Publication Year	2010
Acceptance in OA	2024-01-23T16:53:22Z
Title	GNSS Reconfigurable Antenna Based Enhanced Localization - WP3: Baseband processing and beamforming - D3.2: Beamforming algorithms for GNSS receivers
Authors	Lehmann, Christoph, Consoli, Angelo, Iera, Christian, Moriggia, Lorenzo, Vieira, Rui, Rieg, Dieter, Genghi, Angelo, Ghiggi, Athos, Piazza, Francesco, Materni, Roberto, Francescato, David, NALDI, Giovanni
Handle	http://hdl.handle.net/20.500.12386/34601
Volume	WP3 - D3.2

Project Number: FP7 GRABEL_232016
{REF RTD REG/T.4(2008)D/566378}

Project Title: GNSS Reconfigurable Antenna Based Enhanced Localization

Deliverable Type (Pub/Int)*: Pub

GNSS Reconfigurable Antenna Based Enhanced Localization

FP7 GRABEL_232016 {REF RTD REG/T.4(2008)D/566378}

WP3: Baseband processing and beamforming

D3.2: Beamforming algorithms for GNSS receivers

Contractual Date of Delivery to the CEC: 20/06/2010

Actual Date of Delivery to the CEC: 20/06/2010

Author(s): Christoph Lehmann[1], Angelo Consoli[1], Christian Iera[1], Lorenzo Moriggia[1], Rui Vieira[1], Dieter Rieg[1], Angelo Genghi[2], Athos Ghiggi[2], Francesco Piazza[2], Roberto Materni[2], David Francescato[2], Giovanni Naldi[3]

Participant(s): [1]EXYS, [2]SYN, [2]INAF

Workpackage: WP3: Baseband processing and beamforming

Est. person months: 13 (1^1, 5^2,7^3)

Nature: P (Nature: P-prototype, R-report, O-other)

Dissemination: Int (Type: Pub-public, Int-internal)

Version: 1.10

Total number of pages: 140

Abstract:

The intention of this deliverable is to give an update to all the members of the consortium about the software implementation of the project. A general description of the selected adaptive beamforming algorithm, i.e. MVDR (Minimum Variance Distortionless Response) algorithm, is presented with some implementation notes and first results of tests. In addiction this deliverable contains the SY1031 software implementation notes and time schedule needed to produce it. It gives also a complete description adopting the common UML techniques to show the new design decision and the implementation decision taken and applied during the realizations of the beamforming algorithms for GNSS receivers software. Moreover the performed test and simulation sessions have been described, showing the qualitative and quantitative results.

Keyword list: GNSS receiver, Reconfigurable antenna, beamforming techniques, test, simulation

Symbols

θ	Aerial RF source (SV or interferer) elevation
Ω	Aerial RF source (SV or interferer) azimuth
β	Vehicle heading
N	Number of antennas
$\|_{SV}$	Subscript used to indicate the channel assigned GNSS satellite
$\|_{Int}$	Subscript used to indicate the channel interferes
λ	GPS L1 wavelength

Abbreviations

NYI	Not Yet Implemented
TBC	To Be Confirmed
TBD	To Be Defined

Executive Summary

The aim of this deliverable is to present the already concluded activities (and also the ones still in progress) in the context of the beamforming algorithms for GRABEL.

Chapter 2 of the document deals with a comparison study of some suitable algorithms for GNSS receivers highlighting positive and negative aspects for each of them.

Not only deterministic beamforming methods like *Beamsteering* and *Null Steering* are described, but also adaptive algorithms like LCMV *(Linearly Constrained Minimum Variance)*, *MVDR (Minimum Variance Distortionless Response)* and *GSC (Generalized Sidelobe Canceller)*.

Particular attention is dedicated to the candidate algorithm chosen for the implementation in the GRABEL receiver: i.e. the MVDR algorithm.

Chapter 3 describes in a detailed way (using multiple views) all the new features implemented and also the integration of them inside the project context.

Chapter 4 describes the software implementation in C and the results of the timing tests, as well as the SY1030 platform and the GRABEL beamformer with his different configurations. A C Code software program that implements the MVDR algorithm for a linear array has been developed in order to have first information about the achievable performances in terms of computation time of the beamformer coefficients.

Chapter 5 introduces the test and simulation procedures and explains in detail the performed basic and live tests.

Preliminary considerations about integration of the beamforming network in hardware are introduced in the chapter 6, taking into consideration both the system performances and the hardware complexity.

The last part of the document, chapter 7, gives a hint about the roadmap for the future work.

Table of contents

SYMBOLS 2

ABBREVIATIONS 2

1 INTRODUCTION 9

2 THE MOST SUITABLE BEAMFORMING ALGORITHMS FOR GRABEL 11

2.1 BEAMSTEERING AND NULL STEERING ALGORITHMS 12
2.2 LCMV (LINEARLY CONSTRAINED MINIMUM VARIANCE) BEAMFORMING ALGORITHM 13
2.2.1 MVDR (Minimum Variance Distortionless Response) Beamforming Algorithm 15
2.3 GSC (GENERALIZED SIDELOBE CANCELLER) BEAMFORMING ALGORITHM 16

3 SOFTWARE DEVELOPMENT 18

3.1 DESIGN DECISION 18
3.1.1 Main scenario for beamforming algorithms for GNSS receivers 18
3.1.2 Implementation static structure - Principal component 18
3.1.3 Implementation static structure – Class diagram 19
3.1.4 Dynamic behaviour - statechart 20
3.1.5 Dynamic behaviour – Sequence diagram main scenario 21

4 IMPLEMENTATION NOTES 23

4.1 IMPLEMENTATION OF MVDR ALGORITHM IN C LANGUAGE 23
4.1.1 Preliminary remarks on the software implementation 23
4.1.2 Results 24
4.2 NEW PRODUCTS BOARD AND PLATFORM DEFINITIONS 25
4.2.1 SY1030 Platform 25
4.2.2 GRABEL beamformer Product 25
4.3 ALGORITHMS IMPLEMENTATION NOTES 26
4.3.1 Nominal Mode No Interferer 26
4.3.2 Nominal Mode Interferer enabled 26
4.3.3 Calibration 27
4.3.4 Hosted Mode 32
4.4 CORE SOFTWARE CHANGES 33
4.4.1 CN0 and Signal Level Based Thresholds 33
4.4.2 Note on RQ250 and Noise Floor Normalization 33
4.4.3 Notes on the New Classes 34
4.4.4 Beamforming Scheduling 34
4.5 SY1031 BEAMFORMING 35
4.6 CXY ANTENNA ARRAY CONFIGURATION IN CARTESIAN COORDINATES 36
4.7 CPO ANTENNA ARRAY CONFIGURATION IN POLAR COORDINATES 36
4.8 ATT ANTENNA ARRAY ATTITUDE 37
4.9 INT INTERFERER POSITION 37
4.10 CAL CALIBRATION PARAMETERS SECTION 37
4.11 PH1 USER DEFINED BEAM-FORMER PARAMETERS 38
4.12 PH2 USER DEFINED BEAM-FORMER PARAMETERS 39

5 TEST 40

5.1 INTRODUCTION 40
5.2 BASIC TESTS 41
5.2.1 Calibration Sequence Tests 41

5.2.2 User Defined Antenna Array Patterns 44
5.2.3 Nominal Mode Tests 48
5.2.4 Basic Tests Pass/Fail Table 114
5.3 LIVE TESTS 115
5.3.1 Calibration Test Using the Simulator 115
5.3.2 Further Calibration Tests Using the Simulator 118
5.3.3 Calibration Temperature Sensitivity Assessment 121
5.3.4 Calibration Connecting Simulator to Antenna, using Ant. Array to Receive Signal 123
5.3.5 Nominal mode test without interfering signal 129
5.3.6 New Calibration – Live Test Using the Simulator 134
6 BEAMFORMING NETWORK IN HARDWARE INTEGRATION: PRELIMINARY STUDY 137
7 PLANNED DEVELOPMENTS 139
8 SUMMARY/CONCLUSION 139
9 REFERENCE DOCUMENTS 140
10 BIBLIOGRAPHY 140

List of figures

Figure 1.1: two different methods for creating a beam: narrowband and wideband beamforming 10
Figure 2.1: a plane wave with DOA θ and frequency ω is received by a linear array of N antennas 11
Figure 2.2: block diagram of the MVDR beamforming algorithm 15
Figure 2.3: simplified block diagram of the GSC beamforming algorithm 16
Figure 3.1: Use case diagram – Main scenario 18
Figure 3.2: System description – Components diagram 19
Figure 3.3: Beamformer Computation Class Diagram 19
Figure 3.4: Statechart - Beamformer Computation Module Class 20
Figure 3.5: Sequence 1 – simple test sequence diagram 21
Figure 3.6: Sequence 2 - antenna array calibration 21
Figure 3.7: Sequence 3 - Nominal mode 22
Figure 4.1: The principle of the proposed calibration procedure for the GRABEL receiver. 28
Figure 4.2: Example of an anechoic room. 28
Figure 4.3: sequence of steps on which the proposed calibration procedure is based. 30
Figure 4.4: Default Built-In Antenna Configuration 36
Figure 4.5: Geometric offset definition 38
Figure 4.6: Body frame rotation wrt ENU frame described by the heading and pitch angles. 39
Figure 5.1: Antenna Array Map 49
Figure 5.2: Nominal Mode Test Antenna Pattern For Channel 2 50
Figure 5.3: Nominal Mode Test Antenna Pattern For Channel 3 50
Figure 5.4: Nominal Mode Test Antenna Pattern For Channel 4 51
Figure 5.5: Nominal Mode Test Antenna Pattern For Channel 6 51
Figure 5.6: Nominal Mode Test Antenna Pattern For Channel 7 52
Figure 5.7: Nominal Mode Test Antenna Pattern For Channel 9 52
Figure 5.8: Nominal Mode Test Antenna Pattern For Channel 10 53
Figure 5.9: Nominal Mode Test Antenna Pattern For Channel 11 53
Figure 5.10: Nominal Mode Test Antenna Pattern For Channel 14 54
Figure 5.11: Nominal Mode Test Antenna Pattern For Channel 16 54
Figure 5.12: Antenna Array Map 56
Figure 5.13: Antenna Pattern for Channel 2 57
Figure 5.14: Antenna Pattern for Channel 3 58
Figure 5.15: Antenna Pattern for Channel 4 59
Figure 5.16: Antenna Pattern for Channel 6 60
Figure 5.17: Antenna Pattern for Channel 7 61
Figure 5.18: Antenna Pattern for Channel 9 62
Figure 5.19: Antenna Pattern for Channel 10 63
Figure 5.20: Antenna Pattern for Channel 11 64
Figure 5.21: Antenna Pattern for Channel 14 65
Figure 5.22: Antenna Pattern for Channel 16 66
Figure 5.23: Antenna Array Map 68
Figure 5.24: Antenna Pattern for Channel 2 69
Figure 5.25: Antenna Pattern for Channel 3 70
Figure 5.26: Antenna Pattern for Channel 4 71
Figure 5.27: Antenna Pattern for Channel 6 72
Figure 5.28: Antenna Pattern for Channel 7 73
Figure 5.29: Antenna Pattern for Channel 9 74
Figure 5.30: Antenna Pattern for Channel 10 75
Figure 5.31: Antenna Pattern for Channel 11 76
Figure 5.32: Antenna Pattern for Channel 14 77

Figure 5.33: Antenna Pattern for Channel 16 78
Figure 5.34: Antenna Array Map 80
Figure 5.35: Antenna Pattern for Channel 2 81
Figure 5.36: Antenna Pattern for Channel 3 82
Figure 5.37: Antenna Pattern for Channel 4 83
Figure 5.38: Antenna Pattern for Channel 6 84
Figure 5.39: Antenna Pattern for Channel 7 85
Figure 5.40: Antenna Pattern for Channel 9 86
Figure 5.41: Antenna Pattern for Channel 10 87
Figure 5.42: Antenna Pattern for Channel 11 88
Figure 5.43: Antenna Pattern for Channel 14 89
Figure 5.44: Antenna Pattern for Channel 16 90
Figure 5.45: Antenna Array Map 92
Figure 5.46: Antenna Pattern for Channel 2 93
Figure 5.47: Antenna Pattern for Channel 3 94
Figure 5.48: Antenna Pattern for Channel 4 95
Figure 5.49: Antenna Pattern for Channel 6 96
Figure 5.50: Antenna Pattern for Channel 7 97
Figure 5.51: Antenna Pattern for Channel 9 98
Figure 5.52: Antenna Pattern for Channel 10 99
Figure 5.53: Antenna Pattern for Channel 11 100
Figure 5.54: Antenna Pattern for Channel 14 101
Figure 5.55: Antenna Pattern for Channel 16 102
Figure 5.56: (part 1) Heading profile 108
Figure 5.57: (part 1)Antenna array amplitude gain in static and dynamic conditions (no antenna gain mismatch) 108
Figure 5.58: (part 1)Antenna array attenuation in static and dynamic conditions (no antenna gain mismatch 109
Figure 5.59: (part 2) Heading profile. 109
Figure 5.60: (part 2)Antenna array amplitude gain in static and dynamic conditions with antenna gain mismatch compensation. 110
Figure 5.61: (part 2)Antenna array attenuation in static and dynamic conditions with antenna gain mismatch compensation 110
Figure 5.62: Antenna Array Map 111
Figure 5.63: Antenna Pattern for Channel 16 : unmodelled antenna gains mismatch 112
Figure 5.64: Antenna Pattern for Channel 16 : unmodelled antenna gains mismatch 112
Figure 5.65: Antenna Pattern for Channel 16 : modelled antenna gains mismatch 113
Figure 5.66: Antenna Pattern for Channel 16 : modelled antenna gains mismatch 113
Figure 5.67: 1 Phase Offsets for RF Inputs 2 - 4 115
Figure 5.68: Phase Offsets for RF Inputs 5 – 7 116
Figure 5.69: "De-biased" Phase Offsets for RF Inputs 2 - 7 116
Figure 5.70: Phase Offset Samples Standard Deviation for RF Inputs 2 - 7 117
Figure 5.71: Phase Offsets for RF Inputs 2 – 4 118
Figure 5.72: Phase Offsets for RF Inputs 5 - 7 119
Figure 5.73: "De-biased" Phase Offsets for RF Inputs 2 - 7 119
Figure 5.74: Phase Offset Samples Standard Deviation for RF Inputs 2 – 7 120
Figure 5.75: Phase Offsets for RF Inputs 2 - 4 121
Figure 5.76: Phase Offsets for RF Inputs 5 - 7 122
Figure 5.77: Phase Offsets for RF Inputs 2 - 4 123
Figure 5.78: Phase Offsets for RF Inputs 5 - 7 124
Figure 5.79: "De-biased" Phase Offsets for RF Inputs 2 - 7 125
Figure 5.80: Phase Offset Samples Standard Deviation for RF Inputs 2 – 7 126
Figure 5.81: SNR and Noise Floor vs Time 127
Figure 5.82: Pseudorange and Range Rate Difference 128
Figure 5.83: Map of the Live Test location 129

Figure 5.84: Test Setup at Saphyrion Premises using an antenna to re-irradiate the GPS simulator signal 130
Figure 5.85: Results of the first test. In nominal mode, and with the right attitude, the CN0 is boosted at least by 5 dBHz. 130
Figure 5.86: Results of the second test. While in nominal mode, after a counterclockwise rotation of 90 deg, the correct antenna pattern is restored setting the heading to 270 deg with a new attitude command. 131
Figure 5.87: Synthetic antenna pattern without interference rejection 132
Figure 5.88: Synthetic antenna pattern with interference rejection 133
Figure 5.89: Phase Offsets for RF Inputs 2 - 4 134
Figure 5.90: Phase Offsets for RF Inputs 5 – 7 134
Figure 5.91: "De-biased" Phase Offsets for RF Inputs 2 – 7 135
Figure 5.92: Phase Offset Samples Standard Deviation for RF Inputs 2 – 7 135
Figure 5.93: CN0 and Noise Floor V.S. Time 136
Figure 5.94: Estimated Calibration Parameters 136
Figure 6.1: architecture of the GRABEL GPS Base-Band Processor 137
Figure 6.2: scheme of principle for the integration of beamforming network in hardware 138

List of tables

Table 4.1: results of the timing tests 24
Table 4.2: C++ files added to the project and MATLAB files used for the development 35
Table 4.3: Attitude Sources for Nominal and Test Mode. 37
Table 5.1: Calibration on 9 coincident antennas 41
Table 5.2: Test A2 Calibration on an antenna pattern, cross shape plus central antenna 42
Table 5.3: Test A3 sequence. Calibration of a subset of the full antenna pattern 43
Table 5.4: Test B1 User defined channel beamformer configuration 44
Table 5.5: B2 Test sequence. Antenna Array Pattern Defined Using azimuth and elevation 45
Table 5.6: B3 test sequence. Raw Beamformer Registers Data Read and Write Operations 46
Table 5.7: C1 Nominal mode test sequence 48
Table 5.8: Test C2 Nominal mode test sequence 55
Table 5.9: C3 Nominal mode test sequence and expected results example 67
Table 5.10: C4 Nominal mode test sequence and expected results example 79
Table 5.11: C5 Nominal mode test sequence and expected results example 91
Table 5.12: C6 Nominal mode test sequence and expected results example 103
Table 5.13: C7 Nominal mode test sequence 105
Table 5.14: Tests Results Summary Table. (*) Tests covered by Live Tests 114
Table 9.1: GRABEL reference documents 140
Table 10.1: Bibliography 140

1 Introduction

When both the desired signal and the interferences simultaneously occupy the same band of frequencies, it certainly is not possible to exploit only the time filtering in order to isolate the useful signal; nevertheless, since the wanted signal and the interfering ones usually come from different spatial regions, consequently this spatial diversity can be exploited for this purpose using a receiving spatial filter.

So the term beamforming means the technique by which it is realized a versatile form of spatial filtering, separating out signals that are spectrally overlapped but that come from different spatial directions.

Basically a beamformer is a system used in combination with an array of antennas that realizes, as said before, a form of spatial filtering and it can be, depending on its implementation, analogue or digital. In this document we will refer only to digital beamforming.

A great advantage of the spatial filtering is represented by its versatility.

In many applications it is often necessary to update in real time the spatial filtering function in order to maintain an effective suppression of the interfering signals. This updating can be easily implemented in a discretely sampled system, simply changing the way the beamformer linearly combines the data coming from the antennas.

Array beamforming techniques also allow to simultaneously form a certain number of beams exploiting analogue or digital processing methods. The beams can be formed so that they offer high gain and low sidelobes, or controlled beamwidth. Adaptive beamforming techniques adjust on the fly the array beampattern in order to optimize some characteristics of the received signal. In beam scanning, a single main beam of an array can be continuously steered within the field of view (beam size of the single antenna element).

Antenna arrays using adaptive beamforming techniques can reject interfering signals by pointing the radiation nulls towards the direction of arrival of the interferers.

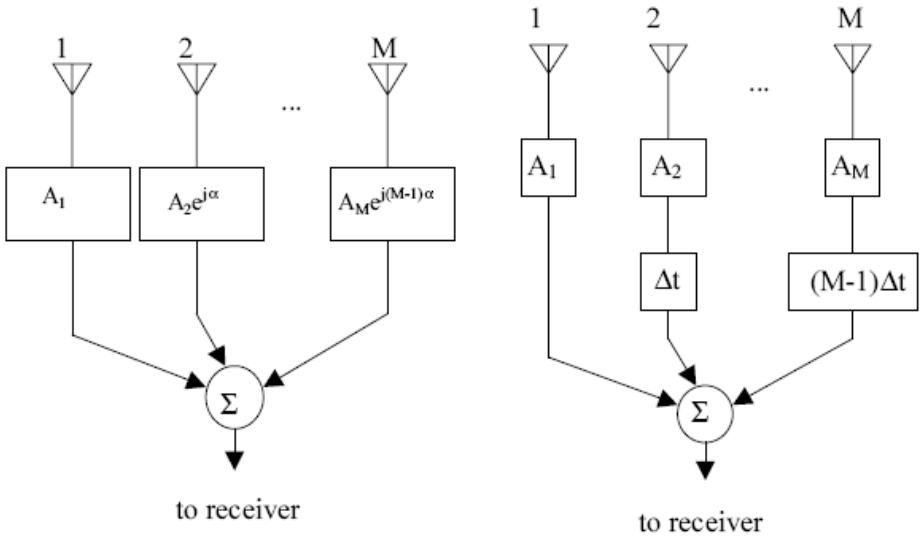


Figure 1.1: two different methods for creating a beam: narrowband and wideband beamforming

In figure 1.1 two different methods to produce a beam are reported. In the case of arrays whose working bandwidth is narrow it is possible to handle just the phases in order to create the beam. In the case of arrays whose working bandwidth is wide it is requested to equalize with an extreme accuracy the arrival time of the signals using variable time delay.

2 The most suitable beamforming algorithms for GRABEL

In order to effectively introduce the most suitable beamforming algorithms for GRABEL, it's convenient (for simplicity reasons) to consider a generic linear array whose elements are equally spaced (with an inter-element spacing of *d*) without loss of generality.
We can also assume that the signal received by the array is a plane wave, with *Direction of Arrival* (*DOA*) θ and frequency ω, and we can consider the signal received by the first element of the array as the reference with null phase.
The *beamformer response* to this kind of signal can be easily expressed in vector form:

$$r(\theta,\omega) = \boldsymbol{w}^H \boldsymbol{d}(\theta,\omega) \tag{2–1}$$

$\boldsymbol{w}$ is the vector containing the beamformer coefficients that have to be multiplied by the time samples coming from the antennas, $(\cdot)^H$ is the Hermitian operator and $\boldsymbol{d}(\theta,\omega)$ is the *response vector of the array*, also called *steering vector* or *directional vector*.
With the hypothesis made before and supposing that the array is calibrated and the antennas of the array are ideal (omnidirectional antennas), the elements of the vector $\boldsymbol{d}(\theta,\omega)$ are:

$$\boldsymbol{d}(\theta,\omega) = \left[1 \quad e^{j\omega\tau_2(\theta)} \quad \cdots \quad e^{j\omega\tau_N(\theta)}\right]^H \tag{2–2}$$

where $\tau_i(\theta)$, for $2 \le i \le N$, are the time delays due to the signal propagation (see figure 2.1).

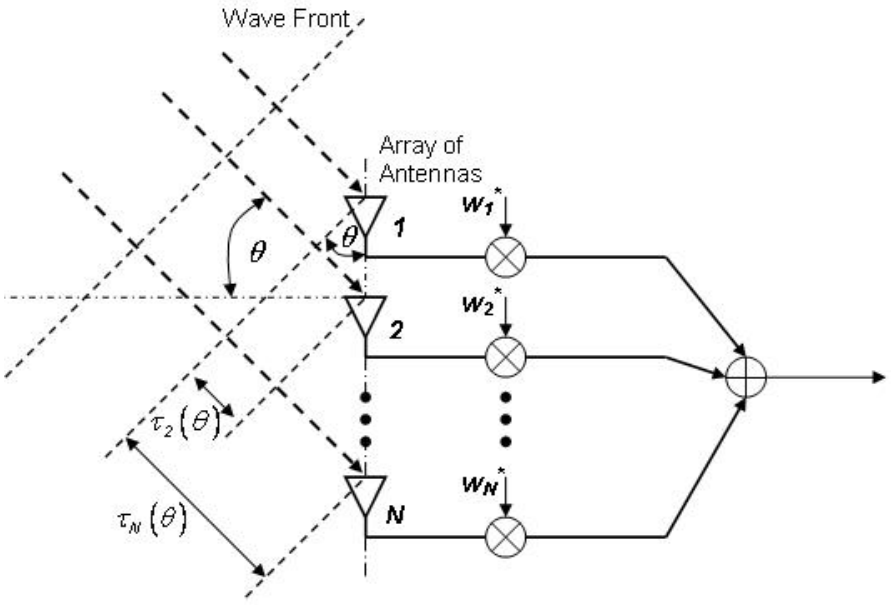


Figure 2.1: a plane wave with *DOA* θ and frequency ω is received by a linear array of *N* antennas

The ideal steering vector (2-2) takes into account the geometry of the array.
The non ideal characteristics of the antennas and receiver electronics (not calibrated electronics, directional antennas, …) can be included in $\boldsymbol{d}(\theta,\omega)$, multiplying it by a function $\boldsymbol{a}(\theta,\omega)$ that

depends on the response of each receiver chain (antennas, LNAs, …) for that particular *DOA* θ and frequency ω .

2.1 Beamsteering and Null Steering algorithms

The **Beamsteering or Classical Beamforming Algorithm** simply aims to separate a signal coming from a certain known direction θ_0 from the other received signals arriving from different directions.

Supposing the signal is narrowband (at frequency ω_0), the desired response of the beamformer is ideally equal to one for (θ_0, ω_0) whereas it is null elsewhere.

A common solution to this problem consists of taking the *steering vector* $\boldsymbol{d}(\theta_0, \omega_0)$ as the vector of the coefficients ***w***:

$$\boldsymbol{w} = \boldsymbol{d}(\theta_0, \omega_0) \tag{2–3}$$

It is demonstrated that this solution is the best choice in order to minimize the square error between the ideal and the real response.

In addition to the steering of the array beam in the direction of the desired signal, it is also possible to force some (limited in numbers) nulls of the beamformer response in the direction of potential undesired signals (i.e. Radio Frequency Interferences or RFIs).
The resulting beamforming algorithm is known as **Null Steering**.
If we suppose to have *K* interferers with different and well-known directions of arrival, the coefficients of the Null Steering algorithm can be calculated with the following expression:

$$\boldsymbol{w} = \boldsymbol{A}^H \left(\boldsymbol{A}\boldsymbol{A}^H \right)^{-1} \boldsymbol{u} \tag{2–4}$$

where:

$$\boldsymbol{A} = \left[\boldsymbol{d}(\theta_0)\ \boldsymbol{d}(\theta_1)\ \boldsymbol{d}(\theta_2)\ \ldots\ \boldsymbol{d}(\theta_K) \right]^H \tag{2–5}$$

$$\boldsymbol{u} = \left[1\ \ 0\ \ \ldots\ \ 0 \right]^H \tag{2–6}$$

Applying these particular coefficients to the input data it is possible not only to point the main beam towards the right direction θ_0, but also to place the radiation nulls in correspondence of the *K* undesired directions (RFIs).
The maximum number of RFIs that can be canceled out is *N* – 2 (*N* is the number of antennas of the array).

The Beamsteering and Null Steering algorithms basically are data-independent in the sense that the coefficients are chosen so that the beamformer response is as close as possible to the desired one (known a priori), regardless of the sequence of data or their statistics.
So they are deterministic algorithms: the directions of arrival of both the signal of interest and the interferers have to be well-known a priori for Null Steering; only the DOA of the desired signal in the case of Beamsteering, of course.

2.2 LCMV (Linearly Constrained Minimum Variance) beamforming algorithm

The **LCMV (Linearly Constrained Minimum Variance)** beamforming algorithm consists in applying linear constraints to the beamformer response so that the signal(s) of interest can be received with the wanted gains and phases and, at the same time, the possible interfering signal(s) can be canceled out. The beamformer coefficients are chosen in order to minimize the output power (or variance), respecting the constraints. In this way the desired signal(s) is (are) received properly whereas the output contribution due to noise and interfering signal(s) (with different directions of arrival) are minimized.
The use of linear constraints is a general approach that allows an extended control on the adapted response of the beamformer.

With the following linear constraint:

$$\boldsymbol{w}^H \boldsymbol{d}\left(\theta,\omega\right) = g \tag{2–7}$$

where *g* is a complex constant, every signal, coming from a DOA θ and having a frequency ω, is certainly received by the beamformer with an output response *g*.
In order to minimize the contribution due to both noise and interfering signal(s), with frequency ω but not coming from θ, the coefficients are calculated to make the statistical expected value of the output power (or variance):

$$E\left[\left|y\right|^2\right] = \boldsymbol{w}^H \boldsymbol{R}_x \boldsymbol{w} \tag{2–8}$$

minimum.
y is the output of the beamformer and $\boldsymbol{R}_x$ is the Auto-Covariance Matrix of the input data.
The LCMV algorithm can be expressed as a minimum search problem, that is:

$$\min_{\boldsymbol{w}} \boldsymbol{w}^H \boldsymbol{R}_x \boldsymbol{w} \tag{2–9}$$

according to this constraint:

$$\boldsymbol{d}^H\left(\theta,\omega\right)\boldsymbol{w} = g^* \tag{2–10}$$

To solve this constrained minimum problem, the Lagrange multipliers method can be used, obtaining this fundamental result:

$$\boldsymbol{w} = g^* \frac{\boldsymbol{R}_x^{-1}\boldsymbol{d}\left(\theta,\omega\right)}{\boldsymbol{d}^H\left(\theta,\omega\right)\boldsymbol{R}_x^{-1}\boldsymbol{d}\left(\theta,\omega\right)} \tag{2–11}$$

Note: in the real applications the presence of uncorrelated noise ensures that $\boldsymbol{R}_x$ can be inverted.

*If g=1 the last equation corresponds to the expression that calculates the beamformer coefficients of the **MVDR (Minimum Variance Distortionless Response) beamforming algorithm** (see section 2.2.1).*

The single linear constraint expressed in (2-10) can be easily extended to the case of more than one linear constraint that allow more control on the array beampattern.
For instance, if there is a fixed interference source whose signal arrives from a well-known direction ϕ, whereas the wanted signal arrives from the direction θ (known as well), it could be desirable to force the gain in direction ϕ to *0* maintaining at the same time the beamformer response equal to *g* in direction θ. This example can be formulated with the following expression:

$$\begin{bmatrix} \boldsymbol{d}^H(\theta,\omega) \\ \boldsymbol{d}^H(\phi,\omega) \end{bmatrix} \boldsymbol{w} = \begin{bmatrix} g^* \\ 0 \end{bmatrix} \tag{2–12}$$

If there are $L < N$ (*N*, the number of the array elements, is also the dimension of the vector space which contains the vectors ***w*** and ***d***) linear constraints on ***w***, they can be expressed as:

$$\boldsymbol{C}^H \boldsymbol{w} = \boldsymbol{f} \tag{2–13}$$

where the matrix ***C*** (with dimension $N \times L$) and the vector ***f*** (with dimension *L*) are called respectively *Matrix of Constraints* and *Response Vector*. The constraints are supposed to be linearly independent, so that ***C*** has maximum rank *L*.

Design of the constraints

The choice of the *Matrix of Constraints* and the *Response Vector* can be made following different methods.
Here below some of these methods are presented.
In many applications a combination of different kinds of methods is used for practical reasons.
Every linear constraint exploits a freedom degree of the coefficients vector.
With an array composed by *N* antennas the coefficients vector has in total *N* freedom degrees.
So with *L* constraints only *N - L* freedom degrees can be used to minimize the variance.

Point constraints

They affect the beamformer response only in particular spatial directions and/or frequencies. The maximum number of points in which the response can be constrained is limited at maximum to *N - 1*.
If all *N - 1* constraints are used consequently no other freedom degree still remains to minimize the variance.

Derivative constraints

They affect the beamformer response in a set of spatial directions and/or frequencies forcing the derivative of the beamformer response to 0 in correspondence to these directions and/or frequencies. Usually they are applied together with the Point Constraints. The Derivative Constraints are useful for example when the *DOA* of the wanted signal is known a priori with uncertainty: if the signal comes from a direction close to the estimated one the use of a Derivative Constraint instead of a Point Constraint avoids that the beamformer places a response null in the direction of the desired signal.

Eigenvector Constraints

They are based on the minimum squares approximation of the desired response and tipically they are used to control the beamformer response correspondingly to a set of directions and/or frequencies.
With these constraints (in the sense of minimum squares) it is assured that the mean square error between the desired response and the actual one, in a specific spatial/temporal domain, is minimized thanks to some constraints. In this sense the Eigenvector Constraints are very effective.

2.2.1 MVDR (Minimum Variance Distortionless Response) Beamforming Algorithm

The **MVDR (Minimum Variance Distortionless Response)** adaptive beamforming algorithm is based on the computation of the beamformer coefficients $\boldsymbol{w}_{MVDR}$ through the following relation:

$$\boldsymbol{w}_{MVDR} = \frac{\hat{\boldsymbol{R}}^{-1}\boldsymbol{d}_0}{\boldsymbol{d}_0^H \hat{\boldsymbol{R}}^{-1}\boldsymbol{d}_0} \tag{2–14}$$

where $\hat{\boldsymbol{R}}$ is the Estimated Auto-Covariance Matrix and $\boldsymbol{d}_0$ is the steering vector (related to the look direction) of the considered array.

The matrix $\hat{\boldsymbol{R}}$ is calculated by means of a *K*-samples time-domain average of the Raw Data Matrix $\boldsymbol{X}$:

$$\hat{\boldsymbol{R}} = \sum_{k=1}^{K} \boldsymbol{x}(t_k)\boldsymbol{x}^H(t_k) = \boldsymbol{X}_k \boldsymbol{X}_k^H \tag{2–15}$$

where $\boldsymbol{X}_k$ is the sub-matrix of $\boldsymbol{X}$ restricted to only *k*-samples and $(\cdot)^H$ is the Hermitian operator.

On the contrary $\boldsymbol{d}_0$ is calculated from the knowledge of the radio-source signal DOA (Direction Of Arrival) θ_0 and of the array geometry with this formula:

$$\boldsymbol{d}_0 = \boldsymbol{d}(\theta_0) = \left[1 \quad e^{j2\pi\frac{d}{\lambda}\sin\theta_0} \quad \cdots \quad e^{j2\pi(N-1)\frac{d}{\lambda}\sin\theta_0}\right]^H \tag{2–16}$$

The block diagram of the MVDR beamforming system is represented in the scheme of figure 2.2.

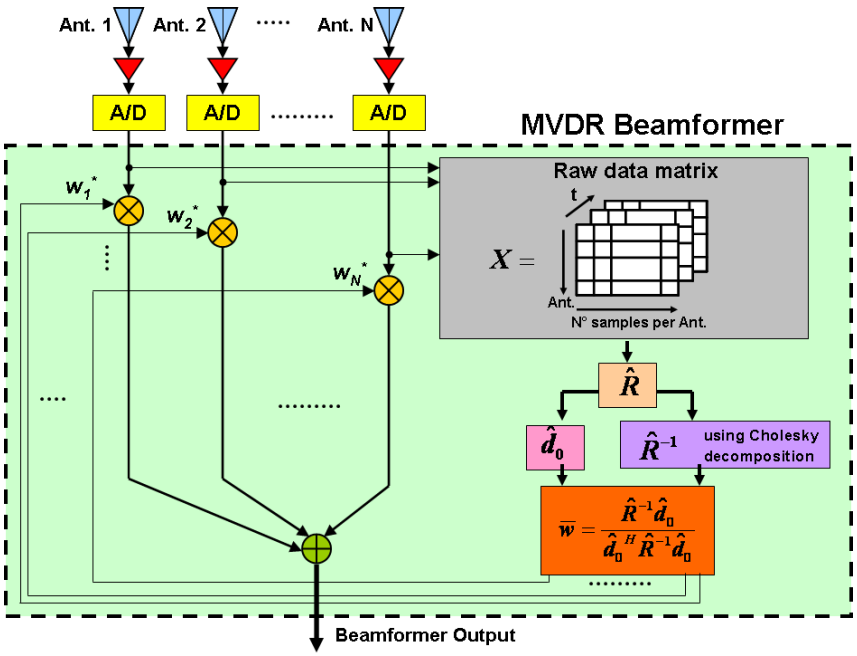


Figure 2.2: block diagram of the MVDR beamforming algorithm

2.3 GSC (Generalized Sidelobe Canceller) beamforming algorithm

The **GSC** algorithm represents an alternative (but equivalent at the same time) formulation of the LCMV problem: it turns a constrained minimum problem (LCMV) into an absolute minimum problem.
This approach in some applications can simplify the implementation of the beamformer that would result too complex if realized through LCMV algorithm.
Suppose to decompose the coefficient vector $\boldsymbol{w}$ in 2 orthogonal components $\boldsymbol{w}_0$ and $-\boldsymbol{v}$ ($\boldsymbol{w} = \boldsymbol{w}_0 - \boldsymbol{v}$) that belong, respectively, in the space of $\boldsymbol{C}$ columns and in its corresponding null space, so that this decomposition can be used to represent every vector $\boldsymbol{w}$.
Since:

$$\boldsymbol{C}^H \boldsymbol{v} = 0 \tag{2–17}$$

it results that:

$$\boldsymbol{w}_0 = \boldsymbol{C}\left(\boldsymbol{C}^H\boldsymbol{C}\right)^{-1}\boldsymbol{f} \tag{2–18}$$

if $\boldsymbol{w}$ has to satisfy the constraints.
The vector $\boldsymbol{v}$ is a linear combination of the columns of a matrix $\boldsymbol{C}_n$ ($\boldsymbol{v} = \boldsymbol{C}_n \boldsymbol{w}_n$) of $N \times (N - L)$ dimensions, provided that the $\boldsymbol{C}_n$ columns form a basis for the null space of the matrix $\boldsymbol{C}$.
$\boldsymbol{C}_n$ can be obtained from $\boldsymbol{C}$ using any orthogonalization algorithm (for example Gram-Schmidt, QR decomposition or SVD).
The vector of coefficients:

$$\boldsymbol{w} = \boldsymbol{w}_0 - \boldsymbol{C}_n \boldsymbol{w}_n \tag{2–19}$$

can be represented through the following block diagram:

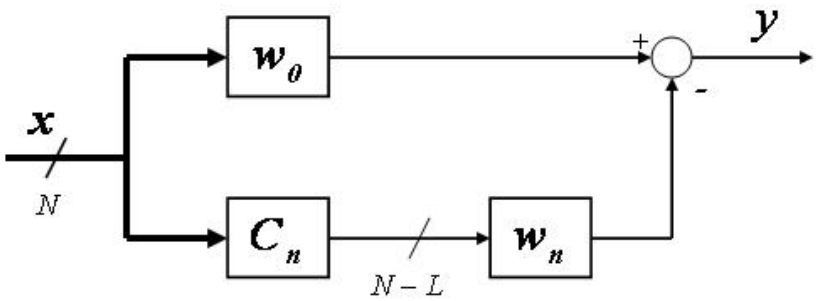


Figure 2.3: simplified block diagram of the GSC beamforming algorithm

The choice of $\boldsymbol{w}_0$ and $\boldsymbol{C}_n$ implies that $\boldsymbol{w}$ satisfies the constraints regardless of $\boldsymbol{w}_n$ and it turns the LCMV problem into the following problem without constraints:

$$\min_{\boldsymbol{w}_n}\left[\left(\boldsymbol{w}_0 - \boldsymbol{C}_n \boldsymbol{w}_n\right)^H \boldsymbol{R}_x \left(\boldsymbol{w}_0 - \boldsymbol{C}_n \boldsymbol{w}_n\right)\right] \tag{2–20}$$

whose solution results to be:

$$\boldsymbol{w}_n = \left(\boldsymbol{C}_n^H \boldsymbol{R}_x \boldsymbol{C}_n\right)^{-1} \boldsymbol{C}_n^H \boldsymbol{R}_x \boldsymbol{w}_0 \tag{2–21}$$

The most important advantage that is obtained with this alternative formulation is that the coefficients $\boldsymbol{w}_n$ are not subject to constraints so this allows using simpler adaptive algorithms.

Moreover a data-independent beamformer $\boldsymbol{w}_0$ is implemented as integral part of the adaptive beamformer: this is very useful especially in situations when a signal cancellation occurs.
Consider this example.
If the constraints to be applied are the same as in (2–10):

$$\boldsymbol{d}^H(\theta,\omega)\boldsymbol{w} = g^* \tag{2–22}$$

the coefficients $\boldsymbol{w}_0$ can be expressed as:

$$\boldsymbol{w}_0 = \frac{g^*\,\boldsymbol{d}(\theta,\omega)}{\left[\boldsymbol{d}^H(\theta,\omega)\boldsymbol{d}(\theta,\omega)\right]} \tag{2–23}$$

$\boldsymbol{C}_n$ satisfies the following equation:

$$\boldsymbol{d}^H(\theta,\omega)\boldsymbol{C}_n = 0 \tag{2–24}$$

so that every column $\left[\boldsymbol{C}_n\right]_i$, in which $1 \le i \le N-L$, can be considered as a data-independent beamformer with a radiation null in the direction θ at the frequency ω: $\boldsymbol{d}^H(\theta,\omega)\left[\boldsymbol{C}_n\right]_i = 0$.
Therefore every signal with *DOA* θ and frequency ω that is received by the array will be blocked or canceled out by the matrix $\boldsymbol{C}_n$.
Generally if the constraints are designed to provide a specific response to the signals having a certain set of *DOAs* and/or frequencies, the columns of $\boldsymbol{C}_n$ will block such signals. This feature of the matrix $\boldsymbol{C}_n$ is the reason for its definition as *Blocking Matrix*. So the signals subject to the constraints are processed only by $\boldsymbol{w}_0$ and, since $\boldsymbol{w}_0$ satisfies the constraints, they are presented to the output with the desired response regardless of $\boldsymbol{w}_n$. Signals having a set of *DOAs* and/or frequencies, for which the response is not constrained, will pass through both the paths of the scheme shown in figure 2.3; the second block in the lower branch chooses the coefficients $\boldsymbol{w}_n$ so that a linear combination of the *Blocking Matrix* output data is as close as possible to the output data of the upper block ($\boldsymbol{w}_0$).

3 Software development

3.1 Design decision

On this first section a complete set of scenarios and diagrams is provided to show the design decision take during the implementation of the beamforming algorithms for GNSS receivers.

3.1.1 Main scenario for beamforming algorithms for GNSS receivers

This section describes the main scenario of the SY1031.

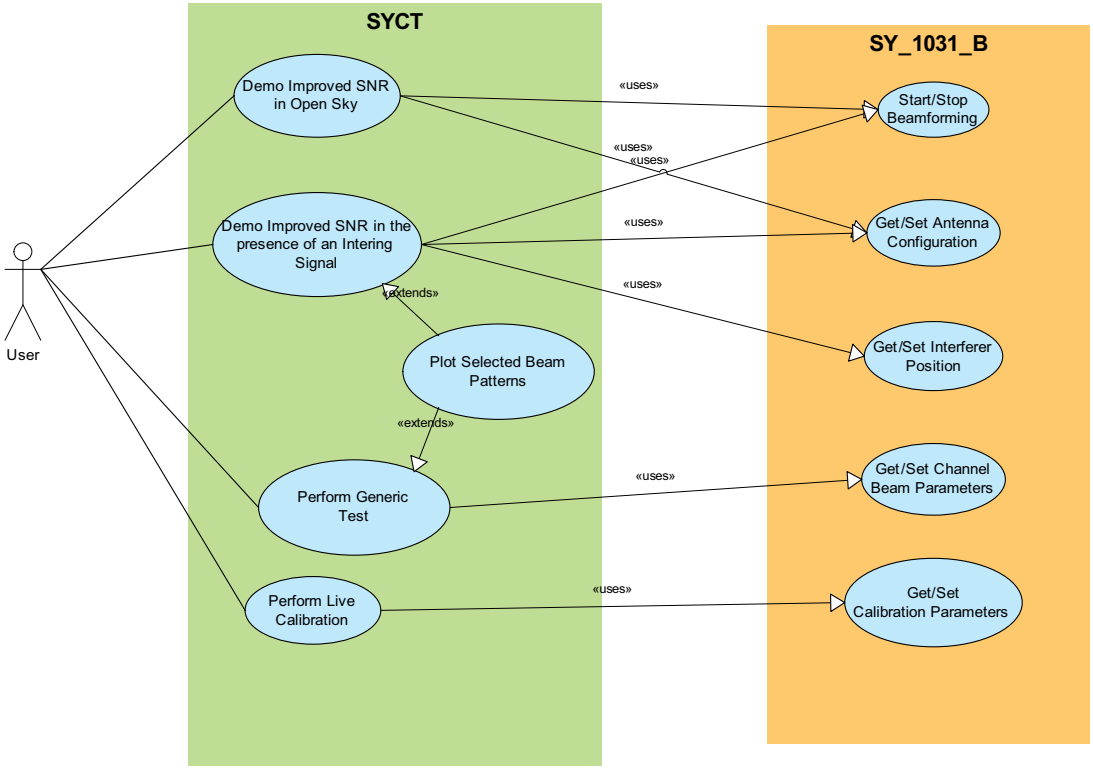


Figure 3.1: Use case diagram – Main scenario

In the USE CASE diagram (figure 3.1) all the functionality that are already available in SY1030 (e.g. display SNR, display Raw Data) have been omitted even if they are useful to demo the beamformer. Basically the software is able to:

1. Steer 16 independent beams in the direction of the tracked satellite and steer a single zero in the direction of the interferer (optional).
2. Perform the calibration of the antenna pattern;
3. Perform tests for diagnostics purposes.

3.1.2 Implementation static structure - Principal component

On this section is described the entire components set present in our system. Figure 3.2 shows the details of the system description.

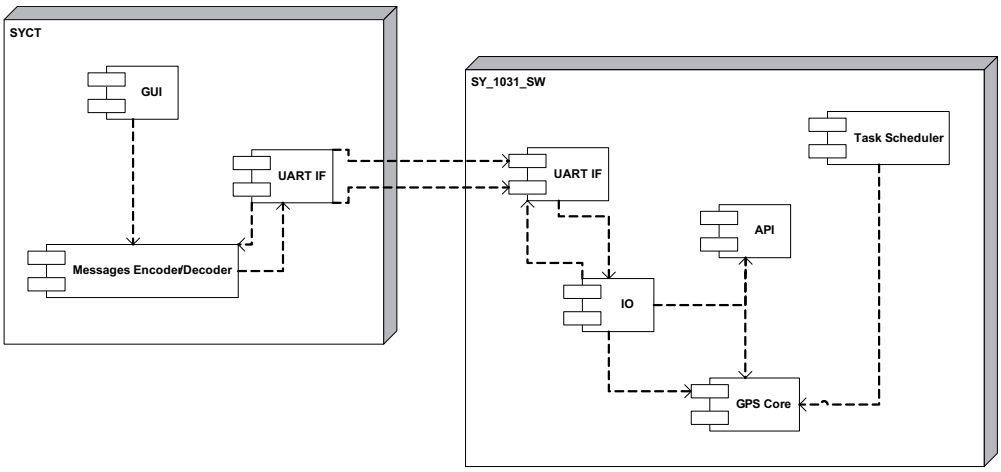


Figure 3.2: System description – Components diagram

To achieve that three classes have been implemented :

1. A beamforming computation class, CBeamformingComputation the core of the beamforming function instantiated in the GPS Core module;
2. A math class Cmatrix and a mathematical library for the solution of non linearly constrained optimization problems instantiated in the GPS Core module;

A class devoted to the handling of the message (SPYN901). The SPYN901 message allows beamforming data/configuration parameters exchange between the user and the core

3.1.3 Implementation static structure – Class diagram

This very important section describes the static behaviour of the algorithms.

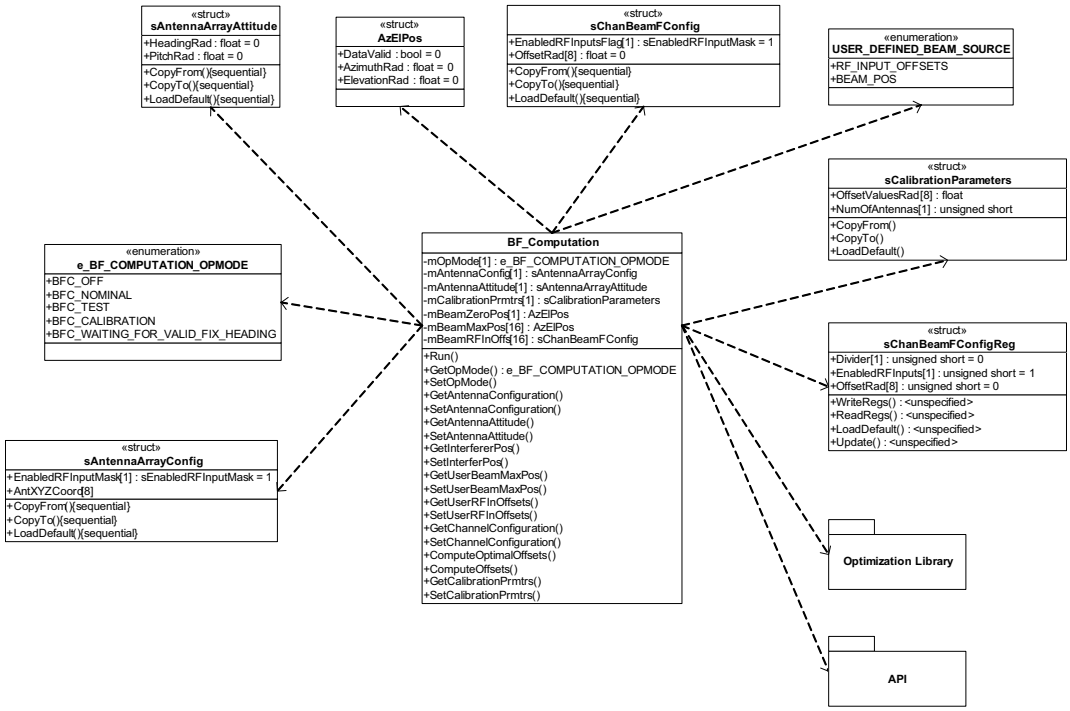


Figure 3.3: Beamformer Computation Class Diagram

The CBeamformingComputation class has the following data members:

- Antenna array configuration (antenna positions, active RF inputs)
- Antenna array attitude (heading , pitch wrt local ENU frame)
- RF input calibration parameters
- User parameters for testing purposes (see message SPYN901 , PH1 and PH2 sections description for more details)

A state called also OpMode has been associated to each functionality :

Beam steering	⇒	BFC_NOMINAL
Calibration	⇒	BFC_CALIBRATION
Test	⇒	BFC_TEST.

As long as the beamsteering in nominal mode requires an accurate position and attitude an additional state has been added BFC_WAITING_FOR_VALID_FIX_HEADING whose name is self explanatory. Last but not least the default state BFC_OFF.

3.1.4 Dynamic behaviour - statechart

Figure 3.4 shows the statechart that represents the obvious valid transitions between the states.

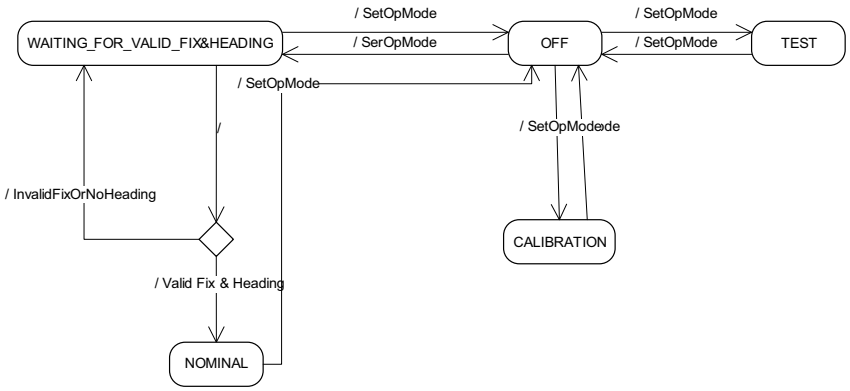


Figure 3.4: Statechart - Beamformer Computation Module Class

3.1.5 Dynamic behaviour – Sequence diagram main scenario

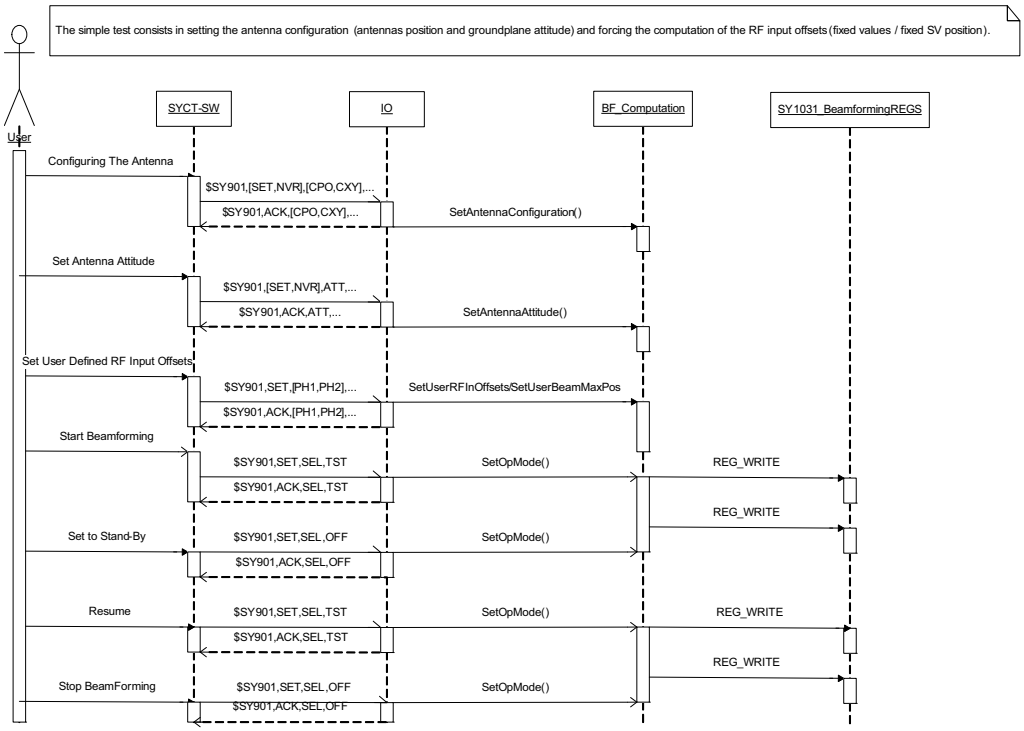


Figure 3.5: Sequence 1 – simple test sequence diagram

Figure 3.5 shows a simple test where the user sets the beam MAX gain position through fixed elevation and azimuth.

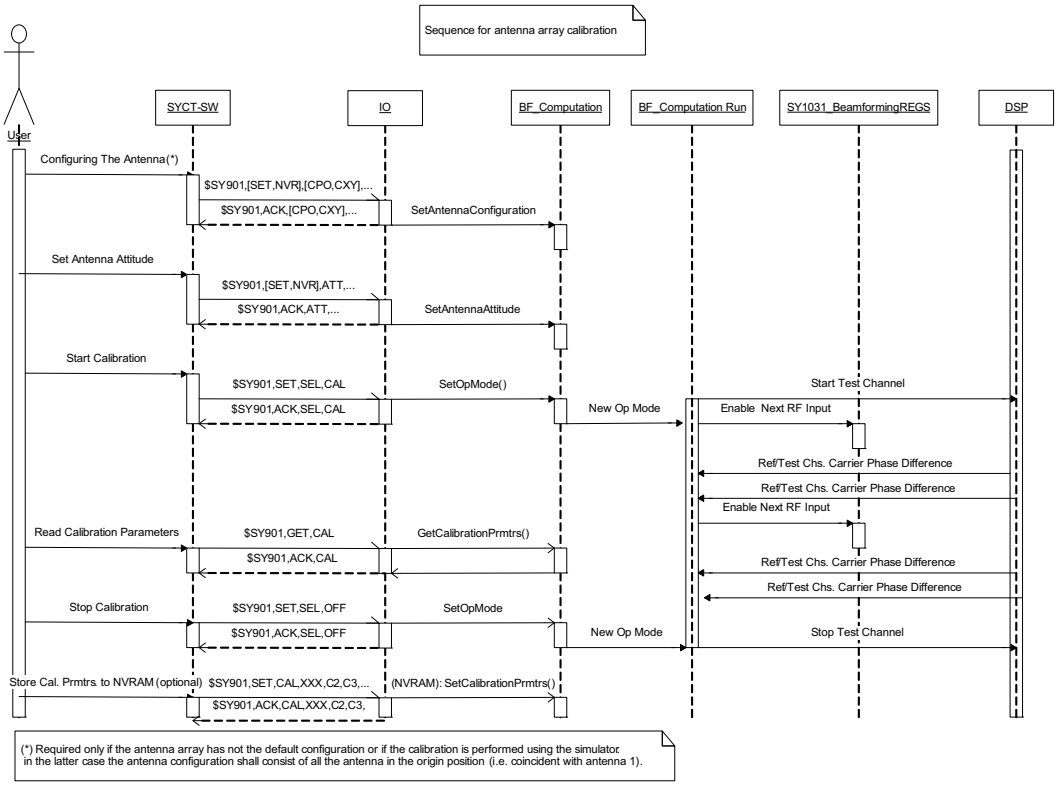


Figure 3.6: Sequence 2 - antenna array calibration

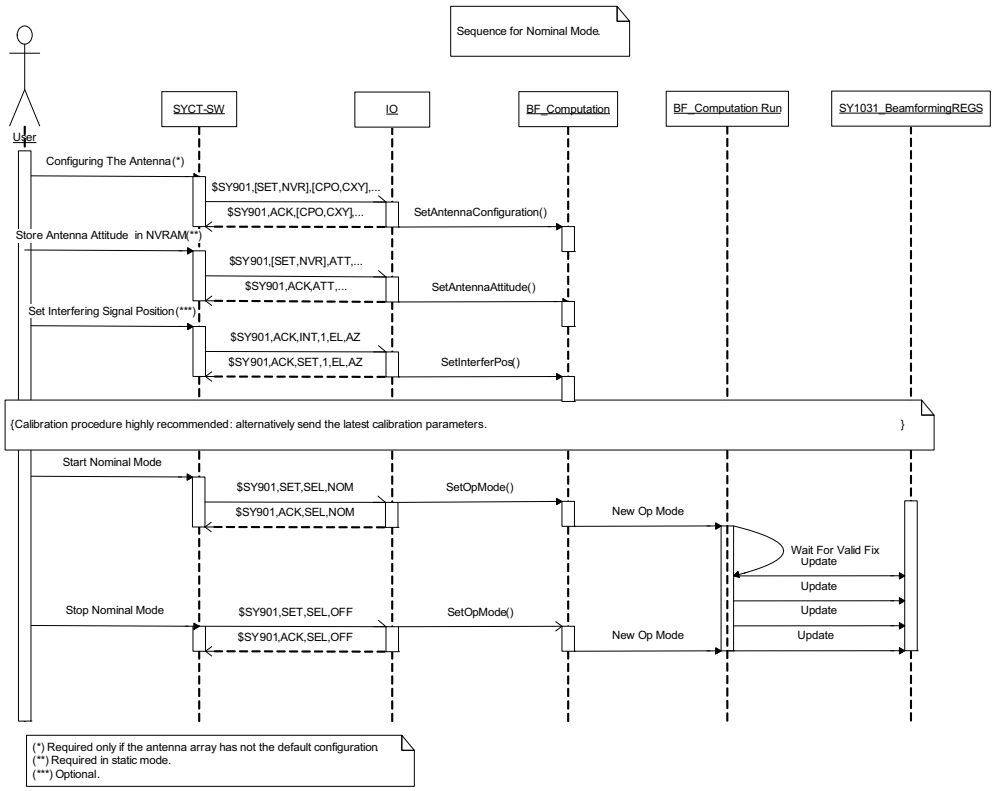


Figure 3.7: Sequence 3 - Nominal mode

4 Implementation notes

4.1 Implementation of MVDR algorithm in C language

Among the candidate beamforming algorithms to be used in the frame of GRABEL project (described in chapter 2) the most suitable one seems to be the MVDR algorithm for various reasons.
First of all it is an optimum algorithm from the statistical point of view: as for the LCMV algorithm it minimizes the output contribution due to noise and interfering signal(s) while maintaining a proper response in the direction of the desired signal.
Moreover it is an adaptive algorithm and its adaptivity is obtained estimating periodically the Auto-Covariance Matrix.
The simulations have demonstrated that this algorithm has a good capability in removing not fixed interfering signals provided that the *DOA*s of the interferences are not very close to the one of the wanted signal.
The MVDR is also a *blind* algorithm in the sense that it does not require the knowledge of the *DOAs* of undesired signals. It only needs as input data the *DOA* of the signal that has to be received.
Finally, since GRABEL receiver will have a small number of antennas, it is not convenient to apply linear constraints on the beamformer response because in this case the reduction of freedom degrees would risk to deeply and negatively affect the response in the direction of the desired signal.

4.1.1 Preliminary remarks on the software implementation

The MVDR algorithm and, more in general, all the digital beamforming algorithms are based on the introduction of appropriate amplitudes and phases in the signals coming from each antenna.
This is achieved multiplying the signals by specific coefficients that have to be consequently complex.
So it is necessary to define the “complex number” type in the program, for example with the use of a structure. Therefore all the arithmetic operations have to be redefined or, more simply, extended to the case of complex numbers.
After having calculated the matrix $\hat{\boldsymbol{R}}_x$ (the Estimated Auto-Covariance Matrix, as stated before) from the data acquired by the receiver system, it is necessary to calculate the inverse of the matrix $\hat{\boldsymbol{R}}_x$ itself.
This matrix is, for its own definition, an Hermitian, positive-definite matrix, so an efficient implementation of $\hat{\boldsymbol{R}}_x^{-1}$, in terms of calculus time, can be obtained using the Cholesky Decomposition algorithm.
This method allows the decomposition of the matrix $\hat{\boldsymbol{R}}_x$ into the product of a lower triangular matrix $\boldsymbol{L}$ and its conjugate transpose:

$$\hat{\boldsymbol{R}}_x = \boldsymbol{L}\boldsymbol{L}^H \tag{4–1}$$

After having calculated $\boldsymbol{L}$ matrix, $\hat{\boldsymbol{R}}_x^{-1}$ can be obtained with the following mathematical operations:

$$\hat{\boldsymbol{R}}_x^{-1} = \left(\boldsymbol{L}\boldsymbol{L}^H\right)^{-1} = \left(\boldsymbol{L}^H\right)^{-1} \cdot \boldsymbol{L}^{-1} \tag{4–2}$$

This expression demonstrates that the inverse of the matrix $\hat{\boldsymbol{R}}_x$ can be derived starting from the knowledge of only $\boldsymbol{L}$ matrix.
This is the reason of the great advantage offered by the Cholesky Decomposition algorithm for the calculus of $\hat{\boldsymbol{R}}_x^{-1}$ matrix as regards both the memory required to store the data and the computation time (fewer mathematical operations have to be executed).

With regard to this aspect, remember that, if $\hat{\boldsymbol{R}}_x$ is a $N \times N$ matrix, the computational cost related to the Cholesky Decomposition is proportional to $N^3 / 6$ whereas the proportionality factor associated to the Gauss factorization with partial pivoting is $N^3 / 3$.
This means that the Cholesky Decomposition causes a considerable saving in terms of FLOPs computed by the CPU, especially if we consider that the application context of this adaptive beamforming algorithm needs a very frequent update of the matrix $\hat{\boldsymbol{R}}_x^{-1}$.
Moreover it is important to remark that an optimized version of the Cholesky Decomposition, which requires the smallest possible amount of memory, has been implemented in the frame of GRABEL project.

4.1.2 Results

The MVDR beamforming algorithm for a linear array has been implemented in a ANSI C program in order to have some realistic information about the achievable performances in terms of computation time.
The C program has been compiled both on Linux and Windows operative system and some simulations have been performed.
The main goal of the simulations was to have a statistical measure, as accurate as possible, of the time required by the program to calculate the coefficients. For this purpose, some timing tests, which take into account the time interval that elapses from the reading of the input data up to the calculation of the final coefficients, have been set up.
The platform that was used for the tests is: Intel® Core™2 Quad CPU Q6600 with 2.40 GHz clock frequency, 3GB of RAM and 64 bits Kernel.
The input data, stored in the Raw Data Matrix $\boldsymbol{X}$ (mentioned before), has been generated using MATLAB, supposing:

- to have a linear array composed by 4 antennas;
- to point the main beam of the antenna array in the direction $\theta_0 = 30°$.

In order to prove that the results obtained with the C code were correct, the beamformer coefficients $\boldsymbol{w}_{MVDR}$ calculated by the C program have been compared with the ones calculated by a MATLAB program, already used for previous simulations of the same algorithm.
The timing tests have been made run many times: the time values obtained as output data of the test have been increasingly added up and accumulated, and then averaged on the total number of times the test has been executed; so a statistical measurement of the computation time has been accomplished.
Table 4.1 shows synthetically the results:

Table 4.1: results of the timing tests

Time window [samples]	Number of test repetitions	Average user time [sec]	Average machine time [sec]
200	20000000 (~ 10 min. of sim.)	0.00003045	0.00003030
200	**60000000 (~ 30 min. of sim.)**	**0.00003049**	**0.00003052**
1000	4500000 (~ 10 min. of sim.)	0.00014389	0.00014422
1000	**1300000 (~ 31 min. of sim.)**	**0.00014389**	**0.00014400**

Two different kinds of time measurement are present in this table: *Average User Time* and *Average Machine Time*.
The *Average User Time* is calculated with the use of the standard C function ***clock*** (Time.h): it provides the number of CPU “clock ticks” elapsed from the moment the program starts running.

The procedure is here resumed: first of all the number of CPU clock ticks measured at the instant the main program reads the input data (beginning of the time test) are subtracted to the ones measured at the instant the main program calculates the beamformer coefficients; then the result of this subtraction has to be divided by the number of CPU clock ticks elapsed in one second, so, in this way, the average user time expressed in [s] is finally derived.
On the contrary the *Average Machine Time* is obtained making use of the standard C function ***time*** (Time.h) which returns the current calendar time. In this case, in order to have the machine time already expressed in [s], it is sufficient to subtract the calendar time at the instant the main program begins to read the input data to the one the coefficients are calculated.
As one can notice from table 4.1, the two measurements of time present more or less the same values: this fact represents a sort of correctness confirmation of the obtained results.
Besides in this table two cases of measurement have been highlighted: in particular we are dealing with the values achieved setting a sufficient number of test repetitions (simulation time) as to have statistically valid results.

4.2 New Products Board and Platform Definitions

Starting from the NS1030 latest software release (4.0.15 production) a new set of products, board and platforms have been added:

- Platform SY1030.
- Board FPGA_SY1030.
- Product GRABEL_BEAMFORMER a standalone GPS receiver able to interface an antenna array for beamforming.

4.2.1 SY1030 Platform

The characteristics of the SY_1030 A are:

- Hardware.
- Scratchpad RAM 64 k.
- AHB RAM 64 k.
- Sin/Cos Map modified to 16 values map with -1,0,1 symbols (it was 8 values with -2, -1, 0, 1, 2 symbols)1.
- A new set of registers to interface the channels.
- Software.
- DSP code runs in the AHB RAM.
- All the data (even in debug mode) in Scratchpad RAM.

4.2.2 GRABEL beamformer Product

For what concerns the software, the main differences with respect to the baseline NS1030 SW 4.0.15 production are:

- Diagnostics messages X401, X402, X404 aligned to more recent versions of the post-processing tools (resp. $PSPYN1401, $PSPYN1402, $PSPYN1404,).
- SNR computation using the channel local best noise floor estimate (and not a fixed noise floor);
- It implements the beamforming algorithms. The beamforming software support can be enabled disabled simply by defining / undefining the macro __BEAMFORMING_SUPPORTED__ in the GRABEL / HOSTED MODE products configuration file “Configure.h”.

[1] This modification was not strictly necessary, but it is a small one and is a good starting point to get confidence with respect to modifications to the SY1016 (former NJ1016) engine.

4.3 Algorithms Implementation Notes

4.3.1 Nominal Mode No Interferer

When the interferer is disabled the synthesized beam of each channel is simply steered in the direction of the tracked satellite.

In details, first the antenna coordinates are converted from body frame to the Local ENU frame. The signal path differences between each antenna and the central antenna (the one connected to RF input 0) are then converted to phase offsets: geometric phase offsets.

The Beamformer parameters of each channel are set equal to the sum of the calibration parameters and the opposite of the geometric phase offsets.

4.3.2 Nominal Mode Interferer enabled

When the interferer rejection function is enabled the synthesized beam of each channel is steered in the direction of the tracked satellite and one null is also steered in the direction of the interferer.

In details, first the antenna coordinates are converted from body frame to the Local ENU frame. The tracked SV signal path differences between each antenna and the central antenna (the one connected to RF input 0) are converted to phase offsets and called max geometric phase offsets. The interferer signal path differences between each antenna and the central antenna (the one connected to RF input 0) are converted to phase offsets and called zero geometric phase offsets.

Four functions are defined:

- Amplitude of the antenna pattern at the elevation and azimuth of the tracked SV;
- Amplitude gradient of the antenna pattern at the elevation and azimuth of the tracked SV;
- Amplitude of the antenna pattern at the elevation and azimuth of the interferer source;
- Amplitude gradient of the antenna pattern at the elevation and azimuth of the interferer source;

The pointers to these functions are passed to the SQP algorithm that solves the problem of computing the optimal beamformer phase offsets. The optimal offsets maximize the antenna pattern amplitude at the elevation and azimuth of the tracked SV minimizing at the same time the amplitude at the elevation and azimuth of the interferer source.

For what concerns the SQP algorithm, it has been implemented according to the requirement document [A1] except for the fact that the Hessian of the objective function and not the Hessian of the Lagrangian has been used: in fact the performance comparison on MATLAB so far demonstrates that there is no advantage in using the Lagrangian but only the disadvantage of additional computations.

For what concerns the algorithm initialization, at the very first computation attempt the opposite of the geometric offsets are used to initialize the vector of unknown phase offsets, while the BFGS Hessian approximation is set equal to the identity matrix. Whether there is convergence or not the following data :

1. X last optimal phase offsets estimate;
2. B last BFGS Hessian approximation;
3. Y last Object Function Gradient;

are saved in external RAM at the end of each computation attempt (less than 6Kb of data). These data will then be used to speed up the next attempt.

.

In general at the very first computation attempt are required a number of iterations ranging from 3 to 9. At the subsequent attempt, as long as we start from the previous values a single iteration is sufficient (3 in the worst case): the above number of iterations estimate are based on MATLAB tests and on a still limited number of scenarios.

When in nominal mode, to the beamformer class is assigned a time slot of 100 ms every second.

The maximum number of allowed iterations per computation attempt has been set to 4.

The following rules apply(refer to [A1] §9.1.3)::

1. If the Beamforming Computation Module runs out of time the next channel to be processed is stored and the computation suspended. At the next call the computation is resumed;
2. If the SQP algorithm is not able to converge in 4 iterations, the computations for that channel is suspended, the intermediate solution and status information are stored in external RAM, the registers are updated with the intermediate solution if its quality is better than the current one. At the next call to the beamforming algorithm the computation will be resumed.
3. The data saved in external ram are ignored in the initialization if the inputs to the algorithm have:
    a. Satellite position;
    b. Interferer position did;
    c. The RF input mask ;
    d. Attitude

   Changed significantly..

4.3.3 Calibration

Calibration is a very important topic strictly connected to beamforming.
Generally speaking the purpose of calibration is to correct (compensate) for all amplitude and phase errors that occur in the analogue part (up to the A/D conversion) of a receiver.
In particular these are errors that occur in the antenna elements and in the RF Front-End (LNA, Down-conversion mixer, amplifiers, filters …).
In other words calibration allows to equalize all the amplitude and phase differences of the receiver chains electronics (antenna, LNA, RF down-converter, …) up to Base-Band Processor.
If the array is NOT calibrated in both amplitude and phase:

- Beamsteering: the main lobe of the beam synthesized by the Beamforming network does NOT properly point towards the desired direction.

- MVDR algorithm: the main lobe is still deeply affected by pointing errors; however the radiation nulls are more robust to errors.

Proposed solution from INAF

The basic principle of the proposed calibration procedure [Smo02] can be represented using the schematic shown in figure 4.1.

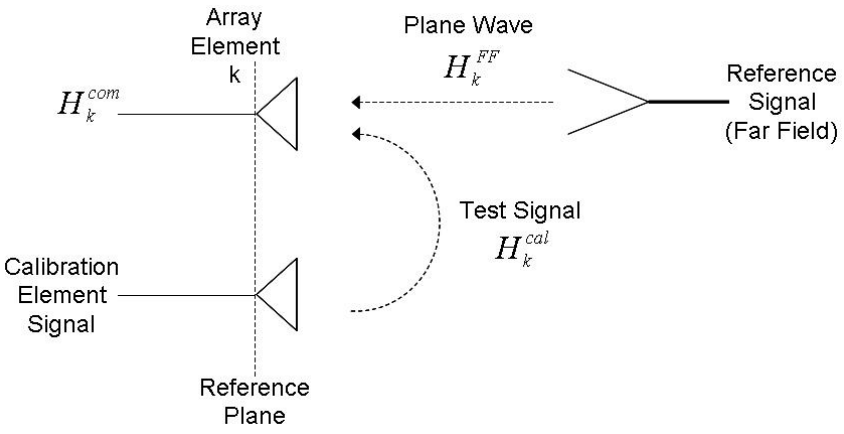


Figure 4.1: The principle of the proposed calibration procedure for the GRABEL receiver.

Two types of signals can be injected into each array element:

- an external signal generated by a far-field source;
- a signal injected through mutual coupling with the calibration elements.

This calibration procedure is based on the relationship between the transfer function at the output of element *k* from one of the calibration elements, $H_k^{cal} H_k^{com}$, and also the transfer function due to one or more incident plane waves, $H_k^{FF} H_k^{com}$.

The first part of the calibration procedure is conducted once inside a high-quality anechoic room as shown in figure 4.2.

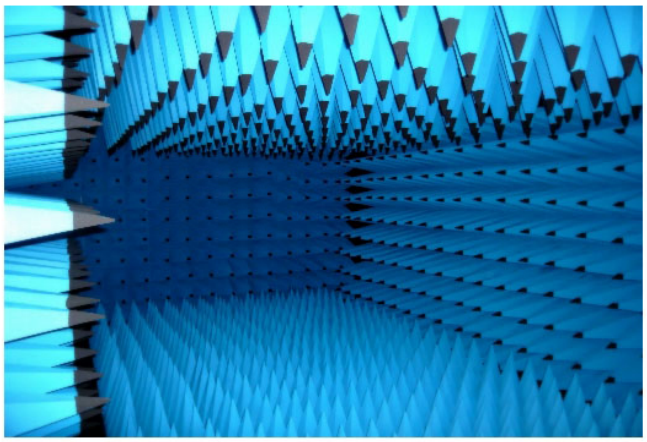


Figure 4.2: Example of an anechoic room.

Two separate measurements occur: the first uses the calibration elements, and the other uses a far-field source. This results in two measured complex signal coefficients per element:

$$a_k = H_k^{FF} H_k^{com} \quad (4\text{–}3)$$

$$b_k = H_k^{cal} H_k^{com} \quad (4\text{–}4)$$

which can then be used to determine the element coefficients, c_k, such that:

$$c_k = \frac{a_k}{b_k} = \frac{H_k^{FF}}{H_k^{cal}} \tag{4–5}$$

The second part of the procedure occurs in the real outdoor environment, where the coefficients of the element electronics, H_k^{com}, are likely to change, and are given by $\hat{H}_k^{com}$.
A far-field source with enough strength to be received by a single element no longer exists, and thus the calibration elements are used. Hence, the measurement using the calibration elements is:

$$\hat{b}_k = H_k^{cal} \hat{H}_k^{com} \tag{4–6}$$

By combining the results from both calibration steps, it is possible to relate the measured gains and phases of each element to an incident plane wave, H_k^{FF}. The following is obtained from Equation (3):

$$\hat{a}_k = H_k^{FF} \hat{H}_k^{com} = \hat{b}_k c_k \tag{4–7}$$

The ratio $\hat{a}_k / a_k$ gives the required complex gain variations of each element.

However, the discussion so far has excluded how to accurately obtain the complex coefficients a_k and b_k.

The remaining part of this section develops a new technique, known as MEP (Multi-Element Phase-toggle), to measure these coefficients.
The MEP method is proposed here in order to simultaneously calibrate groups of elements. This is a significant improvement over other techniques, which typically calibrate elements individually. Additionally, the technique has the advantage of reducing measurement errors through phase toggling.
The method that will be proposed here is an extension of the technique developed by Lee et al. [Lee93].
The number of elements, *K*, that can be calibrated simultaneously with the MEP method depends on the number of phase states, *N*, that are available.

To determine the offset amplitude, g_k, and offset phase, ϕ_k, of *K* array elements, *N* measurements are required (one for each phase state).
The elements to be calibrated are set to their maximum gain and the remaining elements are set to minimum gain, which maximizes the SNR. During the measurements, the phase-shifter settings of the elements that are being calibrated are toggled with a particular step frequency, f_k. The received signal, $S(n)$ for $n = [0...N-1]$, from each measurement is then given by:

$$S(n) = \sum_{k=0}^{K-1} g_k \exp(j\phi_k) \exp\left(\frac{j2\pi n f_k}{N}\right) \tag{4–8}$$

$$= \sum_{k=0}^{K-1} a_k \exp\left(\frac{j2\pi n f_k}{N}\right) \text{ for } n = [0...N-1], \tag{4–9}$$

where *k* is the element index, $k = [0...K-1]$.

The step frequency, f_k, should be an odd number in order to step through all phase states, for example, $f_k = 2k+1$. In this way, the DC component in the spectrum is avoided.

The complex gains of the *K* array elements, a_k, can be calculated by solving the set of *N* equations given in Equation (7). In the scenario where the phase shifter has a range of 0 to 2π, a discrete Fourier transformation can be used to solve for the gains. The required computation time is further reduced when the number of phase states is a power of two, allowing implementation of the Fast Fourier Transformation (FFT). The complex gain, a_k, of element *k* is then found in bin f_k of the FFT spectrum. Once all the complex offsets have been determined for the array, a lookup table can be formed that corrects for the relative amplitude and phase offsets among the array elements.
The MEP method has two main advantages. The first is that an average amplitude and phase offset of each element is measured, not just one particular phase setting. The other is that all interfering signals from other array elements (which are not phase-toggling) are located in bin zero of the FFT spectrum. The MEP technique allows bad isolation between the channels while not affecting the desired result.

The sequence of steps, on which the calibration procedure is based, is summarized in the following flowchart (Figure 4.3).

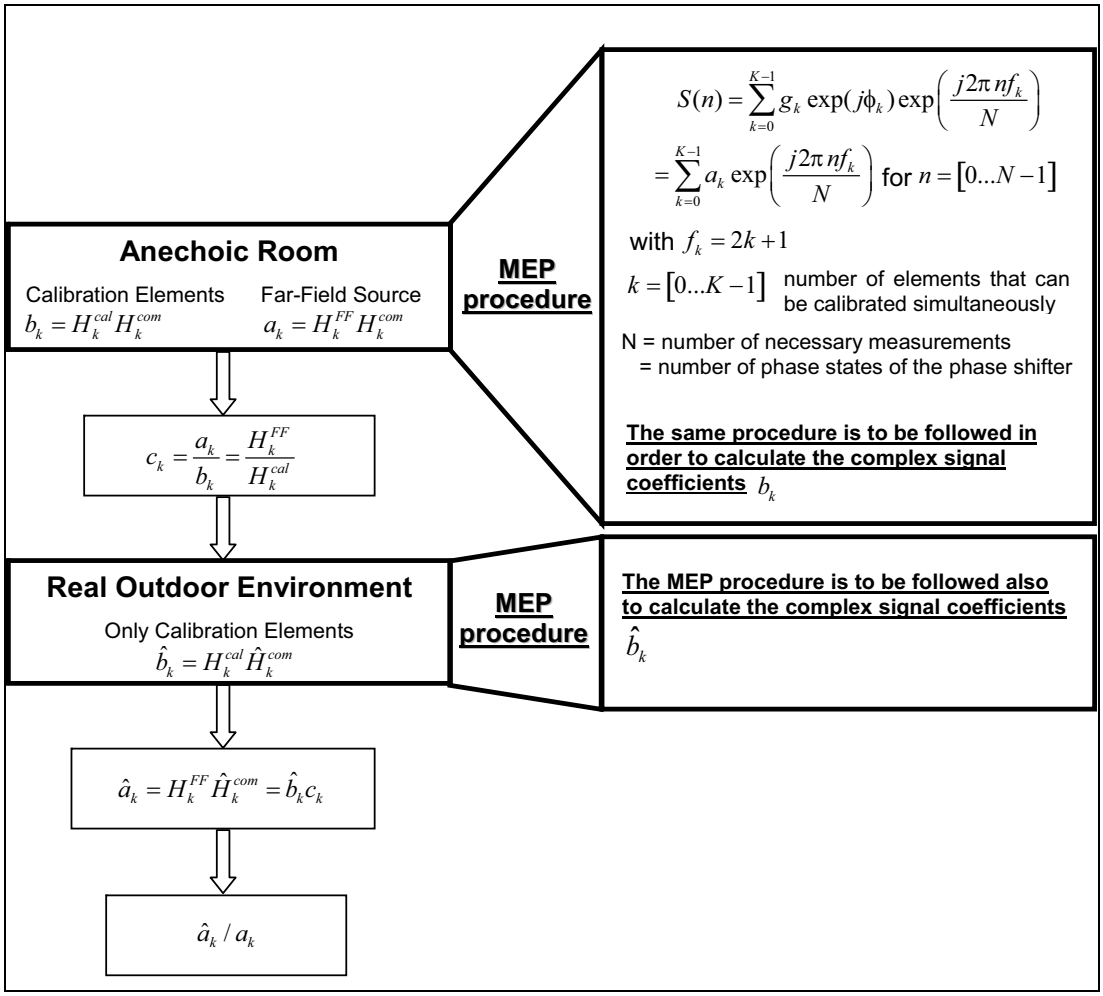


Figure 4.3: sequence of steps on which the proposed calibration procedure is based.

The calibration technique presented in this chapter is suitable for its application in the frame of GRABEL project for various reasons:

- The anechoic chamber can be used to calibrate the GRABEL receiver thanks to the small dimension of the overall system. Moreover the anechoic chamber is necessary only once in the first part of the calibration procedure.

- The antenna system has to be equipped with one or more calibration elements, but it is possible in the case of GRABEL project.
- This procedure allows to calibrate (in both amplitude and phase) simultaneously groups of elements in a simple and fast way.

Although the proposed calibration method and in particular the MEP method has been widely simulated and tested on field with very good results, it can not be adopted in the case of the GRABEL project.
This is due for various reasons. First of all this calibration procedure requires the characterization of every antenna system inside an anechoic room. Even if it has to be done only once, in the first part of the procedure, obviously it is not possible to apply it in case of mass production of the system.
In addition to this fundamental aspect, an other important reason is related to the hardware complexity of its implementation. In fact, as confirmed in chapter 6, the system in charge of the modification of the signal amplitudes inside the baseband processor board would require a very big increase in hardware complexity; moreover tests on field, reported in chapter 5, have demonstrated that errors due to lack of equalization of the signal amplitudes do not affect so deeply the system performances. So the best trade-off between performance and hardware complexity can be achieved keeping the signal amplitudes fixed and modifying only the signal phases.
Also for this reason the proposed calibration method has been discarded for GRABEL project.

On the contrary the calibration algorithm that has been selected is the second of the two proposed in [A1]. The Calibration consists in the following steps: first the satellite in tracking with the highest elevation is selected and the channel tracking it, is defined the *reference channel*. The selected satellite is assigned to the first idle channel: the so called *test channel*.

When the *test channel* is in tracking and the carrier phase measurement is available, the algorithm activates periodically all the input channels enabled in the current antenna configuration (except for the first) on the *test channel.* The period is 10 sec. Every second is available a new carrier phase difference between the *reference channel* and the *test channel*: these differences are input to a variable bandwidth filter (one per active RF input) to compute the calibration parameters. The geometric offset computed based on the current antenna configuration is obviously subtracted from the measurement. In a similar way also the antenna gains mismatch is estimated. estimated (see [A1]).

NOTE: In case of calibration tests using the simulator a special antenna configuration shall be set where all the antennas are at the origin (i.e. all the antennas positions coincides with the position of antenna 1). The status of the calibration can be verified by:

- Checking the trace messages.
- Displaying using SYCT the $PSPYN1401 message RF input field of the test channel.
- Checking the current values of the calibration parameters.

4.3.4 Hosted Mode

The “Hosted Mode” has been restored and the NMRX400 message has been updated to take into account the new channel data fields (i.e. raw carrier phase and per channel Noise Floor). The Hosted mode is also able to simulate the Hardware behaviour in case the Beamformer module enters in calibration mode: for functional tests only.

The “Hosted Mode” is a configuration of the SY103X software that runs in Windows. Currently this configuration is able to run only offline (playback mode) and is used only for debug purposes. In “Hosted Mode” the Signal Processing software is disabled, and the DSP data are received though a dedicated message NMRX400. To run an “Hosted Mode” session run the SY103X with NMRX400 messages enabled (and the PSPYN401 messages disabled in case of slow serial connections) and log the output data. Start the Hosted Mode from Visual Studio.

Select the menu item File > Set Host Log File and input the file name where you want the output data to be saved.

Select the menu item File > Read NMEA File. The “File Open” dialog window is displayed , select the SY103X log file as input file. The “playback” starts: the output will be similar to the one of the log file as expected. Similar position and velocities are computed because the input data are the same , the small differences are due to the fact that the two floating point libraries, native and windows are not identical and the Kalman filter is a non linear algorithm.

With few modifications (two/three weeks work) the “Hosted Mode” could run live on a PC in real time: the PC would be connected to a DSP running the DSP software and the exchange of raw data and predictions would take place using dedicated messages (the NMRX400 for raw data already works like this). This may be useful in case algorithms more cumbersome than the Kalman Filter are needed to compute the Navigation Solution in real time: e.g. an INS-GPS Kalman filter would fit with some difficulties into the SY103X-LEON that misses the FPU.

4.4 Core Software Changes

4.4.1 CN0 and Signal Level Based Thresholds

In the former NS1030A the Noise Floor for carrier to noise ratio CN0 computation was fixed and provided by the initial settings. Moreover the noise floor did not normalize many internal thresholds.

With SY103X SW the assumption of constant noise floor is no more valid, and all the thresholds shall be referred to the channel CN0. Here follow the list of the modifications in the core DSP and NAV algorithm introduced in SY103X.

1) Modified the CN0 computation in GetCtoN0RatiodB() method, based on the signal amplitude term mNPforCN and Noise Power term mNFEst.

2) Added the mNFEst field to the measurement block (per channel noise floor estimate);

3) Added GetCtoN0RatiodB() method also to the measurement block: in this way the CN0 can be computed also using data from the measurement block;

4) Added mNFEst field to X400 message to have the channel estimated Noise floor power available also in hosted mode;

5) Modified the CN0 computation in the Kalman GetMeasurements() method, now uses the GetCtoN0RatiodB method in the measurement block.

6) Substituted all the tracking algorithms thresholds based on accumulated signal power (i.e. on mNPforCN) with CNO dependent ones.

4.4.2 Note on RQ250 and Noise Floor Normalization

The common noise floor normalization was not implemented because the results of the measurements indicated that the maximum difference between the minimum and maximum noise floor in all the possible antenna pattern configurations is about 30% of the average value. Thanks to the beamforming divisor. This difference causes only negligible effects during the short transients after abrupt changes in the antenna array configuration (e.g. the CN0 takes 3 seconds to get to the correct value, but after one second the error is smaller than 2 dB).

4.4.3 Notes on the New Classes

The beamforming class CbeamformingComputation has been instantiated in the gpscore module as a member of the Cgps class. Some structures have been added to the NVRAM to store beamforming class initialization data. As already mentioned in §4.3.2 the beamforming class uses 1K of external RAM data for intermediate computation data storage. For future uses, very likely debugging, a new structure called ExtRamData has been created applying the same concept used for the NVRAM data: the External Ram Data structure is instantiated statically and is defined in a single file.

The class CSY901 devoted to $PSPYN901 message handling has been instantiated in the nmeax module as a static class. This class has been interfaced to the beamforming class adding functions to the core API.

The class SY_Diagnostics_API devoted to $PSPYN1401, $PSPYN1402, $PSPYN1404, message generation has been instantiated in the nmeax module as a static class. This class has been interfaced to the gps core adding functions to the core API.

4.4.4 Beamforming Scheduling

The beamforming module is currently scheduled at the beginning of every measurement cycle (1Hz) nevertheless its implementation is quasi pre-emptive. It can be called in whichever point of the code and will run maximum for 100 ms (this value can be customized).

4.5 SY1031 beamforming

Here follows a list of the C++ files that were added to the project, of the MATLAB file used during the development and for verification purposes together with a brief description of their content.

Table 4.2: C++ files added to the project and MATLAB files used for the development

	File Name	Description	Notes
SY1031 GPS Core	*Beamforming.h* *Beamforming.cpp*	Beamforming core class.	
	BeamformingAPI.h *BeamformingAPI.cpp*	CORE API extension for beamforming.	
	ExtRamdata.h *ExtRamdata.cpp*	External RAM Data Structure definition.	Is used to store inter. results,6Kb.
	SYMatrixMath.h *SYMatrixMath.cpp*	Cmatrix class implementic basic matrix operations.	
	SequentialQuadraticProgramming.h *SequentialQuadraticProgramming.cpp*	SQP specific algorithms	
	SY_Diagnostics_API_Private.h *SY_Diagnostics_API_Private.cpp*	Diagnostics API extension implementation: messages x1401,x1402, x1404	
SY1031 NMEAX	*SY_Diagnostics.h* *SY_Diagnostics.cpp*	X1401, X1402, X1404 message handling	
	SY_901.h *SY_901.cpp*	SPYN901 message handling	
MATLAB	NLConstrOptimization.m	SY1031 SQP algorithms	
	BeamFormingUsingMatlab.m	Compares SY1031 and MATLAB SQP algorithms performance	
	LUSolve.m	Linear Systems of equations solution based on LU decomposition: same implementation as in SY_1031.	
	GetENUAntennaPositionVector.m	Body to ENU antenna coordinates converter.	
	SPYN901_To_AntennaArrayPatternPlot.m	Plots the synthetized beam starting from SPYN901 messages extracted from the log file.	
	Analyze Calibration	Uses the log file of a Calibration Procedure to plot the carrier phase differences and SNR differences between the test and the reference channel versus time.	
	PlotAntennaArrayPattern.m	Antenna Array pattern plot using color scales and antenna patterns cross section at specific azimuth angles.	

4.6 CXY Antenna Array Configuration in Cartesian Coordinates

This section allows the user to define the antenna array. Let N be the number of antennas in the antenna array.

RFInMask is a bit flag that indicates to which RF input the N antennas are connected. The Beamforming SW assumes that the RF input 1 is always enabled, configurations failing to meet this condition (RFInMask & 1 == 1) will be refused with an error message notification.

The sum of bits set to one in RFInMask shall be equal equal to N.

The cartesian coordinates $(X_2,Y_2),\ldots, (X_N,Y_N)$ are the antenna coordinates in the body frame $\{b_x,b_y,b_z\}$ where Antenna 1 is the origin. When the heading, pitch and roll angle are zero $\{b_x,b_y,b_z\}$ is aligned to the ENU geodetic frame.

The default configuration is the one with 7 antennas as in figure 4.4.

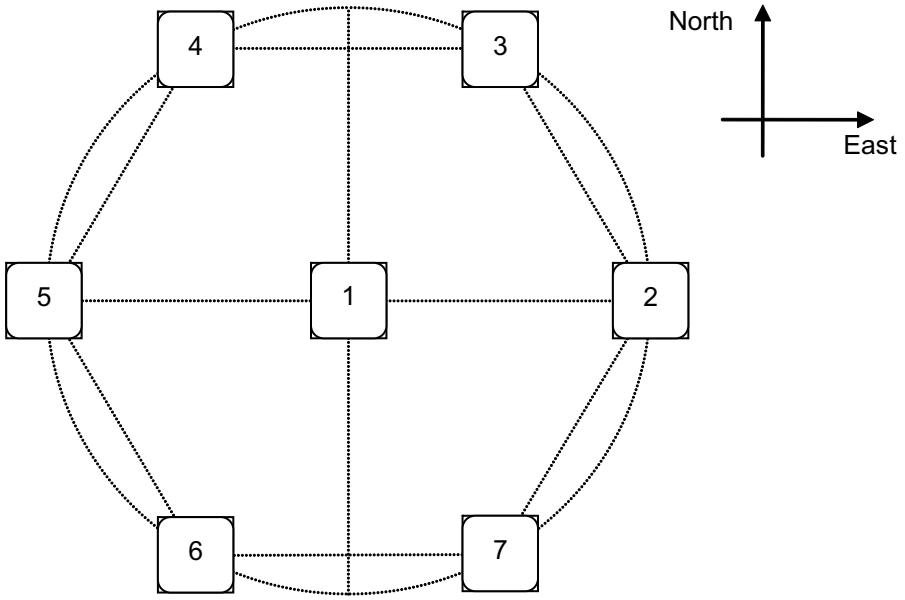


Figure 4.4: Default Built-In Antenna Configuration

4.7 CPO Antenna Array Configuration in Polar Coordinates

This section is an alternative way to define the antenna array. Let N be the number of antennas in the antenna array.

RFInMask is a bit flag that indicates to which RF input the N antennas are connected. The sum of bits set to one in RFInMask shall be equal to N. The Beamforming SW assumes that the RF input 1 is always enabled; configurations failing to meet this condition (RFInMask & 1 == 1) will be refused with an error message notification.

The polar coordinates $(R_2,\Psi_2),\ldots, (R_N,\Psi_N)$ are the antenna polar coordinates in the body frame $\{b_x,b_y,b_z\}$ where Antenna 1 is the origin.

If $\Psi_n = 0$ then the antenna lays on the b_x positive axis, if $\Psi_n = 90$, the antenna lays on the b_y positive axis.

4.8 ATT Antenna Array Attitude

Antenna Array Attitude described by the parameters Heading and Pitch. Antenna Heading is the angle measured from the true North clockwise to the b_y positive axis while Antenna Pitch is the angle measured from the local geodetic plane counter-clockwise to the b_x positive axis.

Depending on the operative mode the SY1031 GPS receiver SW will use the first available attitude resource in table 4.3.

Table 4.3: Attitude Sources for Nominal and Test Mode.

Nominal Mode	Test Mode
Last ATT message (*while in Nominal Mode*).	Last ATT message.
NVRam stored attitude	NVRam stored attitude
Latest valid heading or zero heading and 0 pitch.	Default attitude i.e. 0 heading 0 pitch.

In Nominal mode to tell the receiver to get the pitch and ignore the heading send the message leaving the heading field empty: the latest estimated heading by the navigation solution will be used.

4.9 INT Interferer Position

INT section defines the position of an interferer with respect to local geodetic frame expressed using azimuth and elevation.

Zeros steering is enabled only in Nominal mode and if a valid interferer position is present in NVRAM or if a valid 901 message has been sent to the receiver since the last power on.

To invalidate NVRam info or the last sent message just send and INT section with resp. NVR / SET action and the enable field set to 0.

4.10 CAL Calibration Parameters Section

RFInMask is a bit flag that indicates to which RF input the N antennas are connected. The Beamforming SW assumes that the RF input 1 is always enabled; configurations failing to meet this condition (RFInMask & 1 == 1) will be refused with an error message notification. The calibration parameters are the measured phase offset [deg] and amplitude ratio of the antennas 2 to N wrt antenna. The calibration parameter refers to the antenna configuration loaded with the CXY or CPO commands. There is no coherency check, e.g. the software does not check that the number of antennas indicated in the calibration parameters section is coherent with the number of antennas indicated in the antenna configuration section. When less than 8 calibration parameters are loaded the remaining ones are set to [0,1], and the beamforming software will use the number of antennas declared in the configuration section. The phase offset is defined as the difference between the measured signal phase at RF input 1 and the measured signal phase at RF input n (n = 2.9) when the signal travel distance between the source and the antennas 1 and n is identical (i.e. the geometric offset is 0). The antenna at the centre of the antenna array (the one at position 0,0) has zero offset by definition.

$$C_n = \begin{cases} 0 & ,1 \quad n = 1 \\ \varphi_1 - \varphi_n & ,\frac{A_n}{A_1} \quad n = 2..n \end{cases}$$

Where φ is the measured signal carrier phase, C_n the calibration parameter A the measured signal amplitude.

The geometric offset is the offset due to the geometric path as in the example shown in figure 4.5. A1 and A2 are respectively the reference antenna and the second antenna of the current antenna pattern configuration. The signal source is at elevation θ and at an infinite distance.

The calibration parameters are added to the phase offset of each RF input. For more details refer to [A1].

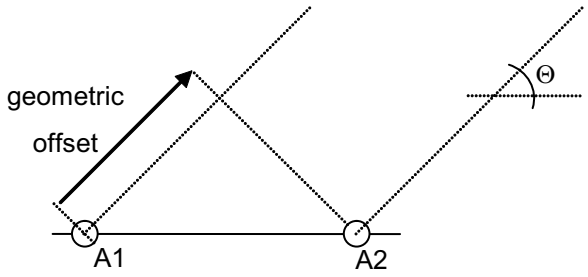


Figure 4.5: Geometric offset definition

4.11 PH1 User Defined beam-former Parameters

This section is for test purposes. With this section is possible to specify a set of enabled RF inputs and their offsets for the specified channel (or for all the channels). When the active operative mode is TEST these parameters are used to drive the beam-former. These data cannot be saved in NVRAM but are retained by the software even after a change of state:

1. $PSPYN901,SET,PH1,A,3,20 → user defined RF input mask and offsets
2. $PSPYN901,SET,SEL,TST → operative mode set to test
3. $PSPYN901,SET,SEL,NOM → operative mode set to nominal
4. $PSPYN901,SET,SEL,TST → operative mode set to test the parameters defined at step 1 are used to drive the beam-former.

At receiver power on the user defined parameters are loaded with the default configuration where just RF input 1 is enabled and all the offsets are set to zero.

The calibration parameters are always added up to the user-defined parameters.

4.12 PH2 User Defined beam-former Parameters

This section is for test purposes and is similar to PH1 except that this time the offsets for the specified channel are computed for each enabled input using the user defined elevation and azimuth according to the following formula.

$$\Delta\Phi_n\big|_{n=2..N} = -\frac{2\pi}{\lambda}\left(\cos(\Theta)\left(X_n^{'}\cos\left(\frac{\pi}{2}-\Omega\right)+Y_n^{'}\sin\left(\frac{\pi}{2}-\Omega\right)\right)+Z_n^{'}\sin(\Theta)\right)$$

Where Θ and Ω are resp. the user defined elevation of the azimuth i.e. the desired position of the maximum antenna pattern gain, while $[X'_n, Y'_n, Z'_n]$ is the position of the antenna in the ENU frame. The position of the antenna in the ENU frame is linked to the position in body frame provided in the antenna configuration message by the following equation:

$$\begin{bmatrix} X_n^{'} \\ Y_n^{'} \\ Z_n^{'} \end{bmatrix} = \begin{bmatrix} \cos(h) & \sin(h) & 0 \\ -\sin(h) & \cos(h) & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \cos(p) & 0 & -\sin(p) \\ 0 & 1 & 0 \\ \sin(p) & 0 & \cos(p) \end{bmatrix}\begin{bmatrix} X_n \\ Y_n \\ Z_n \end{bmatrix}$$

Where h and p are resp. the current heading and pitch angle while $[X_n, Y_n, Z_n]$ is the position of the antenna in the body frame.

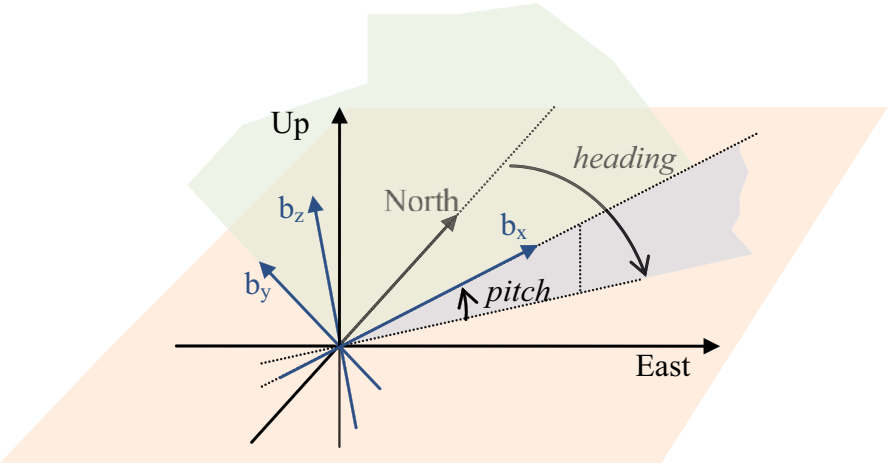


Figure 4.6: Body frame rotation wrt ENU frame described by the heading and pitch angles.

In test mode the heading won't be updated by the current solution. The user using the ATT command can provide the attitude.

The message section PH2 works in both Test mode and Nominal mode (PH1 works only in test mode). The PH2 message can be used in Nominal mode to override the elevation and azimuth provided by the navigation software for the specified channel: this is useful to test the antenna array for interference rejection in indoor environments using an antenna to irradiate the simulator GPS signal. To remove the elevation and azimuth overriding by PH2 while in Nominal mode, there are two methods:

1. Switch off and on the Nominal mode;
2. Send a PH1 message.

5 Test

5.1 Introduction

The test campaign has been subdivided into two phases:

- Phase 1 – Basic Tests
- Phase 2 – Live Tests

Phase 1 aims at testing the basic functionalities using mainly the HOSTED mode.

Each basic test is described by a table that reports a sequence of commands and the corresponding expected responses. Graphic results are reported when this is necessary to demonstrate the correctness of the obtained values.

Phase 2 – Live Tests aims at testing the real hardware in a real environment (whenever this is possible) to check the performance and the robustness : (e.g. robustness of the calibration parameters V.S. heading uncertainty).

5.2 Basic Tests

5.2.1 Calibration Sequence Tests

Test A1: Automatic calibration sequence with all the antennas in the same positions (i.e. constellation simulator input).

Table 5.1: Calibration on 9 coincident antennas

Step	Command	Expected Response	Note
1	$PSPYN901,GET,SEL*00	$PSPYN901,ACK,SEL,OFF	If the response doesn't match send $PSPYN901,SET,SEL,OFF and repeat step 1
2	$PSPYN901,GET,CXY*00	$PSPYN901,ACK,CXY,7F,95,0,48,82,-48,82,-95,0,-48,-82,48,-82*07	Get the current antenna configuration.
3	$PSPYN901,GET,ATT*00	$PSPYN901,ACK,ATT,,0	Get current antenna array attitude (empty heading field means that the default heading , 0 deg, will be used)
4	$PSPYN901,GET,CAL*00	$PSPYN901,ACK,CAL,7F,0.0,1.00,0.0,1.00,0.0,1.00,0.0,1.00,0.0,1.00,0.0,1.00,0.0,1.00*26	Get the current set of calibration parameters.
5	$PSPYN901,SET,CXY,1FF,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0*00	$PSPYN901,ACK,CXY,1FF,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0*00	Set all the antennas in the same positions.
6	$PSPYN901,SET,ATT,0,0*00	$PSPYN901,ACK,ATT,0,0*00	Set default antenna attitude
7			Take note of the highest elevation satellite CAL_PRN and the ID of the channel tracking it CAL_REF.
8	$PSPYN901,SET,SEL,CAL*00	$PSPYN901,ACK,SEL,CAL	Start the calibration sequence.
9			Verify in x401 that a new channel is tracking CAL_PRN take note of the ID of this channel CAL_TST
10	$PSPYN901,GET,RAW,[CAL_TST]*00	$PSPYN901,ACK,RAW,[CAL_TST],0,[2,4,8,10,20,40,80,100],0,0,0,0,0,0,0,0*00	Verify that every 10 seconds the active RF input on channel CAL_TST is changed over the RF inputs 2 to 9.
11	$PSPYN901,GET,CAL*00	$PSPYN901,ACK,CAL,1FF,C2,C3,C4,C5,C6,C7,C8,C9*00	Verify that the calibration values converge to fixed values.
12			Repeat steps 10 and 11 to verify the correctness of the operations.
13	$PSPYN901,SET,SEL,OFF*00	$PSPYN901,ACK,SEL,OFF	Switch off calibration mode
14	$PSPYN901,GET,CAL*00	$PSPYN901,ACK,CAL,1FF,C2,C3,C4,C5,C6,C7,C8,C9	Take not of the calibration values. In case of test in hosted mode the values shall be close to : C2 ~ 0 ; 0.15 C3 ~ 10 ; 1.10 C4 ~ 350 ; 1.20 C5 ~ 20 ; 1.30 C6 ~ 340 ; 1.40 C7 ~ 30 ; 1.50 C8 ~ 180 ; 1.60 C9 ~ 270 ; 9.00

Test A2: Automatic calibration sequence with 7 antennas in the default configuration.

Table 5.2: Test A2 Calibration on an antenna pattern, cross shape plus central antenna

Step	Command	Expected Response	Note
1	$PSPYN901,GET,SEL*00	$PSPYN901,ACK,SEL,OFF	If the response doesn't match send $PSPYN901,SET,SEL,OFF and repeat step 1
2	$PSPYN901,GET,CXY*00	$PSPYN901,ACK,CXY,7F,95,0,48,82,-48,82,-95,0,-48,-82,48,-82*07	Get the current antenna configuration.
3	$PSPYN901,GET,ATT*00	$PSPYN901,ACK,ATT,,0	Get current antenna array attitude (empty heading field means that the default heading , 0 deg, will be used)
4	$PSPYN901,GET,CAL*00	$PSPYN901,ACK,CAL,7F,0.0,1.00,0.0,1.00,0.0,1.00,0.0,1.00,0.0,1.00,0.0,1.00*26	Get the current set of calibration parameters.
5			Take note of the highest elevation satellite CAL_PRN and the ID of the channel tracking it CAL_REF.
6	$PSPYN901,SET,SEL,CAL*00	$PSPYN901,ACK,SEL,CAL	Start the calibration sequence.
7			Verify in x401 that a new channel is tracking CAL_PRN take note of the ID of this channel CAL_TST
8	$PSPYN901,GET,RAW,[CAL_TST]*00	$PSPYN901,ACK,RAW,[CAL_TST],0,[2,4,8,10,20,40],0,0,0,0,0,0,0,0*00	Verify that every 10 seconds the active RF input on channel CAL_TST is changed over the RF inputs 2 to 7.
9	$PSPYN901,GET,CAL*00	$PSPYN901,ACK,CAL,7F,C2,C3,C4,C5,C6,C7 *00	Verify that the calibration values converge to fixed values.
10			Repeat steps 8 and 9 to verify the correctness of the operations.
11	$PSPYN901,SET,SEL,OFF*00	$PSPYN901,ACK,SEL,OFF	Switch off calibration mode
12	$PSPYN901,GET,CAL*00	$PSPYN901,ACK,CAL,7F,C2,C3,C4,C5,C6,C7	Take not of the calibration values. In case of test in hosted mode the values shall be close to : C2 ~ 32 ; 0.15 C3 ~ 26 ; 1.10 C4 ~ 334 ; 1.20 C5 ~ 348 ; 1.30 C6 ~ 324 ; 1.40 C7 ~ 46 ; 1.50

Test A3: Automatic calibration sequence using a subset of the RF Inputs. All the antennas are set in the origin.

Table 5.3: Test A3 sequence. Calibration of a subset of the full antenna pattern

Step	Command	Expected Response	Note
1	$PSPYN901,GET,SEL*00	$PSPYN901,ACK,SEL,OFF	If the response doesn't match send $PSPYN901,SET,SEL,OFF and repeat step 1
2	$PSPYN901,GET,CXY*00	$PSPYN901,ACK,CXY,7F,95,0,48,82,-48,82,-95,0,-48,-82,48,-82*07	Get the current antenna configuration.
3	$PSPYN901,GET,ATT*00	$PSPYN901,ACK,ATT,,0	Get current antenna array attitude (empty heading field means that the default heading , 0 deg, will be used)
4	$PSPYN901,GET,CAL*00	$PSPYN901,ACK,CAL,7F,0.0,1.00,0.0,1.00,0.0,1.00,0.0,1.00,0.0,1.00,0.0,1.00,0.0,1.00*26	Get the current set of calibration parameters.
5	$PSPYN901,SET,CXY,199,0,0,0,0,0,0,0,0*00	$PSPYN901,ACK,CXY,199,0,0,0,0,0,0,0,0*00	Set all the antennas in the same positions.
6	$PSPYN901,SET,ATT,0,0*00	$PSPYN901,ACK,ATT,0,0*00	Set default antenna attitude
7			Take note of the highest elevation satellite CAL_PRN and the ID of the channel tracking it CAL_REF.
8	$PSPYN901,SET,SEL,CAL*00	$PSPYN901,ACK,SEL,CAL	Start the calibration sequence.
9			Verify in x401 that a new channel is tracking CAL_PRN take note of the ID of this channel CAL_TST
10	$PSPYN901,GET,RAW,[CAL_TST]*00	$PSPYN901,ACK,RAW,[CAL_TST],0,[2,4,8,10,20,40,80,100],0,0,0,0,0,0,0,0*00	Verify that every 10 seconds the active RF input on channel CAL_TST is changed over the RF inputs 2 to 9.
11	$PSPYN901,GET,CAL*00	$PSPYN901,ACK,CAL,199,C4,C5,C8,C9*00	Verify that the calibration values converge to fixed values.
12			Repeat steps 10 and 11 to verify the correctness of the operations.
13	$PSPYN901,SET,SEL,OFF*00	$PSPYN901,ACK,SEL,OFF	Switch off calibration mode
14	$PSPYN901,GET,CAL*00	$PSPYN901,ACK,CAL,199,C4,C5,C8,C9	Take not of the calibration values. In case of test in hosted mode the values shall be close to : C4 ~ 350 ; 1.20 C5 ~ 20 ; 1.30 C8 ~ 180 ; 1.60 C9 ~ 270 ; 9.00

5.2.2 User Defined Antenna Array Patterns

Test B1: Antenna Array Pattern Defined Using Offsets.

Table 5.4: Test B1 User defined channel beamformer configuration

N#	Command	Expected Response	Note
1	$PSPYN901,GET,SEL*00	$PSPYN901,ACK,SEL,OFF*0C	If the response doesn't match send $PSPYN901,SET,SEL,OFF and repeat step 1
2	$PSPYN901,GET,PH1,1*00	$PSPYN901,ACK,PH1,1,1*1C	default user beamformer configuration
3	$PSPYN901,GET,RAW,1*00	$PSPYN901,ACK,RAW,1,1*1C	default beamformer configuration
4	$PSPYN901,GET,PH1,A*00	$PSPYN901,ERR*15	Verify that this command is invalid.
5	$PSPYN901,SET,SEL,TST*00	$PSPYN901,ACK,SEL,TST*10	Set the operativ mode to test
6	$PSPYN901,GET,SEL,TST*00	$PSPYN901,ACK,SEL,TST*10	Verify the current operative mode
7	$PSPYN901,GET,CXY*00	$PSPYN901,ACK,CXY,7F,95,0,48,82,-48,82,-95,0,-48,-82,48,-82*07	Verify default antenna configuration
8	$PSPYN901,GET,CAL*00	$PSPYN901,ACK,CAL,7F,0.0,1.00,0.0,1.00,0.0,1.00,0.0,1.00,0.0,1.00,0.0,1.00*26	Verify default calibration parameters
9	$PSPYN901,SET,PH1,A,3,180*00	$PSPYN901,ACK,PH1,A,3,180*7B	Enable for all the channels just the first two RF inputs
10	$PSPYN901,GET,RAW,1*00	$PSPYN901,ACK,RAW,1,1,3,256,0,0,0,0,0,0,0*6F	Verify the content of the raw registers for channel 1
11	$PSPYN901,GET,RAW,2*00	$PSPYN901,ACK,RAW,2,1,3,256,0,0,0,0,0,0,0*6C	Verify the content of the raw registers for channel 2
12	$PSPYN901,GET,RAW,16*00	$PSPYN901,ACK,RAW,16,1,3,256,0,0,0,0,0,0,0*59	Verify the content of the raw registers for channel 16

Test B2 Antenna Array Pattern Defined Using azimuth and elevation

Table 5.5: B2 Test sequence. Antenna Array Pattern Defined Using azimuth and elevation

N#	Command	Expected Response	Note
1	$PSPYN901,GET,SEL*00	$PSPYN901,ACK,SEL,OFF*0C	If the response doesn't match send $PSPYN901,SET,SEL,OFF and repeat step 1
2	$PSPYN901,GET,PH2,1*00	$PSPYN901,ACK,PH2,1,1,90,0*26	default user beamformer configuration
3	$PSPYN901,GET,RAW,1*00	$PSPYN901,ACK,RAW,1,0,0,0,0,0,0,0,0,0,0*6C	default beamformer configuration
4	$PSPYN901,GET,PH2,A*00	$PSPYN901,ERR*15	Verify that this command is invalid.
5	$PSPYN901,SET,SEL,TST*00	$PSPYN901,ACK,SEL,TST*10	Set the operative mode to test
6	$PSPYN901,GET,SEL,TST*00	$PSPYN901,ACK,SEL,TST*10	Verify the current operative mode
7	$PSPYN901,GET,CXY*00	$PSPYN901,ACK,CXY,7F,95,0,48,82,-48,82,-95,0,-48,-82,48,-82*07	Verify default antenna configuration
8	$PSPYN901,GET,CAL*00	$PSPYN901,ACK,CAL,7F,0.0,1.00,0.0,1.00,0.0,1.00,0.0,1.00,0.0,1.00,0.0,1.00*26	Verify default calibration parameters
9	$PSPYN901,SET,PH2,1,7F,90,0*00	$PSPYN901,ACK,PH2,1,7F,90,0*66	Set the elevation to 90 and azimuth to 0
10	$PSPYN901,GET,RAW,1*00 $PSPYN901,GET,RAW,2*00 $PSPYN901,GET,RAW,15*00 $PSPYN901,GET,RAW,16*00	$PSPYN901,ACK,RAW,1,2,7F,512,0,0,512,512,512,0,0*2F $PSPYN901,ACK,RAW,2,0,1,0,0,0,0,0,0,0,0*6E $PSPYN901,ACK,RAW,15,0,1,0,0,0,0,0,0,0,0*58 $PSPYN901,ACK,RAW,16,0,1,0,0,0,0,0,0,0,0*5B	Verify the content of the raw registers for channel 1
11	$PSPYN901,SET,PH2,2,7F,45,90*00	$PSPYN901,ACK,PH2,2,7F,45,90*54	Set the elevation to 90 and azimuth to 0
12	$PSPYN901,GET,RAW,1*00 $PSPYN901,GET,RAW,2*00 $PSPYN901,GET,RAW,15*00 $PSPYN901,GET,RAW,16*00	$PSPYN901,ACK,RAW,1,2,7F,512,0,0,512,512,512,0,0*2F $PSPYN901,ACK,RAW,2,2,7F,331,421,91,181,91,421,0,0*25 $PSPYN901,ACK,RAW,15,0,1,0,0,0,0,0,0,0,0*58 $PSPYN901,ACK,RAW,16,0,1,0,0,0,0,0,0,0,0*5B	Verify the content of the raw registers for channel 1
13	$PSPYN901,SET,PH2,16,30,180*00	$PSPYN901,ACK,PH2,16,7F,30,180*53	Set the elevation to 90 and azimuth to 0
14	$PSPYN901,GET,RAW,1*00 $PSPYN901,GET,RAW,2*00 $PSPYN901,GET,RAW,15*00 $PSPYN901,GET,RAW,16*00	$PSPYN901,ACK,RAW,1,2,7F,512,0,0,512,512,512,0,0*2F $PSPYN901,ACK,RAW,2,2,7F,331,421,91,181,91,421,0,0*25 $PSPYN901,ACK,RAW,15,0,1,0,0,0,0,0,0,0,0*58 $PSPYN901,ACK,RAW,16,2,7F,0,192,192,512,320,320,0,0*1F	Verify the content of the raw registers for channel 1
15	$PSPYN901,SET,SEL,OFF*00	$PSPYN901,ACK,SEL,OFF*0C	Switch off.

Test B3 Raw Beamformer Registers Data Read and Write Operations

Table 5.6: B3 test sequence. Raw Beamformer Registers Data Read and Write Operations

N#	Command	Expected Response	Note
1	$PSPYN901,GET,SEL*00	$PSPYN901,ACK,SEL,OFF*0C	If the response doesn't match send $PSPYN901,SET,SEL, OFF and repeat step 1
2	$PSPYN901,GET,RAW,1*00 $PSPYN901,GET,RAW,2*00 $PSPYN901,GET,RAW,3*00 $PSPYN901,GET,RAW,4*00 $PSPYN901,GET,RAW,5*00 $PSPYN901,GET,RAW,6*00 $PSPYN901,GET,RAW,7*00 $PSPYN901,GET,RAW,8*00 $PSPYN901,GET,RAW,9*00 $PSPYN901,GET,RAW,10*00 $PSPYN901,GET,RAW,11*00 $PSPYN901,GET,RAW,12*00 $PSPYN901,GET,RAW,13*00 $PSPYN901,GET,RAW,14*00 $PSPYN901,GET,RAW,15*00 $PSPYN901,GET,RAW,16*00	$PSPYN901,ACK,RAW,1,0,1,0,0,0,0,0,0,0,0*6D $PSPYN901,ACK,RAW,2,0,1,0,0,0,0,0,0,0,0*6E $PSPYN901,ACK,RAW,3,0,1,0,0,0,0,0,0,0,0*6F $PSPYN901,ACK,RAW,4,0,1,0,0,0,0,0,0,0,0*68 $PSPYN901,ACK,RAW,5,0,1,0,0,0,0,0,0,0,0*69 $PSPYN901,ACK,RAW,6,0,1,0,0,0,0,0,0,0,0*6A $PSPYN901,ACK,RAW,7,0,1,0,0,0,0,0,0,0,0*6B $PSPYN901,ACK,RAW,8,0,1,0,0,0,0,0,0,0,0*64 $PSPYN901,ACK,RAW,9,0,1,0,0,0,0,0,0,0,0*65 $PSPYN901,ACK,RAW,10,0,1,0,0,0,0,0,0,0,0*5D $PSPYN901,ACK,RAW,11,0,1,0,0,0,0,0,0,0,0*5C $PSPYN901,ACK,RAW,12,0,1,0,0,0,0,0,0,0,0*5F $PSPYN901,ACK,RAW,13,0,1,0,0,0,0,0,0,0,0*5E $PSPYN901,ACK,RAW,14,0,1,0,0,0,0,0,0,0,0*59 $PSPYN901,ACK,RAW,15,0,1,0,0,0,0,0,0,0,0*58 $PSPYN901,ACK,RAW,16,0,1,0,0,0,0,0,0,0,0*5B	Read the registers values. The expected answers is the default registers values.
3	$PSPYN901,SET,RAW,1,2,7F,10,20,30,40,50,60*00 $PSPYN901,SET,RAW,2,1,FF,10,20,30,40,50,60,70*00 $PSPYN901,SET,RAW,15,1,1FF,10,20,30,40,50,60,70,80*00 $PSPYN901,SET,RAW,16,0,1F0,0,0,0,0,600,700,800,900,1000*00	$PSPYN901,ACK,RAW,1,2,7F,10,20,30,40,50,60,0,0*28 $PSPYN901,ACK,RAW,2,1,FF,10,20,30,40,50,60,70,0*6E $PSPYN901,ACK,RAW,15,1,1FF,10,20,30,40,50,60,70,80*51 $PSPYN901,ACK,RAW,16,0,1F0,0,0,0,0,600,700,800,900*2D	Set values for channels 1,2,15 and 16.
4	$PSPYN901,GET,RAW,1*00 $PSPYN901,GET,RAW,2*00 $PSPYN901,GET,RAW,3*00 $PSPYN901,GET,RAW,4*00 $PSPYN901,GET,RAW,5*00 $PSPYN901,GET,RAW,6*00 $PSPYN901,GET,RAW,7*00 $PSPYN901,GET,RAW,8*00 $PSPYN901,GET,RAW,9*00 $PSPYN901,GET,RAW,10*00 $PSPYN901,GET,RAW,11*00 $PSPYN901,GET,RAW,12*00 $PSPYN901,GET,RAW,13*00 $PSPYN901,GET,RAW,14*00 $PSPYN901,GET,RAW,15*00 $PSPYN901,GET,RAW,16*00	$PSPYN901,ACK,RAW,1,2,7F,10,20,30,40,50,60,0,0*28 $PSPYN901,ACK,RAW,2,1,FF,10,20,30,40,50,60,70,0*6E $PSPYN901,ACK,RAW,3,0,1,0,0,0,0,0,0,0,0*6F $PSPYN901,ACK,RAW,4,0,1,0,0,0,0,0,0,0,0*68 $PSPYN901,ACK,RAW,5,0,1,0,0,0,0,0,0,0,0*69 $PSPYN901,ACK,RAW,6,0,1,0,0,0,0,0,0,0,0*6A $PSPYN901,ACK,RAW,7,0,1,0,0,0,0,0,0,0,0*6B $PSPYN901,ACK,RAW,8,0,1,0,0,0,0,0,0,0,0*64 $PSPYN901,ACK,RAW,9,0,1,0,0,0,0,0,0,0,0*65 $PSPYN901,ACK,RAW,10,0,1,0,0,0,0,0,0,0,0*5D $PSPYN901,ACK,RAW,11,0,1,0,0,0,0,0,0,0,0*5C $PSPYN901,ACK,RAW,12,0,1,0,0,0,0,0,0,0,0*5F $PSPYN901,ACK,RAW,13,0,1,0,0,0,0,0,0,0,0*5E $PSPYN901,ACK,RAW,14,0,1,0,0,0,0,0,0,0,0*59 $PSPYN901,ACK,RAW,15,1,1FF,10,20,30,40,50,60,70,80*51 $PSPYN901,ACK,RAW,16,0,1F0,0,0,0,0,88,188,288,388*1D	Read the registers values for all the channels. Channels 1,2,15 and 16 shall have Beamforming registers value equal to the one just sent . The registers value for all the other channels shall be the default one.
5	$PSPYN901,SET,RAW,A,0,8,0,0,0,1012*00*00*00	$PSPYN901,ACK,RAW,A,0,8,0,0,0,1012,0,0,0,0*26	Enable RF input 4 on all the channels and set also a phase offset for that input.
6	$PSPYN901,GET,RAW,1*00 $PSPYN901,GET,RAW,2*00 $PSPYN901,GET,RAW,3*00 $PSPYN901,GET,RAW,4*00 $PSPYN901,GET,RAW,5*00 $PSPYN901,GET,RAW,6*00 $PSPYN901,GET,RAW,7*00 $PSPYN901,GET,RAW,8*00	$PSPYN901,ACK,RAW,1,0,8,0,0,0,500,0,0,0,0*61 $PSPYN901,ACK,RAW,2,0,8,0,0,0,500,0,0,0,0*62 $PSPYN901,ACK,RAW,3,0,8,0,0,0,500,0,0,0,0*63 $PSPYN901,ACK,RAW,4,0,8,0,0,0,500,0,0,0,0*64 $PSPYN901,ACK,RAW,5,0,8,0,0,0,500,0,0,0,0*65 $PSPYN901,ACK,RAW,6,0,8,0,0,0,500,0,0,0,0*66 $PSPYN901,ACK,RAW,7,0,8,0,0,0,500,0,0,0,0*67	Verify that all the channels have RF input 4 enabled. Check also the value of the phase offset.

	$PSPYN901,GET,RAW,9*00 $PSPYN901,GET,RAW,10*00 $PSPYN901,GET,RAW,11*00 $PSPYN901,GET,RAW,12*00 $PSPYN901,GET,RAW,13*00 $PSPYN901,GET,RAW,14*00 $PSPYN901,GET,RAW,15*00 $PSPYN901,GET,RAW,16*00	$PSPYN901,ACK,RAW,8,0,8,0,0,0,500,0,0,0,0*68 $PSPYN901,ACK,RAW,9,0,8,0,0,0,500,0,0,0,0*69 $PSPYN901,ACK,RAW,10,0,8,0,0,0,500,0,0,0,0*51 $PSPYN901,ACK,RAW,11,0,8,0,0,0,500,0,0,0,0*50 $PSPYN901,ACK,RAW,12,0,8,0,0,0,500,0,0,0,0*53 $PSPYN901,ACK,RAW,13,0,8,0,0,0,500,0,0,0,0*52 $PSPYN901,ACK,RAW,14,0,8,0,0,0,500,0,0,0,0*55 $PSPYN901,ACK,RAW,15,0,8,0,0,0,500,0,0,0,0*54 $PSPYN901,ACK,RAW,16,0,8,0,0,0,500,0,0,0,0*57	

5.2.3 Nominal Mode Tests

Test C1, Nominal Mode: automatic Antenna Pattern beam steering in the direction of the tracked SVs.

Nominal mode test sequence and expected results example, automatic Antenna Pattern beam steering in the direction of the tracked SVs, no interferer rejection.

Table 5.7: C1 Nominal mode test sequence

N#	Command	Expected Response	Note
1	$PSPYN901,GET,SEL*00	$PSPYN901,ACK,SEL,OFF*0C	If the response doesn't match send $PSPYN901,SET,SEL,OFF and repeat step 1
2	$PSPYN901,GET,CXY*00	$PSPYN901,ACK,CXY,7F,95,0,48,82,-48,82,-95,0,-48,-82,48,-82*07	Get the current antenna configuration.
3	$PSPYN901,NVR,ATT,0,0*00	$PSPYN901,ACK,ATT,0,0*74	Set the current antenna array attitude to 0 heading 0 pitch.
4	$PSPYN901,GET,CAL*00	$PSPYN901,ACK,CAL,7F,0.0,1.00,0.0,1.00,0.0,1.00,0.0,1.00,0.0,1.00,0.0,1.00*26	Get the current set of calibration parameters.
5	$PSPYN901,SET,SEL,NOM*00	$PSPYN901,ACK,SEL,NOM*0F	If the response doesn't match send $PSPYN901,SET,SEL,OFF and repeat step 1
6			Wait for the first fix. Select a tracked satellite, and take a note of the elevation azimuth and channel ID [CHID].
7	$PSPYN901,GET,RAW,1*00 $PSPYN901,GET,RAW,2*00 $PSPYN901,GET,RAW,3*00 $PSPYN901,GET,RAW,4*00 $PSPYN901,GET,RAW,5*00 $PSPYN901,GET,RAW,6*00 $PSPYN901,GET,RAW,7*00 $PSPYN901,GET,RAW,8*00 $PSPYN901,GET,RAW,9*00 $PSPYN901,GET,RAW,10*00 $PSPYN901,GET,RAW,11*00 $PSPYN901,GET,RAW,12*00 $PSPYN901,GET,RAW,13*00 $PSPYN901,GET,RAW,14*00 $PSPYN901,GET,RAW,15*00 $PSPYN901,GET,RAW,16*00	$PSPYN901,ACK,RAW,1,0,1,0,0,0,0,0,0,0,0*6D $PSPYN901,ACK,RAW,2,1,7F,195,201,6,317,311,506,0,0*22 $PSPYN901,ACK,RAW,3,1,7F,379,114,247,133,398,265,0,0*24 $PSPYN901,ACK,RAW,4,1,7F,295,7,224,217,505,288,0,0*22 $PSPYN901,ACK,RAW,5,0,1,0,0,0,0,0,0,0,0*69 $PSPYN901,ACK,RAW,6,1,7F,506,199,206,6,313,306,0,0*2F $PSPYN901,ACK,RAW,7,1,7F,123,252,129,389,260,383,0,0*2B $PSPYN901,ACK,RAW,8,0,1,0,0,0,0,0,0,0,0*64 $PSPYN901,ACK,RAW,9,1,7F,138,51,425,374,461,87,0,0*25 $PSPYN901,ACK,RAW,10,1,7F,454,348,407,58,164,105,0,0*2F $PSPYN901,ACK,RAW,11,1,7F,441,285,356,71,227,156,0,0*20 $PSPYN901,ACK,RAW,12,0,1,0,0,0,0,0,0,0,0*5F $PSPYN901,ACK,RAW,13,0,1,0,0,0,0,0,0,0,0*5E $PSPYN901,ACK,RAW,14,1,7F,116,385,268,396,127,244,0,0*16 $PSPYN901,ACK,RAW,15,0,1,0,0,0,0,0,0,0,0*58 $PSPYN901,ACK,RAW,16,1,7F,502,149,160,10,363,352,0,0*25	Check the offsets using the MATLAB script SPYN901_To_AntennaArrayPatternPlot
8	$PSPYN901,SET,SEL,OFF*00	$PSPYN901,ACK,SEL,OFF*0C	Switch off the beamforming.
9	$PSPYN901,GET,RAW,1*00 $PSPYN901,GET,RAW,2*00 $PSPYN901,GET,RAW,3*00 $PSPYN901,GET,RAW,4*00 $PSPYN901,GET,RAW,5*00 $PSPYN901,GET,RAW,6*00 $PSPYN901,GET,RAW,7*00	PSPYN901,ACK,RAW,1,0,1,0,0,0,0,0,0,0,0*6D $PSPYN901,ACK,RAW,2,0,1,0,0,0,0,0,0,0,0*6E $PSPYN901,ACK,RAW,3,0,1,0,0,0,0,0,0,0,0*6F $PSPYN901,ACK,RAW,4,0,1,0,0,0,0,0,0,0,0*68 $PSPYN901,ACK,RAW,5,0,1,0,0,0,0,0,0,0,0*69 $PSPYN901,ACK,RAW,6,0,1,0,0,0,0,0,0,0,0*6A $PSPYN901,ACK,RAW,7,0,1,0,0,0,0,0,0,0,0*6B	Check that each channel beamformer is set back to its default configuration.

	$PSPYN901,GET,RAW,8*00	$PSPYN901,ACK,RAW,8,0,1,0,0,0,0,0,0,0,0*64	
	$PSPYN901,GET,RAW,9*00	$PSPYN901,ACK,RAW,9,0,1,0,0,0,0,0,0,0,0*65	
	$PSPYN901,GET,RAW,10*00	$PSPYN901,ACK,RAW,10,0,1,0,0,0,0,0,0,0,0*5D	
	$PSPYN901,GET,RAW,11*00	$PSPYN901,ACK,RAW,11,0,1,0,0,0,0,0,0,0,0*5C	
	$PSPYN901,GET,RAW,12*00	$PSPYN901,ACK,RAW,12,0,1,0,0,0,0,0,0,0,0*5F	
	$PSPYN901,GET,RAW,13*00	$PSPYN901,ACK,RAW,13,0,1,0,0,0,0,0,0,0,0*5E	
	$PSPYN901,GET,RAW,14*00	$PSPYN901,ACK,RAW,14,0,1,0,0,0,0,0,0,0,0*59	
	$PSPYN901,GET,RAW,15*00	$PSPYN901,ACK,RAW,15,0,1,0,0,0,0,0,0,0,0*58	
	$PSPYN901,GET,RAW,16*00	$PSPYN901,ACK,RAW,16,0,1,0,0,0,0,0,0,0,0*5B	

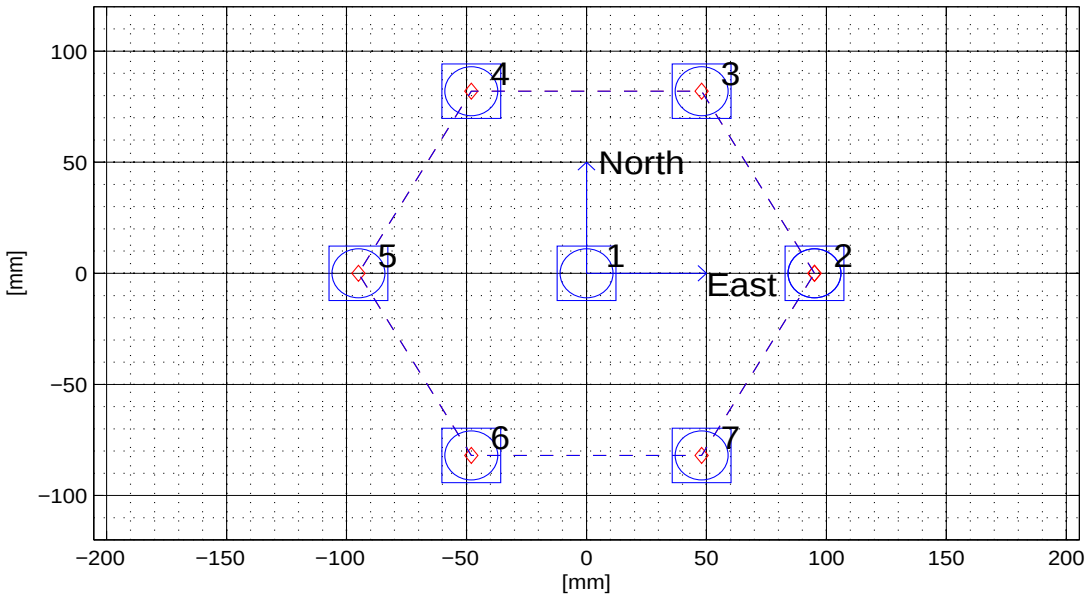


Figure 5.1: Antenna Array Map

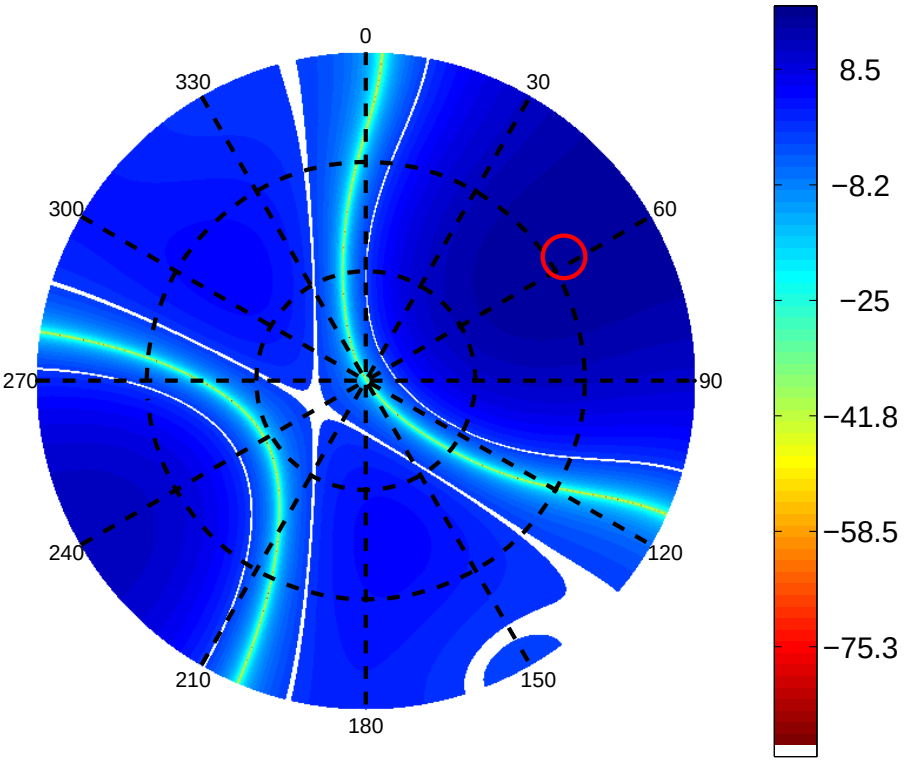


Figure 5.2: Nominal Mode Test Antenna Pattern For Channel 2

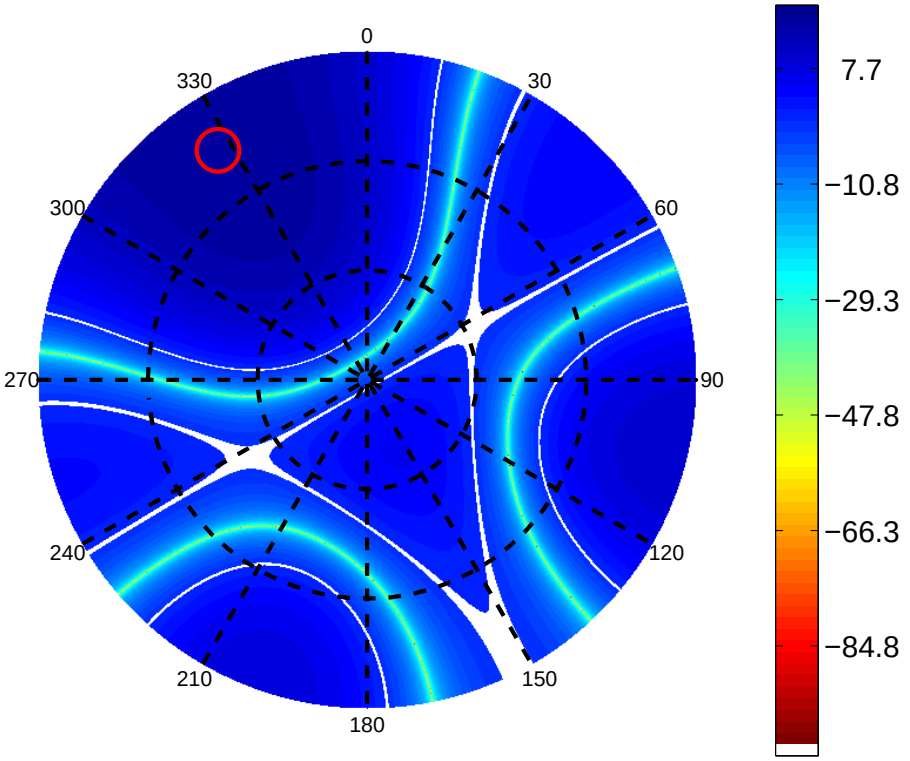


Figure 5.3: Nominal Mode Test Antenna Pattern For Channel 3

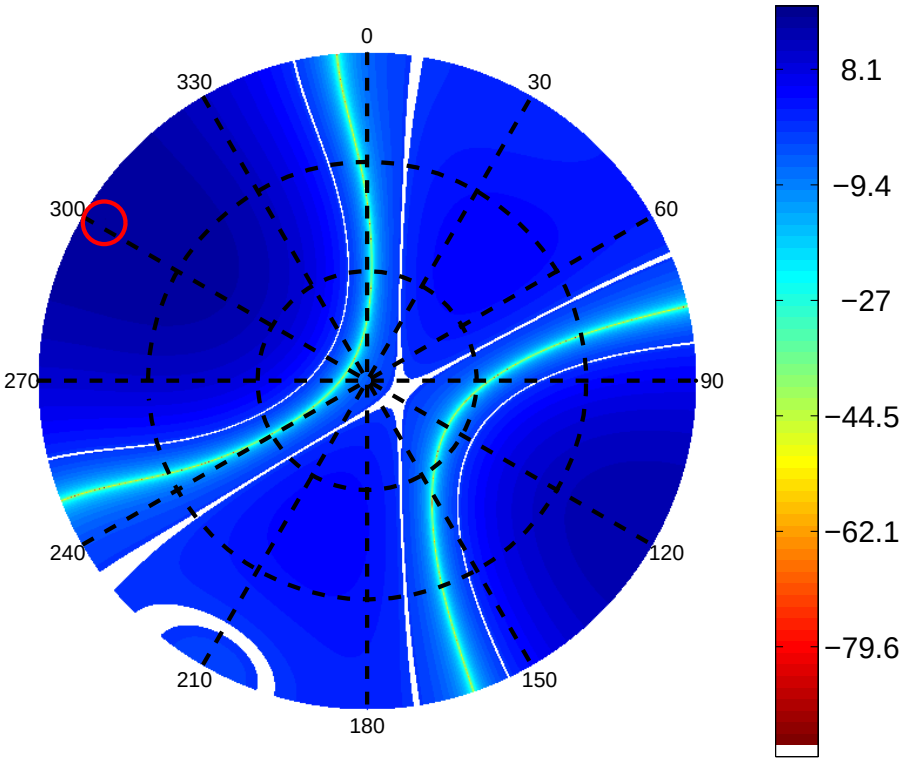


Figure 5.4: Nominal Mode Test Antenna Pattern For Channel 4

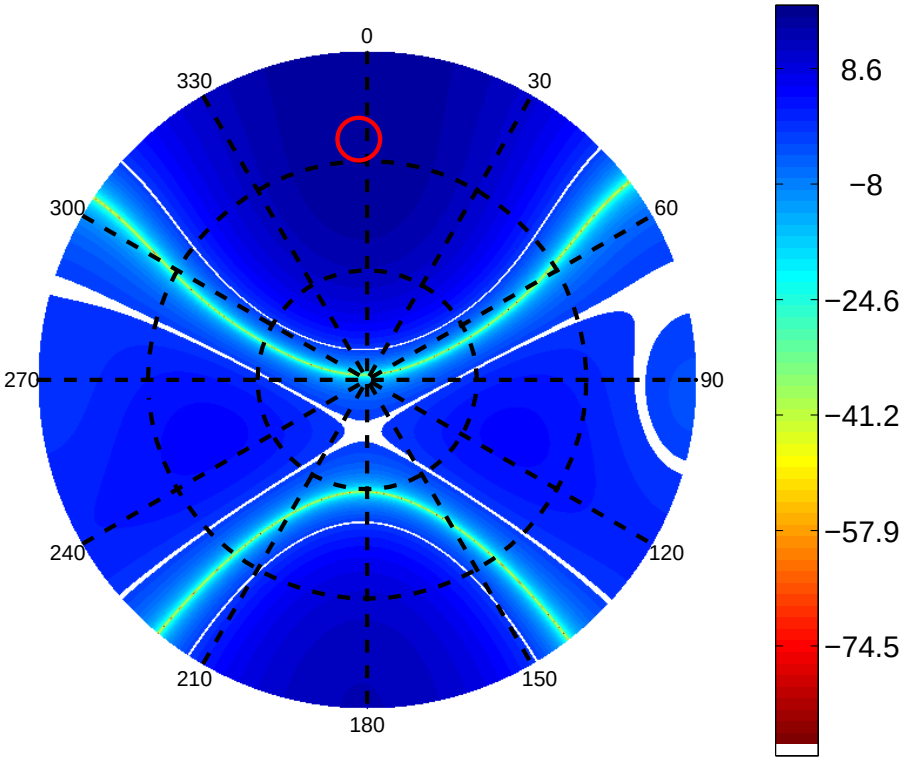


Figure 5.5: Nominal Mode Test Antenna Pattern For Channel 6

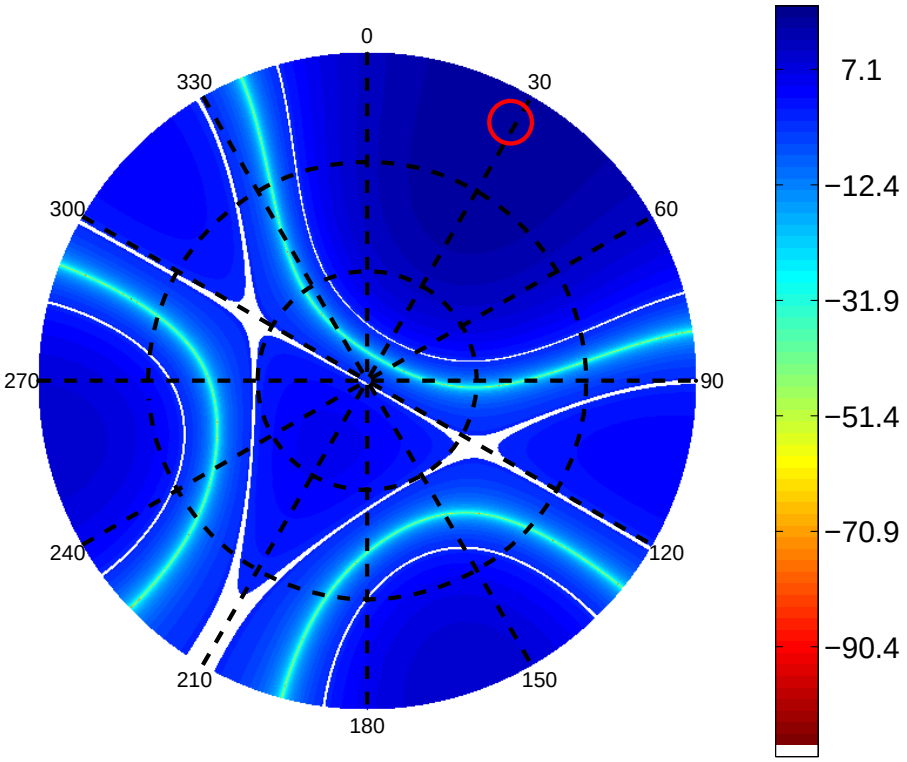


Figure 5.6: Nominal Mode Test Antenna Pattern For Channel 7

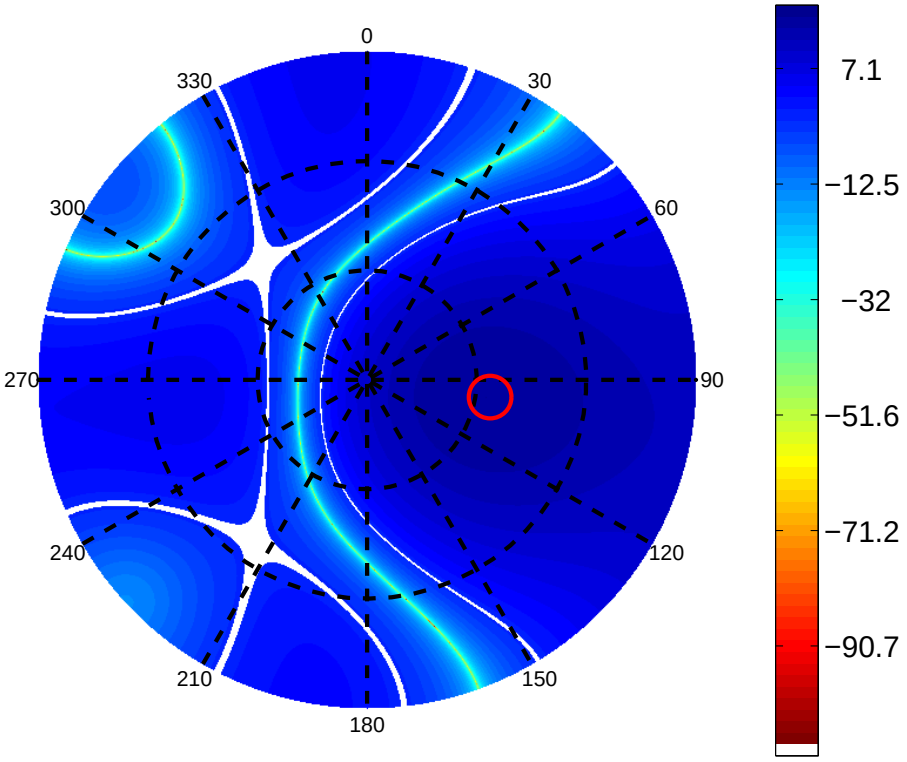


Figure 5.7: Nominal Mode Test Antenna Pattern For Channel 9

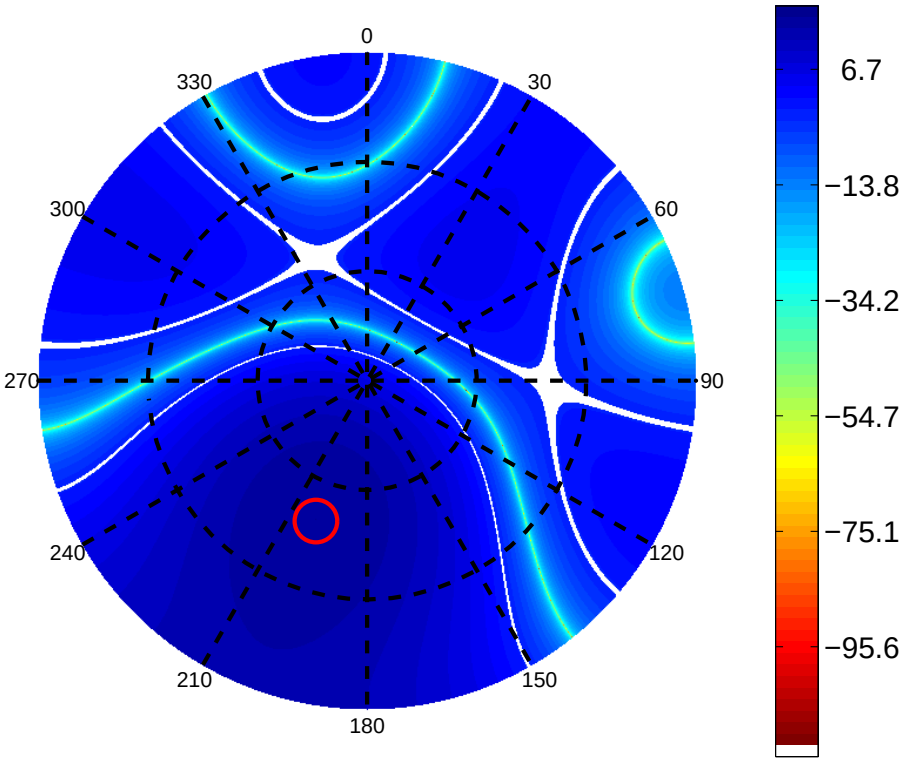


Figure 5.8: Nominal Mode Test Antenna Pattern For Channel 10

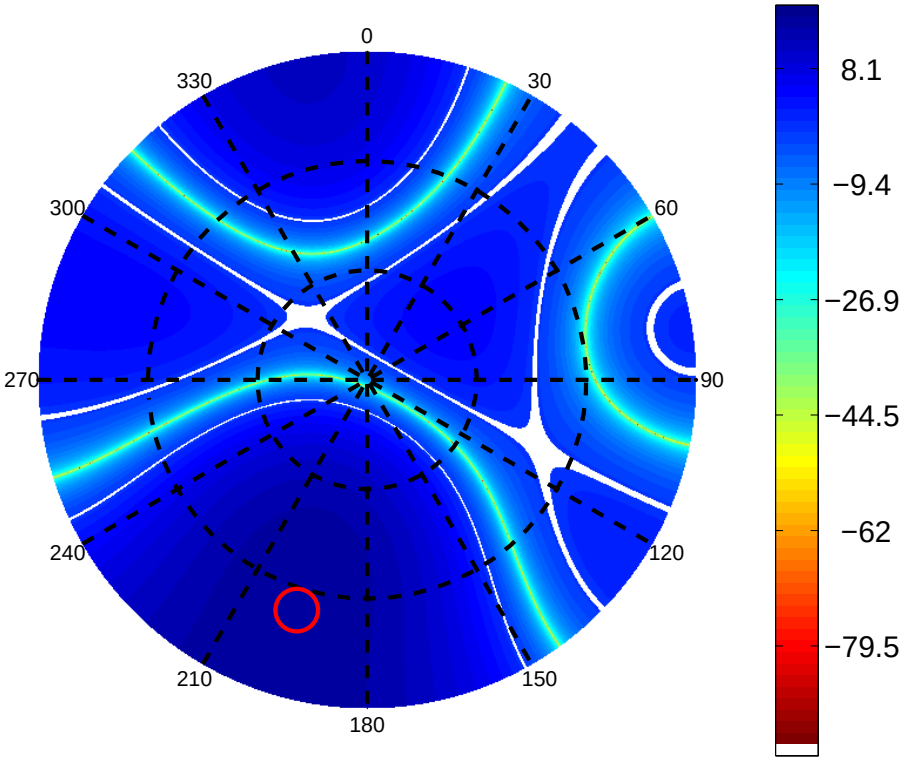


Figure 5.9: Nominal Mode Test Antenna Pattern For Channel 11

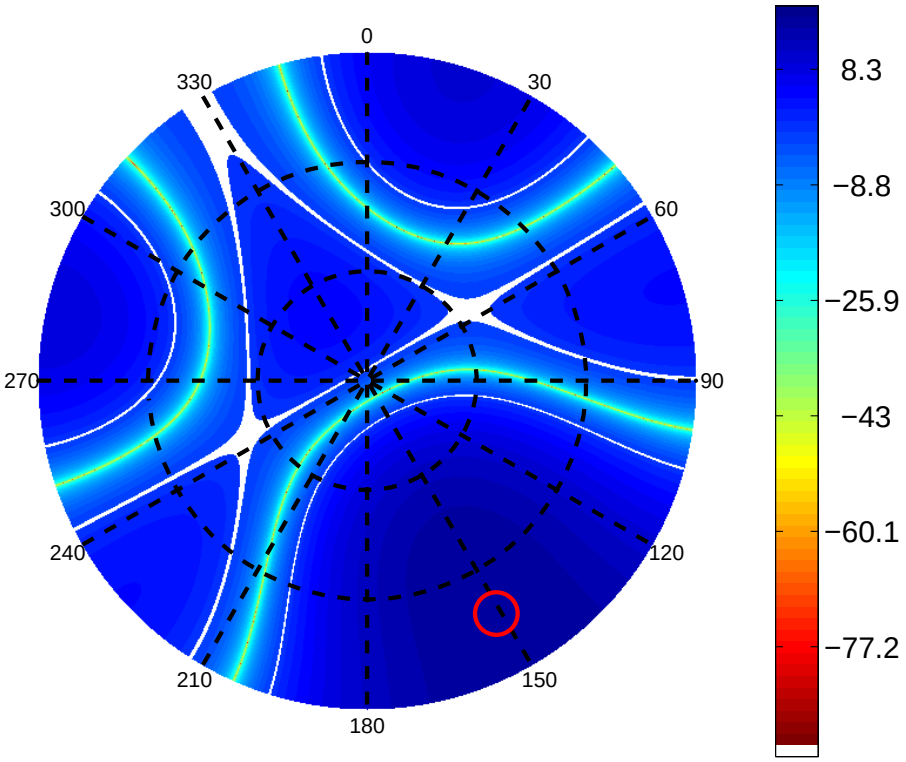


Figure 5.10: Nominal Mode Test Antenna Pattern For Channel 14

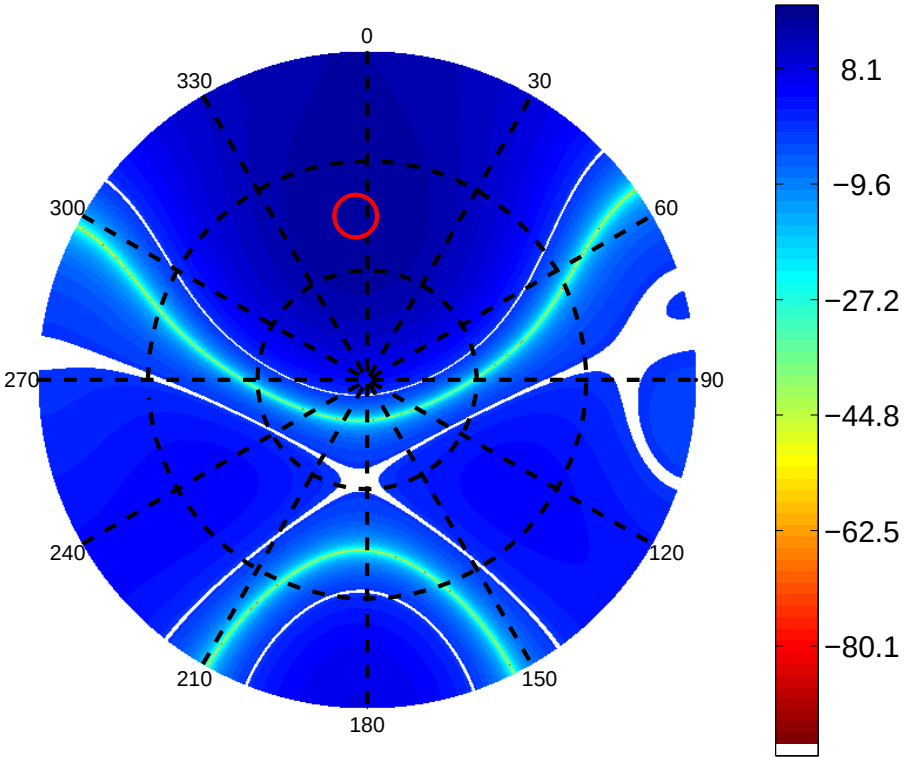


Figure 5.11: Nominal Mode Test Antenna Pattern For Channel 16

Test C2 Nominal Mode + Interferer : automatic Antenna Pattern beam steering in the direction of the tracked SVs, and single null steered in the direction of the interferer zeros.

Nominal mode test sequence and expected results example, automatic Antenna Pattern beam steering in the direction of the tracked SVs with interferer rejection.

Table 5.8: Test C2 Nominal mode test sequence

N#	Command	Expected Response	Note
1	$PSPYN901,GET,SEL*00	$PSPYN901,ACK,SEL,OFF*0C	If the response doesn't match send $PSPYN901,SET,SEL, OFF and repeat step 1
2	$PSPYN901,GET,CXY*00	$PSPYN901,ACK,CXY,7F,95,0,48,82,-48,82,-95,0,-48,-82,48,-82*07	Get the current antenna configuration.
3	$PSPYN901,NVR,ATT,0,0*00	$PSPYN901,ACK,ATT,0,0*74	Set the current antenna array attitude to 0 heading 0 pitch.
4	$PSPYN901,GET,CAL*00	$PSPYN901,ACK,CAL,7F,0.0,1.00,0.0,1.00,0.0,1.00,0.0,1.00,0.0,1.00,0.0,1.00*26	Get the current set of calibration parameters.
5	$PSPYN901,SET,INT,1,45,90*00	$PSPYN901,ACK,INT,1,45,90*73	
6	$PSPYN901,SET,SEL,NOM*00	$PSPYN901,ACK,SEL,NOM*0F	If the response doesn't match send $PSPYN901,SET,SEL, OFF and repeat step 1
7			Wait for the first fix. For each tracked satellite, take a note of the elevation azimuth and channel ID [CHID].
8	$PSPYN901,GET,RAW,1*00 $PSPYN901,GET,RAW,2*00 $PSPYN901,GET,RAW,3*00 $PSPYN901,GET,RAW,4*00 $PSPYN901,GET,RAW,5*00 $PSPYN901,GET,RAW,6*00 $PSPYN901,GET,RAW,7*00 $PSPYN901,GET,RAW,8*00 $PSPYN901,GET,RAW,9*00 $PSPYN901,GET,RAW,10*00 $PSPYN901,GET,RAW,11*00 $PSPYN901,GET,RAW,12*00 $PSPYN901,GET,RAW,13*00 $PSPYN901,GET,RAW,14*00 $PSPYN901,GET,RAW,15*00 $PSPYN901,GET,RAW,16*00	$PSPYN901,ACK,RAW,1,0,1,0,0,0,0,0,0,0,0*6D $PSPYN901,ACK,RAW,2,1,7F,226,281,96,286,231,416,0,0*12 $PSPYN901,ACK,RAW,3,1,7F,386,117,237,126,395,275,0,0*28 $PSPYN901,ACK,RAW,4,1,7F,311,503,215,201,9,297,0,0*2C $PSPYN901,ACK,RAW,5,0,1,0,0,0,0,0,0,0,0*69 $PSPYN901,ACK,RAW,6,1,7F,24,170,229,488,342,283,0,0*1A $PSPYN901,ACK,RAW,7,1,7F,124,249,128,388,263,384,0,0*22 $PSPYN901,ACK,RAW,8,0,1,0,0,0,0,0,0,0,0*64 $PSPYN901,ACK,RAW,9,1,7F,496,407,434,16,105,78,0,0*23 $PSPYN901,ACK,RAW,10,1,7F,8,364,409,504,147,104,0,0*1E $PSPYN901,ACK,RAW,11,1,7F,443,248,376,69,264,136,0,0*29 $PSPYN901,ACK,RAW,12,0,1,0,0,0,0,0,0,0,0*5F $PSPYN901,ACK,RAW,13,0,1,0,0,0,0,0,0,0,0*5E $PSPYN901,ACK,RAW,14,1,7F,115,383,270,397,129,242,0,0*13 $PSPYN901,ACK,RAW,15,0,1,0,0,0,0,0,0,0,0*58 $PSPYN901,ACK,RAW,16,1,7F,23,134,161,489,378,351,0,0*25	Check the offsets using the MATLAB script SPYN901_To_AntennaArrayPatternPlot
9	$PSPYN901,SET,SEL,OFF*00	$PSPYN901,ACK,SEL,OFF*0C	Switch off the beamforming.
10	$PSPYN901,GET,RAW,1*00 $PSPYN901,GET,RAW,2*00 $PSPYN901,GET,RAW,3*00 $PSPYN901,GET,RAW,4*00 $PSPYN901,GET,RAW,5*00 $PSPYN901,GET,RAW,6*00 $PSPYN901,GET,RAW,7*00	$PSPYN901,ACK,RAW,1,0,1,0,0,0,0,0,0,0,0*6D $PSPYN901,ACK,RAW,2,0,1,0,0,0,0,0,0,0,0*6E $PSPYN901,ACK,RAW,3,0,1,0,0,0,0,0,0,0,0*6F $PSPYN901,ACK,RAW,4,0,1,0,0,0,0,0,0,0,0*68 $PSPYN901,ACK,RAW,5,0,1,0,0,0,0,0,0,0,0*69 $PSPYN901,ACK,RAW,6,0,1,0,0,0,0,0,0,0,0*6A $PSPYN901,ACK,RAW,7,0,1,0,0,0,0,0,0,0,0*6B	Check that each channel beamformer is set back to its default configuration.

	$PSPYN901,GET,RAW,8*00	$PSPYN901,ACK,RAW,8,0,1,0,0,0,0,0,0,0,0*64	
	$PSPYN901,GET,RAW,9*00	$PSPYN901,ACK,RAW,9,0,1,0,0,0,0,0,0,0,0*65	
	$PSPYN901,GET,RAW,10*00	$PSPYN901,ACK,RAW,10,0,1,0,0,0,0,0,0,0,0*5D	
	$PSPYN901,GET,RAW,11*00	$PSPYN901,ACK,RAW,11,0,1,0,0,0,0,0,0,0,0*5C	
	$PSPYN901,GET,RAW,12*00	$PSPYN901,ACK,RAW,12,0,1,0,0,0,0,0,0,0,0*5F	
	$PSPYN901,GET,RAW,13*00	$PSPYN901,ACK,RAW,13,0,1,0,0,0,0,0,0,0,0*5E	
	$PSPYN901,GET,RAW,14*00	$PSPYN901,ACK,RAW,14,0,1,0,0,0,0,0,0,0,0*59	
	$PSPYN901,GET,RAW,15*00	$PSPYN901,ACK,RAW,15,0,1,0,0,0,0,0,0,0,0*58	
	$PSPYN901,GET,RAW,16*00	$PSPYN901,ACK,RAW,16,0,1,0,0,0,0,0,0,0,0*5B	

In the following figures are reported the results for test C2. The Antenna Array map used for the test is shown in figure 5.12. Plots from figure 5.13 to figure 5.20 show the antenna patterns synthesized by each channel: each antenna pattern is built using as input the logged receiver messages, in particular the beamer parameters are extracted from the RAW message. Figure 5.14 and figure 5.15 show the current behaviour of the receiver when the angular position of the required maximum and the angular position of the zero are very close.

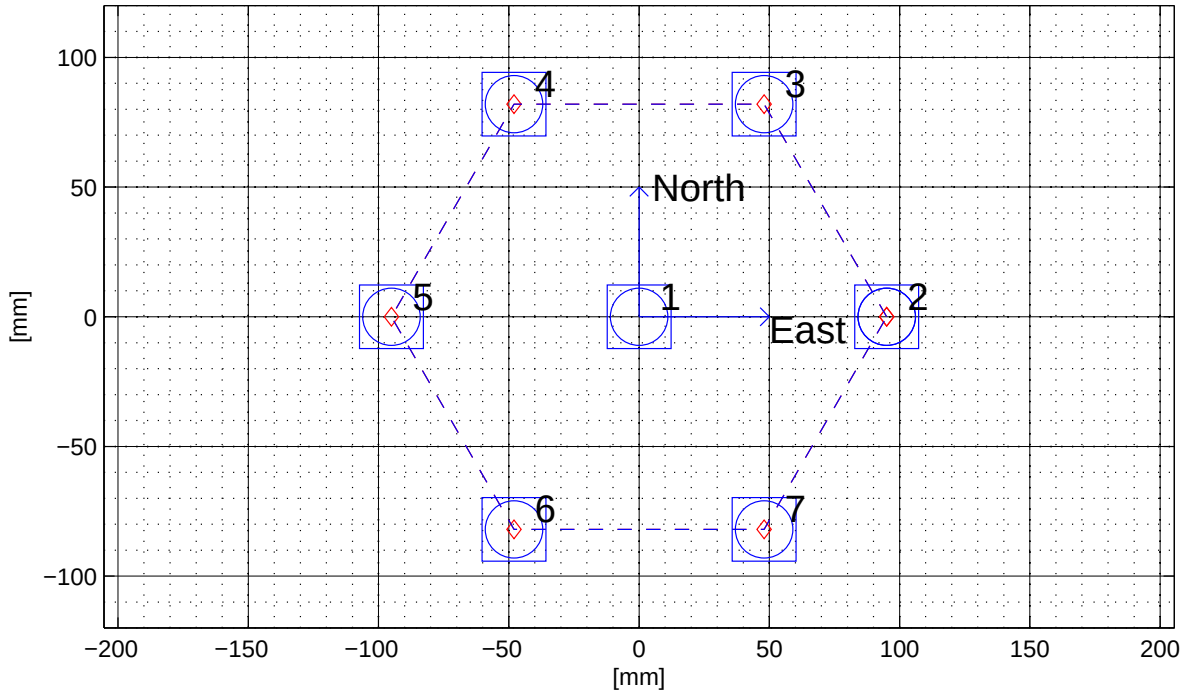


Figure 5.12: Antenna Array Map

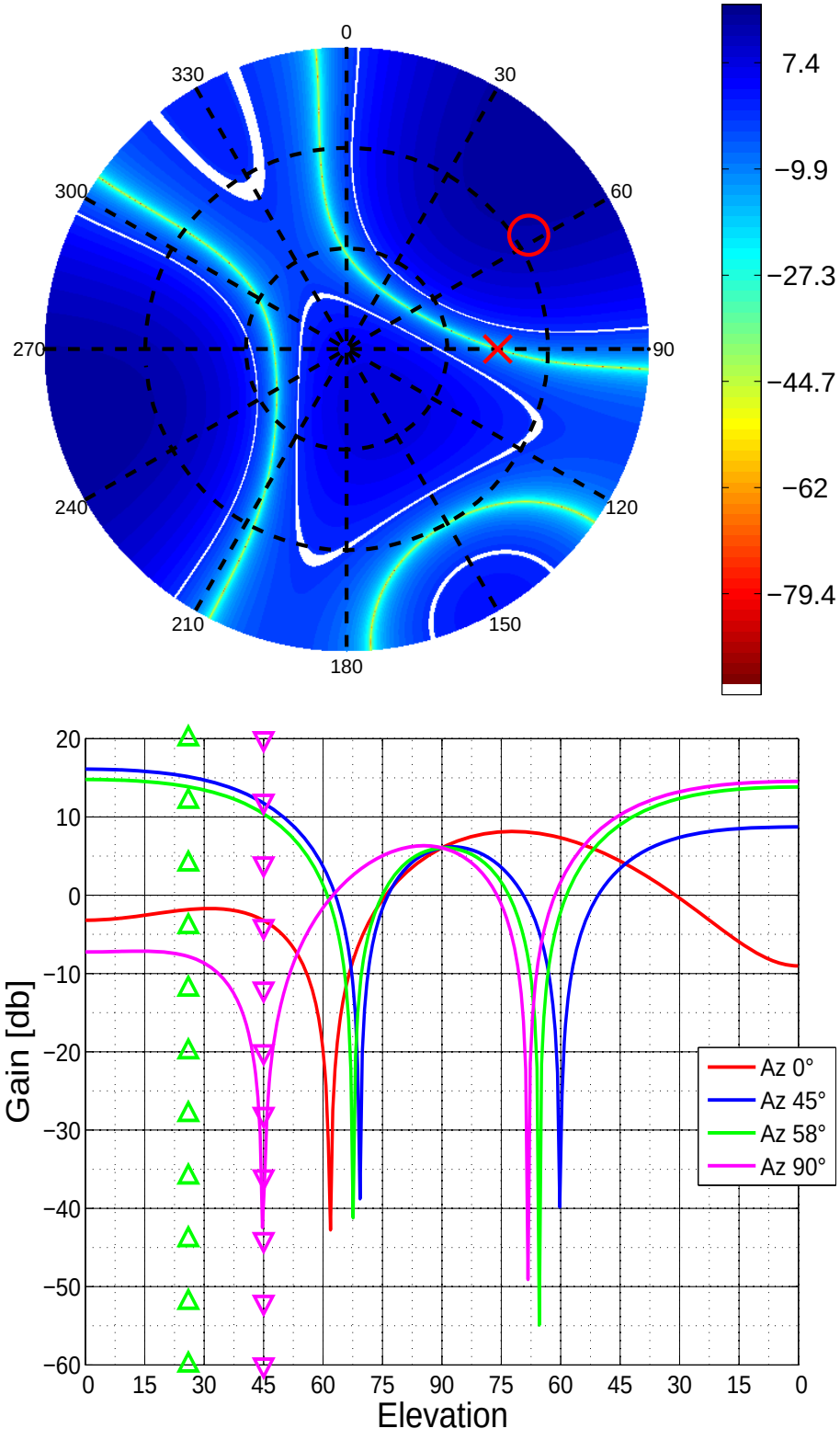


Figure 5.13: Antenna Pattern for Channel 2

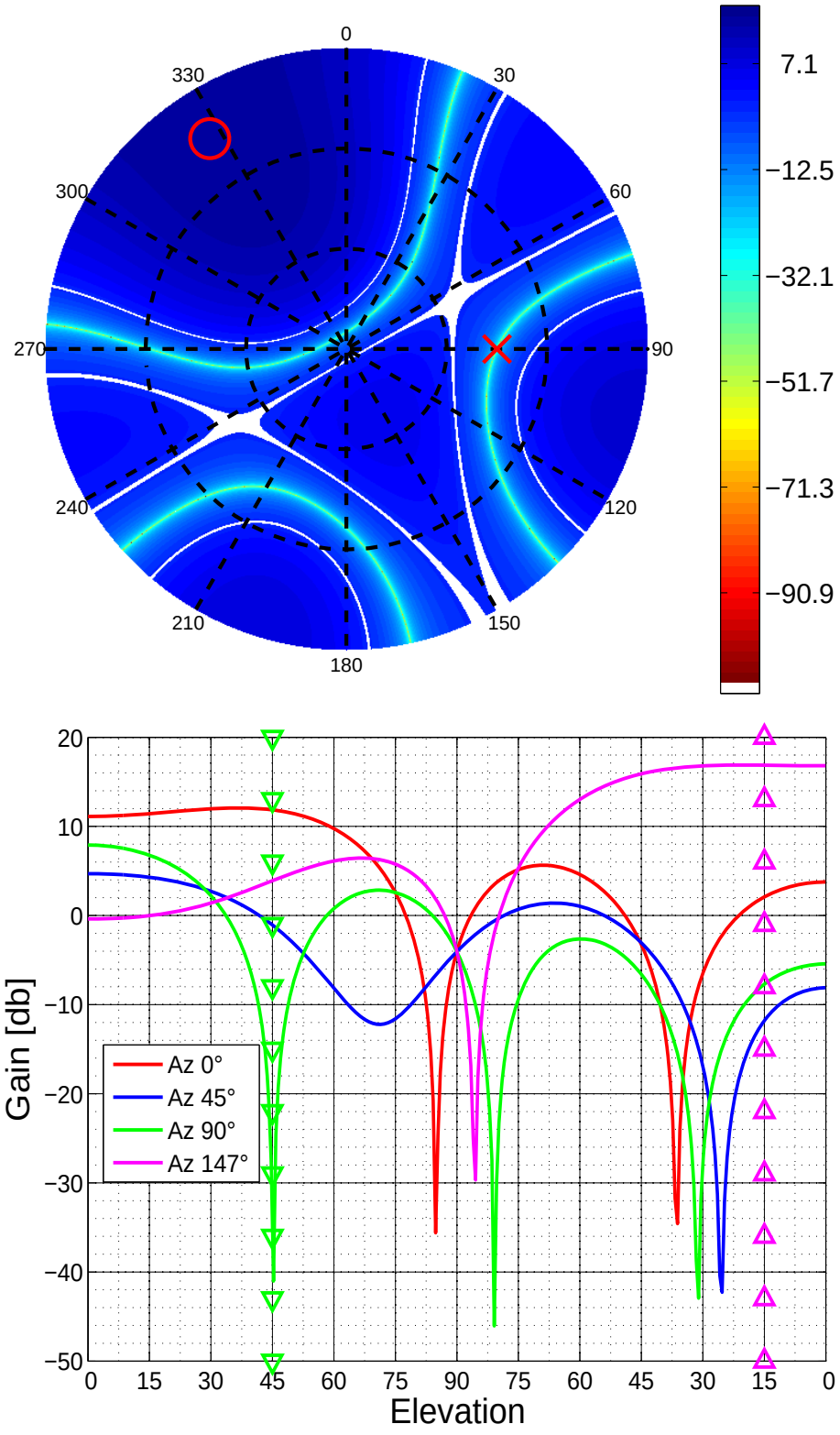


Figure 5.14: Antenna Pattern for Channel 3

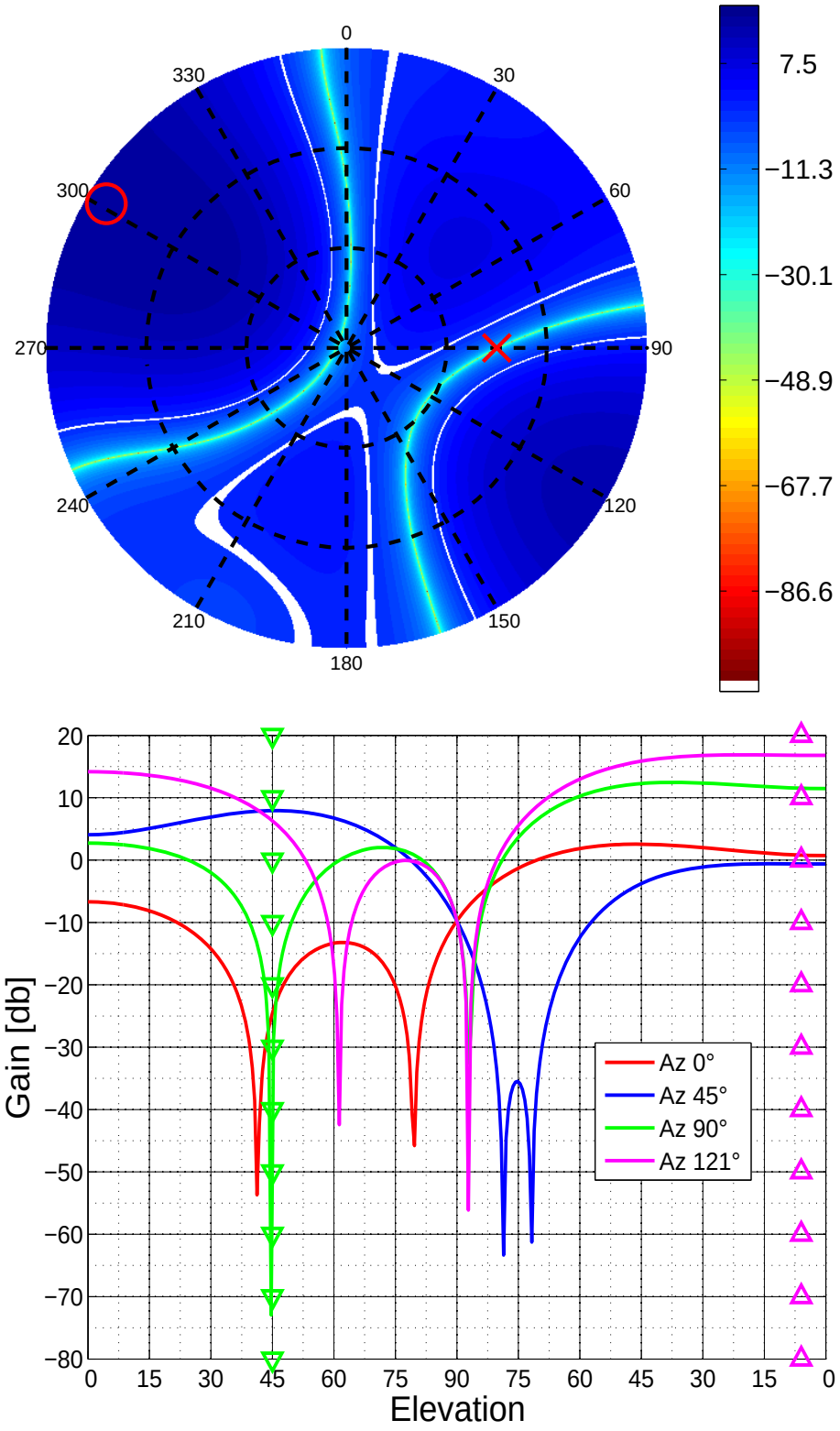


Figure 5.15: Antenna Pattern for Channel 4

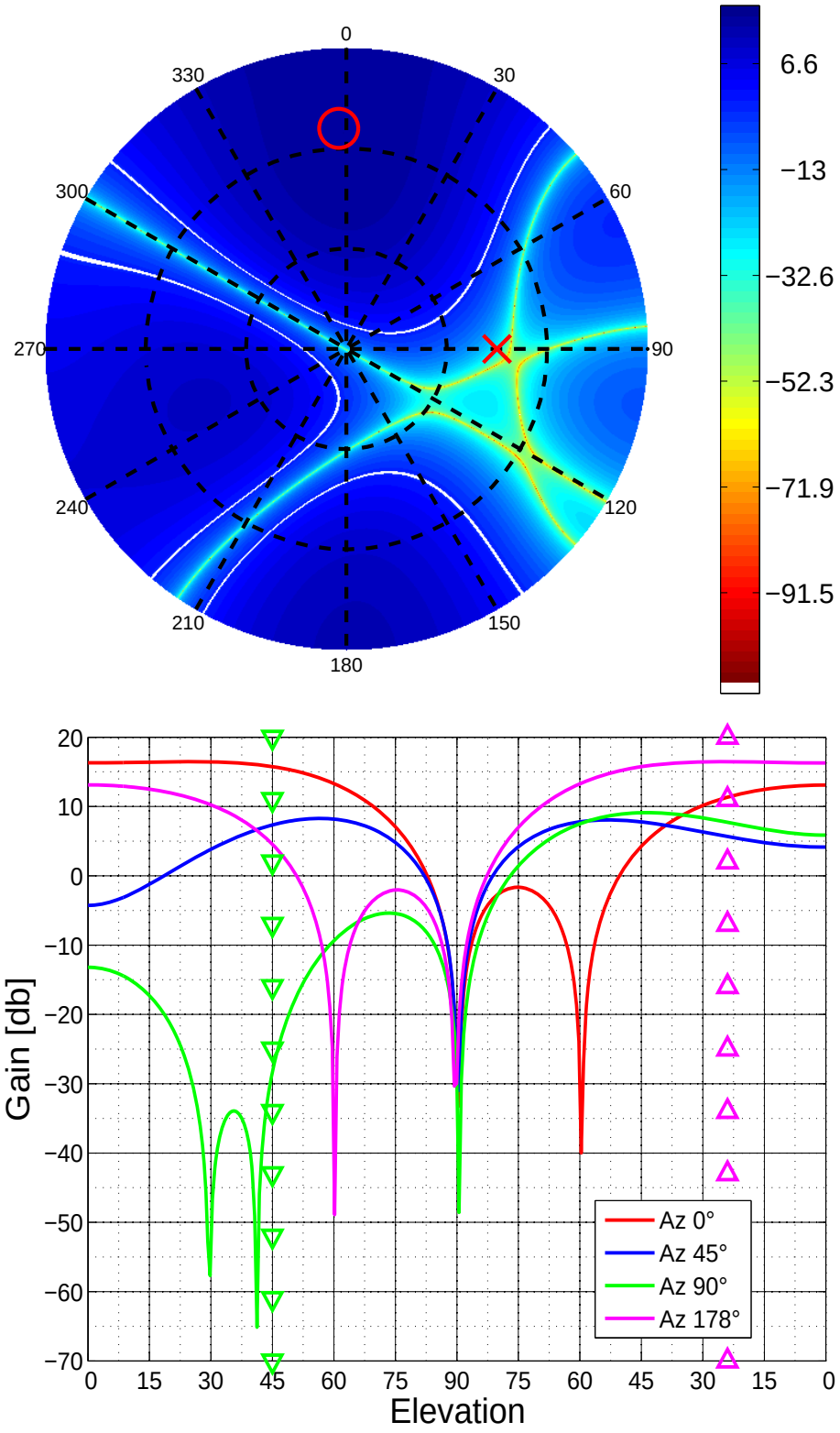


Figure 5.16: Antenna Pattern for Channel 6

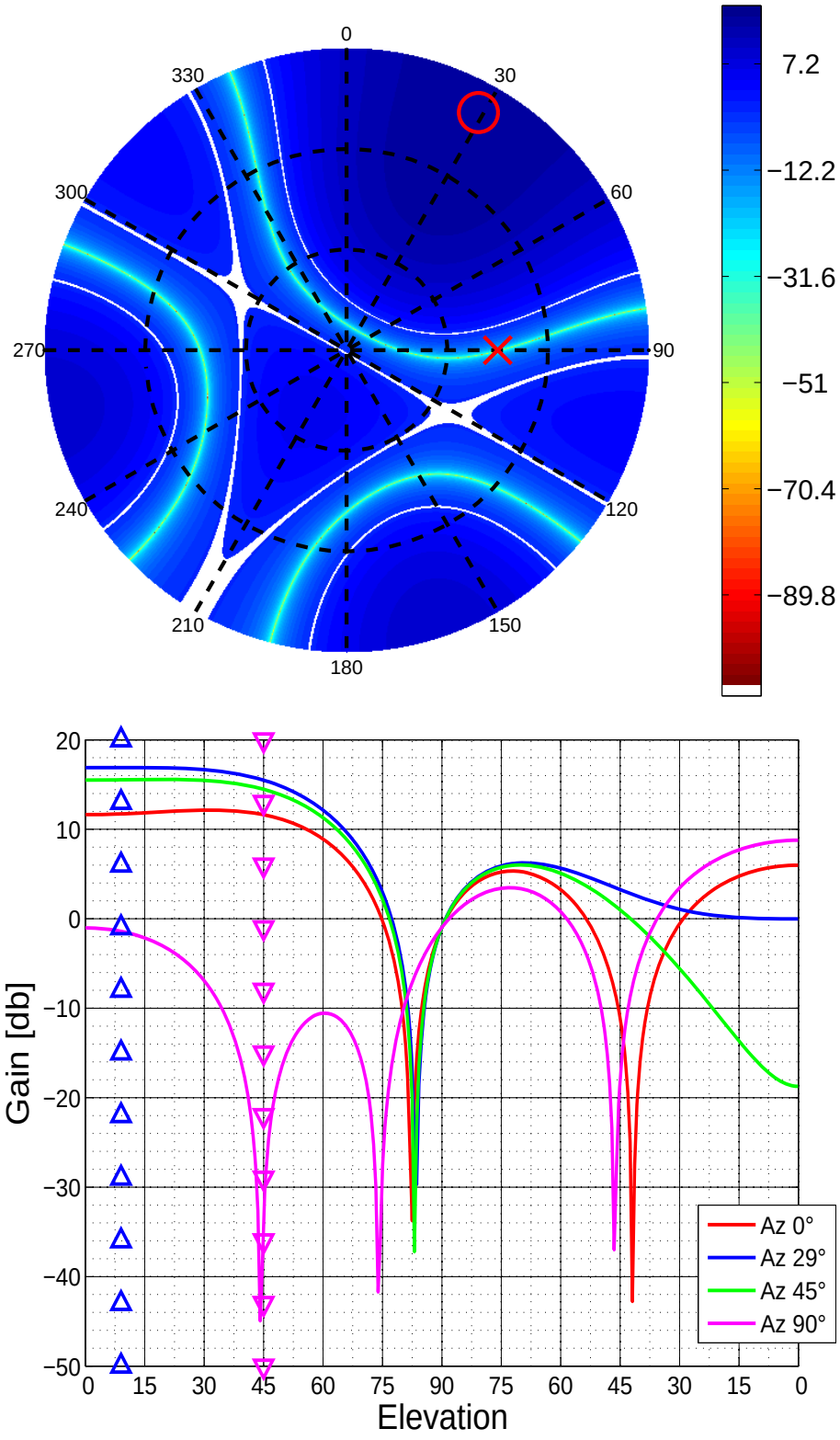


Figure 5.17: Antenna Pattern for Channel 7

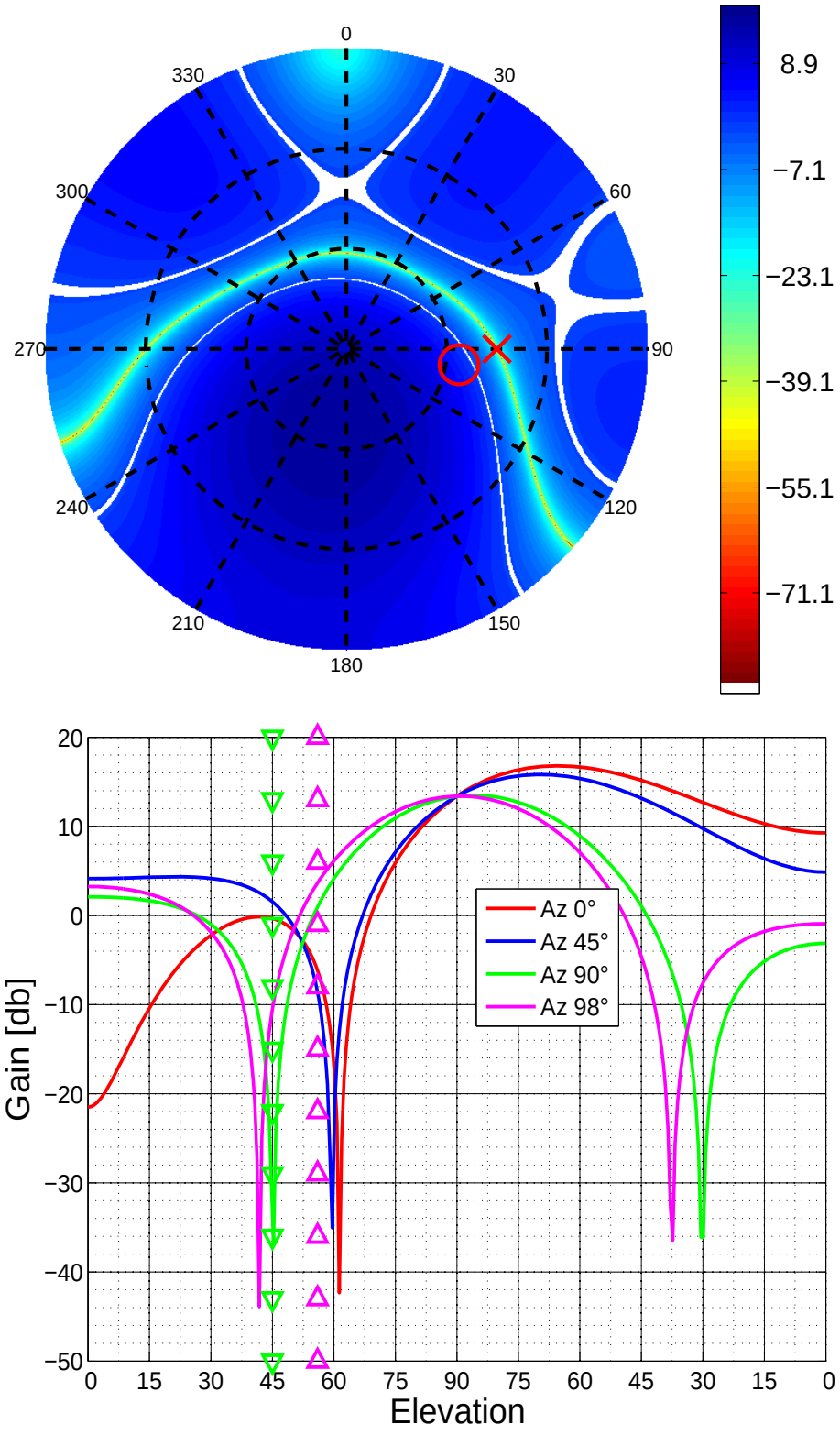


Figure 5.18: Antenna Pattern for Channel 9

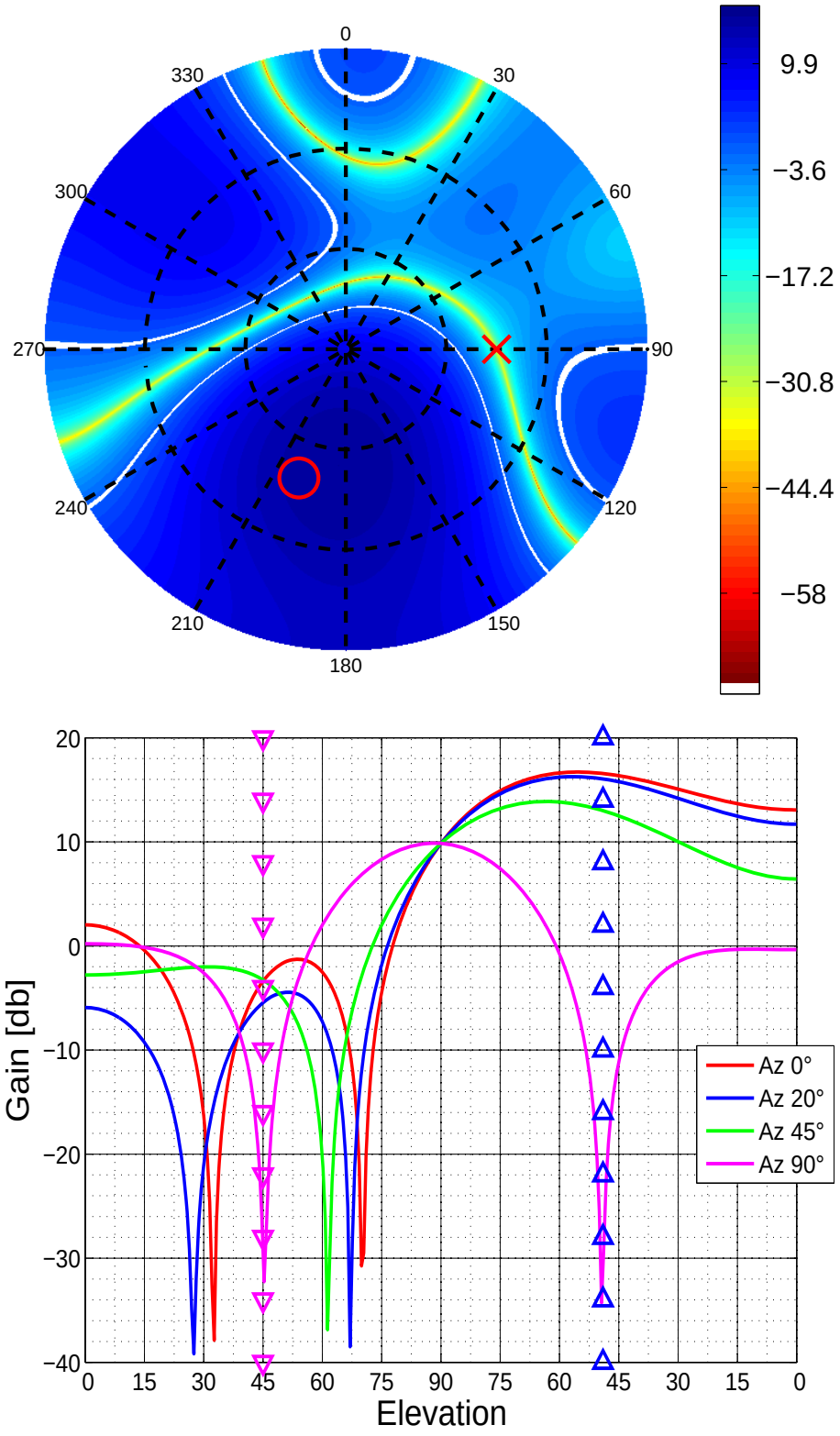


Figure 5.19: Antenna Pattern for Channel 10

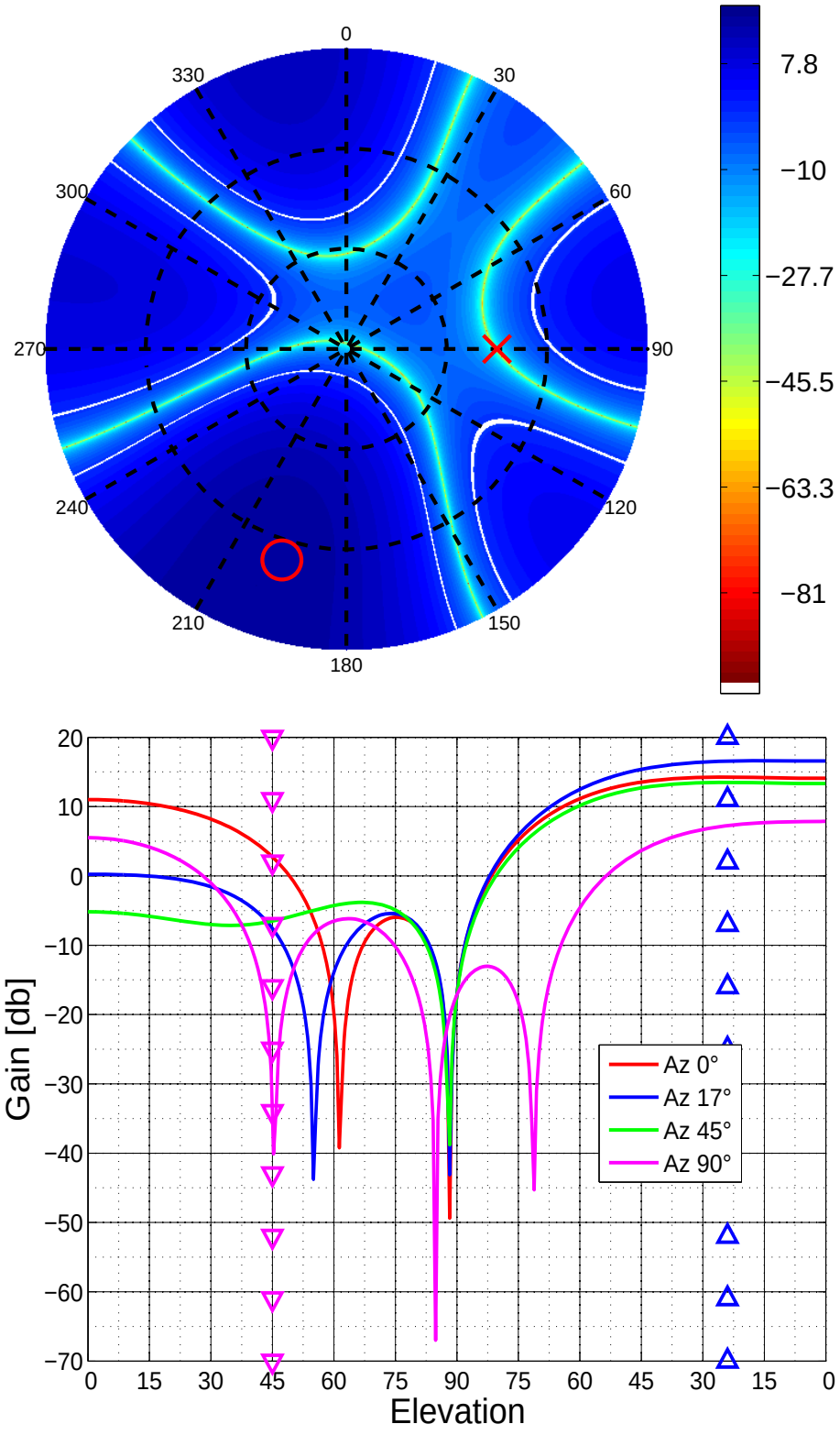


Figure 5.20: Antenna Pattern for Channel 11

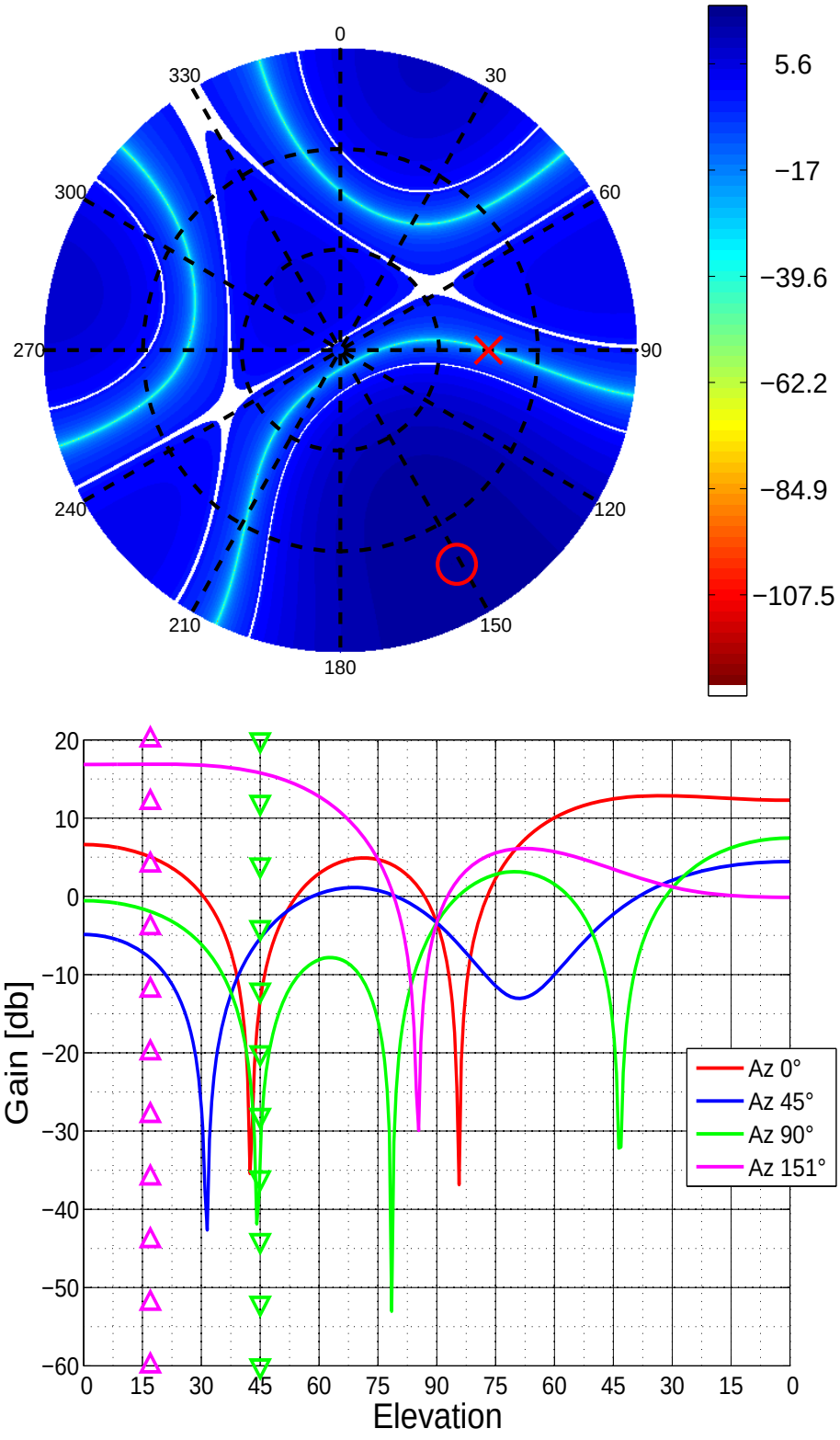


Figure 5.21: Antenna Pattern for Channel 14

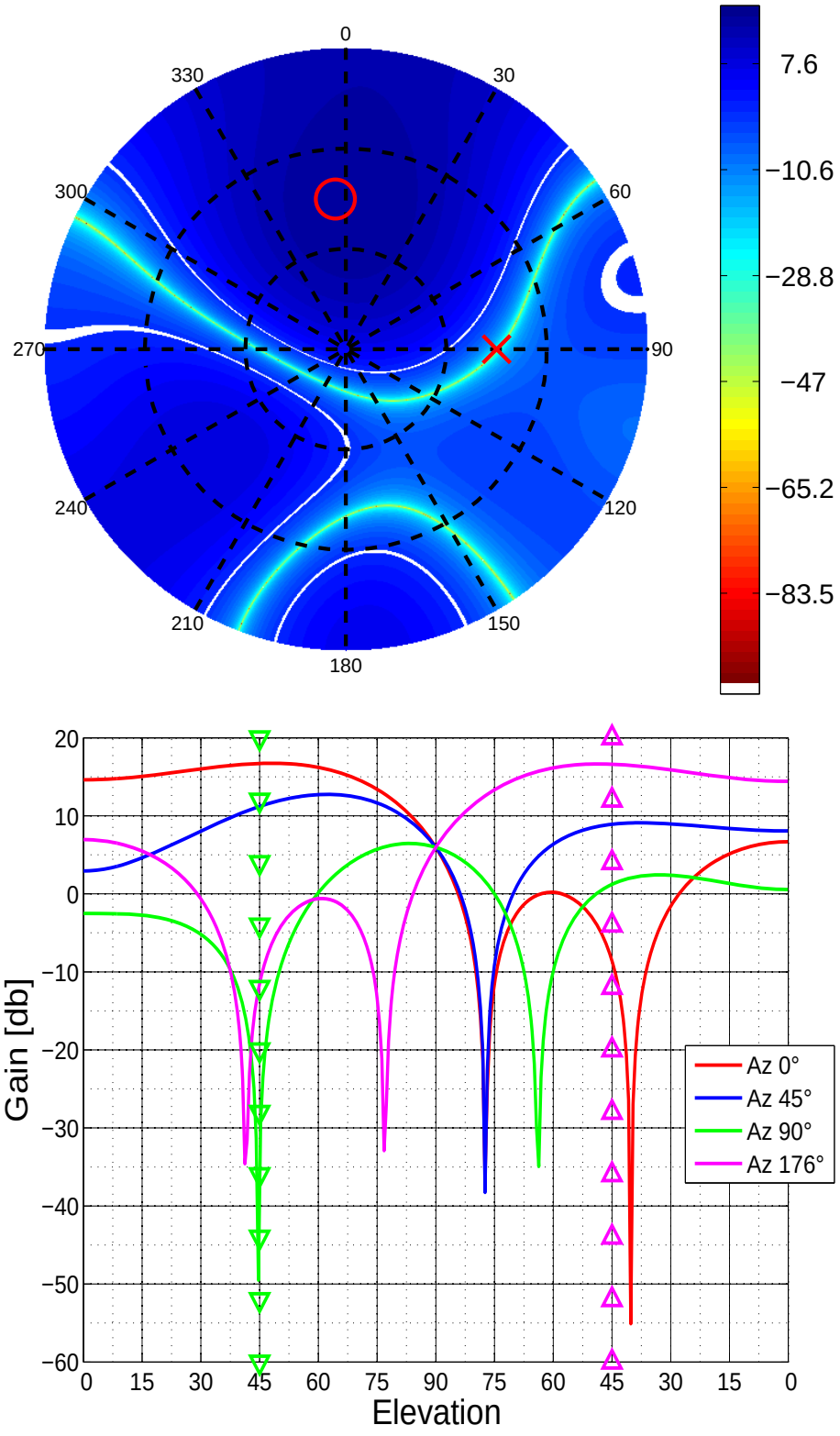


Figure 5.22: Antenna Pattern for Channel 16

Test C3 Nominal Mode + Interferer + 9 antenna circular configuration : automatic Antenna Pattern beam steering in the direction of the tracked SVs, and single null steered in the direction of the interferer zeros, all the 9 RF inputs are used.

Table 5.9: C3 Nominal mode test sequence and expected results example

N#	Command	Expected Response	Note
1	$PSPYN901,GET,SEL*00	$PSPYN901,ACK,SEL,OFF*0C	If the response doesn't match send $PSPYN901,SET,SEL, OFF and repeat step 1
2	$PSPYN901,SET,CPO,1FF,95,0,95,45,95 ,90,95,135,95,180,95,225,95,270,95,315* 00	$PSPYN901,ACK,CXY,1FF,95,0,67,67,0,95, -67,67,-95,0,-67,-67,0,-95,67,-67*6A	
2	$PSPYN901,GET,CXY*00	$PSPYN901,ACK,CXY,1FF,95,0,67,67,0,95, -67,67,-95,0,-67,-67,0,-95,67,-67*6A	Get the current antenna configuration.
3	$PSPYN901,NVR,ATT,0,0*00	$PSPYN901,ACK,ATT,0,0*74	Set the current antenna array attitude to 0 heading 0 pitch.
4	$PSPYN901,GET,CAL*00	$PSPYN901,ACK,CAL,7F,0.0,1.00,0.0,1.00, 0.0,1.00,0.0,1.00,0.0,1.00,0.0,1.00*26	Get the current set of calibration parameters.
5	$PSPYN901,SET,INT,1,45,90*00	$PSPYN901,ACK,INT,1,45,90*73	
5	$PSPYN901,SET,SEL,NOM*00	$PSPYN901,ACK,SEL,NOM*0F	If the response doesn't match send $PSPYN901,SET,SEL, OFF and repeat step 1
6			Wait for the first fix. For each tracked satellite, take a note of the elevation azimuth and channel ID [CHID].
7	$PSPYN901,GET,RAW,1*00 $PSPYN901,GET,RAW,2*00 $PSPYN901,GET,RAW,3*00 $PSPYN901,GET,RAW,4*00 $PSPYN901,GET,RAW,5*00 $PSPYN901,GET,RAW,6*00 $PSPYN901,GET,RAW,7*00 $PSPYN901,GET,RAW,8*00 $PSPYN901,GET,RAW,9*00 $PSPYN901,GET,RAW,10*00 $PSPYN901,GET,RAW,11*00 $PSPYN901,GET,RAW,12*00 $PSPYN901,GET,RAW,13*00 $PSPYN901,GET,RAW,14*00 $PSPYN901,GET,RAW,15*00 $PSPYN901,GET,RAW,16*00	$PSPYN901,ACK,RAW,1,0,1,0,0,0,0,0,0,0,0* 6D $PSPYN901,ACK,RAW,2,2,1FF,216,297,187 ,22,297,215,325,490*58 $PSPYN901,ACK,RAW,3,2,1FF,362,67,193, 263,150,445,319,249*52 $PSPYN901,ACK,RAW,4,2,1FF,291,455,129 ,251,221,57,383,261*56 $PSPYN901,ACK,RAW,5,0,1,0,0,0,0,0,0,0* 69 $PSPYN901,ACK,RAW,6,2,1FF,46,145,211, 209,466,367,301,303*57 $PSPYN901,ACK,RAW,7,2,1FF,128,233,216 ,62,384,279,296,450*5E $PSPYN901,ACK,RAW,8,0,1,0,0,0,0,0,0,0* 64 $PSPYN901,ACK,RAW,9,2,1FF,31,503,357, 493,482,10,155,19*5A $PSPYN901,ACK,RAW,10,2,1FF,474,335,41 0,420,38,177,102,92*5C $PSPYN901,ACK,RAW,11,2,1FF,445,243,32 5,391,67,269,187,121*61 $PSPYN901,ACK,RAW,12,0,1,0,0,0,0,0,0,0, 0*5F $PSPYN901,ACK,RAW,13,0,1,0,0,0,0,0,0,0, 0*5E $PSPYN901,ACK,RAW,14,2,1FF,119,445,30 1,286,393,67,211,226*65 $PSPYN901,ACK,RAW,15,0,1,0,0,0,0,0,0,0, 0*58 $PSPYN901,ACK,RAW,16,2,1FF,24,122,142 ,141,488,390,370,371*62	Check the offsets using the MATLAB script SPYN901_To_Antenna ArrayPatternPlot
8	$PSPYN901,SET,SEL,OFF*00	$PSPYN901,ACK,SEL,OFF*0C	Switch off the beamforming.
9	$PSPYN901,GET,RAW,1*00	$PSPYN901,ACK,RAW,1,0,1,0,0,0,0,0,0,0*	Chech that each

	$PSPYN901,GET,RAW,2*00 $PSPYN901,GET,RAW,3*00 $PSPYN901,GET,RAW,4*00 $PSPYN901,GET,RAW,5*00 $PSPYN901,GET,RAW,6*00 $PSPYN901,GET,RAW,7*00 $PSPYN901,GET,RAW,8*00 $PSPYN901,GET,RAW,9*00 $PSPYN901,GET,RAW,10*00 $PSPYN901,GET,RAW,11*00 $PSPYN901,GET,RAW,12*00 $PSPYN901,GET,RAW,13*00 $PSPYN901,GET,RAW,14*00 $PSPYN901,GET,RAW,15*00 $PSPYN901,GET,RAW,16*00	6D $PSPYN901,ACK,RAW,2,0,1,0,0,0,0,0,0,0,0*6E $PSPYN901,ACK,RAW,3,0,1,0,0,0,0,0,0,0,0*6F $PSPYN901,ACK,RAW,4,0,1,0,0,0,0,0,0,0,0*68 $PSPYN901,ACK,RAW,5,0,1,0,0,0,0,0,0,0,0*69 $PSPYN901,ACK,RAW,6,0,1,0,0,0,0,0,0,0,0*6A $PSPYN901,ACK,RAW,7,0,1,0,0,0,0,0,0,0,0*6B $PSPYN901,ACK,RAW,8,0,1,0,0,0,0,0,0,0,0*64 $PSPYN901,ACK,RAW,9,0,1,0,0,0,0,0,0,0,0*65 $PSPYN901,ACK,RAW,10,0,1,0,0,0,0,0,0,0,0*5D $PSPYN901,ACK,RAW,11,0,1,0,0,0,0,0,0,0,0*5C $PSPYN901,ACK,RAW,12,0,1,0,0,0,0,0,0,0,0*5F $PSPYN901,ACK,RAW,13,0,1,0,0,0,0,0,0,0,0*5E $PSPYN901,ACK,RAW,14,0,1,0,0,0,0,0,0,0,0*59 $PSPYN901,ACK,RAW,15,0,1,0,0,0,0,0,0,0,0*58 $PSPYN901,ACK,RAW,16,0,1,0,0,0,0,0,0,0,0*5B	channel beamformer is set back to its default configuration.

In the following figures are reported the results for test C3. The Antenna Array map used for the test is shown in figure 5.23. Plots from figure 5.24 to figure 5.31 show the antenna patterns synthesized by each channel : each antenna pattern is built using as input the logged receiver messages, in particular the beamer parameters are extracted from the RAW message. Figure 5.25 and figure 5.26 show the current behavior of the receiver when the angular position of the required maximum and the angular position of the zero are very close.

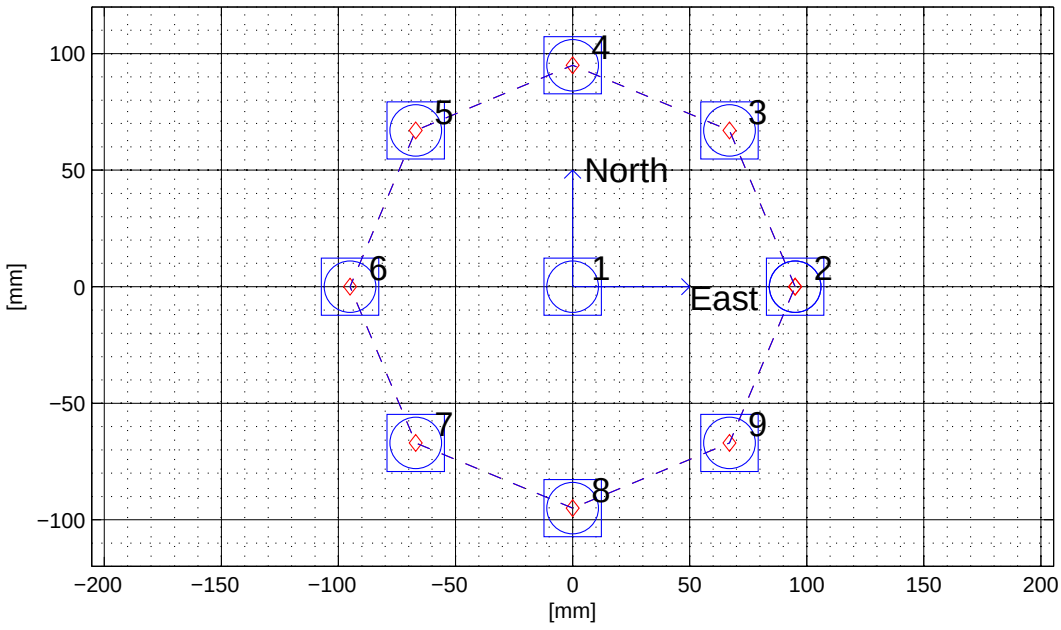


Figure 5.23: Antenna Array Map

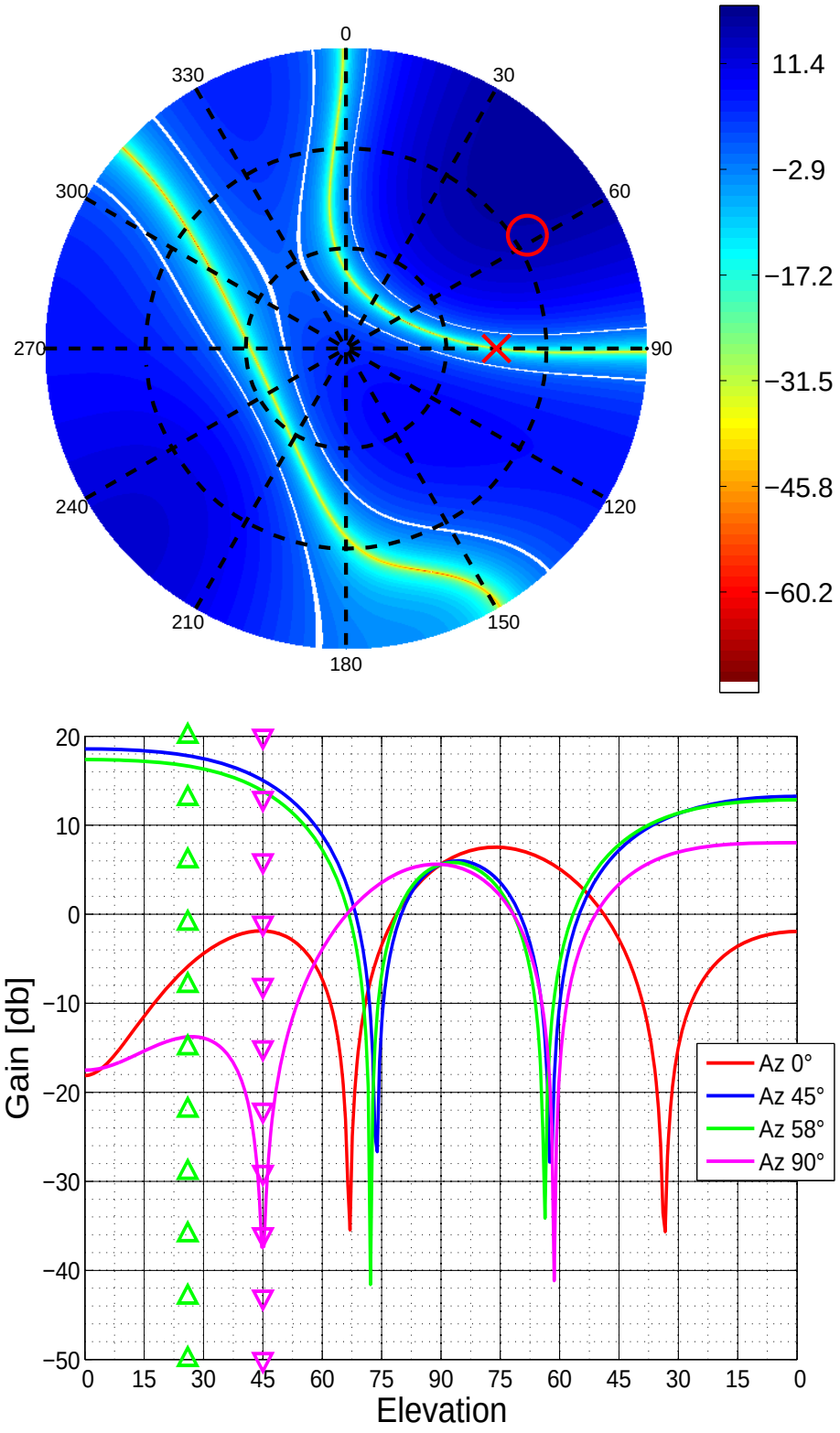
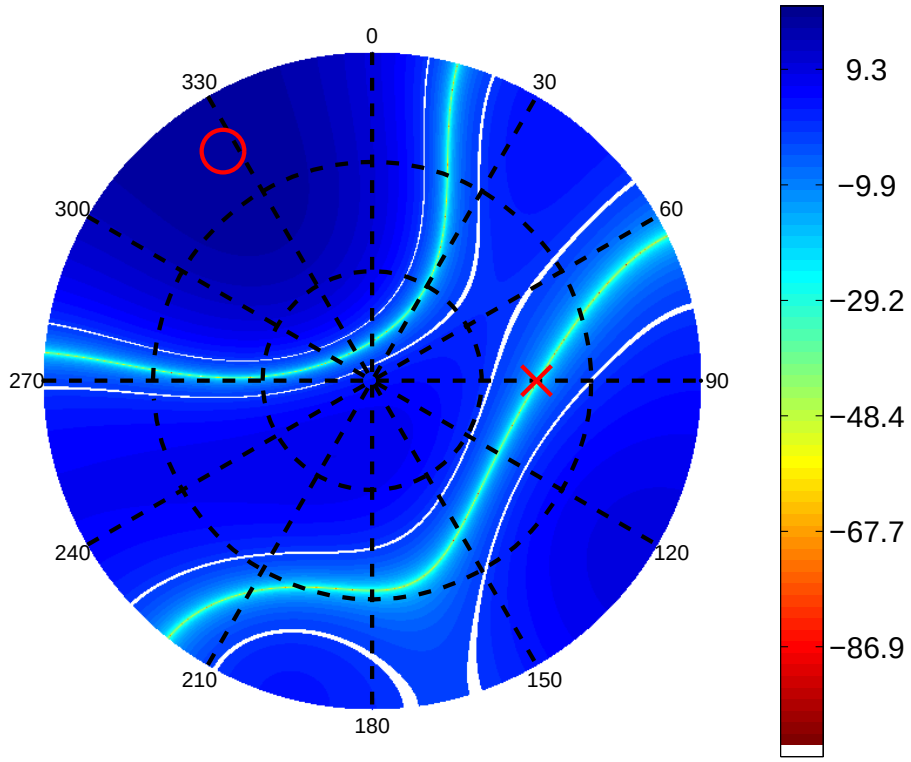


Figure 5.24: Antenna Pattern for Channel 2



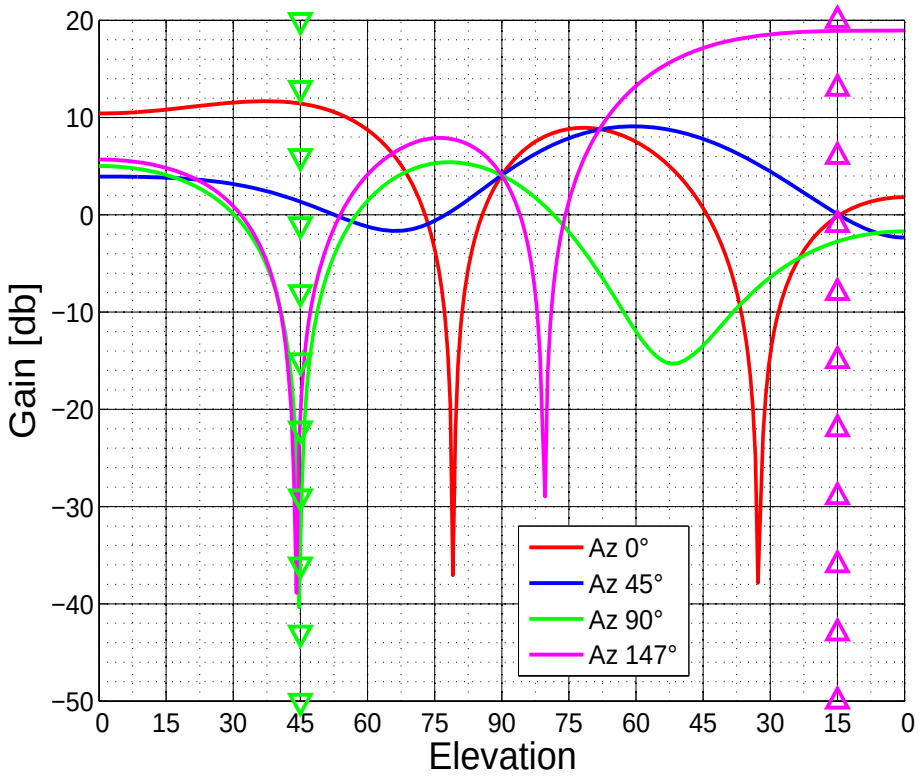


Figure 5.25: Antenna Pattern for Channel 3

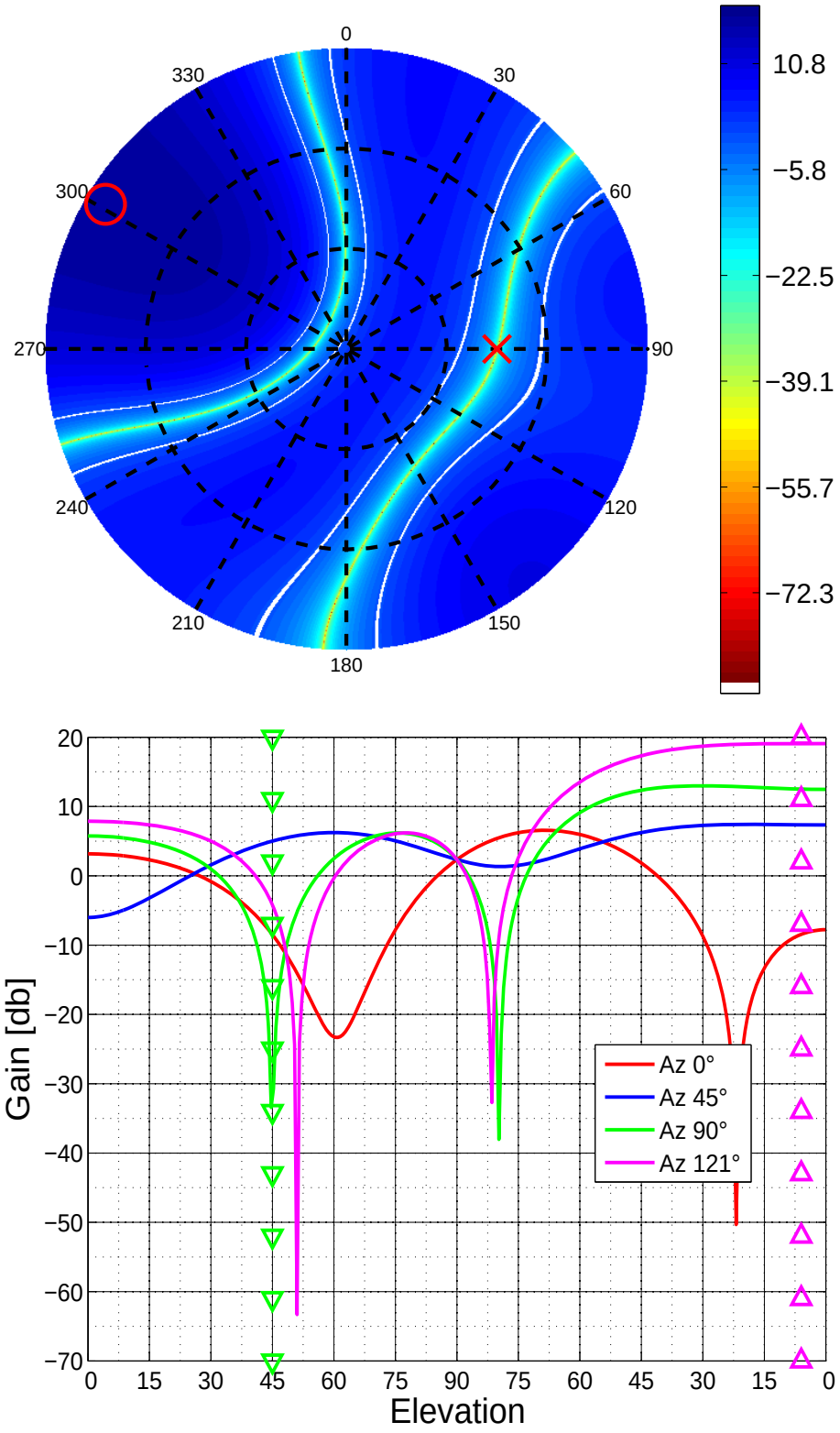


Figure 5.26: Antenna Pattern for Channel 4

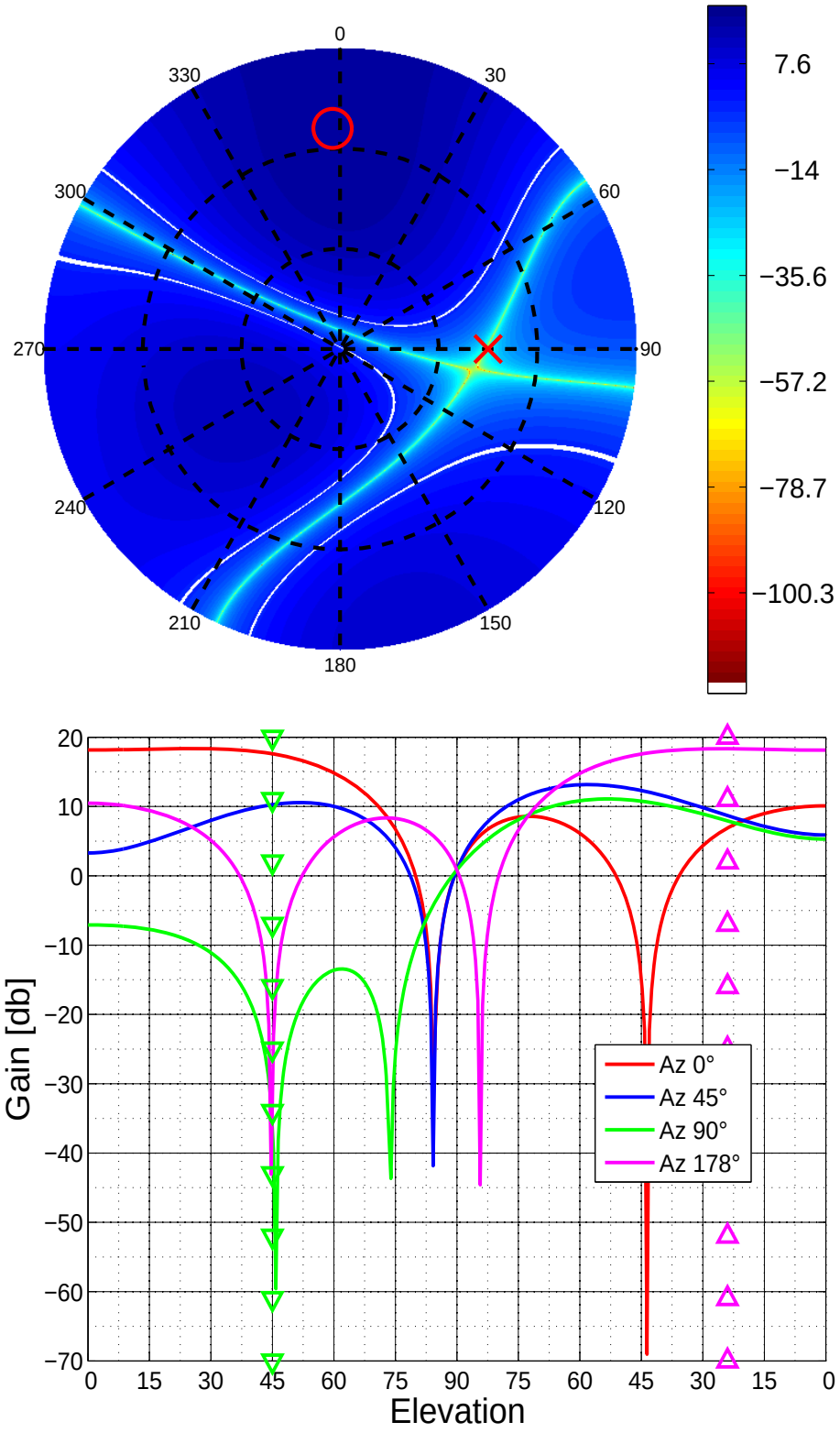


Figure 5.27: Antenna Pattern for Channel 6

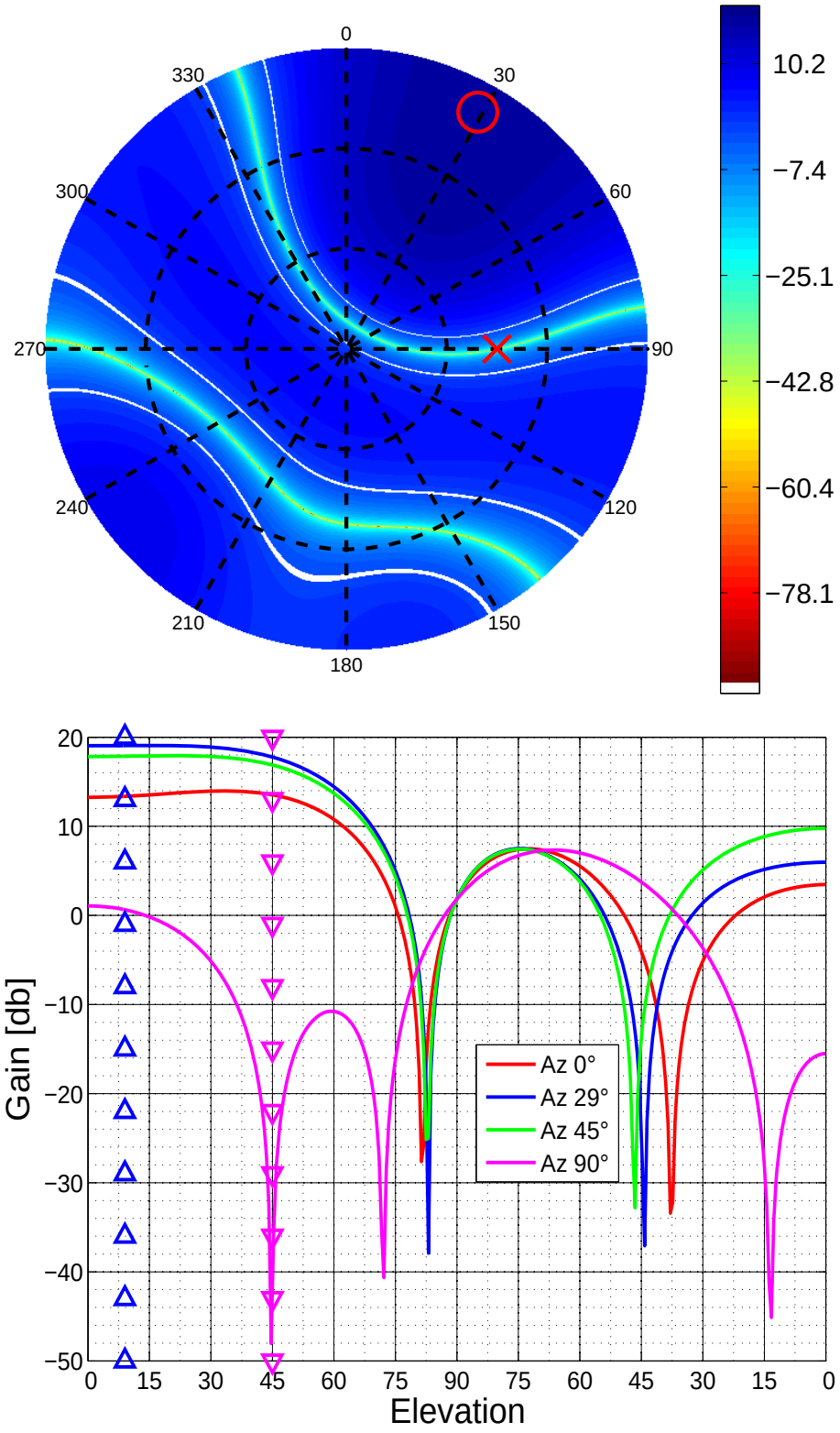


Figure 5.28: Antenna Pattern for Channel 7

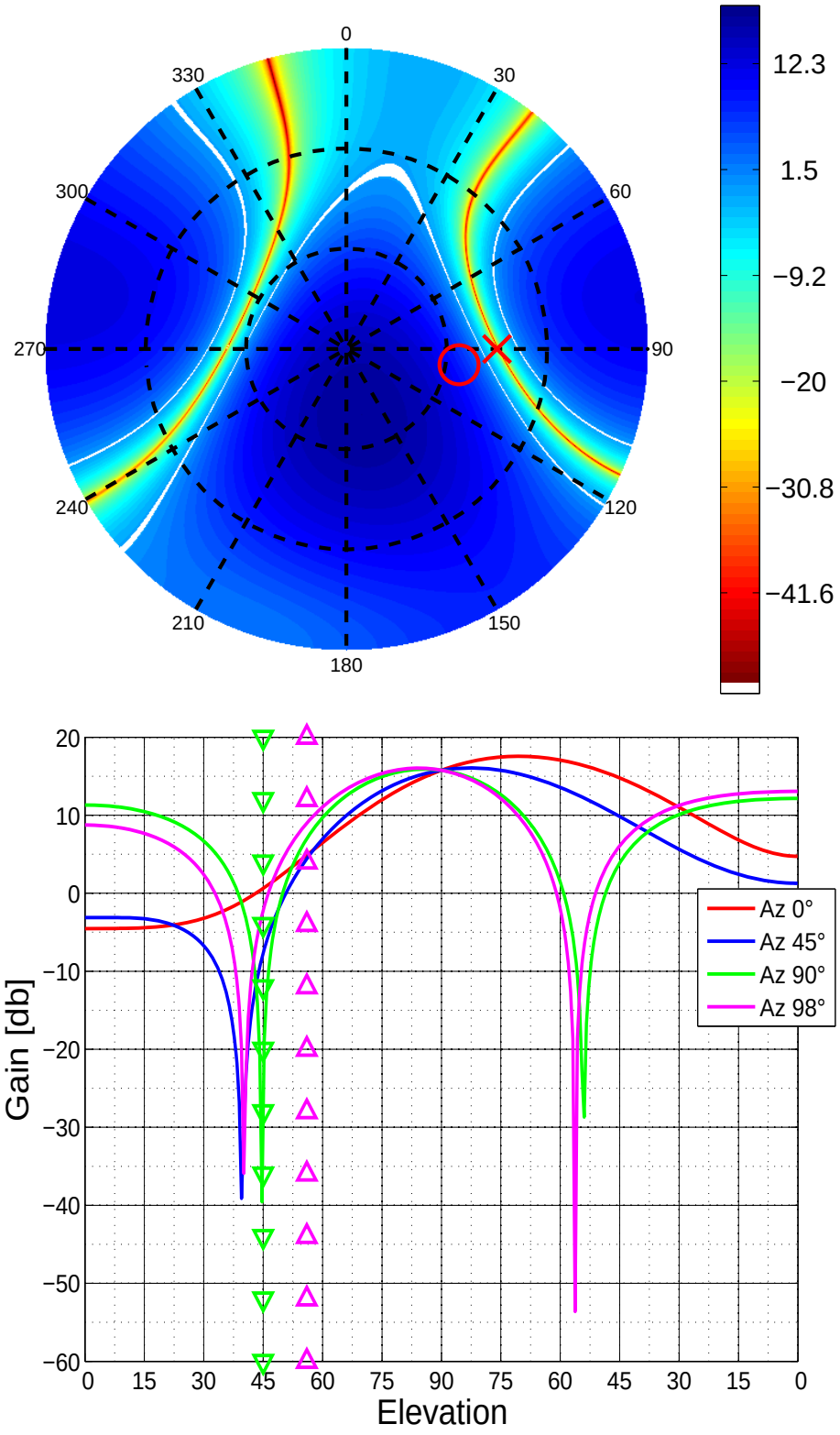


Figure 5.29: Antenna Pattern for Channel 9

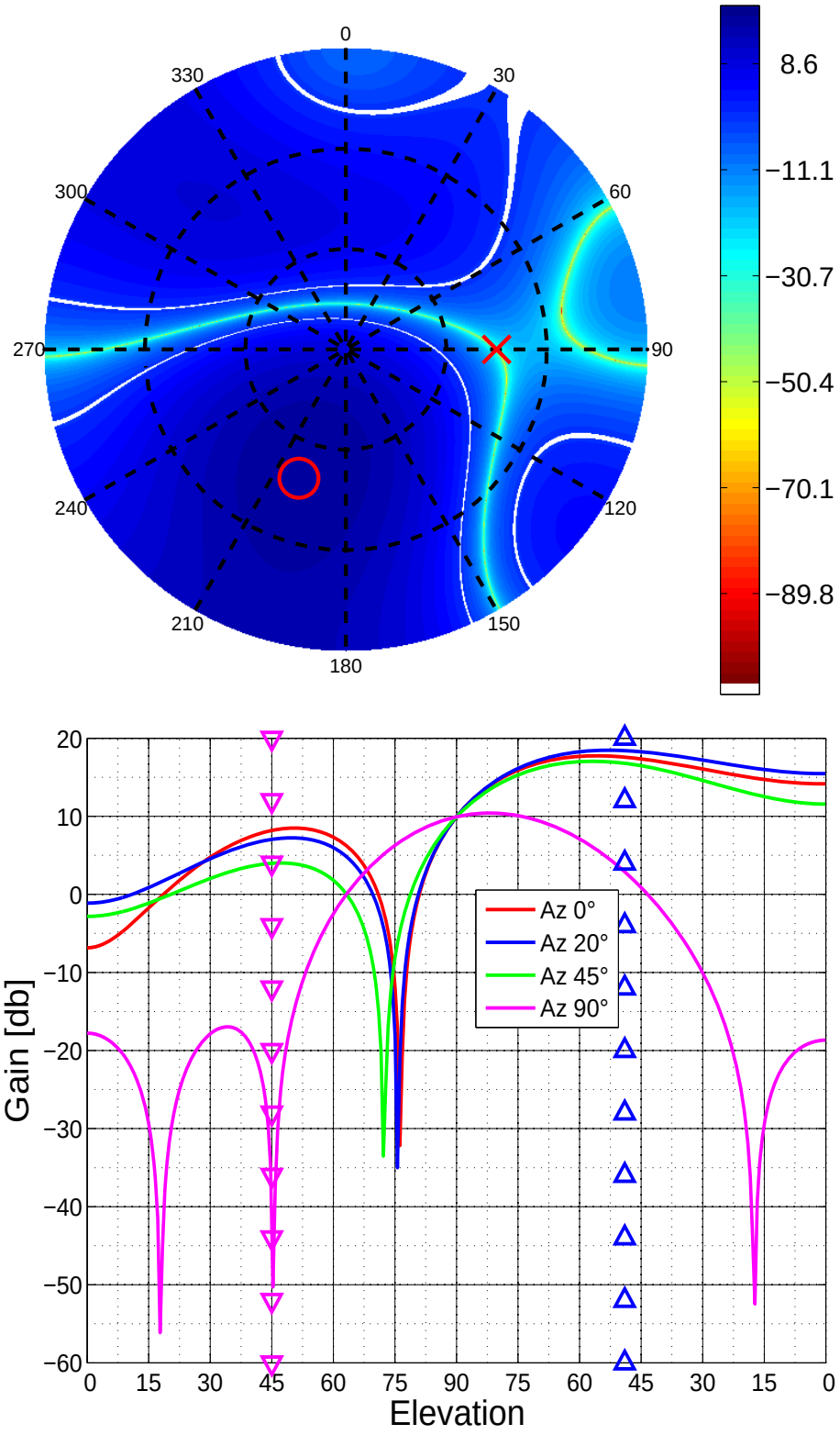


Figure 5.30: Antenna Pattern for Channel 10

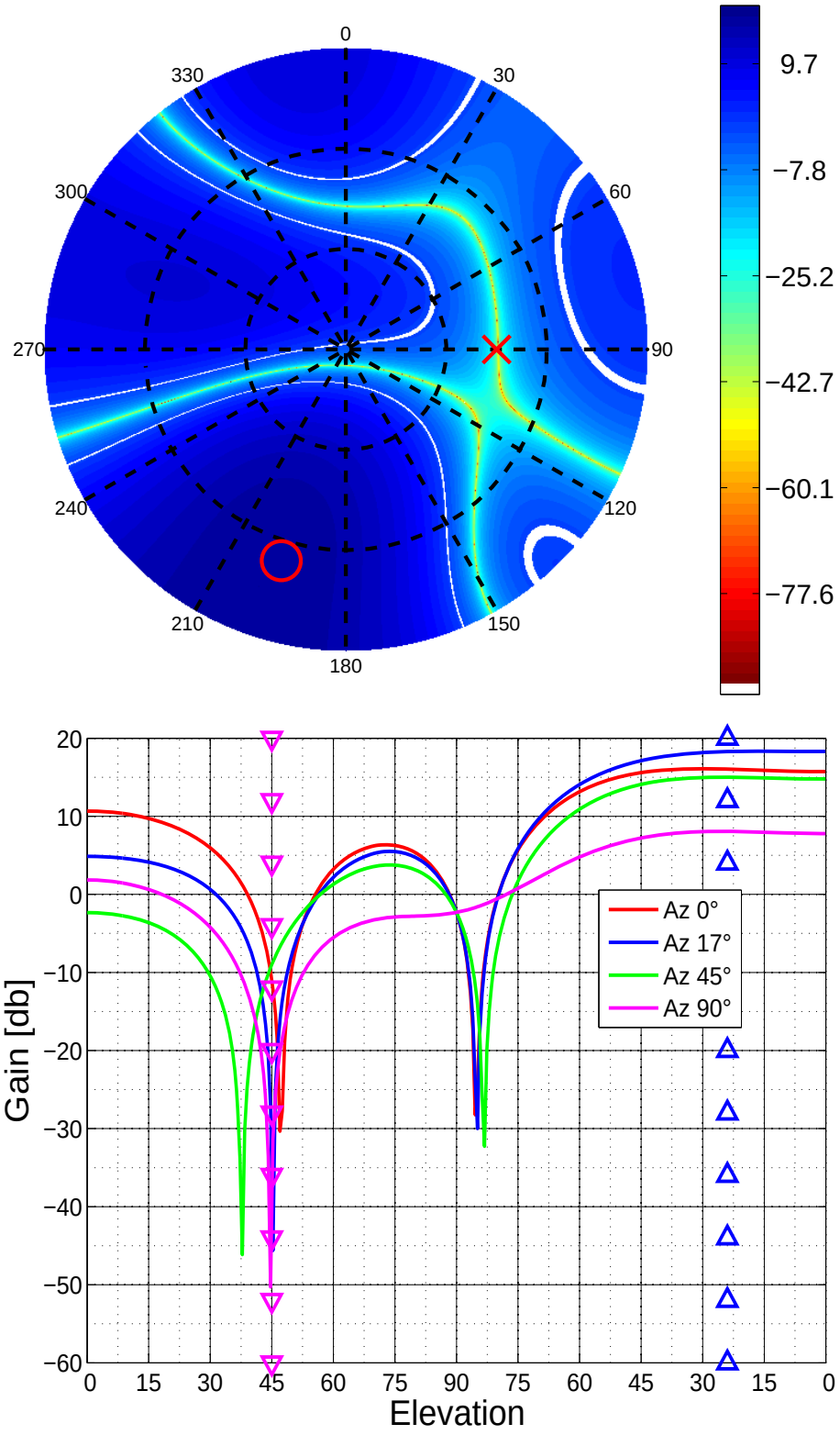


Figure 5.31: Antenna Pattern for Channel 11

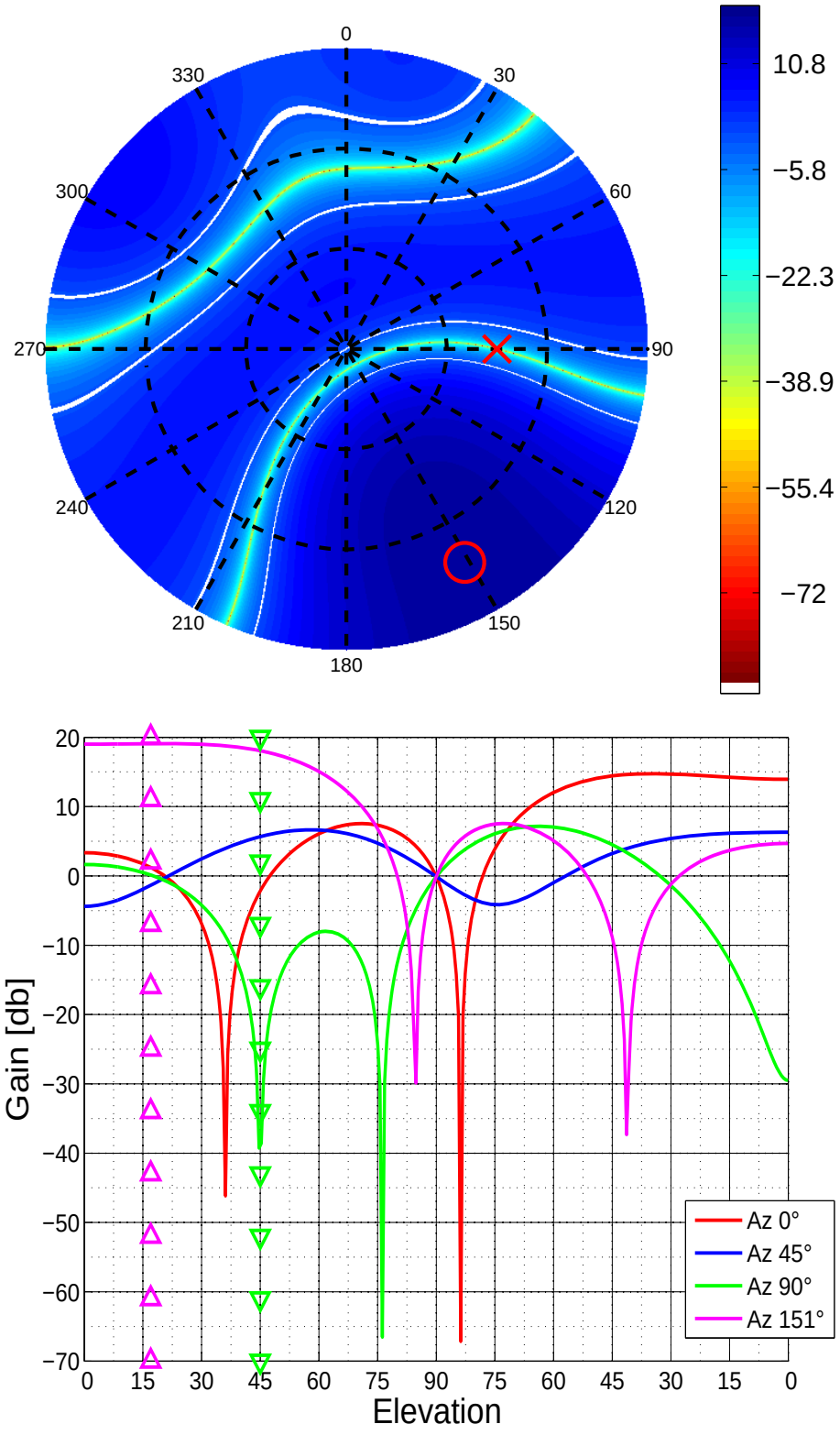


Figure 5.32: Antenna Pattern for Channel 14

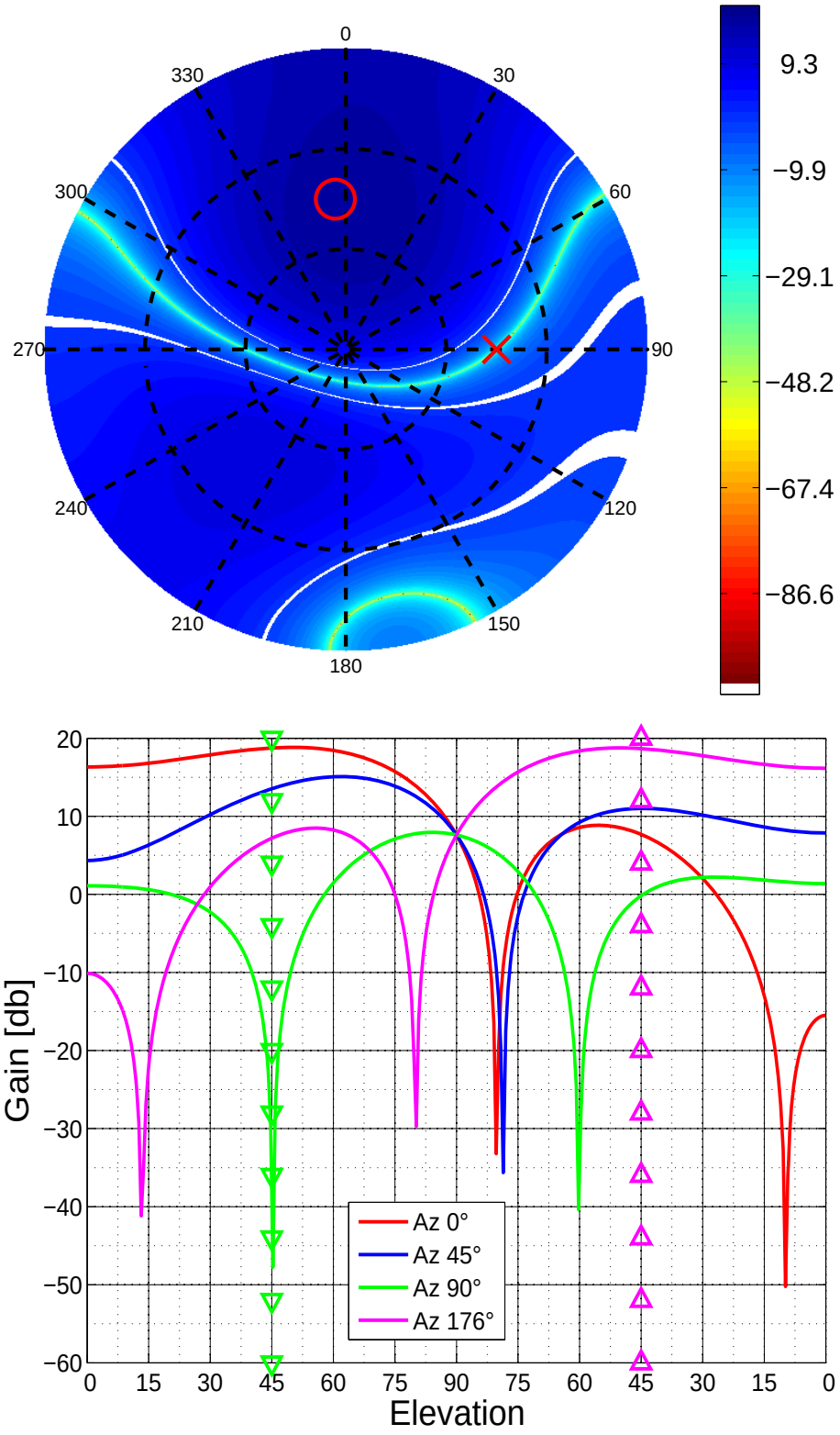


Figure 5.33: Antenna Pattern for Channel 16

Test C4 Nominal Mode + Interferer + 4 antenna in a square configuration : automatic Antenna Pattern beam steering in the direction of the tracked SVs, and single null steered in the direction of the interferer zeros.

Table 5.10: C4 Nominal mode test sequence and expected results example

N#	Command	Expected Response	Note
1	$PSPYN901,GET,SEL*00	$PSPYN901,ACK,SEL,OFF*0C	If the response doesn't match send $PSPYN901,SET,SEL, OFF and repeat step 1
2	$PSPYN901,SET,CXY,F,95,0,95 ,95,0,95*00	$PSPYN901,ACK,CXY,F,95,0,95,95,0,95*1D	
3	$PSPYN901,GET,CXY*00	$PSPYN901,ACK,CXY,F,95,0,95,95,0,95*1D	Get the current antenna configuration.
4	$PSPYN901,NVR,ATT,0,0*00	$PSPYN901,ACK,ATT,0,0*74	Set the current antenna array attitude to 0 heading 0 pitch.
5	$PSPYN901,GET,CAL*00	$PSPYN901,ACK,CAL,7F,0.0,1.00,0.0,1.00,0.0,1.00,0 .0,1.00,0.0,1.00,0.0,1.00*26	Get the current set of calibration parameters.
6	$PSPYN901,SET,INT,1,45,90*00	$PSPYN901,ACK,INT,1,45,90*73	
7	$PSPYN901,SET,SEL,NOM*00	$PSPYN901,ACK,SEL,NOM*0F	If the response doesn't match send $PSPYN901,SET,SEL, OFF and repeat step 1
8			Wait for the first fix. For each tracked satellite, take a note of the elevation azimuth and channel ID [CHID].
9	$PSPYN901,GET,RAW,1*00 $PSPYN901,GET,RAW,2*00 $PSPYN901,GET,RAW,3*00 $PSPYN901,GET,RAW,4*00 $PSPYN901,GET,RAW,5*00 $PSPYN901,GET,RAW,6*00 $PSPYN901,GET,RAW,7*00 $PSPYN901,GET,RAW,8*00 $PSPYN901,GET,RAW,9*00 $PSPYN901,GET,RAW,10*00 $PSPYN901,GET,RAW,11*00 $PSPYN901,GET,RAW,12*00 $PSPYN901,GET,RAW,13*00 $PSPYN901,GET,RAW,14*00 $PSPYN901,GET,RAW,15*00 $PSPYN901,GET,RAW,16*00	$PSPYN901,ACK,RAW,1,0,1,0,0,0,0,0,0,0,0*6D $PSPYN901,ACK,RAW,2,1,F,203,449,264,0,0,0,0,0*1 0 $PSPYN901,ACK,RAW,3,1,F,401,127,242,0,0,0,0,0*1 C $PSPYN901,ACK,RAW,4,1,F,315,55,251,0,0,0,0,0*2F $PSPYN901,ACK,RAW,5,0,1,0,0,0,0,0,0,0,0*69 $PSPYN901,ACK,RAW,6,1,F,498,236,249,0,0,0,0,0*1 1 $PSPYN901,ACK,RAW,7,1,F,159,414,218,0,0,0,0,0*1 A $PSPYN901,ACK,RAW,8,0,1,0,0,0,0,0,0,0,0*64 $PSPYN901,ACK,RAW,9,1,F,497,408,10,0,0,0,0,0*24 $PSPYN901,ACK,RAW,10,1,F,437,287,362,0,0,0,0,0* 21 $PSPYN901,ACK,RAW,11,1,F,439,218,291,0,0,0,0,0* 25 $PSPYN901,ACK,RAW,12,0,1,0,0,0,0,0,0,0,0*5F $PSPYN901,ACK,RAW,13,0,1,0,0,0,0,0,0,0,0*5E $PSPYN901,ACK,RAW,14,1,F,111,369,256,0,0,0,0,0* 23 $PSPYN901,ACK,RAW,15,0,1,0,0,0,0,0,0,0,0*58 $PSPYN901,ACK,RAW,16,1,F,457,165,219,0,0,0,0,0* 23	Check the offsets using the MATLAB script SPYN901_To_Antenna ArrayPatternPlot
10	$PSPYN901,SET,SEL,OFF*00	$PSPYN901,ACK,SEL,OFF*0C	Switch off the beamforming.
11	$PSPYN901,GET,RAW,1*00 $PSPYN901,GET,RAW,2*00 $PSPYN901,GET,RAW,3*00 $PSPYN901,GET,RAW,4*00 $PSPYN901,GET,RAW,5*00 $PSPYN901,GET,RAW,6*00 $PSPYN901,GET,RAW,7*00 $PSPYN901,GET,RAW,8*00 $PSPYN901,GET,RAW,9*00	$PSPYN901,ACK,RAW,1,0,1,0,0,0,0,0,0,0,0*6D $PSPYN901,ACK,RAW,2,0,1,0,0,0,0,0,0,0,0*6E $PSPYN901,ACK,RAW,3,0,1,0,0,0,0,0,0,0,0*6F $PSPYN901,ACK,RAW,4,0,1,0,0,0,0,0,0,0,0*68 $PSPYN901,ACK,RAW,5,0,1,0,0,0,0,0,0,0,0*69 $PSPYN901,ACK,RAW,6,0,1,0,0,0,0,0,0,0,0*6A $PSPYN901,ACK,RAW,7,0,1,0,0,0,0,0,0,0,0*6B $PSPYN901,ACK,RAW,8,0,1,0,0,0,0,0,0,0,0*64 $PSPYN901,ACK,RAW,9,0,1,0,0,0,0,0,0,0,0*65	Chech that each channel beamformer is set back to its default configuration.

	$PSPYN901,GET,RAW,10*00 $PSPYN901,GET,RAW,11*00 $PSPYN901,GET,RAW,12*00 $PSPYN901,GET,RAW,13*00 $PSPYN901,GET,RAW,14*00 $PSPYN901,GET,RAW,15*00 $PSPYN901,GET,RAW,16*00	$PSPYN901,ACK,RAW,10,0,1,0,0,0,0,0,0,0,0*5D $PSPYN901,ACK,RAW,11,0,1,0,0,0,0,0,0,0,0*5C $PSPYN901,ACK,RAW,12,0,1,0,0,0,0,0,0,0,0*5F $PSPYN901,ACK,RAW,13,0,1,0,0,0,0,0,0,0,0*5E $PSPYN901,ACK,RAW,14,0,1,0,0,0,0,0,0,0,0*59 $PSPYN901,ACK,RAW,15,0,1,0,0,0,0,0,0,0,0*58 $PSPYN901,ACK,RAW,16,0,1,0,0,0,0,0,0,0,0*5B	

Figure 5.34 to figure 5.44 report the results for test C4.

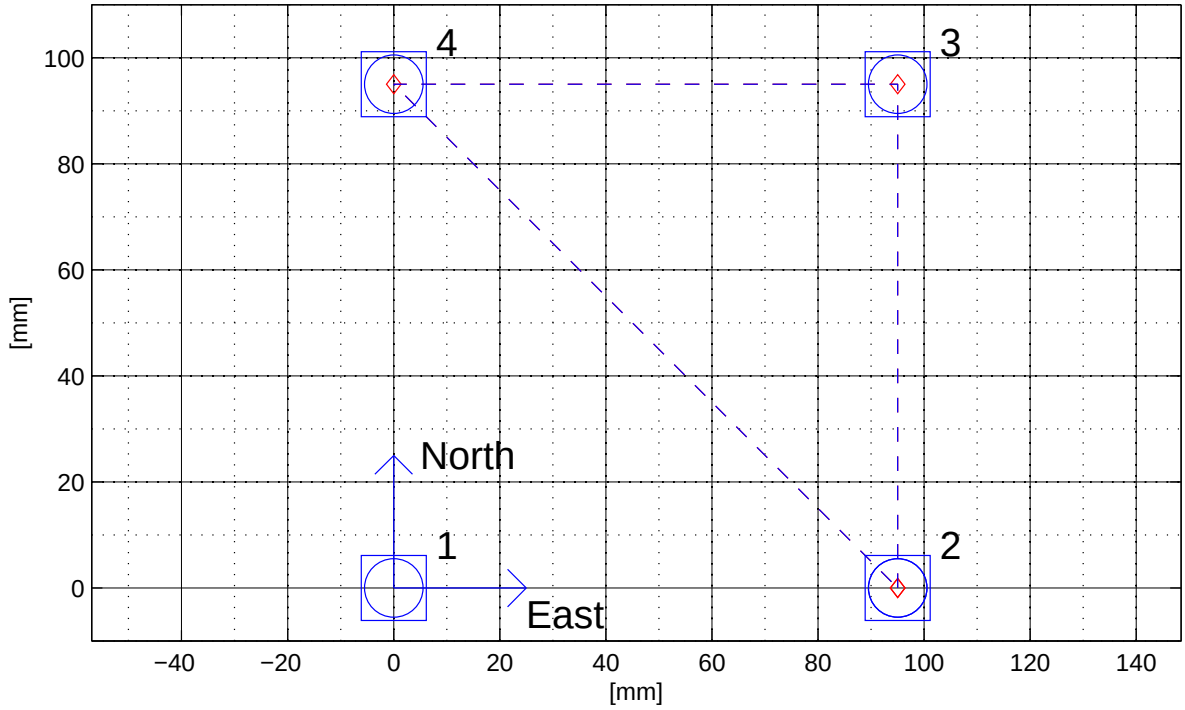


Figure 5.34: Antenna Array Map

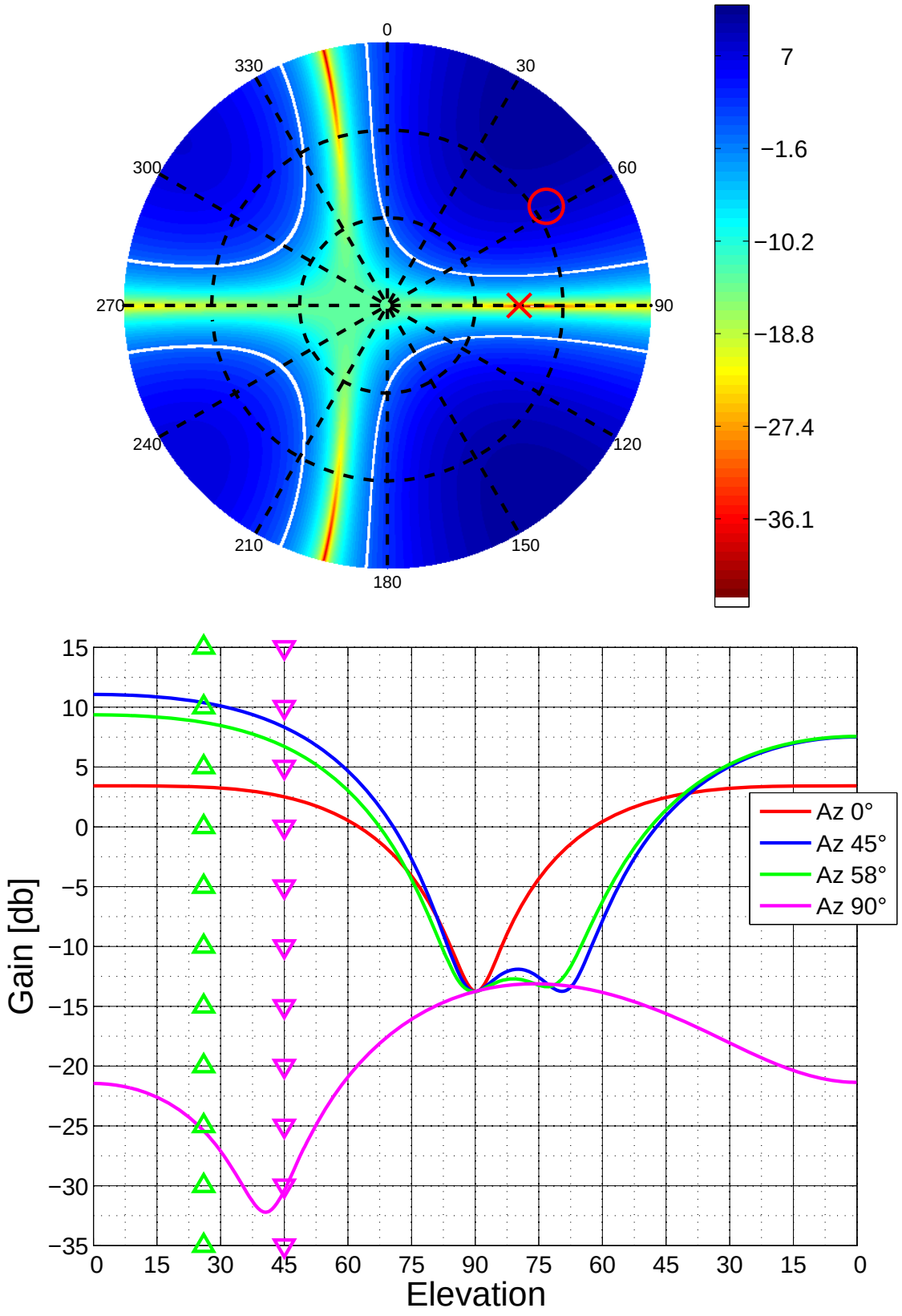


Figure 5.35: Antenna Pattern for Channel 2

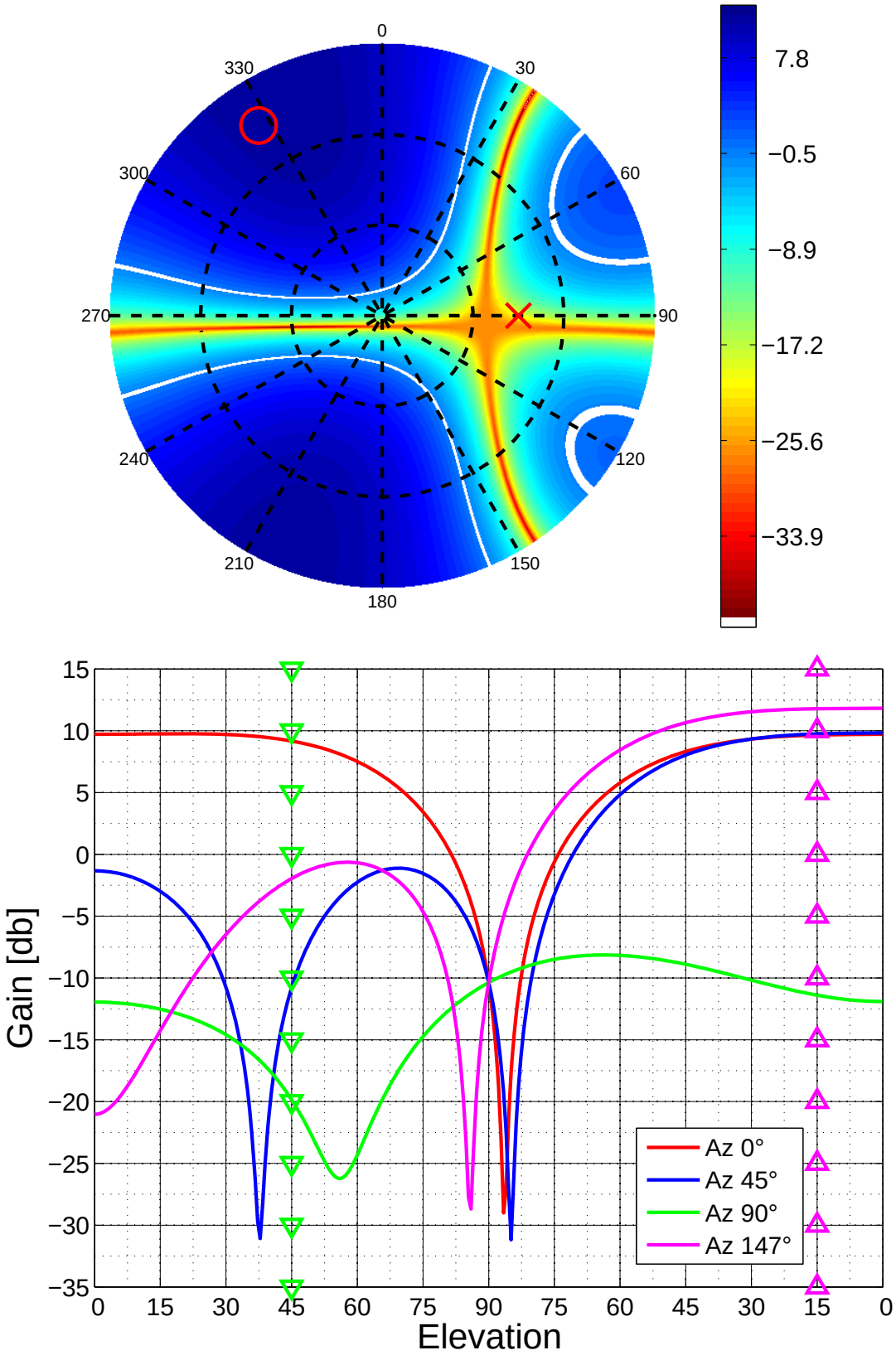


Figure 5.36: Antenna Pattern for Channel 3

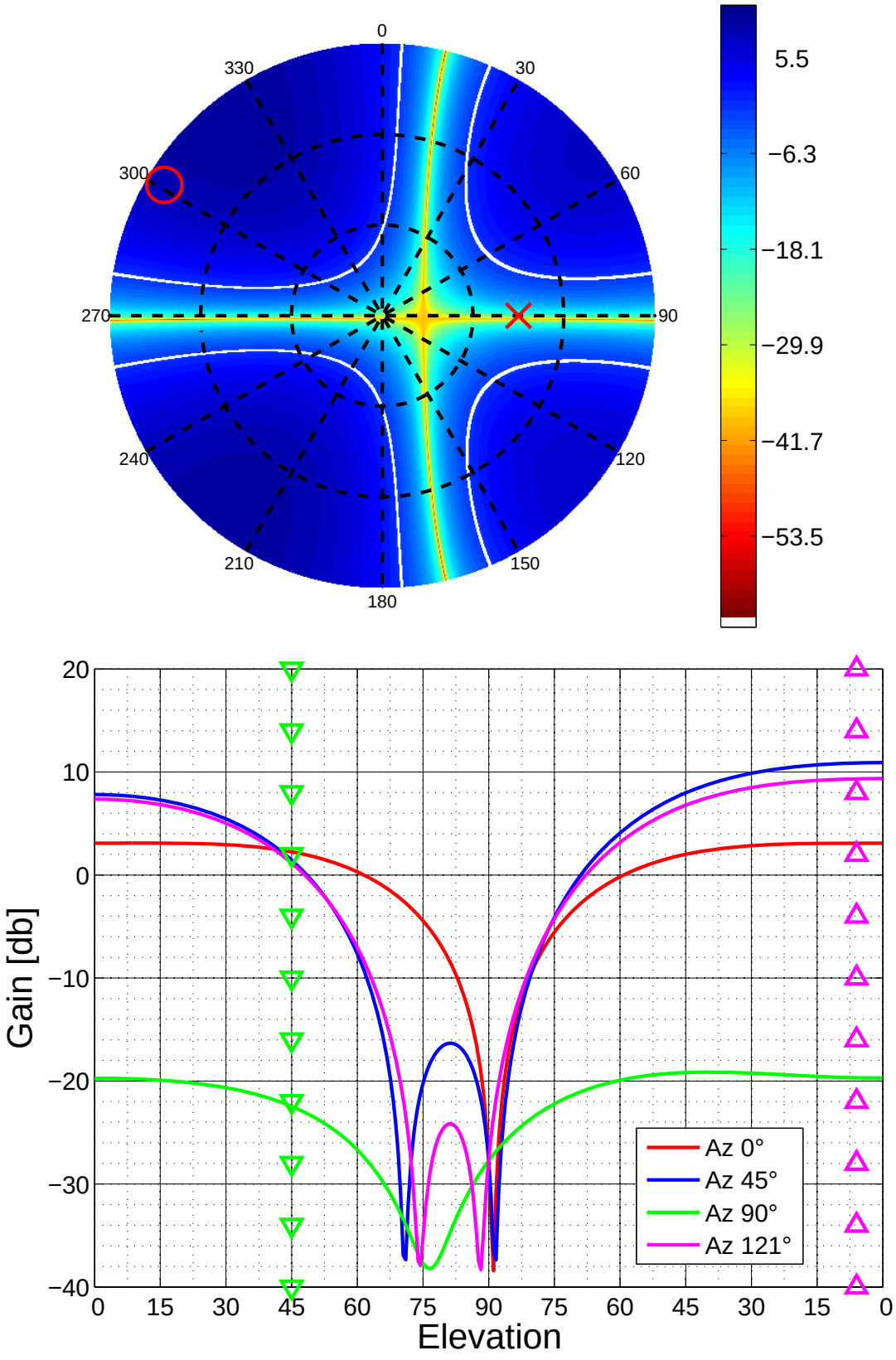


Figure 5.37: Antenna Pattern for Channel 4

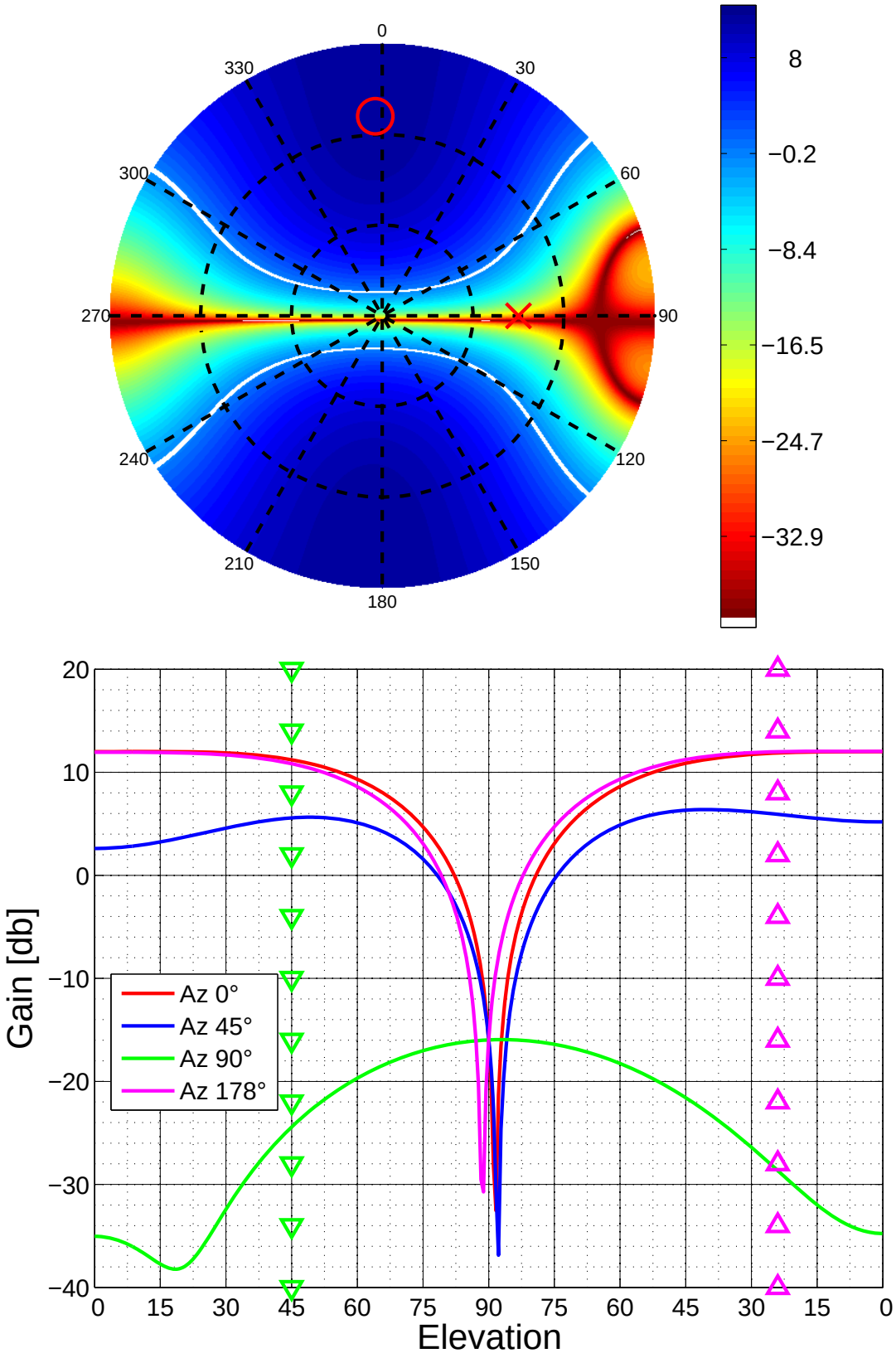


Figure 5.38: Antenna Pattern for Channel 6

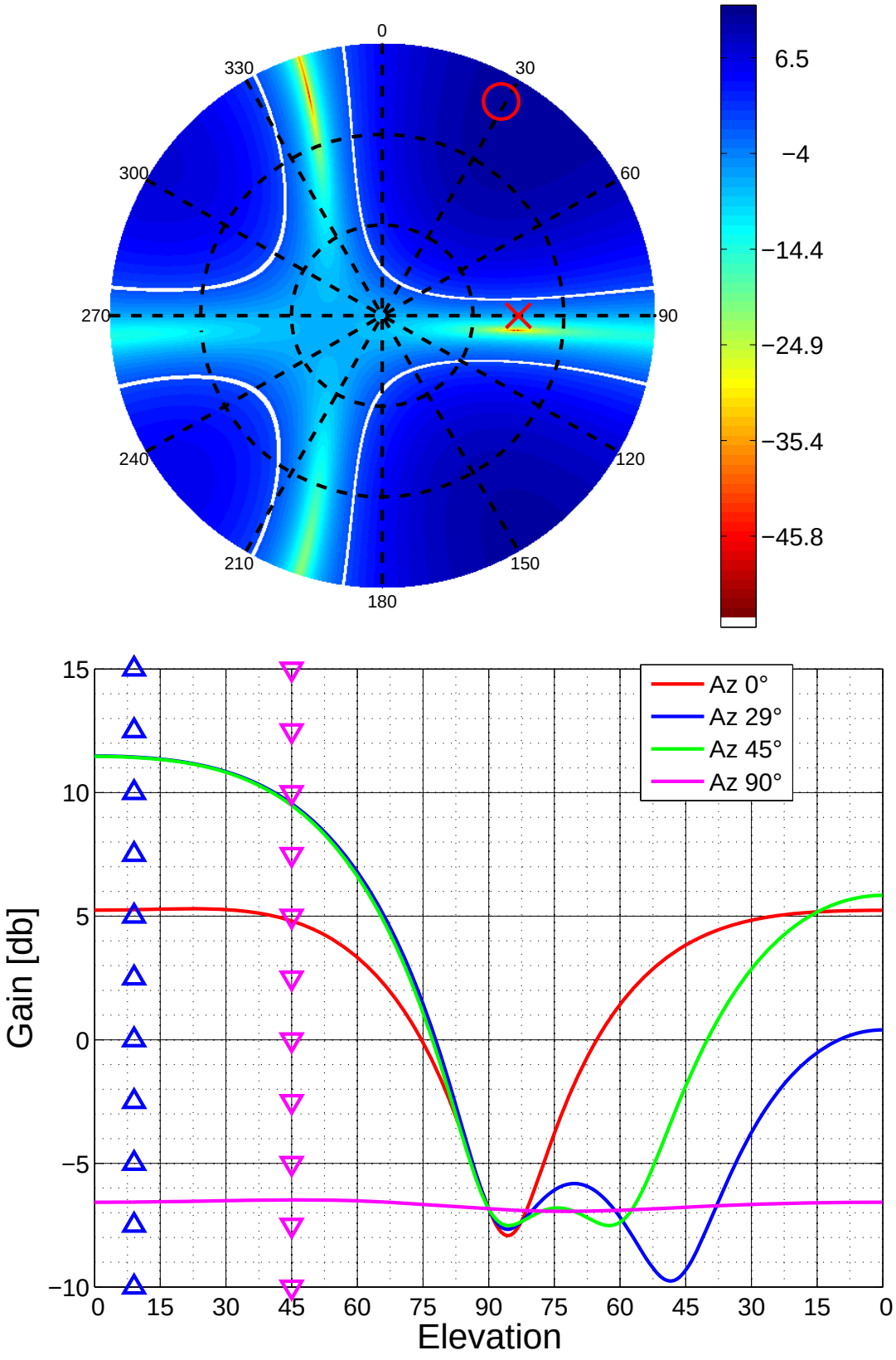


Figure 5.39: Antenna Pattern for Channel 7

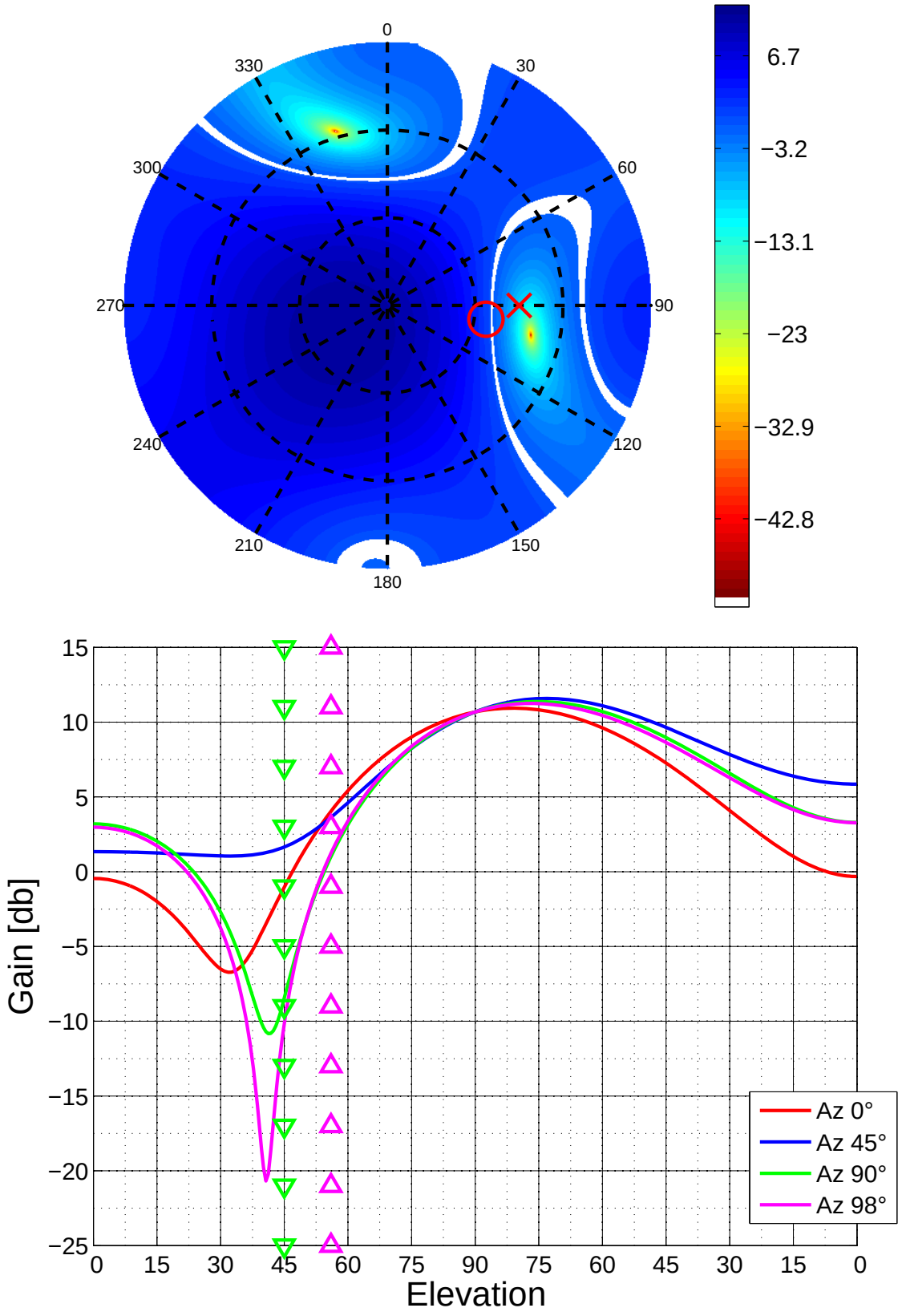


Figure 5.40: Antenna Pattern for Channel 9

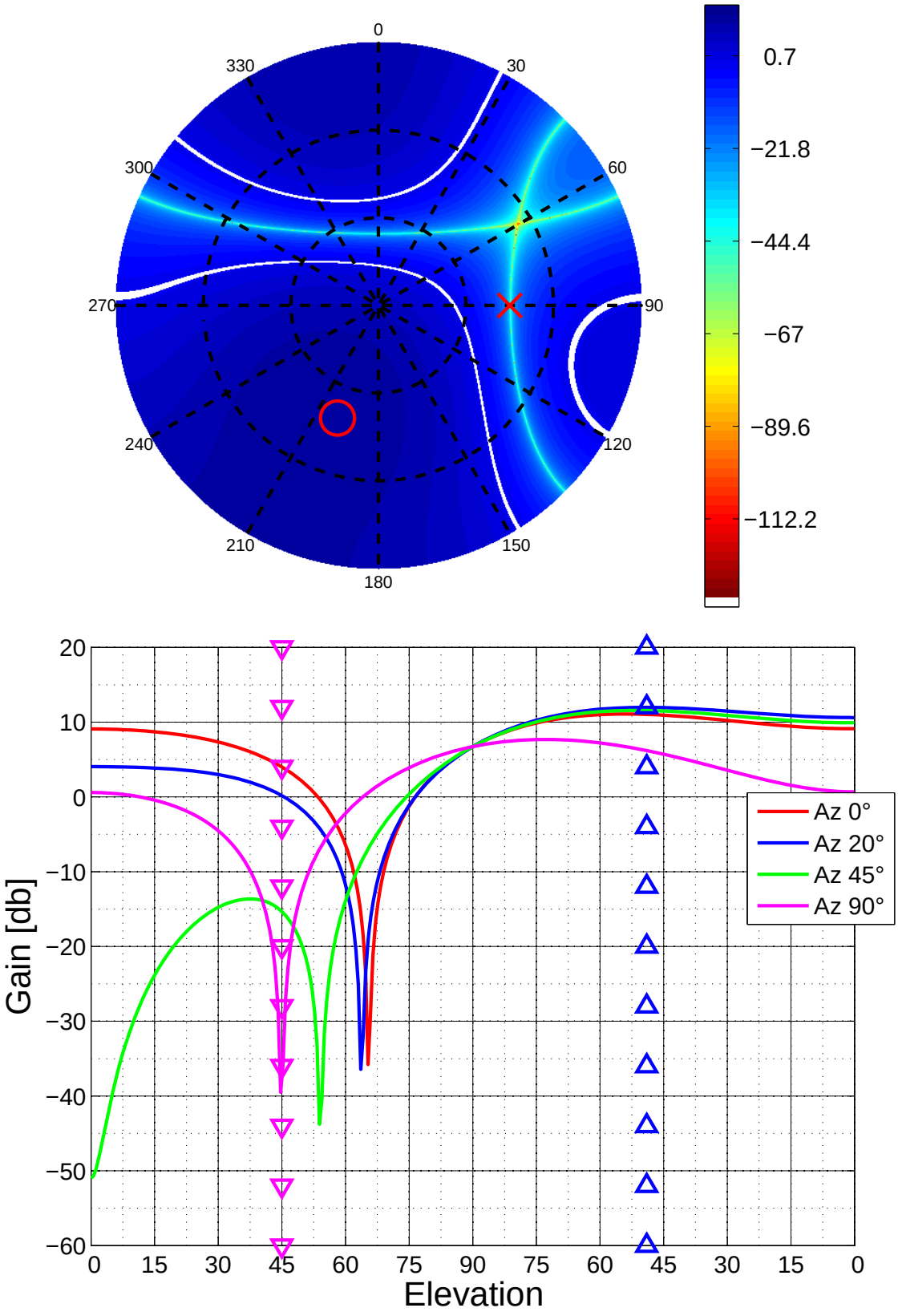


Figure 5.41: Antenna Pattern for Channel 10

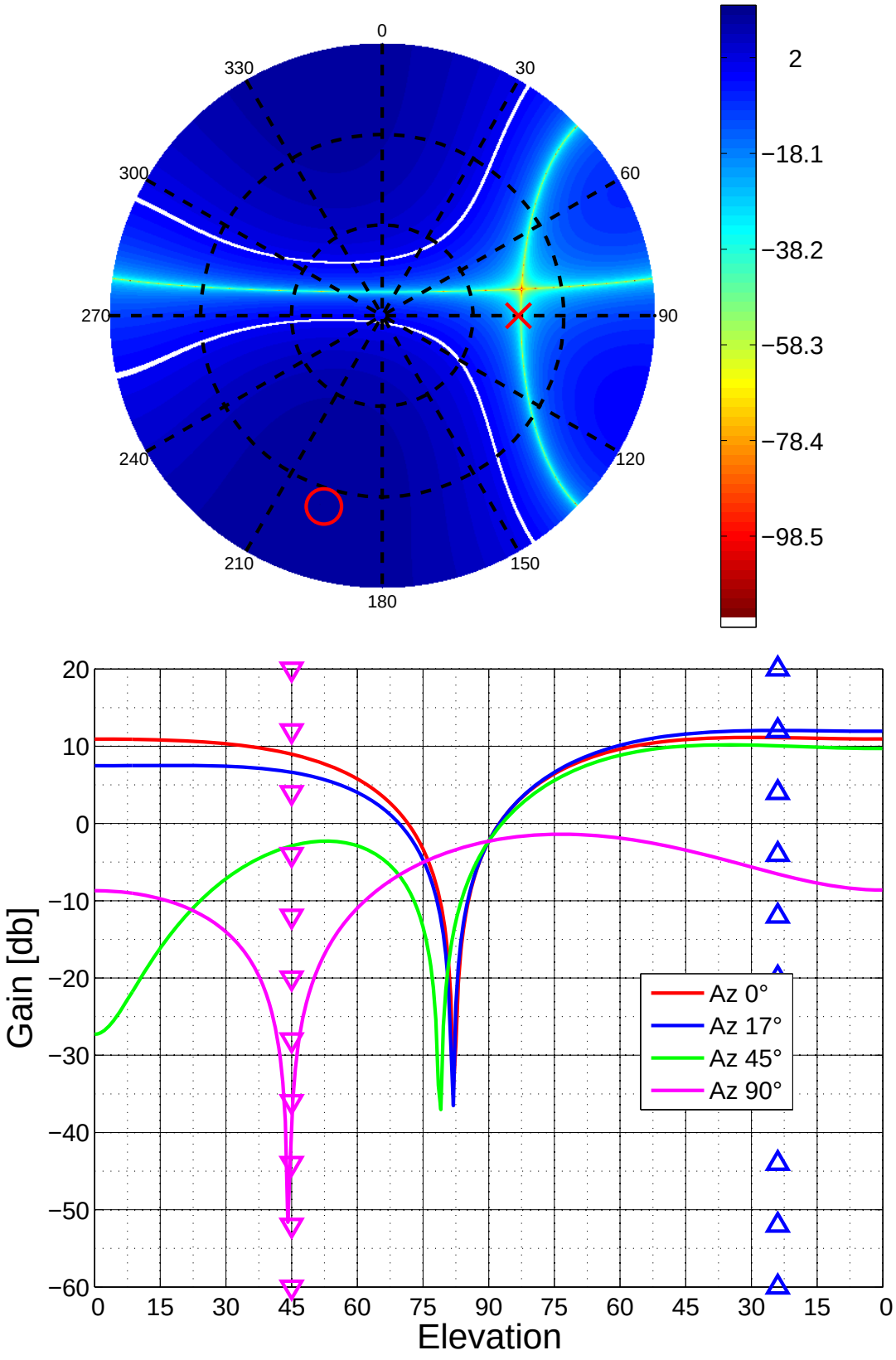


Figure 5.42: Antenna Pattern for Channel 11

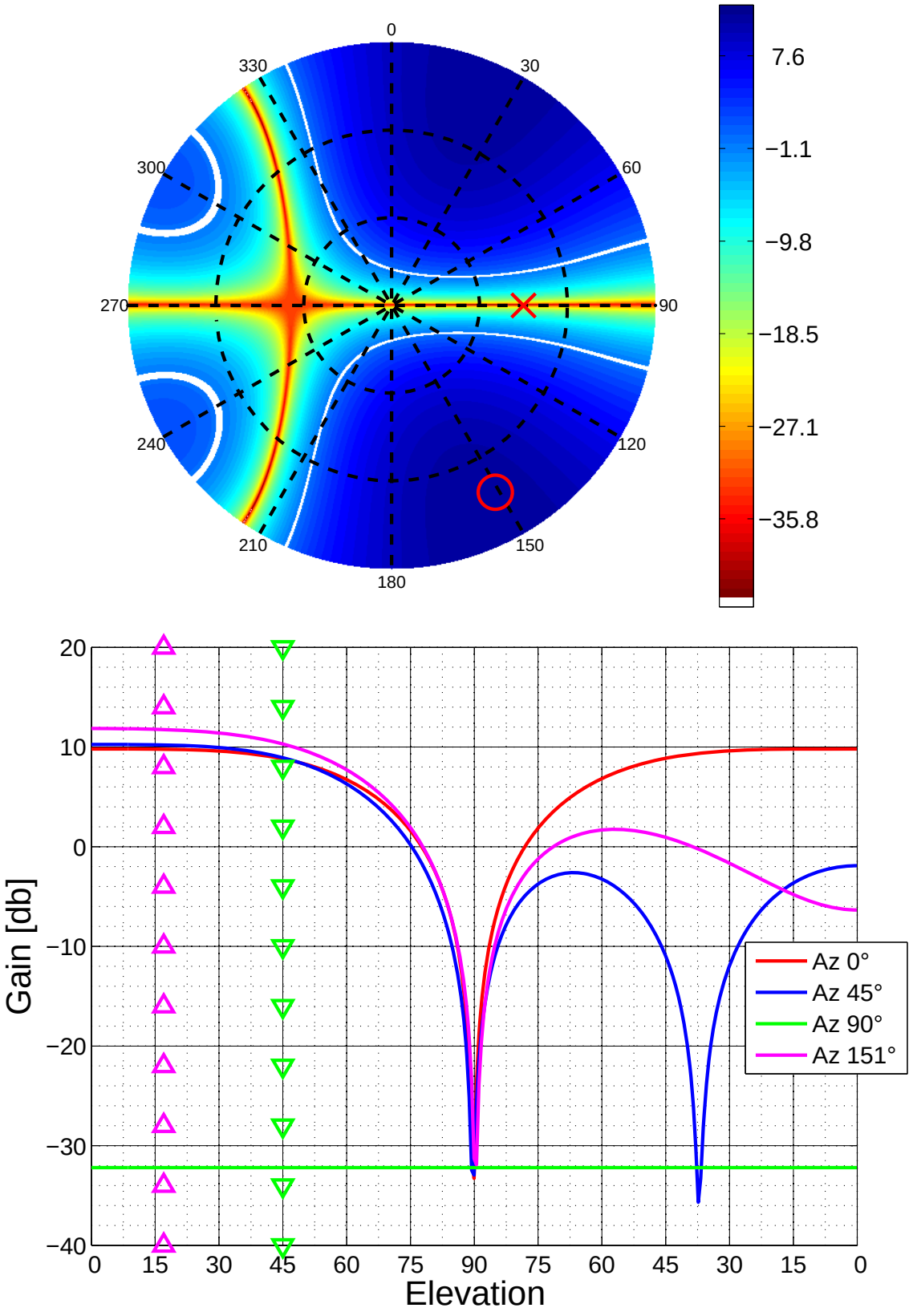


Figure 5.43: Antenna Pattern for Channel 14

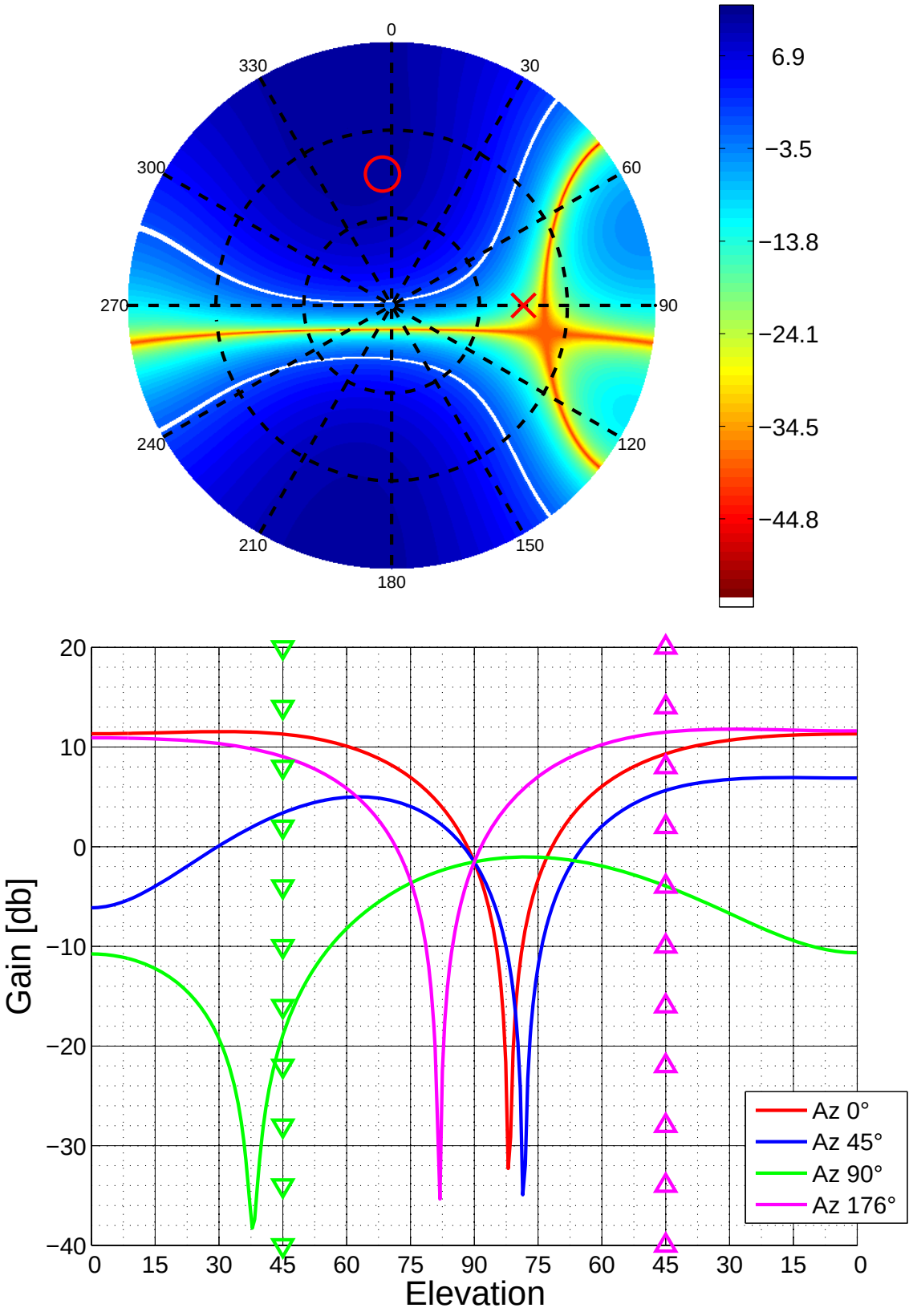


Figure 5.44: Antenna Pattern for Channel 16

Test C5 Nominal Mode + Interferer + 5 antenna in a circular configuration : automatic Antenna Pattern beam steering in the direction of the tracked SVs, and single null steered in the direction of the interferer zeros, alternate RF inputs are used.

Table 5.11: C5 Nominal mode test sequence and expected results example

N#	Command	Expected Response	Note
1	$PSPYN901,GET,SEL*00	$PSPYN901,ACK,SEL,OFF*0C	If the response doesn't match send $PSPYN901,SET,SEL,OFF and repeat step 1
2	$PSPYN901,SET,CPO,155,95,0,95,90,95,180,95,270*00	$PSPYN901,ACK,CPO,155,95,0,95,90,95,180,95,270*41	
3	$PSPYN901,GET,CXY*00	$PSPYN901,ACK,CXY,155,95,0,0,95,-95,0,0,-95*6A	Get the current antenna configuration.
4	$PSPYN901,NVR,ATT,45,0*00	$PSPYN901,ACK,ATT,45,0*45	Set the current antenna array attitude to 0 heading 0 pitch.
5	$PSPYN901,SET,CAL,155,0,0,0,0*00	$PSPYN901,ACK,CAL,155,0.0,1.00,0.0,1.00,0.0,1.00,0.0,1.00*00	Set the current calibration parameters
6	$PSPYN901,GET,CAL*00	$PSPYN901,ACK,CAL,155,0.0,1.00,0.0,1.00,0.0,1.00,0.0,1.00*00	Get the current set of calibration parameters.
7	$PSPYN901,SET,INT,1,45,90*00	$PSPYN901,ACK,INT,1,45,90*73	
8	$PSPYN901,SET,SEL,NOM*00	$PSPYN901,ACK,SEL,NOM*0F	If the response doesn't match send $PSPYN901,SET,SEL,OFF and repeat step 1
9			Wait for the first fix. For each tracked satellite, take a note of the elevation azimuth and channel ID [CHID].
10	$PSPYN901,GET,RAW,1*00 $PSPYN901,GET,RAW,2*00 $PSPYN901,GET,RAW,3*00 $PSPYN901,GET,RAW,4*00 $PSPYN901,GET,RAW,5*00 $PSPYN901,GET,RAW,6*00 $PSPYN901,GET,RAW,7*00 $PSPYN901,GET,RAW,8*00 $PSPYN901,GET,RAW,9*00 $PSPYN901,GET,RAW,10*00 $PSPYN901,GET,RAW,11*00 $PSPYN901,GET,RAW,12*00 $PSPYN901,GET,RAW,13*00 $PSPYN901,GET,RAW,14*00 $PSPYN901,GET,RAW,15*00 $PSPYN901,GET,RAW,16*00	$PSPYN901,ACK,RAW,1,0,1,0,0,0,0,0,0,0,0*6D $PSPYN901,ACK,RAW,2,1,155,0,499,0,284,0,13,0,228*5F $PSPYN901,ACK,RAW,3,1,155,0,308,0,7,0,204,0,505*64 $PSPYN901,ACK,RAW,4,1,155,0,251,0,464,0,261,0,48*50 $PSPYN901,ACK,RAW,5,0,1,0,0,0,0,0,0,0,0*69 $PSPYN901,ACK,RAW,6,1,155,0,363,0,216,0,149,0,296*69 $PSPYN901,ACK,RAW,7,1,155,0,446,0,238,0,66,0,274*54 $PSPYN901,ACK,RAW,8,0,1,0,0,0,0,0,0,0,0*64 $PSPYN901,ACK,RAW,9,1,155,0,100,0,120,0,95,0,38*61 $PSPYN901,ACK,RAW,10,1,155,0,41,0,370,0,471,0,142*68 $PSPYN901,ACK,RAW,11,1,155,0,30,0,340,0,482,0,172*63 $PSPYN901,ACK,RAW,12,0,1,0,0,0,0,0,0,0,0*5F $PSPYN901,ACK,RAW,13,0,1,0,0,0,0,0,0,0,0*5E $PSPYN901,ACK,RAW,14,1,155,0,233,0,440,0,279,0,72*63 $PSPYN901,ACK,RAW,15,0,1,0,0,0,0,0,0,0,0*58 $PSPYN901,ACK,RAW,16,1,155,0,382,0,43,0,130,0,469*6D	Check the offsets using the MATLAB script SPYN901_To_AntennaArrayPatternPlot
11	$PSPYN901,SET,SEL,OFF*00	$PSPYN901,ACK,SEL,OFF*0C	Switch off the beamforming.
12	$PSPYN901,GET,RAW,1*00 $PSPYN901,GET,RAW,2*00 $PSPYN901,GET,RAW,3*00 $PSPYN901,GET,RAW,4*00 $PSPYN901,GET,RAW,5*00 $PSPYN901,GET,RAW,6*00 $PSPYN901,GET,RAW,7*00	$PSPYN901,ACK,RAW,1,0,1,0,0,0,0,0,0,0,0*6D $PSPYN901,ACK,RAW,2,0,1,0,0,0,0,0,0,0,0*6E $PSPYN901,ACK,RAW,3,0,1,0,0,0,0,0,0,0,0*6F $PSPYN901,ACK,RAW,4,0,1,0,0,0,0,0,0,0,0*68 $PSPYN901,ACK,RAW,5,0,1,0,0,0,0,0,0,0,0*69 $PSPYN901,ACK,RAW,6,0,1,0,0,0,0,0,0,0,0*6A $PSPYN901,ACK,RAW,7,0,1,0,0,0,0,0,0,0,0*6B	Chech that each channel beamformer is set back to its default configuration.

	$PSPYN901,GET,RAW,8*00	$PSPYN901,ACK,RAW,8,0,1,0,0,0,0,0,0,0*64	
	$PSPYN901,GET,RAW,9*00	$PSPYN901,ACK,RAW,9,0,1,0,0,0,0,0,0,0*65	
	$PSPYN901,GET,RAW,10*00	$PSPYN901,ACK,RAW,10,0,1,0,0,0,0,0,0,0*5D	
	$PSPYN901,GET,RAW,11*00	$PSPYN901,ACK,RAW,11,0,1,0,0,0,0,0,0,0*5C	
	$PSPYN901,GET,RAW,12*00	$PSPYN901,ACK,RAW,12,0,1,0,0,0,0,0,0,0*5F	
	$PSPYN901,GET,RAW,13*00	$PSPYN901,ACK,RAW,13,0,1,0,0,0,0,0,0,0*5E	
	$PSPYN901,GET,RAW,14*00	$PSPYN901,ACK,RAW,14,0,1,0,0,0,0,0,0,0*59	
	$PSPYN901,GET,RAW,15*00	$PSPYN901,ACK,RAW,15,0,1,0,0,0,0,0,0,0*58	
	$PSPYN901,GET,RAW,16*00	$PSPYN901,ACK,RAW,16,0,1,0,0,0,0,0,0,0*5B	

Figure 5.45 to figure 5.55 report the results for test C5.

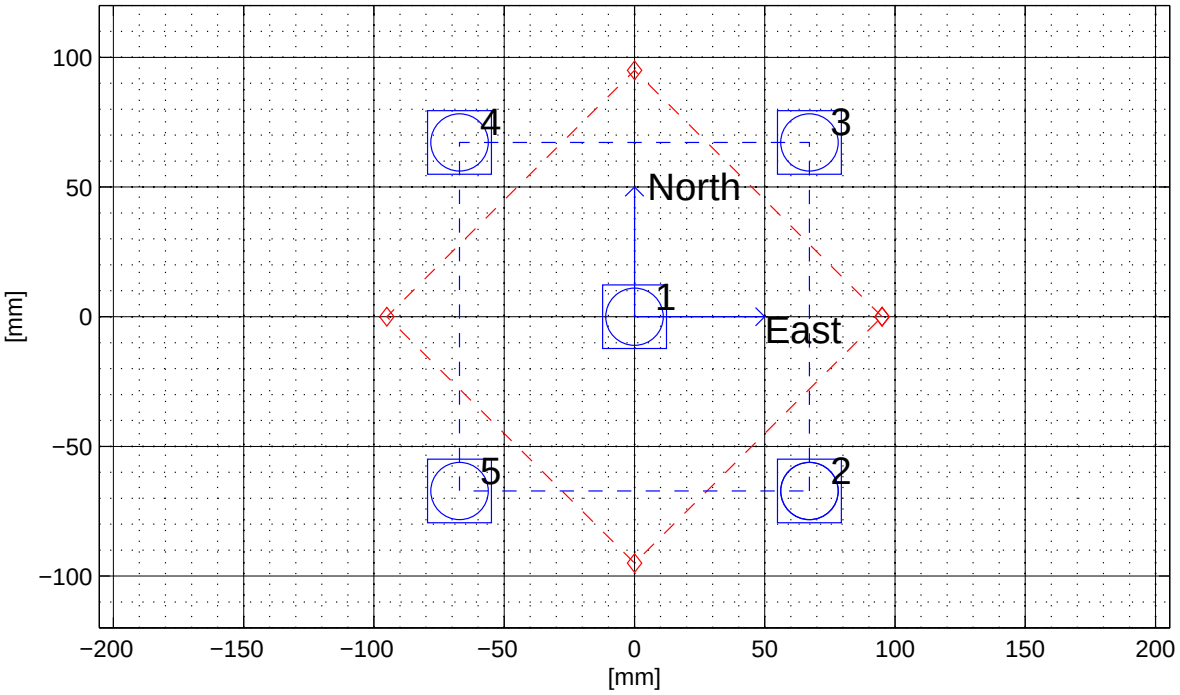


Figure 5.45: Antenna Array Map

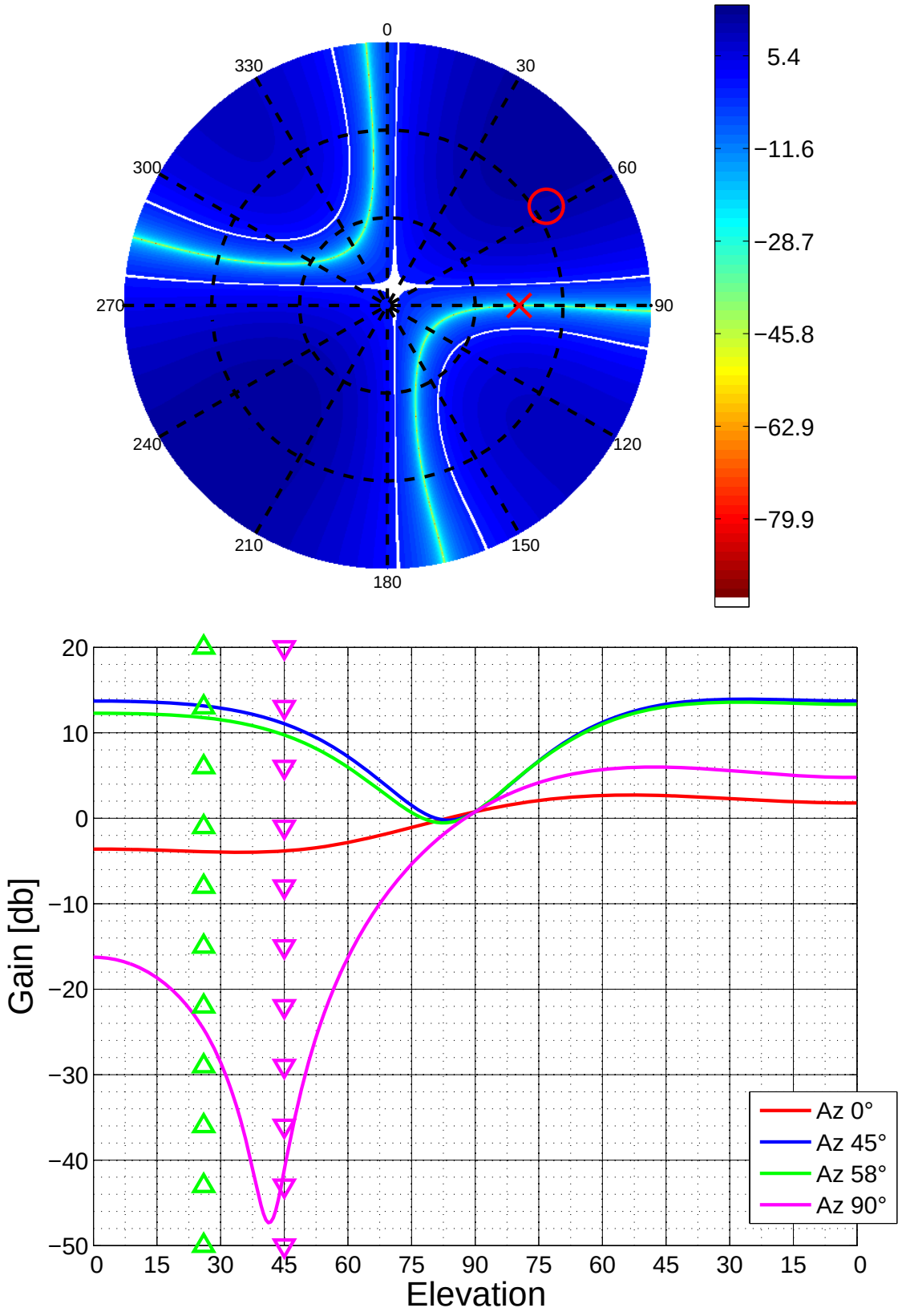


Figure 5.46: Antenna Pattern for Channel 2

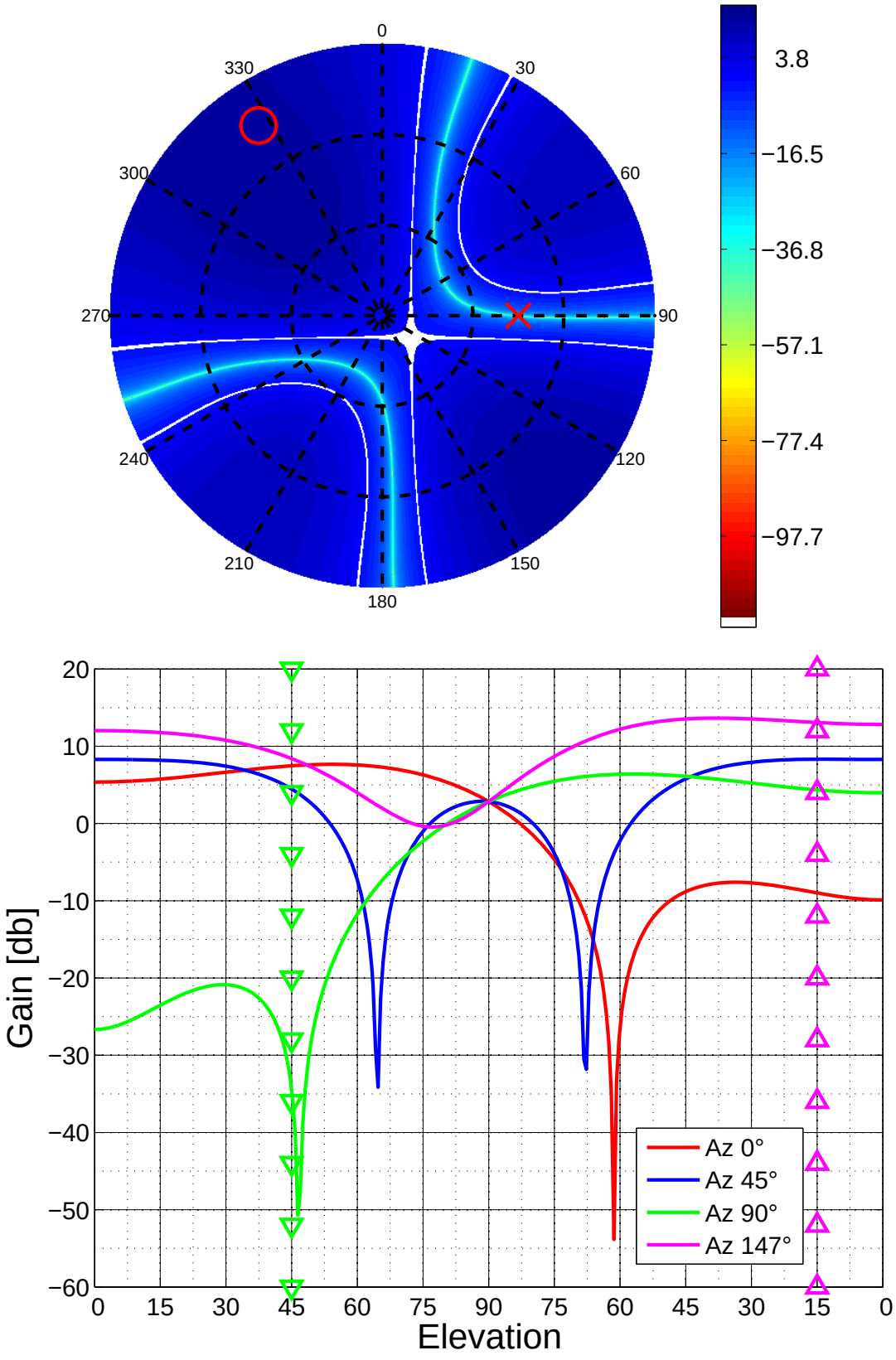


Figure 5.47: Antenna Pattern for Channel 3

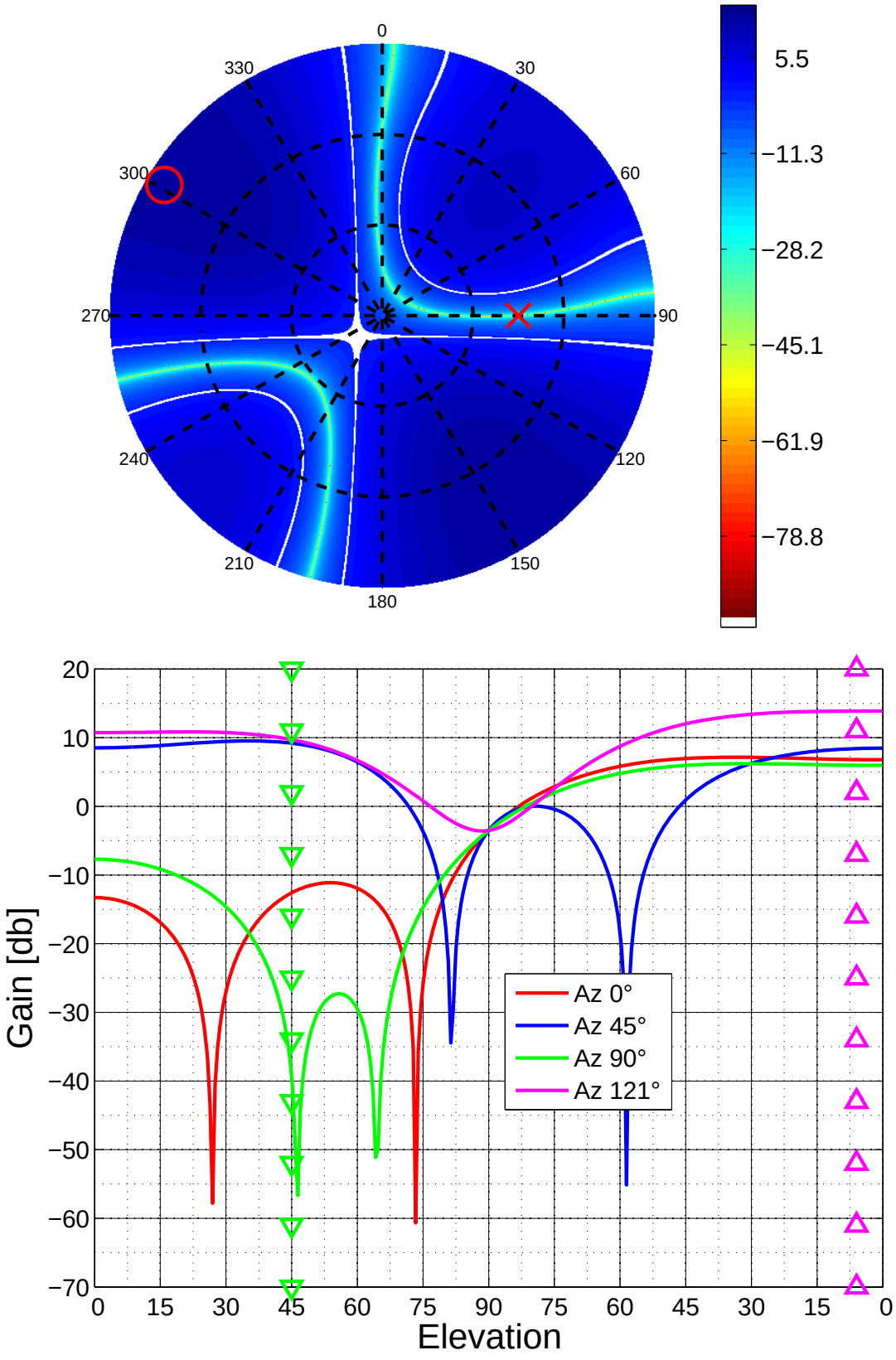


Figure 5.48: Antenna Pattern for Channel 4

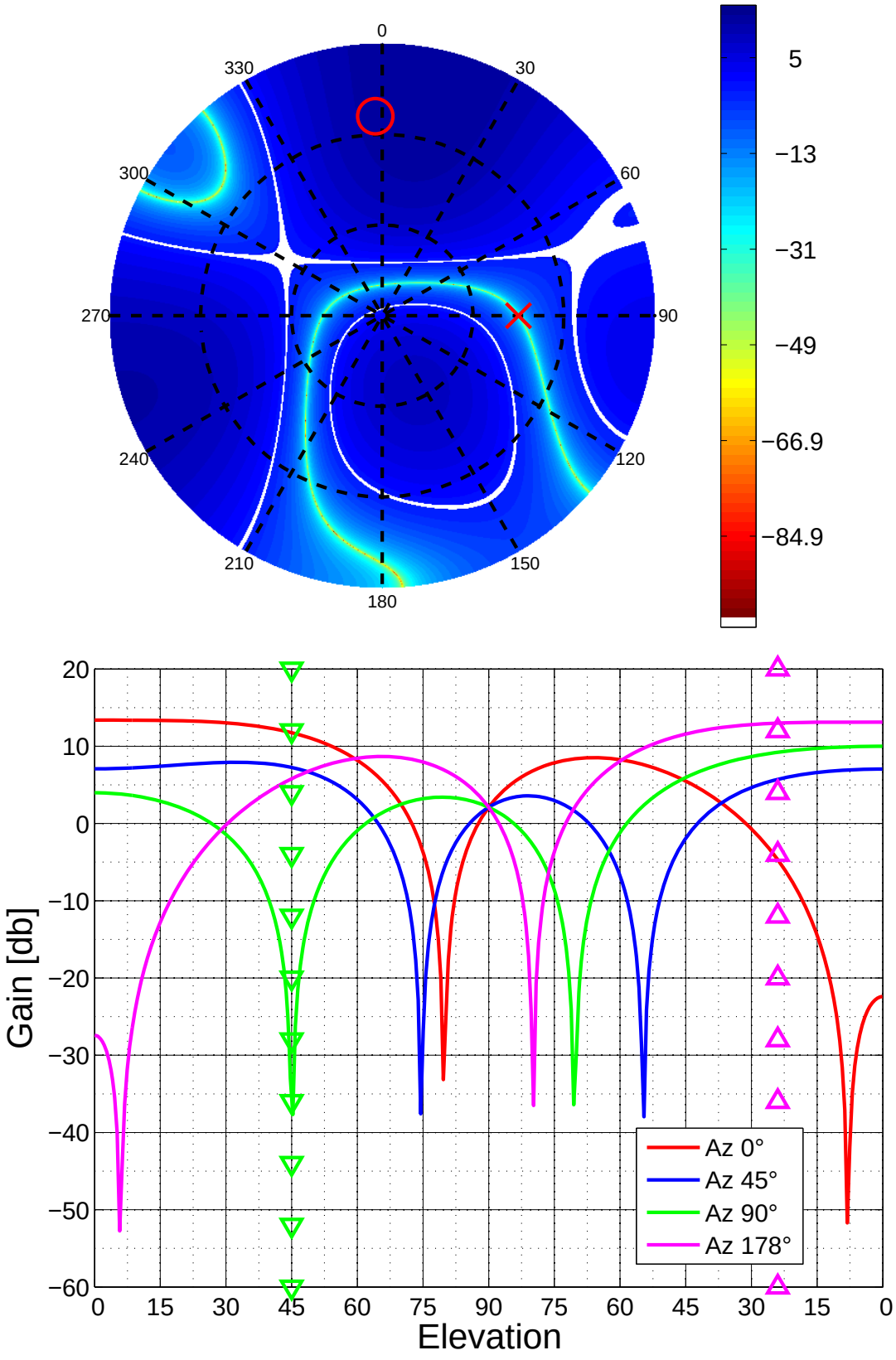


Figure 5.49: Antenna Pattern for Channel 6

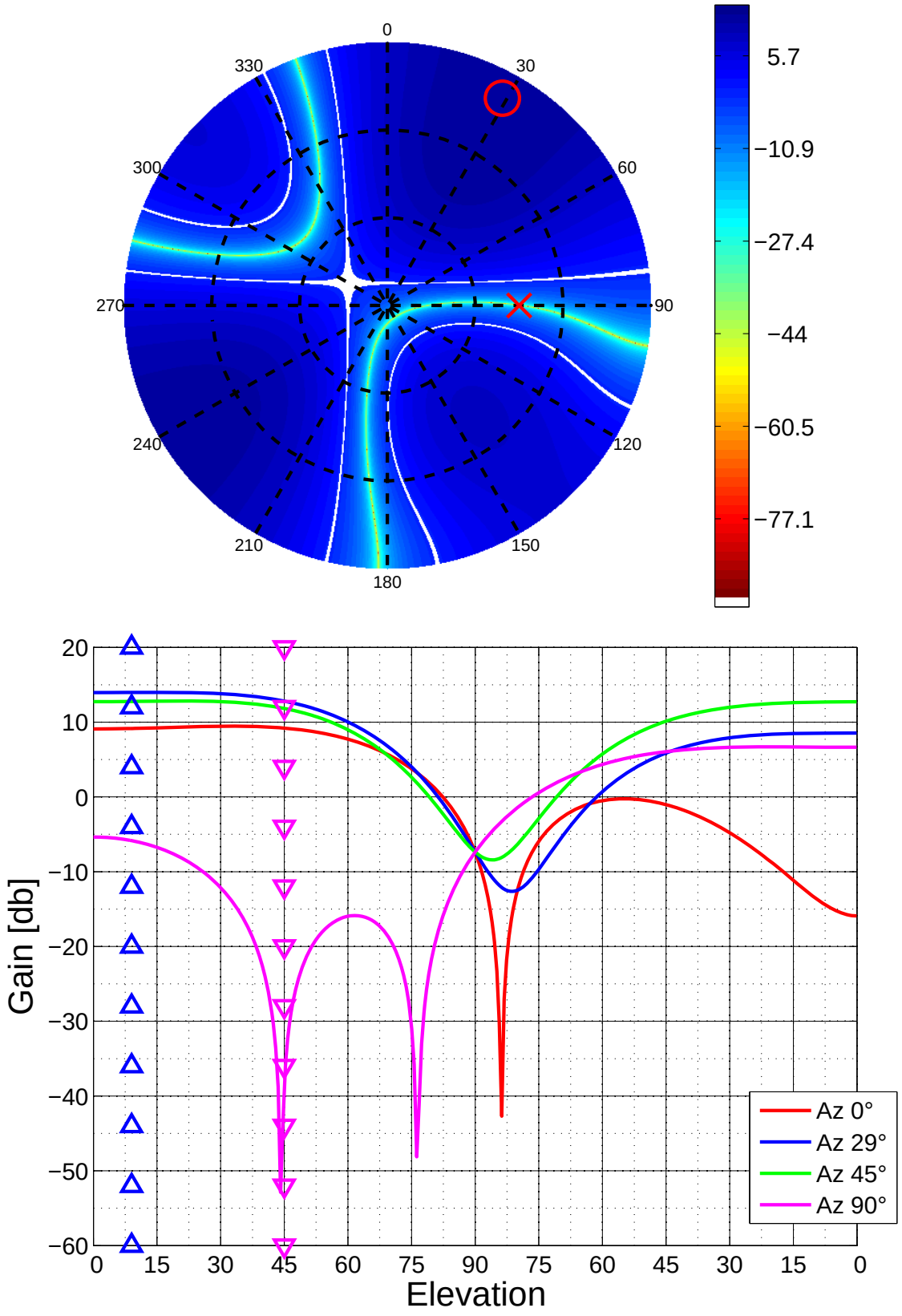


Figure 5.50: Antenna Pattern for Channel 7

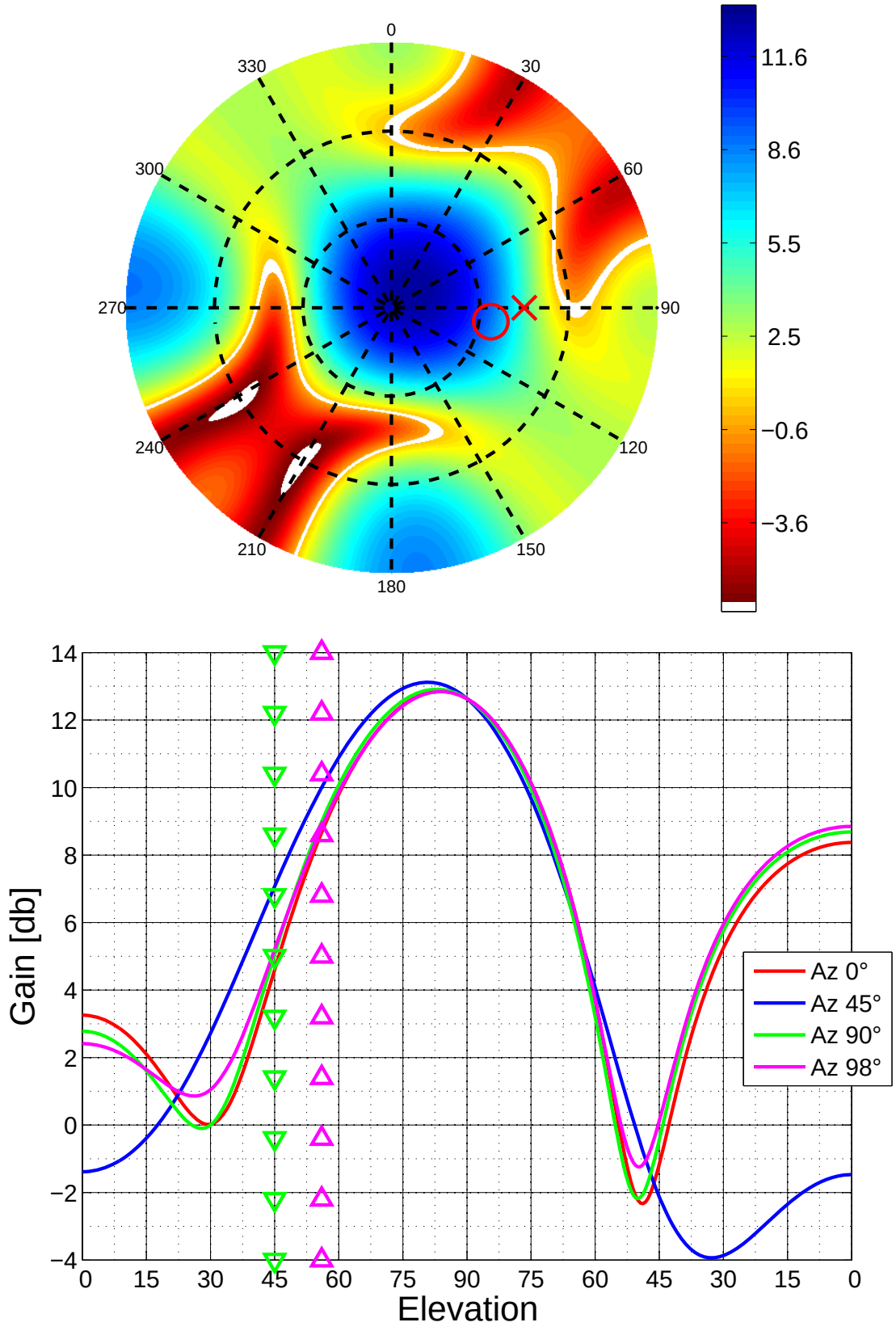


Figure 5.51: Antenna Pattern for Channel 9

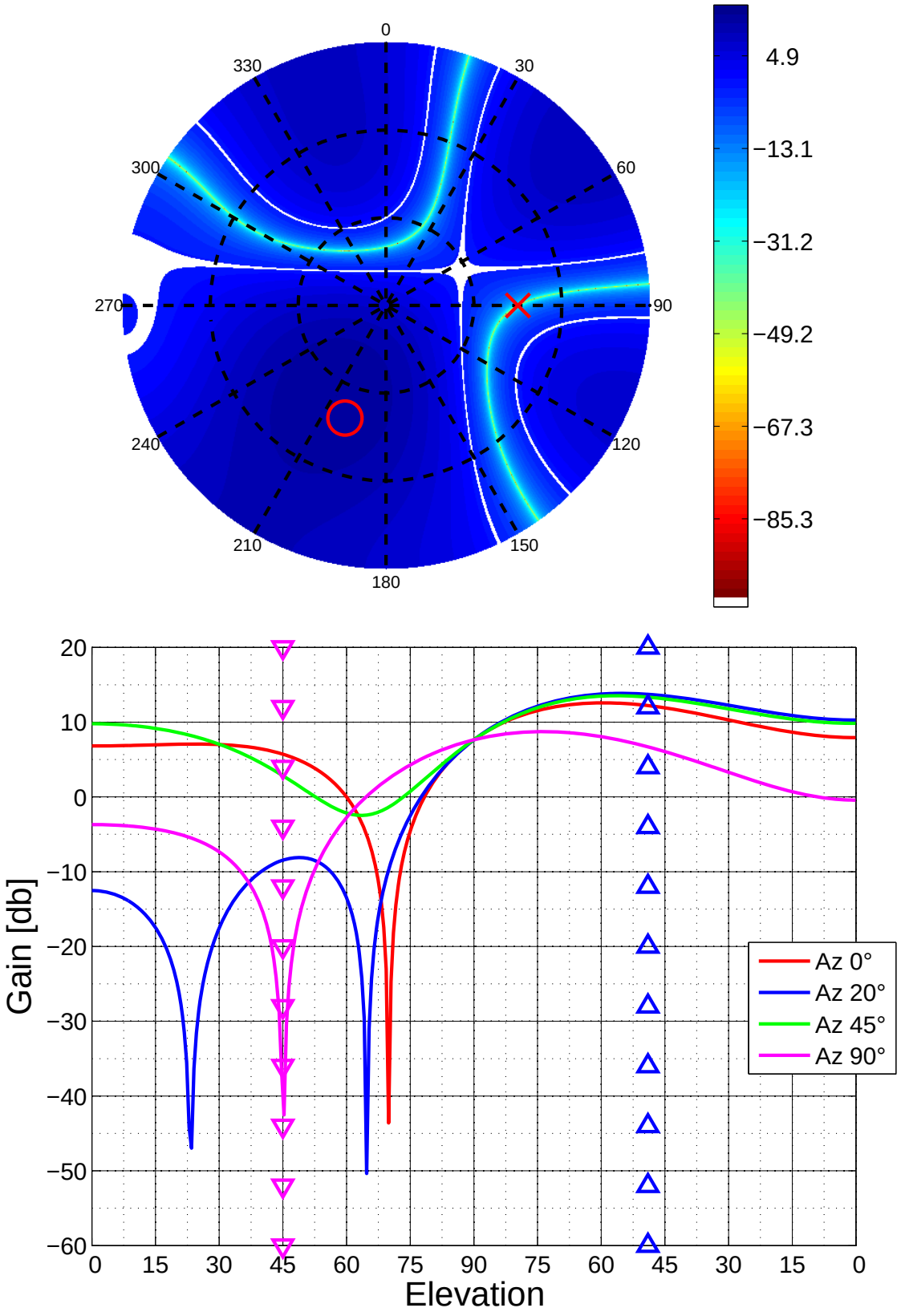


Figure 5.52: Antenna Pattern for Channel 10

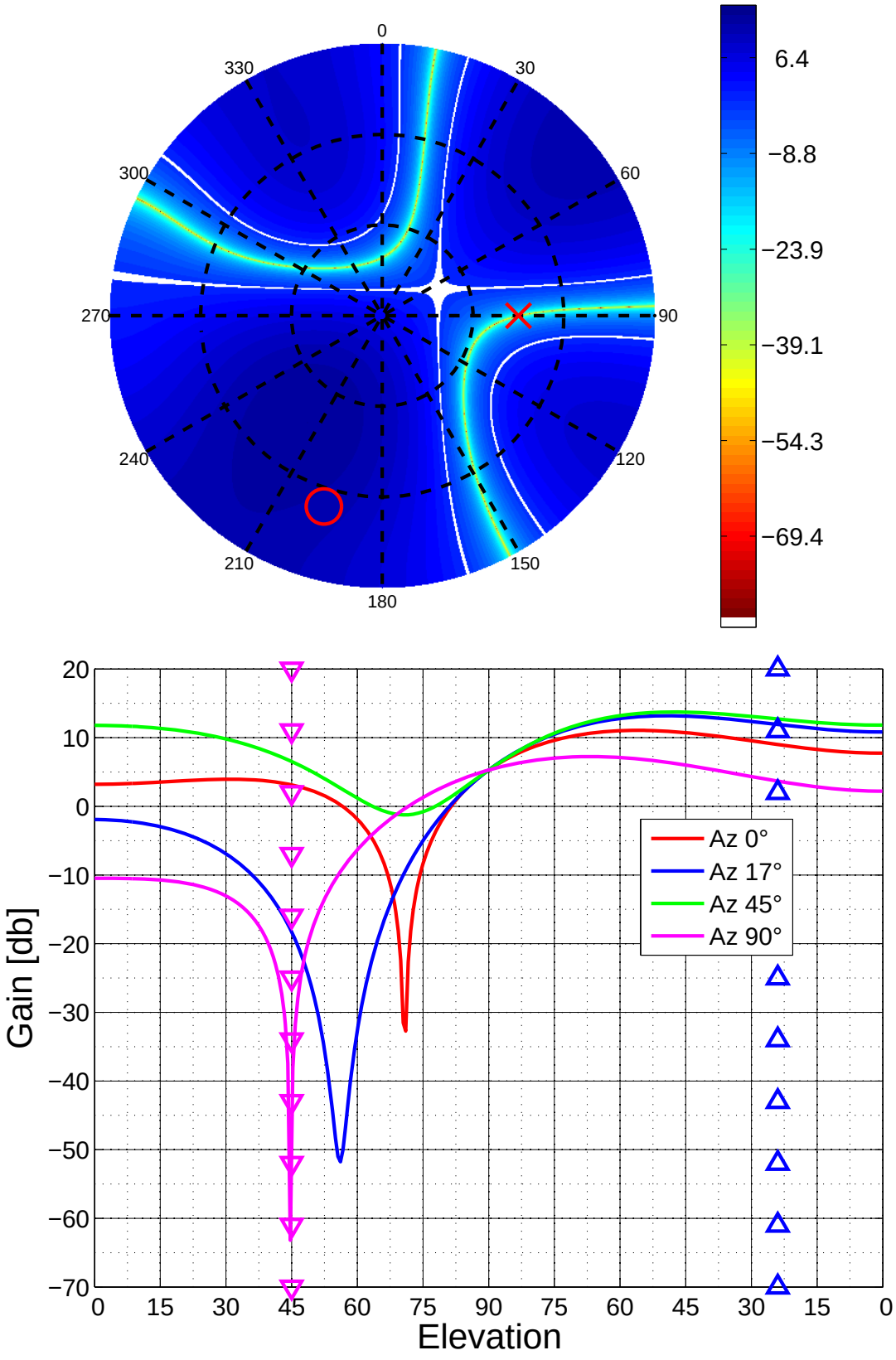


Figure 5.53: Antenna Pattern for Channel 11

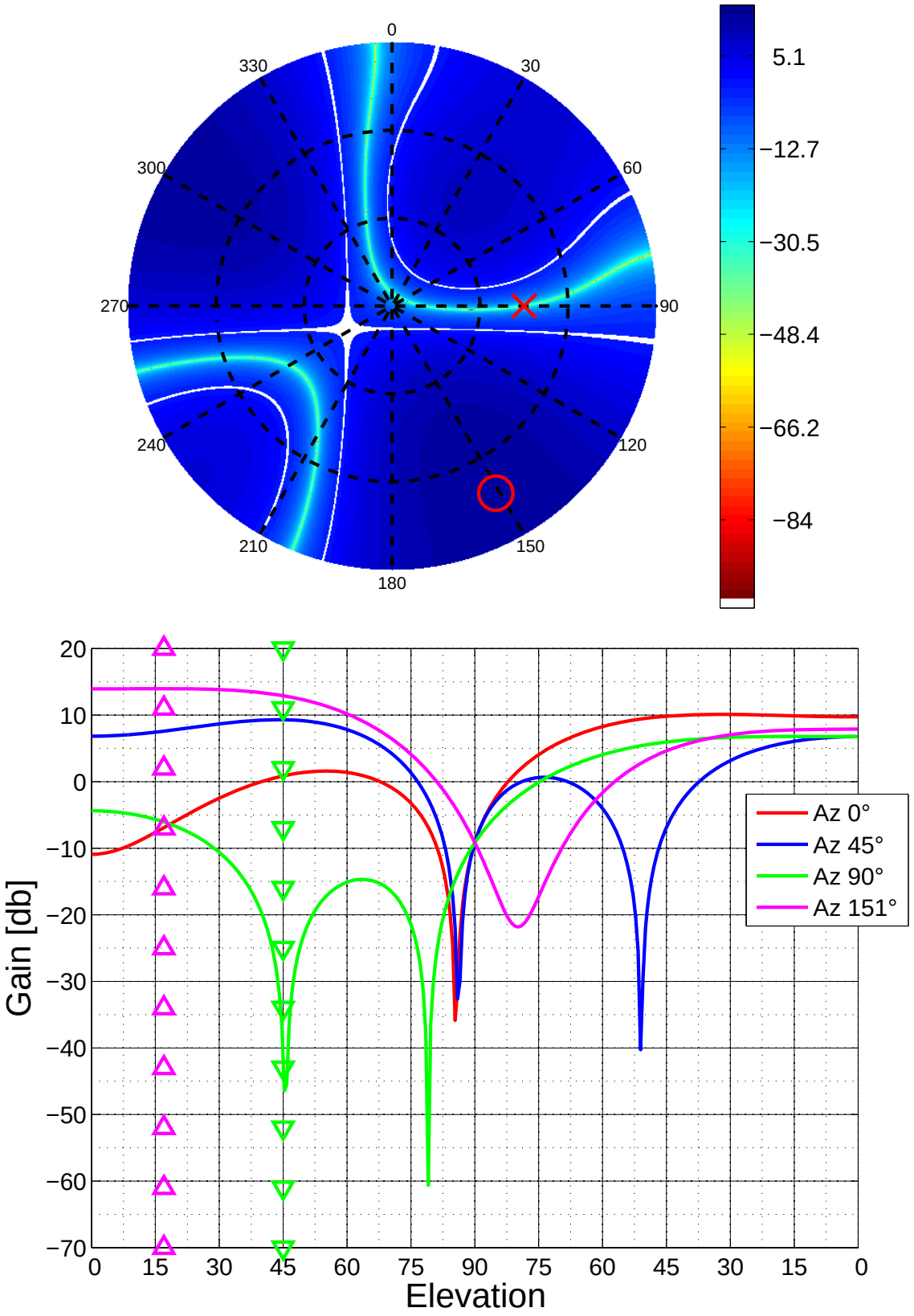


Figure 5.54: Antenna Pattern for Channel 14

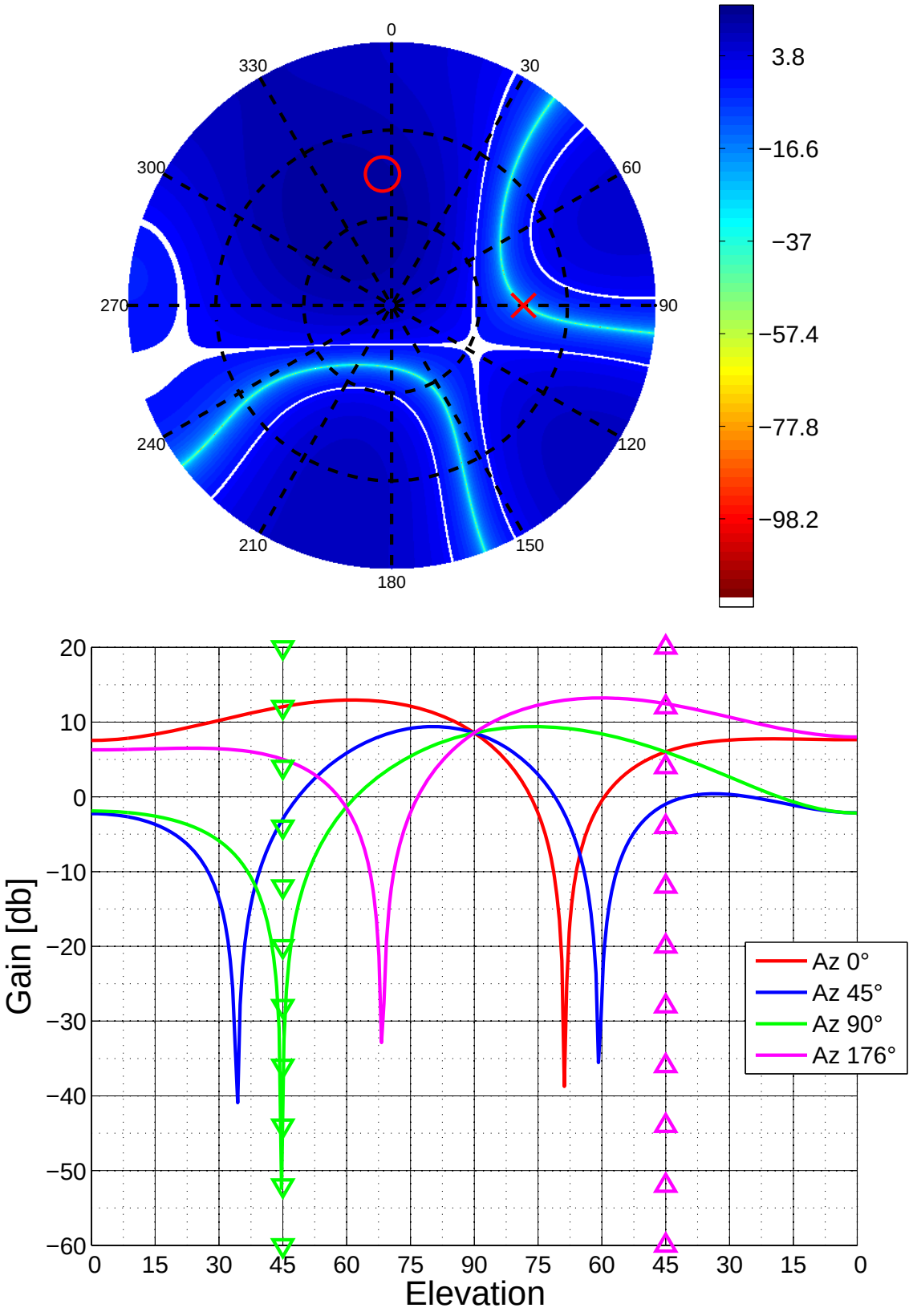


Figure 5.55: Antenna Pattern for Channel 16

Test C6 Nominal Mode + Interferer + 5 antenna in a circular configuration : automatic Antenna Pattern beam steering in the direction of the tracked SVs, and single null steered in the direction of the interferer zeros, alternate RF inputs are used. The calibration parameters are used this time.

Table 5.12: C6 Nominal mode test sequence and expected results example

N#	Command	Expected Response	Note
1	$PSPYN901,GET,SEL*00	$PSPYN901,ACK,SEL,OFF*0C	If the response doesn't match send $PSPYN901,SET,SEL, OFF and repeat step 1
2	$PSPYN901,SET,CPO,155,95,0, 95,90,95,180,95,270*00	$PSPYN901,ACK,CPO,155,95,0,95,90,95,180,95,270 *41	
3	$PSPYN901,GET,CXY*00	$PSPYN901,ACK,CXY,155,95,0,0,95,-95,0,0,-95*6A	Get the current antenna configuration.
4	$PSPYN901,NVR,ATT,45,0*00	$PSPYN901,ACK,ATT,45,0*45	Set the current antenna array attitude to 0 heading 0 pitch.
5	$PSPYN901,SET,CAL,155,50.0, 1.00,100.0,1.00,200.0,1.00,300.0 ,1.00*53	$PSPYN901,ACK,CAL,155,50.0,1.00,100.0,1.00,200. 0,1.00,300.0,1.00*53	Set the current calibration parameters
6	$PSPYN901,GET,CAL*00	$PSPYN901,ACK,CAL,155,50.0,1.00,100.0,1.00,200. 0,1.00,300.0,1.00*53	Get the current set of calibration parameters.
7	$PSPYN901,SET,INT,1,45,90*0 0	$PSPYN901,ACK,INT,1,45,90*73	
8	$PSPYN901,SET,SEL,NOM*00	$PSPYN901,ACK,SEL,NOM*0F	If the response doesn't match send $PSPYN901,SET,SEL, OFF and repeat step 1
9			Wait for the first fix. For each tracked satellite, take a note of the elevation azimuth and channel ID [CHID].
10	$PSPYN901,GET,RAW,1*00 $PSPYN901,GET,RAW,2*00 $PSPYN901,GET,RAW,3*00 $PSPYN901,GET,RAW,4*00 $PSPYN901,GET,RAW,5*00 $PSPYN901,GET,RAW,6*00 $PSPYN901,GET,RAW,7*00 $PSPYN901,GET,RAW,8*00 $PSPYN901,GET,RAW,9*00 $PSPYN901,GET,RAW,10*00 $PSPYN901,GET,RAW,11*00 $PSPYN901,GET,RAW,12*00 $PSPYN901,GET,RAW,13*00 $PSPYN901,GET,RAW,14*00 $PSPYN901,GET,RAW,15*00 $PSPYN901,GET,RAW,16*00	$PSPYN901,ACK,RAW,1,0,1,0,0,0,0,0,0,0,0*6D $PSPYN901,ACK,RAW,2,1,155,0,428,0,142,0,241,0,3 13*60 $PSPYN901,ACK,RAW,3,1,155,0,236,0,377,0,432,0,7 8*50 $PSPYN901,ACK,RAW,4,1,155,0,180,0,322,0,488,0,1 33*66 $PSPYN901,ACK,RAW,5,0,1,0,0,0,0,0,0,0,0*69 $PSPYN901,ACK,RAW,6,1,155,0,292,0,73,0,376,0,38 2*5D $PSPYN901,ACK,RAW,7,1,155,0,375,0,95,0,293,0,36 0*5A $PSPYN901,ACK,RAW,8,0,1,0,0,0,0,0,0,0,0*64 $PSPYN901,ACK,RAW,9,1,155,0,18,0,469,0,363,0,14 2*57 $PSPYN901,ACK,RAW,10,1,155,0,482,0,228,0,186,0, 227*52 $PSPYN901,ACK,RAW,11,1,155,0,471,0,198,0,198,0, 257*5F $PSPYN901,ACK,RAW,12,0,1,0,0,0,0,0,0,0,0*5F $PSPYN901,ACK,RAW,13,0,1,0,0,0,0,0,0,0,0*5E $PSPYN901,ACK,RAW,14,1,155,0,162,0,298,0,506,0, 157*5E $PSPYN901,ACK,RAW,15,0,1,0,0,0,0,0,0,0,0*58 $PSPYN901,ACK,RAW,16,1,155,0,311,0,412,0,357,0, 43*68	Verify that the values differ w.r.t. the ones computed at test C5 by the calibration parameter value.
11	$PSPYN901,SET,SEL,OFF*00	$PSPYN901,ACK,SEL,OFF*0C	Switch off the beamforming.
12	$PSPYN901,GET,RAW,1*00 $PSPYN901,GET,RAW,2*00 $PSPYN901,GET,RAW,3*00	$PSPYN901,ACK,RAW,1,0,1,0,0,0,0,0,0,0,0*6D $PSPYN901,ACK,RAW,2,0,1,0,0,0,0,0,0,0,0*6E $PSPYN901,ACK,RAW,3,0,1,0,0,0,0,0,0,0,0*6F	Chech that each channel beamformer is set back to its default

	$PSPYN901,GET,RAW,4*00 $PSPYN901,GET,RAW,5*00 $PSPYN901,GET,RAW,6*00 $PSPYN901,GET,RAW,7*00 $PSPYN901,GET,RAW,8*00 $PSPYN901,GET,RAW,9*00 $PSPYN901,GET,RAW,10*00 $PSPYN901,GET,RAW,11*00 $PSPYN901,GET,RAW,12*00 $PSPYN901,GET,RAW,13*00 $PSPYN901,GET,RAW,14*00 $PSPYN901,GET,RAW,15*00 $PSPYN901,GET,RAW,16*00	$PSPYN901,ACK,RAW,4,0,1,0,0,0,0,0,0,0,0*68 $PSPYN901,ACK,RAW,5,0,1,0,0,0,0,0,0,0,0*69 $PSPYN901,ACK,RAW,6,0,1,0,0,0,0,0,0,0,0*6A $PSPYN901,ACK,RAW,7,0,1,0,0,0,0,0,0,0,0*6B $PSPYN901,ACK,RAW,8,0,1,0,0,0,0,0,0,0,0*64 $PSPYN901,ACK,RAW,9,0,1,0,0,0,0,0,0,0,0*65 $PSPYN901,ACK,RAW,10,0,1,0,0,0,0,0,0,0,0*5D $PSPYN901,ACK,RAW,11,0,1,0,0,0,0,0,0,0,0*5C $PSPYN901,ACK,RAW,12,0,1,0,0,0,0,0,0,0,0*5F $PSPYN901,ACK,RAW,13,0,1,0,0,0,0,0,0,0,0*5E $PSPYN901,ACK,RAW,14,0,1,0,0,0,0,0,0,0,0*59 $PSPYN901,ACK,RAW,15,0,1,0,0,0,0,0,0,0,0*58 $PSPYN901,ACK,RAW,16,0,1,0,0,0,0,0,0,0,0*5B	configuration.

Here follows an example of the checks on the offset values:

*Test C6 channel 2 : $PSPYN901,ACK,RAW,2,1,155,0,428,0,142,0,241,0,313*60*

*Test C5 channel 2 : $$PSPYN901,ACK,RAW,2,1,155,0,499,0,284,0,13,0,228*5F*

*50/360*512 = 71.1 and 499 – 428 = 71 pass.*
*100/360*512 = 142.2 and 284 – 142 = 142 pass.*
*200/360*512 = 284.4 and mod(13 – 241,512) = 284 pass.*
*300/360*512 = 426.6 and mod(228 – 313,512) = 427 pass.*

Test C7 Nominal Mode plus interfering signal, 5 antenna in a circular configuration, dynamic conditions. Automatic Antenna Pattern beam steering in the direction of the tracked SVs, and single null steered in the direction of the interferer zeros. Some notes on the test :

- antenna configuration ⇒ alternate RF inputs are used;
- antenna gain mismatch ⇒ is simulated in the second part of the test;
- dynamic ⇒ entering appropriate attitude commands we simulate an heading rate of 15 deg/s.

This is mainly an Hosted Mode test: we are interested in verifying the beamforming algorithm performance not the system performance (refer to Live Tests for system performance).

Table 5.13 shows the nominal mode test sequence in dynamic conditions and expected results example.

Table 5.13: C7 Nominal mode test sequence

N#	Command	Expected Response	Note
1	$PSPYN901,GET,SEL*00	$PSPYN901,ACK,SEL,OFF*0C	If the response doesn't match send $PSPYN901,SET,SEL, OFF and repeat step 1
2	$PSPYN901,SET,CPO,155,95,0, 95,90,95,180,95,270*00	$PSPYN901,ACK,CPO,155,95,0,95,90,95,180,95,270 *41	
3	$PSPYN901,GET,CXY*00	$PSPYN901,ACK,CXY,155,95,0,0,95,-95,0,0,-95*6A	Get the current antenna configuration.
4	$PSPYN901,NVR,ATT,0,0*00	$PSPYN901,ACK,ATT,0,0*45	Set the current antenna array attitude to 0 heading 0 pitch.
5	SET,CAL,155,0,1.0,0,1.0,0,1.0,0, 1.0*00	$PSPYN901,ACK,CAL,155,0.0,1.00,0.0,1.00,0.0,1.00, 0.0,1.00*66	Set the current calibration parameters
6	$PSPYN901,GET,CAL*00	$PSPYN901,ACK,CAL,155,0.0,1.00,0.0,1.00,0.0,1.00, 0.0,1.00*66	Get the current set of calibration parameters.
7	$PSPYN901,SET,INT,1,45,90*0 0	$PSPYN901,ACK,INT,1,45,90*73	
8	$PSPYN901,SET,SEL,NOM*00	$PSPYN901,ACK,SEL,NOM*0F	If the response doesn't match send $PSPYN901,SET,SEL, OFF and repeat step 1
9			Wait for the first fix. For each tracked satellite, take a note of the elevation azimuth and channel ID [CHID].
Part 1			
10	$PSPYN901,GET,RAW,2*00 $PSPYN901,GET,RAW,3*00 $PSPYN901,GET,RAW,4*00 $PSPYN901,GET,RAW,6*00 $PSPYN901,GET,RAW,7*00 $PSPYN901,GET,RAW,9*00 $PSPYN901,GET,RAW,10*00 $PSPYN901,GET,RAW,11*00 $PSPYN901,GET,RAW,14*00 $PSPYN901,GET,RAW,16*00	$PSPYN901,ACK,RAW,2,1,155,0,300,0,179,0,212,0,3 32*60 $PSPYN901,ACK,RAW,3,1,155,0,323,0,155,0,189,0,3 57*6C $PSPYN901,ACK,RAW,4,1,155,0,321,0,157,0,191,0,3 55*60 $PSPYN901,ACK,RAW,6,1,155,0,65,0,186,0,447,0,32 6*57 $PSPYN901,ACK,RAW,7,1,155,0,91,0,229,0,421,0,28 3*55 $PSPYN901,ACK,RAW,9,1,155,0,2,0,394,0,510,0,118 *64 $PSPYN901,ACK,RAW,10,1,155,0,475,0,417,0,37,0,9 5*50 $PSPYN901,ACK,RAW,11,1,155,0,469,0,420,0,43,0,9 2*5C $PSPYN901,ACK,RAW,14,1,155,0,90,0,284,0,422,0,2 28*63 $PSPYN901,ACK,RAW,16,1,155,0,25,0,141,0,487,0,3 71*67	Get Raw phase offsets for the channel in tracking and check the values using the usual matlab tools.

11	$PSPYN901,SET,ATT,15,0*00 $PSPYN901,SET,ATT,30,0*00 $PSPYN901,SET,ATT,45,0*00 $PSPYN901,SET,ATT,60,0*00 $PSPYN901,SET,ATT,75,0*00 $PSPYN901,SET,ATT,90,0*00	$PSPYN901,ACK,ATT,15,0*40 $PSPYN901,ACK,ATT,30,0*47 $PSPYN901,ACK,ATT,45,0*45 $PSPYN901,ACK,ATT,60,0*42 $PSPYN901,ACK,ATT,75,0*46 $PSPYN901,ACK,ATT,90,0*4D	(Only Hosted Mode) Simulate Heading Rate Change.
12	$PSPYN901,GET,RAW,2*00 $PSPYN901,GET,RAW,3*00 $PSPYN901,GET,RAW,4*00 $PSPYN901,GET,RAW,6*00 $PSPYN901,GET,RAW,7*00 $PSPYN901,GET,RAW,9*00 $PSPYN901,GET,RAW,10*00 $PSPYN901,GET,RAW,11*00 $PSPYN901,GET,RAW,14*00 $PSPYN901,GET,RAW,16*00	$PSPYN901,ACK,RAW,2,1,155,0,328,0,296,0,185,0,215*61 $PSPYN901,ACK,RAW,3,1,155,0,357,0,323,0,155,0,189*6C $PSPYN901,ACK,RAW,4,1,155,0,354,0,320,0,158,0,192*6C $PSPYN901,ACK,RAW,6,1,155,0,316,0,73,0,196,0,439*5B $PSPYN901,ACK,RAW,7,1,155,0,283,0,91,0,230,0,421*5D $PSPYN901,ACK,RAW,9,1,155,0,187,0,295,0,324,0,217*65 $PSPYN901,ACK,RAW,10,1,155,0,95,0,475,0,417,0,37*50 $PSPYN901,ACK,RAW,11,1,155,0,93,0,470,0,419,0,42*5E $PSPYN901,ACK,RAW,14,1,155,0,224,0,88,0,288,0,424*6C $PSPYN901,ACK,RAW,16,1,155,0,371,0,24,0,141,0,488*69	Get Raw phase offsets for the channel in tracking and check the values using the usual matlab tools.
Part 2			
13	$PSPYN901,SET,CAL,155,0,1.1,0,0.9,0,1.2,0,0.95*00	$PSPYN901,ACK,CAL,155,0.0,1.10,0.0,0.90,0.0,1.20,0.0,0.95*60	Send a new set of calibration parameters to simulate antenna gain mismatch modeled.
14	$PSPYN901,GET,RAW,2*00 $PSPYN901,GET,RAW,3*00 $PSPYN901,GET,RAW,4*00 $PSPYN901,GET,RAW,6*00 $PSPYN901,GET,RAW,7*00 $PSPYN901,GET,RAW,9*00 $PSPYN901,GET,RAW,10*00 $PSPYN901,GET,RAW,11*00 $PSPYN901,GET,RAW,14*00 $PSPYN901,GET,RAW,16*00	$PSPYN901,ACK,RAW,2,1,155,0,326,0,295,0,177,0,219*6D $PSPYN901,ACK,RAW,3,1,155,0,365,0,329,0,154,0,191*6F $PSPYN901,ACK,RAW,4,1,155,0,359,0,319,0,160,0,197*65 $PSPYN901,ACK,RAW,6,1,155,0,318,0,85,0,198,0,427*5D $PSPYN901,ACK,RAW,7,1,155,0,288,0,109,0,230,0,402*67 $PSPYN901,ACK,RAW,9,1,155,0,90,0,474,0,414,0,45*6A $PSPYN901,ACK,RAW,10,1,155,0,104,0,466,0,411,0,40*6D $PSPYN901,ACK,RAW,11,1,155,0,105,0,465,0,409,0,44*63 $PSPYN901,ACK,RAW,14,1,155,0,221,0,104,0,292,0,409*58 $PSPYN901,ACK,RAW,16,1,155,0,380,0,33,0,148,0,501*68	Get Raw phase offsets for the channel in tracking and check the values using the usual matlab tools.
15	$PSPYN901,SET,ATT,75,0*00 $PSPYN901,SET,ATT,60,0*00 $PSPYN901,SET,ATT,45,0*00 $PSPYN901,SET,ATT,30,0*00 $PSPYN901,SET,ATT,15,0*00 $PSPYN901,SET,ATT,00,0*00	$PSPYN901,ACK,ATT,75,0*46 $PSPYN901,ACK,ATT,60,0*42 $PSPYN901,ACK,ATT,45,0*45 $PSPYN901,ACK,ATT,30,0*47 $PSPYN901,ACK,ATT,15,0*40 $PSPYN901,ACK,ATT,0,0*74	(Only Hosted Mode) Simulate Heading Rate Change.
16	$PSPYN901,GET,RAW,2*00 $PSPYN901,GET,RAW,3*00 $PSPYN901,GET,RAW,4*00 $PSPYN901,GET,RAW,6*00 $PSPYN901,GET,RAW,7*00 $PSPYN901,GET,RAW,9*00 $PSPYN901,GET,RAW,10*00	$PSPYN901,ACK,RAW,2,1,155,0,292,0,181,0,215,0,308*63 $PSPYN901,ACK,RAW,3,1,155,0,317,0,161,0,194,0,348*6E $PSPYN901,ACK,RAW,4,1,155,0,310,0,147,0,182,0,342*67 $PSPYN901,ACK,RAW,6,1,155,0,73,0,216,0,436,0,30	Get Raw phase offsets for the channel in tracking and check the values using the usual matlab tools.

	$PSPYN901,GET,RAW,11*00 $PSPYN901,GET,RAW,14*00 $PSPYN901,GET,RAW,16*00	2*5A $PSPYN901,ACK,RAW,7,1,155,0,83,0,245,0,429,0,269*50 $PSPYN901,ACK,RAW,9,1,155,0,469,0,451,0,53,0,55*69 $PSPYN901,ACK,RAW,10,1,155,0,487,0,423,0,30,0,87*5E $PSPYN901,ACK,RAW,11,1,155,0,486,0,426,0,31,0,84*59 $PSPYN901,ACK,RAW,14,1,155,0,79,0,278,0,429,0,242*60 $PSPYN901,ACK,RAW,16,1,155,0,30,0,142,0,491,0,377*61	
17	$PSPYN901,SET,SEL,OFF*00	$PSPYN901,ACK,SEL,OFF*0C	Switch off the beamforming.
18	$PSPYN901,GET,RAW,1*00 $PSPYN901,GET,RAW,2*00 $PSPYN901,GET,RAW,3*00 $PSPYN901,GET,RAW,4*00 $PSPYN901,GET,RAW,5*00 $PSPYN901,GET,RAW,6*00 $PSPYN901,GET,RAW,7*00 $PSPYN901,GET,RAW,8*00 $PSPYN901,GET,RAW,9*00 $PSPYN901,GET,RAW,10*00 $PSPYN901,GET,RAW,11*00 $PSPYN901,GET,RAW,12*00 $PSPYN901,GET,RAW,13*00 $PSPYN901,GET,RAW,14*00 $PSPYN901,GET,RAW,15*00 $PSPYN901,GET,RAW,16*00	$PSPYN901,ACK,RAW,1,0,1,0,0,0,0,0,0,0,0*6D $PSPYN901,ACK,RAW,2,0,1,0,0,0,0,0,0,0,0*6E $PSPYN901,ACK,RAW,3,0,1,0,0,0,0,0,0,0,0*6F $PSPYN901,ACK,RAW,4,0,1,0,0,0,0,0,0,0,0*68 $PSPYN901,ACK,RAW,5,0,1,0,0,0,0,0,0,0,0*69 $PSPYN901,ACK,RAW,6,0,1,0,0,0,0,0,0,0,0*6A $PSPYN901,ACK,RAW,7,0,1,0,0,0,0,0,0,0,0*6B $PSPYN901,ACK,RAW,8,0,1,0,0,0,0,0,0,0,0*64 $PSPYN901,ACK,RAW,9,0,1,0,0,0,0,0,0,0,0*65 $PSPYN901,ACK,RAW,10,0,1,0,0,0,0,0,0,0,0*5D $PSPYN901,ACK,RAW,11,0,1,0,0,0,0,0,0,0,0*5C $PSPYN901,ACK,RAW,12,0,1,0,0,0,0,0,0,0,0*5F $PSPYN901,ACK,RAW,13,0,1,0,0,0,0,0,0,0,0*5E $PSPYN901,ACK,RAW,14,0,1,0,0,0,0,0,0,0,0*59 $PSPYN901,ACK,RAW,15,0,1,0,0,0,0,0,0,0,0*58 $PSPYN901,ACK,RAW,16,0,1,0,0,0,0,0,0,0,0*5B	Chech that each channel beamformer is set back to its default configuration.

In order to properly verify the behaviour of the beamforming in dynamic conditions a debug message was implemented in order to get the phase offsets data at 1Hz rate. The following plot show heading profile, amplitude gain and attenuation vs time for every channel in both cases :

1. Part 1 – No antenna gains mismatch;
2. Part 2 – Antenna gains mismatch modelled;

Antenna amplitude gain and attenuation in both cases are computed using the same model used by the beamforming algorithm.

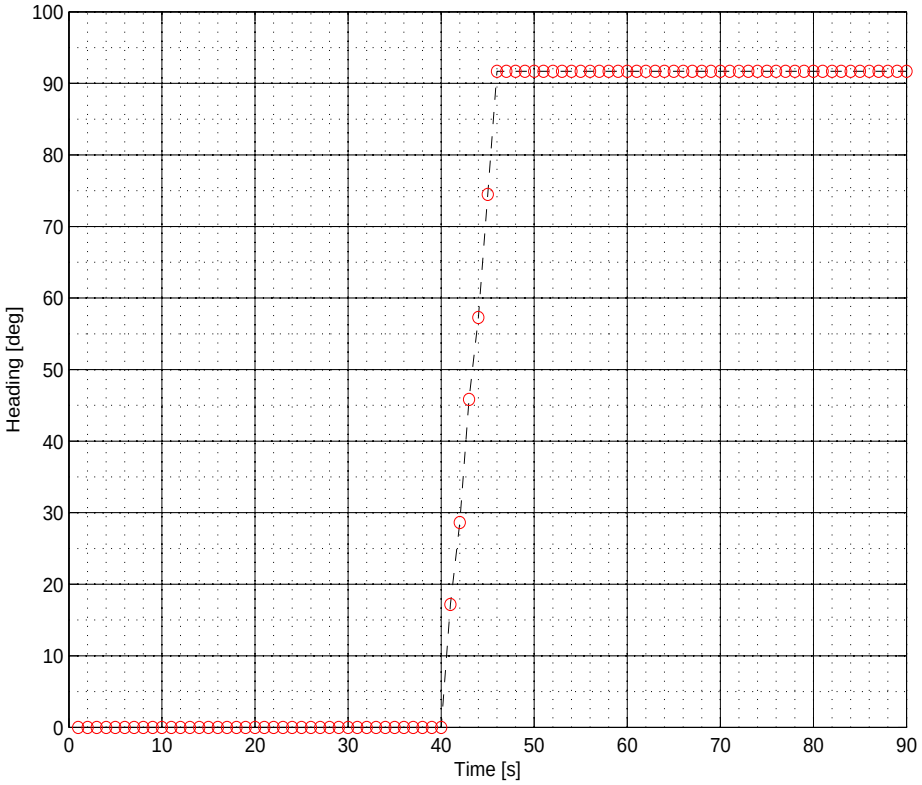


Figure 5.56: (part 1) Heading profile

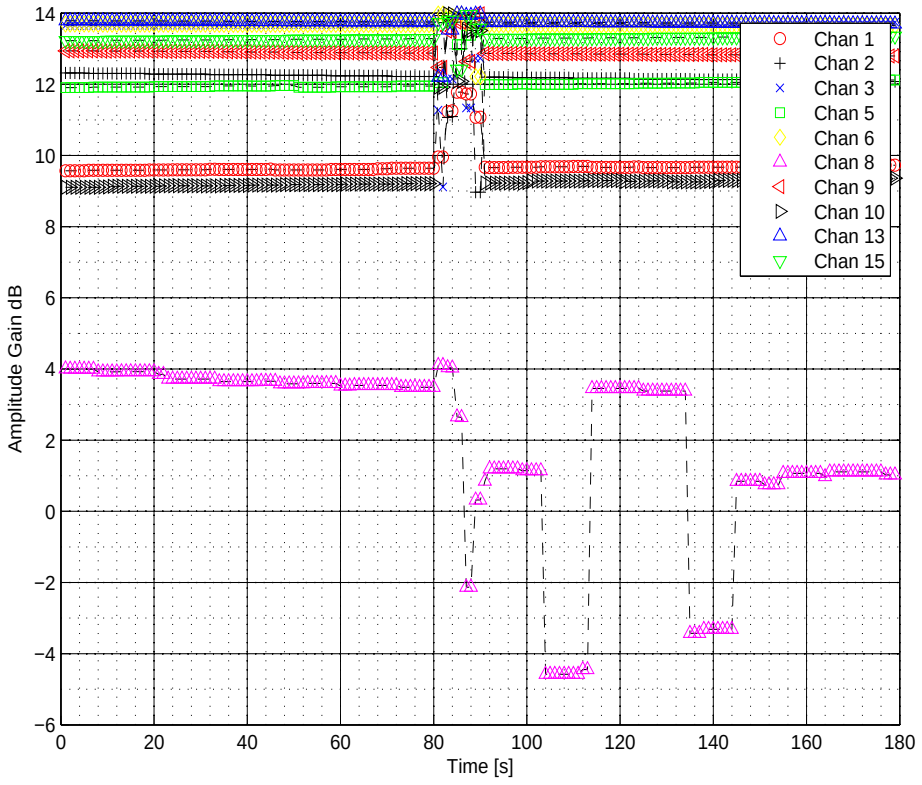


Figure 5.57: (part 1)Antenna array amplitude gain in static and dynamic conditions (no antenna gain mismatch)

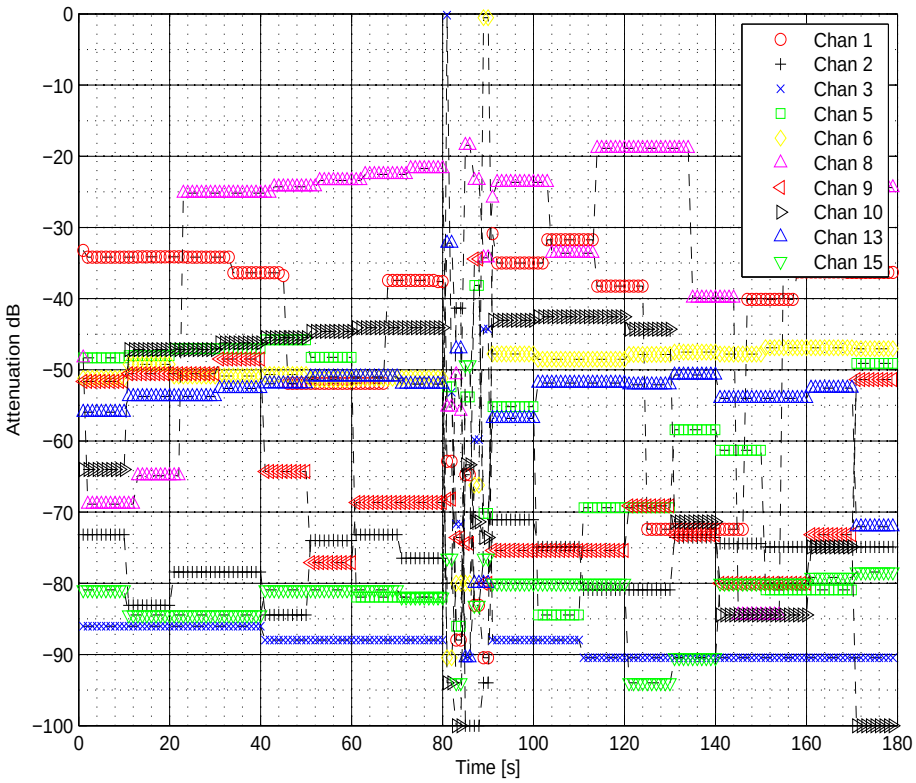


Figure 5.58: (part 1)Antenna array attenuation in static and dynamic conditions (no antenna gain mismatch

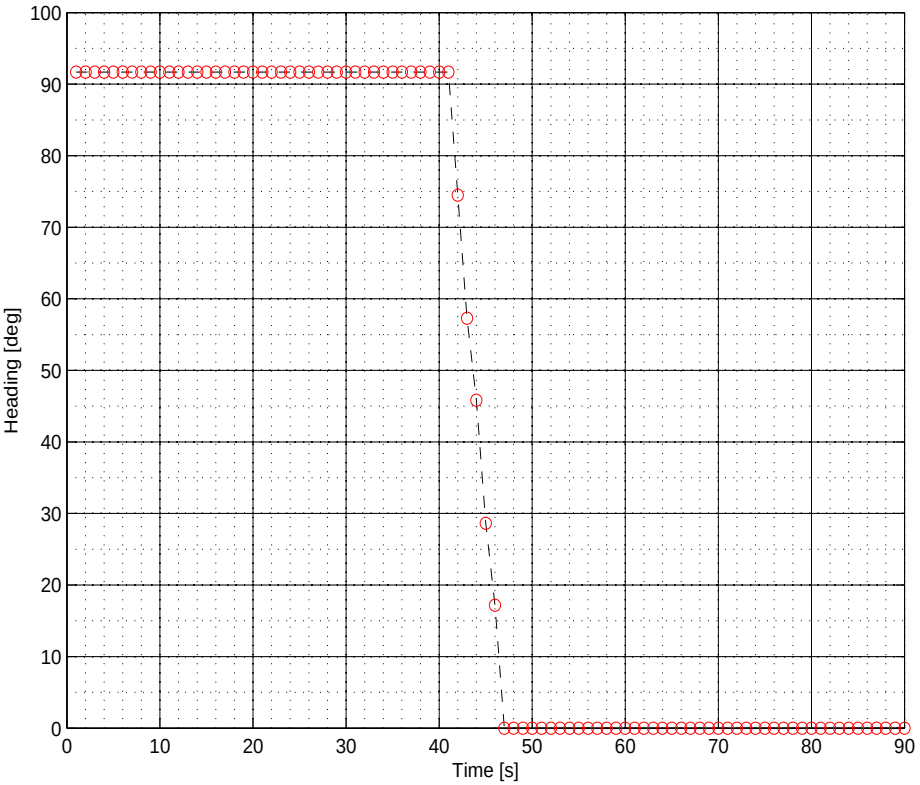


Figure 5.59: (part 2) Heading profile.

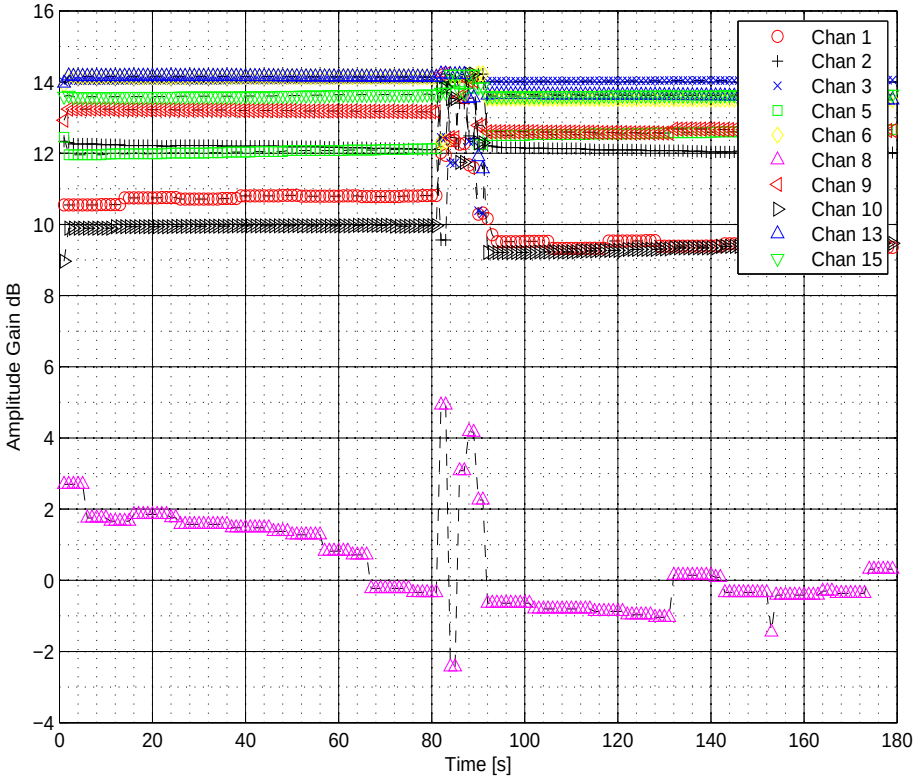


Figure 5.60: (part 2)Antenna array amplitude gain in static and dynamic conditions with antenna gain mismatch compensation.

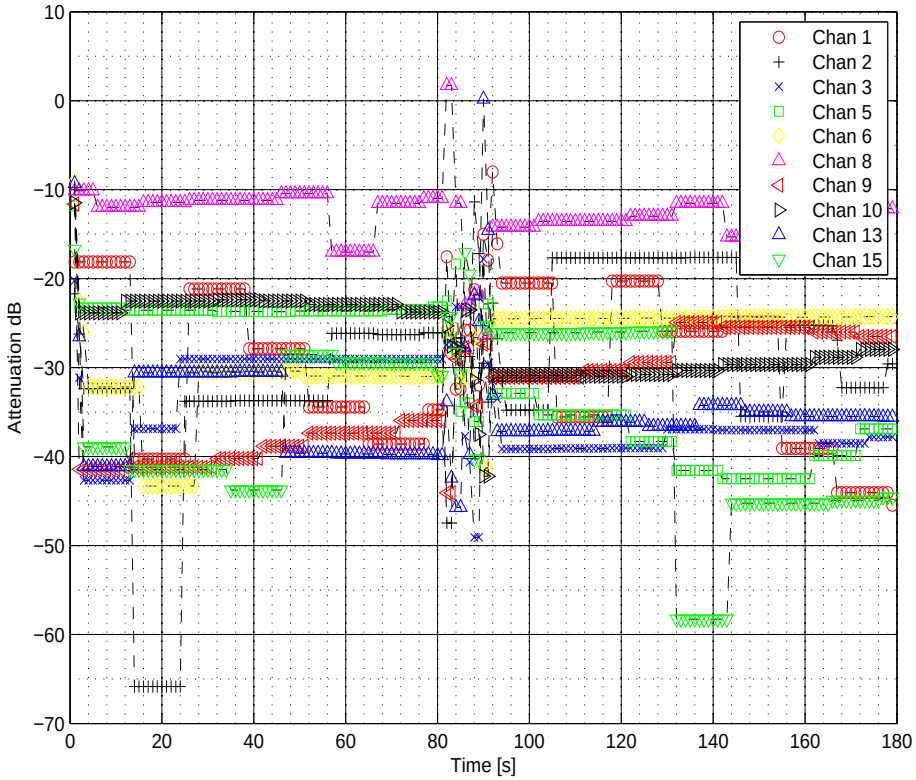


Figure 5.61: (part 2)Antenna array attenuation in static and dynamic conditions with antenna gain mismatch compensation

Figure 5.56 to figure 5.61 show that :

- except for channel number 8 tracking an SV at 56° elevation 98° azimuth (i.e. less than 15° away from the interference), all the other channels exhibit acceptable behavior.
- Except for channel 8 also in case of antenna gains mismatch the minimum attenuation wrt to the single antenna is around 20 dB or higher.

Figure 5.62 to figure 5.66 refer to channel 16 and show the improvement (10-15 [dB]) associated to antenna gains mismatch modeling.

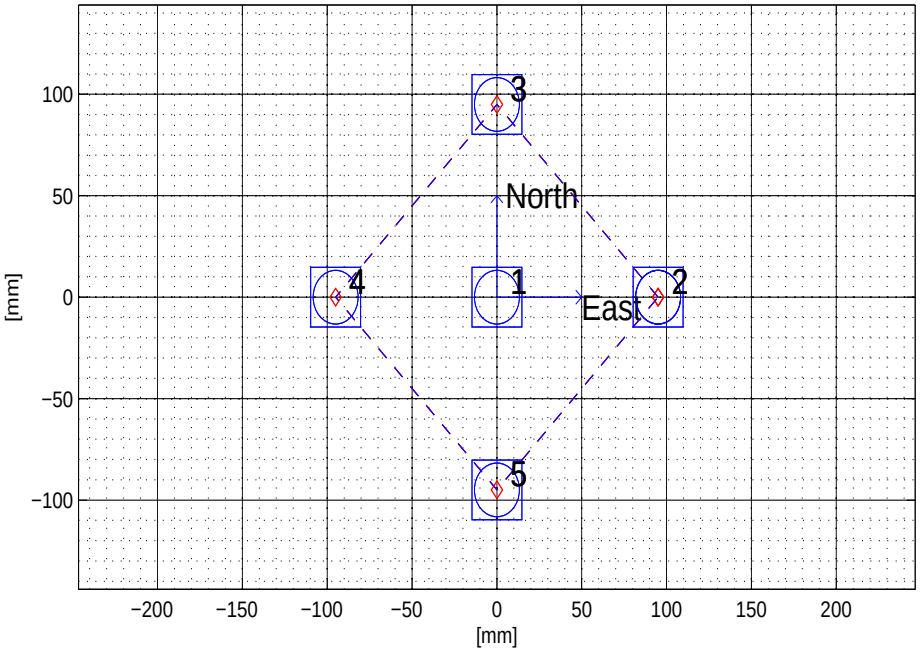


Figure 5.62: Antenna Array Map

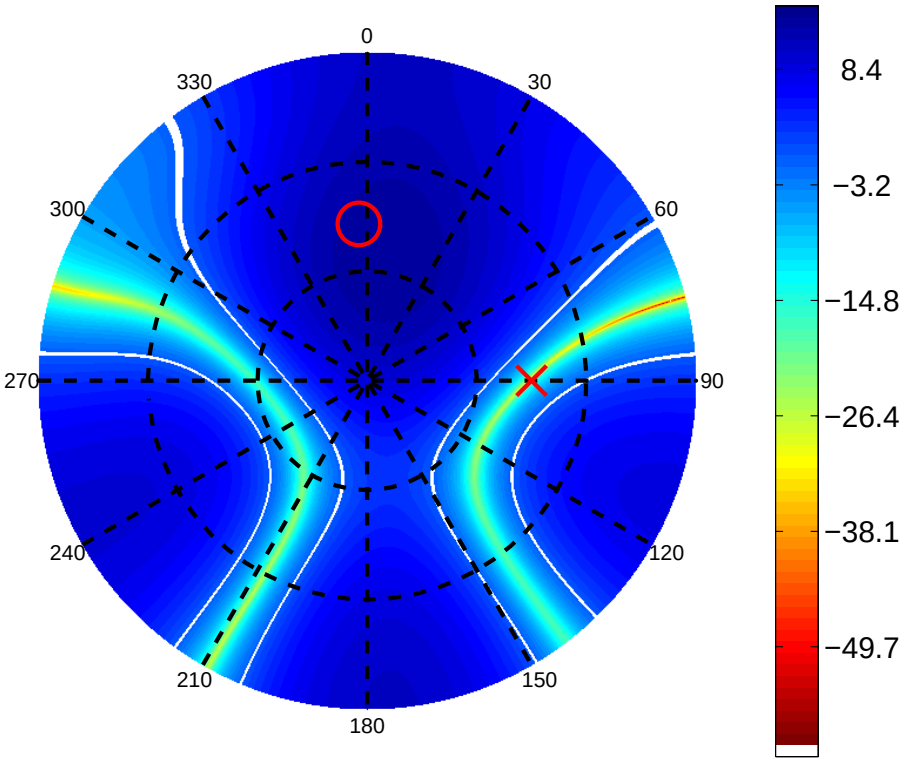


Figure 5.63: Antenna Pattern for Channel 16 : unmodelled antenna gains mismatch

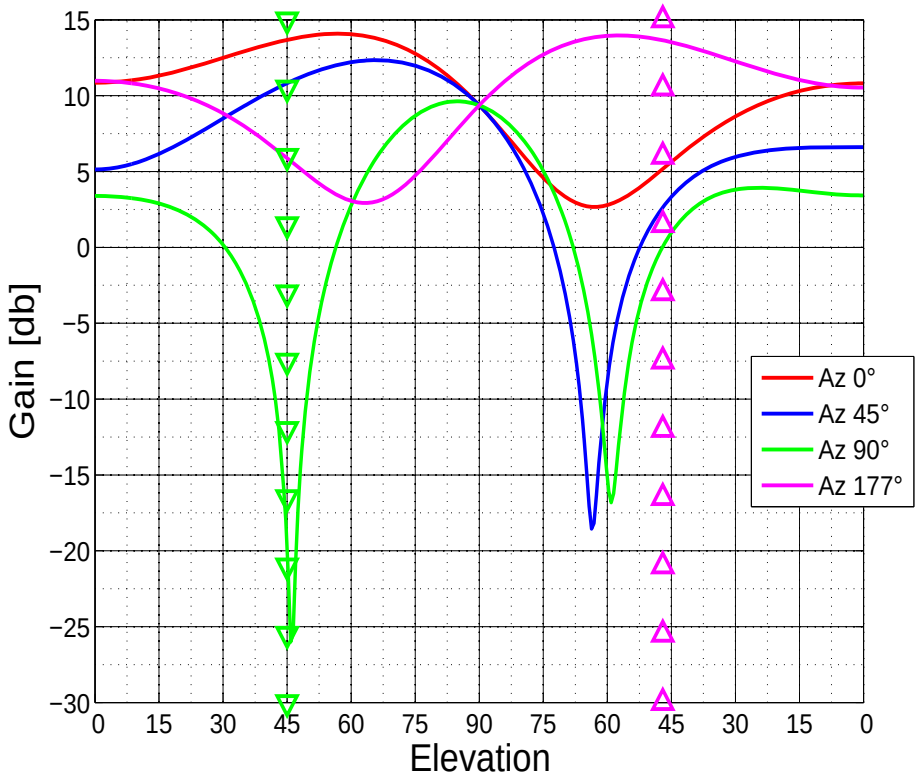


Figure 5.64: Antenna Pattern for Channel 16 : unmodelled antenna gains mismatch

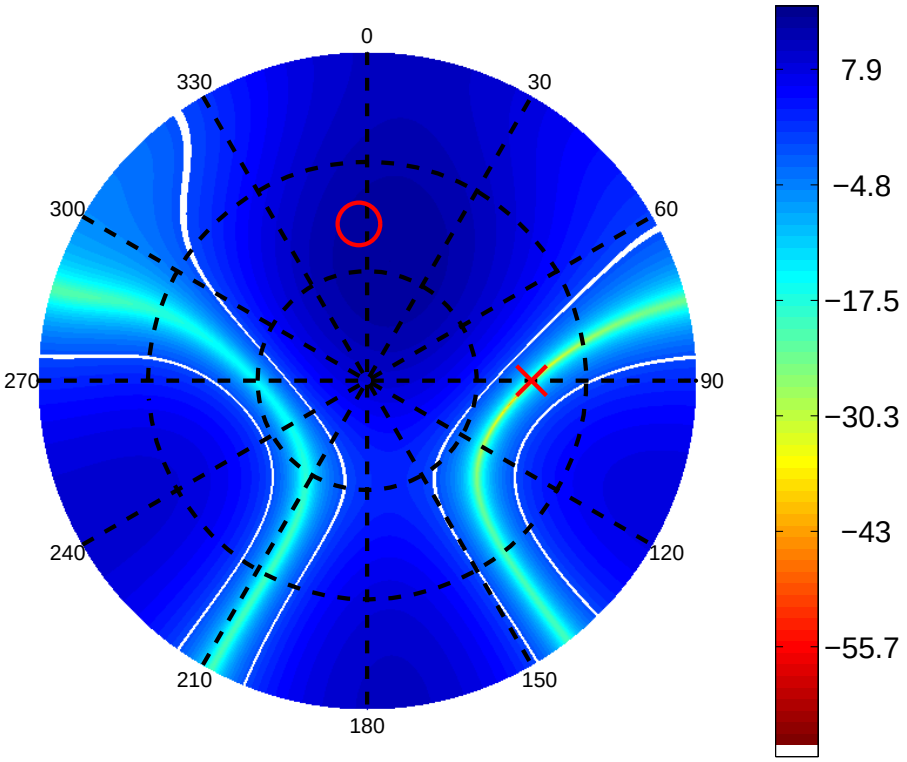


Figure 5.65: Antenna Pattern for Channel 16 : modelled antenna gains mismatch

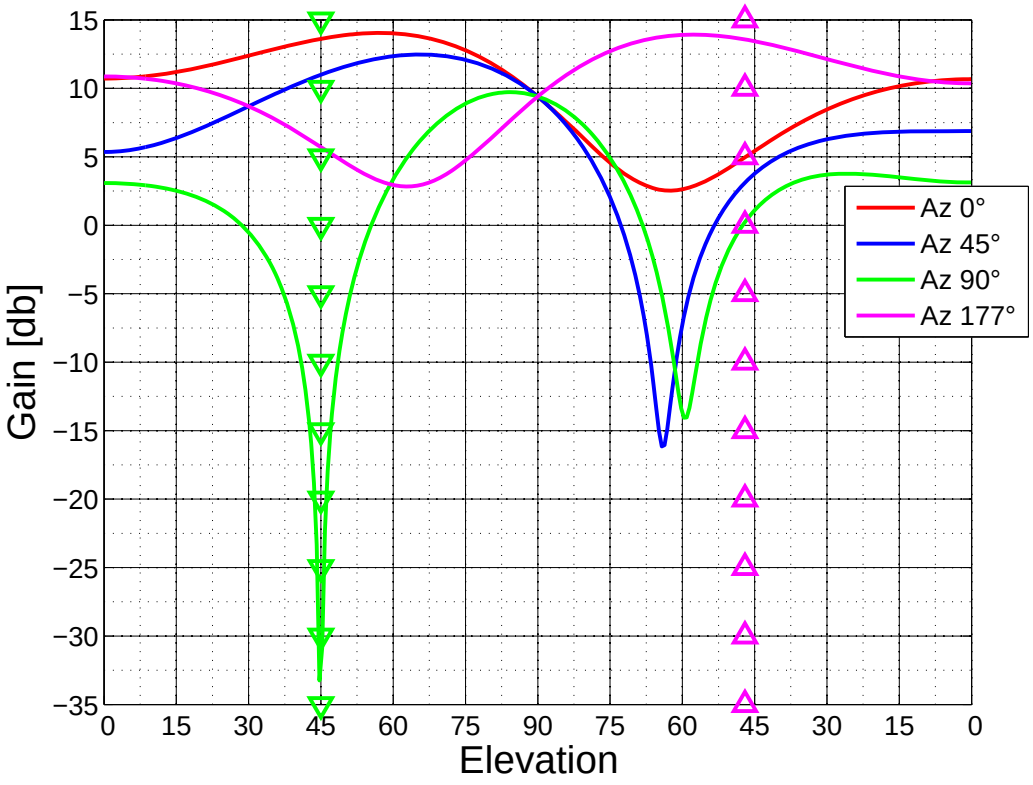


Figure 5.66: Antenna Pattern for Channel 16 : modelled antenna gains mismatch

5.2.4 Basic Tests Pass/Fail Table

Table 5.14: Tests Results Summary Table. (*) Tests covered by Live Tests

Test	Pass/Fail		Notes
	WinXP	SY1031	
A1	**Pass**	**Pass (*)**	
A2	**Pass**	**Pass (*)**	
B1	**Pass**	**Pass (*)**	
B2	**Pass**	**Pass (*)**	
C1	**Pass**	**Pass (*)**	
C2	**Pass**	**Pass (*)**	
C3	**Pass**	**Pass (*)**	
C4	**Pass**	**Pass (*)**	
C5	**Pass**	**Pass (*)**	
C6	**Pass**	**Pass (*)**	

5.3 Live Tests

5.3.1 Calibration Test Using the Simulator

The following plots show the results of the calibration test at the simulator. For obvious reasons the active antenna configuration was set to 7 virtual antennas all seven in the same position : the calibration procedure in this way computes a null geometric offset for the selected satellite. The GPS simulator signal is amplified by an active RF splitter and from this device distributed to 7 1006 RFFEs.

The automatic calibration procedure consists in selecting the highest elevation satellite among the tracked satellites, allocating a second channel (test channel) to track that satellite, and executing phase measurements differences (phase offsets measurement) between the *Test channel* and the *Reference channel*. While the *Reference channel* is in its default configuration (RF input 1 enabled) on the *Test channel* the active RF input is changed every measurement period among all (except Input 1) the RF Inputs of the current active antenna configuration. Each measurements period lasts 10 seconds and consists in 8 measurements.

The first two plots shows the value of each phase offset measurement for each RF input versus time. In each plot the black dotted line is a moving average over 20 samples and the red lines the moving standard deviation.

To verify that the calibration is channel independent the calibration was executed on 2 SVs , PRN 3 and PRN 19 involving in this way 4 different channels.

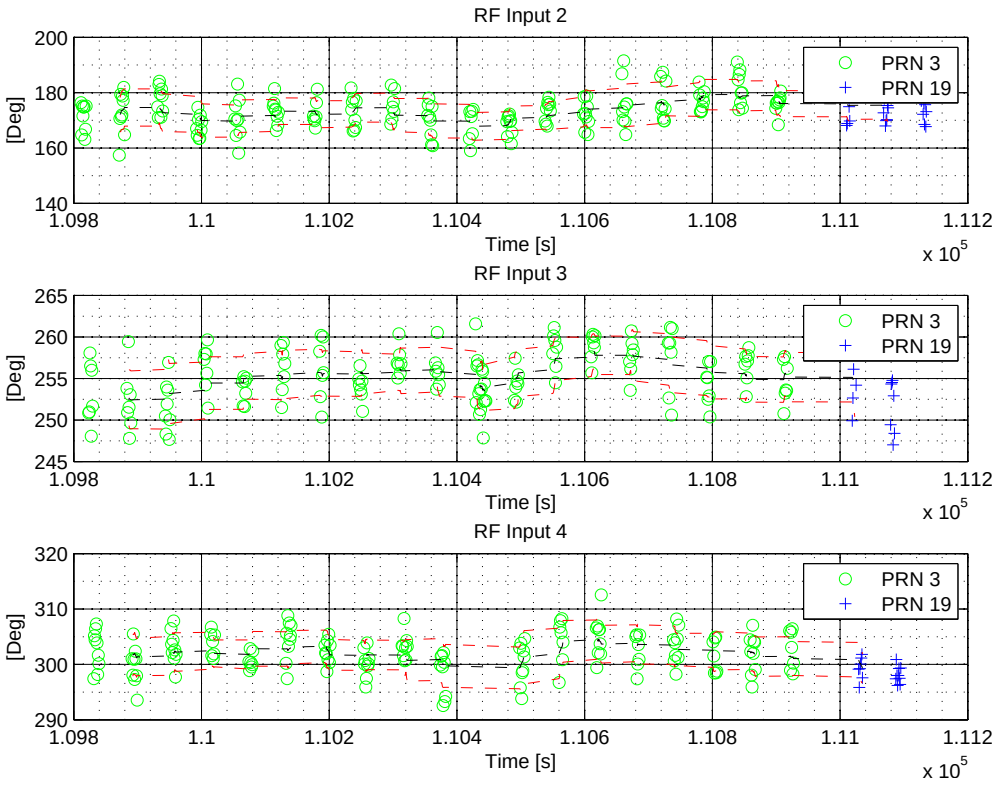


Figure 5.67: 1 Phase Offsets for RF Inputs 2 - 4

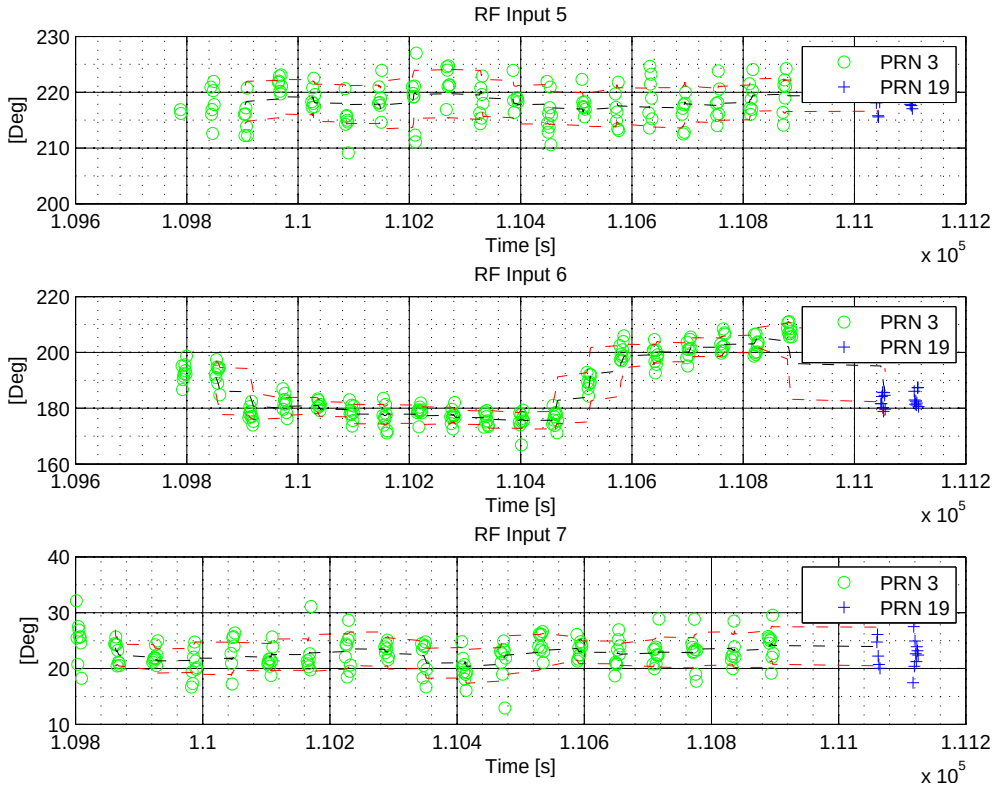


Figure 5.68: Phase Offsets for RF Inputs 5 – 7

In figure 5.69 the 20 samples unbiased moving average has been plotted for each RF input.

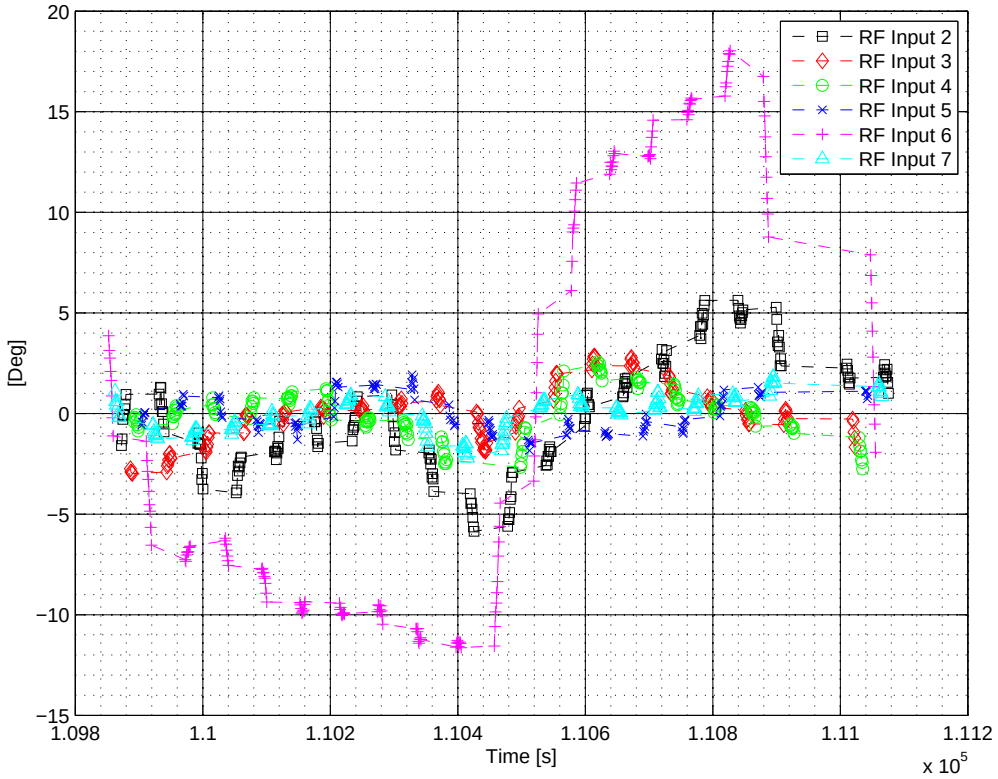


Figure 5.69: "De-biased" Phase Offsets for RF Inputs 2 - 7

In figure 5.70 the 20 samples moving standard deviation has been plotted for each RF input.

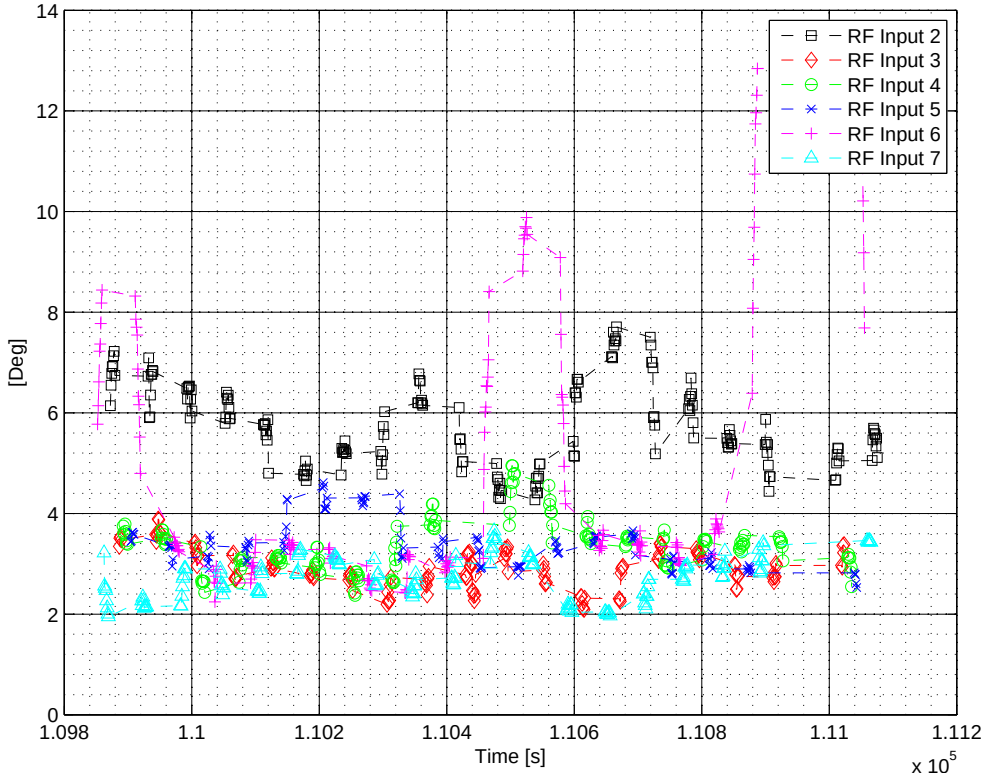


Figure 5.70: Phase Offset Samples Standard Deviation for RF Inputs 2 - 7

From the above results we can draw the following conclusions :

1. The calibration is GPS channel independent (as expected): the measured phase offset did not change after changing the selected PRN.

2. All the RF inputs exhibit slow temporal variation of the channel delay , in the order of +/- 2 deg. Two RF inputs , 2 and 6 exhibits a not acceptable wandering behavior. with variations above 5 degrees. A test with passive splitters is needed to understand whether the sources of the slow varying delay and of the erratic behavior of RF inputs 2 and 6 are the RFFEs, the active splitter or both.

3. The standard deviation of each sample is in average 3 deg. To get a calibration error below 1 deg with 99% confidence in the ideal case, at least 100 measurements are needed, this takes the minimum required calibration time to 15 min.

5.3.2 Further Calibration Tests Using the Simulator

(source LogCalibrazione 9) : all the RFFEs have been covered with a card box, the SMA connector were fastened , the results improved in terms of drift magnitude over time.

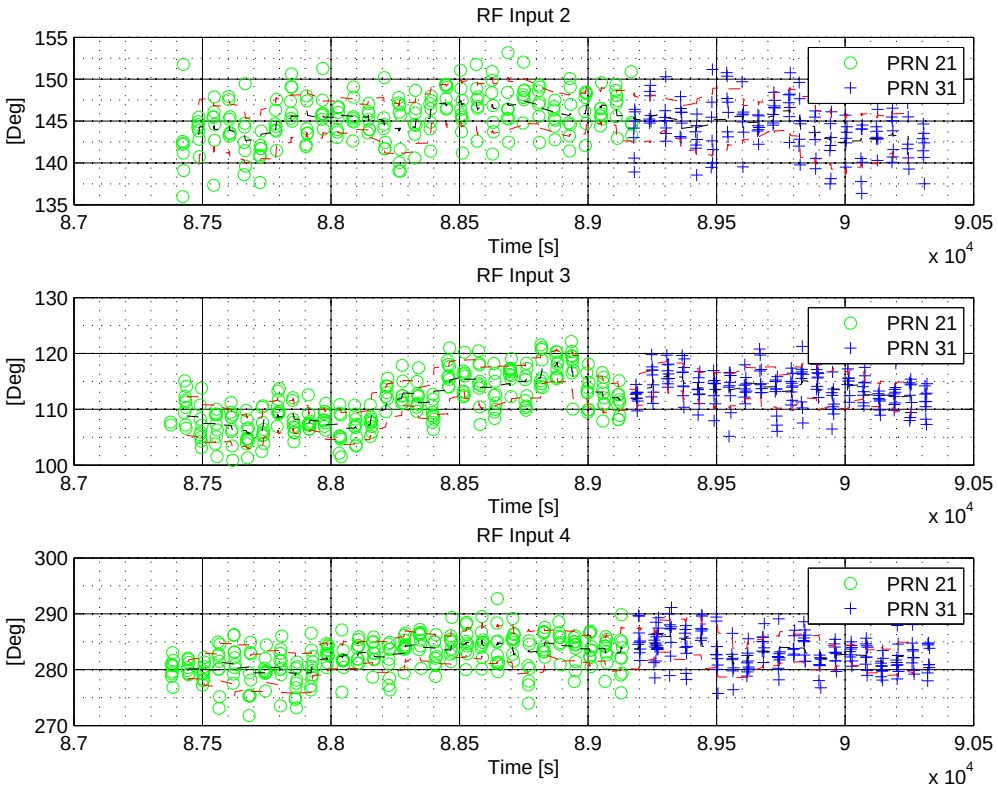


Figure 5.71: Phase Offsets for RF Inputs 2 – 4

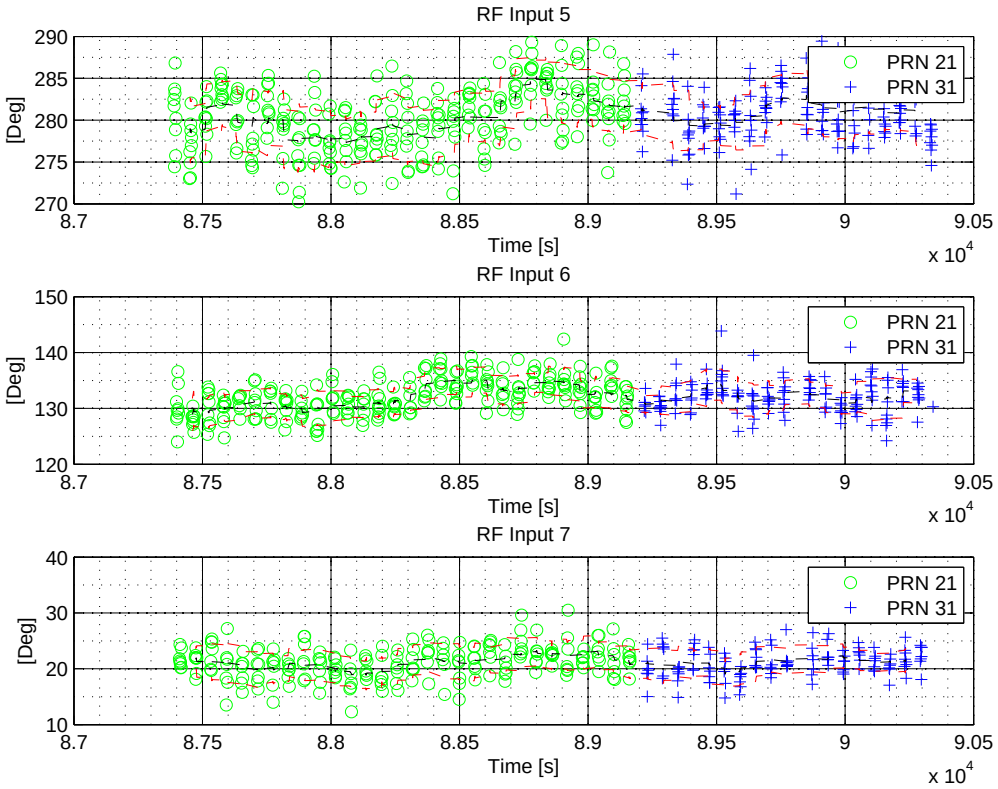


Figure 5.72: Phase Offsets for RF Inputs 5 - 7

In figure 5.73 the 20 samples unbiased moving average has been plotted for each RF input.

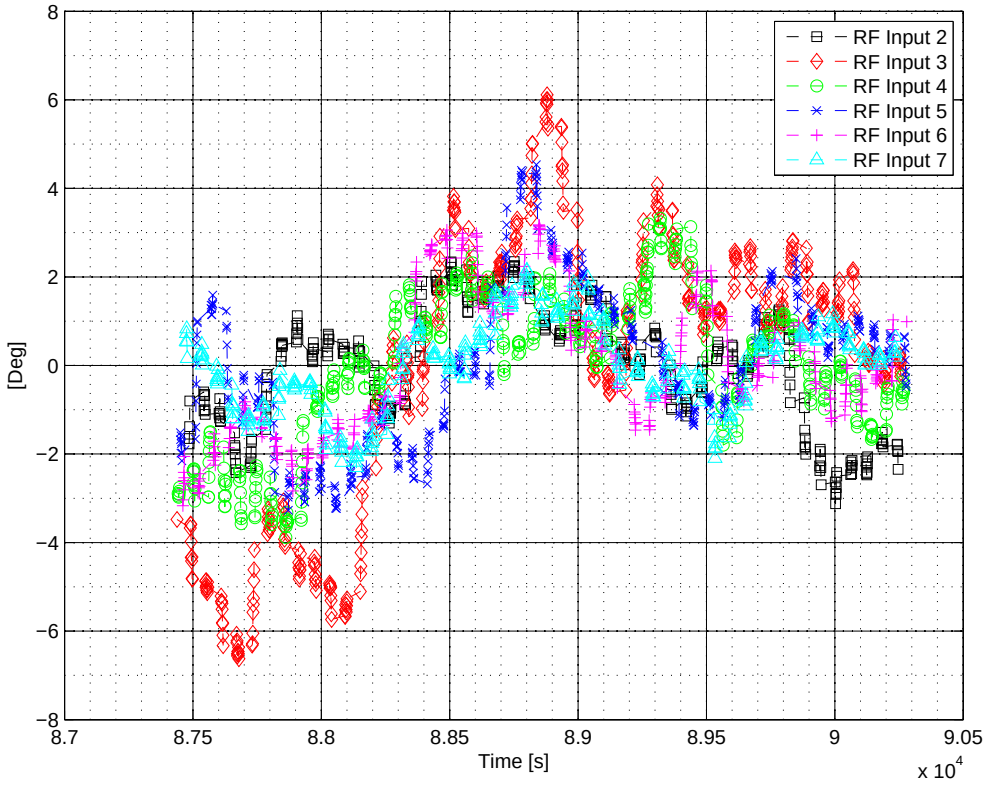


Figure 5.73: "De-biased" Phase Offsets for RF Inputs 2 - 7

In figure 5.74 the 20 samples moving standard deviation has been plotted for each RF input.

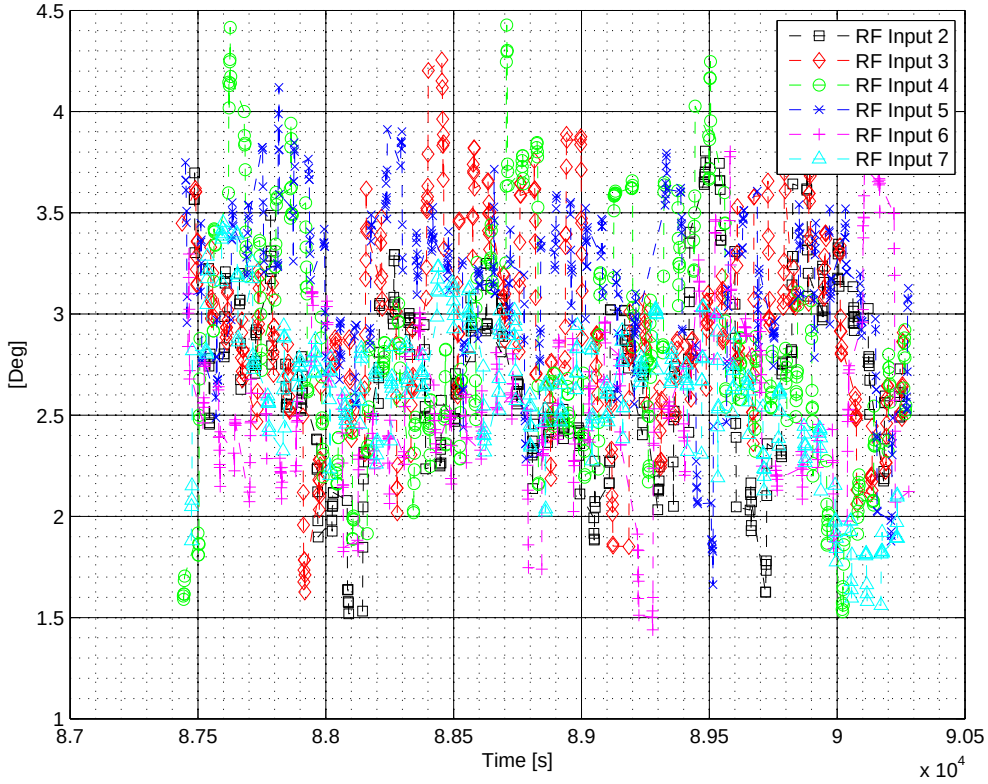


Figure 5.74: Phase Offset Samples Standard Deviation for RF Inputs 2 – 7

From the above results we can draw the following conclusions :

4. The calibration is GPS channel independent (as expected): the measured phase offset did not change after changing the selected PRN.

5. All the RF inputs exhibit slow temporal variation of the channel delay , in the order of +/- 3 deg (per channel.

6. The standard deviation of each sample is in average 3 deg. To get a calibration error below 1 deg with 99% confidence in the ideal case, at least 100 measurements are needed, this takes the minimum required calibration time to 15 min.

5.3.3 Calibration Temperature Sensitivity Assessment

Results for ~50 deg temperature variation cycle (25 -> 75 -> 25) : beware, the phase offset is represented modulo 360 deg.

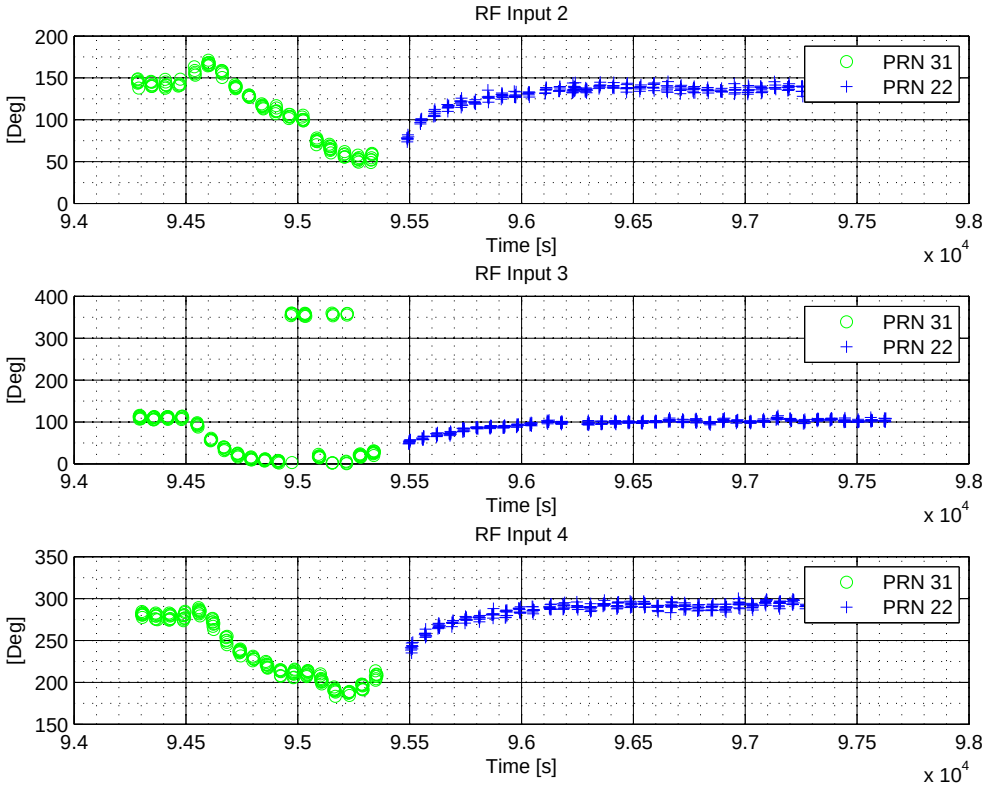


Figure 5.75: Phase Offsets for RF Inputs 2 - 4

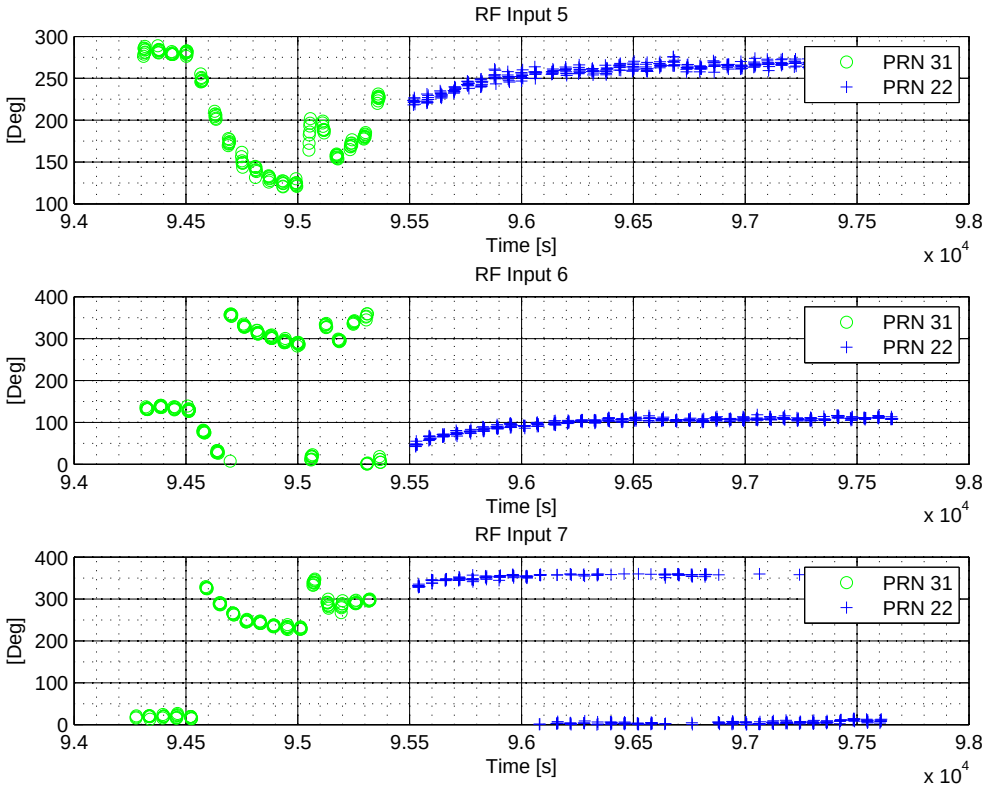


Figure 5.76: Phase Offsets for RF Inputs 5 - 7

The aim of the test was to measure the effect of the temperature on the phase offset. We tried to warm up all the LNAs trying to keep them at the same temperature but this was difficult to achieve because we could not use a thermal chamber and the RFFE boards are not thermally coupled. The huge variation of the phase offset can be explained with temperature sensitivity coefficient differences between two board higher than 2 deg per degree Celsius or by temperature sensitivity coefficients higher than 20 deg per degree Celsius.

We can notice also a non negligible hysteresis.

5.3.4 Calibration Connecting Simulator to Antenna, using Ant. Array to Receive Signal

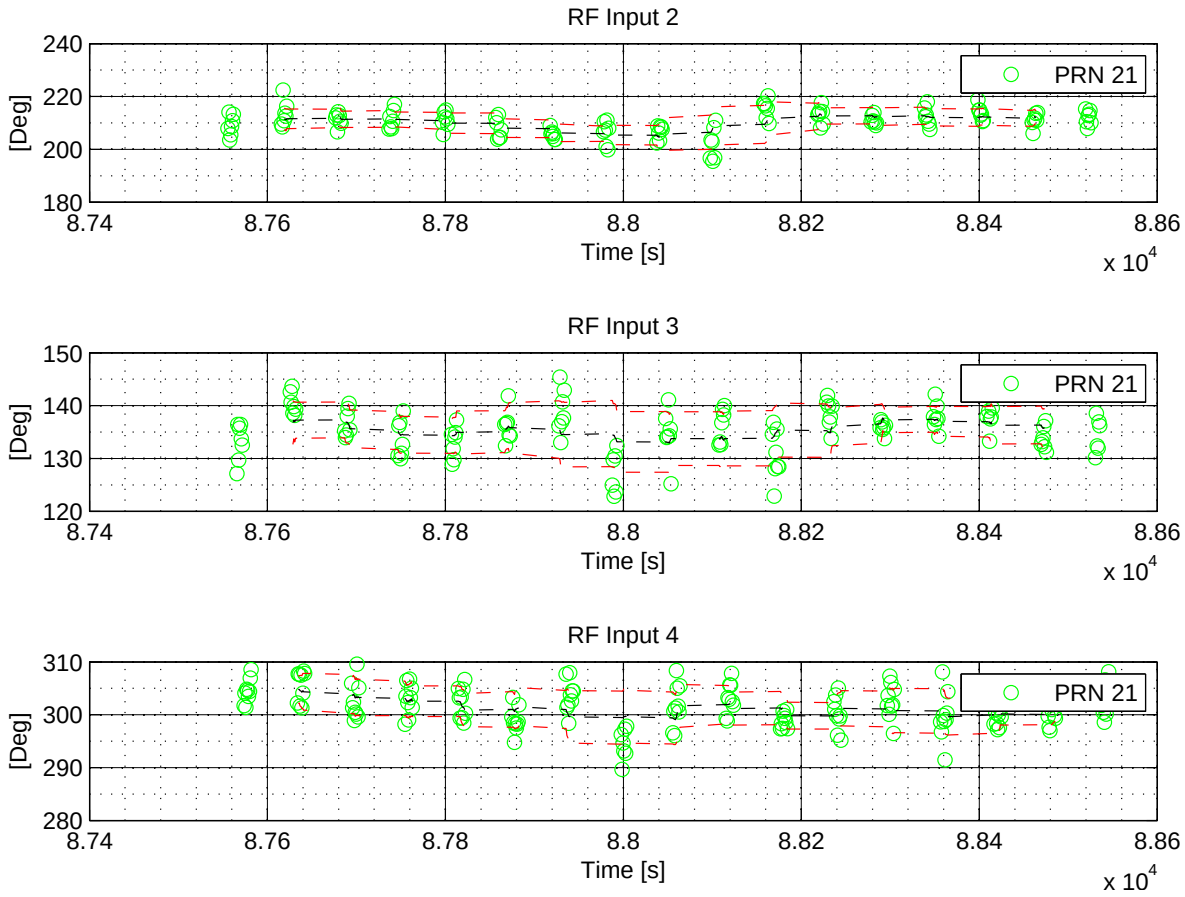


Figure 5.77: Phase Offsets for RF Inputs 2 - 4

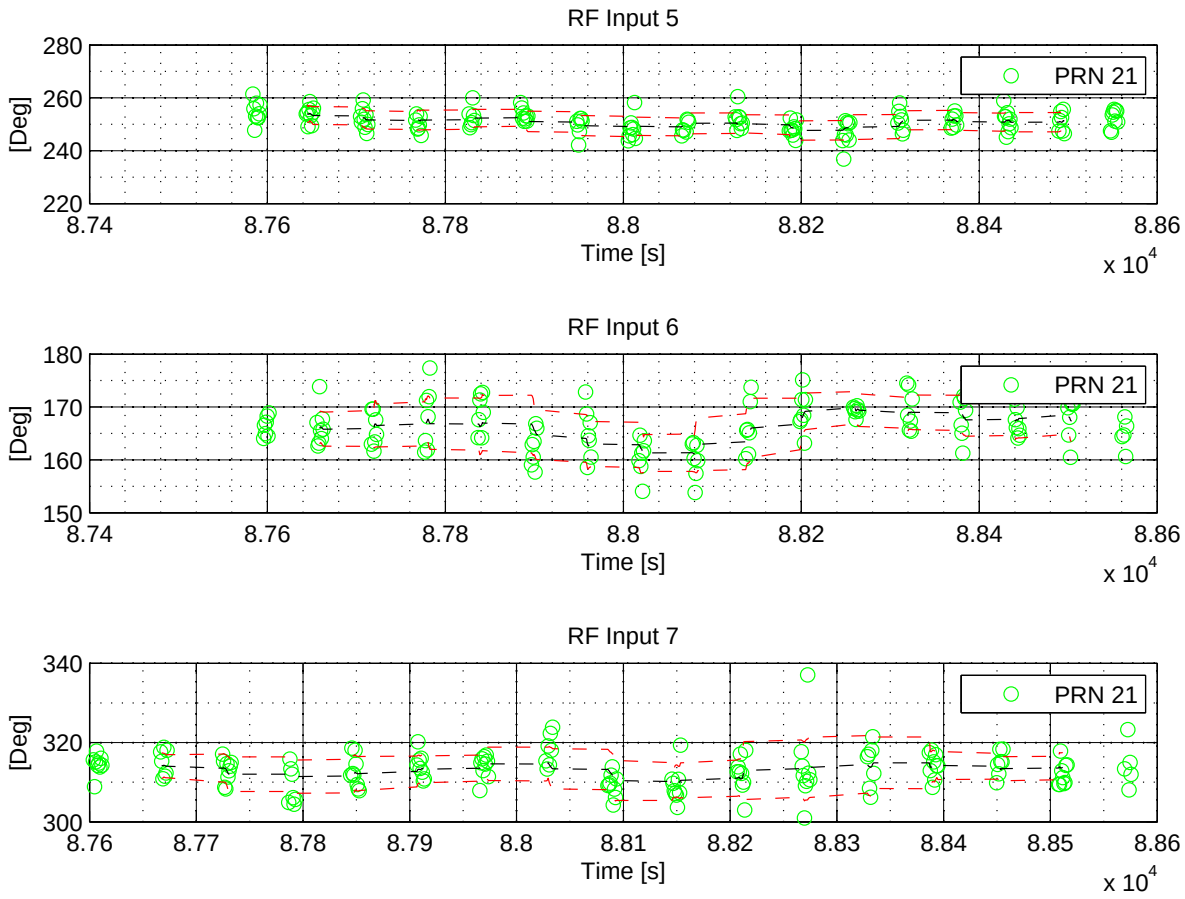


Figure 5.78: Phase Offsets for RF Inputs 5 - 7

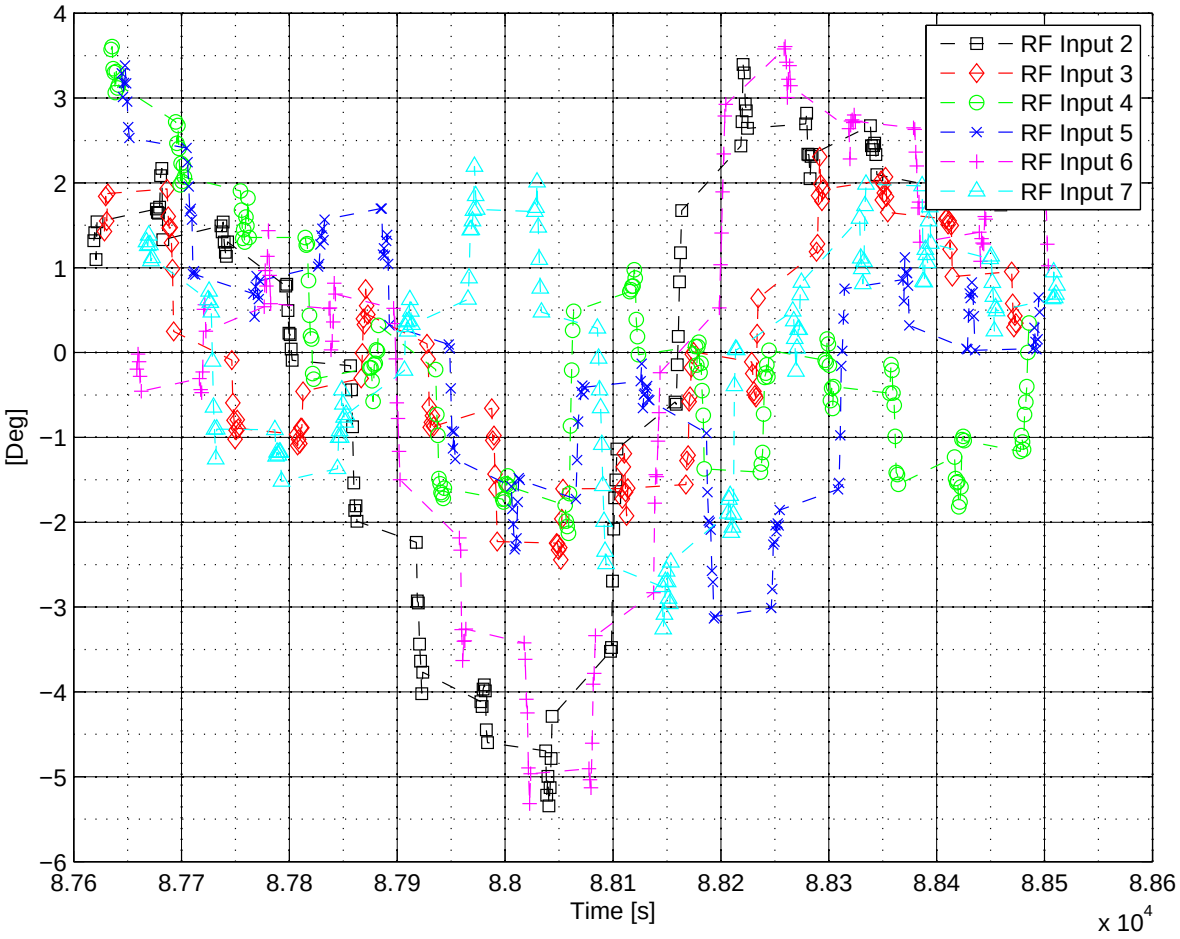


Figure 5.79: "De-biased" Phase Offsets for RF Inputs 2 - 7

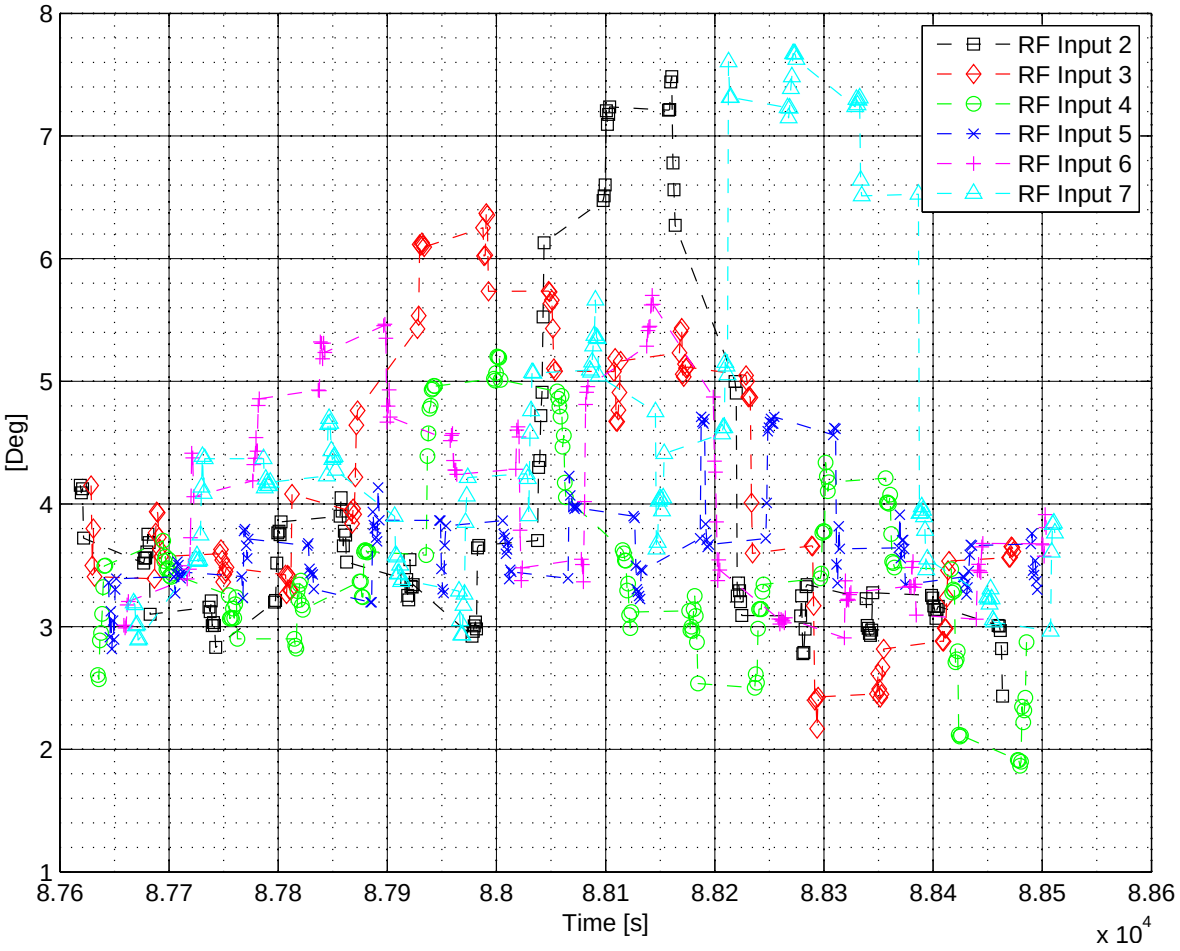


Figure 5.80: Phase Offset Samples Standard Deviation for RF Inputs 2 – 7

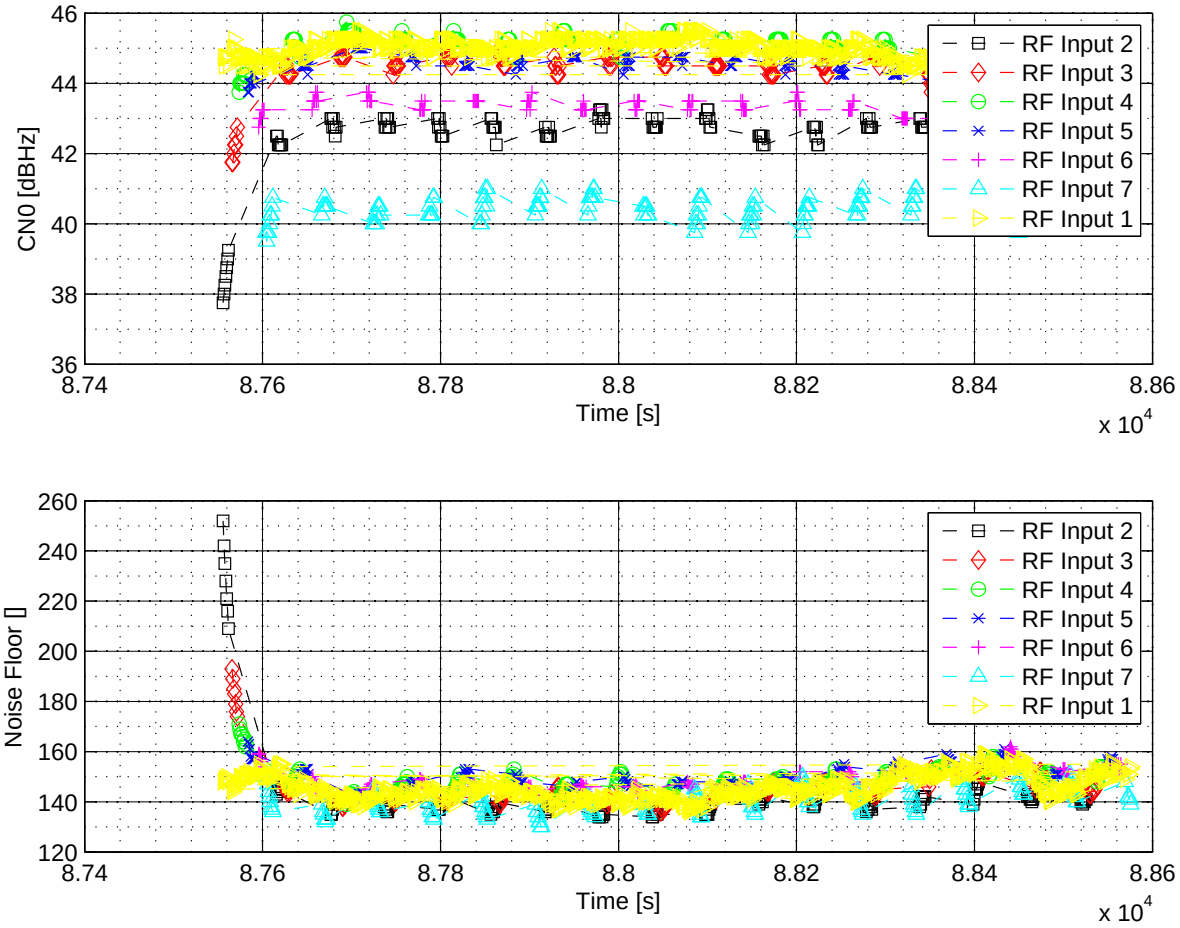


Figure 5.81: SNR and Noise Floor vs Time

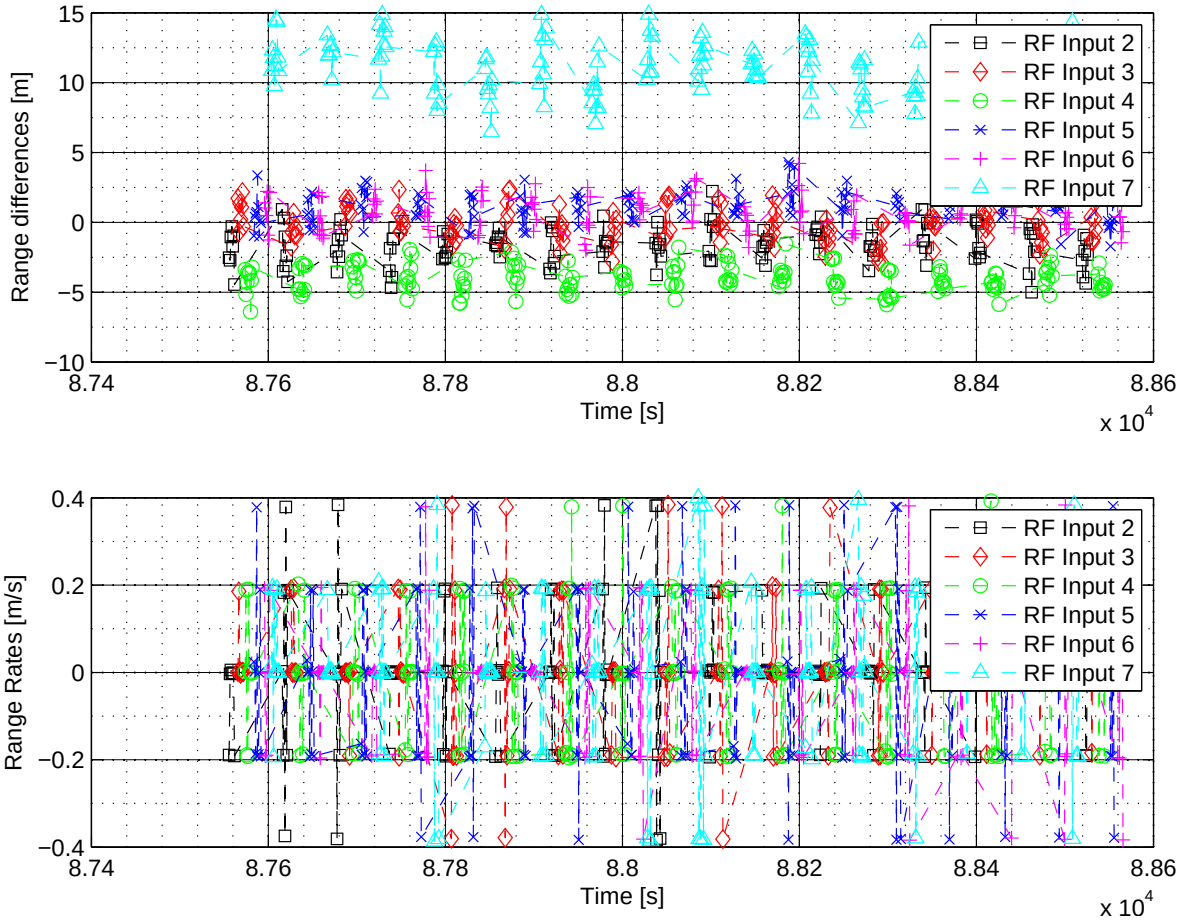


Figure 5.82: Pseudorange and Range Rate Difference

From the above results we can draw the following conclusions :

1. The RF Inputs phase offsets temporal variations are similar to the ones recorded in the previous tests.

2. The standard deviation also is similar.

3. The Antenna attached to RF input 7 exhibits lower CN0 and a not acceptable Range Delay : it should be checked if this is due to the RF circuitry or to destructive interference affecting antenna 7.

5.3.5 Nominal mode test without interfering signal

The test consists in the verification of the automatic live calibration and of the nominal mode without interference.

The test was performed in Lugano , the antenna array was placed over the car roof and aligned along the north direction. As long as the vehicle was static and the SY1031 SW does not implement a static filter the heading is not accurate and may drift with time, so the attitude was set using the message $PSPYN901 ATT section. The automatic calibration was performed before the beginning of the test. During the test a few problems have been noticed :

- the antenna array does not perform as expected. When the central antenna only is active, the maximum C/N0 is lower than 41 dBHz, i.e. at least 6 dB less w.r.t. the expected value and the highest elevation satellite, SV13, is tracked at 5 dBHz less w.r.t. the strongest tracked signal.
- the carrier phase is lost from time to time on all the channels at the same epoch.

Figure 5.83: Map of the Live Test location

The reason for this last problem was discovered after the test: the TCXO mounted on the FPGA board in fact is not shielded so any breeze was cooling the TCXO causing a too fast drift in the bias frequency.

Even with these problems it was possible to run the test and pass it successfully the at least from a functional point of view.

The test starts with a 20 minutes calibration session. The session actually lasted 45 minutes because of the above mentioned second problem. The automatic calibration ended reporting the following values :

- $PSPYN901,ACK,CAL,7F,131,113,268,310,118,340*27

These values are very close to the one found in Saphyrion premises using an antenna driven by the simulator a few days before and in a very different environmental condition.

- $PSPYN901,ACK,CAL,7F,139,131,313,306,157,315*2E

Figure 5.84: Test Setup at Saphyrion Premises using an antenna to re-irradiate the GPS simulator signal

The next test consisted in verifying the nominal mode. To activate the nominal mode the following commands sequence was used :

- $PSPYN901,ACK,SEL,NOM*0F
- $PSPYN901,ACK,ATT,0,0*74

and the results in terms of carrier to noise ratio levels for the satellites in view are shown in figure 5.85.

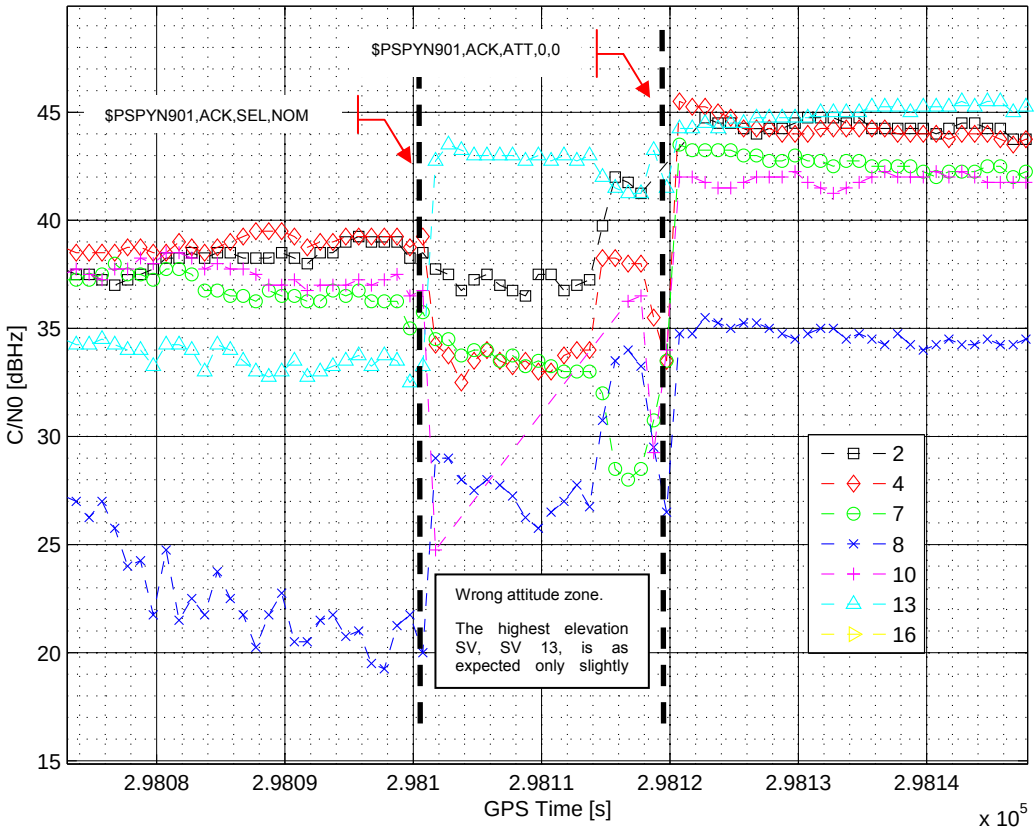


Figure 5.85: Results of the first test. In nominal mode, and with the right attitude, the CN0 is boosted at least by 5 dBHz.

The average increase in the signal to noise ratio is 6 dBHz , it ranges from 5 dBHz to 11 dBHz. The expected CN0 gain is between 8 and 9 dBHz for all the SVs above 30-40 deg provided all the RF paths have the same performance. The capability of the system to match the expected CN0 gain was verified during the above mentioned tests with the re-irradiated simulator signal at Saphyrion premises. In view of the poor performance of the antenna array the test is considered passed from a functional point of view.

The last test consists in a further verification of the attitude command : while in nominal mode the antenna array is rotated counterclockwise by 90 deg (i.e. the antenna array north axis now points in the west direction) after few seconds a new attitude command is issued to set the heading to 270 deg. The test passes and the results are shown in figure 5.86.

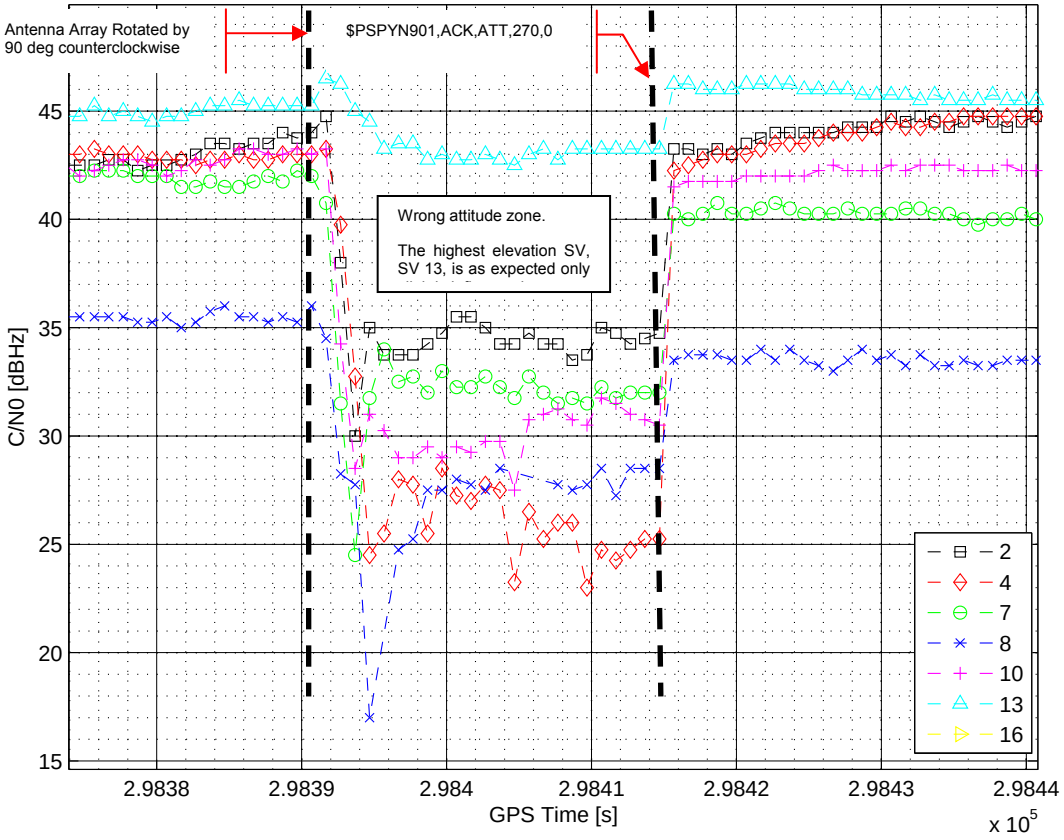


Figure 5.86: Results of the second test. While in nominal mode, after a counterclockwise rotation of 90 deg, the correct antenna pattern is restored setting the heading to 270 deg with a new attitude command.

Using the same test setup of the calibration test we' attempted to verify the performance of the interference rejection.

The Interfering signal was generated by FM modulation of L1 using a noisy signal and re-radiating the with an antenna patch. The antenna radiating the interfering signal was located at 2 meters and 90 degrees vertically with respect to the antenna array. The antenna radiating the GPS signal was placed at 2 meters and 40 degrees vertically and 270 ° azimuth with respect to the antenna array.

The test consists of:

1. In Nominal mode with fixed attitude, creating a pattern without taking into account interference and note the CN0: $ PSPYN901, SET, PH2, A, 40.270 * 00
2. A set at this point that the power of interfering CN0 calasse 10 dB.
3. Resetting the new antenna pattern taking into account the interference this time: $ PSPYN901, SET, INT, 1,90,0 * 00

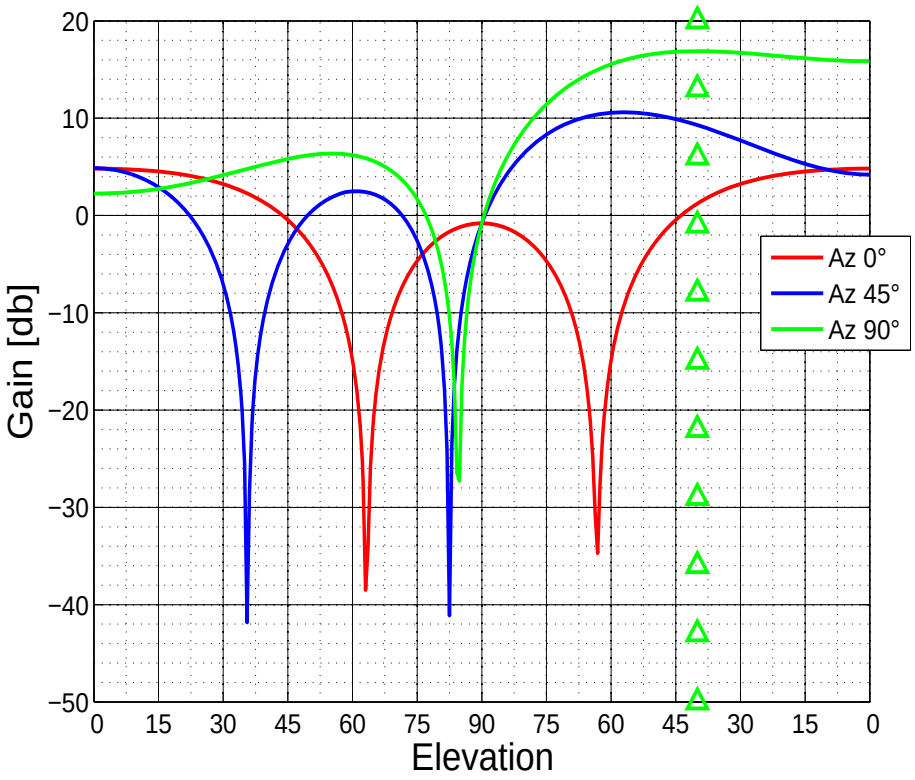


Figure 5.87: Synthetic antenna pattern without interference rejection

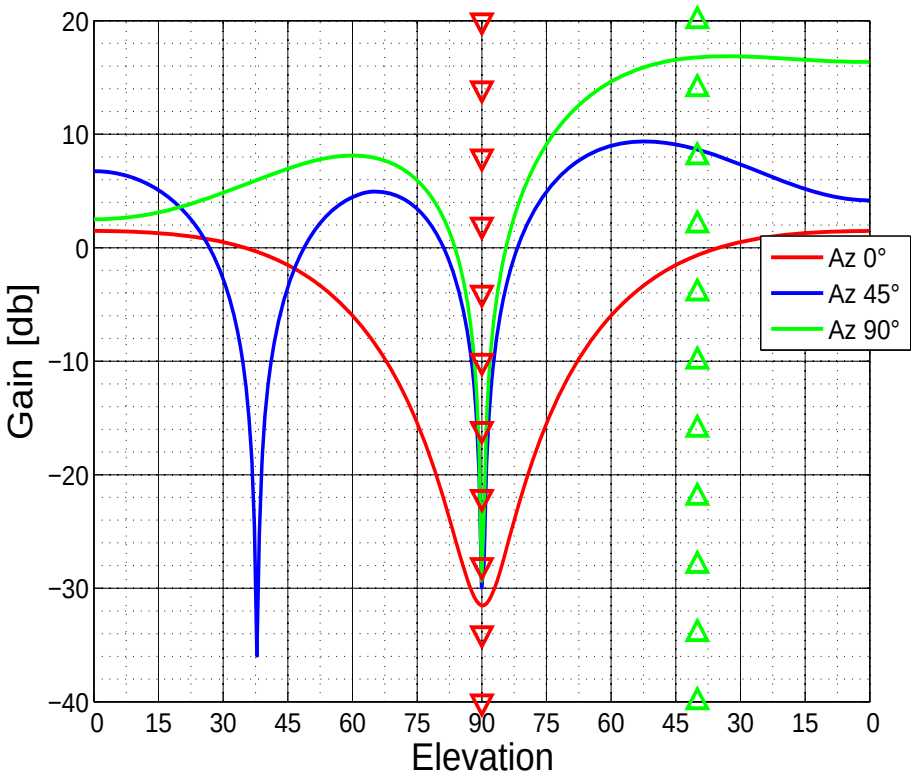


Figure 5.88: Synthetic antenna pattern with interference rejection

The test was not successful. In particular the CN0 did not improve after the interferer had been defined with the 901 INT command. One of the possible causes can be the mismatch between the antenna gains. At the time this test was executed the mismatch was still not taken into account in the computation model and in the calibration procedure. Actually the results of this test suggested the idea to model the RF gains mismatches in the beamforming computation and to measure them in the calibration procedure.

5.3.6 New Calibration – Live Test Using the Simulator

The following plots shows the results of the new calibration procedure at the simulator. In the new calibration procedure also the amplitude gain ratio between each RF input and RF input 1 is estimated.

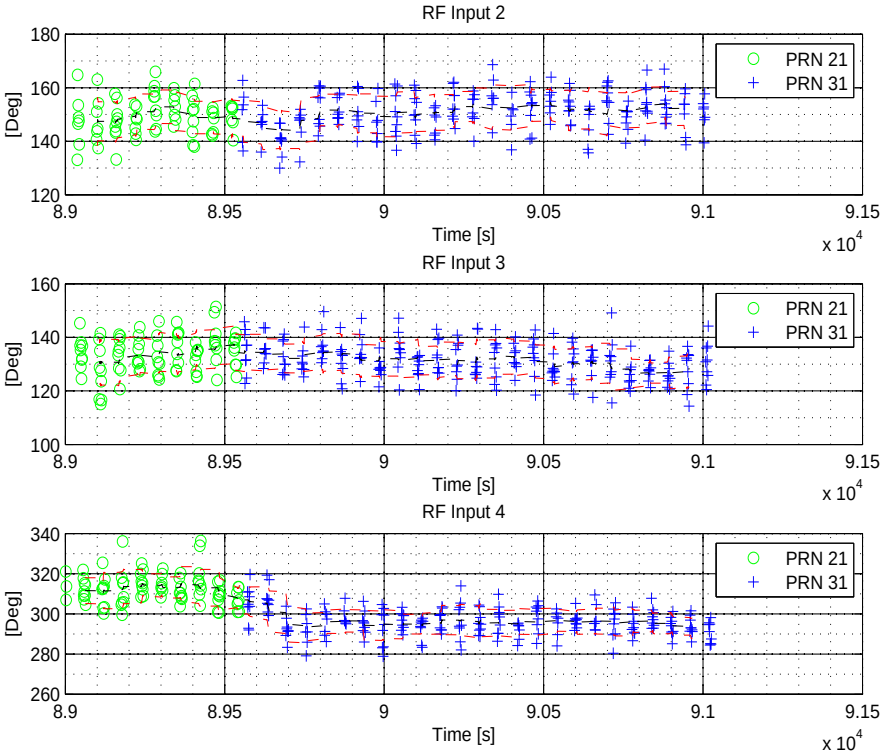


Figure 5.89: Phase Offsets for RF Inputs 2 - 4

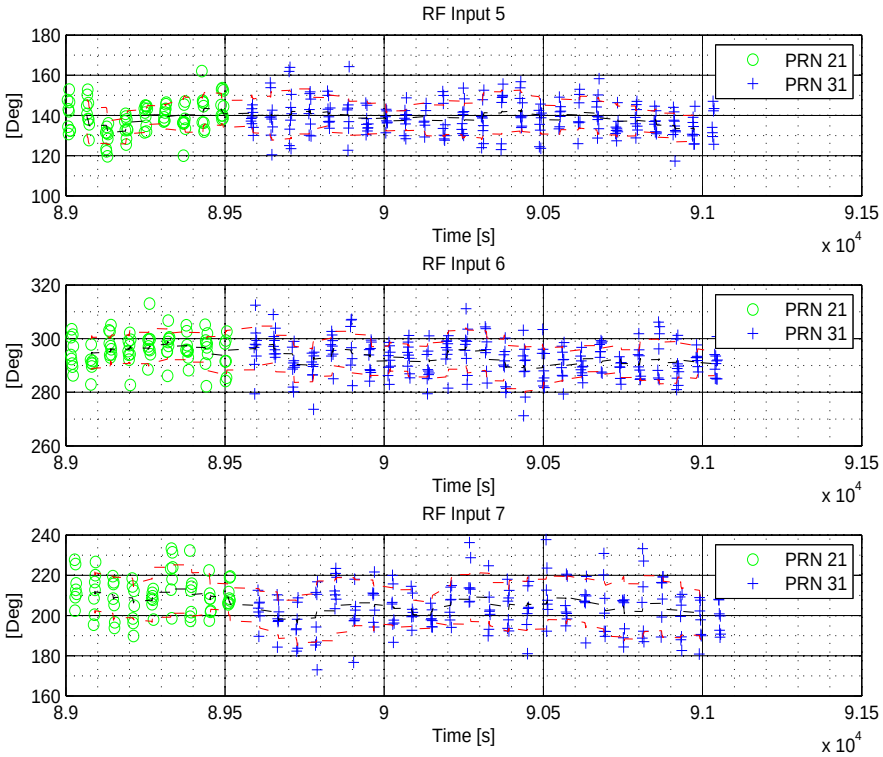


Figure 5.90: Phase Offsets for RF Inputs 5 – 7

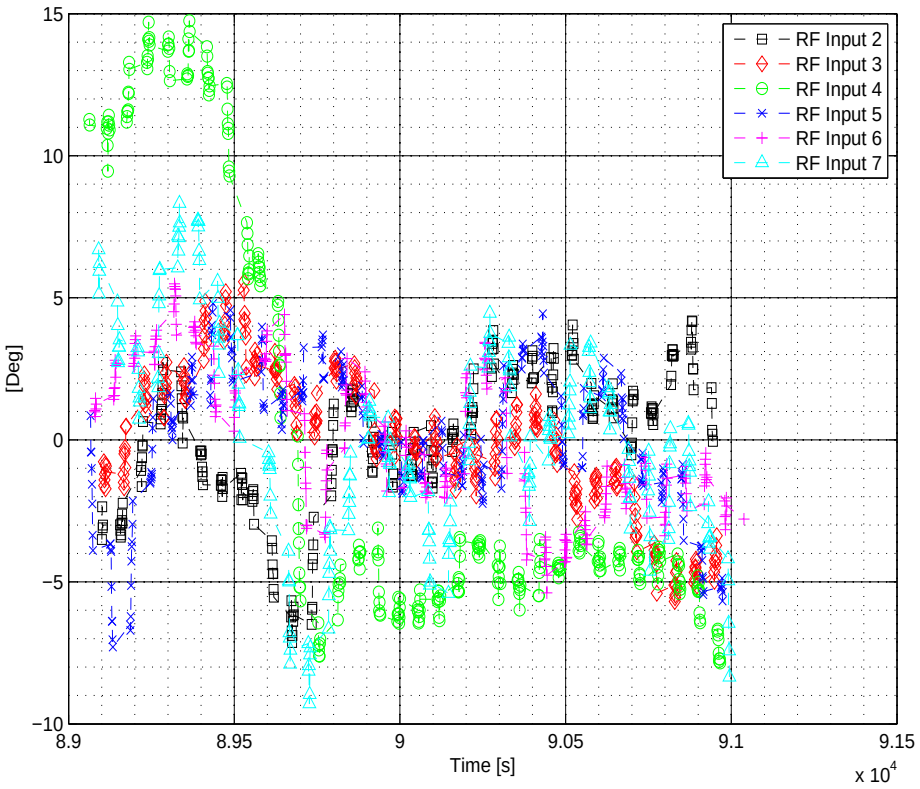


Figure 5.91: "De-biased" Phase Offsets for RF Inputs 2 – 7

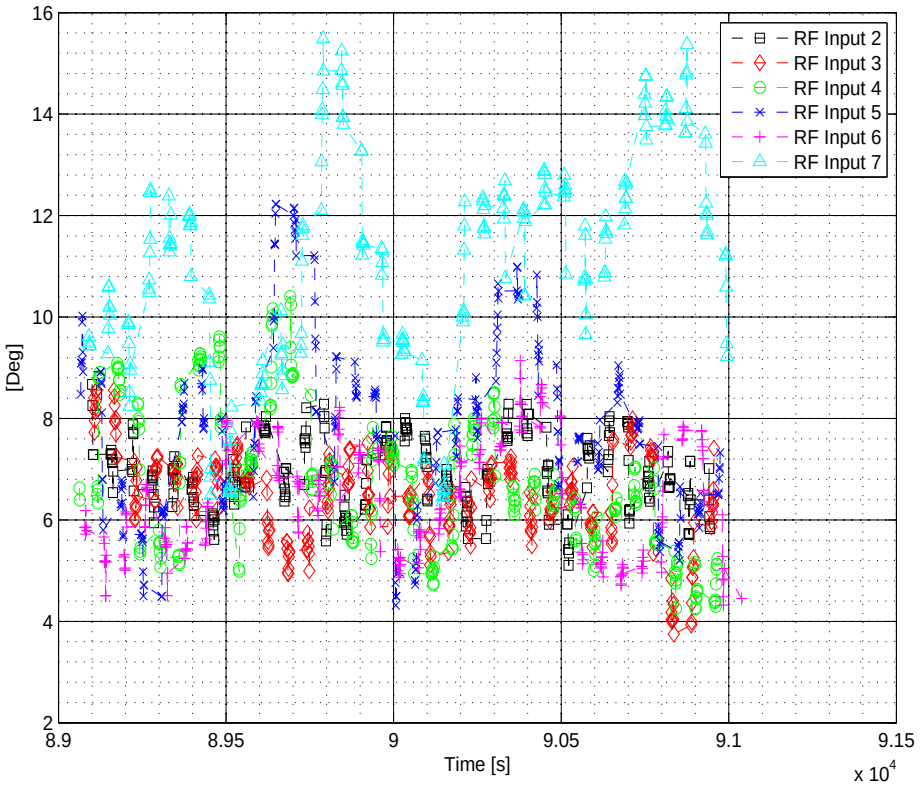


Figure 5.92: Phase Offset Samples Standard Deviation for RF Inputs 2 – 7

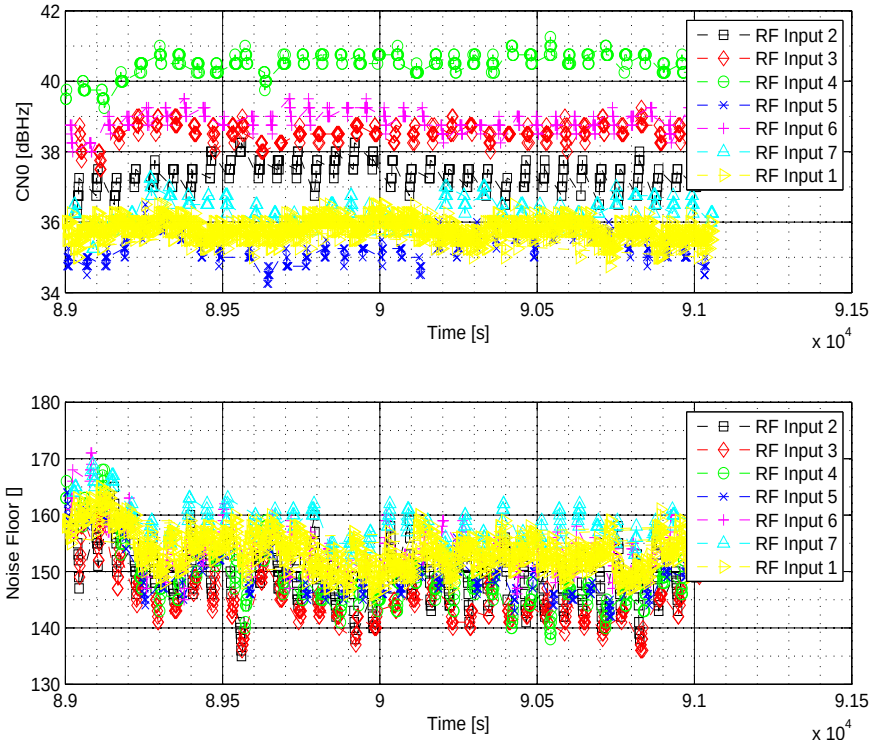


Figure 5.93: CN0 and Noise Floor V.S. Time

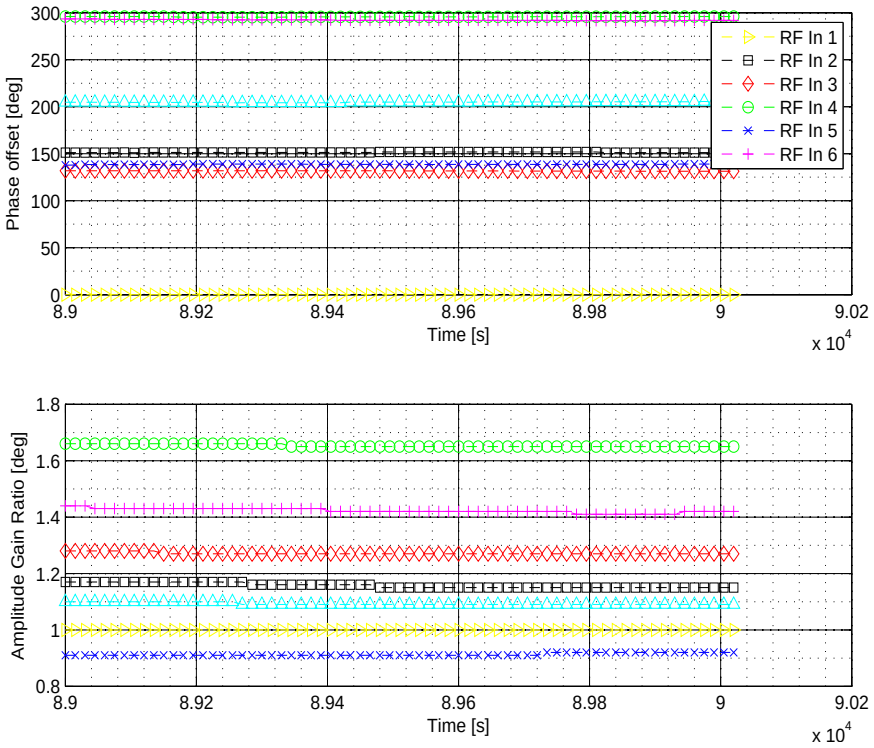


Figure 5.94: Estimated Calibration Parameters

The correctness of the calibration parameters can be proven by comparing figure 5.94 with figure 5.89 and figure 5.90 for the phase offsets and figure 5.93 for the amplitude gain ratios.

6 Beamforming Network in Hardware Integration: preliminary study

A first analysis concerning the integration of the beamforming network in the GRABEL GPS Base-Band Processor board has been taken into consideration.
The purpose of the accuracy improvement of the GRABEL receiver can be achieved reducing the antenna gain at low elevation angles (from which multipath signals and the strongest Radio Frequency Interferences generally arrive) and/or increasing the antenna gain in the direction of the satellites. So the beamforming technique has to be used having this aim but, in order to be really effective, it should be applied in a satellite-by-satellite way. In other words it is necessary to implement as many beamforming networks as the satellite signals (to be received) are.
So the idea is to integrate in the same baseband board a beamforming network for every tracking channel (16) that can work in parallel and independently.
This means that 16 independent radiation patterns can be generated using a single antenna array, i.e. one for every tracked satellite.
The down-converted signals from the antenna array are fed to the beamformers and each beamforming network is connected to the correlation engine as shown in figure 6.1.

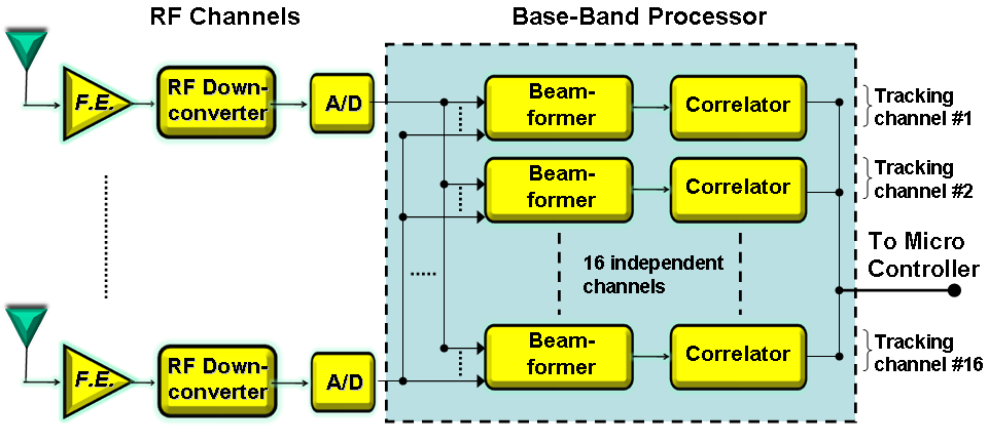


Figure 6.1: architecture of the GRABEL GPS Base-Band Processor

With this particular architecture it is therefore possible to have simultaneously:

- a single lobe towards the tracked satellite;
- a low radiation pattern at low elevation angles.

Since only a single lobe has to be generated by every beamformer, the complexity of the antenna array can be kept low.

However, as described in chapters 2.2.1 and 4.1, MVDR beamforming algorithm requires the handling of matrices, even if with a small number of elements; therefore it is recommended that all the operations with matrices need to be processed in a dedicated Beamforming Engine of the µProcessor. Anyway the multiplications of the signals by the beamformer coefficients and the final sum can be conveniently managed by the FPGA device as shown in figure 6.2.

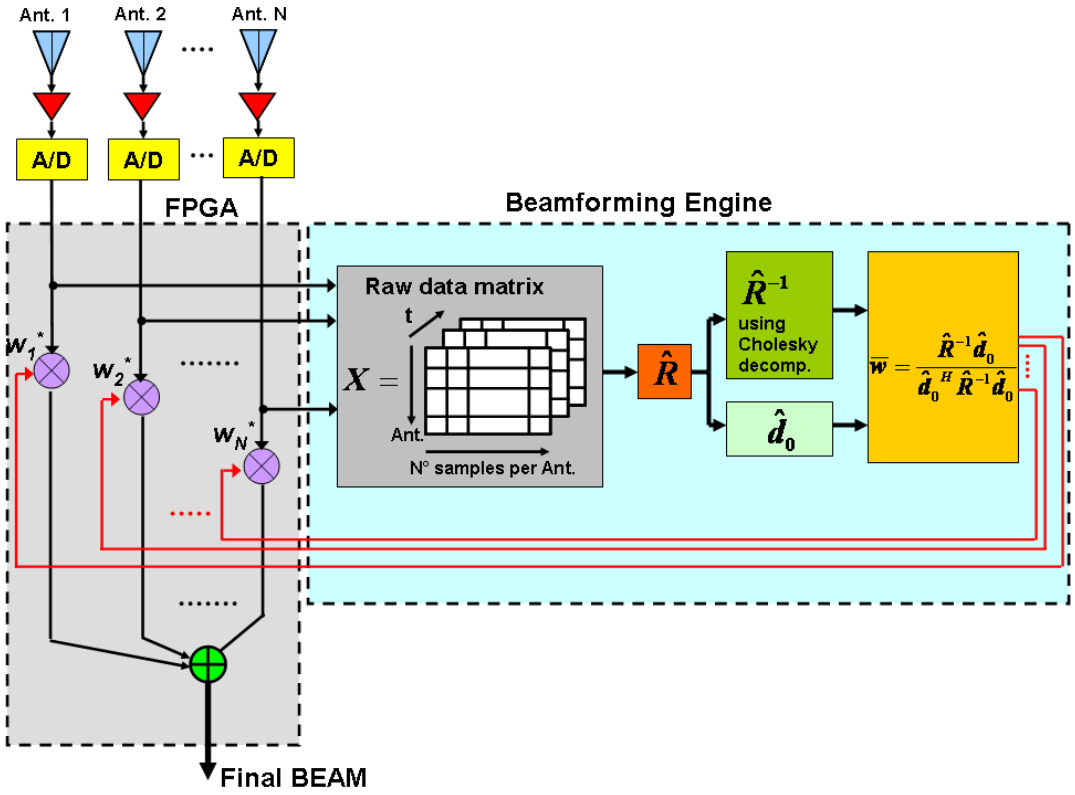


Figure 6.2: scheme of principle for the integration of beamforming network in hardware

Another important consideration about the implementation of MVDR algorithm in the GRABEL receiver has come out after a preliminary study.
Although theoretically the coefficients calculated by the Beamforming Engine have in general different amplitudes (and of course different phases), it has been considered the opportunity of maintaining all the amplitudes fixed. In fact this procedure seems to represent the best trade-off between performance and hardware complexity.

7 Planned developments

The next activities, scheduled to continue the work in the frame of Beamforming Algorithms for GRABEL, are summarized here below:

- Adapt the C code software for a 2-D array (the case of GRABEL). The array configurations to be considered are 2X2 (square), 3X3 (square), 6+1 (hexagon+center), 5+4 (cross).
- Definition of the best interfacing solution to the software.
- Development and integration of Beamformer system in the GRABEL receiver.
- Tests on field of the developed prototype and verification of the benefits introduced by the beamforming technique in comparison to the standard performances.

8 Summary/Conclusion

The beamformer output signal is a weighted linear combination of the data collected by the array antennas. The different weights determine the spatial filtering characteristics of the beamformer enabling the separation of signals with the same frequency but arriving from different directions. The weights in a data independent beamformer are selected in order to offer a fixed response independently from the received data. Statistically optimum beamformers operate the better selection of the weights in order to maximize the response of the beamformer in the direction of the desired signal and, if necessary, in order to minimize it in the direction of interfering signals taking into consideration the statistics of the data. Since sometimes the data statistics is unknown and changes with time, the adaptive algorithms are used to obtain weights that allow to converge to the statistically optimum solution. Beamforming represents a real, effective and versatile approach to spatial Filtering.

9 Reference Documents

Table 9.1: GRABEL reference documents

Ref.	Title	Doc.-ID	Version	Date
[RD1]	Definitive Description of Work	DoW_GRABELv12_00	12	21.04.2009
[RD2]	Minutes of KO meeting	Minutes_of_meeting_10_June_2009		10.06.2009
[RD3]	Minutes of Meeting	GRABEL_20100108_Minutes_of_Meeting		08.01.2010
[RD4]	Minutes of KP Meeting 1	GRABEL_20100224_Minutes_of_Meeting		24.02.2010
[RD5]	Minutes of MTR Meeting	20100604_GRABEL_Minutes_of_Meeting_MTR_Bologna		04.06.2010

10 Bibliography

Table 10.1: Bibliography

[Ref.Y]	Complete reference
[A1]	SY1031 Adaptive Multi-Beamformer Requirements & Specification Document
[Sah07]	M. Sahmoudi and M. G. Amin, “Optimal Robust Beamforming For Interference And Multipath Mitigation In GNSS Arrays”, Wireless Communications and Positioning Lab Center for Advanced Communications Villanova University, Villanova, PA 19085, USA, IEEE ICASSP 2007
[Chu08]	Yi Chu and Wei-Yau Horng, “ A Robust Algorithm for Adaptive Interference Cancellation”, IEEE Transactions On Antennas And Propagation, Vol. 56, No. 7, July 2008
[Ham98]	G. Hampson, M. Goris, A. Joseph and F. Smits, “The adaptive antenna demonstrator”, Published in 1998 IEEE Digital Signal Processing Workshop, Bryce Canyon, Utah, USA
[Lor05]	R. Lorenz and S. Boyd, “Robust Minimum Variance Beamforming”, Information Systems Laboratory, Stanford University, Stanford, CA 94305-9510, IEEE Transactions On Signal Processing, Vol. 53, No. 5, May 2005
[Smo02]	B. Smolders and G. Hampson, “Deterministic RF Nulling in Phased Arrays for the Next Generation of Radio Telescopes”, IEEE Antenna’s and Propagation Magazine, Vol. 44, No. 4, August 2002
[Lee93]	Kuan-Min Lee, Ruey-Shi Chu, and Sien-Chang Liu , “A Built-In Performance-Monitoring/Fault Isolation and Correction (PM/FIC) System for Active Phased-Array Antennas”, IEEE Transactions On Antennas And Propagation, Vol. 41, No. 11, November 1993

- End of document -