithm exhibits the most lag for identical values of β . The FDPM algorithm takes the most time to converge initially. The FDPM noise subspace variant displays much finer resolution in azimuth, but is also much more sensitive to the influence of noise.

More differences between the algorithms become evident when tracking several signals that make step changes in direction-of-arrival. The PASTd algorithm displays slightly more sensitivity to noise than the eigen decomposition. The responses produced by the OPERA algorithm closely resemble those of the reference eigen decomposition, although they exhibit more lag for the same values of β .

The OPERA tracking method has the best performance (of the methods considered) under very low SNR conditions. At very low values of SNR, PASTd performs better than FDPM, although the converse becomes true as the SNR improves.

OPERA is also best at resolving very closely spaced signals, followed by FDPM and then PASTd.

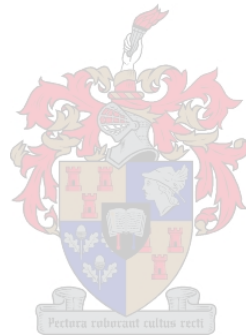
The subspace tracking algorithms are ranked below in terms of the parameters examined:

Sensitivity to noise (least sensitive to most sensitive)	
1.	OPERA
2.	PASTd
3.	FDPM

Speed of initial convergence (fastest to slowest)	
1.	OPERA
2.	PASTd
3.	FDPM

Lag for identical values of β (least to most)	
1.	PASTd
2.	FDPM
3.	OPERA

Minimum angular separation of signals (smallest to largest)	
1.	OPERA
2.	FDPM
3.	PASTd



3.4 Efficiency of the selected subspace tracking algorithms

If a subspace tracking algorithm is to be useful for practical direction finding, it must provide eigen decomposition results at an adequate rate. The minimum acceptable rate is usually determined by the application, although most practical applications will require several results a second.

If the signal environment consists of many stationary signals and the antenna array is a permanent installation, a DF rate of one result per second would be acceptable. A much higher DF rate would be required if the antenna array is mobile (e.g. a ship-borne installation) and the targets are close and moving rapidly (e.g. fighter planes). For example, a jet plane 1 kilometre away travelling at Mach 1 changes azimuth by 20° every second. Each DF calculation should take no longer than 50 milliseconds under such a circumstance for an overall DF accuracy better than 1° . The requirements are even more stringent for agile and hopping signals, which may change frequency hundreds of times a second.

Efficiency may be characterised using two complementary approaches: Firstly, a complexity analysis analyses the behaviour of an algorithm mathematically as one or more parameters vary. The end result is the **order** of the algorithm - an asymptotic limit to the number of calculations required. This is useful for determining whether the algorithm may be applied practically to large problems as well as small problems. It does **not** give an indication of exactly how long a particular algorithm will take on a given hardware platform. The second approach is that of **benchmarking**. Benchmarking is the creation of a simulated problem on an actual hardware platform. Timing measurements may then be extrapolated to determine whether the algorithm will be of practical use.

Benchmarking is frequently more useful than complexity analysis, because the end result consists of wall-clock times that may be related easily to the particular application. When the target platform is a microprocessor or PC, benchmarking also includes non-linear effects that may not readily be incorporated in a complexity analysis, such as:

1. data caching
2. instruction pipelining and the penalty of pipeline stalls
3. branch prediction
4. dynamic memory allocation
5. memory paging
6. hardware vector processing (e.g. SIMD - Single Instruction Multiple Data or SSE - Streaming SIMD Extensions)

The subspace tracking algorithms presented have been benchmarked on a typical desktop PC. Implementation using an FPGA (Field Programmable Gate Array) or DSP (Digital Signal Processor) was not considered for this thesis, owing to the complex nature of the algorithms, although future work may investigate these possibilities.

In addition, the theoretical complexity of each algorithm is presented, to allow extrapolation of the benchmark results as the problem dimensions are changed.

3.4.1 The benchmark program

The target platform for the benchmark was an Intel® Pentium 4 desktop PC with the following specifications:

Table 3.1: *Subspace tracking benchmarks: target platform*

CPU	Intel® Pentium 4 (Prescott-2M)
Level 1 cache	16 KB
Level 2 cache	2048 KB
System RAM	2 GB DDR2-533
Pagefile	2046 MB
Hyperthreading	enabled

The PASTd, FDPm and EVD-OPERA algorithms were rewritten in Standard C++ for the benchmark program¹. The Intel® C++ Compiler (version 9.1), a state-of-the-art optimising compiler, was used to produce the benchmark executables. The following compiler optimisation switches were enabled:

1. /O2 - general optimisations
2. /O3 - more aggressive optimisations that may not improve performance for all programs
3. /Ob1 - Inline functions declared with `__inline`, and perform C++ inlining
4. /Ob2 - Inline any function, at the compiler's discretion
5. /QxP - Code is optimised for Pentium 4 with SSE3 support

Timing was performed using the high-resolution Windows performance counter (functions `QueryPerformanceFrequency` and `QueryPerformanceCounter`). The scheduling priority of the benchmark process was raised to HIGH for each program run, to avoid corruption of the timing results by background services and other programs.

¹Source code is available from the author on request.

The benchmark program created simulated data for processing. The following input parameters could be set:

- **M**: the number of antenna elements (rows)
- **N**: the number of array snapshots (columns). This was set to one million for all of the benchmark tests.
- **K**: the number of simulated signals, equal to the rank of the covariance matrix
- $\rho_1 \dots \rho_K$: the eigenvalues relating to each simulated signal
- σ : the magnitude of the uncorrelated system noise

The following algorithm was used to create the simulated data:

1. Create a random M by M matrix basis with mean 0 and variance 1

$$\mathbf{A} = \text{randn}(M, M) \quad (3.8)$$

2. Orthormalise this basis using the QR (orthogonal-triangular) decomposition

$$\mathbf{A} = \mathbf{Q}\mathbf{R} \quad (3.9)$$

3. Use the first K columns of \mathbf{Q} ($\mathbf{q}_1 \dots \mathbf{q}_k$) as a basis for the covariance matrix, \mathbf{R}

$$\mathbf{R} = \sum_{k=1}^K \rho_k \mathbf{q}_k \mathbf{q}_k^H \quad (3.10)$$

where the ρ_k are the eigenvalues of \mathbf{R} (supplied as input parameters to the program).

4. Ensure that \mathbf{R} is positive definite by replacing each element on the diagonal of \mathbf{R} by its absolute value.
5. Add a small scalar σ representing the system “noise” to each element on the diagonal of \mathbf{R} .
6. Factorise this synthetic covariance matrix via a Cholesky decomposition

$$\mathbf{R} = \mathbf{L}\mathbf{L}^H \quad (3.11)$$

where \mathbf{L} is lower triangular.

7. Generate an M by N matrix of random samples with mean 0 and variance 1.

$$\mathbf{B} = \text{randn}(M, N) \quad (3.12)$$

8. Pre-multiply \mathbf{B} by \mathbf{L} .

$$\mathbf{X} = \mathbf{L}\mathbf{B} \quad (3.13)$$

The estimated covariance of the simulated sample \mathbf{X} is now equal to \mathbf{R} .

Benchmark results

The time in seconds taken to process one million data snapshots is tabulated as a function of the number of antenna elements. Each entry represents the averaged result of 5 program runs. Separate tables are generated as the dimension of the signal subspace (number of significant signals) is varied from one to five. Five was chosen as the maximum number of simultaneous signals, although it is rare in practice to encounter more than two or three signals at the same frequency. The number of antenna elements varies from five (a minimum for unambiguous results in practice) to twenty. Typical HF direction finding installations do not usually exceed nine elements, although an upper bound of twenty is used to examine the algorithms' behaviours in the limit. The tabulated data is plotted in the following section.

Table 3.2: Time [s] to calculate one million subspace tracking updates: one signal

No. elements	PASTd	FDPM	EVD-OPERA	eigen decomposition
5	0.2818	1.9522	6.6661	21.6222
6	0.3054	2.2347	6.9479	27.0909
7	0.3192	2.4574	7.0577	30.1659
8	0.3411	2.7294	7.2616	34.4619
9	0.3636	3.1182	7.4154	41.6027
10	0.3891	3.3775	7.6047	47.5862
11	0.4031	3.7281	7.8194	56.5084
12	0.4580	3.9959	8.0557	61.7837
13	0.4518	4.3451	8.1760	73.0868
14	0.4668	4.5481	8.4127	81.6662
15	0.4890	4.8358	8.5607	95.5159
16	0.5134	5.1614	8.7626	102.0600
17	0.5307	5.3074	9.0059	118.8810
18	0.5615	5.8460	9.1288	128.5250
19	0.6025	6.0363	9.3404	150.0790
20	0.6290	6.3071	9.5792	160.2680

Table 3.3: Time [s] to calculate one million subspace tracking updates: two signals

No. elements	PASTd	FDPM	EVD-OPERA
5	0.5775	3.1414	14.7458
6	0.6127	3.5794	14.9101
7	0.6392	4.0055	15.2023
8	0.6781	4.4796	15.4223
9	0.7185	4.9267	15.5565
10	0.7586	5.3612	15.8130
11	0.7902	5.8582	16.0888
12	0.8463	6.2737	16.2454
13	0.8859	6.6970	16.4763
14	0.9094	7.1263	16.7151
15	0.9516	7.6036	16.9257
16	0.9918	8.0866	17.1087
17	1.0290	8.4692	17.4339
18	1.0946	8.9307	17.6630
19	1.1781	9.4066	17.8601
20	1.2179	9.8678	18.1714

Table 3.4: Time [s] to calculate one million subspace tracking updates: three signals

No. elements	PASTd	FDPM	EVD-OPERA
5	0.8350	4.2552	19.5361
6	0.9209	4.8400	19.8553
7	0.9530	5.4643	20.1503
8	1.0032	6.0887	20.4419
9	1.0740	6.6580	20.6475
10	1.1179	7.2741	20.8817
11	1.1746	7.8234	21.1747
12	1.2527	8.4328	21.4653
13	1.2843	9.0111	21.7015
14	1.3418	9.6457	21.9833
15	1.4015	10.1948	22.1659
16	1.4664	10.8474	22.3997
17	1.5164	11.3624	22.5822
18	1.5801	11.9820	22.8432
19	1.7187	12.6698	23.1053
20	1.7645	13.2828	23.4828

Table 3.5: *Time [s] to calculate one million subspace tracking updates: four signals*

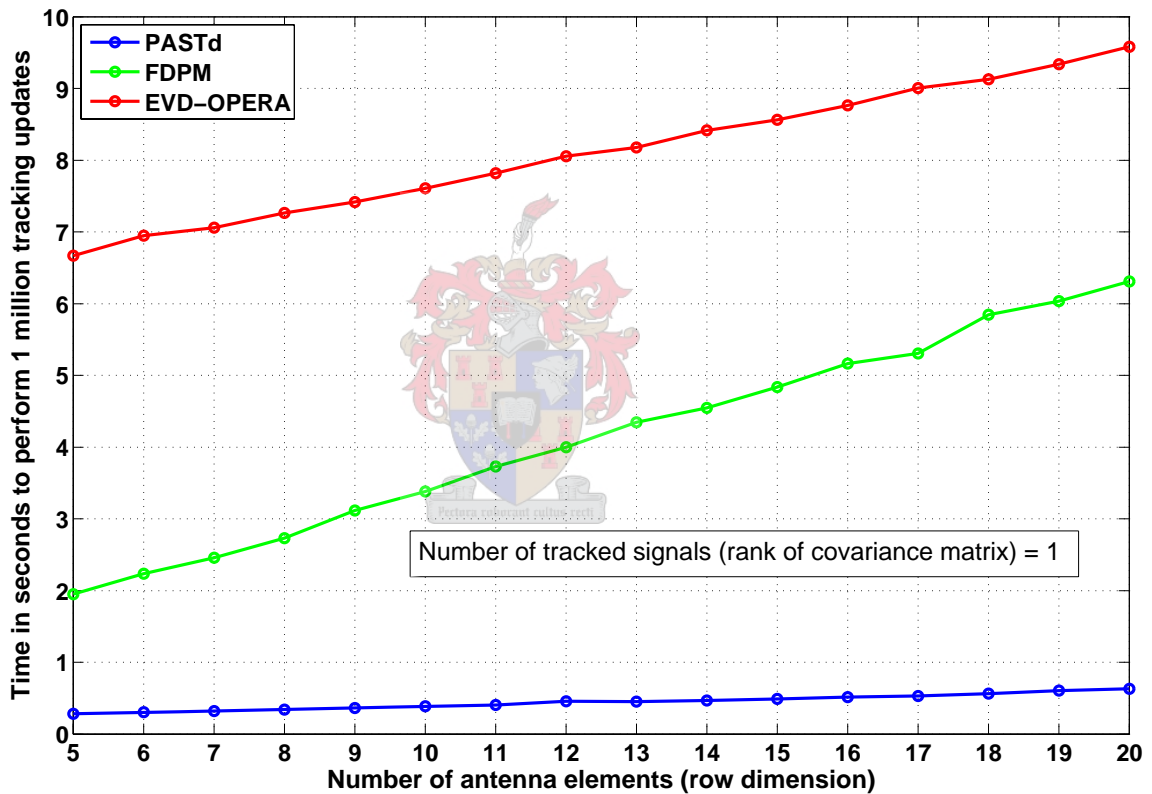
No. elements	PASTd	FDPM	EVD-OPERA
5	1.1128	5.3557	24.8100
6	1.2281	6.1256	25.2264
7	1.2648	6.8950	25.4296
8	1.3340	7.6300	25.8112
9	1.4295	8.3654	26.1719
10	1.4808	9.0988	26.4298
11	1.5563	9.8580	26.7343
12	1.6294	10.6092	27.0070
13	1.7046	11.3631	27.3649
14	1.7799	12.1178	27.7449
15	1.8770	12.8034	28.0188
16	1.9641	13.6094	28.2933
17	2.0136	14.2911	28.6674
18	2.1133	15.1118	28.9258
19	2.2668	15.9235	29.1411
20	2.3342	16.7658	29.5176

Table 3.6: *Time [s] to calculate one million subspace tracking updates: five signals*

No. elements	PASTd	FDPM	EVD-OPERA
5	1.4035	6.4109	30.1512
6	1.5290	7.3808	30.2756
7	1.5809	8.2119	30.8890
8	1.6625	9.1937	31.0134
9	1.7773	10.1040	31.5436
10	1.8434	10.9649	31.8268
11	1.9400	11.9745	32.5000
12	2.0290	12.7854	32.7042
13	2.1156	13.6899	32.4874
14	2.2045	14.6510	32.8026
15	2.3090	15.4165	33.5414
16	2.4137	16.3780	33.7660
17	2.4833	17.3988	34.0658
18	2.6161	18.2287	34.4918
19	2.8227	19.3441	34.5386
20	2.9158	20.1710	34.8894

Figure 3.17(a) shows the total time to perform one million subspace tracking updates using the FDPM, PASTd and EVD-OPERA algorithms, when only one signal is present (table 3.2). PASTd is clearly the fastest algorithm and does not exceed one second for any of the runs. PASTd takes 0.281806 seconds to perform one million updates when five elements are used and one signal is present. This represents in excess of 3 000 000 tracking updates per second (if FFT processing is performed, each bin would require a separate tracking update). Under the same conditions, FDPM can perform roughly 500 000 updates per second and OPERA, 150 000.

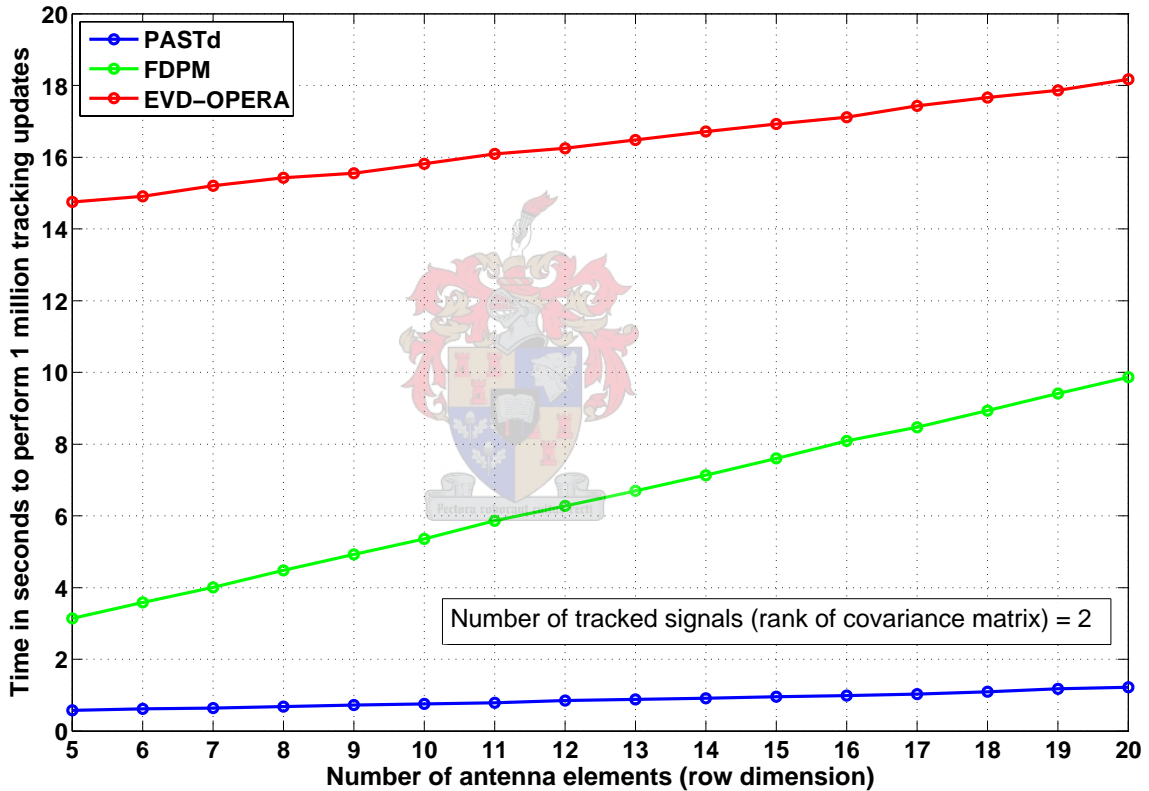
Figure 3.17: Execution time of tracking algorithms versus number of elements



(a) One signal

Figure 3.17(b) shows benchmark timing results for the case of two signals. The PASTd times appear to double, compared to the one signal case. The FDPM times appear to be multiplied by approximately 1.6, which indicates that there is a constant amount of set-up time that becomes less significant as the number of signals increases. The OPERA time for five elements is multiplied by approximately 2.2, whereas for twenty elements, the time is multiplied by approximately 1.9. This indicates that at least some portion of the time spent by the OPERA algorithm is not proportional to the number of elements. This time becomes less significant as the number of elements increases.

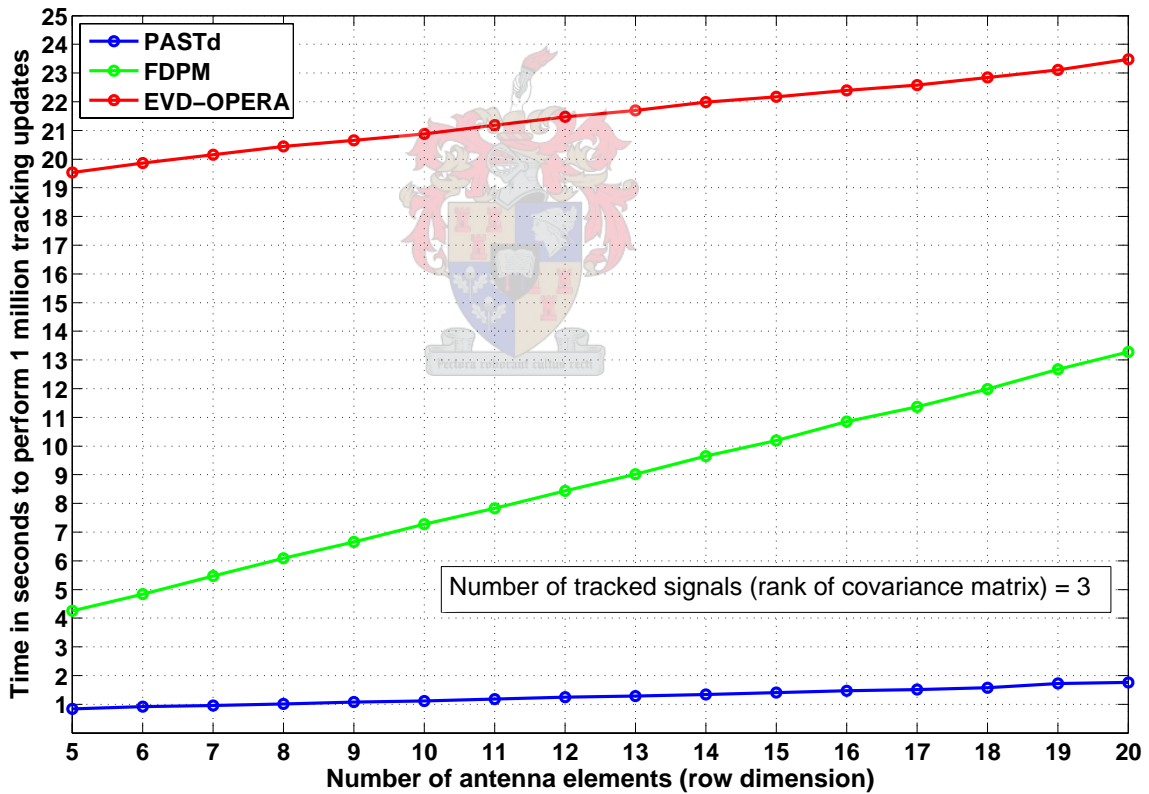
Figure 3.17: Execution time of tracking algorithms versus number of elements



(b) Two signals

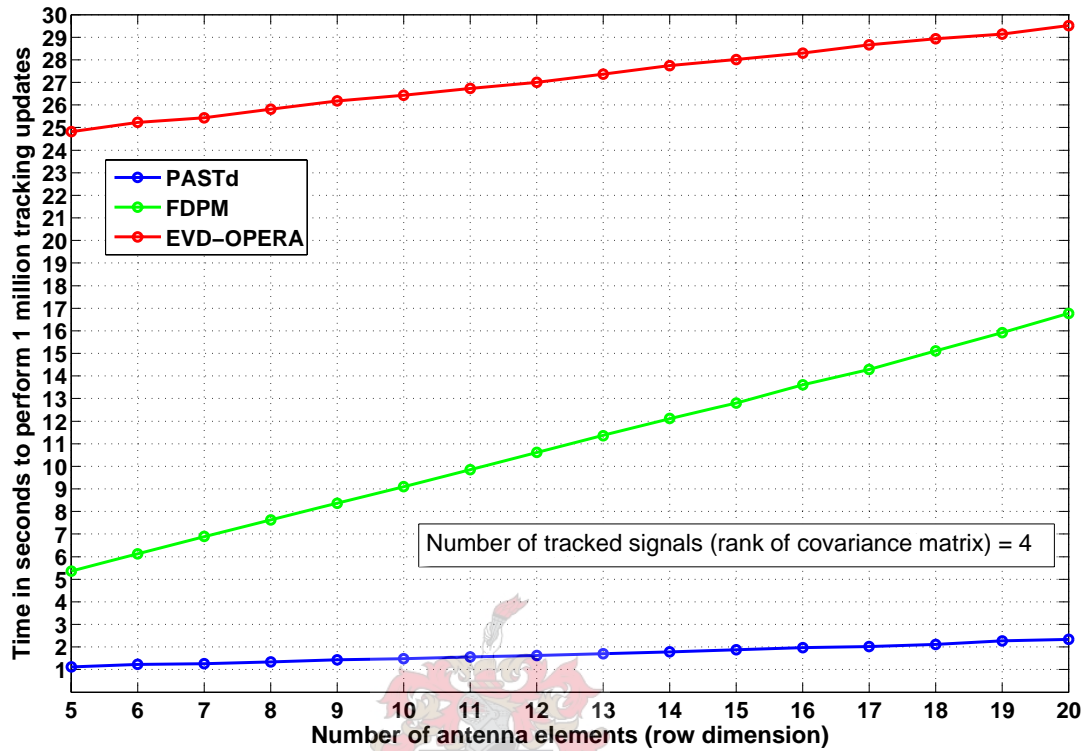
The cases of three and four signals are displayed in figures 3.17(c) and 3.17(d). Figure 3.17(e) shows the five signal case. The PASTd times for five and twenty elements are multiplied by approximately 4.6 and 5 respectively, compared to the one signal case. This suggests that there is a small amount of fixed overhead cost that is proportionally more significant in the case of five elements and one signal. For FDPM, the run times are all multiplied by approximately 3.2, compared to the one signal case - indicating that the algorithm has a moderate amount of overhead that is most significant in the one signal case, and is not a function of the number of elements. The OPERA algorithm exhibits the most variation: times are multiplied by 4.5 (five elements) and 3.6 (twenty elements), compared to the one signal case. These results suggest that OPERA has high overhead costs that becomes proportionally large in the cases of few elements and few signals.

Figure 3.17: Execution time of tracking algorithms versus number of elements (cont'd)

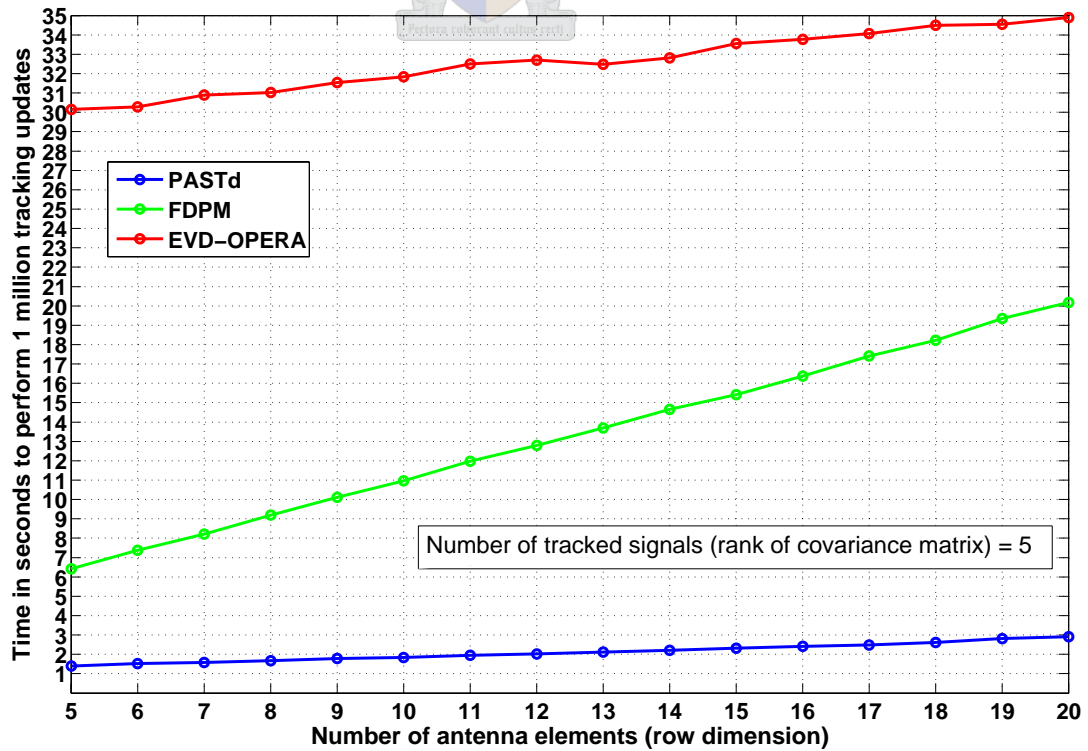


(c) Three signals

Figure 3.17: Execution time of tracking algorithms versus number of elements (cont'd)



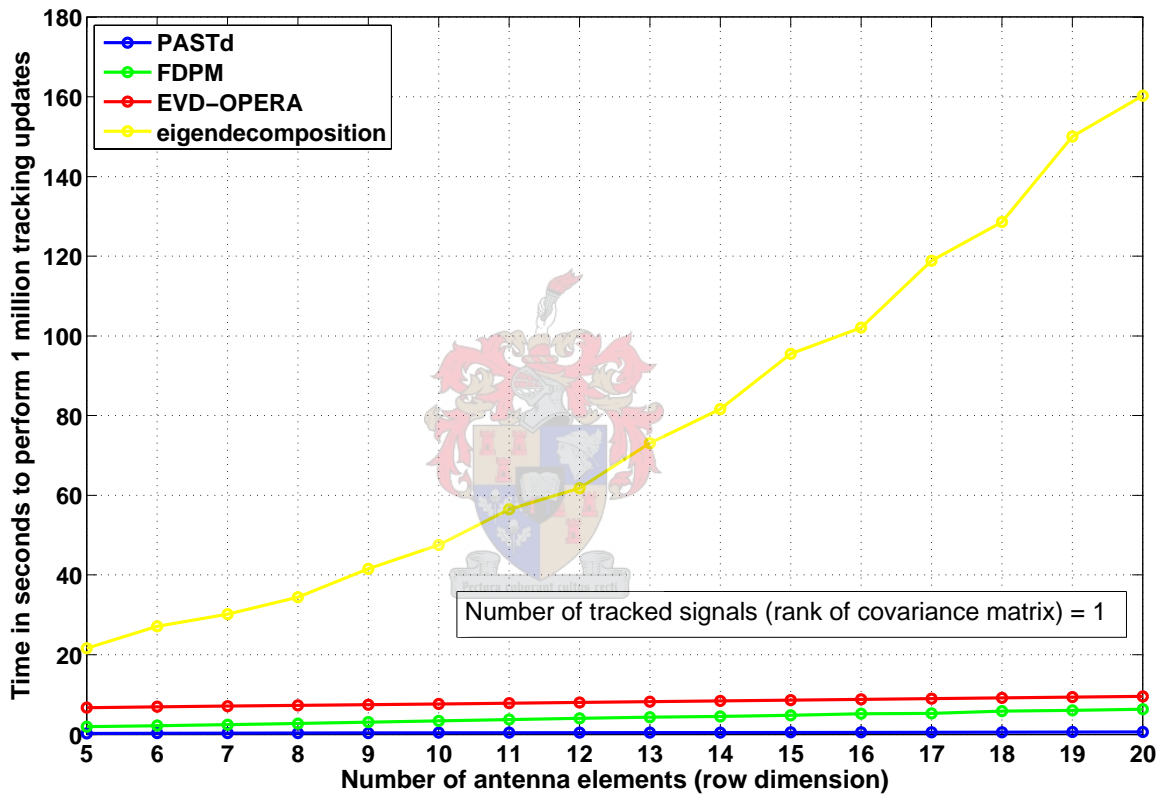
(d) Four signals



(e) Five signals

Figure 3.18 gives an idea of the improvement of the three subspace tracking algorithms over the standard eigen decomposition. This plot is identical to figure 3.17(a), although the time taken by the eigen decomposition (using rank-one updates of the covariance matrix) is indicated in yellow. The exponential increase in complexity makes it quite clear why approximate tracking algorithms are necessary to produce practical subspace direction finding systems. Note that the eigen decomposition used is the highly optimised version supplied by the Intel® Math Kernel Library (the function name is `cheev`).

Figure 3.18: Execution time of tracking algorithms compared to the eigen decomposition



3.4.2 Complexity analysis

Each algorithm is presented in terms of the number of multiply-accumulate (MAC) operations required.

The multiplication of two complex scalars, $x = a + ib$ and $y = c + id$ may be performed using 4 real multiplications and 2 real additions:

$$xy = (ac - bd) + i(ad + bc) \quad (3.14)$$

or 3 real multiplications and 5 real additions:

$$xy = (ac - bd) + i((a + b)(c + d) - ac - bd) \quad (3.15)$$

assuming that the products ac and bd are stored as temporaries during the calculation.

A single complex multiply-accumulate operation thus requires a minimum of 3 real multiplications and 7 real additions. Floating point divisions are more expensive than multiplications, and the square root operation is still more expensive if full precision is needed.

The number of MAC operations required by the selected tracking algorithms at each iteration were counted. The MAC counts were based on the mathematical pseudocode (section 2.5) and the standard C++ benchmark program.

Computational complexity of FDPM

The FDPM algorithm requires $10MK + 2M + 4K$ complex MAC operations per iteration, where M is the number of antenna elements and K is the dimension of the signal subspace (equal to the number of eigenvectors calculated). In addition, the algorithm calculates $3K$ square roots (the normalisation steps of equation 2.66a). Full floating point precision is generally not necessary for the square root operation, although most modern CPU's have square root optimisations in hardware. For example, the Intel® Pentium 4 requires 9.4 clock cycles to calculate the square root of a 32-bit floating point number with full precision. Using slightly reduced accuracy to within 3 units in the last place (ULP), the clock cycle count per square root operation can be reduced to 4.94 [7].²

Since the $10MK$ term dominates, the FDPM algorithm has an overall complexity of $\mathcal{O}(MK)$.

Computational complexity of PASTd

The PASTd algorithm requires $5MK + 2K$ complex MAC operations at each iteration. It thus has an identical complexity to FDPM, $\mathcal{O}(MK)$, although the coefficient of the

²Note that clock cycle counts are often not integers, because of pipelining and other optimisations.

MK term is smaller and it requires no square root operations. These differences suggest a slight speed advantage over the FDPM algorithm in practice - which is supported by the benchmarking results of the previous section.

Computational complexity of OPERA

At each iteration, the EVD-OPERA algorithm requires $MK^2 + 2MK + 3M + 3K^2 + K^3$ complex MAC operations. The K^3 term represents the complexity of the reduced size eigen decomposition in the transformed coordinate system. The most significant term is MK^2 , which represents the projection of the rank-one update to the eigen decomposition back to the original coordinate system. The algorithm has an overall complexity of $\mathcal{O}(MK^2)$. This is somewhat higher than the FDPM and PASTd algorithms, although the algorithm may be relatively efficient when only a few dominant eigenvectors are required. The benchmarking results support the conclusion that the OPERA algorithm should be less efficient than FDPM and PASTd in practice.

3.4.3 Conclusions regarding efficiency of the selected algorithms

FDPM and PASTd are theoretically of equal complexity, $\mathcal{O}(MK)$. Both algorithms scale well as the number of antenna elements and significant signals are varied. PASTd may be slightly faster in practice, particularly if efficient square root operations are not available on the target platform. OPERA has a complexity of $\mathcal{O}(MK^2)$. This indicates that it scales well as the number of antenna elements is increased, although it becomes quadratically less efficient as the number of significant signals is increased. In practice, it is slower than both FDPM and PASTd, even for small problem sizes.

The subspace tracking algorithms are ranked in terms of efficiency below:

Computational efficiency	
1.	PASTd
2.	FDPM
3.	OPERA

Chapter 4

Performance of subspace tracking algorithms using real-world data

4.1 Equipment and test setup

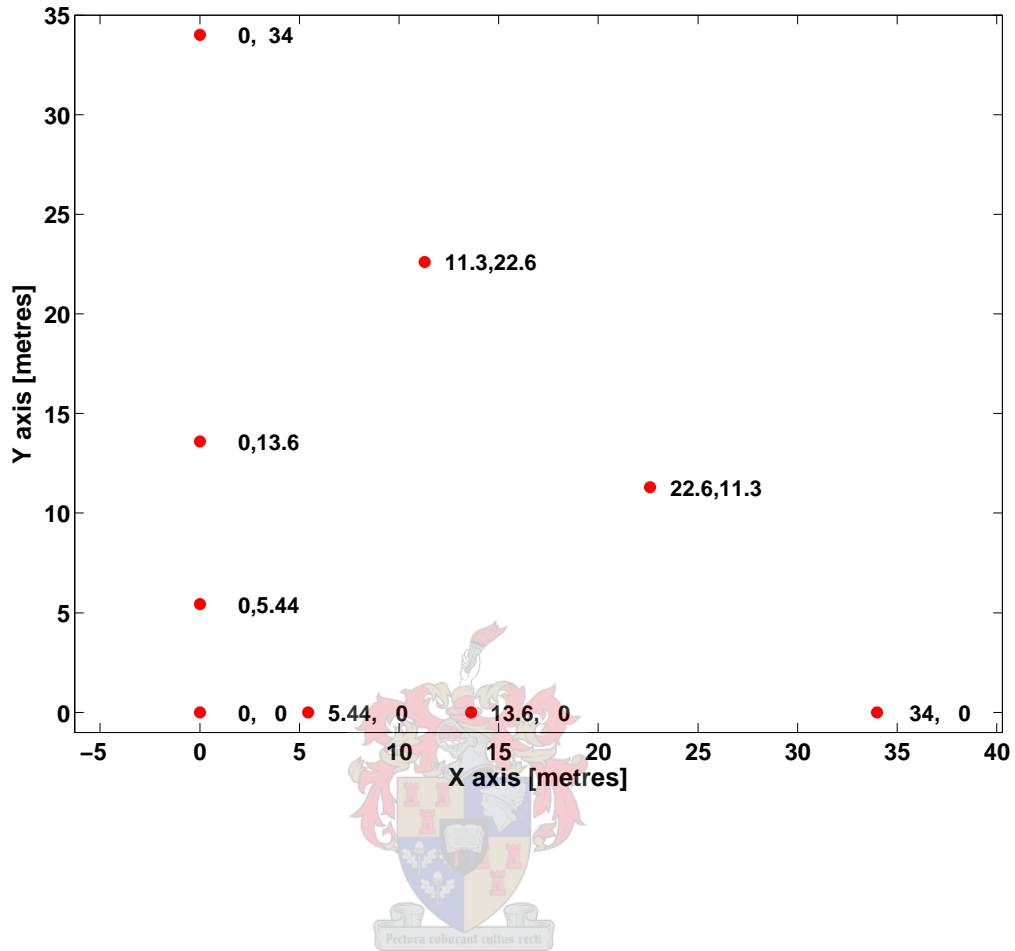
The Superresolution Direction Finding Client (section 3.2) was used to process data gathered by an actual antenna array in Pretoria. Test signals were used to assess the field performance of the subspace direction-finding algorithms. All sets of data were gathered at the Grintek Ewation building in Pretoria on 25 and 26 July, 2006. An array of nine monopole antenna elements was arranged in the field behind the Grintek Ewation building. The antenna array configuration consisted of two logarithmically spaced L arms and two elements on the diagonal between the ends of the L. The x- and y-coordinates of the elements are listed in table 4.1.

Table 4.1: *Subspace tracking DF tests: antenna array coordinates [metres]*

Element no.	x-coordinate	y-coordinate
1	0.0	0.0
2	5.44	0.0
3	0.0	5.44
4	13.6	0.0
5	0.0	13.6
6	34.0	0.0
7	0.0	34.0
8	22.66	11.33
9	11.33	22.66

Figure 4.1 depicts the layout of the antenna array.

Figure 4.1: Subspace tracking DF tests: antenna array layout



The particular antenna elements used are usually assembled as a Watson Watt Adcock array (MRCM MRA2004), although nine such elements were obtained and configured as above for the tests. Each antenna element had the profile shown in figure 4.2. Figure 4.3 shows two of the antenna elements in the field. The element to the left is the reference element ($x = 0$ and $y = 0$).

Figure 4.2: Profile of each antenna element (MRA2004), shown upside-down

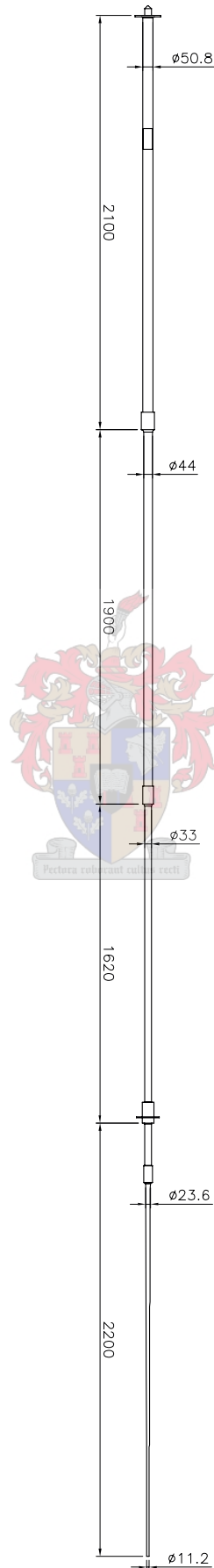


Figure 4.3: *A picture of two of the monopole antenna elements in the field*



Ground straps were connected to each element (effectively turning it into a monopole), although the ground was very dry and dusty, following a recent fire. Figure 4.4 shows the ground straps at the base of a particular monopole element.

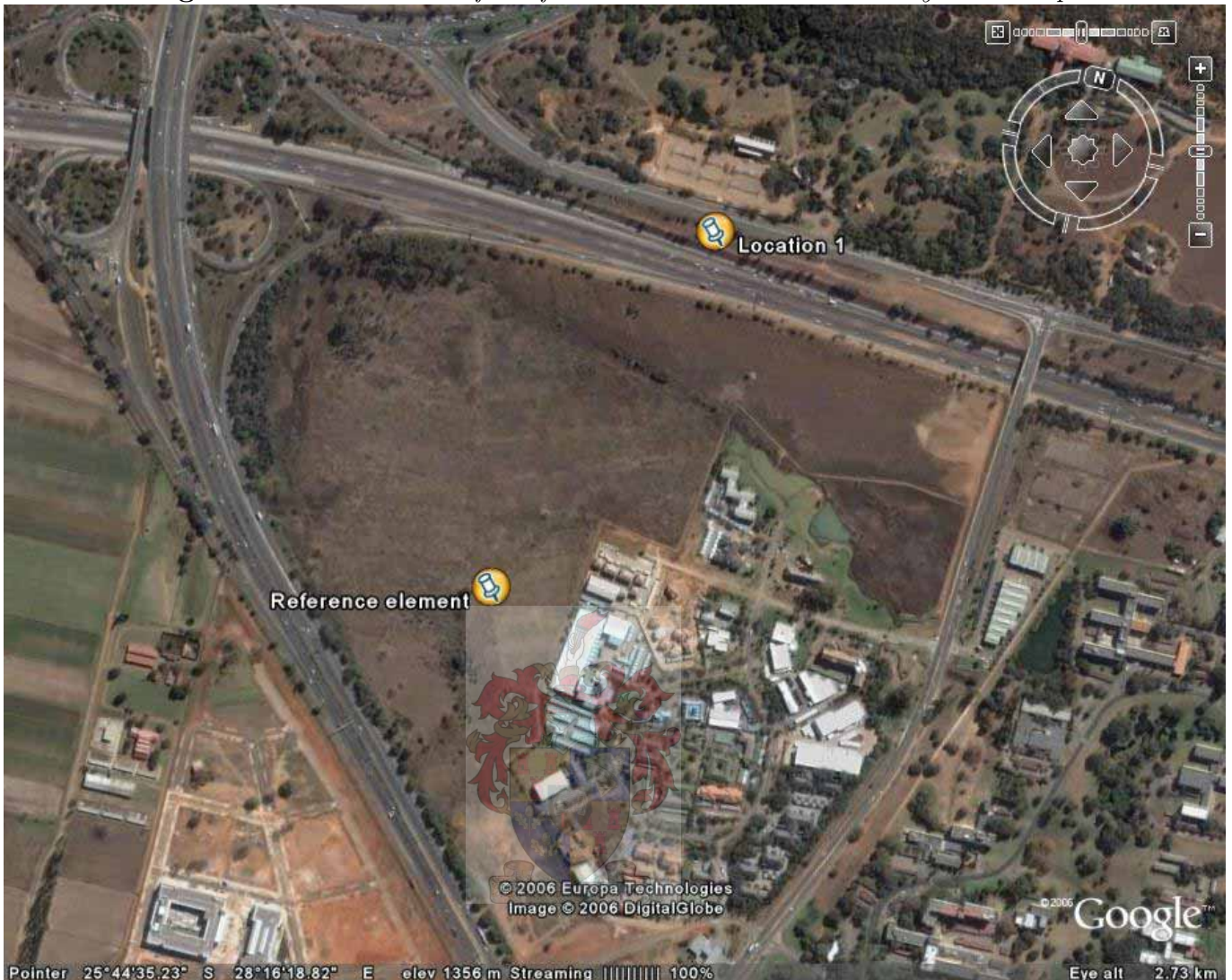
Figure 4.4: *Base of a monopole element, showing the ground straps used*



The latitude and longitude of the reference element ($x = 0$ and $y = 0$) were determined to be $S25.74450^\circ$ and $E28.27059^\circ$ respectively. The altitude was approximately 1372 metres above sea level. The deviation from magnetic north was measured as 68.5° using a surveyor's compass. The declination from magnetic north was determined to be -17.35° , using the online service provided by the U.S. National Geophysical Data Center [20]. Any azimuth calculations by the system were thus subject to a correction constant of $+51.15^\circ$.

Figure 4.5 shows the field in which the antenna array was set up [5]. The location of the reference element is marked. The Grintek Ewation building is located to the south-east of the reference element. The first test location (right next to the Pretoria National Botanical Garden) is visible in this view.

Figure 4.5: Aerial view of the field in which the antenna array was set up



Five locations were selected for test measurements. Signals were broadcast from these locations and received by the antenna array. The sites were chosen for relatively unobstructed paths to the antenna array and ease of access by road. The surrounding area consisted of a fairly densely populated urban environment. Although there didn't appear to be many hills of much significance, there were plenty of buildings and metallic structures that may have worsened multipath propagation effects. The latitude and longitude of each site were measured using a handheld GPS. The details of the five sites are listed below in table 4.2. All locations were less than 3 km away from the antenna array. At the test frequency (24.050 MHz) this implies chiefly groundwave propagation, rather than skywave propagation. Azimuth and distance values were calculated using the WGS-84 Earth ellipsoid model [22]. "DF azimuth" indicates the azimuth after subtraction of the correction constant ($+51.15^\circ$). Figure 4.6 shows an aerial view indicating the locations used for broadcasting relative to the reference element [5].

Table 4.2: *Subspace tracking DF tests: broadcast locations*

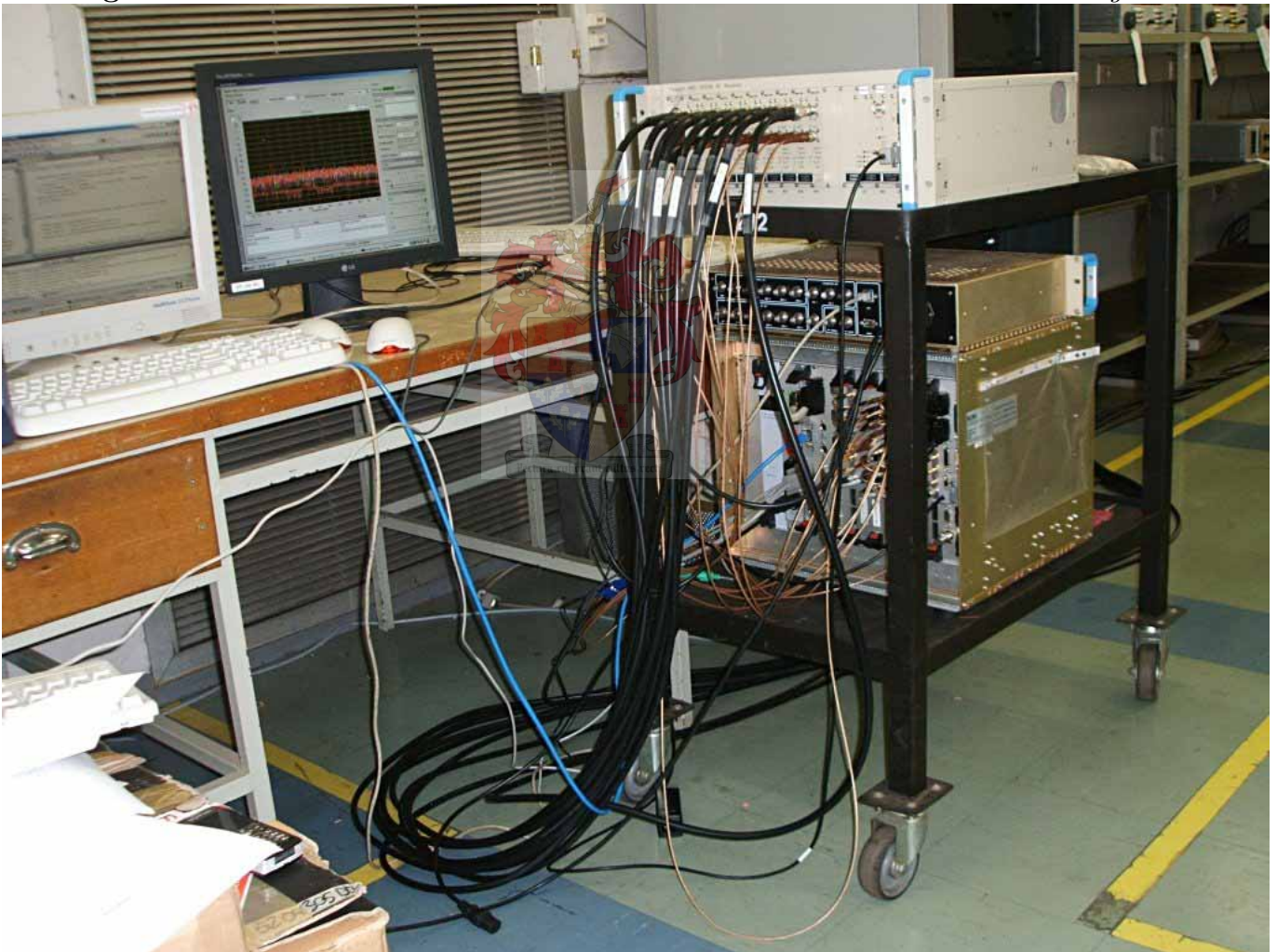
Location no.	Latitude	Longitude	Distance	True azimuth	DF azimuth
1	25.73987° S	28.27239° E	543.8 m	19.39°	327.70°
2	25.74316° S	28.24703° E	2368.5 m	273.58°	222.44°
3	25.74926° S	28.24490° E	2624.0 m	258.40°	207.25°
4	25.75466° S	28.27396° E	1175.3 m	163.28°	112.13°
5	25.74428° S	28.29368° E	2316.8 m	89.40°	38.25°

Figure 4.6: *Aerial view of the locations used for test measurements*



A 10-channel receiver (MRCM MRD5000W) was used to perform filtering and down-conversion to an IF (Intermediate Frequency) of 12.8 MHz. The IF was recorded using an MRCM MRX3000W10 Data Processing Unit. The system was capable of recording continuous IF data at a maximum bandwidth of 10 MHz (approximately 400 megabytes per second) or at a minimum bandwidth of 800 kHz (approximately 30 megabytes per second). The 800 kHz bandwidth was used for the narrowband tests. At this bandwidth, a maximum of 16 seconds of IF data could be recorded. The MRD5000W receiver and MRX3000W10 DPU are shown in figure 4.7. The receiver is on the top of the rack and the MRX3000W10 DPU is below. The cables connected to the receiver terminate at the antennas in the field.

Figure 4.7: *MRD5000W 10-channel receiver and MRX3000W10 Data Processing Unit*



4.2 Calibration of the narrowband results

The cables connecting the receiver to the antennas were disconnected at each antenna, and were connected instead to a 9-way, phase-linear splitter. The input to the splitter was connected to a white noise generator (Rohde & Schwarz SUF2), which was powered by a petrol generator in the antenna field. The receiver was tuned to 24 MHz, and the downconverted IF was recorded for a duration of 15.6 seconds. Three successive recordings were made using capture bandwidths of 800 kHz and 2 MHz (six recordings in total). The resulting samples of noise were processed using FFT transformation, averaging and inversion to produce a calibration (correction) table.

The portion of the calibration table corresponding to the narrowband frequency band used (24 MHz to 24.1 MHz) is depicted below for each channel in terms of magnitude (figure 4.8(a)) and phase (figure 4.8(b)). Note that the tables represent the required *correction*, i.e. they are the inverse of the actual response to the noise signal. Both correction tables use channel 1 (which is not displayed) as the reference. Note that the phase corrections required are on the order of tens of degrees. These deviations arise principally in the receiver channels, although the connectors and cables are also subject to mismatches. The calibration procedure is unable to compensate for mismatched antenna elements (since the calibration noise source is injected where the elements would be connected).

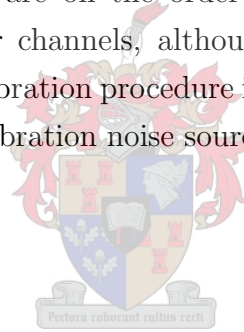
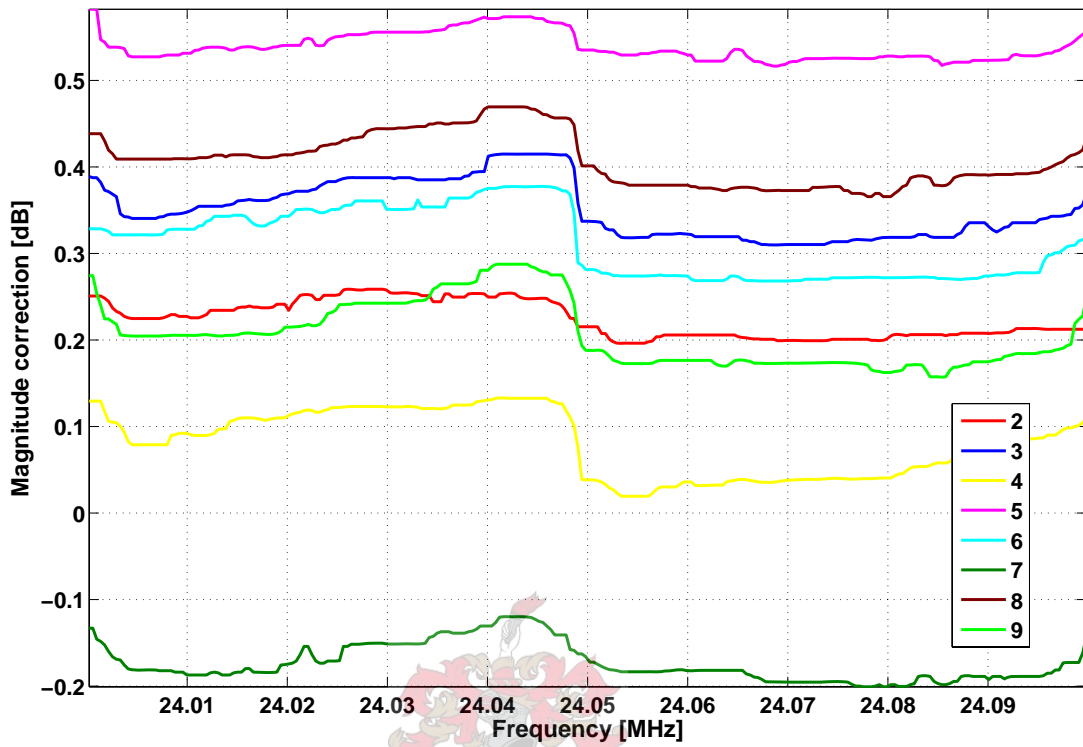
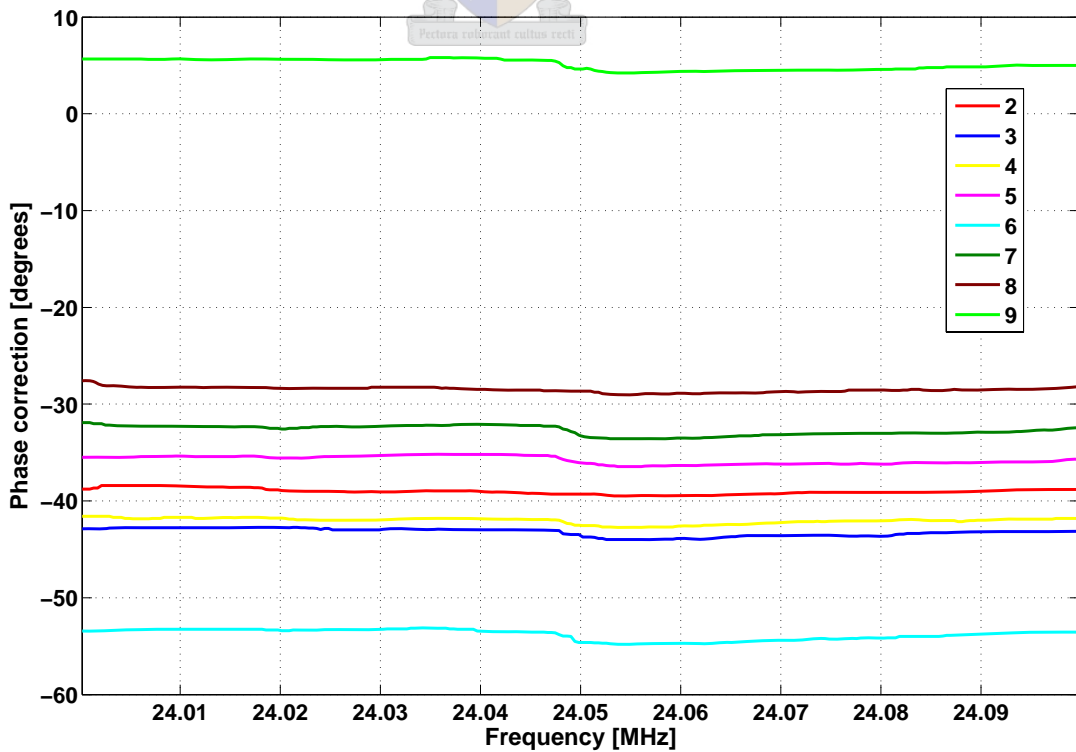


Figure 4.8: Calibration correction tables



(a) Magnitude correction table



(b) Phase correction table

4.3 CW (constant sinusoidal) signal measurements

An ICOM IC-706 transceiver was used to generate constant-wave (sinusoidal) signals at a frequency of 24.050 MHz at all of the test locations. The bearing estimates are analysed at each location, using each subspace tracking method.

Early on into the post-processing of the results, it became obvious that the mutual coupling between antenna elements was significant. In fact, many of the bearing estimates are incorrect if mutual coupling is not taken into account. For this reason, the following analyses are presented for both cases when mutual coupling is compensated, and when it is not. The coupling matrix \mathbf{C} indicates the coupling (a complex scalar) between each pair of elements, denoted by the rows and columns of the matrix. \mathbf{C} is square and symmetric with M rows and columns (where M is the number of antenna elements). The coupling matrix may be estimated as follows:

$$\mathbf{C} = (Z_A + Z_T)(\mathbf{Z} + Z_T\mathbf{I})^{-1} \quad (4.1)$$

where Z_A is the antenna impedance, Z_T is the input (termination) impedance of the measurement equipment and \mathbf{Z} is the mutual impedance matrix [19]. The mutual impedance matrix may be estimated using a Method of Moments calculation.

A computer simulation of each antenna element was formulated, using an approximation of the geometry (figure 4.2). The simulation took as parameters the lengths and radii of the antenna segments, as well as the frequency of operation. The impedance matrix for the above antenna array was then calculated using a MATLAB implementation of Gauss-Legendre quadrature integration (a type of finite element modelling) [13]. Using sixteen evaluation points per segment, the integration produced an estimate of the coupling between each pair of antenna elements. The impedance matrix was calculated as

$$\mathbf{Z} = \begin{bmatrix} 157+271i & -5-76i & -5-76i & 27+26i & 27+26i & -16-1i & -16-1i & 7+20i & 7+20i \\ -5-76i & 157+271i & -53-26i & -55-13i & 34+8i & 17-7i & -15+3i & -23-11i & -13+18i \\ -5-76i & -53-26i & 157+271i & 34+8i & -55-13i & -15+3i & 17-7i & -13+18i & -23-11i \\ 27+26i & -55-13i & 34+8i & 157+271i & -12-25i & -22-13i & -4+14i & 34+11i & -19+14i \\ 27+26i & 34+8i & -55-13i & -12-25i & 157+271i & -4+14i & -22-13i & -19+14i & 34+11i \\ -16 & 17-7i & -15+3i & -22-13i & -4+14i & 157+271i & -8+8i & 29-14i & -9-14i \\ -16 & -15+3i & 17-7i & -4+14i & -22-13i & -8+8i & 157+271i & -9-14i & 29-14i \\ 7+20i & -23-11i & -13+18i & 34+11i & -19+14i & 29-14i & -9-14i & 157+271i & 29-14i \\ 7+20i & -13+18i & -23-11i & -19+14i & 34+11i & -9-14i & 29-14i & 29-14i & 157+271i \end{bmatrix}$$

The coupling matrix could then be calculated using equation 4.1:

$$\mathbf{C} = \begin{bmatrix} 55.80+5.63i & 13.93+7.30i & 13.93+7.30i & -6.00+0.96i & -6.00+0.96i & 1.71-1.79i & 1.71-1.79i & -1.20-2.45i & -1.20-2.45i \\ 13.93+7.30i & 53.19-0.01i & 10.14-1.30i & 3.40-5.58i & -4.20+1.72i & 0.65+1.84i & 1.75-2.23i & 3.03-2.91i & -1.51-4.73i \\ 13.93+7.30i & 10.14-1.30i & 53.19-0.01i & -4.20+1.72i & 3.40-5.58i & 1.74-2.23i & 0.65+1.84i & -1.51-4.73i & 3.03-2.91i \\ -6.00+0.96i & 3.40-5.58i & -4.20+1.72i & 50.82-2.30i & 5.27+1.57i & 3.18-2.11i & -1.50-2.00i & -2.88+3.46i & -1.35-3.77i \\ -6.00+0.96i & -4.20+1.72i & 3.40-5.58i & 5.27+1.57i & 50.82-2.30i & -1.50-2.00i & 3.18-2.10i & -1.35-3.77i & -2.88+3.46i \\ 1.71-1.79i & 0.65+1.84i & 1.75-2.23i & 3.18-2.10i & -1.50-2.00i & 49.39-0.53i & -0.57-1.29i & -0.82+5.92i & 2.24-0.70i \\ 1.71-1.79i & 1.75-2.23i & 0.65+1.84i & -1.50-2.00i & 3.18-2.11i & -0.57-1.29i & 49.39-0.53i & 2.24-0.70i & -0.82+5.92i \\ -1.20-2.45i & 3.03-2.91i & -1.51-4.73i & -2.88+3.46i & -1.35-3.77i & -0.82+5.92i & 2.24-0.70i & 48.50-0.56i & -0.26+5.74i \\ -1.20-2.45i & -1.51-4.73i & 3.03-2.91i & -1.35-3.77i & -2.88+3.46i & 2.24-0.70i & -0.82+5.92i & -0.26+5.74i & 48.50-0.56i \end{bmatrix}$$

Note that there are several significant entries in the off-diagonal elements of the coupling matrix, indicating that the mutual coupling is not negligible.

The MUSIC algorithm (equation 2.46) may be modified to incorporate the estimate of the coupling matrix:

$$P_{MUSIC}(\bar{\psi}) = \frac{1}{|\mathbf{V}_N^H \mathbf{C} \mathbf{s}(\bar{\psi})|^2} \quad (4.2)$$

The memory factor β was set to 0.95 for all of the following tests. The DOA spectrum is evaluated after processing 500 snapshots using each subspace tracking algorithm.



4.3.1 Location 1

Figure 4.14 (a) shows the MUSIC spectrum for the results obtained when broadcasting from Location 1 (DF azimuth of 327.7°), using the eigen decomposition and noise subspace. There appear to be several significant maxima in the MUSIC spectrum, possibly indicating multipath propagation. The bearing is estimated as 326° when coupling is compensated (an error of less than 2°), although it is incorrectly estimated as 209° if coupling is not compensated.

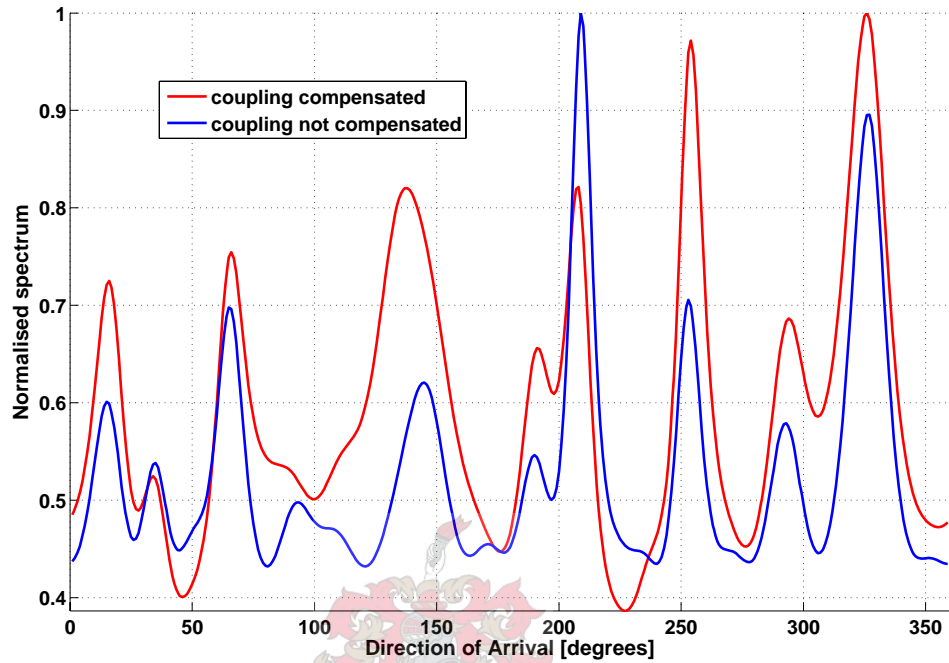
Figure 4.14(b) plots results for the same scenario as above, although the PASTd algorithm and signal subspace are used. The bearing is incorrectly estimated as 208° when compensating for coupling, and 209° when not (an error of more than 118°).

Using FDPm and the signal subspace, the bearing is incorrectly estimated as 208° (figure 4.14(c)), although a local maximum is visible at approximately 327° . Compensation for coupling does not alter the location of the global maximum. Using the noise subspace, the bearing was incorrectly estimated as 127° (figure 4.14(d)).

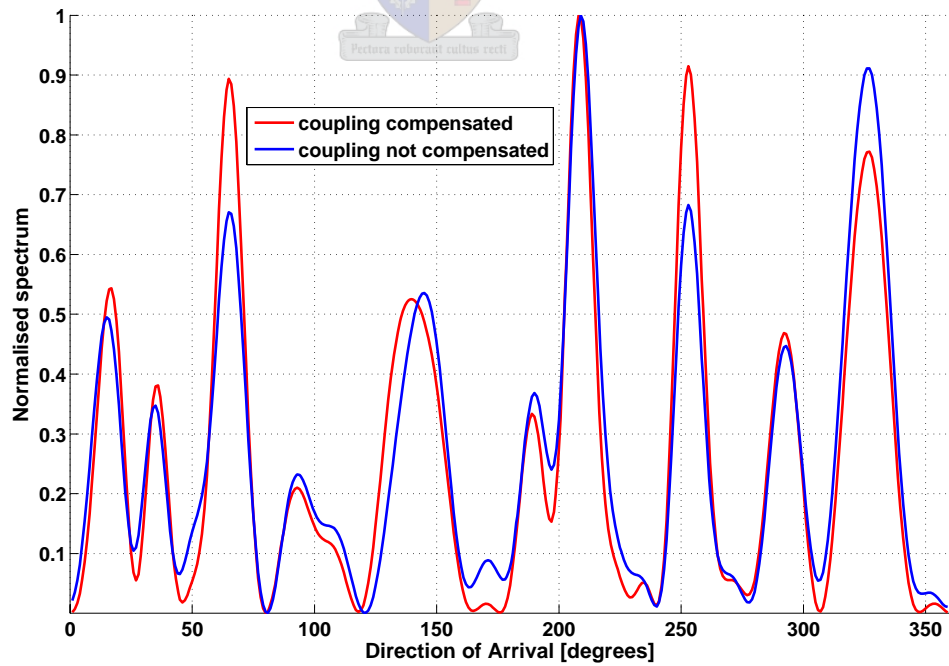
Figure 4.14(e) shows the results using the OPERA algorithm and signal subspace. The bearing is incorrectly estimated as 208° . A local maximum is visible at approximately the correct azimuth.



Figure 4.9: Location 1 (327.7°): DOA spectra

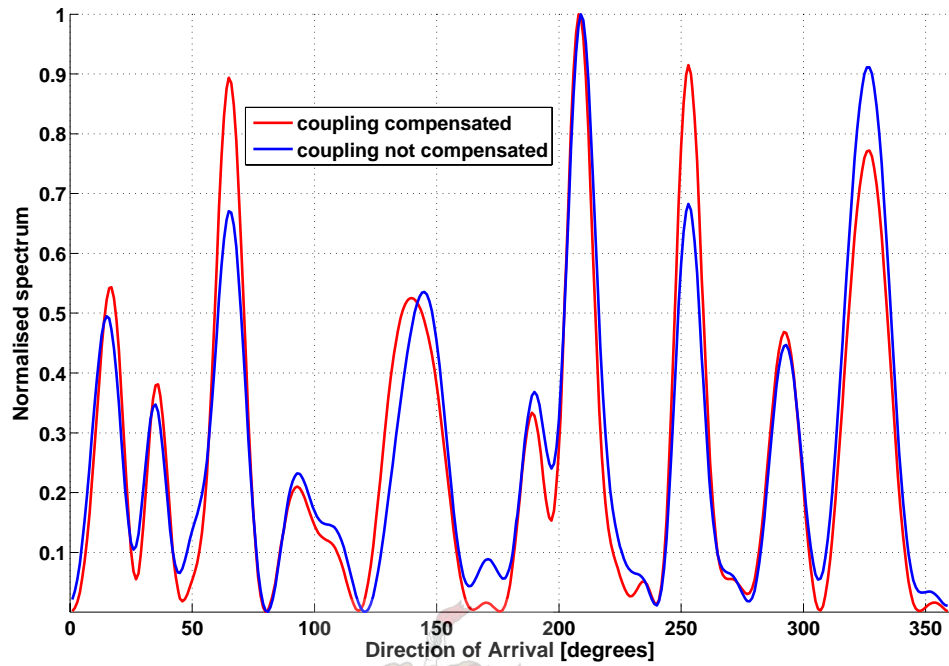


(a) Eigen decomposition, noise subspace. Azimuth peak = 326° .

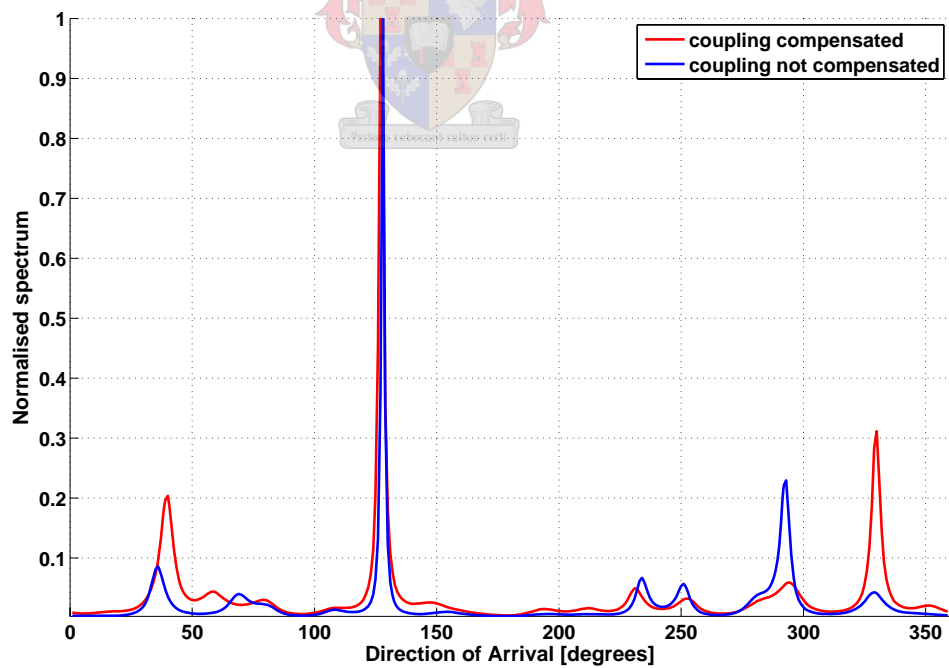


(b) PASTd, signal subspace. Azimuth peak = 208°

Figure 4.9: Location 1 (327.7°): DOA spectra (cont'd)

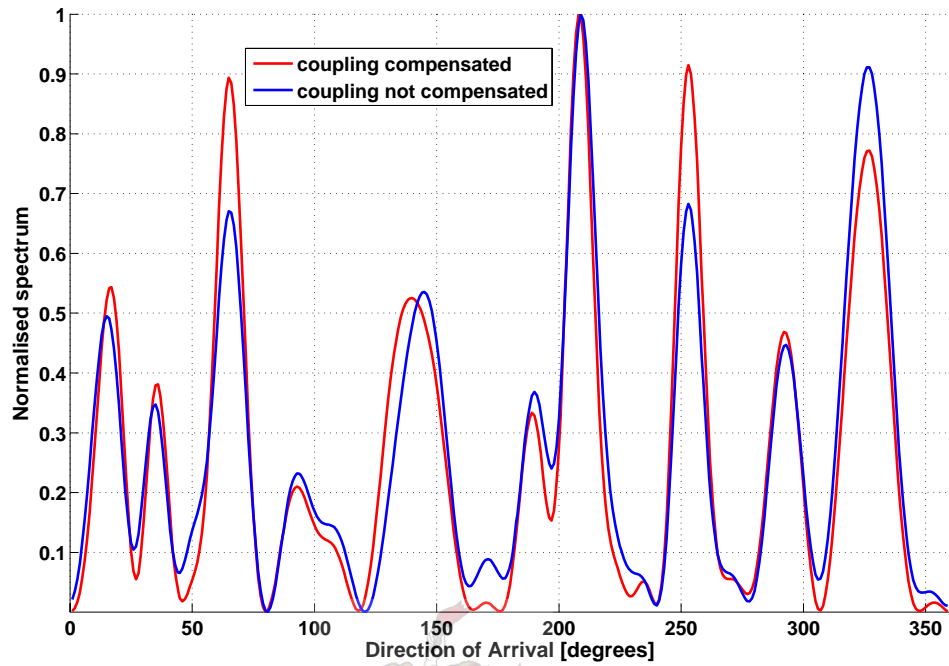


(c) FDPM, signal subspace. Azimuth peak = 208°

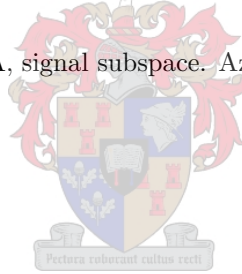


(d) FDPM, noise subspace. Azimuth peak = 127° .

Figure 4.9: Location 1 (327.7°): DOA spectra (cont'd)



(e) OPERA, signal subspace. Azimuth peak = 208° .



4.3.2 Location 2

Figure 4.14 (a) shows the MUSIC spectrum for the results obtained when broadcasting from Location 2 (DF azimuth of 222.4°), using the eigen decomposition and noise subspace. The bearing is estimated as 224° (an error of 2°), and compensation for coupling has negligible effect on the global maximum.

Figure 4.15(b) plots results using the PASTd algorithm and signal subspace. The bearing is estimated as 224° . Compensation for coupling has little effect on the DOA spectrum.

Using FDPD and the signal subspace, the bearing is also estimated as 224° (figure 4.15(c)). Coupling compensation does not seem necessary. Using the noise subspace, the FDPD algorithm estimates the bearing as 226° , an error of 4° (figure 4.15(d)). Compensating for mutual coupling alters the DOA spectrum significantly. Without such compensation, the bearing is incorrectly estimated as 33° (an error of 189°).

Figure 4.15(e) shows the results using the OPERA algorithm and signal subspace. The bearing is estimated as 208° (an error of approximately 14°). Compensation for coupling appears to cause secondary peaks (ambiguities) to arise at 60° and 252° .

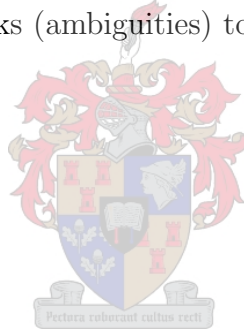
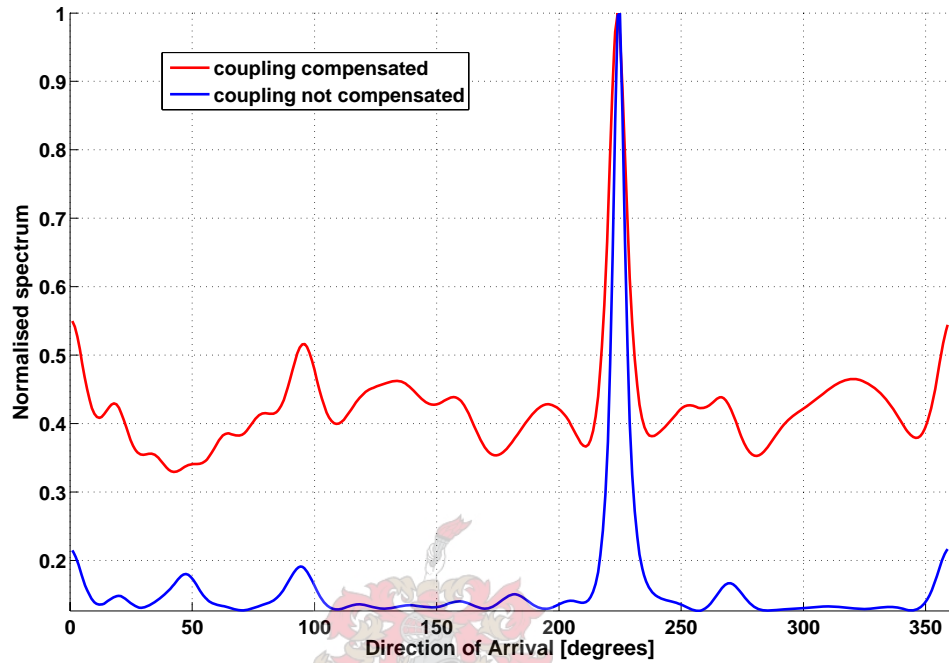
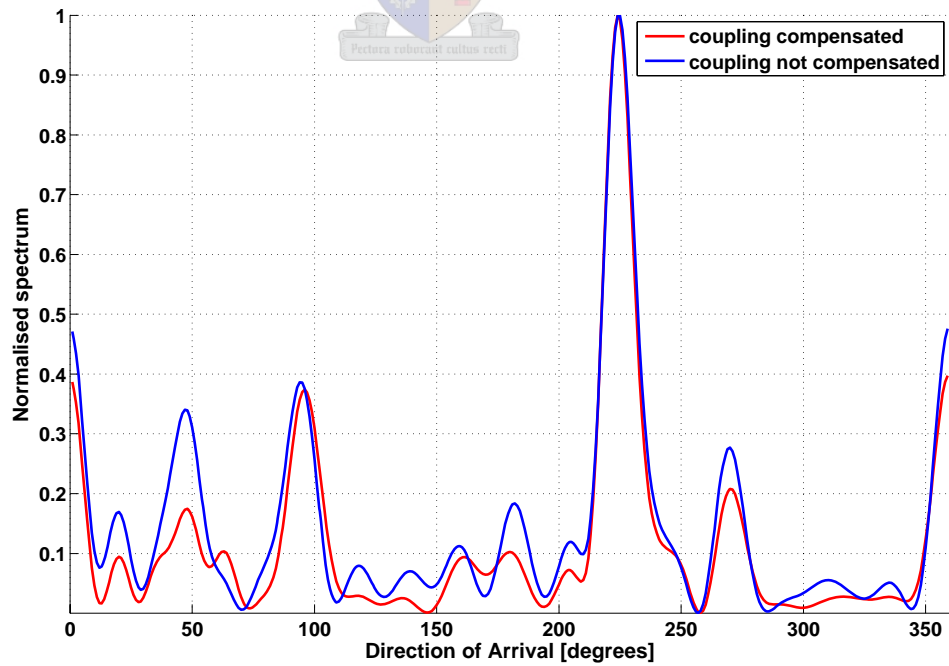


Figure 4.10: Location 2 (222.4°): DOA spectra

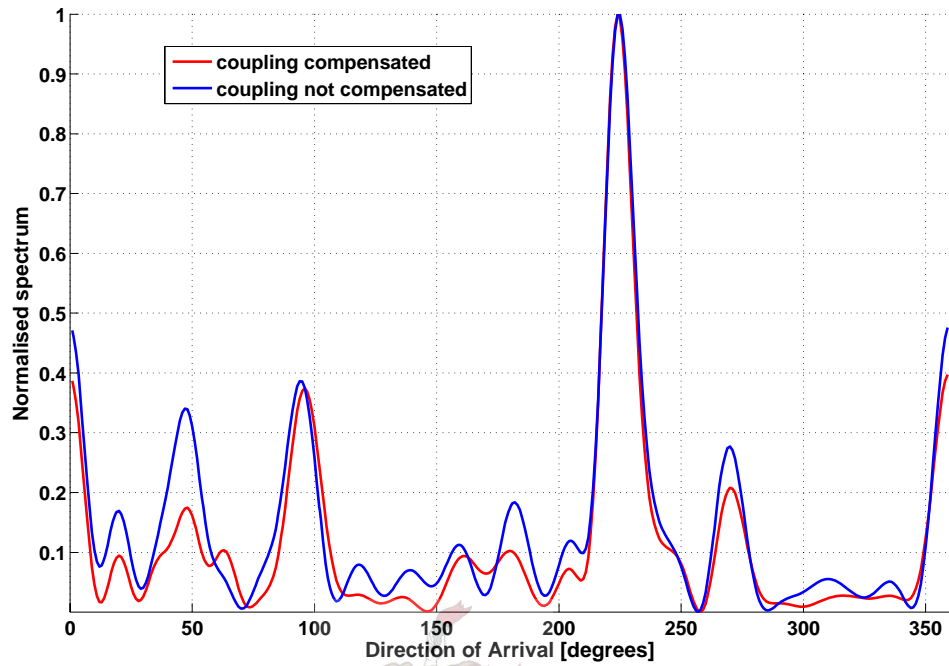


(a) Eigen decomposition, noise subspace. Azimuth peak = 224° .

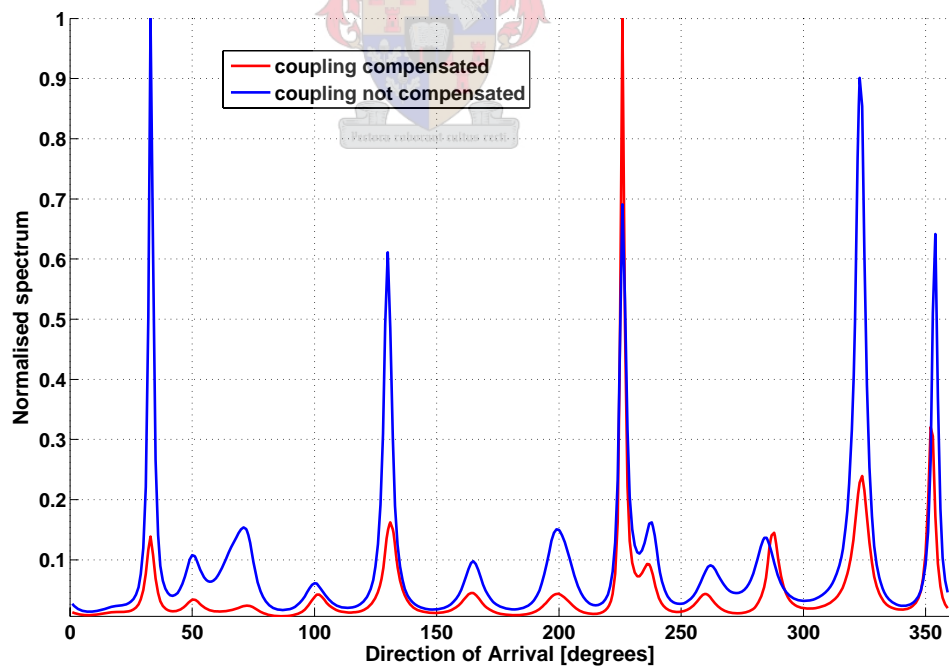


(b) PASTd, signal subspace. Azimuth peak = 224° .

Figure 4.10: Location 2 (222.4°): DOA spectra (cont'd)

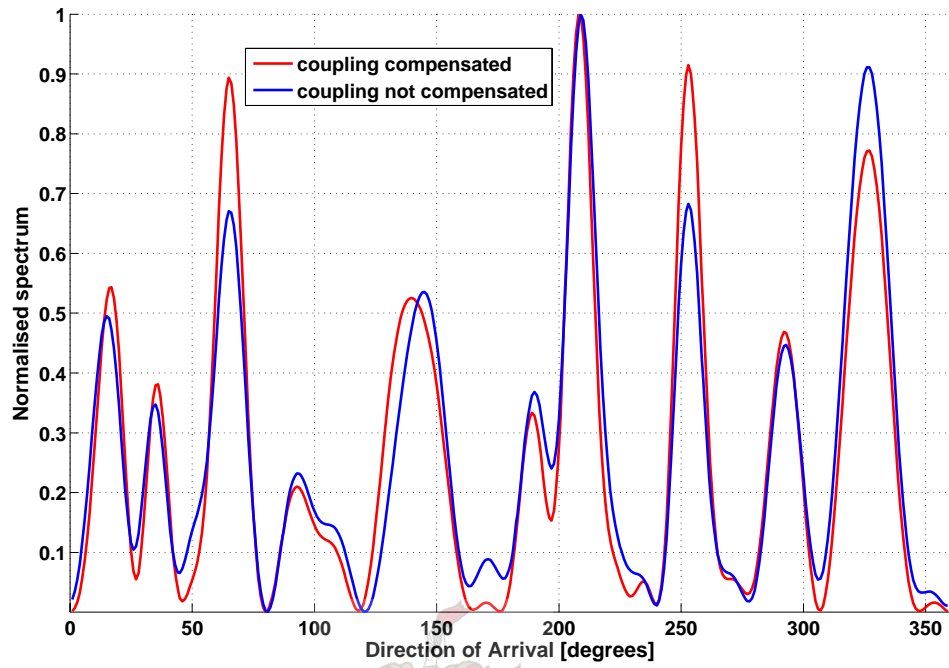


(c) FDPM, signal subspace. Azimuth peak = 224° .



(d) FDPM, noise subspace. Azimuth peak = 226° .

Figure 4.10: Location 2 (222.4°): DOA spectra (cont'd)



(e) OPERA, signal subspace. Azimuth peak = 224° .



4.3.3 Location 3

Figure 4.16 (a) shows the MUSIC spectrum for the results obtained when broadcasting from Location 3 (DF azimuth of 207.3°), using the eigen decomposition and noise subspace. The bearing is incorrectly estimated as 330° . Coupling compensation does not affect the DOA spectrum significantly.

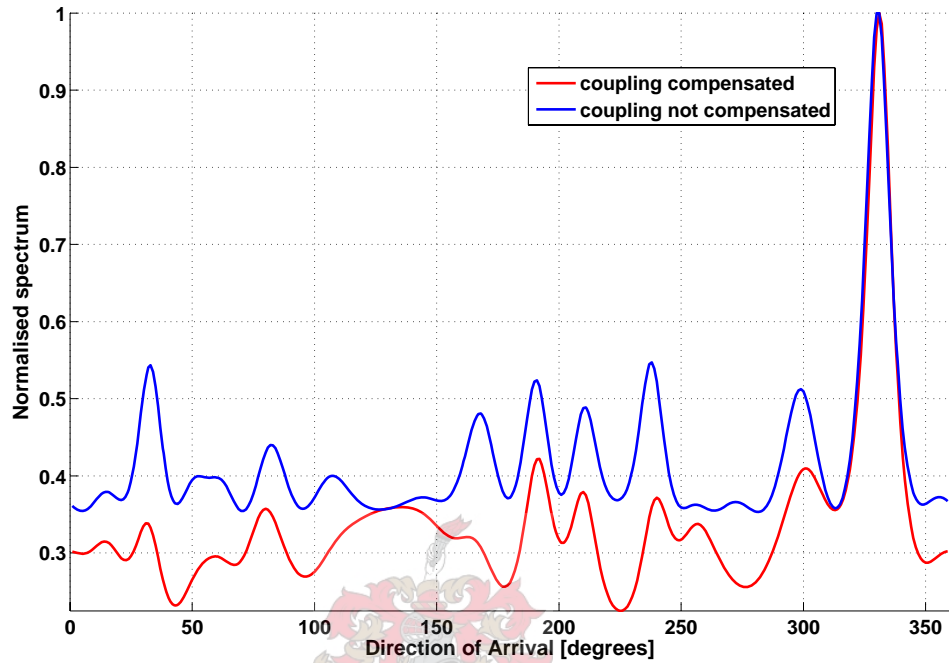
Figure 4.16(b) plots results using the PASTd algorithm and signal subspace. The bearing is incorrectly estimated as 332° . Compensation for coupling has little effect on the DOA spectrum.

Using FDPM and the signal subspace, the bearing is also incorrectly estimated as 332° (figure 4.16(c)). Coupling compensation does not seem necessary. Using the noise subspace, the FDPM algorithm estimates the bearing as 333° (figure 4.16(d)). Compensating for mutual coupling alters the DOA spectrum significantly, although the estimated bearing is not correct in either case (the DOA peak is located at 301° in the uncompensated case).

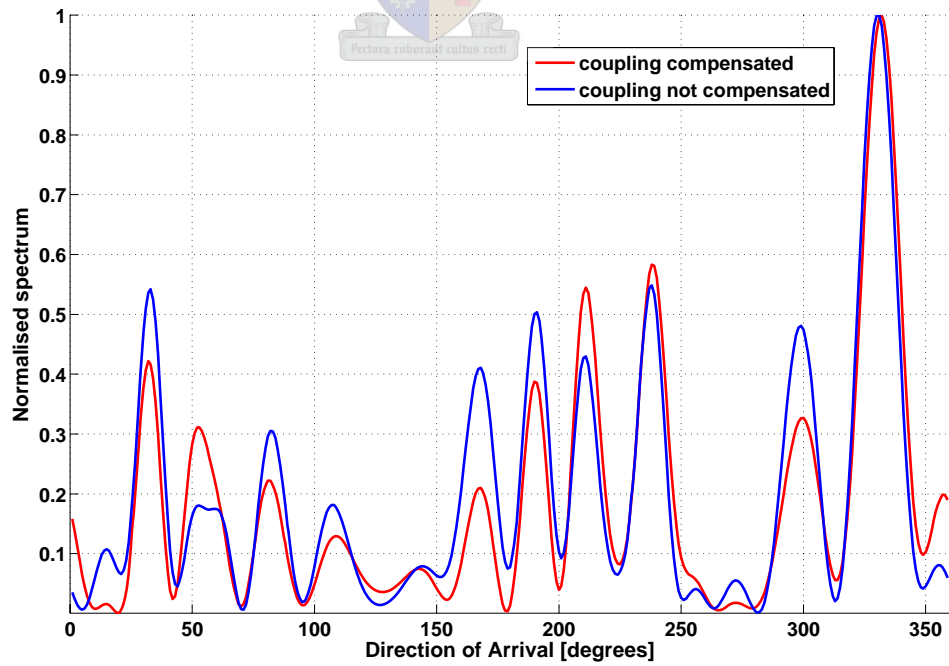
Figure 4.16(e) shows the results using the OPERA algorithm and signal subspace. The bearing is incorrectly estimated as 332° . Compensation for coupling has little effect on the DOA spectrum.



Figure 4.11: Location 3 (207.3°): DOA spectra

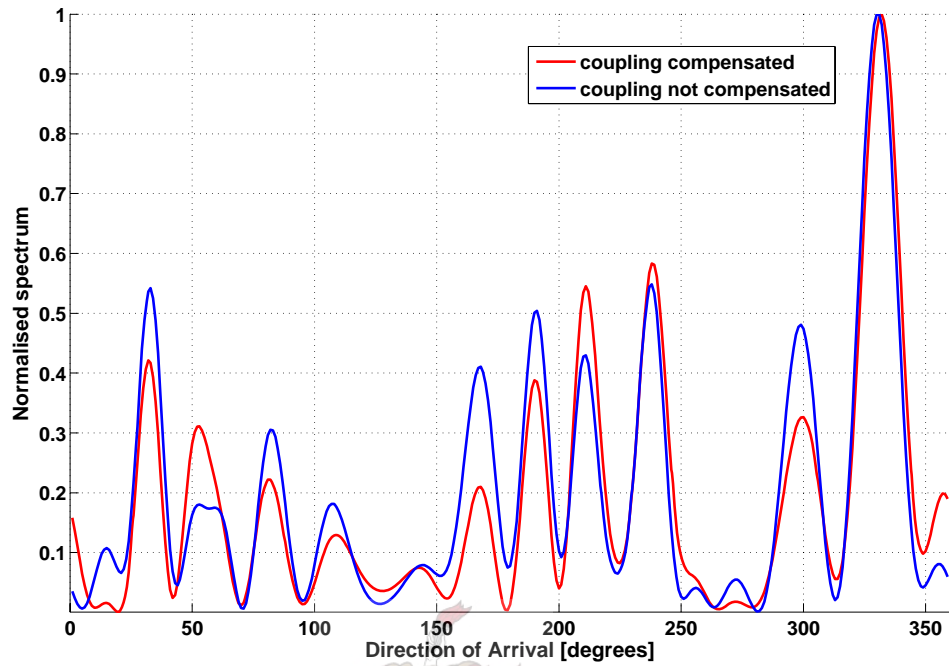


(a) Eigen decomposition, noise subspace. Azimuth peak = 330°.

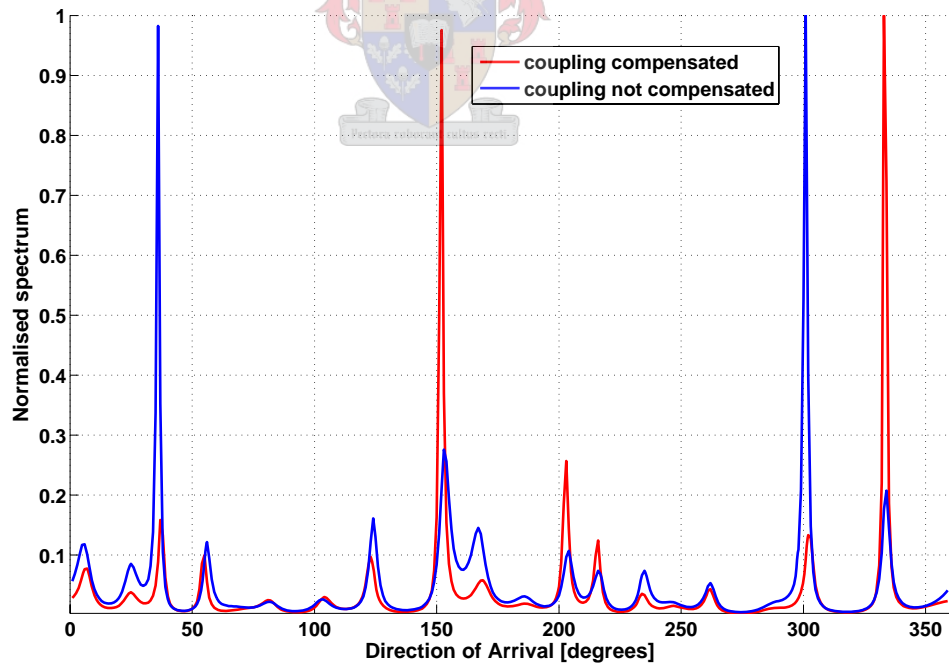


(b) PASTd, signal subspace. Azimuth peak = 332°.

Figure 4.11: Location 3 (207.3°): DOA spectra (cont'd)

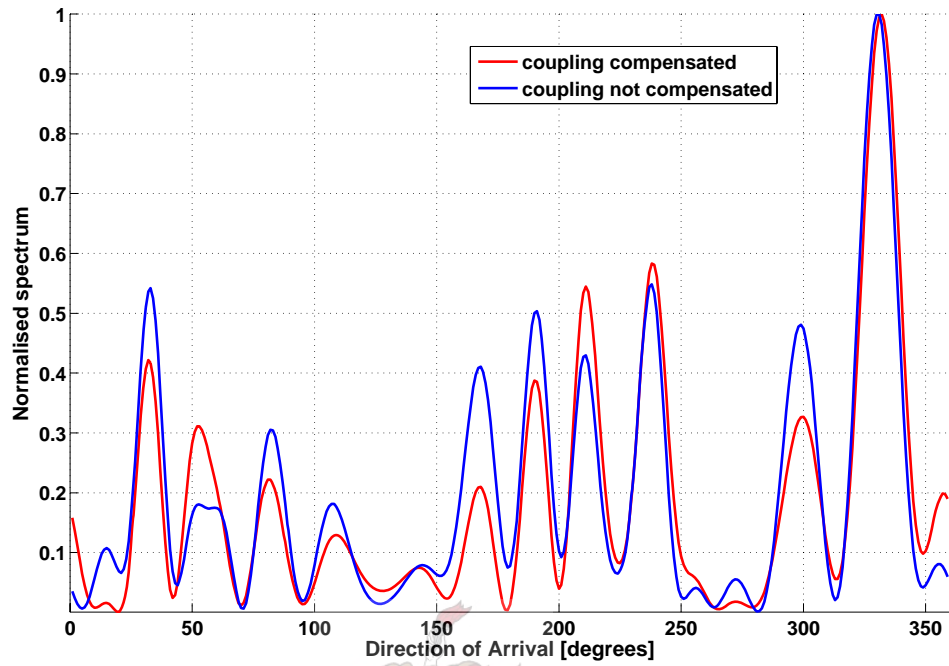


(c) FDPM, signal subspace. Azimuth peak = 332°.

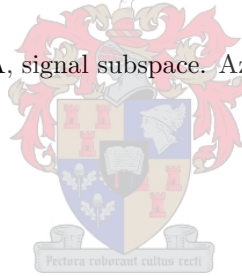


(d) FDPM, noise subspace. Azimuth peak = 333°.

Figure 4.11: Location 3 (207.3°): DOA spectra (cont'd)



(e) OPERA, signal subspace. Azimuth peak = 332°.



4.3.4 Location 4

Figure 4.17 (a) shows the MUSIC spectrum for the results obtained when broadcasting from Location 4 (DF azimuth of 112.1°), using the eigen decomposition and noise subspace. The bearing is estimated as 114° (an error of approximately 2°) when compensating for coupling, although it is incorrectly estimated as 247° when coupling is not considered.

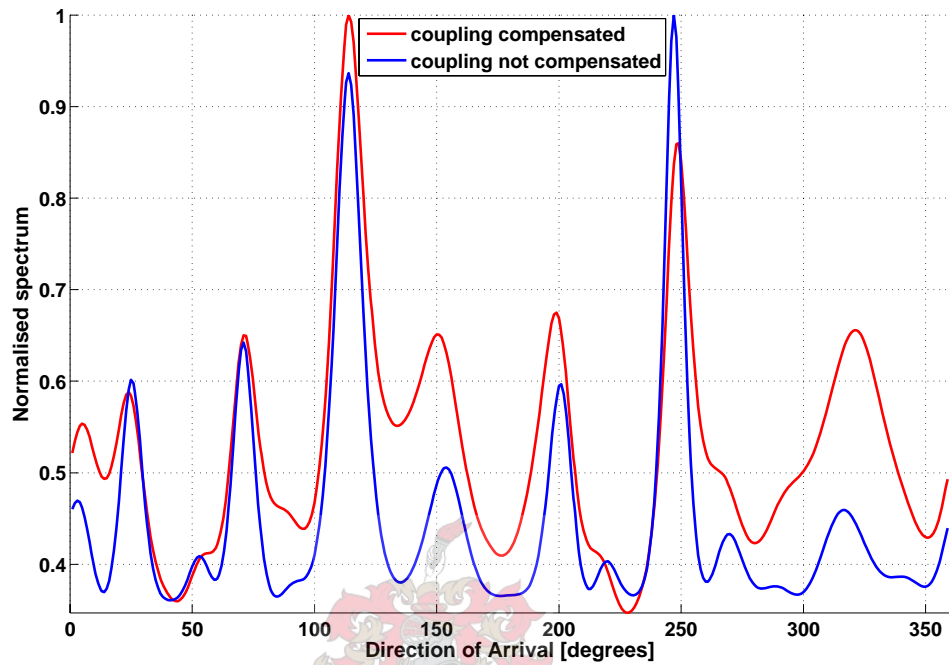
Figure 4.17(b) plots results using the PASTd algorithm and signal subspace. The bearing is estimated as 113° when compensating for coupling (an error of approximately 1°). When coupling is not compensated, the bearing is incorrectly estimated as 247° .

Using FDPM and the signal subspace, the bearing is again estimated as 113° (figure 4.17(c)), when compensating for coupling. Without coupling compensation, the bearing is estimated as 247° . Using the noise subspace, the FDPM algorithm incorrectly estimates the bearing as 293° (figure 4.17(d)). Compensating for mutual coupling alters the location of the global maximum in the DOA spectrum by only 2° .

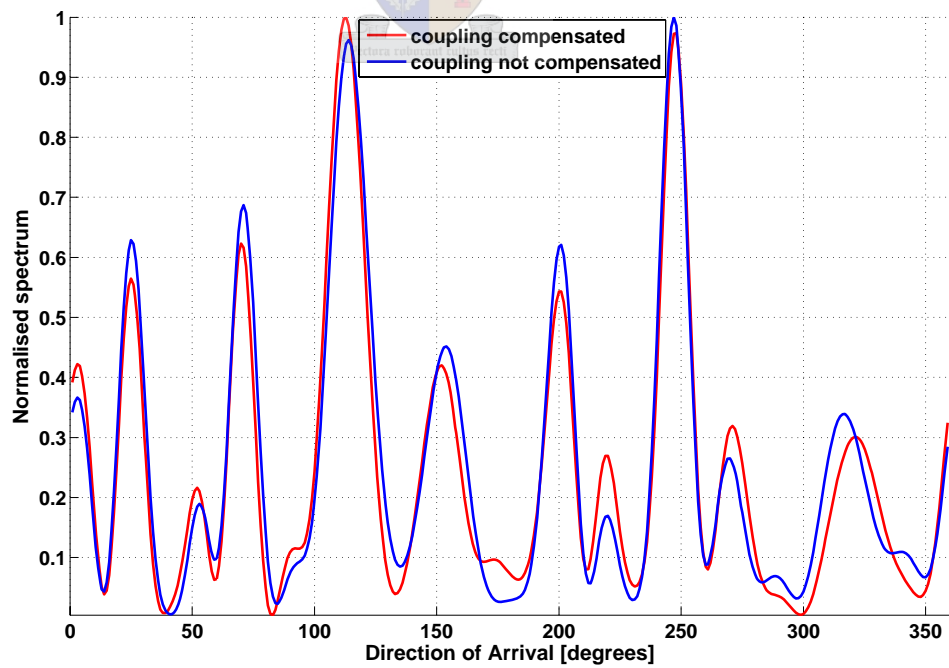
Figure 4.17(e) shows the results using the OPERA algorithm and signal subspace. The results closely resemble the PASTd and FDPM signal cases. The bearing is estimated as 113° . Compensation for coupling ensures that the correct global maximum is identified (a significant local maximum is present at 247°).



Figure 4.12: Location 4 (112.1°): DOA spectra

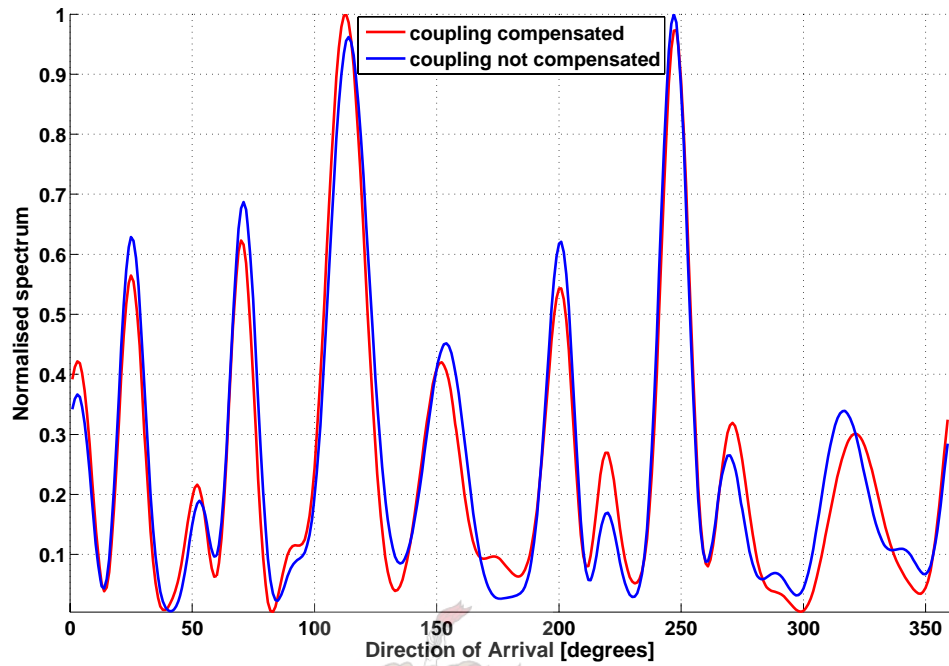


(a) Eigen decomposition, noise subspace. Azimuth peak = 114°.

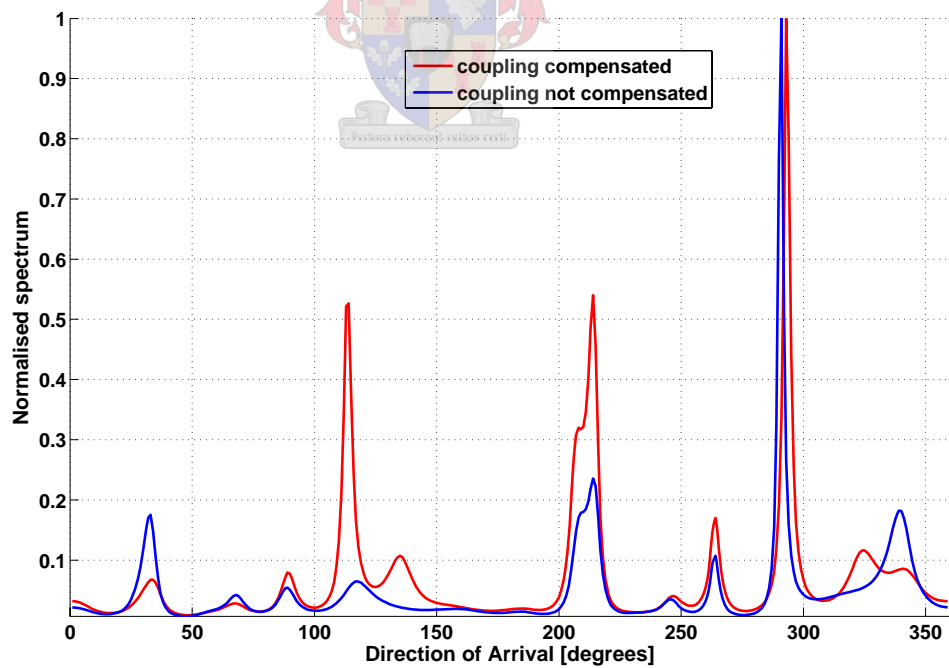


(b) PASTd, signal subspace. Azimuth peak = 113°.

Figure 4.12: Location 4 (112.1°): DOA spectra (cont'd)

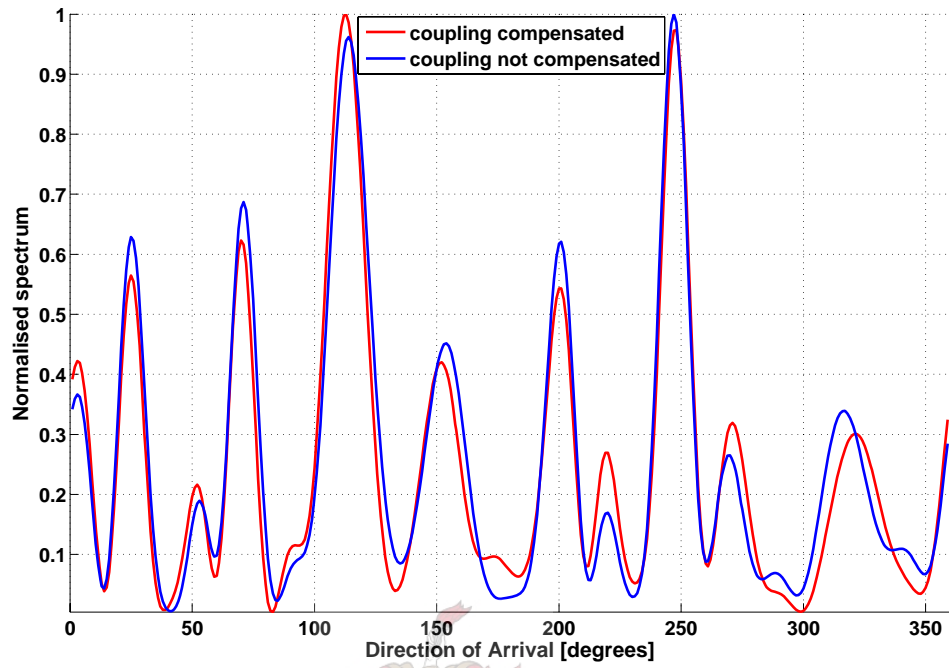


(c) FDPM, signal subspace. Azimuth peak = 113°.

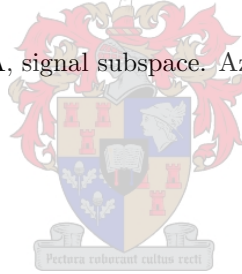


(d) FDPM, noise subspace. Azimuth peak = 247°.

Figure 4.12: Location 4 (112.1°): DOA spectra (cont'd)



(e) OPERA, signal subspace. Azimuth peak = 113°.



4.3.5 Location 5

Figure 4.18 (a) shows the MUSIC spectrum for the results obtained when broadcasting from Location 5 (DF azimuth of 38.3°), using the eigen decomposition and noise subspace. The bearing is estimated as 40° (an error of less than 2°) when compensating for coupling. The results when not considering coupling are nearly identical.

Figure 4.18(b) plots results using the PASTd algorithm and signal subspace. The bearing is estimated as 41° when compensating for coupling (an error of less than 3°). Compensation for coupling does not appear to be necessary.

Using FDPM and the signal subspace, the bearing is again estimated as 41° (figure 4.18(c)), when compensating for coupling. The difference in the global maximum is approximately 1° when compensating for coupling. Using the noise subspace, the FDPM algorithm incorrectly estimates the bearing as 120° (figure 4.18(d)) when compensating for mutual coupling. A local maximum is apparent at approximately the correct location (when compensating for coupling), although a spurious global maximum is incorrectly identified as the bearing.

Figure 4.18(e) shows the results using the OPERA algorithm and signal subspace. The bearing is estimated as 41° . Compensation for coupling has negligible effect of the location of the global maximum in the DOA spectrum.

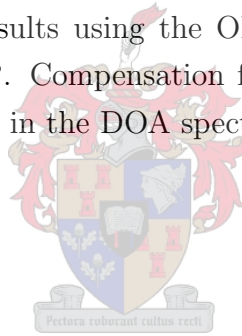
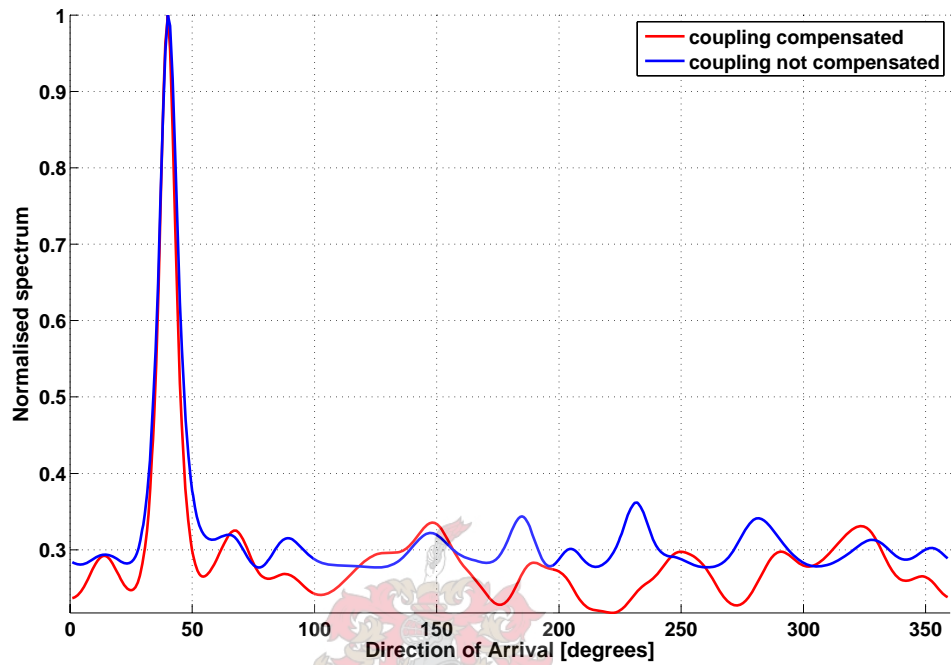
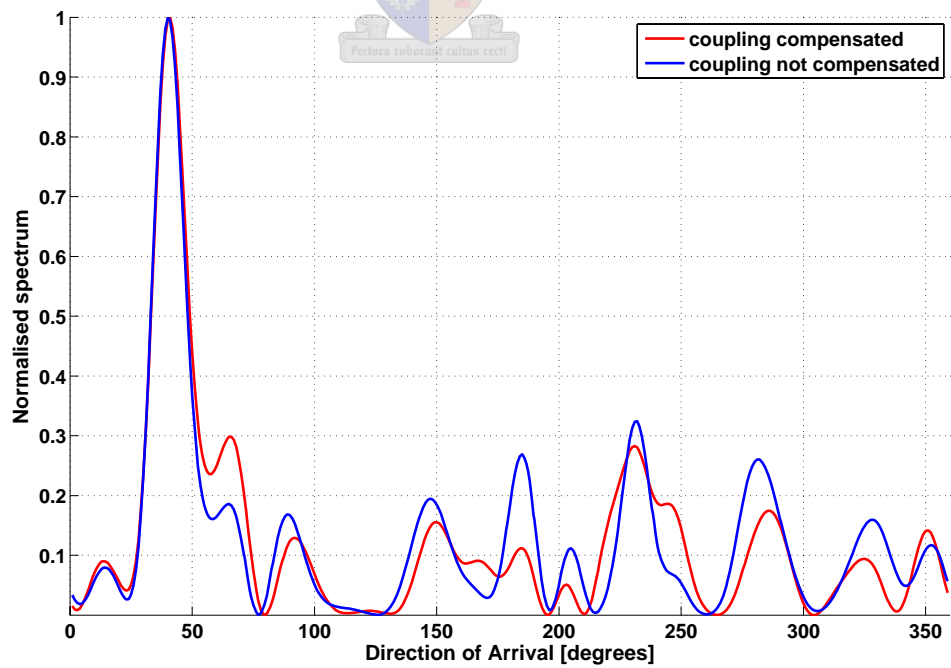


Figure 4.13: Location 5 (38.3°): DOA spectra

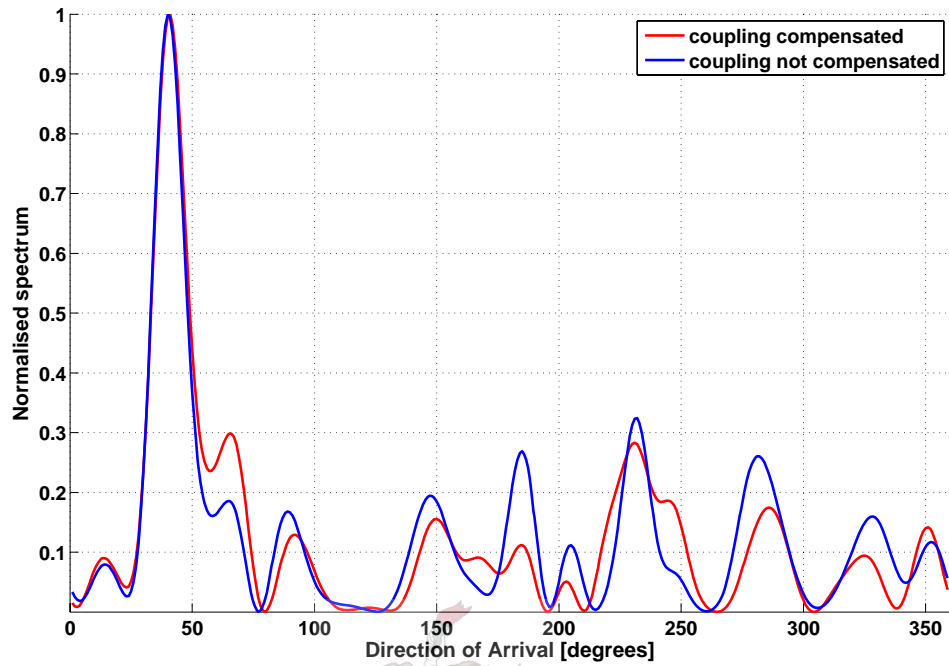


(a) Eigen decomposition, noise subspace. Azimuth peak = 40° .

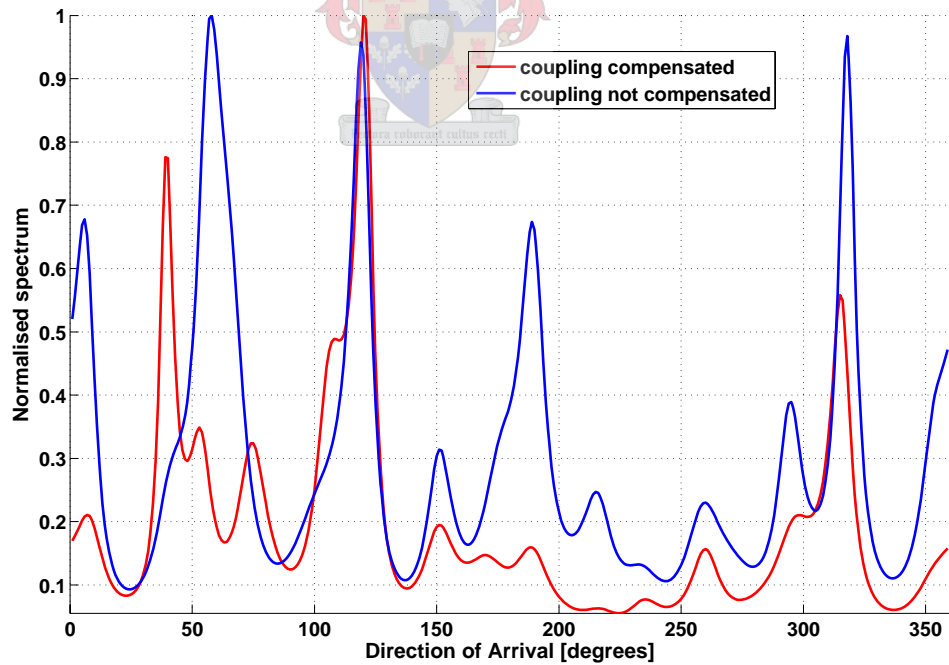


(b) PASTd, signal subspace. Azimuth peak = 41° .

Figure 4.13: Location 5 (38.3°): DOA spectra (cont'd)

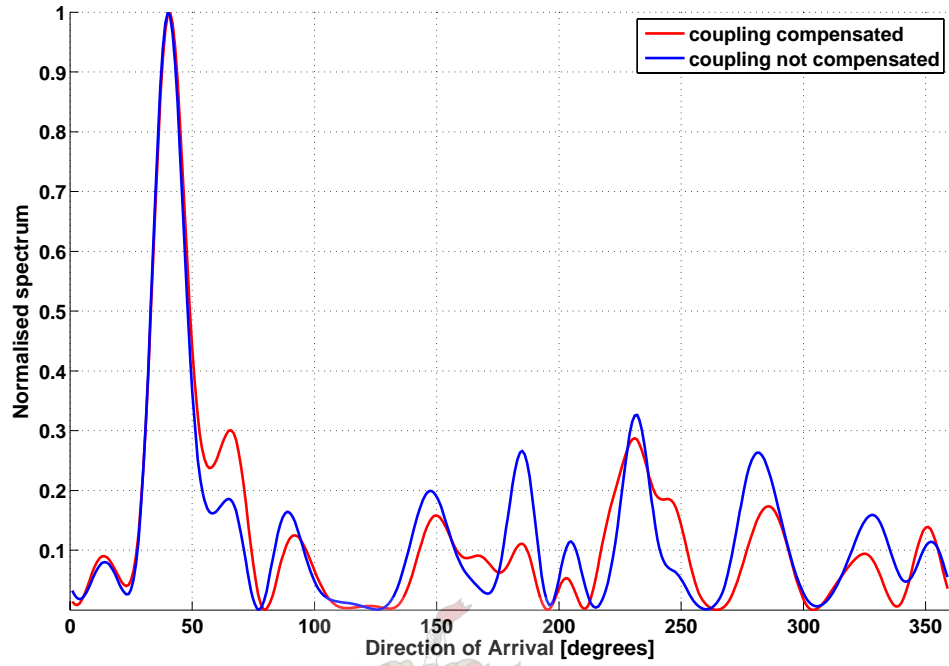


(c) FDPM, signal subspace. Azimuth peak = 41° .



(d) FDPM, noise subspace. Azimuth peak = 120° .

Figure 4.13: Location 5 (38.3°): DOA spectra (cont'd)



(e) OPERA, signal subspace. Azimuth peak = 41° .

The errors at each location are summarised for each subspace tracking method in table 4.3.5.

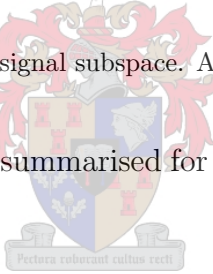


Table 4.3: Subspace tracking DF tests: bearing errors

Location	Eigen decomposition	PASTd	FDPM (signal subspace)	FDPM (noise subspace)	OPERA
1	2°	119°	119°	159°	119°
2	2°	2°	2°	4°	14°
3	123°	125°	125°	126°	125°
4	2°	1°	1°	179°	1°
5	2°	3°	3°	82°	3°

4.3.6 Simulation of the field tests

The simulator described in section 3.1 was loaded with the spacings of the antennas used for the field tests in Pretoria. The azimuth, elevation and distance values for each location were then used to generate simulated data. Phase differences arising from the spherical wavefront were incorporated into the model for best accuracy (this is most significant at small distances from the array). Amplitude differences and “shadowing” (irregular illumination when elements are coincident with the Direction-Of-Arrival) were not considered. The simulated channel data was premultiplied by the coupling matrix (calculated in section 4.3) to incorporate the effects of mutual coupling into the simulation.

A 10 kHz (narrowband) simulated bandwidth was used, and the uncorrelated, spherically isotropic and cylindrically isotropic noise floors were set to -50 dBm, -90 dBm and -80 dBm respectively. A narrowband noise source, 9 kHz wide, was used as the simulated signal at each location.

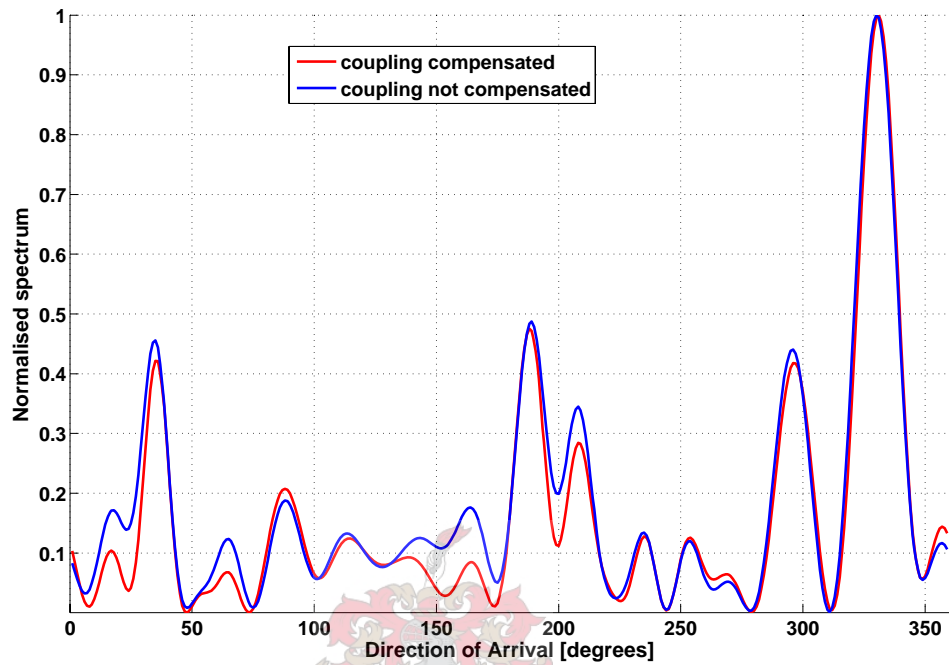
The MUSIC spectrum results arising from the simulated data at each location are presented below. Results are again obtained for the cases when coupling is compensated and when it is not.

The results of performing Direction Finding on the simulated data produce estimates within 2° of the simulated azimuths for all cases. The signal subspace algorithms all have very similar DF spectra, indicating that none of them struggled to converge. Interestingly, the noise subspace algorithm seems quite sensitive to mutual coupling. At the simulated SNR, compensation for mutual coupling was necessary to eliminate wild bearings. If the SNR is increased by lowering the uncorrelated noise floor to -60 dBm, the FDPM noise subspace spectrum shows fewer wild bearings. (see figure 4.19).

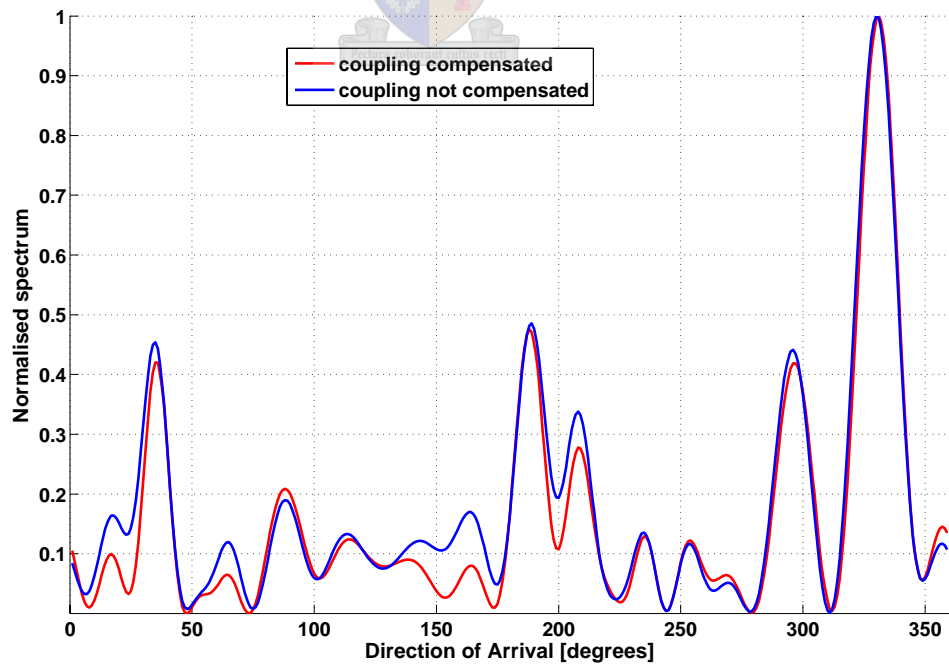
Since the results obtained using simulations of the field tests produce extremely satisfactory DF estimates, one must conclude that the poor field results are not indications of essential ambiguities in the antenna configuration or errors in the processing algorithms. Instead, uncontrolled factors in the tests (including multipath propagation, poor grounding and uncalibrated differences between antenna elements) must be postulated as possible factors.

Location 1

Figure 4.14: Location 1 (327.7°): DOA spectra

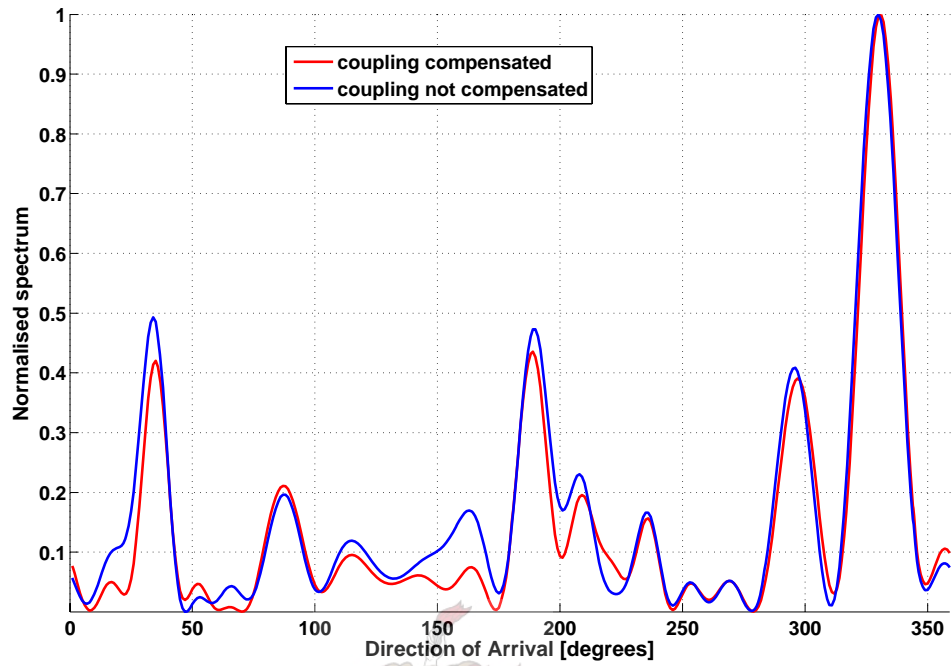


(a) PASTd, signal subspace. Azimuth peak = 331° .

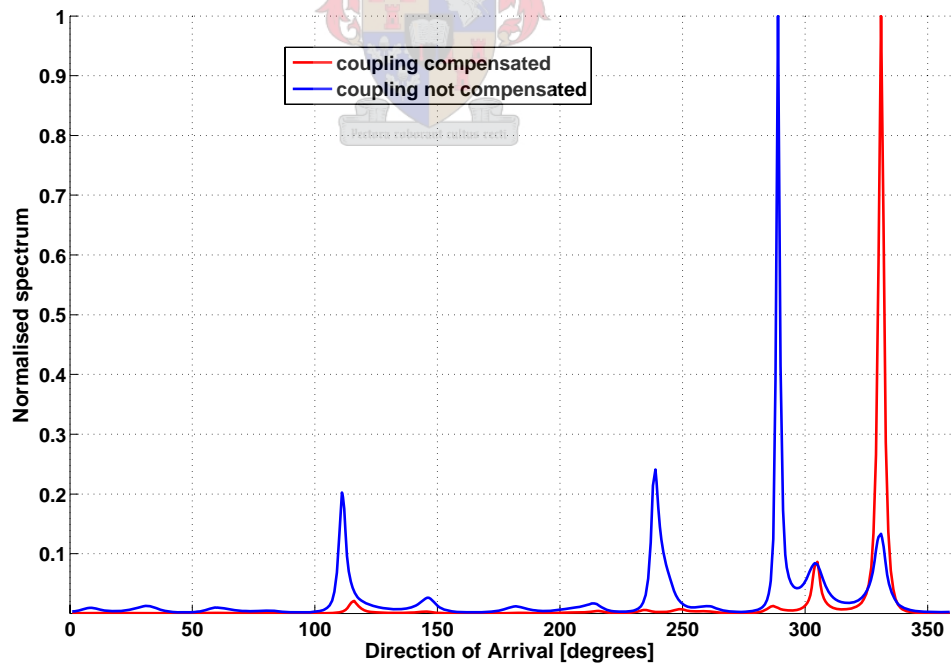


(b) FDP, signal subspace. Azimuth peak = 331° .

Figure 4.14: Location 1 (327.7°): DOA spectra (cont'd)



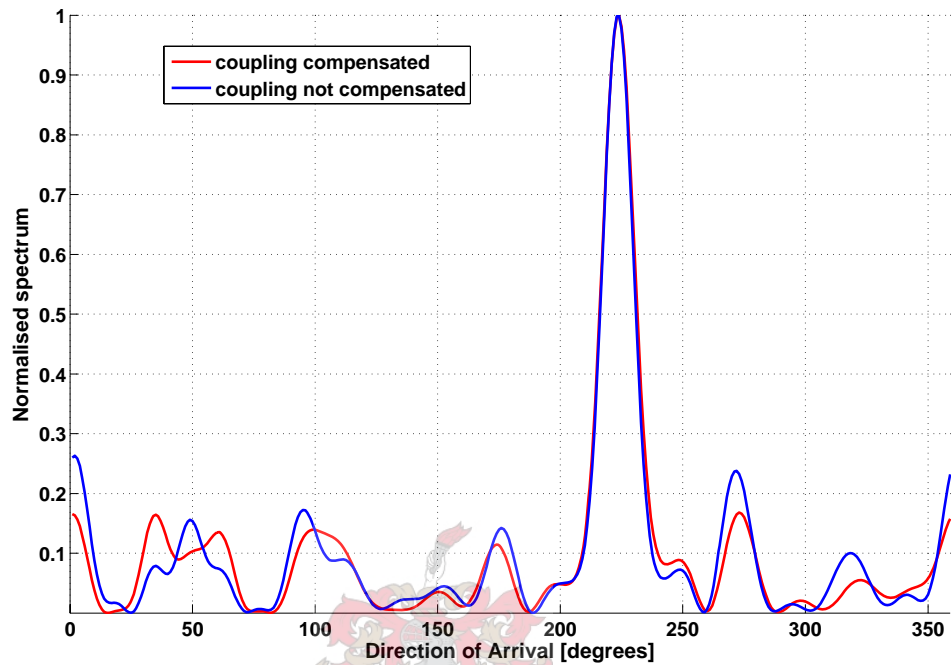
(c) SVD-OPERA, signal subspace. Azimuth peak = 331° .



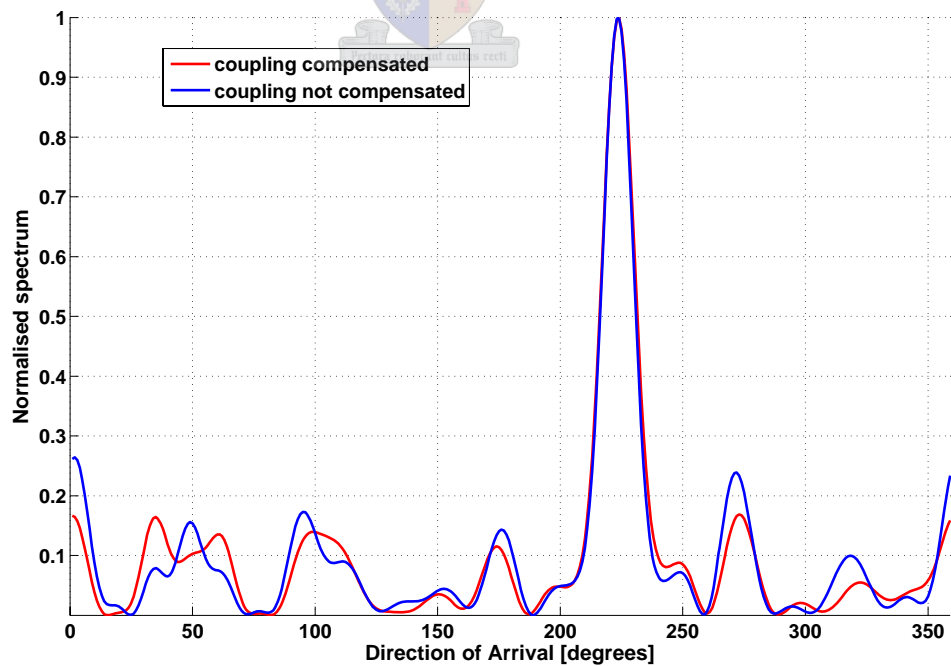
(d) FDP, noise subspace. Azimuth peak = 330° . (292° uncompensated)

Location 2

Figure 4.15: Location 2 (222.44°): DOA spectra

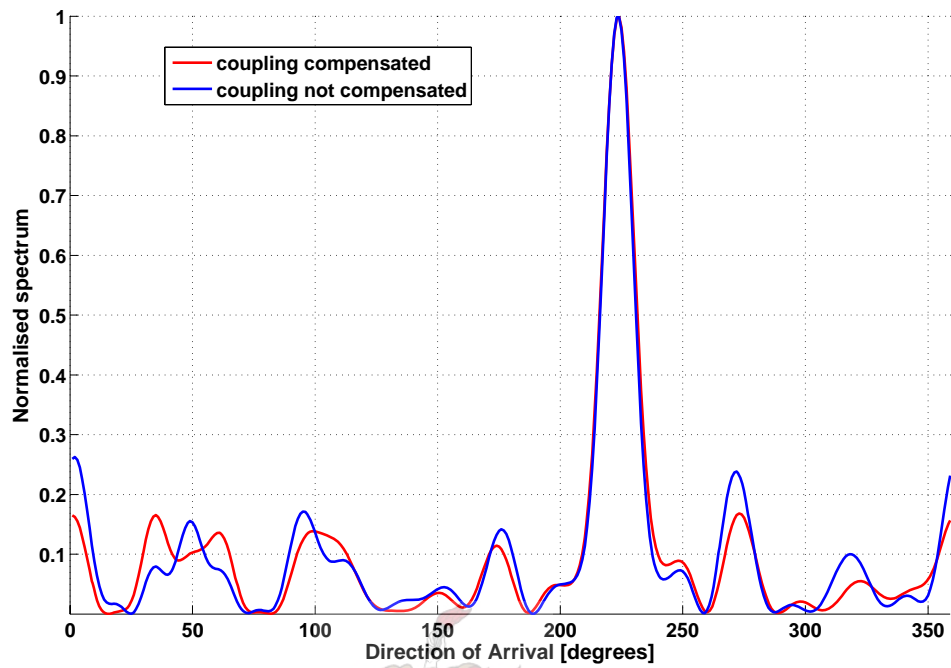


(a) PASTd, signal subspace. Azimuth peak = 224° .

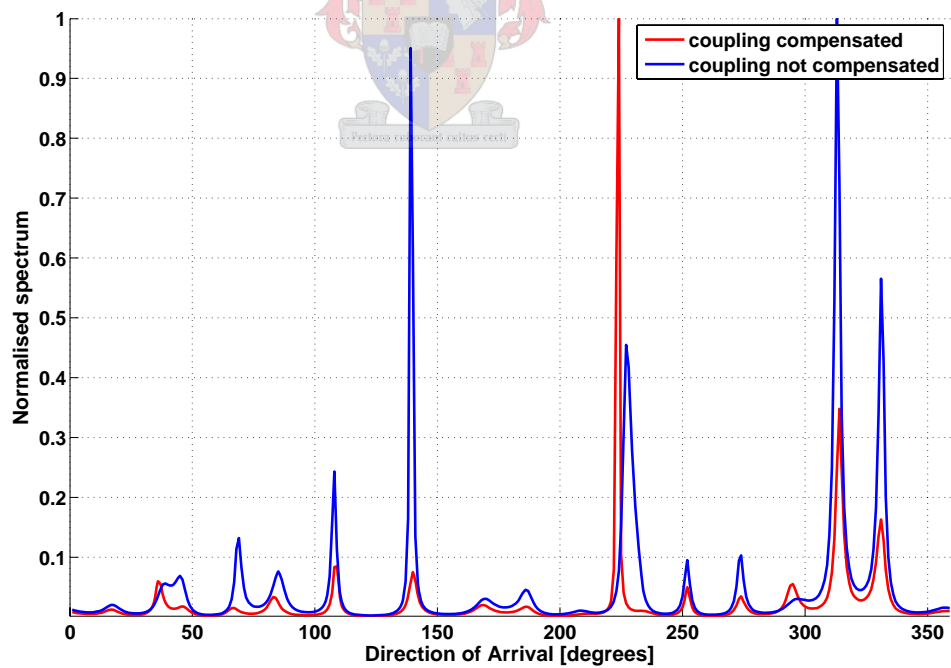


(b) FDPM, signal subspace. Azimuth peak = 224° .

Figure 4.15: Location 2 (222.44°): DOA spectra (cont'd)



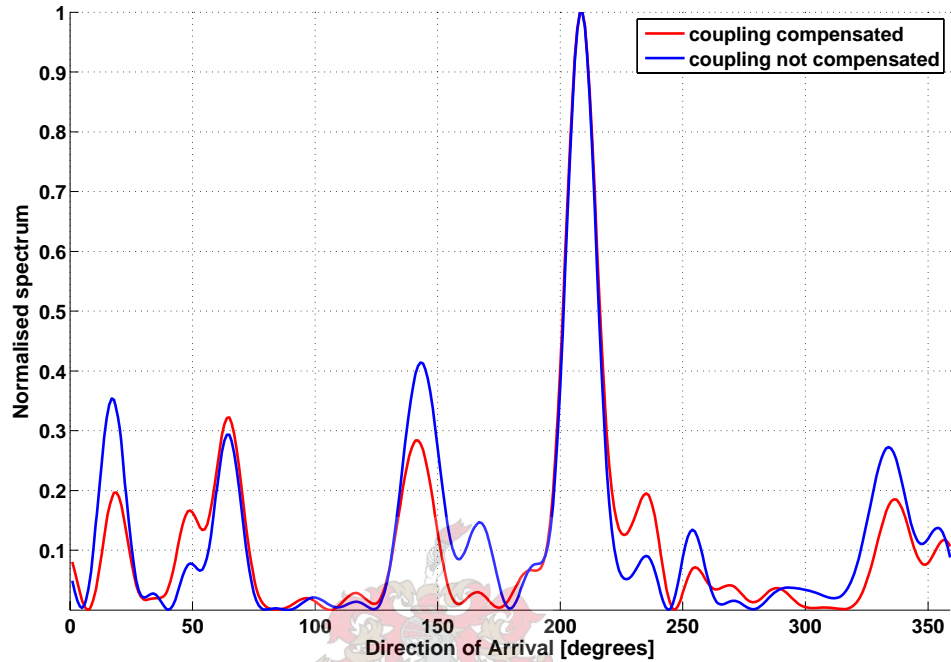
(c) SVD-OPERA, signal subspace. Azimuth peak = 224° .



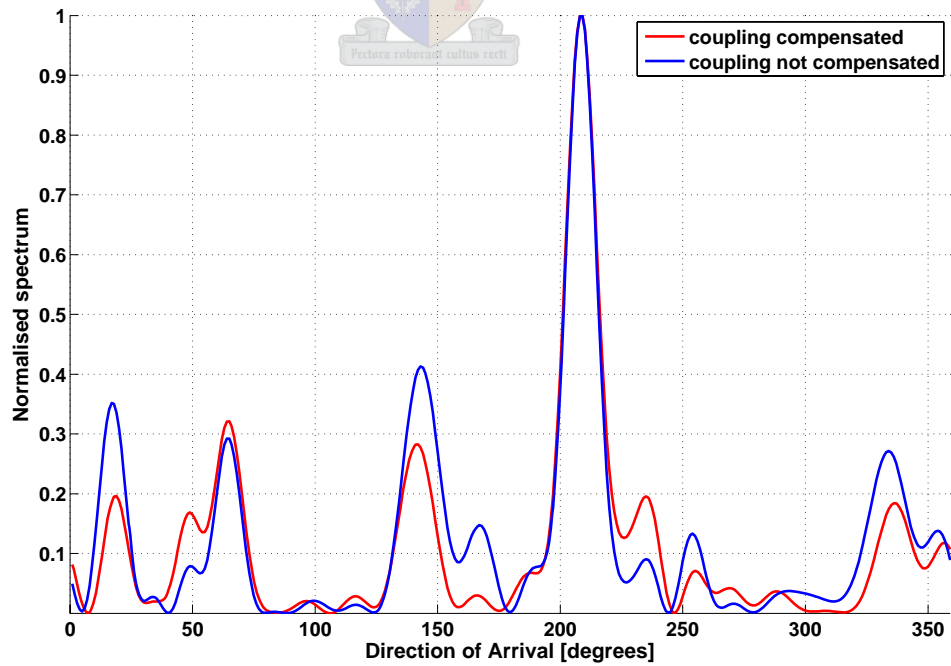
(d) FDPM, noise subspace. Azimuth peak = 223° . (309° uncompensated)

Location 3

Figure 4.16: Location 3 (207.25°): DOA spectra

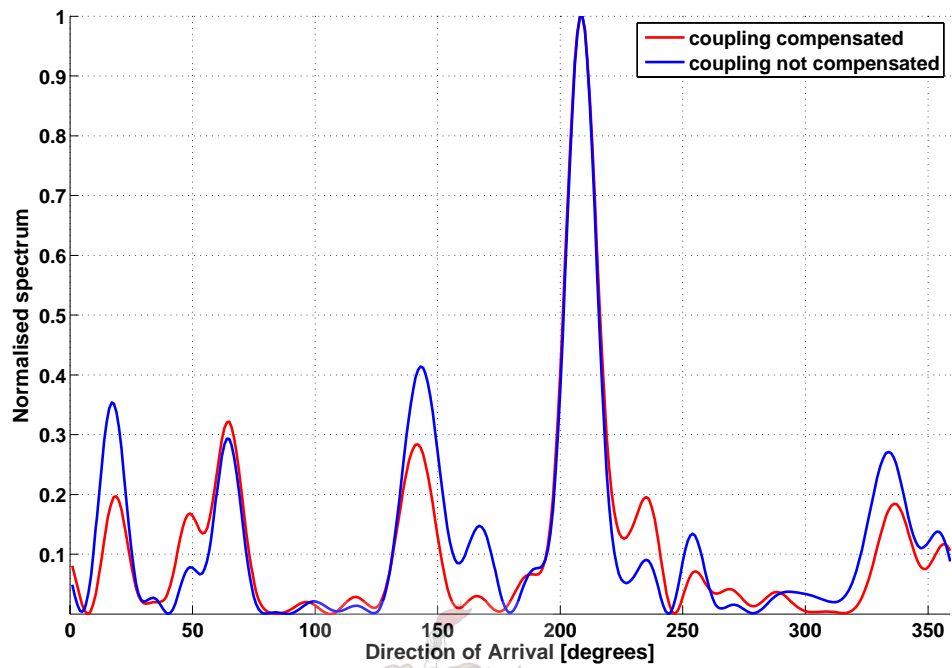


(a) PASTd, signal subspace. Azimuth peak = 209° .

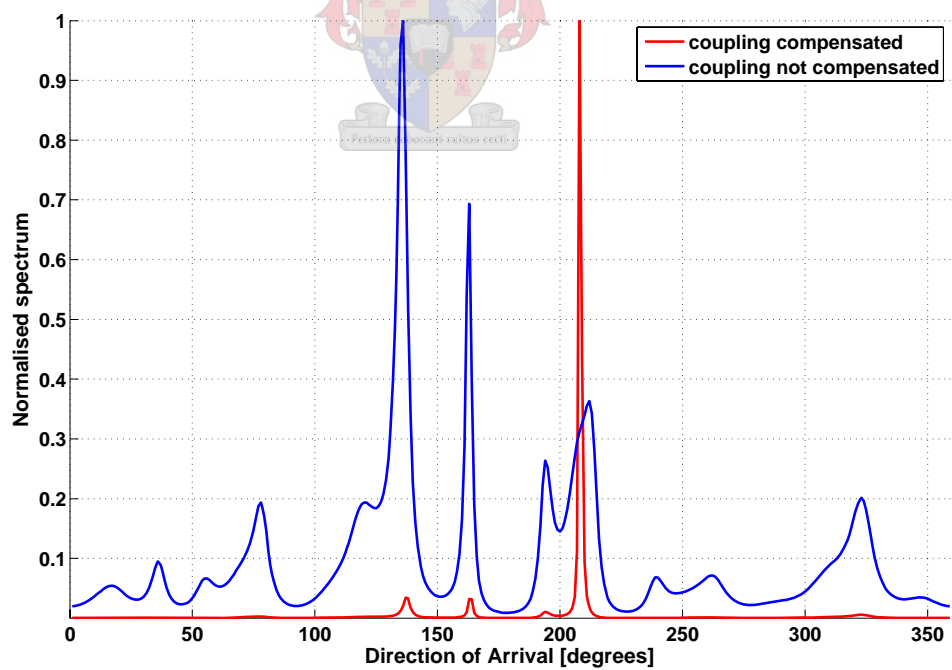


(b) FDP, signal subspace. Azimuth peak = 209° .

Figure 4.16: Location 3 (207.25°): DOA spectra (cont'd)



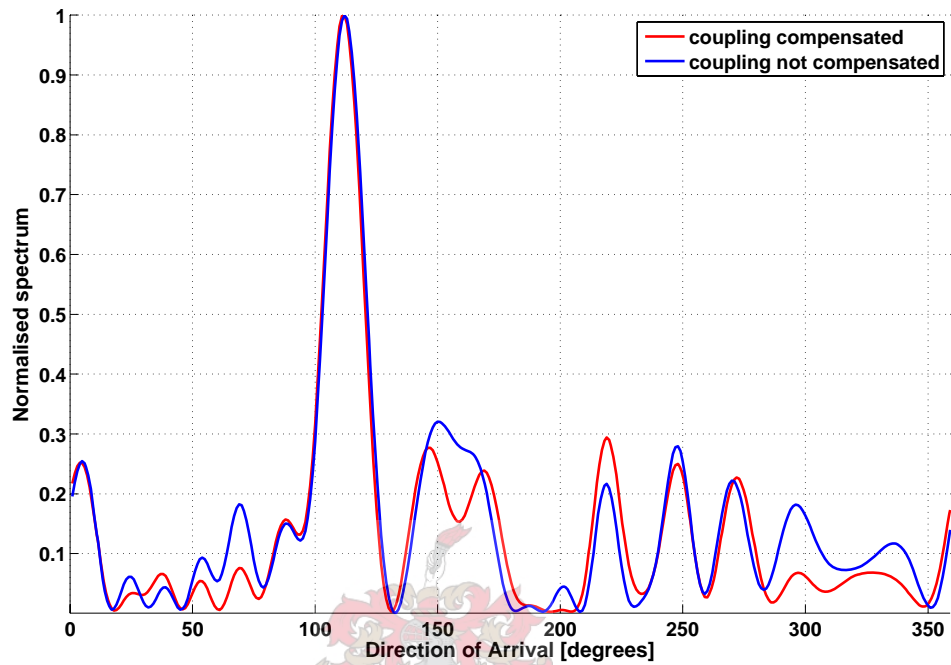
(c) SVD-OPERA, signal subspace. Azimuth peak = 209°.



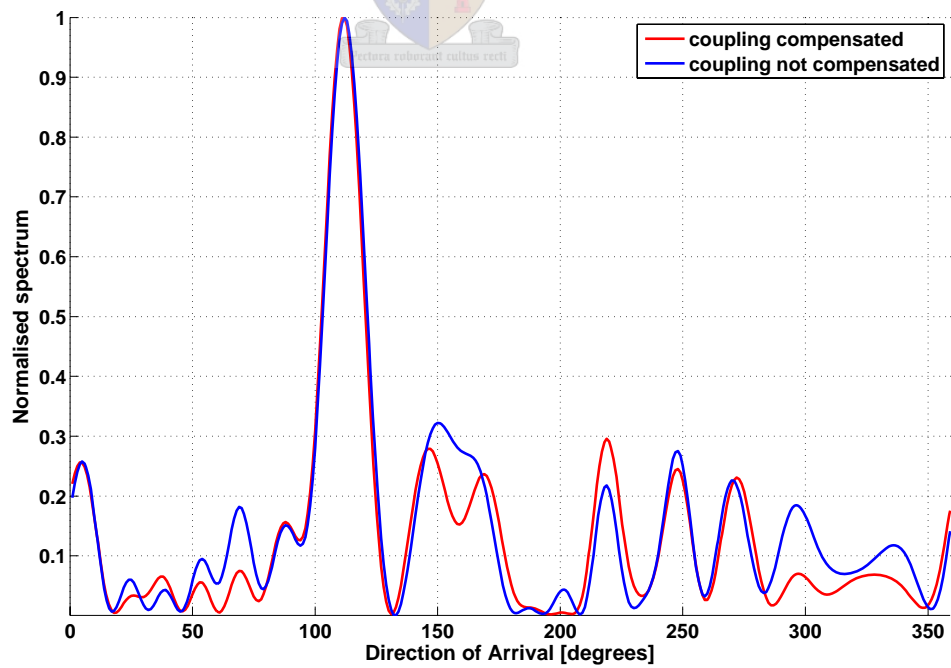
(d) FDPM, noise subspace. Azimuth peak = 208°. (138° uncompensated)

Location 4

Figure 4.17: Location 4 (112.13°): DOA spectra

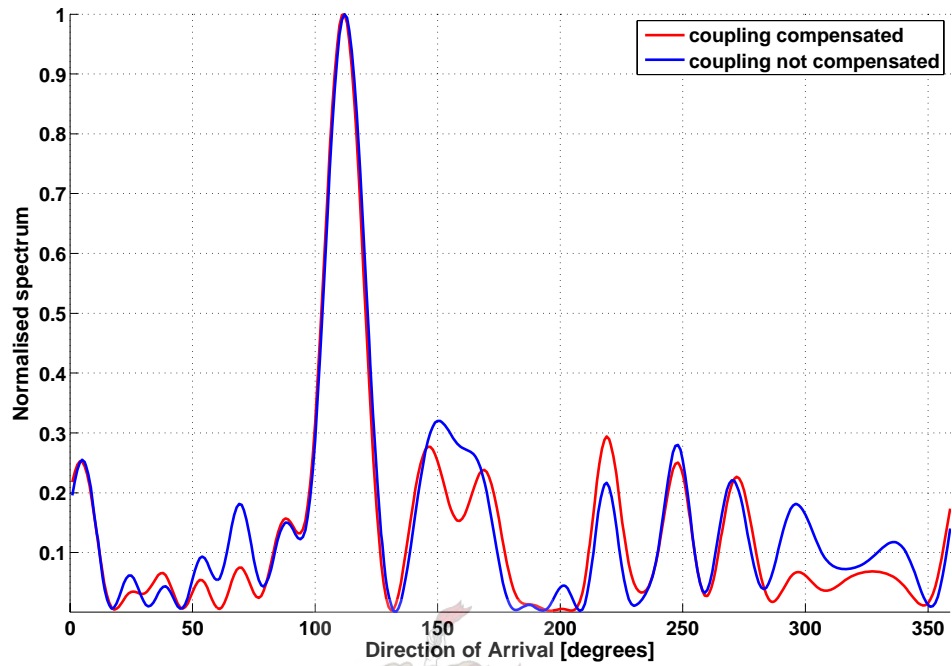


(a) PASTd, signal subspace. Azimuth peak = 112° .

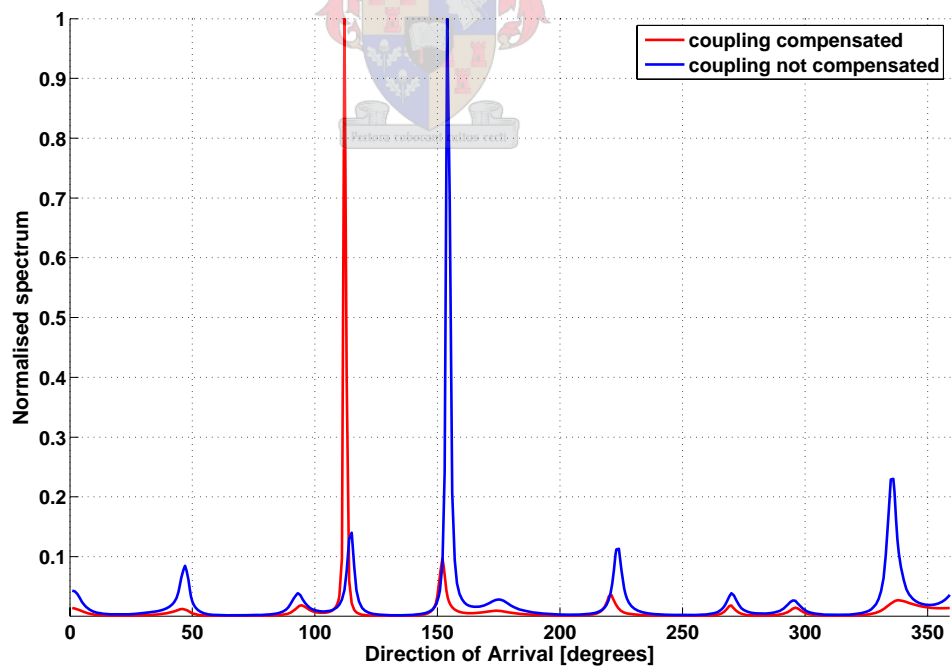


(b) FDP, signal subspace. Azimuth peak = 112° .

Figure 4.17: Location 4 (112.13°): DOA spectra (cont'd)



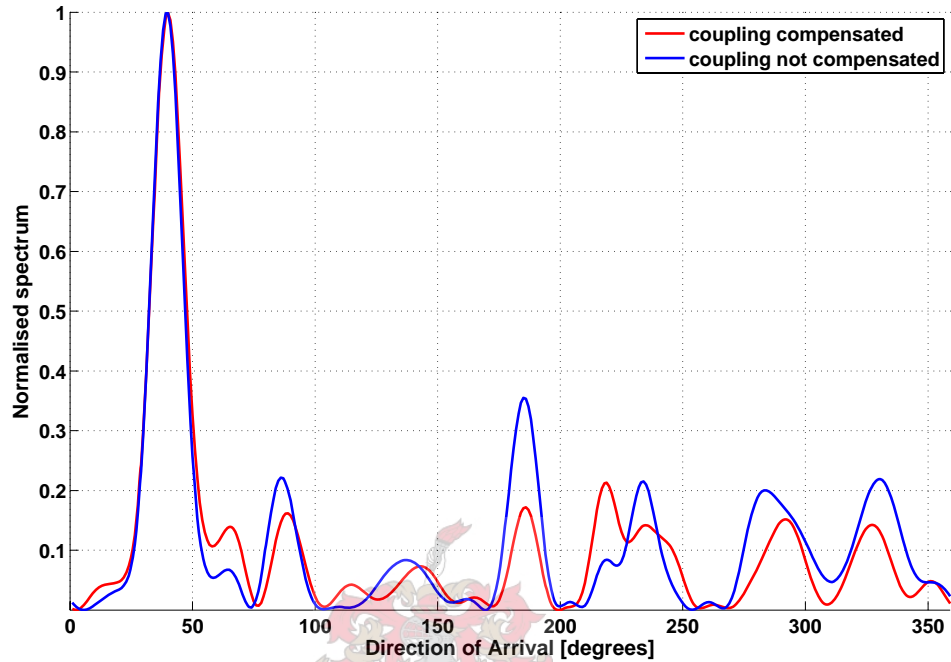
(c) SVD-OPERA, signal subspace. Azimuth peak = 112°.



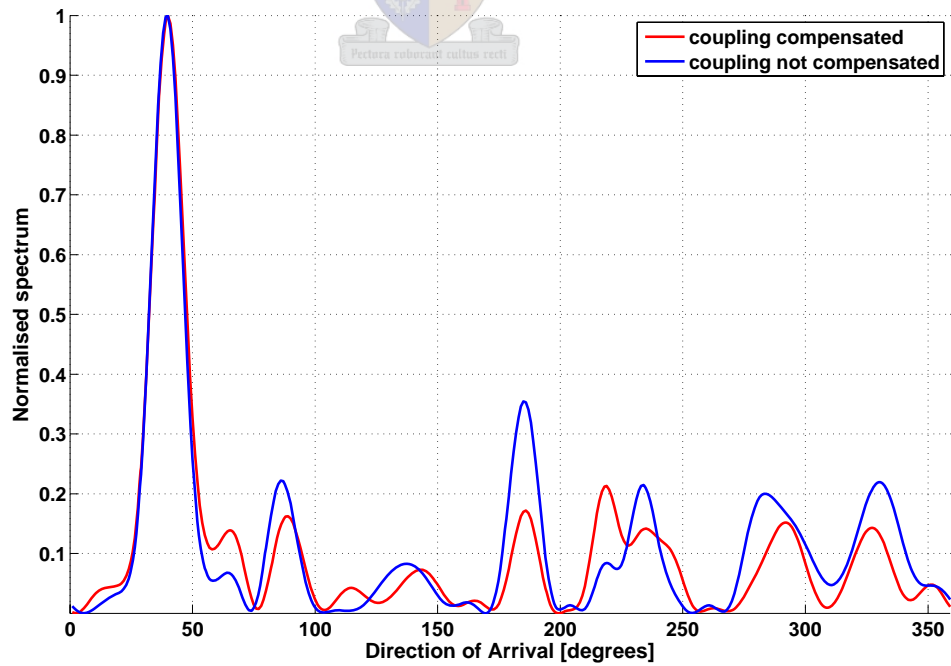
(d) FDPM, noise subspace. Azimuth peak = 112°. (153° uncompensated)

Location 5

Figure 4.18: Location 5 (38.25°): DOA spectra

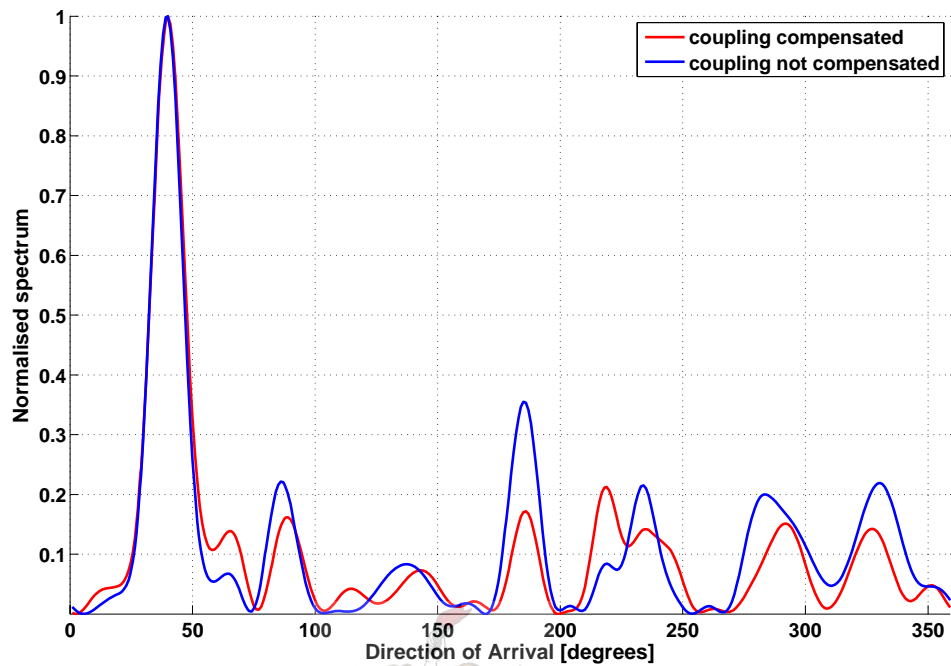


(a) PASTd, signal subspace. Azimuth peak = 40° .

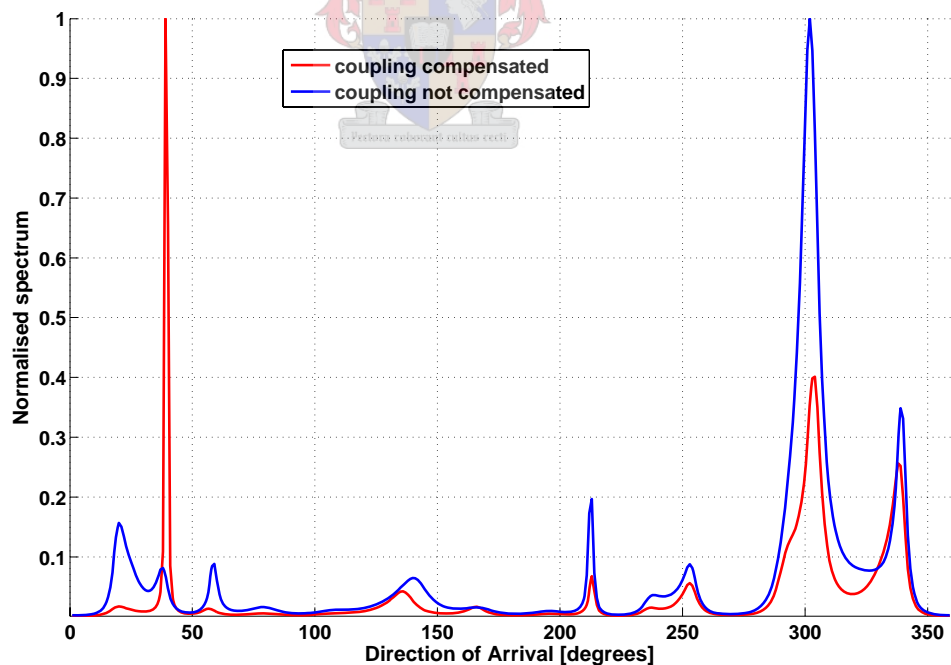


(b) FDPM, signal subspace. Azimuth peak = 40° .

Figure 4.18: Location 5 (38.25°): DOA spectra (cont'd)



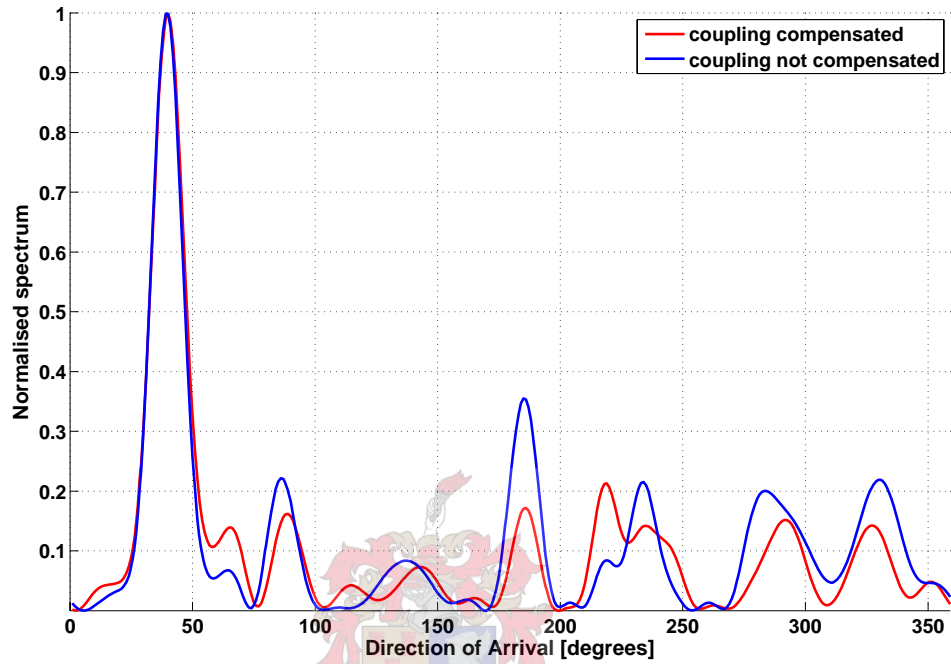
(c) SVD-OPERA, signal subspace. Azimuth peak = 40° .



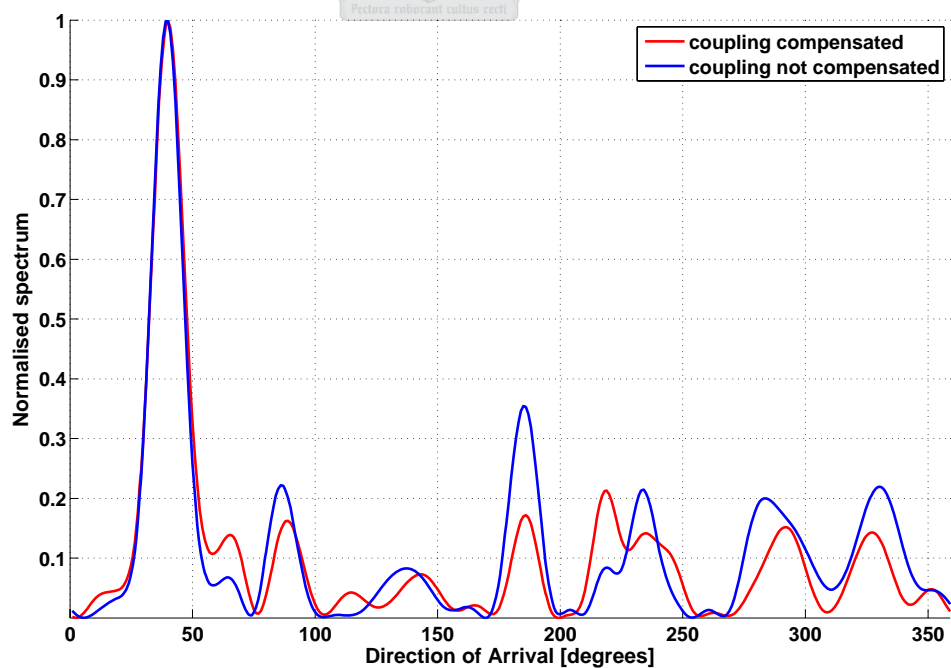
(d) FDPM, noise subspace. Azimuth peak = 39° . (302° uncompensated)

Location 1: Using FDPM (noise subspace) with uncorrelated noise floor lowered to -50 dBm

Figure 4.19: Location 1 (327.7°): DOA spectra



(a) FDPM, noise subspace. Azimuth peak = 330° (331° uncompensated).



(b) FDPM, noise subspace. Azimuth peak = 330° (333° uncompensated).

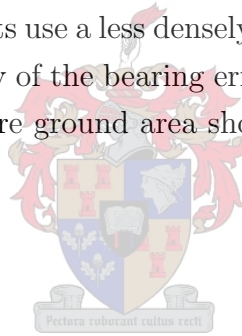
4.3.7 Conclusions regarding the field tests

On the whole, the field tests show some fairly large errors, especially at locations 1 and 3. These errors are not confined to specific subspace tracking methods, which suggests that the antenna setup and environment may be subject to ambiguities and multipath propagation. It is unclear whether the dense urban environment was a significant factor in the tests. The FDP noise subspace algorithm gives incorrect results at locations 4 and 5, which may confirm its high sensitivity to noise (see section 3.3.5). The dry ground may have affected antenna grounding, which could have been significantly different at each monopole element.

The eigen decomposition gives DF results that are accurate to within 2° at all locations other than location 3. PASTd and the signal subspace variant of FDP appear to perform similarly. The noise subspace variant of FDP appears to be less robust than the other algorithms. OPERA has the largest error of the methods tested at location 2.

It is clear that mutual coupling needs to be compensated, especially if antennas are spaced as closely as they were for the field tests.

It is proposed that future tests use a less densely populated area with fewer buildings to investigate whether the majority of the bearing errors were due to multipath propagation and scattering. Ideally, the entire ground area should also be wet to ensure a more even ground plane.



Chapter 5

Conclusions

The goal of this thesis was to examine the practicality of less complex alternatives to the SVD and eigen decomposition for radio direction finding. Subspace direction finding and the MUSIC algorithm were introduced in the context of spatial sampling by an antenna array. Following a literature study, three promising subspace tracking algorithms were identified as candidates for further examination. The theoretical basis for each algorithm was presented and each algorithm was implemented as part of a C++ client program for direction finding. A software simulator was also created, which allowed the relative performance of the algorithms to be evaluated under various simulated scenarios. A separate benchmark program was developed to quantify the relative efficiencies of the selected tracking methods. The theoretical complexities of the algorithms were evaluated, and these results were compared to the measured benchmark timing data. Finally, a field test was conducted as an initial assessment of the algorithms' performance when using data captured by a real antenna array in Pretoria. Practical complications such as antenna grounding, multipath propagation and mutual coupling were identified. Mutual coupling was examined in greater detail, by evaluating the coupling matrix of the particular antenna array using a computer simulation. Broadcasts were made from five selected locations, and the results using each algorithm were assessed. The effect of compensating for mutual coupling in the MUSIC algorithm was examined for each set of results.

The PASTd, FDPM and OPERA algorithms are all valid alternatives to the eigen decomposition and Singular Value Decomposition, when reduced complexity is required. They may be used in the context of Direction Finding to estimate the signal or noise subspace, followed by bearing estimation using the MUSIC algorithm. The AIC and MDL criteria may be used to estimate the number of significant signal sources present.

All three tracking algorithms examined exhibit different tradeoffs and behaviours as the Signal-to-Noise-Ratio is varied and as the minimum angular separation of sources decreases. They also present different computational loads as the number of antenna elements and number of tracked signals is varied.

The OPERA algorithm performed best under low SNR conditions and when signal

sources were narrowly separated (i.e. highly correlated). This indicates that it was the most robust algorithm out of those tested. However, the robustness comes at the cost of the highest complexity, particularly when many signals need to be tracked.

The PASTd algorithm performs similarly to OPERA when the SNR is very low, although it behaves less favourably when the SNR is large. It is also the worst performing method when signals are highly correlated, for example when signals are narrowly separated. This may be due to the fact that it does not produce strictly orthonormal estimates of the eigenvectors. However, it requires the least computational power out of all the algorithms tested.

The FDPM algorithm appears to be the most sensitive to noise when the SNR is negative, although it improves as the SNR becomes positive. It deals with correlated signals slightly better than PASTd. FDPM is the only algorithm tested that is able to give estimates of the noise subspace. This is important when superresolution direction-finding results are required. However, the FDPM noise subspace algorithm appears to be most sensitive to noise out of the methods tested.

The field tests in Pretoria gave some initial indication that the subspace tracking methods may be useful for practical direction-finding. However, many of the results showed ambiguities and errors that may have been due to multipath propagation in the dense urban environment. The tests confirmed that compensation for mutual coupling between antenna elements is necessary if the elements are closely spaced. The reference eigen decomposition performed better than any of the reduced complexity tracking algorithms analysed, although the OPERA, PASTd and FDPM signal subspace methods appeared to perform similarly, with errors chiefly at certain of the locations. The OPERA algorithm did appear to perform more poorly at one particular location (location 2). The FDPM noise subspace algorithm gave the worst bearing estimates out of the methods tested. It is suggested that future tests alternately eliminate some of the variables (such as multipath propagation and antenna grounding), to further investigate their influence on the bearing estimates.

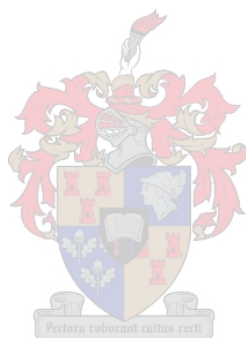
Simulations of the field tests produced satisfactory results and indicated that essential ambiguities in the antenna configuration or errors in the DF implementation were not possible causes of the poor DF results recorded by the field tests.

Bibliography

- [1] AKAIKE, H., “A new look at the statistical model identification.” *IEEE Transactions on Automatic Control*, December 1974, Vol. 19, No. 6, pp. 716–723.
- [2] DOUKOPOULOS, X. G. and MOUSTAKIDES, G. V., “The fast data projection method for stable subspace tracking.” in *Proceedings 13th European Signal Processing Conference, EUSIPCO 2005*, (Antalya, Turkey), September 2005.
- [3] GODARA, L. C., “Application of Antenna Arrays to Mobile Communications, Part II: Beam-Forming and Direction-of-Arrival Considerations.” in *Proceedings of the IEEE*, vol. 85, no. 8, pp. 1195–1245, The Institute of Electrical and Electronics Engineers, Inc. (IEEE), August 1997.
- [4] GOLUB, G. H. and LOAN, C. F. V., *Matrix Computations, Third Edition*. The Johns Hopkins University Press, 1996.
- [5] Google™, “Google™ Earth, version 4.0.2019 beta.” <http://earth.google.com/>, 2006. Aerial footage as of 26 November, 2006.
- [6] HUDSON, J. E., *Adaptive Array Principles*. The Institution of Engineering and Technology (IET), 1981.
- [7] Intel® Corporation, “Vector Math Library (VML) for Intel® Architecture Processors: Performance and Accuracy Data.” <http://www.intel.com/software/products/mkl/data/vml/vmldata.htm>. (Part of the Intel® Math Kernel Library 9.0 documentation).
- [8] JOHNSON, D. H. and DUDGEON, D. E., *Array Signal Processing*. PTR Prentice Hall, 1993.
- [9] KRIM, H. and VIBERG, M., “Two Decades of Array Signal Processing Research: The Parametric Approach.” *IEEE Signal Processing Magazine*, July 1996, pp. 67–94.
- [10] LEON-GARCIA, A., *Probability and Random Processes for Electrical Engineering, Second Edition*. Addison-Wesley Publishing Company, 1994.

- [11] MACINNES, C. S., “Fast, accurate subspace tracking using operator restriction analysis.” in *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech, and Signal Processing, 1998. ICASSP '98.*, vol. 3, pp. 1357–1360, The Institute of Electrical and Electronics Engineers, Inc. (IEEE), 1998.
- [12] NETLIB, “Netlib Repository at UTK and ORNL.” <http://www.netlib.org/>.
- [13] ORFANIDIS, S. J., “Electromagnetic Waves and Antennas.” <http://www.ece.rutgers.edu/~orfanidi/ewa/>, June 2004. MATLAB code is supplied along with the book, which is only available online.
- [14] PROAKIS, J. G. and MANOLAKIS, D. G., *Digital Signal Processing: Principles, Algorithms, and Applications, Third Edition*. Prentice Hall, 1996.
- [15] Radar Recollections - A Bournemouth University / CHiDE / HLF project, “After the Daventry Demonstration - Robert Watson Watt.” http://histru.bournemouth.ac.uk/Oral_History/Talking_About_Technology/radar_research/robert_watson_watt.html.
- [16] RAPPAPORT, D. T. S. (Ed.), *Smart Antennas: Adaptive Arrays, Algorithms, & Wireless Position Location*. The Institute of Electrical and Electronics Engineers, Inc. (IEEE), 1998.
- [17] RISSANEN, J., “Modeling by shortest data description.” *Automatica*, 1978, Vol. 14, pp. 465–471.
- [18] STRANG, G., “Linear Algebra Class Lecture Videos.” <http://web.mit.edu/18.06/www/Video/video-fall-99.html>, 1999.
- [19] SVANTESSON, T., “Methods of Mitigating the Effects of mutual Coupling in Antenna Arrays: A Signal Processing Approach.” *Proc. Radio Vetenskap och Kommunikation*, June 1999, pp. 441–445. Also available at: http://db.s2.chalmers.se/download/publications/svantesson_522.pdf.
- [20] U.S. National Geophysical Data Center, “Magnetic Declination.” <http://www.ngdc.noaa.gov/seg/geomag/declination.shtml>.
- [21] VAN TREES, H. L., *Optimum Array Processing*. No. IV in Detection, Estimation and Modulation Theory. Wiley-Interscience, 2002.
- [22] VINCENTY, T., “Direct and Inverse Solutions of Geodesics on the Ellipsoid with Application of Nested Equations.” *Survey Review*, April 1975, Vol. 23, No. 176, pp. 88–93. Also available at: http://www.ngs.noaa.gov/PUBS_LIB/inverse.pdf.

- [23] WAX, M. and SHEINVALD, J., “Direction finding of coherent signals via spatial smoothing for uniform circular arrays.” *IEEE Transactions of Antennas and Propagation*, May 1994, pp. 613–620.
- [24] YANG, B., “Projection Approximation Subspace Tracking.” *IEEE Transactions on Signal Processing*, January 1995, Vol. 43, No. 1, pp. 95–107.
- [25] YANG, J. and KAVEH, M., “Adaptive Eigensubspace Algorithms for Direction or Frequency Estimation and Tracking.” *IEEE Transactions of Acoustics, Speech and Signal Processing*, February 1988, Vol. 36, pp. 241–251.



Appendix A

MATLAB code for the selected subspace tracking methods

A.1 PASTd

```
function [W, C] = PASTd(v, InW, InC, Beta, K);

% Simulate the PASTd algorithm (Bin Yang) for iterative eigen decomposition
% (Projection Approximation Subspace Tracking with Deflation)
%
% v      : input array snapshot, corresponding to the most recent instant in time
% InW    : input matrix containing eigenvectors as columns
% InC    : input matrix containing the corresponding eigenvalues
%         on the main diagonal
% Beta   : forgetting factor
% W      : output matrix containing eigenvectors as columns
% C      : output matrix containing the corresponding eigenvalues
%         on the main diagonal

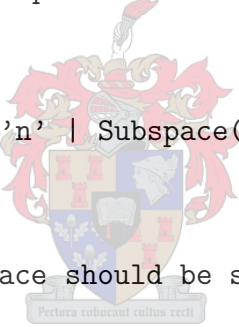
c = diag(InC);
W = InW;
for k = 1:1:K % calculate all M eigenvalues/eigenvectors
    x = (W(:,k))'*v1;
    e = v1 - W(:,k)*x;
    W(:,k) = W(:,k) + e*(x'/c(k));
    v = v - W(:,k)*x;
end;
C = diag(c); % return eigenvalues as a diagonal matrix
```

A.2 FDPDM

```
function [W] = FDPDM(v, InW, Beta, Subspace);

% Fast DPM algorithm for iterative eigen decomposition
% (Doukopoulos and Moustakides)
%
% v          : input array snapshot, corresponding to the most recent
%              instant in time
% InW        : input matrix containing eigenvectors as columns (M x K)
% Beta       : forgetting factor
% W          : output matrix containing eigenvectors as columns (M x K)
% Subspace   : 's' or 'n', corresponding to the signal or noise subspace

[M, K] = size(InW);
if (Subspace(1) == 's' | Subspace(1) == 'S')
    Signal = 1;
else
    if (Subspace(1) == 'n' | Subspace(1) == 'N')
        Signal = 0;
    else
        disp('Subspace should be specified as signal or noise.');
```



```
    end;
end;

mu = 1 - Beta;
r = InW' * v;
if (Signal == 1)
    T = InW + (mu / norm(v).^2) * v * r';
else
    T = InW - (mu / norm(v).^2) * v * r';
end;
a = r - norm(r) .* eye(K, 1);
Z = T - (2/(norm(a).^2)) * (T*a)*a';
for k = 1:K
    Z(:,k) = Z(:,k) .* (1/norm(Z(:,k)));
end;
W = Z;
```

A.3 EVD-OPERA

```
function [ww, llam] = opera(ww, llam, alpha1, alpha2, v)

% Operator Restriction Analysis (OPERA)
%
% This routine restricts an M x M covariance matrix to a
% subspace of dimension (K+1) defined by the span of the K
% current vecs and the data snapshot, v. If K signals are
% present and there is only uncorrelated noise, the
% covariance matrix is of rank K.
% Inputs:
% ww, LxM; the M orthonormal signal subspace vecs and
% llam (a column vector), the corresponding evals. The
% entries of llam are arranged in decreasing order of
% magnitude.
% alpha1, alpha2: the forgetting factors
% v, the data snapshot
% Note: If the signal subspace dimension, M has been
% overestimated, it will be incorporated into the
% new matrix.
% If the signal subspace dimension, M has been
% underestimated, no action is taken.

[M, K] = size(ww)

% KM flops here.
ss = ww'*x;

% KM flops here.
s = x - ww*ss;

ssm = norm(s);

% ss is a column vector; it is the data snapshot in local
% coordinates.
ss = [ss' ssm']';
```

APPENDIX A. MATLAB CODE FOR THE SELECTED SUBSPACE TRACKING METHODS

```
% Add a new column to W to handle the case when the rank-one
% update causes an increase in rank by 1
% (i.e. when the new data vector lies partially outside of
% the column space of W).
ww = [ww s/ssm];

% B is the covariance matrix in local coordinates. B
% is always diagonal, with a zero at the bottom of the
% diagonal.
B = diag([llam' 0]');

% Calculate the eigen decomposition of the rank-one update
% in the transformed coordinate system.
% eeig() is a version of eig() that orders the evals.
[wx, llamx] = eeig(alpha1*B+alpha2*ss*ss');

% Now, find the evecs in global coordinates. This is
% the most costly step;  $MK^2$ .
wwx = ww*wx;

% Drop the evec with a close to zero eval, and drop the
% corresponding eval from the vector of evals.
wwx(:, M+1) = [];
llamx(M+1) = [];
ww = wwx;
llam = llamx;
```

Appendix B

Some notes regarding the software simulator

As a first stage of construction, classes were created to support vector and matrix algebra. The cornerstone classes are `cRealMatrix` and `cComplexMatrix`, both derived from `cMatrix`. Matrix multiplication, Singular Value Decomposition, QR factorisation and other CPU intensive methods make use of the Intel® Performance Primitives and Math Kernel Library. Operators are overloaded so that the syntax resembles The MathWorks MATLAB® language as closely as possible. The matrix classes are constructed so that temporaries are created only when absolutely necessary, to avoid many spurious `memcpy` operations. Other classes were added to provide facilities such as thread-safe queues (`cPipeSharedBuffer`) and processing elements or “blocks” (`cProcessBlock`). `cProcessBlock` elements can be connected using the concept of “ports” (`cPort`) and “pipes” connecting ports (`cPipe`), making possible a style of programming similar to that provided by The MathWorks Simulink®. The `cBufferManager` class allocates temporary buffers (per-thread) as needed, performing lazy deletion so that fewer calls are made to `new` and `delete`. Almost all classes are derived from a single ancestor (`cBase`) and multiple inheritance has been almost entirely avoided. The `cSensorArray` class handles antenna configurations and calculates response vectors for parameterised scenarios. The `cSignalSinusoidal`, `cSignalModAudio`, `cNoiseNB` (narrowband noise) and `cNoiseWB` (wideband noise) represent possible signal sources and are all derived from a common ancestor (`cSignalSource`).

The central class responsible for signal simulation is `cSignalGen`. It repeatedly generates instances of `cIQchunk`, which contain IF data:

```
void cSignalGen::Generate(shared_ptr<cIQchunk> pIQchunk)
{
    complex<float> * pcfIQ = (complex<float> *) (pIQchunk->m_IQchunk);
    unsigned uNumChannels = m_pSensorArray->GetNumChannels();

    pIQchunk->m_IFtag.m_lTimestamp_microSec = m_lTime_uSec;
    pIQchunk->m_IFtag.m_uNumChannels = uNumChannels;
    pIQchunk->m_IFtag.m_lSampleRate_uHz = m_lBandwidth_Hz * int64_t(1e6);
    pIQchunk->m_IFtag.m_lBandwidth_uHz = m_lBandwidth_Hz * int64_t(1e6);
    pIQchunk->m_IFtag.m_f_dBFS = 0.0f;
    pIQchunk->m_IFtag.m_fTotalGain_dB = 0.0f;
    pIQchunk->m_IFtag.m_bDiscontinuity = (m_lFramePeriod_uSec !=
        m_lFrameSize_uSec);
    for (unsigned uCh = 0; uCh < uNumChannels; uCh++)
        pIQchunk->m_IFtag.m_plCentreFreq_uHz[uCh] =
            m_lCentreFreq_Hz * int64_t(1e6);

    memset(pcfIQ, 0, uNumChannels * m_lFrameSize_samples *
        sizeof(complex<float>));

    // Generate each signal impinging on the array in turn
    shared_ptr<cCollection::cIterator> pSignalIter;
    pSignalIter = m_Signals.CreateIterator();
    for (pSignalIter->Reset(); !pSignalIter->AtEnd(); (*pSignalIter)++) {
        shared_ptr<cSignalSource> pSignalSource =
            (shared_dynamic_cast<cSignalSource>(
                pSignalIter->pItem()));
        pSignalSource->AddChunk(pcfIQ, shared_ptr<cSignalGen>(this,
            NullDeleter()));
    }

    // increment time by the specified frame period
    m_lTime_uSec += m_lFramePeriod_uSec;

    // Sleep, if we have time (allow 10ms preemption time)
    if (m_lFramePeriod_uSec != m_lFrameSize_uSec)
        Sleep((m_lFramePeriod_uSec - m_lFrameSize_uSec)/1e3f);
}
```

}

AddChunk is overridden in each cSignalSource descendant.

The complete list of classes used by the simulator project is as follows:

cAngle	cFrequency	cQueue
cAngle3D	cGLGraph	cRealMatrix
cAppGUI	cIFtag	cSensorArray
cArray	cIQchunk	cSignalGen
cBase	cList	cSignalModAudio
cBlockAudio	cMathCore	cSignalSinusoidal
cBlockAudioIn	cMatrix	cSignalSource
cBlockAudioOut	cNoiseEnvironment	cSocket
cBlockBeamforming	cNoiseNB	cSocketClient
cBlockCalibrate	cNoiseWB	cSocketServer
cBlockEigenDecomp	cNull	cSocketSession
cBlockMUSIC	cOverlapAdd	cSortableArray
cBlockPXGF	cPipe	cSortableList
cBlockPXGFfile	cPipeSharedBuffer	cSpectralDisplay
cBlockPXGFsocket	ccPipeSocket	cString
cBufferManager	cPlot	cSubspace
cCollection	cPort	cTokeniser
cComplexMatrix	cPortBinaryPipe	cTrajectory
cConvert	cPortLossyQueue	cTrajectoryOrbit
cCoord	cPortSharedQueue	
cFactory	cProcessBlock	