Correlation Structures and Prediction Models for Radar Signal Values of Sea Waves

Hans Krause

Department of Mathematics

Utrecht University

Supervisor:	DRS. EMIEL STOLP, THALES COMPANY, HENGELO
Co-supervisor:	DR. CRISTIAN SPITONI, UTRECHT UNIVERSITY
Second reader:	DR. MARTIN BOOTSMA, UTRECHT UNIVERSITY

A thesis submitted for the degree of

Mathematical Sciences

August 24, 2015

Abstract

A naval radar, though designed to detect objects or targets, also receives backscatter from wave peaks. Better understanding of the signals generated by the background waves can improve the detection of small targets. This thesis examines the correlation structure of the sea waves along the dimensions Doppler, range, azimuth and scan. The targets are mathematically simulated, as are the sample waves used alongside the measured waves. Using two prediction models – Linear Prediction and Sequential Decorrelation – examination of each cell illustrates the difference between predicted and measured signals in a plot of residual signals. High values in the resulting plot indicate target presence. Further suppressing the wave signals enhances anomalies caused by target signals via discrimination and calibration techniques. The findings matter to those wanting clearer target detection on a wave-covered sea surface situation.

Preface

The foundation of this master's thesis was an internship at Thales company in Hengelo. A separate paper focusing more heavily on the physics aspects of the problem was delivered and presented in December 2014. The next semester was spent deepening the mathematical aspects of the project, culminating in this master thesis, and marking the end of my MSc program "Mathematical Sciences" at Utrecht University.

Hans Krause
Utrecht, August 24, 2015

Contents

Symbols and Units Used

Symbol	Measure	Unit (abbr.)
A	Amplitude	-
A_{acc}	Accuracy	-
A_{score}/A_{error}	Accuracy score/error	-
AUC	Area under curve	-
B	Beamwidth	rad (degrees if specified)
c	Speed of light	$(3 \cdot 10^8)$ m/s
D	Doppler	bins
D_H	Horiz. diameter of radar antenna	m
F	Frequency	Hz
FN	False negatives	-
FP	False positives	-
fp_{rate}	False positive rate	-
g	Gravity of Earth	(9.81) m/s^2
h	Water depth	m
k	Wave number	rad/m
k_x	Wave number in x direction	rad/m
k_y	Wave number in y direction	rad/m
NM	Distance (of max. radar range)	Nautical miles
PRF	Pulse repetition frequency	Hz
PRT	Pulse repetition time	s
R	Range	m
s	Standard deviation	-
S	Scan	-
S_{dB}	Signal strength	dB
t	Time	s
T	Wave period	s
T_{rot}	Radar's rotation time	s
TN	True negatives	-
TP	True positives	-
tp_{rate}	True positive rate	-
W	Forecast probability variable	-
V	Velocity	m/s
V_R	Range velocity	m/s
V_θ	Angular velocity	rad/s
V_ϕ	Wave phase velocity	rad/s
α	Angle	rad (degrees if specified)
θ	Azimuth	rad (degrees if specified)
λ, L (diag.)	Wavelength	m
π	Actual outcome variable	-
ϕ	Angle modulation	-
ω	Angular frequency	rad/s
ω_c	Centre frequency	Hz
$\perp$	"is independent from"	-

Chapter 1

Introduction

In this thesis, we are dealing and working with radar images of the sea. The radar picks up signal values for a two-dimensional area of the sea surface during one rotation. It makes multiple rotations giving an indication of the change in the signals over time. Objects that have considerable electrical conductivity, including breaking waves, reflect or backscatter radar signals especially well, generating higher echo signals than background seawater. Thus littering the radar plot are sea waves that appear as high signal line segments. In Figure 1.1 (via the program in Appendix B.1) is a typical scan, the standard reference radar plot in this thesis, showing a sector of the sea:

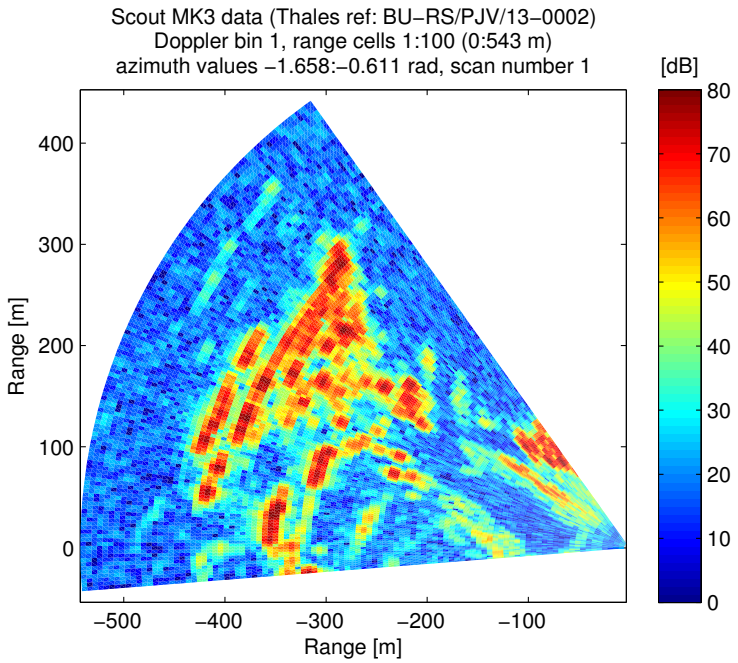


Figure 1.1: Standard reference radar plot.

The aim of this thesis is to find correlation structures of the sea waves, that is to search for their behavioural patterns. We shall recruit two different prediction models for this. They will aid in the practical goal of being able to detect small objects, or targets, among the sea waves – "small" meaning

that their signals are not stronger than those of the background waves. For this thesis, targets are simulated – injected by adding signals to the wave data. By filtering out the sea clutter, i.e. unwanted waves, via a correlation structure, the simulated targets become exposed in a wave-free setting. In summary, the anticipated result of this research is to detect small targets concealed among strong background wave signals or to disentangle the target signal from the noise.

Figure 1.2: White water peaks.

Pictured in Figure 1.2 is an example photographic view from the on-shore radar's perspective of approaching waves with their foaming, or white water, crests or peaks discernible. These correspond to the red high signal line segments on the previous radar plot, whereas the low background signals are plotted in shades of blue.

1.1 Outline of the Thesis

In the current Introduction Chapter 1, the thesis commences with the necessary explanation of the background physics, as certain terms and principles may be unclear to mathematician readers. These include the basic principle of how a radar works, the origin of our logged data (i.e. recorded data, see Appendix A), the physical problem, and a summary of the mathematical theory that will then attempt to tackle it. Upcoming Theory Chapter 2 then lays the mathematical foundations and theory. These include the construction of the simulated waves – an optimal testing environment before moving to the less-perfect logged waves – and of the simulated targets, the prediction models, algorithms and accuracy of measures. Two prediction models – Linear Prediction and Sequential Decorrelation – will be implemented. Next, in Practical Application Chapter 3, theory is put into practice, and the procedure is performed in both wave environments. Finally, Conclusion Chapter 4 then completes by summarising the findings and results. Further material includes a summary of precautions and limitations for practical use.

1.2 Basics of Radar Physics

Before delving into the mathematical aspect of the thesis, this section provides some background knowledge and description of radar physics.

A photograph of the naval radar and its setup is shown in Figure 1.3 (for technical characteristics see Appendix A):

Figure 1.3: Photograph of Scout MK3 radar.

Radars rotate around their vertical axis at a given rate of 1-10 s per rotation – our particular type has a rotation time of 1.5 s. Viewed from above, the radar's rotational direction is clockwise which dictates the angle measurement – the so-called azimuth as the angle starting from North, moving clockwise. Whilst rotating, the radar sends out a signal with a wavelength λ of $\sim$3 cm at a given repetition frequency (the *PRF* or pulse repetition frequency) – typically in the order of 1 kHz equivalent to 1000 per second. The pulses are then 1 ms apart (the *PRT* or pulse repetition time). The signal hits a target at range R and time t and backscatters then with a total travel time $2t$ back to its origin as shown in Figure 1.4 from [1] below:

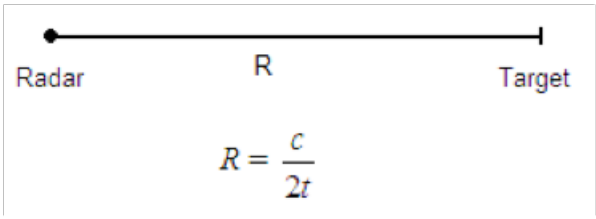


Figure 1.4: Signal travel path.

This is part of the information required to graph a radar plot – the distance from the radar. The

maximum range of the radar varies, depending on type, and can reach several nautical miles (see Appendix A). Note that the received echo signal for targets declines as the fourth power of the range, so received power from distant targets is disproportionally small.

As the radar rotates, it obtains a signal for each burst – an angular section containing a group of radar pulses. As we pass over a backscattering target, we see how the received signal evolves. The first one is weak, as the target is in the flank of the beam. The strength increases as the radar rotates further until the centre of the beam is pointing at the true azimuth of the target, after which it decreases again. This forms a rough bell curve for the signal strengths along the bursts as shown in Figure 1.5:

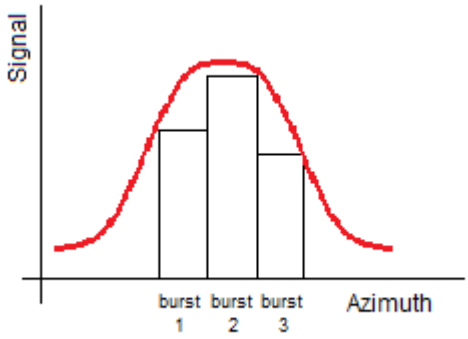


Figure 1.5: Bell curve for signal strengths of several bursts.

Note that the familiar Gaussian profile will later be used as an estimate for simulated targets (cf. Section 2.1.2), mimicking the above.

The signals in the logged data, or "loggings", are in complex amplitude form. This is because the radar receiver contains a digital Hilbert filter [25], which converts the real received signal of the form $A \cdot \cos(\omega_c t + \phi(t))$ into a complex one $A \cdot e^{i(\omega_c t + \phi(t))}$, where ω_c is the center frequency, and $A(t)$ and $\phi(t)$ are the amplitude and angle modulation respectively [16], as shown in Figure 1.6 below:

Figure 1.6: Hilbert filter.

The real part of the signal is the In Phase component (I-signal), whereas the imaginary part is the Quadrature component (Q-signal). We convert the complex signal amplitude A into S_{dB} – where dB (decibel) is the logarithmic unit expression for the ratio of power P. This is done by the relation [24]:

$$P = |A|^2,$$
$$S_{dB} = 10 \cdot \log_{10}(P).$$

The reason for this conversion is two-fold. On the one hand it gives indication of the absolute – non-complex – signal strength. On the other hand the amplitude can vary over multiple orders of magnitude, so compressing this non-linearly is more convenient. Higher signals tend to be in the region of 40-80 dB corresponding to an amplitude of 100 to 10 000. Lower signals containing weak sea waves and background signals are normally less than 20 dB, equivalent to amplitude 10. The radar plots in this thesis include a colour scale indicating the dB signal strength for each cell – the smallest unit of a two-dimensional area of the radar plot.

A further aspect of the radar is that it operates on a given beamwidth. The relation of beamwidth B (where λ is the signal wavelength, and D_H is the horizontal diameter of the radar antenna):

$$B = \frac{\lambda}{D_H} = \frac{0.03}{1.5} = 0.02 \text{ rad (or } 1.15^\circ).$$

This is at 3 dB width, which is the width at half-power or $\sim 0.71A$ of the one-way beam. The beam pattern is shown in Figure 1.7 from [23]:

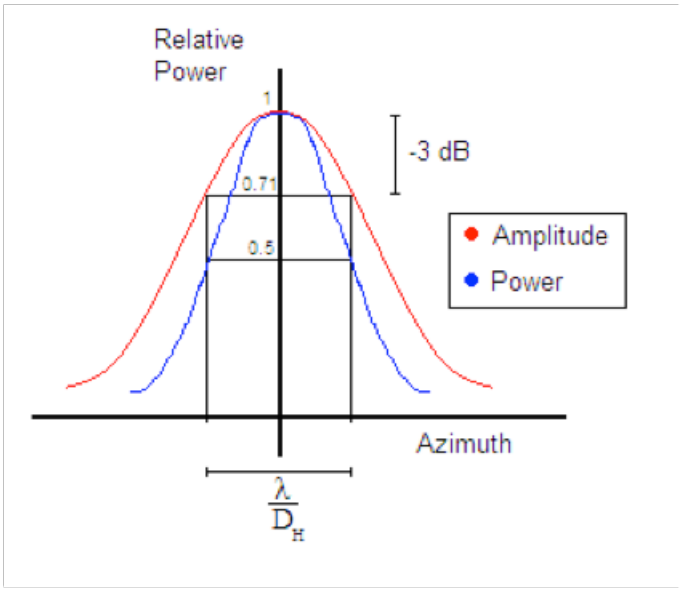


Figure 1.7: Beamwidth at 3 dB.

The radar wavelength $\lambda = 3$ cm has the repercussion of the smallest detectable targets being in the few centimetre range when using the current radar setup. However, this is not our cell resolution yet, which is to be stated in Section 1.4.

A further characteristic of our radar is the use of three *PRF*s staggered from scan to scan: 3.6, 4.0 and 4.4 kHz in our case. Bursts consisting of 8 sweeps are transmitted – though they are processed as bursts of 16 sweeps with an overlap of 8 sweeps. The FIR (finite impulse response) filter within the radar then interpolates the 16 sweeps to 24 "bins". Figure 1.8 provides a visualisation of this, whereas

the pulses, bursts (groups of pulses or Doppler bins) and rotation movement can be summarised in Figure 1.9:

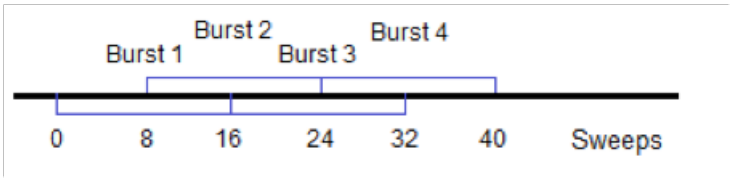


Figure 1.8: Overlapping burst widths.

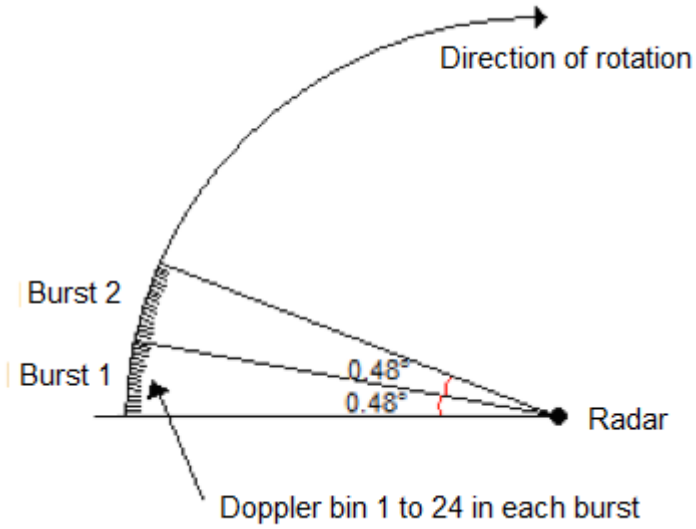


Figure 1.9: Burst Doppler bins.

Our radar's rotation time 1.5 s and average frequency 4.0 kHz implies 240°/s, corresponding to 0.02° per bin after FIR filter interpolation. With 24 bins per burst, this would mean a burst width of 0.48°. The knowledge of this burst width provides further required information to create a radar plot – allocating the azimuth bursts to their respective specific azimuth angles.

The arrangement of groups of pulses or bins in each burst serves to distinguish the various velocities that a target located in the burst can have. Moving targets cause a frequency shift in the echo signal – the Doppler effect explained in next section. This frequency shift is detected by applying an FFT (fast Fourier transform) on the received signal. Figure 1.10 shows a simplified example of this procedure:

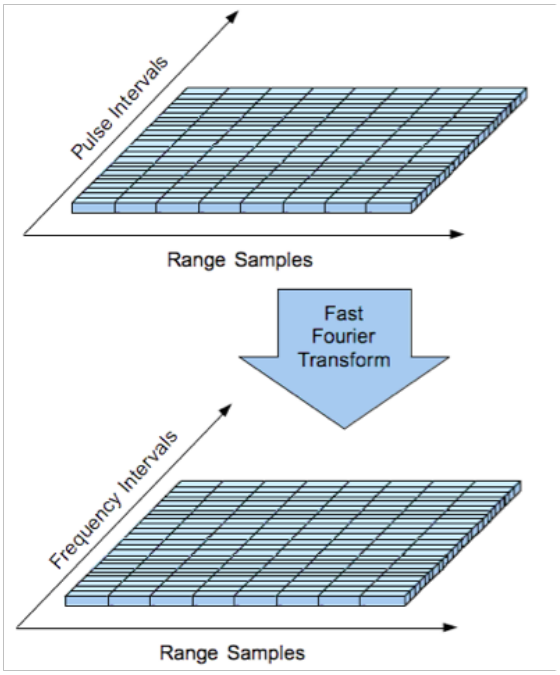


Figure 1.10: Fast Fourier transform.

The signals of the pulse intervals within each burst are processed together in the FFT, yielding the respective frequency intervals. The magnitudes of these are discussed forthwith.

1.3 Doppler Dimension

The Doppler Effect is the shift in frequency of a wave train for an outside observer. Figure 1.11 provides a two-dimensional visualisation of this phenomenon:

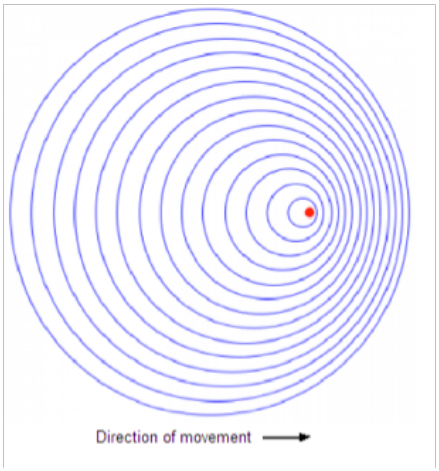


Figure 1.11: Doppler Effect visualisation.

Above the circles show the waves emitted by a moving red source at successive moments in time. From it, we see that the frequency of waves ahead of the source is higher than that of the ones behind. A familiar example of this: the siren of an emergency vehicle moving towards a stationary bystander will have a higher frequency (pitch) than when stationary, and a distancing vehicle siren a lower frequency.

Note that only targets moving radially with respect to the radar will cause a phase shift in the received radar signal, giving the next radar pulse a slightly different run time. A target moving tangentially to the radar, however, has an unchanged distance to the radar origin and will not be detected in this manner. Both the ability to determine a target's radial velocity and to discriminate between moving targets and sea clutter give the Doppler Effect its significant role in radar applications [2].

The number of pulses in each burst dictates the Doppler dimension size – 24 pulses here correspond to 24 Doppler bins. The first half of these indicates approaching targets with their speeds in ascending order. The latter 12 bins pertain to distancing targets with speeds in descending order.

The relation of the frequency F_{bin} of one Doppler bin reveals the precise velocity V_{bin} of the bin step size, also known as the Doppler resolution:

$$F_{bin} = \frac{PRF}{24},$$

and:

$$F_{bin} = \frac{2V_{bin}}{\lambda}.$$

So:

$$\frac{PRF}{24} = \frac{2V_{bin}}{\lambda}$$

$$V_{bin} = \frac{\lambda \cdot PRF}{48}. \tag{1.1}$$

In the case of our radar, the PRF is on average 4.0 kHz with a wavelength λ of 0.03 m. The step size for each bin is therefore a velocity of 2.5 m/s. Hence, the Doppler bins correspond to the velocities given in Table 1.1 below:

Doppler bin	Velocity (m/s)
1	0
2	-2.5
3	-5
4	-7.5
5	-10
6	-12.5
7	-15
8	-17.5
9	-20
10	-22.5
11	-25
12	**-27.5**
13	**30**
14	27.5
15	25
16	22.5
17	20
18	17.5
19	15
20	12.5
21	10
22	7.5
23	5
24	2.5

Table 1.1: Velocities for each Doppler bin.

Note that negative velocities signify targets moving towards the radar, and positive ones away from it (notice the large discrepancy between bins 12 and 13, highlighted in bold in the above table). This is convention throughout this thesis.

Should a target accelerate beyond the maximum approaching velocity -27.5 m/s, then it would enter the next Doppler bin 13. Further increases in speed correspond to higher bins in the same step sizes – including from bin 24, which precedes bin 1. Thus, we have a cycle where an accelerating target goes through all Doppler bins in a periodic manner. A target showing a high signal for bin 12 could therefore equally be travelling at range velocities -27.5 m/s, -87.5 m/s, -147.5 m/s, etc.

Our loggings deal with sea waves only (cf. Section 3.2.3), and in this coastal region, waves certainly do not race faster than $\sim$100 km/h (27.5 m/s). There can therefore be no confusion as to the Doppler bin's corresponding wave velocity here.

A logged data example containing the signal values is shown in Figure 1.12, as similarly created in [22]. The high signal cells mark the corresponding Doppler and range of backscattering waves.

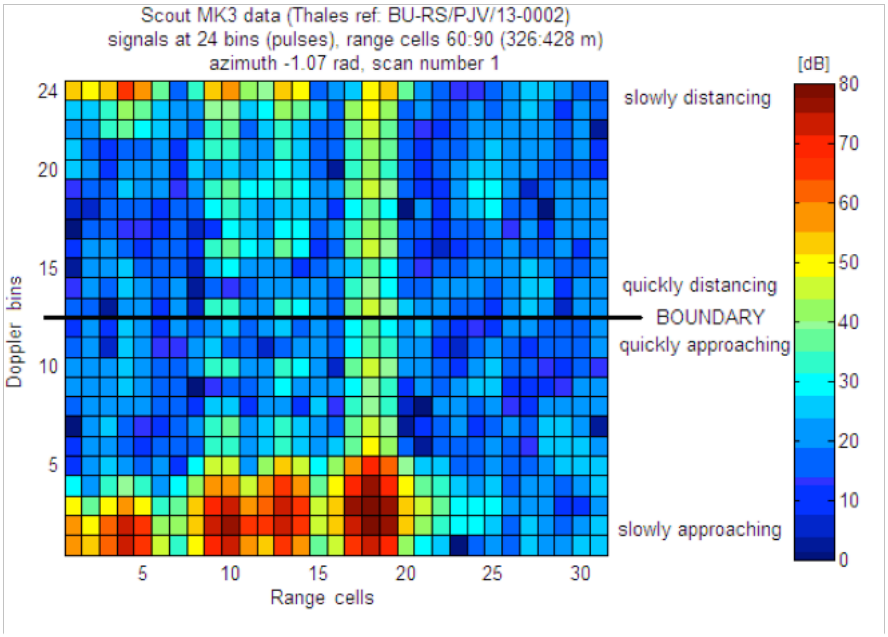


Figure 1.12: Range-Doppler bin graph.

The above contains the data as logged in original form. However, we can adjust this to a more discernible plot by rearranging the bins – individually inverting the lower and upper half of the bins. Figure 1.13 shows this updated order on the y-axis:

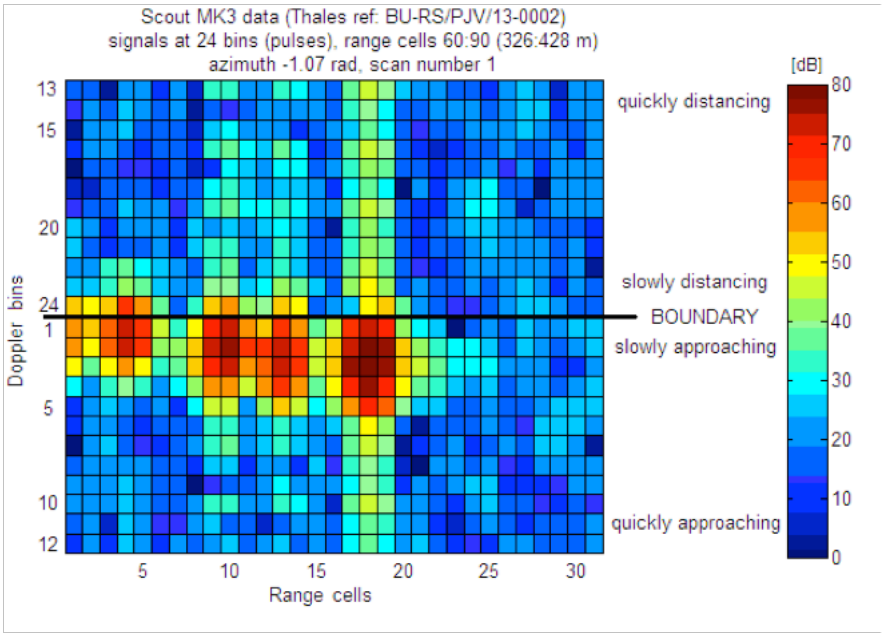


Figure 1.13: Range-Doppler bin graph with bins rearranged.

Now that the gradual signal strength transitions across the speed-zero (boundary) are clearer, the

plot yields groups of high signal cells congregated in the first few Doppler bins and at intervals of $\sim$6 range cells. In fact, this plot shows the velocity of the backscattering wave crests. From the table, Doppler bin 2 to 2.5 corresponds to a plausible wave velocity of -2.5 m/s to -3.75 m/s. Furthermore, the range cell intervals mark the distance between wave crests – the wavelength of the sea wave.

1.4 Basic Mathematical Task

The data contains a complex signal for each specific cell, which corresponds to a specific Doppler bin, area (range by azimuth) and scan. We place the data into a four-dimensional array, dimensions being:

Doppler:	bin of the velocity the target is moving at. Here 24 bins;
Range:	the index/cell distance from the radar. The individual cell-length is usually in the order of a few metres. Here 5.428 m;
Azimuth:	index/cell of the angle starting from North going clockwise. The index-width is equal to the burst width. Here $\sim 0.48^\circ$ after interpolation (cf. Section 3.2.4);
Scan:	the n^{th} picture received.

Throughout this thesis and in associated programs, dimensions are strictly in this order: Doppler, range, azimuth, scan. In addition, all radar plots have azimuth zero conforming to the aforementioned North as the true orientation.

Once a four-dimensional array of sea wave signals is constructed (See Appendix B.2), the task is to find correlation structures between the dB signals of this data set. We also calculate correlation coefficients matrices in order to obtain useful wave parameters.

Subsequently, we calculate the predictor cells and coefficients for cells of interest – multiple cells that then individually become "cells under test". Each is predicted via two prediction models or methods. These provide a correlation structure of the sea waves, helping to predict the cell of interest based on other cell signals in the wave data.

At this very stage, a simulated target is injected into the same sea wave environment. Via the prediction models, we predict each cell of interest in this new environment. Should on the one hand the predicted signal be too close to the true one, then the cell under test contains background sea wave signal. On the other hand a large difference between the two, the residual signal, gives notable indication that there is a target in the cell under test. Targets can subsequently be highlighted in their appropriate positions on a plot, thus filtering out the waves, or "de-cluttering" the sea, and effectively revealing target locations or positions.

The task procedure in summary form:

1. place raw data in *ArrayData* array, *AzimuthValues* matrix and *ScanCheckVector* vector, and subsequently convert the complex signal into S_{dB} form;

2. calculate correlation coefficients to obtain wave parameters;

3. in a sea wave setting, find predictor cells and coefficients for all cells of interest X via: Linear Prediction and Sequential Decorrelation;

4. in the same setting with injected targets introduced, use the predictor cells and coefficients to expose target locations based on larger residual signals.

Chapter 2

Theory associated with Prediction Models

2.1 Simulated Waves and the Simulated Moving Target

The search procedure described above is carried out in two sea wave environments: simulated and logged. What follows is the mathematical construction of the former.

2.1.1 Simulated Waves

These waves are designed to be of a simple standard but simultaneously non-perfect – with an intentional element of randomness which is found in real waves [18].

The general formula for the relation of waves, for x-coordinate x, y-coordinate y and time t:

$$S_{dB}(x,y,t) = A \cdot \cos(k_x x + k_y y - \omega t) + C,$$

or in vector form with 2-dimensional vectors $\vec{k}$ and $\vec{x}$:

$$S_{dB}(x,y,t) = A \cdot \cos(\vec{k}\vec{x} - \omega t) + C,$$

for amplitude A, wave number in the x direction k_x [rad/m], wave number in the y direction k_y [rad/m], angular frequency of the wave ω [rad/s] and constant C. Further parameters are the wave number $|k| = \frac{2\pi}{\lambda}$ [rad/m] and wave phase velocity $V_\phi = \frac{\omega}{|k|}$ [m/s]. The basis or justification for this simulation model lies in the Gerstner waves which are a good approximation [28].

In the simulated waves example plotted above, named *Simdata1*, k_x=-0.5 and k_y=0 mean that the waves are moving westwards. Further setting A to 40, and C to 40, yields the waves going from 0 to 80 dB.

Finding the relation between $\vec{x}$ and t to keep the wave phase constant is equivalent to finding $x(t)$ such that $\cos(\vec{k}\vec{x} - \omega t)$ remains constant. So setting the derivative to 0:

$$\vec{k}\delta\vec{x} - \omega\delta t = 0,$$

$$V = \frac{\delta\vec{x}}{\delta t} = \frac{\omega}{\vec{k}} = \frac{\lambda\omega}{2\pi} = \lambda f \text{ [m/s]}.$$

B.3 in Appendix uses this theory to construct the simulated waves in Figure 2.1. The signal data are produced in S_{dB} form from 0 to 80 dB. Thereafter B.4 converts them into corresponding complex signal data with a random phase. The origin (0,0) marks the point where the imaginary radar is located. As with the logged data the output is a four-dimensional array, with 50 scans here:

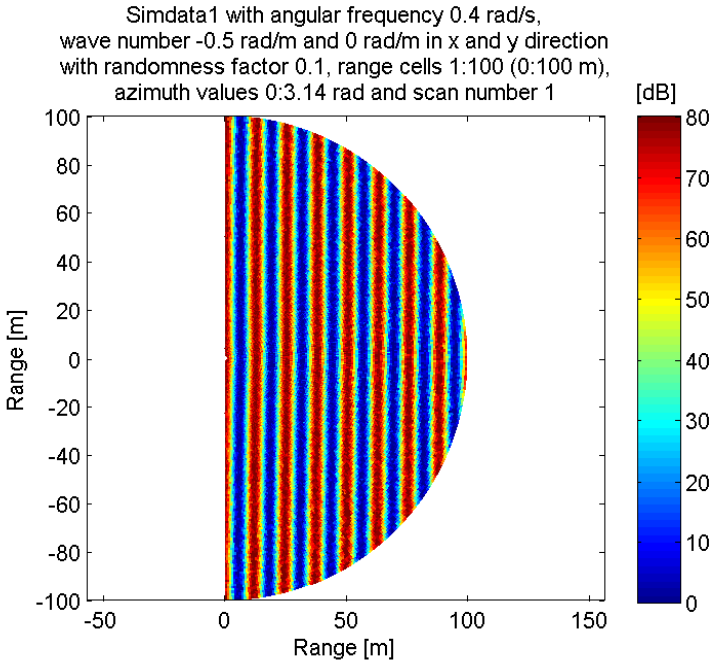


Figure 2.1: Simulated waves.

This now provides a mathematical basis for simulated waves, a useful benchmark for testing prediction models before proceeding to the less perfect or predictable logged waves.

2.1.2 Simulated Moving Target

Within both simulated and logged sea waves, we will inject moving simulated targets as realistic replicas of actual targets. "Moving" encompasses both targets that are actively self-propelling and passively floating, to include small boats, swimmers, birds and marine debris.

Firstly, the target we want to mimic is of small size, its signal being less than or equal to the background sea waves peak.

Signal values "spill over" a given number of cells in the dimensions Doppler, range and azimuth – gradually increasing then decreasing as the beam passes the target's true position. At 3 dB width:

Doppler:	$\sim$1.5 bins;
Range:	$\sim$1.5 cells;
Azimuth:	$\sim$2.5 cells (at range 100 m).

In this thesis, we use the Gaussian profile as a straightforward and acceptable approximation for the radar's resolution of the point target's three dimensions Doppler, range and azimuth, as used in

[29]. The reason for omitting the scan dimension in this profile is that its inclusion would lead to the illogical implication of a target that gradually appears, and then mysteriously disappears as time passes. Instead, the target should be consistently apparent over the given number of scans.

We define D_k, R_l and θ_m as each the Doppler, range and azimuth of the data cell under observation, whereby "0"-subscripts mark the true location of the target [m/s, m and rad respectively]. b_D, b_R and b_θ are width parameters [units as above], A represents amplitude [no unit], and T_{rot} marks the radar rotation time, again $\sim$1.5 [s]. The target profile for a single scan is:

$$S_{dB}(D_k, R_l, \theta_m) = A \cdot e^{-(\frac{D_k - D_0}{b_D})^2} e^{-(\frac{R_l - R_0}{b_R})^2} e^{-(\frac{\theta_m - \theta_0}{b_\theta})^2}, \tag{2.1}$$

whereby the time function for a given scan S is:

$$t(S) = T_{rot} S.$$

Figure 2.2 gives a simplified two-dimensional azimuth-range plot for a given Doppler bin, showing equal signal strengths within each red ellipse around a target's true location marked in green:

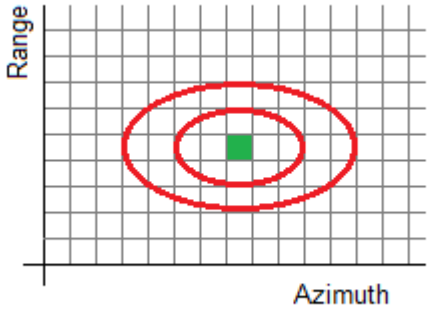


Figure 2.2: Target signal in azimuth-range graph.

The signal value takes the form of an ellipse – such that the further we distance ourselves from the true location the weaker the signal becomes. The eccentricity of the ellipse depends on the b_R and b_θ given.

In the case of the logged data, for subscripts "start" and "stop" marking the dimension bounds, we use:

R_l taking values $R_{start} : 5.428 : R_{stop}$ (5.428 m – range cell size);
θ_m taking values $\theta_{start} : 0.48 : \theta_{stop}$ (0.48° – approx. azimuth cell size).

The amplitude A should be set at a value less than or equal to the background wave signal which peaks at about 80 dB. We ambitiously elect 60 dB for the simulated waves and for the logged waves we cautiously take the maximum suitable value of 80 dB.

The profile thus far in Equation (2.1) is for a merely stationary target. The next task is to alter this to reflect a target that moves at constant velocity. Constant velocity means constant Doppler bin, so the Doppler dimension stays untouched, meaning that D_0 is constant. Only the range and azimuth dimensions in the target profile need adjusting.

For this, we define the target as starting at (R_0, θ_0) in polar coordinates. The velocity of the target is given by $\vec{V}$, whereby V_R highlights the range velocity [m/s] and V_θ signifies the angular velocity [rad/s], as shown in Figure 2.3:

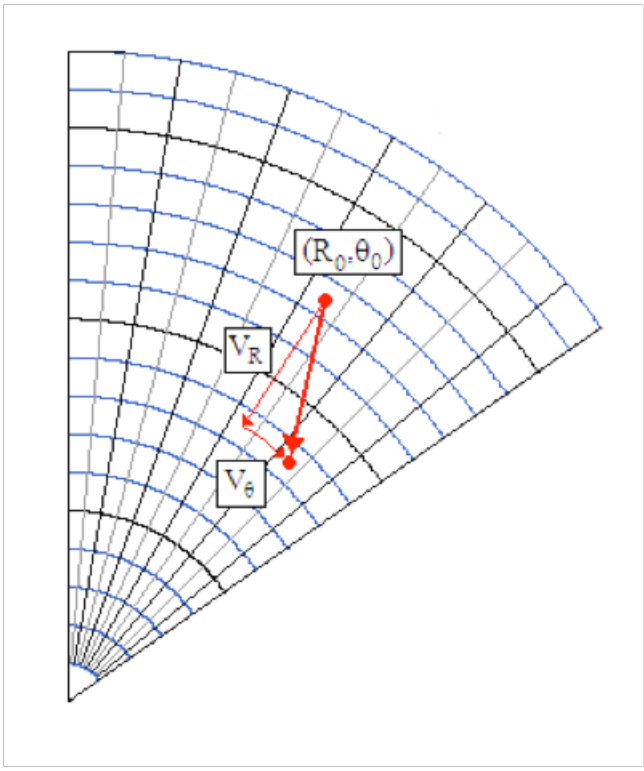


Figure 2.3: Polar plot of target position and velocity.

The red dot target's position at time t:

$$(R_0 + V_R t, \theta_0 + V_\theta t).$$

Note that, as mentioned previously, a negative range velocity V_R implies a target approaching the radar location at the pole – analogous to the origin (0,0) in the Cartesian system.

These considerations are valid for small angles ($\sim 10^\circ$), as the angular velocity of a straight travelling target can vary significantly over larger azimuth scopes.

The moving target profile equation now becomes:

$$S_{dB}(D_k, R_l, \theta_m, S) = A \cdot e^{-(\frac{D_k - D_0}{b_D})^2} e^{-(\frac{R_l - (R_0 + V_R(T_{rot}S))}{b_R})^2} e^{-(\frac{\theta_m - (\theta_0 + V_\theta(T_{rot}S))}{b_\theta})^2}.$$

Crucially, we elect the signs and values of the constant Doppler and the range velocity to be consistent with one another. From earlier Equation (1.1) the relation between Doppler and range velocity is:

$$D_0 = \frac{V_R}{2.5}.$$

Next, real-life targets do not have steady signals but tend to have fluctuating ones from one scan to the next. In a further attempt to create as lifelike a target as possible, the Swerling fluctuation model [4] is used here to imitate this effect. On the one hand, Swerling V, also known as Swerling 0, as the simple case implies constant amplitude or no fluctuation, which is the profile's portrayal until now. On the other hand, Swerling I signifies fluctuating amplitude, constant within a scan, but uncorrelated from scan to scan. Figure 2.4 [4] containing all five models, illustrates this:

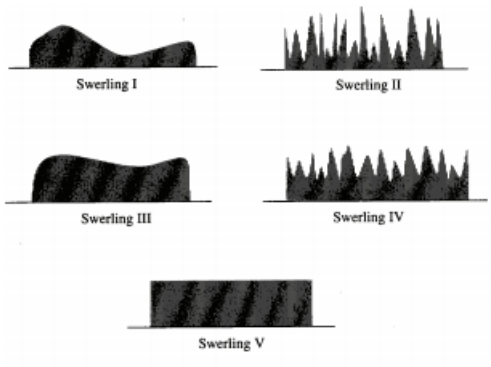


Figure 2.4: Swerling models (Source: G. Brooker, *Sensors and Signals*, Australian Centre for Field Robotics, 2006).

At this stage it may be helpful to provide an interim alteration suggestion for the sourced figure above. The gradual slopes in the Swerling I case suggest a correlation between neighbouring scans. A more appropriate portrayal would be that shown in Swerling II (where fluctuations are independent from pulse to pulse, not our choice here). While the figure is technically not incorrect, it is important to have an accuracte understanding of the fluctuating amplitude for our choice.

In either Swerling case the amplitude is complex, as the signals of the sea are complex due to the Hilbert filter, meaning it follows the relation:

$$A^2 = A_I^2 + A_Q^2. \tag{2.2}$$

This means that the values A_I (In phase – the real part of the signal) and A_Q (Quadrature phase – the imaginary part of the signal) can be any of the points lying on the perimeter of a circle of radius A where A_I and A_Q represent the real and imaginary axes respectively. Figure 2.5 gives an illustration of this:

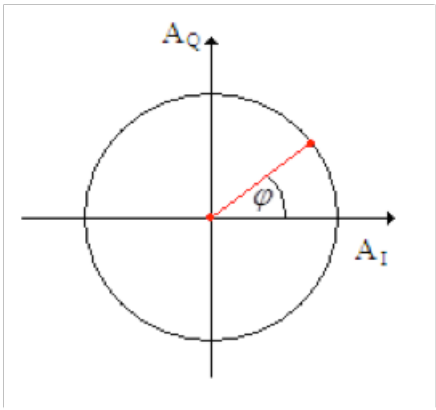


Figure 2.5: In phase and Quadrature phase signal.

$$A_I = A \cdot \cos(\phi),$$

$$A_Q = A \cdot \sin(\phi),$$

$$0 \leq \phi < 2\pi.$$

On the other hand, Swerling I is the more realistic option with fluctuating amplitude A from scan to scan – it sets the amplitude as Rayleigh-distributed. Generally, $Z \sim Rayleigh(\sigma)$ is Rayleigh distributed if $Z = \sqrt{X^2 + Y^2}$, where $X \sim N(0, \sigma^2)$ and $Y \sim N(0, \sigma^2)$. This means that by Equation (2.2) holds, A_I and A_Q are each normally distributed with mean zero and variance equalling half the power level. Further the phase is uniformly distributed, i.e. the position is random anywhere on the circle's perimeter. Summarised, the distributions are as follows:

$$\phi \sim U(0, 2\pi).$$

$$A_I \sim N(0, \frac{P}{2}),$$

$$A_Q \sim N(0, \frac{P}{2}).$$

So under Swerling 0 the signals are non-fluctuating, whereas under Swerling I they are of constant amplitude throughout the scan but fluctuate and are uncorrelated from scan to scan [4].

To demonstrate the target signals, we construct (see Appendix B.5) four example targets for each of the wave environments. For the simulated waves we choose four example targets named *1c* to *4c*, size (2-by-2 in range and azimuth, unused targets *1b* to *4b* are of size 1-by-1). Though sizes are equal, they have different velocities and starting positions. Their trails B.20 (see Appendix) across all scans ("multiple exposure") are inserted into a "blank" (blue in this colour scale) zero value background in Figure 2.6:

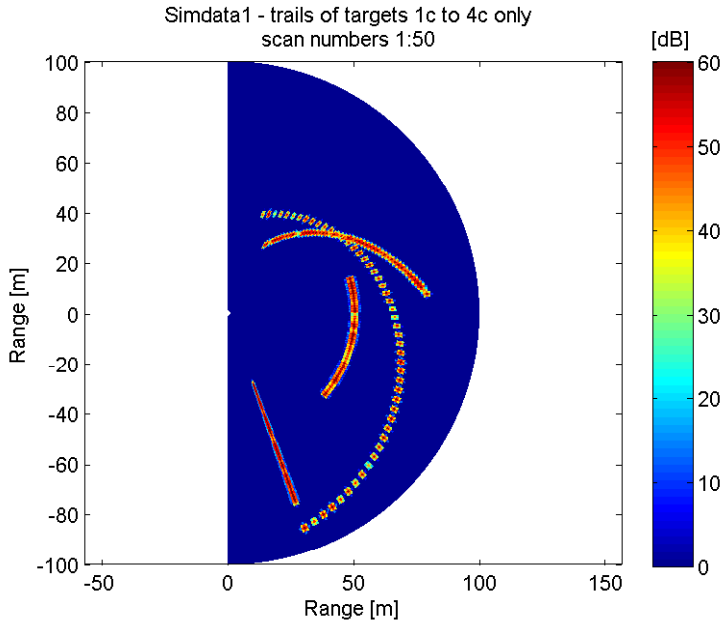


Figure 2.6: Target trails (for simulated waves).

In the case of the logged waves the target amplitude is 80 dB. This is the maximum value it can take – the smaller the target amplitude, the more difficult it will be to find it later on. Further, we explore different spatial sizes (1-by-1, 2-by-2, 3-by-3, 4-by-4 cells in range and azimuth), bearing in mind this may cause issues later (cf. Section 3.5.1). Figure 2.7 shows the trails of the target signals for our standard radar plot:

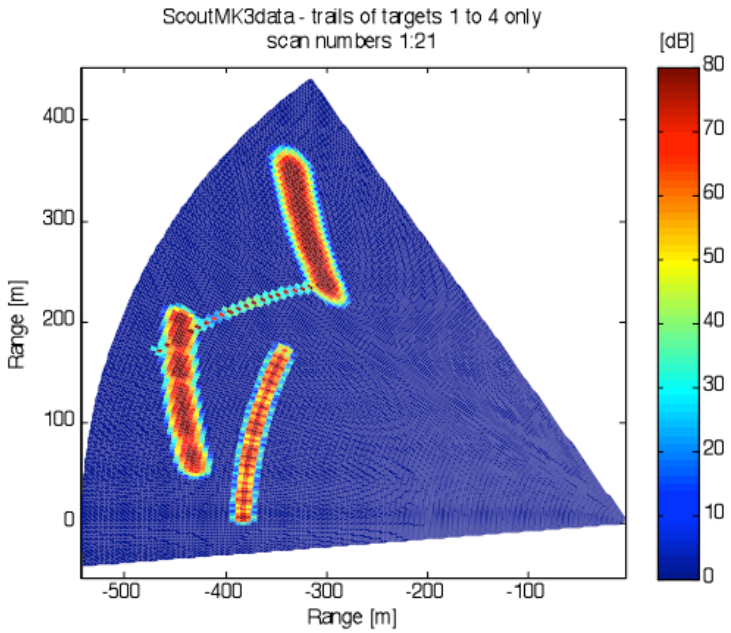


Figure 2.7: Target trails (for logged waves).

Using a single scan instead, the signals for four example targets at Swerling 0, with their properties entitled and colour axes adjusted, present themselves in Figure 2.8:

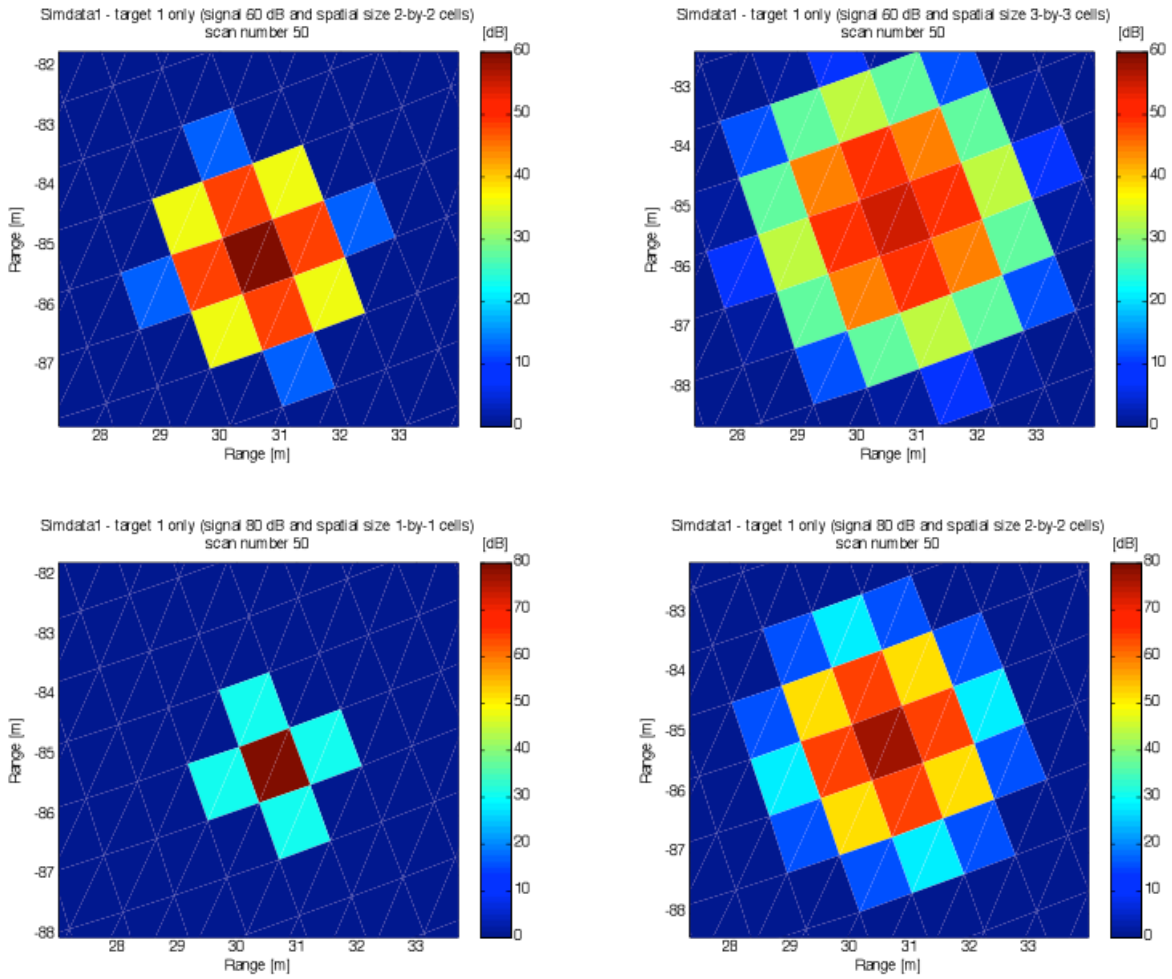


Figure 2.8: Targets of various sizes and maximum signal amplitudes.

These complex target signals in a zero-signal background are later injected into the wave data, by adding them to the complex wave signals. Note that their complex phases are independent of one another.

We now have discussed the mathematical basis for the simulated moving target, to be injected into wave data.

2.2 Covariance and Correlation

Fundamental to our operations and calculations are the tools of covariance and the (Pearson's) correlation coefficient. Both measures are widely used to quantify the degree of linear dependence between two variables, or their degree of association [26].

While the covariance is dependent on the units obtained from the two variables, the correlation

is dimensionless – a value between [-1,1], where 1 is total positive correlation, -1 is total negative correlation and 0 signifies no correlation.

The relation for the covariance σ_{XY} and correlation ρ_{XY} for two random variables X and Y, with mean μ_X and μ_Y and standard deviation σ_X and σ_Y is [13]:

$$\sigma_{XY} = \mathrm{E}[(X-\mu_X)(Y-\mu_Y)],$$
$$\rho_{XY} = \frac{\mathrm{E}[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}.$$

Covariance and correlation are defined by expected values. In practice however, they are estimated by the empirical average. When X and Y are both mean zero, then the estimator is by [31]:

$$\hat{\sigma}_{XY} = \frac{1}{N}\sum_{i=1}^{N} X_i Y_i. \tag{2.3}$$

A larger data set N gives a better estimate. The quality of an estimator can be assessed by calculating its standard deviation or variance, as is considered in the following section.

Note that when applying this to the wave data, the mean for a specific Doppler must be subtracted in order to bring all signals within that Doppler to mean zero.

In using this principle, we employ a vector containing the signal values of a group of cells along multiple dimensions. We then calculate the correlation coefficient with respect to the same group of cells shifted by a specific cell-amount along one or multiple dimensions. This procedure will be carried out on a given number of shifts in each dimension.

Importantly, these calculations are only valid for a limited area of the sea waves, as statistical properties vary across the data set. Multi-dimensional correlation coefficient plots within these areas will reveal useful wave parameters – such as the wavelength and period or frequency.

2.3 Accuracy of Measures

We must determine a suitable number of indices when calculating e.g. covariance or correlation. To determine an appropriate minimum index sample size we calculate the standard deviation of the estimated correlation. This will also aid us in calculating a confidence interval at a given confidence level.

Let us begin with finding an expression for the variance of our samples based on sample size N.

Lemma 1:
Given two random samples:

$$\mathbb{X} = \{X_1, ..., X_N\},$$
$$\mathbb{R} = \{r_1, ..., r_N\},$$

where $\{X_i, r_i\}$, $i = 1, ..., N$ are N independent individuals (cell signals in our case) of the stochastic variables X and r. $X \perp r$ (they are independent). Suppose that:

$$X_i \sim N(0, \sigma_X^2),$$

$$r_i \sim N(0, \sigma_r^2).$$

Then let us further assume random variable Y_i such that :

$$Y_i = aX_i + r_i,$$

for $i = 1, ..., n$. We further define Z such that:

$$Z = \frac{1}{N} \sum_{i=1}^{N} X_i Y_i,$$

as an unbiased estimator of σ_{XY} (as $\mathrm{E}[X] = \mathrm{E}[Y] = 0$) for sample size N. Then:

$$\mathrm{Var}(Z) = \frac{\sigma_X^2(\sigma_r^2 + 2a^2\sigma_X^2)}{N}.$$

Proof. The variance of Y is (by independence):

$$\sigma_Y^2 = a^2\sigma_X^2 + \sigma_r^2. \tag{2.4}$$

$$\begin{aligned}
\mathrm{Var}(Z) &= \mathrm{Var}(\frac{1}{N}\sum_{i}^{N}(X_i Y_i)) \\
&= \frac{1}{N^2} N \cdot \mathrm{Var}(X_i Y_i) \\
&= \frac{1}{N} \mathrm{Var}(aX_i^2 + rX_i) \\
&= \frac{1}{N} (\mathrm{E}[aX_i^2 + rX_i]^2 - (\mathrm{E}[aX_i^2 + rX_i])^2) \\
&= \frac{1}{N} (a^2 \mathrm{E}[X_i^4] + r^2 \mathrm{E}[X_i]^2 + 2\,\mathrm{E}[rX_i^3] - a^2\sigma_X^4),
\end{aligned}$$

where by the fourth moment of normal distribution:

$$\begin{aligned}
&= \frac{1}{N} (3a^2\sigma_X^4 + \sigma_r^2\sigma_X^2 - a^2\sigma_X^4) \\
&= \frac{\sigma_X^2(\sigma_r^2 + 2a^2\sigma_X^2)}{N}.
\end{aligned}$$

□

This result is plausible as increasing N causes the variance to decrease. Now to calculate the variance for a given number of samples N we must calculate the variance of the correlation coefficient.

The variance of the estimator of the covariance above is the numerator of the correlation equation:

$$\rho_{XY} = \frac{\mathrm{E}[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}.$$

Given samples of data, let us consider the estimator of this parameter in the following lemma.

Lemma 2:
Given random samples $\{X_1,...,X_N\} = \mathbb{X}$ and $\{Y_1,...,Y_N\} = \mathbb{Y}$, we name each element of the random samples as an individual. Let us consider the estimator of ρ:

$$\hat{\rho}_{XY} = \frac{\frac{1}{N}\sum_{i=1}^{N} X_i Y_i}{\sigma_X \sigma_Y}, \tag{2.5}$$

then:

$$\mathrm{Var}(\hat{\rho}_{XY}) = \frac{2}{N} - \frac{\sigma_r^2}{N(a^2\sigma_X^2 + \sigma_r^2)}.$$

Proof.

$$\begin{aligned}
\mathrm{Var}(\hat{\rho}_{XY}) &= \frac{\mathrm{Var}(\frac{1}{N}\sum_{i=1}^{N} X_i Y_i)}{\sigma_X^2 \sigma_Y^2} \\
&= \frac{\sigma_X^2(\sigma_r^2 + 2a^2\sigma_X^2)}{N} \cdot \frac{1}{\sigma_X^2(a^2\sigma_X^2 + \sigma_r^2)} \\
&= \frac{\sigma_r^2 + 2a^2\sigma_X^2}{N(a^2\sigma_X^2 + \sigma_r^2)} \\
&= \frac{2(\sigma_r^2 + a^2\sigma_X^2)}{N(a^2\sigma_X^2 + \sigma_r^2)} - \frac{\sigma_r^2}{N(a^2\sigma_X^2 + \sigma_r^2)} \\
&= \frac{2}{N} - \frac{\sigma_r^2}{N(a^2\sigma_X^2 + \sigma_r^2)}.
\end{aligned} \tag{2.6}$$

□

We will make the following assumption:

Assumption 1:

$$\sigma_Y^2 = \sigma_X^2$$

or

$$a^2\sigma_X^2 + \sigma_r^2 = \sigma_X^2,$$

due to equal statistical properties within the dataset. The physical justification for this is that the waves behave very similarly in the same area of sea – an ergodic process [6]. However, having said that, over a large area of sea there will certainly be change in these properties – which calls for the requirement of segregating into subdivisions to be defined later.

Corollary 1: Given the random samples $\mathbb{X} = \{X_1, ..., X_N\}$ and $\mathbb{Y} = \{Y_1, ..., Y_N\}$.
Under Assumption 1:

$$\sigma_Y^2 = \sigma_X^2.$$

Then:

$$\text{Var}(\hat{\rho}_{XY}) \approx \frac{(1+\rho_{XY}^2)}{N}.$$

Proof. By Equation (2.4) it follows that Equation (2.6) becomes:

$$\begin{aligned}
\text{Var}(\hat{\rho}_{XY}) &= \frac{2}{N} - \frac{(1-a^2)\sigma_X^2}{N(a^2\sigma_X^2 + (1-a^2)\sigma_X^2)} \\
&= \frac{2}{N} - \frac{(1-a^2)\sigma_X^2}{N\sigma_X^2} \\
&= \frac{(1+a^2)}{N}.
\end{aligned} \tag{2.7}$$

Now finding a relation between a and ρ_{XY}:

$$\begin{aligned}
\rho_{XY} &= \frac{\text{E}[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y} \\
&= \frac{\text{E}[(X-0)(Y-0)]}{\sigma_X \sqrt{a^2\sigma_X^2 + \sigma_r^2}} \\
&= \frac{a\sigma_X^2}{\sigma_X \sqrt{a^2\sigma_X^2 + \sigma_r^2}} \\
&= \frac{a\sigma_X}{\sqrt{a^2\sigma_X^2 + (1-a^2)\sigma_X^2}} \\
&= \frac{a\sigma_X}{\sqrt{\sigma_X^2}} \\
&= a.
\end{aligned}$$

Using this finding, we can substitute this into the earlier Equation (2.7):

$$\text{Var}(\hat{\rho}_{XY}) \approx \frac{(1+\rho_{XY}^2)}{N},$$

or translated to the standard deviation $s_{\hat{\rho}_{XY}}$:

$$s_{\hat{\rho}_{XY}} \approx \sqrt{\frac{(1+\rho_{XY}^2)}{N}}. \tag{2.8}$$

□

Constructing a confidence interval means we have:

$$\hat{\rho}_{XY} \pm z(\frac{\alpha}{2}) s_{\hat{\rho}_{XY}}.$$

For a 95% confidence interval, i.e. $\alpha = 0.05$ and substituting Equation (2.8), yields a confidence interval size Δ of:

$$\Delta = 1.96\sqrt{\frac{(1+\rho_{XY}^2)}{N}}.$$

We want to make the Δ sufficiently small. We know that when r is a constant rather than a random variable, then $\rho_{XY} = 1$. If a random variable, then $\rho_{XY} < 1$. So taking the maximum value of 1, we want to find the number of indices N that yields a lowest confidence interval. This Δ shall adhere to the quantity 0.1 used later and reasoned in Subsection 2.4.2. Calculated this means an index quantity of 768. We choose the higher and numerically more convenient quantity 1000, and the number of indices for calculations in this thesis will adhere to the herewith mathematically justified quantity.

Importantly, we must strike a balance between the greatest number of indices while keeping statistical property across the cells to a minimum. Thus, the N cells must be confined to a specific area or subdivision where statistical properties are relatively constant. Should this subdivision host an insufficient cell quantity, then scan-shifted cells from that subdivision can accommodate for this.

2.4 Prediction Models

The idea behind prediction models here is to best predict each cell based on what is expected in a wave-only setting. For this, we calculate predictor cells and coefficients for each subdivision (cf. Section 3.2.5) of a wave-only area. Into this, one or more simulated targets are injected. The sea area, now containing waves as well as targets somewhere among them, is then scanned cell by cell – each treated as a cell under test and each assigned a predicted value based on predictor cells and coefficients for the respective subdivision. On the one hand, a cell with little difference between the predicted and true signal, the dB residual signal, likely contains merely background waves. On the other hand, a significant gap, or high residual signal, indicates that the cell is likely hosting a target. In effect, we are "combing" through the area of sea with a comb of predictor coefficients, exposing any targets located there.

Two prediction models, or methods, are appropriate for this task – first we discuss the Linear Prediction model.

2.4.1 Linear Prediction

Linear Prediction modelling is used in a wide variety of fields, including data forecasting [30]. We use it here to predict the signal of a cell under test X, based on the signal values of m specific predictor cells $Y_1, Y_2, ..., Y_m$. In a subdivision of the sea containing solely sea waves we calculate the optimal predictor coefficients $a_1, a_2, ..., a_m$ valid for the following equation:

$$X = a_1 Y_1 + a_2 Y_2 + ... + a_m Y_m + Noise. \tag{2.9}$$

To emphasise, these coefficients are calculated for a specific subdivision, and are valid for that subdivision only.

A more detailed explanation of the method: First, a choice of predictor cells must clearly be made. While cells close to the cell under test have the most predictive capability, there is a limitation in both range and azimuth. The reason for this is that there is a correlation caused by waves and a separate correlation caused by the beamwidth – the target's overflowing signal effects on neighbouring cells. Since our interest lies in the correlation caused by waves only, we skip the neighbouring cells, "guard cells", and limit the choice of predictor cells to ones beyond. This prevents "self-killing" of the target, the principle whereby its own signal hampers its detection. In spite of this, we do have the freedom to use the neighbouring cells along time-shifted data, as passing time neutralises this effect.

We choose 25 predictor cells within a two-cell distance from the cell under test from each the previous (Scan -1) and imminent (Scan 1) scans, and also 16 at distance two from the cell under test from the current (Scan 0) scan. This gives a total of 66 predictor cells. These are marked "•" in Figure 2.9, with the yellow-coloured guard cells surrounding the cell under test X:

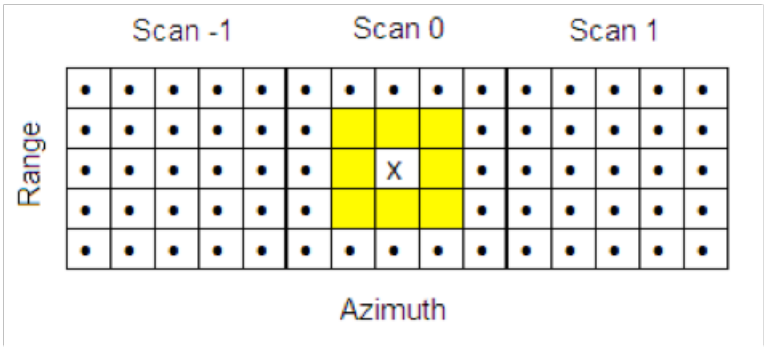


Figure 2.9: Predictor cells for Linear Prediction.

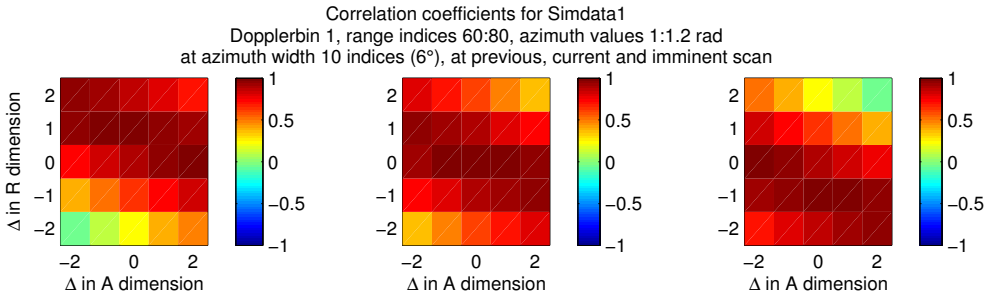


Figure 2.10: Correlation coefficient plots.

The actual correlation coefficients for the simulated waves across the three scans are shown in Figure 2.10 (see Appendix B.6). They show diagonally running high correlation coefficients, which portray the wave peak in that sector. Furthermore, they show the peak of the wave shifting negatively in range as scan increases. This confirms the notion of the waves approaching.

Next, we must calculate the predictor coefficients $a_1, a_2, ..., a_m$. We do this via random samples $\mathbb{X} = [X_1, X_2, ..., X_N]^T$ and $\mathbb{Y}_1 = [Y_{1,1}, Y_{2,1}, ..., Y_{N,1}]^T, ..., \mathbb{Y}_m = [Y_{1,m}, Y_{2,m}, ..., Y_{N,m}]^T$, where $Y_{i,j}$ is the i$^{\text{th}}$ individual in the sample $\mathbb{Y}_j$. Using the above notation in the context of matrix operations means we

are dealing with this matrix definition rather than merely non-matrix sample sets. So the same notation usage as used in Lemma 1 should not be considered problematic, but helpful as these are essentially the same concepts. These random samples contain N zero-mean stochastic [16] signals or individuals, taken from a subdivision of the sea containing solely waves, i.e. no targets.

The expectation of the optimal predictor coefficients $a_1, a_2, ..., a_m$ are the ordinary least square solution of the system:

$$\begin{bmatrix} \mathbb{Y}_1 & \mathbb{Y}_2 & \dots & \mathbb{Y}_m \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} = \mathbb{X}.$$

The solution to this system of equations:

$$\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} = (\begin{bmatrix} \mathbb{Y}_1 & \mathbb{Y}_2 & \dots & \mathbb{Y}_m \end{bmatrix}^T \begin{bmatrix} \mathbb{Y}_1 & \mathbb{Y}_2 & \dots & \mathbb{Y}_m \end{bmatrix})^{-1} \begin{bmatrix} \mathbb{Y}_1 & \mathbb{Y}_2 & \dots & \mathbb{Y}_m \end{bmatrix}^T \mathbb{X}.$$

The inner products resulting from the first two matrices are the sum of the component-wise products Y_j and Y_k, which is $\sum_{i=1}^{N} Y_{i,j}^T Y_{i,k}$ The same holds true for the latter two matrices which equal $\sum_{i=1}^{N} Y_{i,j}^T X_i$. We can multiply each by factor $\frac{1}{N}$, as they cancel:

$$\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} = (\frac{1}{N} \begin{bmatrix} \mathbb{Y}_1 & \mathbb{Y}_2 & \dots & \mathbb{Y}_m \end{bmatrix}^T \begin{bmatrix} \mathbb{Y}_1 & \mathbb{Y}_2 & \dots & \mathbb{Y}_m \end{bmatrix})^{-1} \frac{1}{N} \begin{bmatrix} \mathbb{Y}_1 & \mathbb{Y}_2 & \dots & \mathbb{Y}_m \end{bmatrix}^T \mathbb{X}. \tag{2.10}$$

We estimate the covariance using the previous Equation (2.3):

$$\hat{\sigma}_{XY} = \frac{1}{N} \sum_{i=1}^{N} X_i Y_{i,j}.$$

We can then simplify by replacing each sum with the covariance, which is by definition the expected value of the inner product. So the estimation of the right-hand side of Equation (2.10) becomes:

$$= \begin{bmatrix} \mathrm{E}[\mathbb{Y}_1^T \mathbb{Y}_1] & \mathrm{E}[\mathbb{Y}_1^T \mathbb{Y}_2] & \dots & \mathrm{E}[\mathbb{Y}_1^T \mathbb{Y}_m] \\ \mathrm{E}[\mathbb{Y}_2^T \mathbb{Y}_1] & \mathrm{E}[\mathbb{Y}_2^T \mathbb{Y}_2] & \dots & \mathrm{E}[\mathbb{Y}_2^T \mathbb{Y}_m] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{E}[\mathbb{Y}_m^T \mathbb{Y}_1] & \mathrm{E}[\mathbb{Y}_m^T \mathbb{Y}_2] & \dots & \mathrm{E}[\mathbb{Y}_m^T \mathbb{Y}_m] \end{bmatrix}^{-1} \begin{bmatrix} \mathrm{E}[\mathbb{Y}_1^T \mathbb{X}] \\ \mathrm{E}[\mathbb{Y}_2 \mathbb{X}] \\ \vdots \\ \mathrm{E}[\mathbb{Y}_m \mathbb{X}] \end{bmatrix}.$$

What follows is an explanation of each of these matrices.

First, the beginning matrix above contains the m-by-m covariances, where m is the number of predictor cells (66 in our calculations). The expected values $\mathrm{E}[\mathbb{X}\mathbb{Y}_j]$ are the covariances $\sigma_{\mathbb{X}\mathbb{Y}_j}$ between the signals, because the samples are mean zero:

$$\begin{aligned}\sigma_{\mathbb{X}\mathbb{Y}_j} &= \mathrm{E}[(\mathbb{X}-\mathrm{E}[\mathbb{X}])(\mathbb{Y}_j-\mathrm{E}[\mathbb{Y}_j])] \\ &= \mathrm{E}[\mathbb{X}\mathbb{Y}_j].\end{aligned}$$

This estimation is justified by the assumption that the statistical properties are constant over the data set. In addition, replacing exact covariances by the average over a data set introduces a statistical error proportional to the standard deviation of the estimator. For this we refer to the earlier recommended sample size $N > 1000$. With far less than 1000 scans, we will achieve this by taking the average over range and azimuth of neighbouring cells.

Note that at this point the rank of the covariance matrix is $\min(m,N)$. In order for the above equation to hold, the rank of the covariance matrix cannot be smaller than m. So in order to estimate m coefficients, the number of individuals N must be at least m. More is preferable. Here the $N = 1000$ cells far exceed the $m = 66$ predictor cells, highly satisfying this condition.

The second matrix is m-by-1, which contains the covariance of the predictor cells and the cell under test.

So if the covariances are known from the data, the optimal predictor coefficients can be calculated.

2.4.2 Sequential Decorrelation

Our second method used for predicting a cell under test is a more refined version of Linear Prediction. This is a signal processing technique for radar systems [14] which is closely related to the Gram-Schmidt orthogonalisation process [20].The predictor cells are updated to $\tilde{Y}_1, \tilde{Y}_2, ..., \tilde{Y}_m$, as distinct from the former method. Again the goal is to find coefficients $a_1, a_2, ..., a_m$ for optimal prediction satisfying the optimisation relation:

$$X = a_1\tilde{Y}_1 + a_2\tilde{Y}_2 + ... + a_m\tilde{Y}_m + Noise.$$

In contrast to the former method, the predictor cells $\tilde{Y}_1, \tilde{Y}_2, ..., \tilde{Y}_m$ are not predetermined, but chosen from a set of possible cells shifted in range, azimuth and scan. It is wise to provide a selection of cells that have high predictive capability for this set of possible cells, B.7 (see Appendix) picks the best from among them.

As before we will use random samples, but now with regards to the new predictor cells:

$$\mathbb{X} = [X_1, X_2, ..., X_N]^T, \tilde{\mathbb{Y}}_1 = [\tilde{Y}_{1,1}, \tilde{Y}_{2,1}, ..., \tilde{Y}_{N,1}]^T, ..., \tilde{\mathbb{Y}}_m = [\tilde{Y}_{1,m}, \tilde{Y}_{2,m}, ..., \tilde{Y}_{N,m}]^T,$$

where $\tilde{Y}_{i,j}$ is the i$^{\text{th}}$ individual in the sample $\tilde{\mathbb{Y}}_j$.

The choice for available predictor cells are those shifted both positively and negatively up to five, five and one cell in dimensions range, azimuth and scan respectively. However, as before, the guard cells must be "guarded" for the current scan only. This results in 121 cells from each the previous and following scan in addition to 112 cells from current scan, totalling 354 available predictor cells, where the process begins with the $\tilde{Y}_1$ cell with the highest correlation with the cell under test X. This

is shown with available predictor cells marked "•", yellow-coloured guard cells, and the cell under test X in Figure 2.11 below:

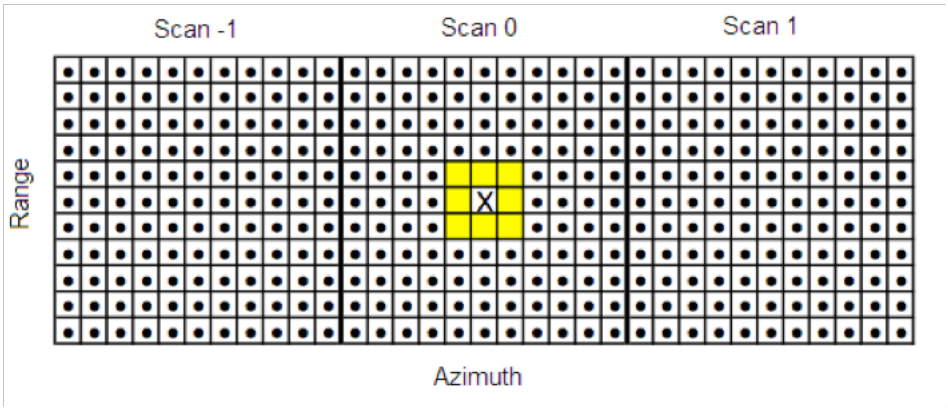


Figure 2.11: Available predictor cells for Sequential Decorrelation.

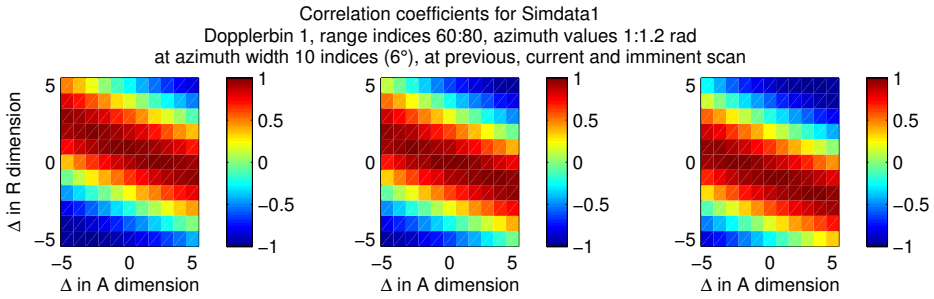


Figure 2.12: Correlation coefficient plots.

The actual correlation coefficients for the simulated waves across the three scans are shown in Figure 2.12. They are an expanded image of the coefficient plots shown in the previous section, giving a clearer image of the diagonally-running correlations and the negative shift in range as scan increases.

The steps to find the predictor cells and corresponding prediction coefficients are described as follows:

Algorithm:

Step 1: Start with the sample $\tilde{\mathbb{Y}}_1$ with the largest correlation with $\mathbb{X}$;

Step 2: Find the corresponding prediction coefficient a_1;

Step 3: Decorrelate remaining signal samples $\tilde{\mathbb{Y}}_2, \ldots, \tilde{\mathbb{Y}}_m$ with $\tilde{\mathbb{Y}}_1$. Their correlation coefficients tend closer to 0;

Step 4: From decorrelated samples $\tilde{\mathbb{Y}}_2^{(1)}, \ldots, \tilde{\mathbb{Y}}_m^{(1)}$, find the one with the largest correlation with $\mathbb{X}^{(1)}$. If its magnitude is above a given threshold, then continue the decorrelation process:

perform Step 2 with sample $\tilde{\mathbb{Y}}_2^{(1)}$, finding a_2 and yielding decorrelated samples $\tilde{\mathbb{Y}}_3^{(2)}, ..., \tilde{\mathbb{Y}}_m^{(2)}$, etc.;

Step 5: Continue this process until all correlations are below the set threshold. Then stop.

Describing these steps individually in greater depth:

Step 1: Observing the available predictor cells in Figure 2.11 we elect the one with the correlation of greatest magnitude.

Step 2: To find the first coefficient a_1, the residue sample $\mathbb{X}^{(1)}$ (individual signals after one decorrelation step) must be uncorrelated with $\tilde{\mathbb{Y}}_1$:

$$\mathbb{X}^{(1)} = \mathbb{X} - a_1 \tilde{\mathbb{Y}}_1$$
$$\frac{1}{N}(\tilde{\mathbb{Y}}_1^T \mathbb{X}^{(1)}) = \frac{1}{N}(\tilde{\mathbb{Y}}_1^T \mathbb{X} - a_1 \tilde{\mathbb{Y}}_1^T \tilde{\mathbb{Y}}_1)$$
$$\mathrm{E}[\tilde{\mathbb{Y}}_1^T \mathbb{X}^{(1)}] = \mathrm{E}[\tilde{\mathbb{Y}}_1^T \mathbb{X} - a_1 \tilde{\mathbb{Y}}_1^T \tilde{\mathbb{Y}}_1] = 0$$
$$a_1 = \frac{\mathrm{E}[\tilde{\mathbb{Y}}_1^T \mathbb{X}]}{\mathrm{E}[\tilde{\mathbb{Y}}_1^T \tilde{\mathbb{Y}}_1]}.$$

Step 3: To continue with finding the next weight a_2, we can repeat this procedure. Bearing in mind that $\tilde{\mathbb{Y}}_1, \tilde{\mathbb{Y}}_2, ..., \tilde{\mathbb{Y}}_m$ are also mutually correlated, we must decorrelate $\tilde{\mathbb{Y}}_2, ..., \tilde{\mathbb{Y}}_m$ with $\tilde{\mathbb{Y}}_1$ in order to avoid losing the decorrelation with $\tilde{\mathbb{Y}}_1$. This can be done by using:

$$\tilde{\mathbb{Y}}_2^{(1)} = \tilde{\mathbb{Y}}_2 - c_{2,1} \tilde{\mathbb{Y}}_1,$$

where, by the same reasoning as in the previous step:

$$c_{i,1} = \frac{\mathrm{E}[\tilde{\mathbb{Y}}_i^T \tilde{\mathbb{Y}}_1]}{\mathrm{E}[\tilde{\mathbb{Y}}_1^T \tilde{\mathbb{Y}}_1]}.$$

Now the whole process can be repeated with $\mathbb{X}^{(1)}, \tilde{\mathbb{Y}}_2^{(1)}, ..., \tilde{\mathbb{Y}}_m^{(1)}$, (1) superscripts denoting the samples after one decorrelation step, where (2) superscripts would denote samples after two decorrelation steps, etc. This completes the first step of the decorrelation procedure. The variances of the samples become smaller with each decorrelation step, as shown in the following step.

Step 4: The order of decorrelation is such that the first sample $\tilde{\mathbb{Y}}_1$ has the largest correlation with $\mathbb{X}$. Thereafter the second sample $\tilde{\mathbb{Y}}_2^{(1)}$ is the one that has the highest correlation with the decorrelated sample $\mathbb{X}^{(1)}$ etc. As the samples have varying variances, the highest covariance does not necessarily mean the highest correlation, as correlation is defined as:

$$\rho_{XY} = \frac{\mathrm{E}[\tilde{\mathbb{Y}}^T \mathbb{X}]}{\sqrt{\mathrm{E}[\tilde{\mathbb{Y}}^T \tilde{\mathbb{Y}}]\,\mathrm{E}[\mathbb{X}^T \mathbb{X}]}}.$$

The covariance between $\tilde{\mathbb{Y}}_j^{(1)}$ and $\tilde{\mathbb{Y}}_k^{(1)}$ we obtain via $\sigma^T_{\tilde{\mathbb{Y}}_j,\tilde{\mathbb{Y}}_k} = \sigma_{\tilde{\mathbb{Y}}_j,\tilde{\mathbb{Y}}_k}$:

$$\begin{aligned}
\sigma_{\tilde{\mathbb{Y}}_j^{(1)},\tilde{\mathbb{Y}}_k^{(1)}} &= \mathrm{E}[\tilde{\mathbb{Y}}_j^{(1)T}\tilde{\mathbb{Y}}_k^{(1)}] \\
&= \mathrm{E}[(\tilde{\mathbb{Y}}_j - c_{j,1}\tilde{\mathbb{Y}}_1)^T(\tilde{\mathbb{Y}}_k - c_{k,1}\tilde{\mathbb{Y}}_1)] \\
&= \mathrm{E}[\tilde{\mathbb{Y}}_j^T\tilde{\mathbb{Y}}_k] - c_{k,1}\,\mathrm{E}[\tilde{\mathbb{Y}}_j^T\tilde{\mathbb{Y}}_1] - c_{j,1}^T\,\mathrm{E}[\tilde{\mathbb{Y}}_1^T\tilde{\mathbb{Y}}_k] + c_{j,1}^T c_{k,1}\,\mathrm{E}[\tilde{\mathbb{Y}}_1^T\tilde{\mathbb{Y}}_1] \\
&= \sigma_{\tilde{\mathbb{Y}}_j,\tilde{\mathbb{Y}}_k} - \frac{\sigma^T_{\tilde{\mathbb{Y}}_1,\tilde{\mathbb{Y}}_j}\sigma_{\tilde{\mathbb{Y}}_1,\tilde{\mathbb{Y}}_k}}{\sigma_{\tilde{\mathbb{Y}}_1,\tilde{\mathbb{Y}}_1}}.
\end{aligned}$$

In the event when $j = k$, it is clear that the above that the variances of the sample become smaller with each decorrelation step. New correlations with $j = 1$ or $k = 1$ are zero, so the correlation with $\tilde{\mathbb{Y}}_1$ is completely removed.

Defining $\sigma_{\tilde{\mathbb{Y}},\tilde{\mathbb{Y}}}$ as the covariance matrix of all the $\tilde{\mathbb{Y}}$ samples, the above result can be written as:

$$\sigma^{(1)}_{\tilde{\mathbb{Y}},\tilde{\mathbb{Y}}} = \sigma^{(0)}_{\tilde{\mathbb{Y}},\tilde{\mathbb{Y}}} - \frac{1}{\sigma^{(0)}_{\tilde{\mathbb{Y}},\tilde{\mathbb{Y}}_{(1,1)}}} \begin{bmatrix} \sigma^{(0)}_{\tilde{\mathbb{Y}},\tilde{\mathbb{Y}}_{(1,1)}} \\ \vdots \\ \sigma^{(0)}_{\tilde{\mathbb{Y}},\tilde{\mathbb{Y}}_{(j,1)}} \end{bmatrix} \begin{bmatrix} \sigma^{(0)}_{\tilde{\mathbb{Y}},\tilde{\mathbb{Y}}_{(1,1)}} & \cdots & \sigma^{(0)}_{\tilde{\mathbb{Y}},\tilde{\mathbb{Y}}_{(1,k)}} \end{bmatrix}.$$

This new covariance matrix contains only zeros in the first row and column.

Step 5: The decorrelation process continues one signal at a time until the power level of $\mathbb{X}^{(z)}$, the sample after z decorrelation steps, is near noise level or until the remaining correlations with $\mathbb{X}^{(z)}$ are all very low. The limit for reaching "low" correlation is difficult to define precisely, but some authors [5] have offered table guidelines. We elect 0.1 as a threshold for little to no correlation. At this point the remaining Y signals no longer have predictive value and the decorrelation process stops.

This now provides a novel and valuable prediction model i.e. a self-calibrating one, as in [11].

Either prediction model has advantages and disadvantages. While the former can be considered more straightforward with its predetermined predictor cells, the latter has the benefit of distinguishing those $\tilde{Y}$ signals that have predictive value from those that do not.

2.5 Target Detection and Comparison of Prediction Models

The resulting residual signals from each of the methods are then analysed and treated by form of minimum threshold-setting. Detections are those cells containing residual signals that are above the threshold. This threshold will be based on a false alarm rate, 10^{-6} being the typical criteria in radar systems [4] – i.e. of 10^6 measurements, one may generate a false positive. As the distribution of the residual signals is unknown, however, the threshold cannot be found through simple probability density function calculations. Instead, we predict the area of sea without injected targets, generating relatively low residual signals. Among a quantity of 10^6, the highest one is set as the threshold. This threshold is expected to be lower than residual signals generated by targets.

The two distinct prediction models will yield distinct results, i.e. detections. We will use two different measures to compare those results, giving a numerical score.

2.5.1 Discrimination and Calibration

There are several ways to mathematically assess the accuracy of a model, two major ones being discrimination and calibration [9].

For the former concept, we use the ROC (receiver operating characteristic) – a technique to visualise classifiers based on performance [15]. It is a useful tool to portray the trade-off between the true positive and false positive rate (in signal detection theory the hit and false alarm rate respectively) – used in analysis of radar systems in World War II [7]. Interestingly it portrays this graphically with respect to many thresholds, not just our single threshold hitherto based on the false alarm rate 10^{-6}.

We define each *FP, TP, FN, TN* as the number of false positives, true positives, false negatives, true negatives, and fp_{rate} and tp_{rate} as the false positive rate ($1-$ specificity) and true positive rate (sensitivity) respectively. We estimate them by [15]:

$$\hat{fp}_{rate} = \frac{FP}{FP+TN},$$

$$\hat{tp}_{rate} = \frac{TP}{FN+TP}.$$

In the plot each threshold represents a corresponding point – false positive versus true positive rate. The connected points form the ROC curve.

The curve's first point is at $(0,0)$ and the last one at $(1,1)$, where in between these extremes the useful discrimination information lies. A ROC curve based on random guessing would produce the diagonal line between them. Pleasing ROC curves move steeply upwards before tending right, with a larger gap between them and the diagonal. A perfect case would be one moving vertically up to $(0,1)$ before moving horizontally to $(1,1)$.

A useful numerical measure in this realm is the *AUC* (area under curve) [9]. This is at most 1, and in theory at least 0, though as 0.5 is the area from random guessing, this is the practical lower bound [15]. The larger the *AUC*, the more discriminating the model.

Figure 2.13 is an example from [15] of a typical ROC curve:

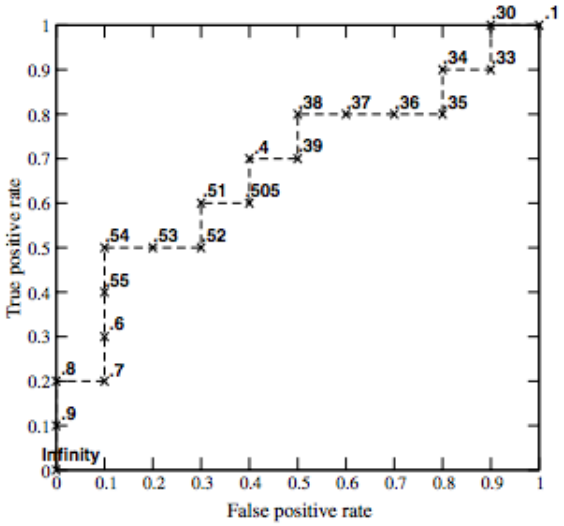


Figure 2.13: Example ROC curve (Source: T. Fawcett "An introduction to ROC analysis", *Pattern Recognition Letters*, 2005).

Each of the points corresponds to a chosen threshold, gradually forming the steps that move right, upwards or both (not in the above case). This threshold starts at infinity, when all detections are negative ("target not present") and decreases steadily to zero, where all positives mean targets present everywhere. In between these extremes is a trade-off between the specificity $(1-fp_{rate})$ and sensitivity (tp_{rate}).

For the latter concept calibration, we use the score function. Calibration is a measure of prediction accuracy or of how close the prediction is to the data, and the score function helps measure this numerically. Several score systems exist, a widely used one being Brier's score [21], which operates in the following manner:

Given two variables $W(X_i)$ and $\pi(X_i)$, for a given cell of interest X_i such that:

$$W(X_i) = \begin{cases} 1 & \text{if detection made} \\ 0 & \text{otherwise} \end{cases},$$

$$\pi(X_i) = \begin{cases} 1 & \text{if target present} \\ 0 & \text{otherwise} \end{cases},$$

then the score A_{score} is:

$$A_{score} = \frac{1}{N}\sum_{i=1}^{N}(W(X_i) - \pi(X_i))^2.$$

$W(X_i)$ is more generally the probability of the forecast (e.g. an 80% chance of rain). In our case however, for a given detection threshold it is binary – detection made or not made. The actual outcome $\pi(X_i)$ is also binary – target present or not present. A distinct threshold applies here, where the signal must be above a specific fraction of the maximum target signal in order to be classified as "target present". As both variables are binary, the Brier's score in this case is equal to the error rate A_{error}.

The value of the score can be between 0 and 1 – similar to the *AUC* value, as a distinct measure of discrimination – the closer to 0, the better the model.

Accuracy A_{acc} is defined as $1 - A_{error}$. In terms of our previous notation it is estimated by:

$$\hat{A}_{acc} = \frac{TP + TN}{FP + TP + FN + TN}.$$

In our case $A_{score} = A_{error}$, which means A_{acc} is also between 0 and 1, where 1 is most accurate. It is this value that we will be measuring and plotting. Each point on the ROC plot contains an accuracy, but the value cannot be determined from the plot per se. So for the practical results later we will add a plot depicting the calculated accuracy for each of the thresholds.

Chapter 3

Implementation and Results

The first step of the applying the theory into practice is to examine the resulting correlation coefficient plots for given areas of the sea waves, which reveal useful parameters such as the wavelength and period or frequency of the waves. Additionally the velocity can be deduced from these.

3.1 Correlation Coefficient Plots and Parameters for Simulated Waves

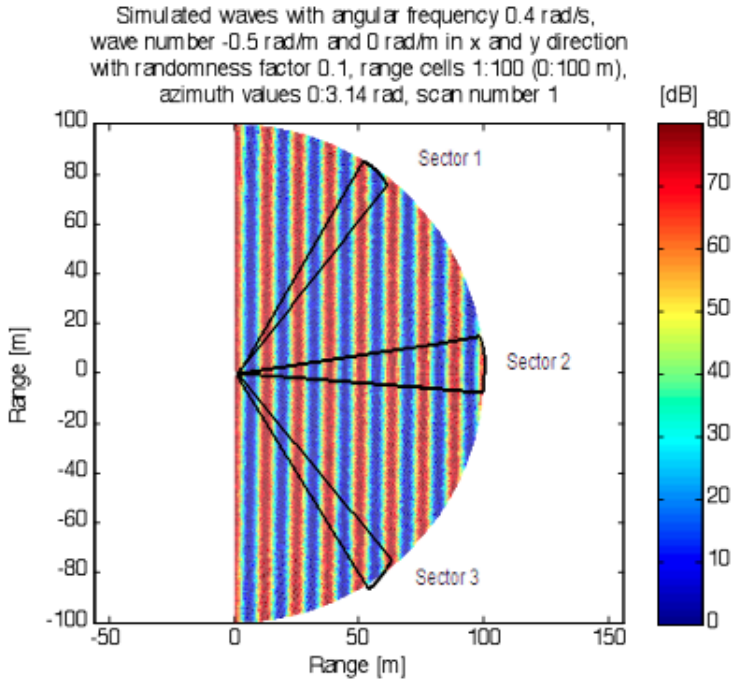


Figure 3.1: Sectors (for simulated waves).

Statistical properties vary across distinct azimuths. We must therefore calculate correlation coefficients within sectors of small width. A sector of 21 indices wide, equivalent to $\sim 10^{\circ}$, is sufficiently

small, where incident angles from incoming waves do not greatly differ. Three sectors 1, 2 and 3 bounded by azimuth indices 68:88, 147:167 and 225:245 respectively (whose central azimuths correspond to $\pi/4$, $\pi/2$ and $3\pi/4$ rad) drawn in black in the simulated waves plot in Figure 3.1:

Our first and simplest case is Sector 2 where the waves move perpendicularly towards the origin. B.8 (see Appendix) plots the correlations yielding in Figure 3.2:

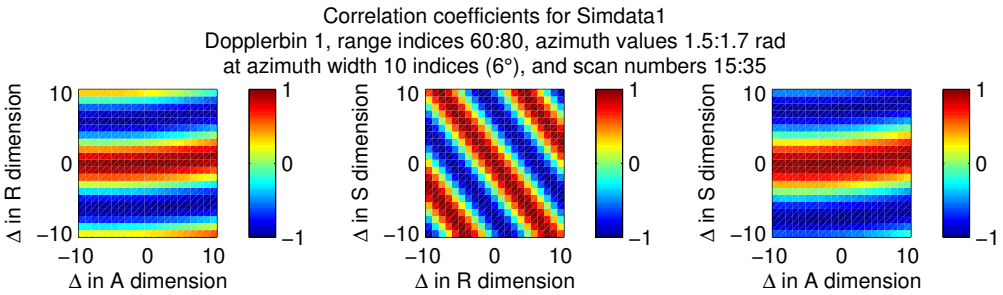


Figure 3.2: Correlation coefficient plots for simulated waves - middle sector 2.

Firstly, in the azimuth-range plot, a zero shift in range maintains a high correlation regardless of the azimuth. This reflects the horizontal position of the wave. Indeed this plot can be seen as simply a two-dimensional plot of the waves from the perspective of the radar. Next, in the range-scan plot, high correlations are inversely related. This reflects the movement of the scene: as time passes, or scan increases, the range decreases, marking the approaching movement of the wave crests. The following azimuth-scan plot shows horizontal correlation patterns of a fixed scan movement along the azimuth means movement along the wave crest or trough. As scan increases, correlation fades due to the wave passing. The correlation then turns negative, becoming strongly negatively correlated at the wave trough. The arrival of a new wave brings high correlation once again – cut off here due to limited scans.

We can also infer wave parameters. The simulated waves plot shows the wavelength to be $\sim$12 range cells, as confirmed in the azimuth-range plot via the vertical distance. Furthermore, a radar-image sequence of the simulated waves (Appendix B.9) counts 16 scans to complete one wavelength, which neatly fits with the inferred period of 16 scans in the correlation plot of scans – the vertical distance between similar correlations. From these, and as the waves are approaching, the velocity is -0.75 m/s (whereby among simulated waves 1 range cell := 1 m and 1 scan := 1 s). Figures 3.3 and 3.4 show the plots for the other sectors:

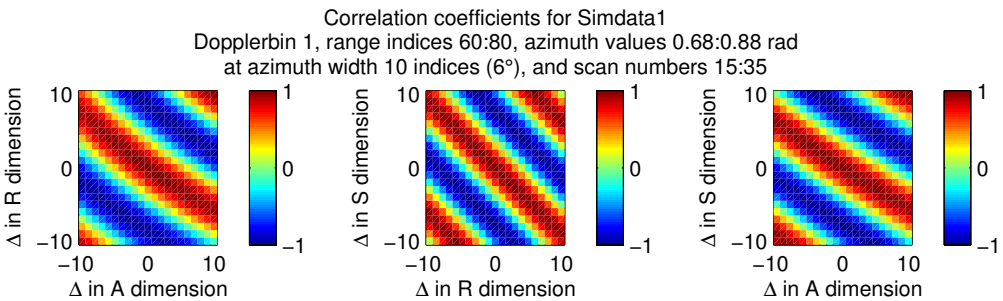


Figure 3.3: Correlation coefficient plots for simulated waves - upper sector 1.

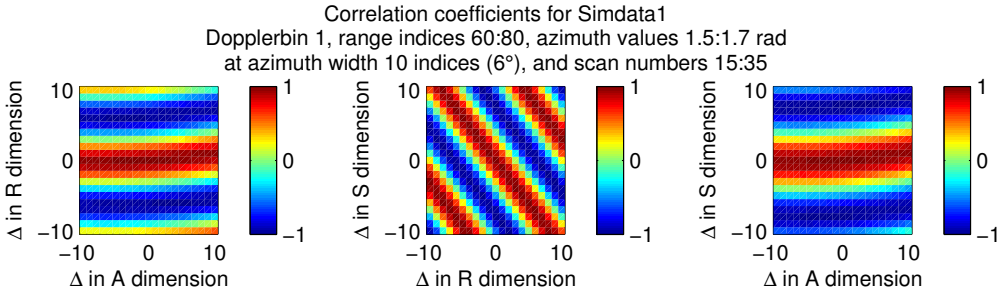


Figure 3.4: Correlation coefficient plots for simulated waves - lower sector 3.

Firstly, the azimuth-range plot again depicts the respective two-dimensional plot of the waves from the perspective of the radar. Next, the range-scan plot shows no difference between upper and lower sector, however both are less steep than in the central sector. This is due to the fact that along a fixed, but slanted, azimuth the wave velocity is greater than along the perpendicular. The following azimuth-scan plot shows that as scan increases, azimuth stays correlated in different manners. For the upper sector we have decreasing azimuth because the wave moving towards the radar starts reaching the left portion of the sector, at constant range. The reverse holds for the lower sector.

As for the parameters, we observe an altered wavelength of $\sim$16 range cells due to the new slanted perspective. The wave period stays unchanged at $\sim$16 scans which is plausible as it is constant regardless of sector. An amended velocity of -1 m/s results. These parameters are confirmed by the simulated waves' and correlation plots.

3.2 Correlation Coefficient Plots and Parameters for Logged Waves

For performing the above procedure on logged sea waves, a number of hurdles must first be overcome. These are confining the azimuth due to viewing constraints, setting range boundaries due to varying wave behaviour, establishing that the waves are free from actual targets, and interpolating the azimuth data. The *ArrayData* array and *AzimuthValues* matrix can be altered accordingly, subsequently allowing these calculations to proceed.

3.2.1 Confinement of the Azimuth

Acceptable azimuth confinements are those, which permit an unobstructed view of the sea for the radar. The measurement report of the logged data at Katwijk states that the radar was situated in an open coastal region, but with sand dunes on either side blocking the radar's view beyond. A photographic view of this with one of the sand dunes in shot is in Figure 3.5:

Figure 3.5: Photograph of sand dunes from radar's perspective.

The dunes can also be clearly seen in the Google-Earth aerial image in Figure 3.6. The inserted red lines and green angle labels determine azimuths of interest to be between -1.658 and -0.611 rad.

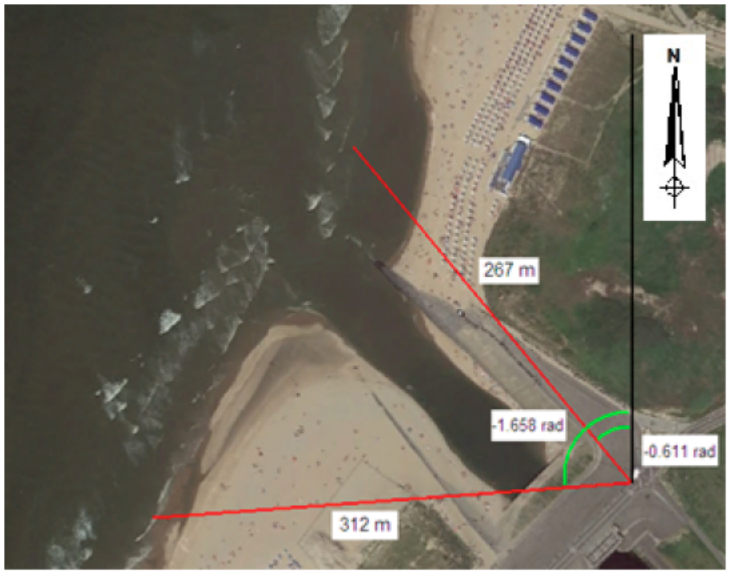


Figure 3.6: Aerial image of radar location with azimuth limits.

This explains the choice of angles for our standard radar plot of the logged waves, and reflects the orientation and size of these plots in this thesis.

3.2.2 Confinement of the Range

When dealing with logged sea waves, boundaries in the range dimension need to be determined. The first is the shoreline, as dry land is obviously of no concern here. Established from the aerial image, the shoreline extends to a maximum of approximately 312 m, equivalent to 57 range cells, which acts as the first boundary. Focusing on the sea beyond, there is a second range boundary. A less clear-cut boundary arises due to changes in wave behaviour as waves move from the open sea into the surf zone (Dutch: "branding zone"). This zone adjacent to the shoreline is where wave shoaling takes place – an increase in height caused by waves moving into shallower waters towards the upward-sloping seabed, a homogeneous bathymetry assumed on a small spatial scale [10]. This wave height increase can be great enough to bring waves to the point of collapse, or breaking.

We determine the end of the surf zone, or the start of the open sea, by examining the radar plot and establishing where the high signal breaking waves cease. A larger-range version of our standard radar plot, with the surf bordered by inserted black boundaries, is shown in Figure 3.7 below:

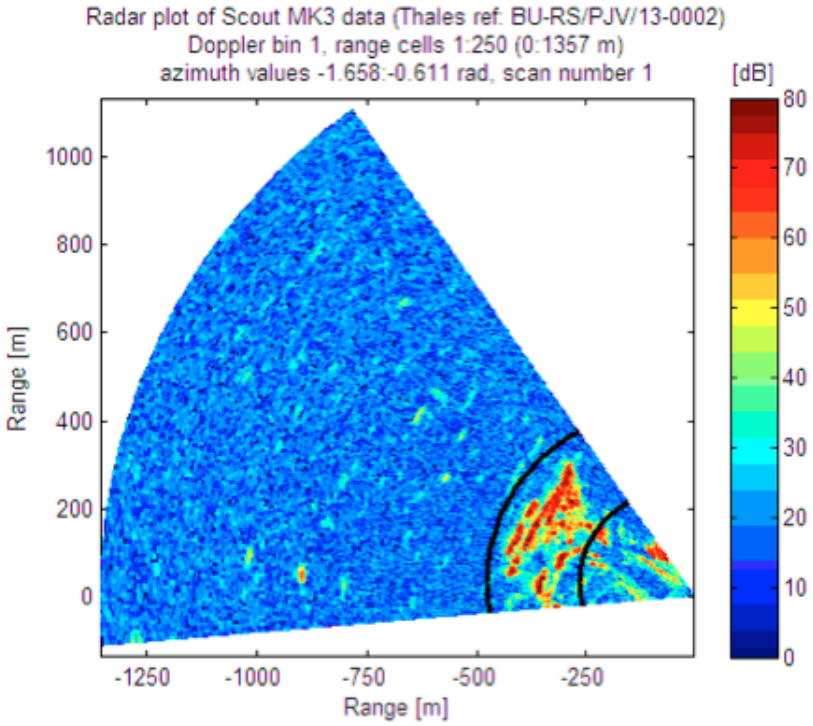


Figure 3.7: Surf zone boundaries.

As seen, the waves in the open sea beyond the surf are relatively calm and mostly not breaking. There are times in which also breaking waves occur and hence make determining the surf-to-open sea boundary more difficult – wind blowing against the direction of incoming waves provides calm open sea and is hence optimal for determining that boundary.

This suggests that the radar plot above originated from data gathered on an East wind day. To confirm this, the KNMI (Royal Netherlands Meteorological Institute) provides (Dutch) historical weather data. Ijmuiden is their closest coastal weather station, lying roughly 30 km north of the radar's location in Katwijk. Their downloaded chart in Figure 3.8 certifies that the wind was indeed blowing from

the East, availing the required knowledge of where the surf-to-open sea boundary lies for this coastal region.

Klimatologie

Daggegevens van het weer in Nederland

Kies station, jaar, maand, dag en druk vervolgens op de knop "toon"

IJmuiden sinds 01/01/1971 | 2013 | april | 9 | toon

Het weer op dinsdag 9 april 2013 te IJmuiden

Temperatuur		**Normaal**	**Neerslag**		
Gemiddelde	- °C	-	Hoeveelheid	- mm	
Maximum	- °C	-	Duur	- uur	
Minimum	- °C	-			
Zon, bewolking & zicht			**Wind**		
Duur zonneschijn	- uur		Gemiddelde snelheid	5.1 m/s	= 3 Bft
Rel. zonneschijnduur	- %	-	Maximale uurgemiddelde snelheid	8.0 m/s	= 5 Bft
Gem. bedekkingsgraad	- octa's		Maximale stoot	10.0 m/s	
Minimaal zicht	- km		Overheersende richting	91 °	= O
Relatieve luchtvochtigheid			**Luchtdruk**		
Gemiddelde	%	-	Gemiddelde luchtdruk	- hPa	

Figure 3.8: KNMI weather data.

Transcribing the pertinent information above: "a near-perfect East wind blowing at an average speed of 5.1 m/s".

Note that the surf is dependent on the bathymetry, or topography of the seabed below the sea surface. This is clearly a feature specific to each geographic location and thus the extent of the surf cannot be generalised for other coastal regions.

Finally, the tautological limit of the maximum range of the radar (6 NM or $\sim$11 km) acts as the outer boundary for the open sea.

To summarise, when dealing with logged sea waves, there must be a distinction between dry land and the sea as well as between the surf and open sea beyond – the two relevant range zones of the sea.

3.2.3 Logged Sea Waves Free from Targets

With the azimuth and range confinements now determined, the relevant area of sea can now be analysed. The first step is to ascertain that the data for this sea area contain waves only. This is necessary in order to ensure that we later calculate accurate predictor coefficients. Wave signal values polluted by any targets would distort predictor coefficients at this stage.

From the radar-image sequence of the scans, we see the movement of the waves towards the shore with little apparent interference. This supports the notion that in the area of interest there are sea waves only, with no interference from incidental targets such as boats, swimmers or other foreign objects.

Further supporting this line of reasoning are both the report's coastal photograph showing clear sea and reported date of measurement logging, April 9th, 2013, as this is neither sailing nor swimming season in this coastal region.

3.2.4 Interpolation of Azimuth Data

The raw data are treated and placed into *ArrayData* and *AzimuthValues*. We create arrays containing the correlation coefficients between sets of data that are displaced by Δ – a given number of indices along the four dimensions. For changes in any of Doppler, range and azimuth, we can readily use unaltered data. However, to achieve this in the case of ΔS, or a change in the scan number, we have a small challenge ahead.

This hurdle arises due to the readings from the radar giving measurements at "skewed" azimuth angles, that is to say that for a given azimuth index the azimuth values for two different scans can differ slightly. The main reason for this is that most radars, including ours, use several *PRF*s which in turn results in different burst widths. These are plotted in Figure 3.9, making the burst width changes apparent. Furthermore, even in the case with just one *PRF*, the angular rotation of the radar is not fully controlled and external physical factors such as wind mean that the burst width is not perfectly constant. Therefore, whenever comparing multiple scans, azimuths do not perfectly overlap and the data in the array must be interpolated [19] (Appendix B.10).

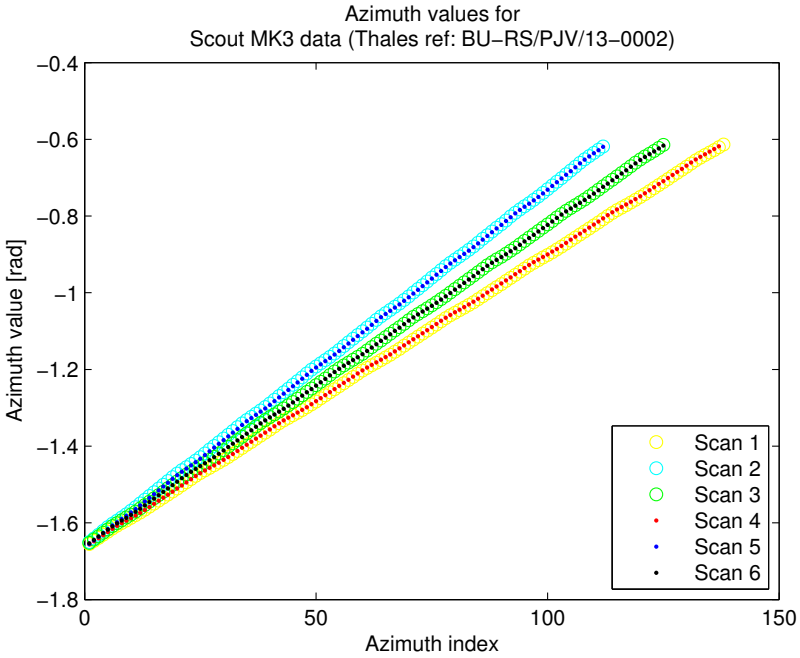


Figure 3.9: Azimuth index values showing three *PRF*s.

This interpolation is fitted to azimuth values defined by a grid starting from the highest of the first values of each scan to the lowest of the last values of each scan. This way we ensure that scans contain all azimuth values in the logged data. The framework consists of the azimuth values that are spaced out equally, and its total number of indices equals that of the scan with the largest amount of indices.

3.2.5 Areal Subdivisions for Prediction Coefficients

With both simulated and logged data, we calculate within distinct azimuth sectors, due to the varying statistical properties over the data. Let us add caution, due to the less predictable nature of logged

waves, where statistical properties may vary more greatly due to the topology of the seabed. So halfing the sector width to $\sim 5^\circ$, equivalent to ~ 10 logged data indices, is appropriate here. For logged data, additional boundaries must be set in to determine the surf zone and the open sea in the range dimension. These two should additionally be split up into subdivisions as even within range zones, statistical properties can vary due to uneven bathymetry. Sectors equivalent to ~ 50 m or ~ 10 logged data indices are used here as sufficiently small.

The resulting areal subdivisions have predictor coefficients valid for that subdivision only. They are shown in Figure 3.10 as a black grid in our standard radar plot, with the bold grid line marking the surf boundary:

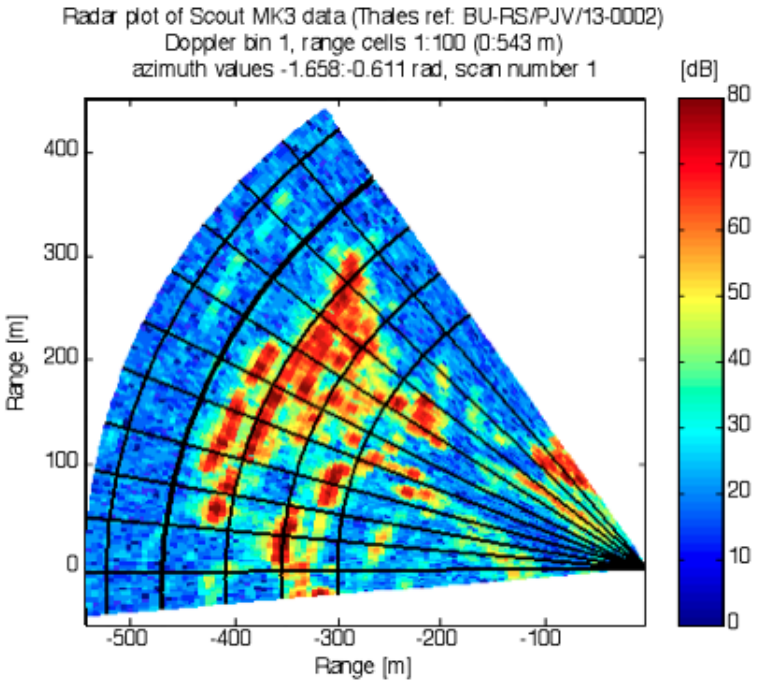


Figure 3.10: Areal subdivisions (for logged waves).

These areal restrictions introduce a new requirement. The number of indices in range, azimuth and scan multiplied must exceed the minimum sample size 1000 as was reasoned earlier. In this case, an area 10 range cells long and 10 azimuth cells wide means that 10 scans suffice to satisfy this requirement.

3.2.6 Correlation Coefficient Plots

Thus obtaining correlation coefficient plots B.21 (see Appendix) for the logged data yields in Figure 3.11:

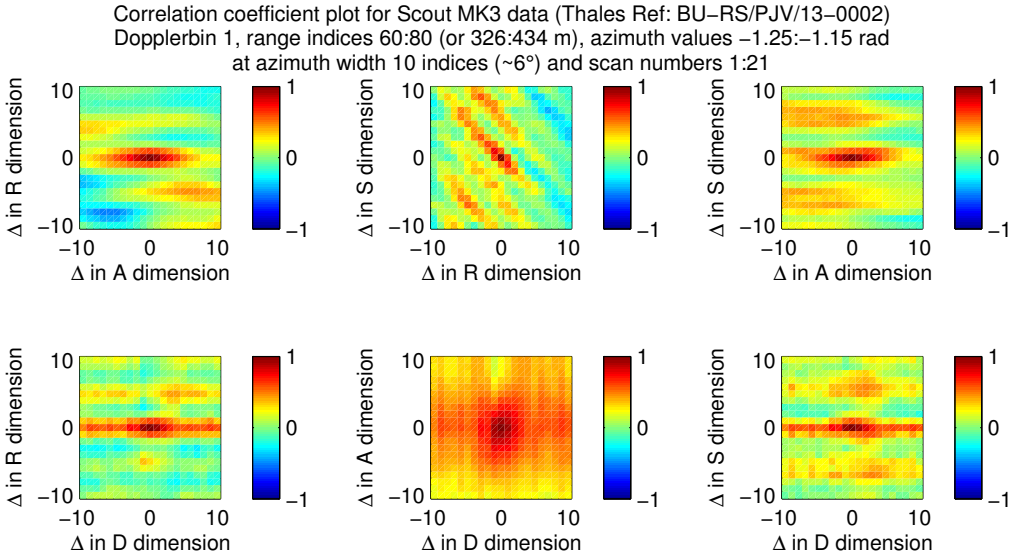


Figure 3.11: Correlation coefficient plots (for logged waves).

Shown in the upper row are again the various correlation patterns for each set of dimensions range, azimuth and scan of the data. Though not as clear-cut as with the simulated waves, where negative correlations at the wave trough are not as strong, they nevertheless reveal wave parameters. Firstly the azimuth-range plot shows the wavelength as the vertical distance between high correlations, which we observe as $\sim$6 range cells, equivalent to $\sim$32.5 m. Next is the range-scan plot, which shows both the wavelength and the period of the wave – the horizontal and vertical distance between similarly valued correlations respectively. This confirms the wavelength's just observed value and sets the wave period at $\sim$6 scan indices, equivalent to $\sim$9 s. The following azimuth-scan plot confirms this period via the vertical distance between coinciding correlations. From this discovered information, we find that the wave velocity has the plausible magnitude:

$$\frac{32.5}{9} \approx 3.6 \text{ m/s } (\sim 13 \text{ km/h}).$$

Both the wavelength and velocity found here corroborate earlier findings (cf. Section 1.2).

The lower row shows the correlation coefficients of the Doppler dimension with the three other ones. The range and scan dimension with respect to Doppler confirm the calculated period and wavelength information, whereas the Doppler-azimuth alone reveals little useful information. Generally, for these plots the simulated target is designed to reflect a real-life target where all parts move at constant velocity. For this reason, we do not expect much correlation for targets along the Doppler axis. A wave, however, has parts moving with different velocities. Therefore, while the waves have spread in the Doppler dimension so that some correlation can be expected for them, this is not the case for the targets.

3.3 Dispersion Relation

In fluid dynamics, the dispersion relation provides a relation between the angular frequency and the wave number depending on the seabed or water depth. In [10] the wave velocity was used to determine

the shallow water bathymetry, as the wave velocity directly depends on the water depth h by the dispersion relation. At this stage, we can emulate this by taking the earlier-calculated wave parameters and determining if they concur with plausible water depths.

The dispersion relations, dependent on various water depths, are given in Figure 3.12 from [8] and Table 3.1, where:

Deep water, or the open sea beyond the surf: $h > \frac{\lambda}{2}$;

Shallow zone: $h < \frac{\lambda}{20}$;

Transition zone: between the two.

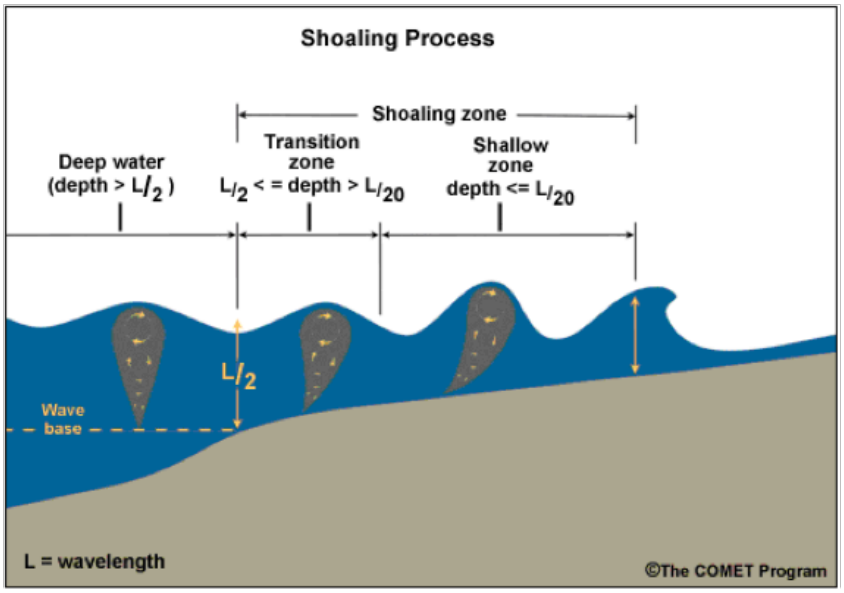


Figure 3.12: Shoaling process (Source: COMET®Website at http://meted.ucar.edu/ of the University Corporation for Atmospheric Research (UCAR)).

Quantity	Symbol	Units	Deep water	Transition zone	Shallow zone
Dispersion relation	$\omega(k)$	rad/s	$\sqrt{gk} = \sqrt{\frac{2\pi g}{\lambda}}$	$\sqrt{gk \cdot \tanh(kh)} = \sqrt{\frac{2\pi g}{\lambda} \tanh(\frac{2\pi h}{\lambda})}$	$k\sqrt{gh} = \frac{2\pi}{\lambda}\sqrt{gh}$

Table 3.1: Dispersion relation table.

Earlier, the wavelength for the logged waves was observed to be ∼32 m. The dispersion relation for the transition zone is valid for the other zones and is applicable at a water depth in this region estimated to be between one and few metres. By the table's dispersion relation and by the relation for the wave's angular frequency, we can find the wavelength in terms of either the angular frequency or wave period:

$$\frac{2\pi}{T} = \omega = \sqrt{\frac{2\pi g}{\lambda} \tanh(\frac{2\pi h}{\lambda})}$$

$$T = \frac{2\pi}{\sqrt{\frac{2\pi g}{\lambda}\tanh(\frac{2\pi h}{\lambda})}}.$$

While the wave period stays constant, the water depth and wavelength vary depending on location. So, using the Newton-Raphson method [17], we plot water depth h versus wavelength λ at fixed T (the estimated value 9 s and neighbouring integer values):

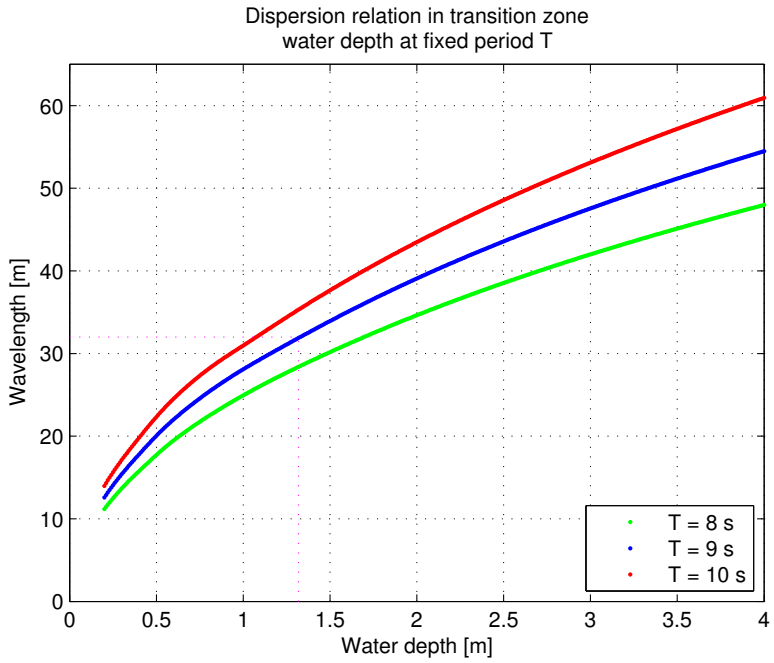


Figure 3.13: Dispersion relation.

The earlier estimated wave period 9 s, shown in blue above, results in the magenta-dotted water depths of $\sim$1.3 m – a credible depth and within the earlier estimation. Close period values, red and green above, result in similarly convincing water depths of 1 to 2 m. To conclude, the dispersion relation confirms the plausibility of the observed wavelength for this coastal region.

3.4 Residual Signal Plots

With the basis of simulated and logged waves and their areal subdivisions established, we obtain predictor coefficients for each of the subdivisions. We then inject targets and find the residual signal, the absolute difference between the predicted and true dB signal, for each cell of interest. A plot of all residual signals will startle observers as to the likely location of targets.

3.4.1 After Linear Prediction

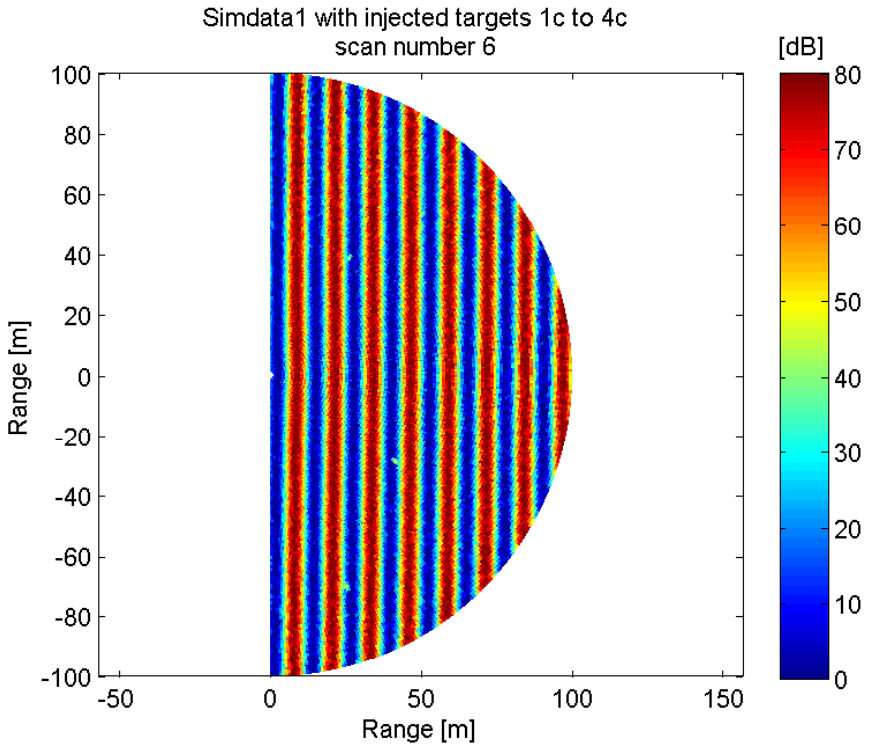
The first step is to elect 66 Y predictor cells, our earlier-reasoned choice, and calculate the corresponding predictor coefficients (Appendix B.11) for each of the subdivisions of a waves-only setting. B.12

calculates this for multiple subdivisions where its required inputs include surf boundaries, maximum range length and maximum azimuth width.

There are restrictions on the choice of cells of interest. Cells that are within a two-cell distance from data limits lack at least one corresponding predictor cell – "invalid" cells of interest. This is certainly true for the azimuths, which are absolutely limited. The range dimension, however, can have data extending beyond the area of interest. In this case, the desired quantity of cells of interest should be two less than the data at hand. This boundary delimitation B.22 (see Appendix) can be seen in white in the residual signal plots that follow. A further restriction occurs due the predictor cells shifted by one scan before or after the current one, meaning that the first and last scan must be sacrificed.

As repeatedly cautioned throughout this thesis, there is a hazard of calculating correlation structures for areas within which statistical properties may vary. The limited size of the subdivisions in range and azimuth were designed to counter this. Predictor cells, however, can cross the edge of the subdivisions by two cells. Since this shift is relatively small, and the subdivision size has a degree of haziness and subjectivity, it is unlikely to cause serious discrepancies.

Follow-up B.13 (see Appendix) scans through the same setting of waves containing incidental targets, and linearly predicts all cells of interest via the just calculated predictor coefficients. Those that lack predictor cells are assigned value zero, whereas the first and last scans are simply omitted. The output of residual values is a four-dimensional array of equivalent size but with two fewer scans. In the same way as the actual signal data, a sample scan of these showing the sea area virtually decluttered can be seen in the centre plot of Figure 3.14, and the waves with injected targets and bare targets only for comparison:



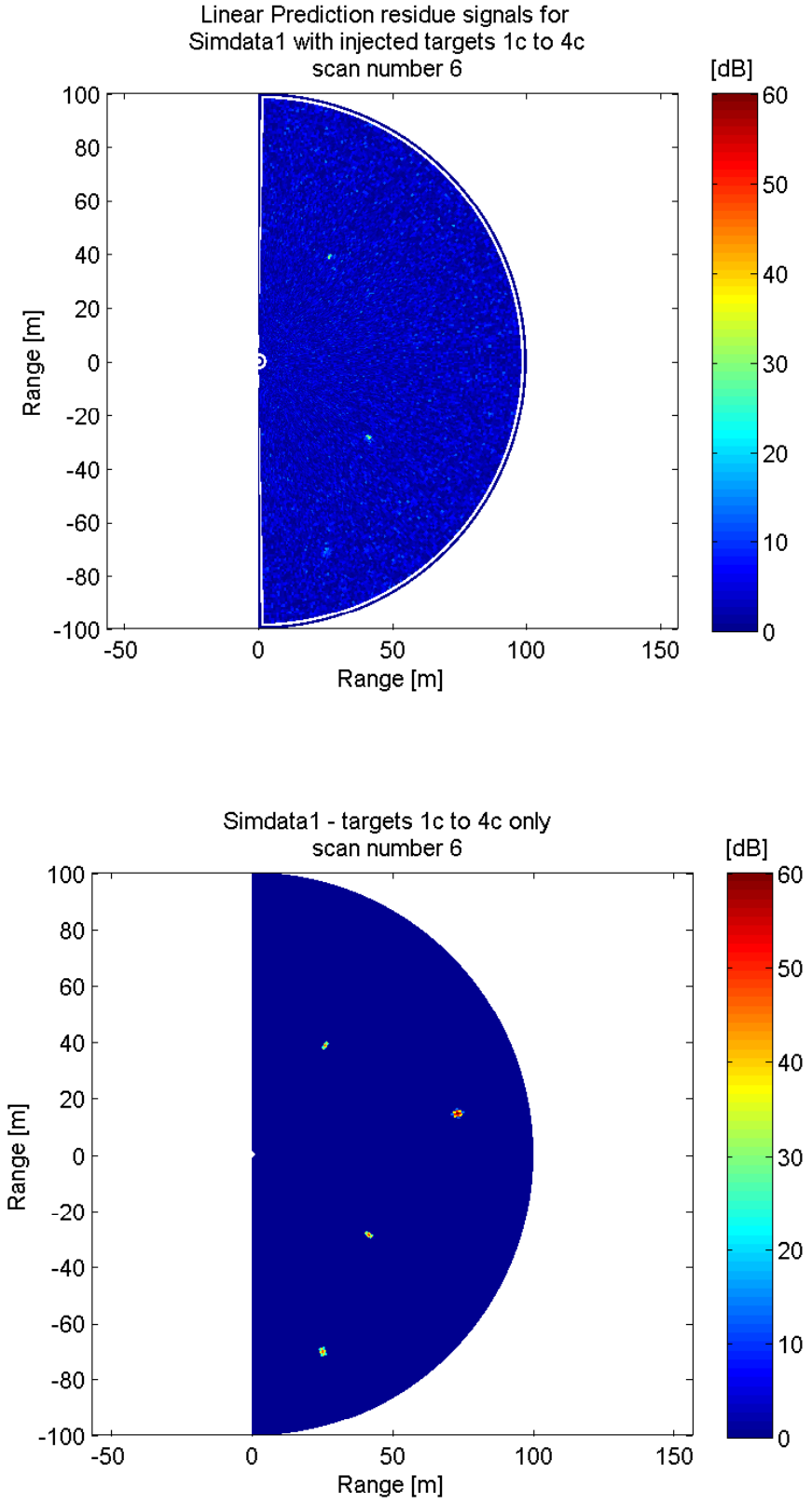


Figure 3.14: Waves with target, residual signals and bare targets (for simulated waves).

Most notable in the residual signal plot are two cell clusters containing high residual signal. Such clusters should indicate the likely presence and size of a target. This provides a practical, albeit subjective, visible check of a sea area.

The outer edge values are assigned value zero due to the predictor cell shift, with white boundaries fencing off these "invalid" cells. Furthermore, time shifted predictor cells have the repercussion of the first and last scans being omitted in the radar image sequence.

A comparison of the plots shows that the two targets are indeed located in the correct position as anticipated. There are however four injected targets! A radar-image sequence of all residual signal scans shows that four targets are indeed visible, but by varying degree – even disappearing in certain scans. This brightening and fading process is illustrated in two two-dimensional scan-azimuth and scan-range residual signal plots B.14 (see Appendix) below, where targets are programmed to move along a fixed azimuth, or range respectively:

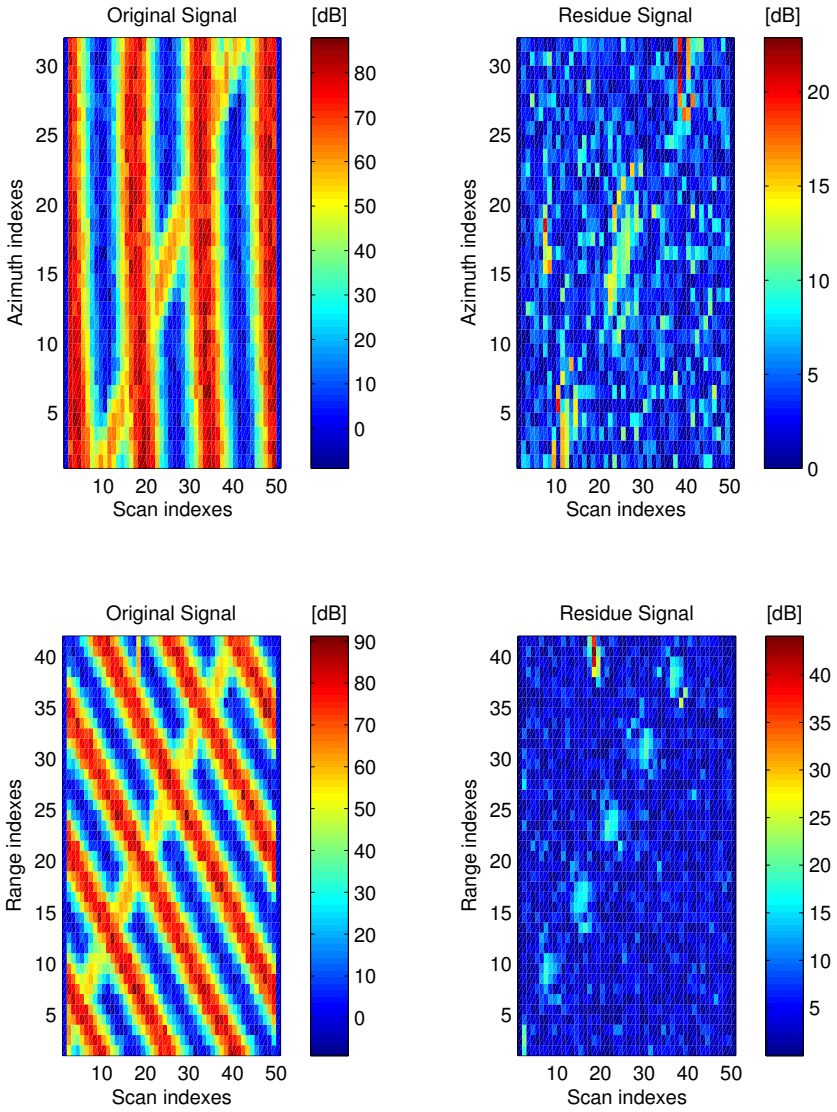


Figure 3.15: Residual signals after Linear Prediction.

Both left plots in Figure 3.15 show the actual, or true, signals of the simulated waves including the targets, with waves peaking at ∼80 dB and targets at ∼60 dB, and both right ones show the residual signal with colour axes adjusted for enhanced visibility. From these, the target visibility can be seen to fluctuate along the scan dimension, in some scans even invisible. A comparison of either shows on the one hand that when located at a wave trough, the target stands out prominently against low signal background, appearing clearly in the residual signal plot. If on the other hand, a target is positioned at the wave peak, then it is camouflaged among the high signals, rendering the target lost. To remedy the potential problem of not detecting so concealed targets, a sufficient number of scans is recommended.

Next, we proceed with residual signal plots, again via 66 predictor cells, for one sample scan. The residual signal plot is shown on the left, with bare targets only on the right for comparison (with added light-grey reference grids henceforth to facilitate position comparison) in Figure 3.16:

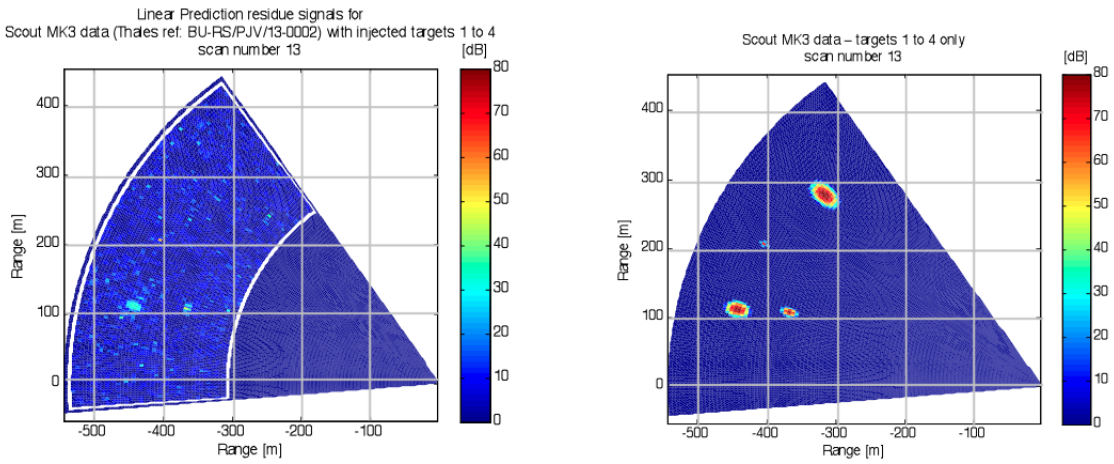


Figure 3.16: Residual signals after Linear Prediction.

As before, cells of interest must be at least two range or azimuth cells away from data limits. Additionally they must here maintain this distance from dry land, as our interest lies in activity on the water only. The white boundaries border the "valid" cells, as seen above.

Obviously, the left residual signal plot has a greater degree of background noise than the simulated environment. It shows clusters of higher signal cells and a single orange-coloured cell. A comparison with the right plot shows that the spatially smallest target in particular corresponds to this high residual signal. The other targets are not discernible in this manner.

3.4.2 After Sequential Decorrelation

B.15 (see Appendix) assists in finding predictor coefficients and corresponding Y signal locations for a specific area, in our case an 11-by-11 azimuth-range subdivision, with time-shifted predictor cells. Via the correlation matrix we perform k decorrelation steps, where the correlations decrease with each step. The process continues until all predictor cell correlations are below a sufficiently low threshold (magnitude 0.1). The final correlation matrix is shown in Figure 3.17, where the k decorrelated predictor cells are assigned white colour:

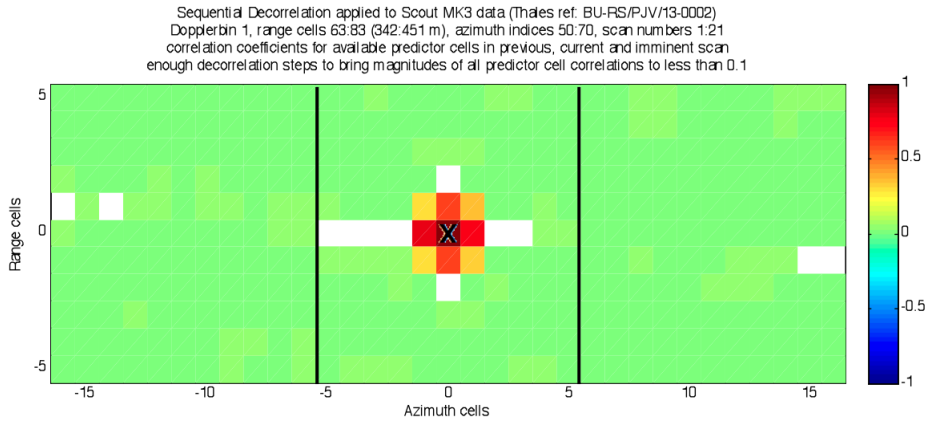


Figure 3.17: Correlation coefficient decrease after Sequential Decorrelation.

In this instance $z = 10$ steps lead to sufficiently low correlations of maximum magnitude 0.1. Decorrelated signals $Y_1, Y_2, ..., Y_z$ all have zero correlation with the cell under test.

Applying this principle to multiple subdivisions (B.16) yields the predictor cells and coefficients. The follow-up B.17 provides an array of predicted data based on the just calculated cells and coefficients. So predicting the simulated waves with time shifted predictor cells and 11-by-11 subdivisions yields for the same sample scan as previously used:

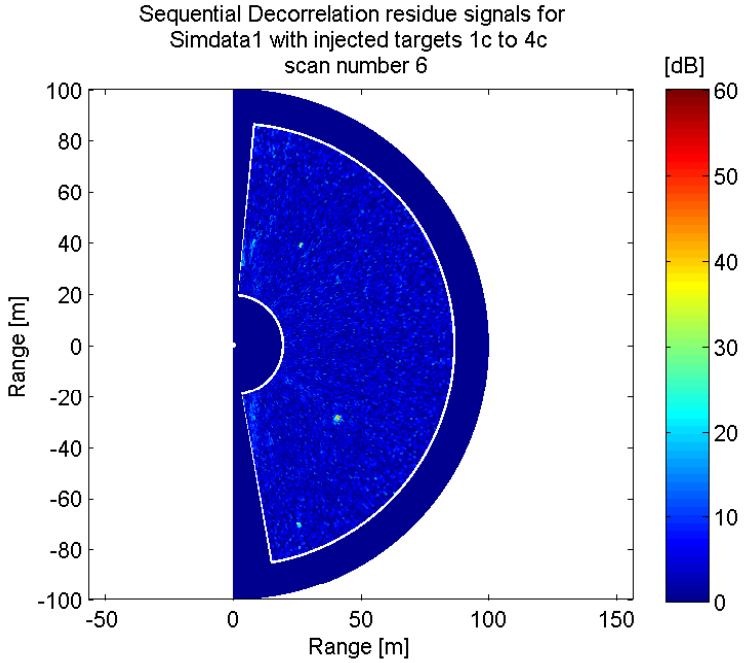


Figure 3.18: Residual signals after Sequential Decorrelation.

Note that the earlier-mentioned restriction on cell of interest locations is tightened with this method. As cells of interest can now have predictor cells shifted by up to five cells in range or azimuth, this sets the minimum distance from those dimension limits to at least five. Furthermore,

the covariance matrix calculated for each predictor cell requires additional "space" of five cells (when constructing an 11-by-11 covariance matrix). Combined, this means that valid cells of interest must be at least 11 cells from range azimuth limits, in addition to, as before, not originating from the first or last scan. As a consequence, white borders are more delimited, i.e. the field of vision has shrunk.

Next, we apply this approach on the logged waves. Taking the same sample scan used in the former method yields:

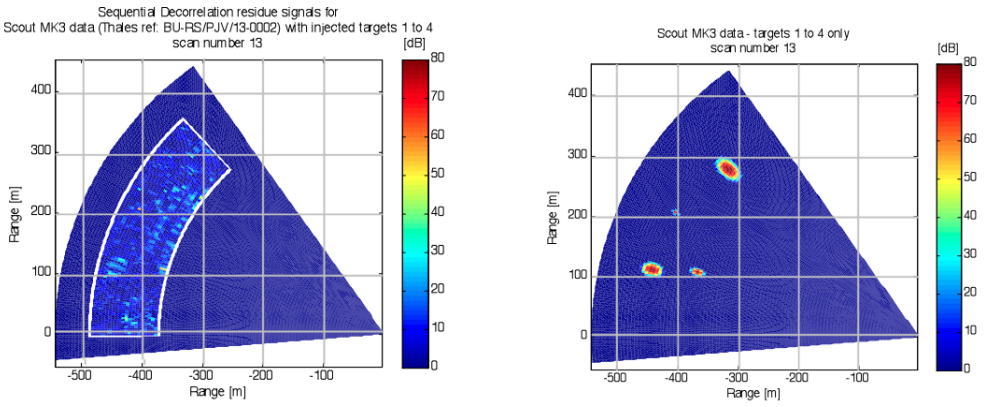


Figure 3.19: Residual signals after Sequential Decorrelation.

The left residual signal plot shows a greater degree of background noise than under the previous model. Nevertheless, there are now two orange cells, or small clusters of such, discernible. A comparison with the right plot shows that these correspond to locations of the two spatially smallest targets. Indeed, the radar-image sequence shows that the smallest targets appear pronounced over the largest ones. We take note of this observation.

3.5 Determining Target Presence

Thus far, visual checks of the residual signals have ascertained the target presence. Now strategies to make this determination more objective are explored – the null hypothesis being that the cell contains background waves only, with the alternative hypothesis meaning target presence.

The first step is to set a threshold for the minimum residual signal – a compromise between being high enough to reject negatives and sufficiently low to accept positives. A false alarm rate is needed – 10^{-6} in our case as reasoned earlier.

The second step is to find all cells of interest whose residual values exceed the threshold – in which case target detection occurs. A 1-0 colour plot then highlights such detections – objectively startling the observer to likely target locations.

3.5.1 After Linear Prediction

Several scans may be necessary to achieve the million cell criteria. In the example for the simulated waves, with 96 range and 311 azimuth available cells of interest per linearly predicted scan, the total

number of scans required is:

$$\frac{10^6}{96 \cdot 311} = 33.5.$$

Among all residual signals of 34 scans, the maximum outcome is 20.18 dB – a plausible magnitude for highlighting. Taking the trails of the detections B.18 (see Appendix) for the second to the penultimate scan (due to time shifted predictor cells):

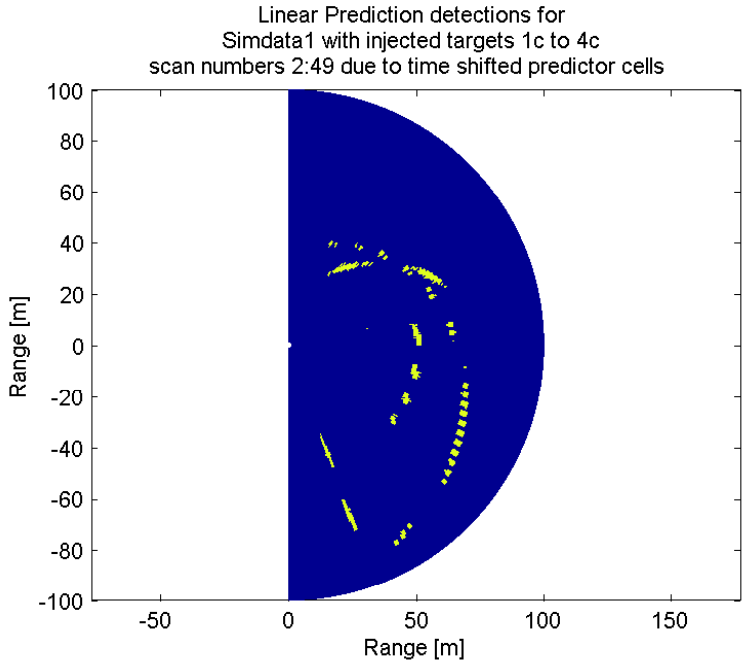


Figure 3.20: Detections for Linear Prediction.

This treatment yields startling detections. Trails of targets are clearly visible. A comparison with the trail of targets from earlier shows that they correspond to true positions, save for a single solitary outlier at approximate coordinates (30,5) as expected.

Continuing with setting the threshold for the Linear Prediction of the logged data, the threshold is established to be 49.9 dB. Note that the limited quantity of the logged data, merely $39 \cdot 124 \cdot 19$ (for ranges, azimuths and scans of the) cells of interest, has the repercussion of a false alarm rate of $\sim 10^{-5}$. With this data setting $\sim$200 scans would be necessary to reach the desired false alarm rate.

Figure 3.21 contains the resulting plot of detections:

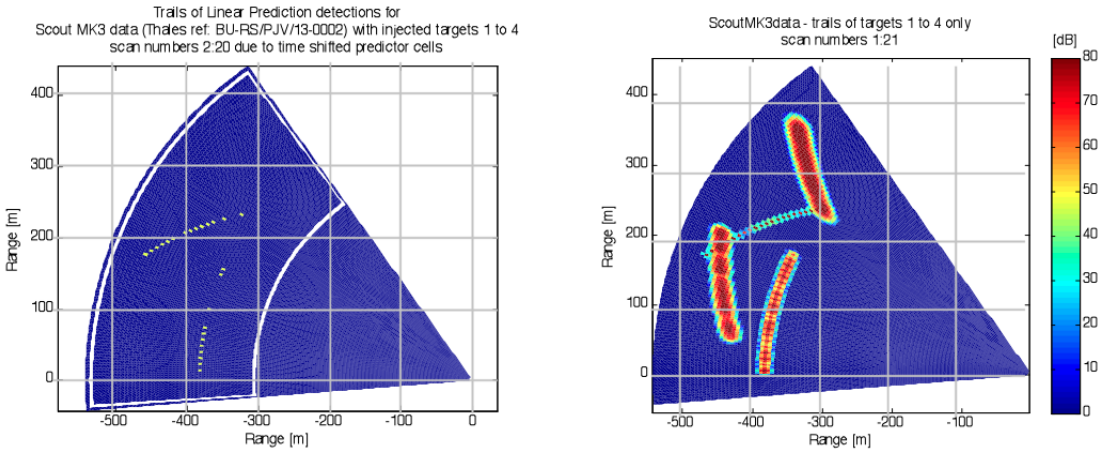


Figure 3.21: Detections for Linear Prediction.

From the above, two trails of detections are visible. A comparison with the right plot shows that these correspond the two smaller targets. The two larger targets intriguingly are not detected at all, despite even being partially located in the wave-free area beyond the surf.

3.5.2 After Sequential Decorrelation

Unfazed, we proceed with creating thresholds for the residual signals resulting from the second method. First for the simulated waves, the threshold is 36.1 dB. Figure 3.22 shows a plot of detections:

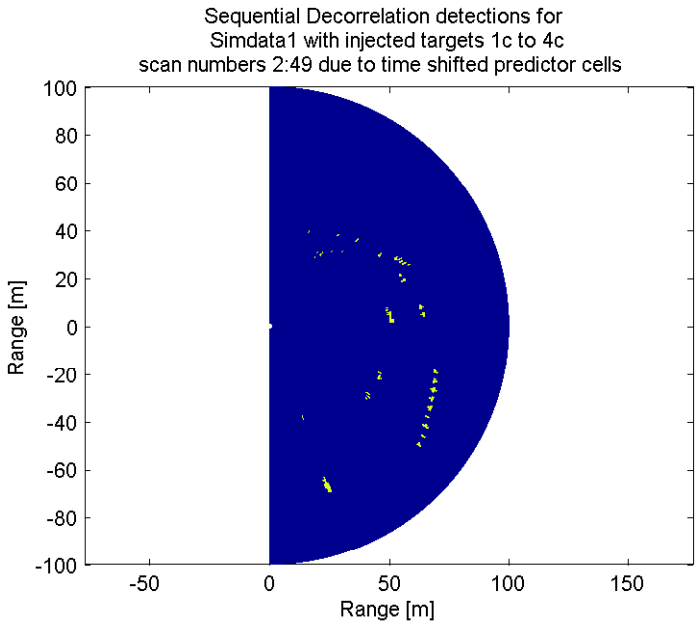


Figure 3.22: Detections for Sequential Decorrelation.

The above shows the trails of the target detections again appearing and disappearing as expected. The detections are accurate by comparison with the trails plot of the bare targets only. Among the detections is there are no apparent false alarms.

Proceeding with the logged waves, the threshold is 55.2 dB, calculated from a lesser cell of interest quantity: $21 \cdot 116 \cdot 19$, meaning a larger false alarm rate of $\sim 2 \cdot 10^{-5}$. Figure 3.23 shows the previous sample scan and the trail for all scans, for both the detections and bare targets only:

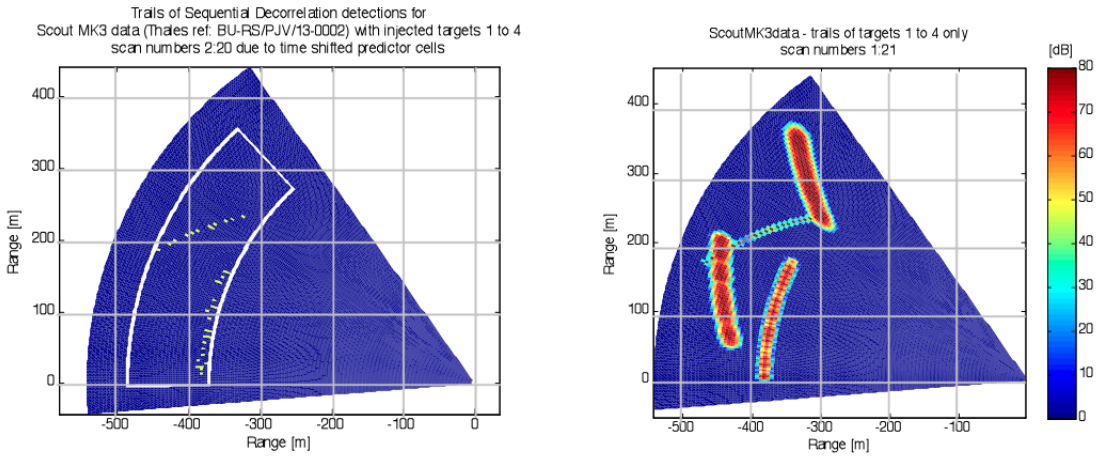


Figure 3.23: Detections for Sequential Decorrelation.

From the above, the spatially smallest target, i.e. the thin trail in the right plot, is recognisable on the left. The spatially second-smallest target is also recognisable, however the position is frequently skewed and less accurate than the other target. Again intriguingly, the two largest targets are not detected at all, despite being partially located at a near wave-free setting.

There is an unclear bias for small targets being more easily detectable. Referring to Section 2.1.2 where the target signals for each cell become visible, it is reasonable that the chosen guard cells should protect against unwanted overflowing target signal, to prevent the previously-cautioned principle of self-killing. Targets should thus be resized to be sufficiently spatially small (1-by-1 range and azimuth for an 80 dB target). Testing this hypothesis, we create four new targets of aforementioned size, ones that are entirely among the waves throughout, and rerun the process. Figure 3.24 shows the result for renamed targets *1a* to *4a*:

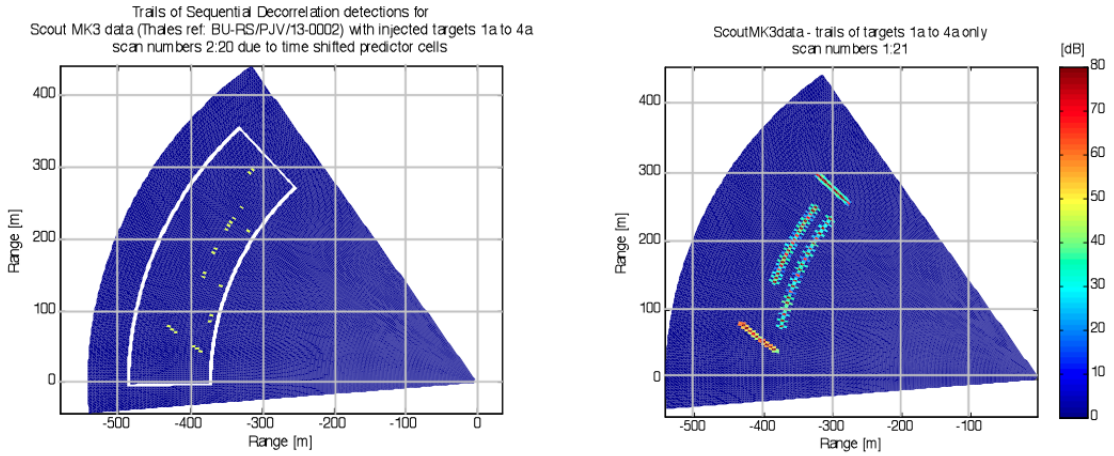


Figure 3.24: Detections for Sequential Decorrelation with new targets.

Now the four trails in the right plot of bare targets are highly recognisable on the left. They are visible at varying degree, with the most visible one attaining 9 detections, and the least visible 2 detections.

3.6 Comparison of Prediction Models

For completion, we firstly perform Linear Prediction on the updated targets *1a* to *4a* in Figure 3.25:

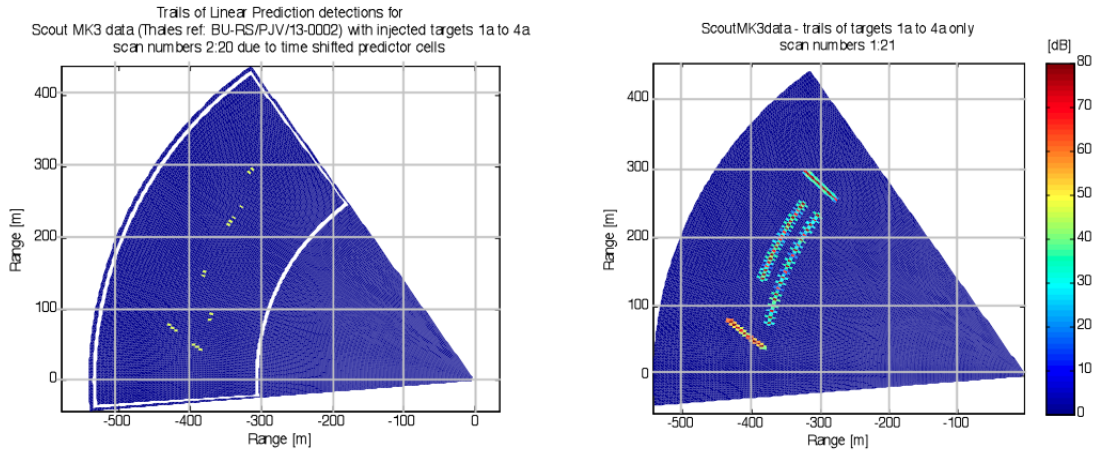


Figure 3.25: Detections for Linear Prediction for new targets.

Now these updated targets can be seen at varying detection levels. Further there is a noticeable qualifying contrast between the two methods for each of the wave environments. Applied to the simulated waves, the simpler Linear Prediction is more useful at locating targets. The nature of the simulated waves means that the previous scan shows similar behaviour compared to other scans,

with wave peaks appearing consistently over several scans. This leads to Sequential Decorrelation's inferiority in this environment.

The logged data yield opposite results for the methods. While Linear Prediction is constrained into choosing predetermined predictor cells, Sequential Decorrelation has the great benefit of picking those cells with the highest predictive capability. This means that in an environment of sea waves of less predictable nature, a prediction model that picks predictor cells with highest predictive capability yields more accurate results than one with a possibly unwise choice of predictor cells. This leads to the latter method's superiority.

Both methods show preference for spatially small targets. Extended targets tend not only to have less detection, but often at inaccurate positions. The reason for this is that the extended targets' signal overflows beyond the guard cells (cf. Section 2.1.2) and the predictor cells contain target signal, which erode their predictive capability. Our model is therefore a filtering mechanism designed for spatially small targets – 1-by-1 cell (in range and azimuth) at 80 dB or 2-by-2 cells at 60 dB – in short those that keep within the guard cell confinement.

A distinct advantage of Linear Prediction is that more area of sea can be predicted, as the "space" required by the chosen predictor cells is less than in Sequential Decorrelation. When space is tight, as it is here, that can be quite disadvantageous.

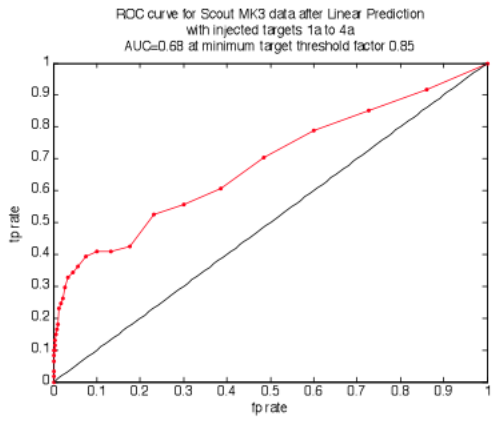
Now, in the upcoming two subsections, we mathematically quantify these findings.

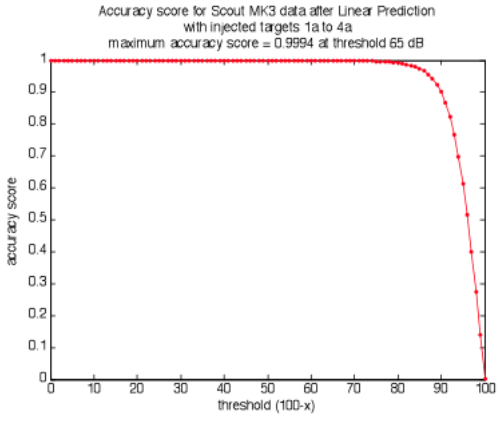
3.6.1 ROC Analysis and Accuracy Score

Here we address the discrimination and calibration via the two mentioned concepts. We begin by plotting the ROC (Receiver operating characteristic) curve for each of our prediction models. The diagonal – the expectation from random guessing – is shown in black for reference. To reiterate, the more the curve tends to the upper left hand corner, the larger the *AUC* (Area under curve) and the better the classifier.

While in theory we start at detection threshold infinity, B.19 (see Appendix) starts at threshold 100 dB and decreases steadily to 0, meaning 100 points are plotted. We can also alter the other threshold which determines target presence, in turn resulting in a different *AUC* value.

For the logged waves the top row of Figure 3.26 shows for Linear Prediction the ROC curve on the left and on the right the accuracy score for each threshold. The bottom row shows the same for Sequential Decorrelation:





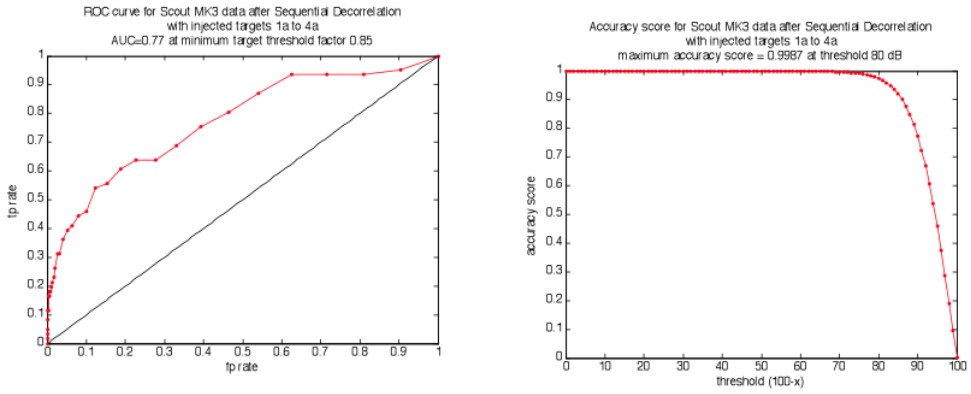


Figure 3.26: ROC curves at threshold factor 0.85 and accuracy scores.

These plots were found at minimum target threshold factor 0.85 – i.e. if the signal is above this fraction of maximum target signal (80 dB here), then a target is determined to be present in reality. For other values, the ROC curve and *AUC* vary accordingly in Figures 3.27 and 3.28:

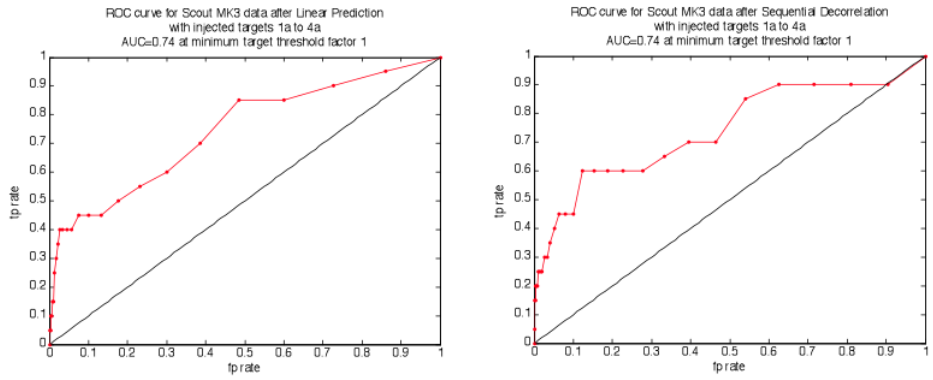


Figure 3.27: ROC curves at threshold factor 1.

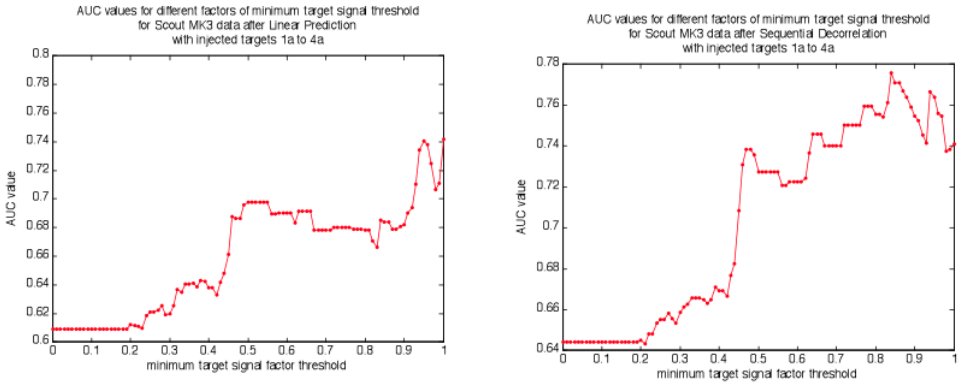


Figure 3.28: *AUC* values at various minimum target threshold factors.

From the above plots, it is apparent, that factors 1 for Linear Prediction and 0.85 for Sequential Decorrelation lead to the highest *AUC* in each respective method. The corresponding ROC curves are visually pleasing, as outlined in the theory section description. The *AUC* confirms this numerically with values 0.6-0.7 being in the realm of "sufficient" and 0.7-0.8 in the realm of "good" [27].

We can compare the maximum *AUC* values of the two methods defined as AUC_{LP} and AUC_{SD} via factor:

$$\frac{AUC_{LP} - AUC_{SD}}{AUC_{LP}} = \frac{0.74 - 0.77}{0.74},$$

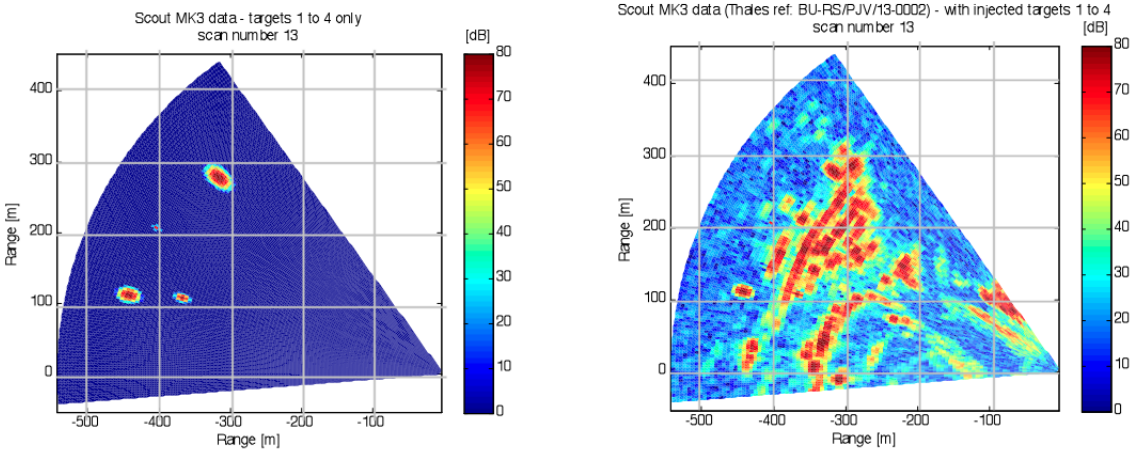
which is a factor of approximately 0.04. Regarding the accuracy A_{acc} – at highest *AUC* – for the former method it is value 0.9994 at threshold 65 dB. The latter method's value is 0.9987 at threshold 80 dB. The accuracy values resulting from our earlier-chosen thresholds based on false alarm rate 10^{-6} are 0.9990 and 0.9987 respectively showing that the difference is practically negligible, and therefore we can be comfortable and confident with the chosen false alarm rate and corresponding threshold.

Chapter 4

Conclusion

The two prediction models yielded qualititatively different outcomes for each of the two environments. The first Linear Prediction model provided a good visible detection method for targets injected into both simulated and logged waves. Limiting detection above the threshold gave accurate target locations. By its nature, this method requires less space for predictor cells, and thus more cells can be predicted – a distinct benefit.

The second model Sequential Decorrelation provided good results. For simulated waves, the targets were again detectable by both a visual check and by tweaking the threshold. However, this was at times less prominent and accurate than the former method. For logged waves, the ROC curve and *AUC* value indicated this method's superiority. The value was maximised for minimum target threshold detection factors 1 and 0.85 respectively. The succession of waves with injected targets, bare targets for comparison, residual signals after prediction, and detections for the radar plot is shown in Figure 4.1:



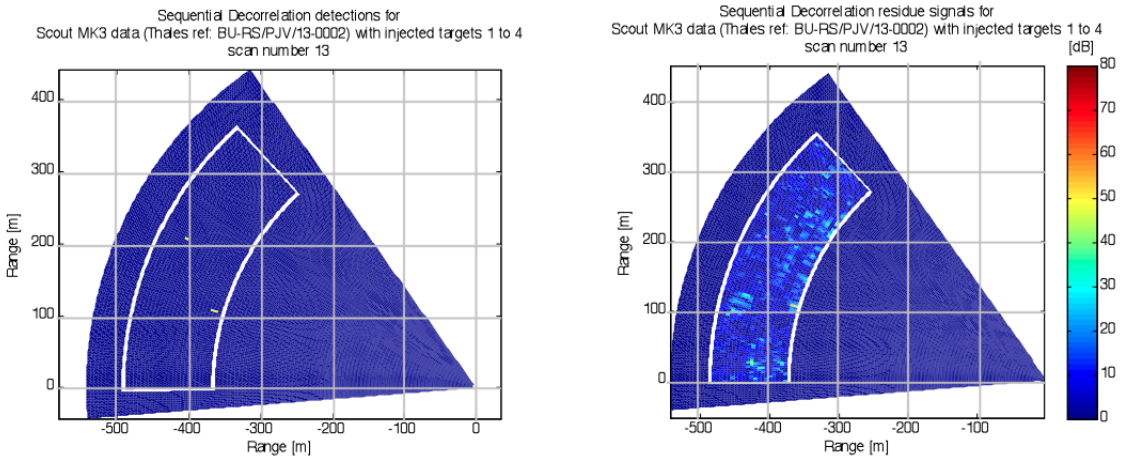


Figure 4.1: Bare targets, waves with targets, residual signals and detections.

This recapitulates (clockwise) the complete process of filtering out the waves, coaxing the targets out from among the wave crests, but also the shrunk field of vision due to predictor cell shifts. The first filtering step, the residual signals, reveals among considerable background noise several high signal cells. A comparison shows that these correspond to true target locations. The second filtering step does a fair job at eliminating the noise, highlighting in this scan the two spatially smallest of the total four targets at the true positions.

This is perhaps the key finding of this thesis – the principal suitability to detect simulated targets (of Gaussian profile and Swerling I model for signal fluctuation) among logged waves, especially spatially small targets (1-by-1 and 2-by-2 cells in range and azimuth) up to dB signal strength equal to that of the wave peak. The self-calibrating Sequential Decorrelation prediction model achieved this best using confined areal subdivisions (20 range cells long, equivalent to $\sim$100 m, and 20 azimuth indices wide, equivalent to $\sim 10^\circ$), with predictor cells shifted by up to one scan. Subsequent threshold setting, at false alarm rate 10^{-5}, then gave a reliable radar plot of target detections.

"Remember that all models are wrong, the practical question is how wrong do they have to be to not be useful" – George E. P. Box [3]

4.1 Applications and Uses

The practical benefit of this research is to be able to filter out waves in the radar plot of a sea area and reveal targets located there. These targets can include small boats, swimmers out to sea (after a maritime disaster for instance), sea birds and aquatic animals, marine debris and floating objects such as lost shipping containers and even old sea mines. This is surely a highly valuable resource for coast guards, wildlife surveyors, and navies – not least of commercial value to Thales company.

4.2 Limitations

With both methods, there was a restriction on the choice of cells of interest due to the predictor cell shifts. This requires a sufficiently large data set in order to both maximise the amount of valid cells of interest and ensure a sufficiently high false alarm rate. While the simulated waves fulfilled the usual

false alarm rate in this field of study of 10^{-6}, the limited nature of our logged data meant that we were dealing with ones of only $\sim 10^{-5}$ and $\sim 2 \cdot 10^{-5}$ for each of the methods.

Furthermore, the recommended minimum number of scans to avoid the risk of targets becoming undetectable was emphasised. An additional and more rigorous scan quantity criterion was due to the constrained areal subdivisions, which required an expanding scan dimension to accommodate the necessary cell sample size (1000 giving a 4.5% accuracy).

All measurements here were taken from a stationary radar on dry land. Radars fixed on a moving host such as a ship or helicopter would bring new challenges, as parameters such as the host's speed must then be taken into consideration, as well as questions of resolution.

A final word of caution: working with uncorrupted signal data is best practice. At times, radar signal data can be wrongly stored, configured or otherwise polluted, by high signal demarcation lines for instance. These can skew predictor coefficients and hence frustratingly harm the detection capability.

4.3 Alternative Model Approaches and Outlook

In this thesis, several mathematical models were used to mimic actual target signal behaviour. The simulated target was assigned a Gaussian profile, though alternative profiles of symmetric probability distributions, such as the Cauchy, Logistic and von Mises, may also be suitable imitations. In addition, the Swerling I model served to mimic target fluctuating signal amplitude, though the alternative Swerling II, III and IV [4] could be worth exploring.

Further strategies for highlighting target locations, such as taking the exponential of residual signals may be effective.

For solidifying the findings of this thesis, useful research would be to test the detection procedure with real life targets, with differing physical properties even rather than merely simulated ones.

Furthermore, testing this procedure on several sea geographic locations (with differing surfs and wave behaviours) would reinforce the findings of this thesis, making them general enough to apply ubiquitously.

In order to determine the surf for these various locations, a valuable resource would be a correlation structure between the extent of the waves in the surf on the one hand and on the speed and direction of the wind (historical, due to fetch and swell) on the other. This would involve a large sample size of measurements from different days together with accurate weather information – either from weather stations nearby or with simultaneous use of wind vanes and anemometers alongside the radar [12].

Acknowledgements

I would like to extend my gratitude and sincere thanks to Emiel Stolp, my supervisor and mentor at Thales company, for patiently explaining the mathematical and physical aspects of radar technology, as well as to Cristian Spitoni, my supervisor offering helpful advice here in Utrecht. Secondly, to my parents for consistently and unabatedly supporting me throughout my endeavour. Finally, "dank jullie wel" to my fellow interns at Thales, and those at the company's "Volonta" student social scene, for keeping me social and happy during my time in Hengelo.

On a more profound level, it is an honour and a privilege to be furthering our knowledge and understanding of the physics of sea wave behaviour through applied mathematics. It is thanks to the intellectual foundation laid previously (see Bibliography) that this research was possible.

Appendix A

Radar:

SCOUT MK3, FMCW (Frequency-modulated continuous-wave)-Doppler radar

Operational performance and technical characteristics:

Selectable range scales	6, 12 & 24 NM
Minimum range	< 15 m
Range accuracy	1 m (at 6 NM)
Azimuth accuracy	0.15° (at 10/20 RPM)
Doppler accuracy	0.5 m/s (at 10 RPM)
Rotation speed	10, 20 and 40 RPM
Frequency band	I/J-band (17 frequencies, operator selectable)
Beamwidth	1.2 degrees horizontal, 20 degrees vertical
Transmitter output power	10 mW up to 3 W (operator selectable)
Clutter suppression (SCIF)	> 60 dB

Logged data:

SCOUT MK3 trials i.s.m. KNRM Katwijk (Thales ref: BU-RS/PJV/13-0002)

Software:

MATLAB Version: 8.1.0.604 (R2013a)
License Number: 724***
Operating System: Microsoft Windows XP Version 5.1 (Build 2600: Service Pack 3)

MATLAB data terminology:

ArrayData:	four-dimensional array containing the signal values for dimensions Doppler, range, azimuth and scan respectively
AzimuthValues:	matrix containing the azimuth values for the respective azimuth indices and scan
ScanCheckVector:	vector containing the scan numbers of the data

Aerial imagery:

Google Earth (imagery date: July 8th 2013)

Oceanography and radar terminology:

Bathymetry:	the study of underwater depths of lake or ocean floors.
Clutter:	unwanted echoes in the radar system, which can include echoes from sea waves, ground, rain and animals, depending on the intended use of the radar.
Open sea:	the area of sea beyond the surf close to the shore.
Fetch:	area of ocean surface over which wind blows in a constant direction, causing swell.
Sea:	a body of salt water (distinct from lakes containing fresh water). The term includes the ocean (as stated in the "United Nations Convention on the Law of the Sea").
Shoaling:	the increasing wave height of surface waves when entering shallower waters.
Surf:	the breaker waves that occur near the seashore due to shoaling.
Swell:	waves that propagate along the interface between water and air, driven by distant weather systems, rather than local wind.
Target:	a signal-reflecting object, as distinct from clutter.

Appendix B

This appendix chapter contains the MATLAB codes used.

B.1 *radarplot*

```
function radarplot( datadB,range,azimuth,shad,plottitle,fig )
%radarplot( datadB,range,azimut,shad,plottitle,fig )
%This function makes a radar plot of dB data
%datadB     the data in dB form, range x azimuth   (n x m)  [dB]
%range      vector with range grid                 (1 x n)  [any unit]
%azimuth    standing vector of the azimuth grid    (m x 1)  [rad]
%shad       shading 'flat' or 'interp'
%plottitle  title above the plot                   [text]
%fig        [figurenumber, subplotnumber]
range=range';
azimuth=-azimuth+pi/2;
rangegrid=range*ones(size(azimuth))';
azigrid=ones(size(range))*azimuth';
figure(fig(1));
if length(fig)>1;    subplot(fig(2)); end
pcolor(rangegrid.*cos(azigrid),rangegrid.*sin(azigrid),datadB);
shading(shad);
colorbar;title(colorbar,'[dB]')
caxis([0,60]);
%caxis([0,80]);
axis( 'equal')
%suitable Scout Logged data (Range 1:100)
%   set(gca,'XTick',linspace(-92.1,0,6)),
%   set(gca,'XTickLabel','-500|-400|-300|-200|-100|0'),
%   set(gca,'YTick',linspace(0,73.7,5)),
%   set(gca,'YTickLabel','0|100|200|300|400');
title(plottitle);
xlabel(['Range [m]']);
ylabel(['Range [m]']);
end
```

B.2 *leesdata8*

```
function [ArrayData, AzimuthValues, ScanCheckVector] = leesdata8(filename,...
    FigureNum,Rangecells,NumberofScans,Azimuthbounds,Asize)
%[ArrayData, AzimuthValues, ScanCheckVector] = leesdata8(filename,FigureNum,...
    Rangecells,NumberofScans,Azimuthbounds,Asize)
%This function reads and plots the data (which does NOT contain scan
%numbers) in a Doppler-range-azimuth-scannumber plot and also gives a
```

```
%matrix containing the values of the azimuth indexes. Further it allows us
%to state the azimuthbounds we are interested in. The scannumbers are
%subsequently assigned in ScanCheckVector starting from 1.
%Input Variables
%filename       this is our input filename - including the path
%FigureNum      figure number for plots
%               if FigureNum==0 no plots are made
%Rangecells     the vector with the rangecell indexes that will be read
%NumberofScans  the scalar of the number of scan-number indexes
%Azimuthbounds  the 2 element vector containing the azimuthbounds [rad]
%Asize          the scalar giving the maximum number of azimuth indexes per
%               scan. Enter an overestimate, but limit size for memory
%Output variables
%ArrayData          this is a 4 dimensional array containing the complex
%                   amplitude (Doppler, Range, Azimuth, Scannumber) (data
%                   that is not present in the data is given as NaN)
%AzimuthValues      A matrix containing the azmiuth values and scannumbers
%ScanCheckVector    A vector giving the scannumbers that were read
%The following is the contents of the data variable:
%in_azimuth: 3.0862 [rad], the angle from North in clockwise direction
%out_doppler_video: [24x2048 struct], 24 Doppler rows & 2048 Range columns,
%out_noise_level: 20.9404, TBC??
%inout_noise_level_correction_avg_dB: [1x2048 single], TBC??
%COUNT: 6942, TBC??
%LOG64TIME: 1372338258082332000, TBC??
%SUBJECT_INFO: [1x1 struct], TBC??

%% open file and assess content
closelogall
fid=openlog(filename);
[msg, data] = readlog(fid);
[Dsize,MaxRange]=size(data.out_doppler_video);
Rsize=length(Rangecells);
disp(['The number of Dopplerbins is ',num2str(Dsize)])
disp(['The maximum number of range cells is ',num2str(MaxRange)])
ArrayData=NaN(Dsize,Rsize,Asize,NumberofScans);
a_index=1;
s_index=1;
AzimuthValues=NaN(Asize,NumberofScans);
while ~isempty(msg);
    [msg2 data2] = readlog(fid);
        if s_index>NumberofScans
            disp('Desired number of scans reached (more still in data however)...
                ')
            ScanCheckVector=1:NumberofScans;
        return;
        end
        if isempty(msg2), disp('END OF FILE !!!'),
            if min(isnan(AzimuthValues(:,s_index)))==1;%if the last scan ...
                contains no azimuth values
                s_index=s_index-1;
            end
            ArrayData=ArrayData(:,:,:,1:s_index);
            AzimuthValues=AzimuthValues(:,1:s_index);%should we enter more ...
                scannumbers than our data contains
            ScanCheckVector=1:s_index;
        return;
```

```
        end
        Vid=[data2.out_doppler_video];
        if Vid(2,3).i==0 && Vid(2,3).q==0;%check if there is data
            disp('move to the next one');
        else
            if data.in_azimuth>Azimuthbounds(1) && data.in_azimuth<...
                Azimuthbounds(2)
            AzimuthValues(a_index,s_index)= data.in_azimuth;
            disp(['Scannumber ', num2str(s_index),', Azimuth index ', num2str(...
                a_index), ', Azimuth value ', num2str(data.in_azimuth)])
            for r=1:Dsize;%these are the Dopplerbins
                for k=Rangecells;%these are the rangecells
                    ArrayData(r,k-Rangecells(1)+1,a_index,s_index)=Vid(r,k).i...
                        +1i*Vid(r,k).q;
                end;
            end
            if FigureNum>0;
                figure(FigureNum)
                Video2=squeeze(ArrayData(:,:,a_index,s_index));
                LogModVideo=10*log10(abs(Video2).^2);
                pcolor(1:Rsize,1:Dsize,double(LogModVideo));colorbar;shading('...
                    flat');caxis([0,80])
                axis([0,Rsize,0,Dsize]);
                title(['Scan nr = ',num2str(s_index),' azimuth = ',num2str(...
                    data.in_azimuth/pi*180),' [degree]']);
                xlabel(['Range cells going from ', Rangecells(1) ]);ylabel('...
                    Dopplerbin')
                drawnow;
            end
            a_index=a_index + 1;
            else
                disp('Azimuth value currently not within our specified range ...
                    of interest')
            end
        end %end of data check
        if (data.in_azimuth < 0 && data2.in_azimuth >= 0) == 1
            s_index=s_index+1;
            a_index=1;
        end
    msg = msg2;
    data = data2;
end
return
```

B.3 *TestwavesTailor*

```
function [SimulatedWaves, AzimuthValues] = TestWavesTailor(NumberofRangeCells,...
     NumberofScans,omega,kx,ky,RandomnessFactor)
%[ArificialTestWaves, AzimuthValues] = TestWavesTailor(NumberofRangeCells, ...
    NumberofScans,omega,kx,ky,RandomnessFactor)
%This function creates Artificial Test Waves on a plot and also the
%corresponding matrices for the 4 dimensional array of signalvalues
%(for dimensions Doppler, Range, Azimuth, Scan). The Doppler dimension is
%given a size 2, with the second "bin" simply being a copy of the first.
%The Azimuths are from 0:pi and the signal strength is 80dB.
%
```

```
%Input variables
%NumberofRangeCells    The scalar of the number of range cells we want
%NumberofScans         The scalar of the number of scans we want
%omega                 The angular frequency of the wave [rad/s]
%kx                    The wave number in the x direction [rad/m]
%ky                    The wave number in the y direction [rad/m]
%RandomnessFactor      The scalar of the randomness factor. Should be
%                      between 0 and 1. (about 0.2 is recommended)
%Output variables
%SimulatedWaves        The 4 dimensional array containing the signal values
%                      in the order D,R,A,S
%AzimuthValues         The matrix containing the azimuth angles for each of
%                      the scans and scan indexes (Azimuths-by-scans)
r=1:1:NumberofRangeCells;
b=0:0.01:pi;
t=linspace(0,NumberofScans,NumberofScans);
[R B T]= meshgrid(r,b,t);
[Y,X]=pol2cart(B,R);
A=cos(kx*X+ky*Y-omega*T)+RandomnessFactor*randn(size(X));
A =permute(A,[2 1 3]);%this ensures that our output is in dimension order R,A,...
    S
A=(A+1)*40;

for i=1:size(A,3)
radarplot(r',b',A(:,:,i),'flat',['Simulated waves with angular frequency ' ...
    num2str(omega) ' rad/s,' 10 'wave number ' num2str(kx) ' rad/m in x ...
    direction and ' num2str(ky) ' rad/m in y direction' 10 'and randomness ...
    factor ' num2str(RandomnessFactor)],42),caxis([0,80])
pause(0.1)
end
SimulatedWaves=zeros(2,NumberofRangeCells,315,NumberofScans);
for d=1:2
    SimulatedWaves(d,:,:,:)=A;
end
%this gives us a 4-dimensional array including Doppler
Asize=size(A);
NumberofAzimuthIndexes=Asize(2);

AzimuthValues=linspace(0,pi,NumberofAzimuthIndexes);
AzimuthValues=AzimuthValues';
for i=1:NumberofScans
AzimuthValues(:,i)=AzimuthValues(:,1);
end
```

B.4 *makecomplex*

```
function[complexdata]=makecomplex(dBdata, phase)
%this program takes real data (in decibels) and converts it to complex data
%with a random phase
%input variables
%dBdata             4 dimensional array containing real signal values (in
%                   order Doppler, Range, Azimuth, Scan)
%phase              determines if we want a random phase for each cell or a
%                   specific random phase throughout. 1 means random phase
%                   for each cell and 0 means one specific (random) phase
%output variable
```

```
%complexdata        4 dimensional array containing complex signal values
%                   (in same order as the above)
dBdatasize=size(dBdata);
complexdata=zeros(dBdatasize(1),dBdatasize(2),dBdatasize(3),dBdatasize(4));

if phase==1
    for D=1:dBdatasize(1)
        for R=1:dBdatasize(2)
            for A=1:dBdatasize(3)
                for S=1:dBdatasize(4)
                    complexdata(D,R,A,S)=10^(dBdata(D,R,A,S)/20)*exp(2*pi*rand...
                        (1)*1i);
                end
            end
        end
    end
else
    OneRandomPhase=exp(2*pi*rand(1)*1i);
    for D=1:dBdatasize(1)
        for R=1:dBdatasize(2)
            for A=1:dBdatasize(3)
                for S=1:dBdatasize(4)
                    complexdata(D,R,A,S)=10^(dBdata(D,R,A,S)/20)*...
                        OneRandomPhase;
                end
            end
        end
    end
end
```

B.5 *maketarget*

```
function [data] = maketarget(dat0,x0,v0,A,B,Sw )
%[data] = maketarget(dat0,x0,v0,A,B,Sw );
%Makes multidimensional data (output) for a moving target
%target position and speed in grid units
%dat0       Empty matrix with D dimensions (Doppler,Range,Azimut,Scan)
%               (for 1-D data colomnvector)
%x0         Target's starting position in all dimensions
%           (ensure yourself that the target Doppler is correct)
%V0         Target's speed in all dimensions
%           For Doppler enter 0 and for time enter 1
%A          Amplitude   (complex)
%B          Vector with target width in all dimensions (at 3 dB)
%Sw         Swerling case target Sw=0    Constant amplitude
%                                Sw=1    exponentially distributed power
D=size(dat0);
b=B/(2*sqrt(2*log(2)));         %widthparameter of amplitude

if length(D)==2 & D(2)==1;      %1-D data
    D=D(1);
end
DIM=length(D);
data=dat0;
switch DIM
    case 1;     %1 dim, no time dimension
```

```
        x=(1:D)';
        data=A*exp(-((x-x0)/b).^2);
    case 2      %1 spatial and 1 time dimension
        if Sw==0;AA=A*ones(D(2),1);else AA=A*sqrt(-log(1-rand(D(2))));end
        x=(1:D(1))';
        for i=1:D(2);
            data(:,i)=AA(i)*exp(-((x-(x0(1)+v0(1)*i) )/b(1)).^2);
        end
    case 3      %2 spatial and 1 time dimension
        if Sw==0;AA=A*ones(D(3),1);else AA=A*sqrt(-log(1-rand(D(3))));end
        x1=(1:D(1))'*ones(1,D(2));  %first spatial coordinate
        x2=ones(D(1),1)*(1:D(2));   %second spatial coordinaat
        for i=1:D(3);
            data(:,:,i)=AA(i)*exp( - ((x1-(x0(1)+v0(1)*i))/b(1)).^2  -...
                                ((x2-(x0(2)+v0(2)*i))/b(2)).^2);
        end
end
end
```

B.6 *CorrelationPlot_2dimTshifted*

```
function[Correlation_Matrix, AmountOfCorrData] = CorrelationPlot_2dimTshifted(...
    ArrayData, AzimuthValues, ScanCheckVector, Dopplerbin, Rangecells, ...
    Azimuthbounds, Scannumbers, DeltaVec)
%[Correlation_Matrix, AmountOfCorrData] = CorrelationPlot_2dimTshifted(...
    ArrayData, AzimuthValues, ScanCheckVector, Dopplerbin, Rangecells, ...
    Azimuthbounds, Scannumbers, DeltaVec)
%This program reads the correlation coefficients between the 2 dimensions
%range, azimuth only, but also of the previous and imminent scan (the first
%dimension Doppler is ommited).
%Input variables
%ArrayData          This is our 4D-array which contains the complex signal
%                   values in dimensions order (D, R, A, S). Comes from
%                   leesdata6, 7 or 8
%AzimuthValues      The matrix containing the raw data from azmiuth values
%                   and scannumbers. Also comes from leesdata6, 7 or 8
%ScanCheckVector    A vector giving the scannumbers that were read, also
%                   coming from leesdata6, 7 or 8
%Dopplerbin         This is one scalar Dopplerbin we wish to set. Must be a
%                   positive integer within the Dopplers of our ArrayData ...
    array.
%Rangecells         The unbroken vector with the rangecell indexes we take
%                   to later calculate our correlation coefficients along
%Azimuthbounds      The 2-element vector containing the lower and upper bound
%                   of the azimuth values along which we calculate the
%                   correlation coefficients. The azimuthwidth is added onto
%                   each azimuth index of the first value.
%Scannumbers        The unbroken vector of the correct scannumbers - in ...
    ascending order
%DeltaVec           This is a 2-element vector containing two scalars (the
%                   dimension-number 1,2,3,4 for D,R,A,S respectively) that
%                   we want in our graph. The first scalar for the x-axis,
%                   the second for the y-axis.
%For the program leesdata8, we use the input variable fid (file-identifier)
%which comes from program openlog.
%Output variable
```

```
%Correlation_Matrix    this is a matrix giving the correlation coefficient
%                      values for the 2 dimensions along the respective
%                      number of displacements for the dimensions. The size
%                      of the matrix depends on the dimensions chosen.
%AmountOfCorrData      a scalar of the quantity of data from which we are
%                      calculating the correlation coefficients.
DimensionInitials='DRAS';
Azimuthwidth=10;
AmountOfCorrData=length(Rangecells)*(Azimuthwidth)*length(Scannumbers);
Dopplerbin_Original=Dopplerbin;
DopplerbinLongVector=[1:24 1:24 1:24];
Azimuthindexes=find(Azimuthbounds(1)<=AzimuthValues(:,1) & Azimuthbounds(end)...
    >=AzimuthValues(:,1));
TotalDimVec=[3,2;3,2;3,2];
DeltaVec4D=zeros(4,2);
DeltaVec4D(2,1:2)=[-DeltaVec(1),DeltaVec(1)];
DeltaVec4D(3,1:2)=[-DeltaVec(2),DeltaVec(2)];
DeltaVec4D(4,1:2)=[-DeltaVec(3),DeltaVec(3)];

for plot=1:3
    DimVec=TotalDimVec(plot,:);
    Correlation_Matrix=zeros(DeltaVec4D(DimVec(2),2)-DeltaVec4D(DimVec(2),1)...
        +2,DeltaVec4D(DimVec(1),2)-DeltaVec4D(DimVec(1),1)+2);
    DispVec=[0,0,0,0];

for m=DeltaVec4D(DimVec(2),1):DeltaVec4D(DimVec(2),2)
    DispVec(DimVec(2))=m;
    for n=DeltaVec4D(DimVec(1),1):DeltaVec4D(DimVec(1),2)
        DispVec(DimVec(1))=n;
        for j=Azimuthindexes(1):Azimuthindexes(end)

            Azimuths_Original=      j:j+Azimuthwidth-1;
            Scanindexes_Original=   Scannumbers-ScanCheckVector(1)+1;
            Dopplerbin_New=         DopplerbinLongVector(Dopplerbin+24+DispVec...
                (1));
            Rangecells_New=         Rangecells(1)+DispVec(2):Rangecells(end)+...
                DispVec(2);
            Azimuths_New=           j+DispVec(3):j+Azimuthwidth-1+DispVec(3);
            Scanindexes_New=        Scannumbers-ScanCheckVector(1)+-1+plot;

            Original_Signal_Values= 10*log10( abs( squeeze(ArrayData(...
                Dopplerbin_Original,Rangecells,Azimuths_Original,...
                Scanindexes_Original))).^2 );
            New_Signal_Values=      10*log10( abs( squeeze(ArrayData(...
                Dopplerbin_New,Rangecells_New,Azimuths_New,Scanindexes_New))).....
                ^2 );
            %the above give us our original vector and a displaced vector (...
                displaced
            %according to our input)
            Vector_Original_Signal_Values=Original_Signal_Values(:)';
            Vector_New_Signal_Values=New_Signal_Values(:)';
            CCValues(j-Azimuthindexes(1)+1)= mean( (...
                Vector_Original_Signal_Values - mean(...
                Vector_Original_Signal_Values)).*(Vector_New_Signal_Values - ...
                mean(Vector_New_Signal_Values))) / (std(...
                Vector_Original_Signal_Values,1)*std(Vector_New_Signal_Values...
                ,1));
```

```
        end
        Correlation_Matrix(m+1-DeltaVec4D(DimVec(2),1),n+1-DeltaVec4D(DimVec...
            (1),1))=mean(CCValues);
    end
end
subplot(130+plot)
pcolor(DeltaVec4D(DimVec(1),1)-0.5:DeltaVec4D(DimVec(1),2)+1-0.5,DeltaVec4D(...
    DimVec(2),1)-0.5:DeltaVec4D(DimVec(2),2)+1-0.5,double(Correlation_Matrix));...
    shading('flat');caxis([-1,1]);%colorbar
end
subplot(131)
axis([DeltaVec4D(DimVec(1),1)-0.5,DeltaVec4D(DimVec(1),2)+0.5,DeltaVec4D(...
    DimVec(2),1)-0.5,DeltaVec4D(DimVec(2),2)+0.5]);colorbar
                xlabel(['\Delta in ' num2str(DimensionInitials(DimVec(1))) ' ...
                    dimension']);ylabel(['\Delta in ' num2str(DimensionInitials...
                    (DimVec(2))) ' dimension'])
subplot(132)
axis([DeltaVec4D(DimVec(1),1)-0.5,DeltaVec4D(DimVec(1),2)+0.5,DeltaVec4D(...
    DimVec(2),1)-0.5,DeltaVec4D(DimVec(2),2)+0.5]);colorbar
                xlabel(['\Delta in ' num2str(DimensionInitials(DimVec(1))) ' ...
                    dimension']);
subplot(133)
axis([DeltaVec4D(DimVec(1),1)-0.5,DeltaVec4D(DimVec(1),2)+0.5,DeltaVec4D(...
    DimVec(2),1)-0.5,DeltaVec4D(DimVec(2),2)+0.5]);colorbar
                xlabel(['\Delta in ' num2str(DimensionInitials(DimVec(1))) ' ...
                    dimension']);
subplot(132),title(['Correlation coefficients for Simdata1' 10 'Dopplerbin ' ...
    num2str(Dopplerbin) ', range indices ' num2str(Rangecells(1)) ':' num2str(...
    Rangecells(end)),', azimuth values ' num2str(Azimuthbounds(1)) ':' num2str(...
    Azimuthbounds(end)) ' rad' 10 'at azimuth width ' num2str(Azimuthwidth) ' ...
    indices (6^{\circ}), at previous, current and imminent scan']);
return
end
```

B.7 *SD_coefficients_multiple*

```
function[anM, YnM, RSM, ASM] = SD_coefficients_multiple(ArrayData, doppler, ...
    range, azimuth, scan, maxd, Surf, MaxR, MaxA)
%[anM, YnM, RSM, ASM] = SD_coefficients_multiple(ArrayData, doppler, range, ...
    azimuth, scan, maxd, Surf, MaxR, MaxA)
%This program reads the ArrayData matrix of waves only and splits it into
%various range azimuth sectors. Then via the follow up program
%"SD_coefficients_single", it calculates the predictor cells and
%coefficients for each sector, with enough decorrelation steps to bring
%correlations below magnitude 0.1.
%Input variables:
%ArrayData          This is our 4 dimensional array containing the complex
%                   signal values of waves only (in order D,R,A,S)
%doppler            This is the scalar Doppler we wish to examine
%range              The unbroken vector containing the all range cells we
%                   wish to obtain prediction coefficients for
%azimuth            The unbroken vector of all azimuth indexes of interest
%scan               The unbroken vector containing the scan numbers we are
%                   interested in in our dataset
%maxd               The maximum size of the covariance matrix, in range and
%                   azimuth [2-element vector]
```

```
%Surf               The 2-element vector containing the start and end of
%                   surf zone ("branding")
%MaxR               The maximum index width that the range sector for
%                   which we calculate the prediction coefficients
%MaxA               The maximum index width that the azimuth sector for
%                   which we calculate the prediction coefficients
%Output variables:
%anM                A cell array containing the prediction coefficients
%                   for the respective range and azimuth sectors [R-by-A]
%YnM                A cell array containing the predictor cells for the
%                   respective range and azimuth sectors [R-by-A]. The
%                   cell is from top left, where the 3 scans are placed in
%                   a row (scan -1, 0, 1)
%RSM                A matrix containing the start and end cell for dry land,
%                   the surf and open sea respectively [R-by-2]
%ASM                A matrix containing the start and end cell of the
%                   azimuth sectors [A-by-2]
%(where R and A signify the calculated number of range and azimuth sectors)
ArrayData=20*log10(abs(ArrayData));

allR1=ismember(range,Surf);
R1first=find(allR1,1,'first');
R1end=find(allR1,1,'last');
R1sectornum=floor((length(R1first:R1end)-2*maxd(1))/(MaxR+1));
RSM1=NaN(R1sectornum,2);
R1length=MaxR+1;
for i=1:R1sectornum
    RSM1(i,1)=maxd(1)+(MaxR/2)+range(R1first)+(R1length*(i-1));
    RSM1(i,2)=maxd(1)+(MaxR/2)+range(R1first)-1+(R1length*(i));
end
shift=Surf(end)-RSM1(R1sectornum,2);
RSM1=RSM1+shift;
allR2=find(range>Surf(end)+1);
R2sectornum=floor((length(allR2)-2*maxd(1))/(MaxR+1));
RSM2=NaN(R2sectornum,2);
R2length=MaxR+1;
for i=1:R2sectornum
    RSM2(i,1)=range(R1end+1)+(R2length*(i-1));
    RSM2(i,2)=range(R1end+1)-1+(R2length*(i));
end
Rsectornum=R1sectornum+R2sectornum;
RSM=NaN(Rsectornum,2);
RSM(1:R1sectornum,:)=RSM1;
RSM(R1sectornum+1:end,:)=RSM2;
Asectornum=floor((length(azimuth)-2*maxd(2)-MaxA)/(MaxA+1));
ASM=NaN(Asectornum,2);
Alength=MaxA+1;
for i=1:Asectornum
    ASM(i,1)=maxd(2)+MaxA/2+azimuth(1)+(Alength*(i-1));
    ASM(i,2)=maxd(2)+MaxA/2+azimuth(1)-1+(Alength*(i));
end
anM=cell(Rsectornum,Asectornum);
YnM=cell(Rsectornum,Asectornum);
for D=doppler
    for R=1:Rsectornum
        for A=1:Asectornum
            CovR=RSM(R,:);
```

```
            CovA=ASM(A,:);
            ranges=(CovR(1)-maxd(1)):(CovR(2)+maxd(1));
            azimuths=(CovA(1)-maxd(2)):(CovA(2)+maxd(2));
            Y=squeeze(ArrayData(D,ranges,azimuths,scan));
            [Rbig,centre,Rsmalldims] =Grandcovariance(Y,maxd);
            [Yn, a]=SD_coefficients_single(Rbig, [Rsmalldims(1),Rsmalldims(2)...
                ]);
            anM{R,A}=a;
            YnM{R,A}=Yn;
        end
    end
end
```

B.8 *CorrelationPlot_3dim*

```
function[Correlation_Matrix, AmountOfCorrData] = CorrelationPlot_3dim(...
    ArrayData, AzimuthValues, ScanCheckVector, Dopplerbin, Rangecells, ...
    Azimuthbounds, Scannumbers, DeltaVec)
%[Correlation_Matrix, AmountOfCorrData] = CorrelationPlot_3dim(ArrayData, ...
    AzimuthValues, ScanCheckVector, Dopplerbin, Rangecells, Azimuthbounds, ...
    Scannumbers, DeltaVec)
%This program reads the correlation coefficients between the 3 dimensions
%range, azimuth and scan only (the first dimension Doppler is ommited). It
%is the sequel to "CorrCoeffPlot_2dim", but with the dimensions predetermined
%and 3 plots are produced in total. It is most applicable to simulated
%waves, and includes a title as such.
%Input variables
%ArrayData          This is our 4D-array which contains the complex signal
%                   values in dimensions order (D, R, A, S). Comes from
%                   leesdata6, 7 or 8
%AzimuthValues      The matrix containing the raw data from azmiuth values
%                   and scannumbers. Also comes from leesdata6, 7 or 8
%ScanCheckVector    A vector giving the scannumbers that were read, also
%                   coming from leesdata6, 7 or 8
%Dopplerbin         This is one scalar Dopplerbin we wish to set. Must be a
%                   positive integer within the Dopplers of our ArrayData ...
    array.
%Rangecells         The unbroken vector with the rangecell indexes we take
%                   to later calculate our correlation coefficients along
%Azimuthbounds      The 2-element vector containing the lower and upper bound
%                   of the azimuth values along which we calculate the
%                   correlation coefficients. The azimuthwidth is added onto
%                   each azimuth index of the first value.
%Scannumbers        The unbroken vector of the correct scannumbers - in ...
    ascending order
%DeltaVec           This is a 2-element vector containing two scalars (the
%                   dimension-number 1,2,3,4 for D,R,A,S respectively) that
%                   we want in our graph. The first scalar for the x-axis,
%                   the second for the y-axis.
%For the program leesdata8, we use the input variable fid (file-identifier)
%which comes from program openlog.
%Output variable
%Correlation_Matrix    this is a matrix giving the correlation coefficient
%                      values for the 2 dimensions along the respective
%                      number of displacements for the dimensions. The size
%                      of the matrix depends on the dimensions chosen.
```

```
%AmountOfCorrData      a scalar of the quantity of data from which we are
%                      calculating the correlation coefficients.
DimensionInitials='DRAS';
Azimuthwidth=10;
AmountOfCorrData=length(Rangecells)*(Azimuthwidth)*length(Scannumbers);
Dopplerbin_Original=Dopplerbin;
DopplerbinLongVector=[1:24 1:24 1:24];
Azimuthindexes=find(Azimuthbounds(1)<=AzimuthValues(:,1) & Azimuthbounds(end)...
    >=AzimuthValues(:,1));

TotalDimVec=[3,2;2,4;3,4];
DeltaVec4D=zeros(4,2);
DeltaVec4D(2,1:2)=[-DeltaVec(1),DeltaVec(1)];
DeltaVec4D(3,1:2)=[-DeltaVec(2),DeltaVec(2)];
DeltaVec4D(4,1:2)=[-DeltaVec(3),DeltaVec(3)];

for plot=1:3
    DimVec=TotalDimVec(plot,:);
    Correlation_Matrix=zeros(DeltaVec4D(DimVec(2),2)-DeltaVec4D(DimVec(2),1)...
        +2,DeltaVec4D(DimVec(1),2)-DeltaVec4D(DimVec(1),1)+2);
    DispVec=[0,0,0,0];
for m=DeltaVec4D(DimVec(2),1):DeltaVec4D(DimVec(2),2)
    DispVec(DimVec(2))=m;
    for n=DeltaVec4D(DimVec(1),1):DeltaVec4D(DimVec(1),2)
        DispVec(DimVec(1))=n;
        for j=Azimuthindexes(1):Azimuthindexes(end)
            Azimuths_Original=      j:j+Azimuthwidth-1;
            Scanindexes_Original=   Scannumbers-ScanCheckVector(1)+1;
            Dopplerbin_New=         DopplerbinLongVector(Dopplerbin+24+DispVec...
                (1));
            Rangecells_New=         Rangecells(1)+DispVec(2):Rangecells(end)+...
                DispVec(2);
            Azimuths_New=           j+DispVec(3):j+Azimuthwidth-1+DispVec(3);
            Scanindexes_New=        Scannumbers-ScanCheckVector(1)+1+DispVec...
                (4);

            Original_Signal_Values= 10*log10( abs( squeeze(ArrayData(...
                Dopplerbin_Original,Rangecells,Azimuths_Original,...
                Scanindexes_Original))).^2 );
            New_Signal_Values=      10*log10( abs( squeeze(ArrayData(...
                Dopplerbin_New,Rangecells_New,Azimuths_New,Scanindexes_New))).....
                ^2 );
            %the above give us our original vector and a displaced vector (...
                displaced
            %according to our input)
            Vector_Original_Signal_Values=Original_Signal_Values(:)';
            Vector_New_Signal_Values=New_Signal_Values(:)';
            CCValues(j-Azimuthindexes(1)+1)= mean( (...
                Vector_Original_Signal_Values - mean(...
                Vector_Original_Signal_Values)).*(Vector_New_Signal_Values - ...
                mean(Vector_New_Signal_Values))) / (std(...
                Vector_Original_Signal_Values,1)*std(Vector_New_Signal_Values...
                ,1));
        end
        Correlation_Matrix(m+1-DeltaVec4D(DimVec(2),1),n+1-DeltaVec4D(DimVec...
            (1),1))=mean(CCValues);
    end
```

```
end
subplot(130+plot)
pcolor(DeltaVec4D(DimVec(1),1)-0.5:DeltaVec4D(DimVec(1),2)+1-0.5,DeltaVec4D(...
    DimVec(2),1)-0.5:DeltaVec4D(DimVec(2),2)+1-0.5,double(Correlation_Matrix));...
    colorbar;shading('flat');caxis([-1,1])
                axis([DeltaVec4D(DimVec(1),1)-0.5,DeltaVec4D(DimVec(1),2)+0.5,...
                    DeltaVec4D(DimVec(2),1)-0.5,DeltaVec4D(DimVec(2),2)+0.5]);
                xlabel(['\Delta in ' num2str(DimensionInitials(DimVec(1))) ' ...
                    dimension']);ylabel(['\Delta in ' num2str(DimensionInitials...
                    (DimVec(2))) ' dimension'])
end
subplot(132),title(['Correlation coefficients for Simdata1' 10 'Dopplerbin ' ...
    num2str(Dopplerbin) ', range indices ' num2str(Rangecells(1)) ':' num2str(...
    Rangecells(end)),', azimuth values ' num2str(Azimuthbounds(1)) ':' num2str(...
    Azimuthbounds(end)) ' rad' 10 'at azimuth width ' num2str(Azimuthwidth) ' ...
    indices (6^{\circ}), and scan numbers ' num2str(Scannumbers(1)) ':' num2str...
    (Scannumbers(end))]);
return
end
```

B.9 *radarplotVideo*

```
function [Moviegetframe]=radarplotVideo(ArrayData,AzimuthValues,dopplerbin,...
    range,azimuthindexes,scannumbers,shad,plottitle,fig)
%[Moviegetframe]=radarplotVideo(ArrayData,AzimuthValues,dopplerbin,range,...
    azimuthindexes,scannumbers,shad,plottitle,fig)
%This function makes a video radar plot of the complex data
%ArrayData       array containing the complex signal values
%                (dopplerbins x n x m x scannumbers)
%AzimuthValues   matrix containing the azimuth values for the respective
%                scan numbers (m x scannumbers) [rad]
%dopplerbin      the scalar Dopplerbin of interest
%range           vector with range grid             (1 x n)   [no unit]
%azimuthindexes  vector of the azimuth indexes of interest [no unit]
%scannumbers     the vector containing the scannumbers we wish to examine
%shad            shading 'flat' or 'interp'
%plottitle       title above the plot               [text]
%fig             [figurenumber, subplotnumber]
azimuth=AzimuthValues(azimuthindexes);
Rangeindexes=1:size(range');
for i=scannumbers
    radarplot( squeeze(ArrayData(dopplerbin,Rangeindexes,azimuthindexes,...
        scannumbers(i))),range,azimuth,shad,plottitle,fig );
    Moviegetframe(i)=getframe(gcf);
    pause(0.9)
end
```

B.10 *interpolate*

```
function [ArrayData_interpolated, Azimuthgrid] = interpolate(ArrayData, ...
    AzimuthValues, ScanCheckVector)
%[ArrayData_interpolated, Azimuthgrid] = interpolate(ArrayData, AzimuthValues,...
     ScanCheckVector)
%This program interpolates the data which is necessary when we are
```

```
%observing many scans with various PRFs (Pulse Repetition Frequencies)
%Input variables
%ArrayData          This is the 4 dimensional array containing the complex
%                   signal values. Comes from leesdata6, 7 or 8. This is
%                   the data containing ocean waves only, which is later
%                   used for determining the prediction coefficients
%AzimuthValues      The matrix containing the azimuth angles in radians
%                   for the respective azimuth indexes and scannumbers.
%                   Also comes from leesdata6, 7 or 8.
%ScanCheckVector    The vector containing the scannumbers for the
%                   corresponding scanindexes
%Output variables
%ArrayData_interpolated     The 4 dimensional array containing the
%                           interpolated data
%Azimuthgrid                The row vector containing the interpolated
%                           azimuth angles for the respective indexes
ADSize=size(ArrayData);
DopplerbinsinData=1:ADSize(1);
RangeindexesinData=1:ADSize(2);
ScanindexesLength=ADSize(4);

NumofAzimuthsperScan=sum(~isnan(AzimuthValues));
[EndofAzimuths ScanIndexwithmostAzimuths]=max(NumofAzimuthsperScan);
ArrayData_interpolated=zeros(DopplerbinsinData(end),RangeindexesinData(end),...
    EndofAzimuths,ScanindexesLength);
Azi_interpolated=zeros(EndofAzimuths,ScanindexesLength);
AzimuthValues=unwrap(AzimuthValues);%this changes any absolute jumps in 2pi

for m=1:length(ScanCheckVector)
LastAzimuthValue(m)= AzimuthValues(NumofAzimuthsperScan(m),m);
end

for m=1:ScanindexesLength
        g = AzimuthValues(1:NumofAzimuthsperScan(m),m);
        az = 1:NumofAzimuthsperScan(m);
        g_i = linspace(max(AzimuthValues(1,:)),min(LastAzimuthValue),...
            EndofAzimuths);
        Azi_interpolated(1:EndofAzimuths,m) = interp1(g,az,g_i);
end

for d=DopplerbinsinData
    for r=RangeindexesinData
        for s=1:ScanindexesLength
            for a=1:EndofAzimuths-1
                ArrayData_interpolated(d,r,a,s)= (1-(Azi_interpolated(a,s)-...
                    floor(Azi_interpolated(a,s)))) * ArrayData(d,r,floor(...
                    Azi_interpolated(a,s)),s) + (Azi_interpolated(a,s)-floor(...
                    Azi_interpolated(a,s))) * ArrayData(d,r,ceil(...
                    Azi_interpolated(a,s)),s);
            end
        end
    end
end
Azimuthgrid=g_i';
```

B.11 *LP_coefficients_single*

```
function[coefficients] = LP_coefficients_single(ArrayData, doppler, range, ...
    azimuth, scan, Ysignals)
%[coefficients] = LP_coefficients_single(ArrayData, doppler, range, azimuth, ...
    scan, Ysignals)
%This program reads the ArrayData matrix and gives the coefficients after
%linear prediction, for a specific sector, where predictor cells are
%those that are 2 cells apart exactly from our signal of interest X in the
%current scan and within a 2 cell distance from the previous and imminent
%scan
%Input variables
%ArrayData          This is our 4 dimensional array containing the complex
%                   signal values.
%doppler            This is the Doppler in which we are taking our dataset
%range              The unbroken vector containing the rangecells that we
%                   elect for our dataset
%azimuth            The unbroken vector containing the azimuthcells that we
%                   elect for our dataset
%scan               The unbroken vector containing the scan numbers we are
%                   examining in our dataset
%Ysignals           the vector containing the Ysignals that are at a distance
%                   2 from X, that we wish to examine. In the order of a matrix
%Output variables
%coefficients       Our output containing the coefficients for the
%                   specified Ysignals that are 2 away from X [vector]
ArrayData=10*log10(abs(ArrayData).^2);
ArrayData=ArrayData-mean(mean(mean(mean(ArrayData(doppler,range,azimuth,scan...
    (2:(end-1)))))));

DataSet=ArrayData(doppler,range,azimuth,scan(2:(end-1)));
DataSetVector=DataSet(:); %has length N, m is the number of Y's
N=length(range)*length(azimuth)*(length(scan)-2);
M=length(Ysignals);

Yentire=zeros(M,length(range),length(azimuth),length(scan)-2);
for S=scan(2:(end-1))
    for R=range
        for A=azimuth
            for Di=-2:2
                for Dj=-2:2
                    for Dk=-1:1
                        Ymatrix(Di+3, Dj+3, Dk+2)=ArrayData(doppler,R+Di,A+Dj,...
                            S+Dk);
                    end
                end
            end
            Yentire(:,R-range(1)+1,A-azimuth(1)+1,S-scan(1))=Ymatrix([1:25 ...
                26:30 31 35 36 40 41 45 46:50 51:75]);
        end
    end
end
Yentire = (reshape(Yentire,[M,N]))';%NbyM matrix [Y1 Y2 ...] of Y values

A1=Yentire(:,Ysignals)'*Yentire(:,Ysignals);
A2=(Yentire(:,Ysignals))';
A3=DataSetVector;
coefficients=A1\(A2*A3);
```

B.12 *LP_coefficients_multiple*

```
function[PCM, RSM, ASM] = LP_coefficients_multiple(ArrayData, doppler, range, ...
    azimuth, scan, Surf, MaxR, MaxA, Ysignals)
%[PCM, RSM, ASM] = LP_coefficients_multiple(ArrayData, doppler, range, azimuth...
    , scan, Surf, MaxR, MaxA, Ysignals)
%This program reads the ArrayData array (should be of waves only) and
%gives the predictor coefficients from linear prediction where predictor
%cell options are those that are up to 2 cells from cell-under-test X in the
%previous and future scan and exactly 2 cells from X in the current scan.
%This is done for specific input areal sectors in range and azimuth.
%Input variables:
%ArrayData          This is our 4 dimensional array containing the complex
%                   signal values of waves only (in order D,R,A,S)
%doppler            This is the scalar Doppler we wish to examine
%range              The unbroken vector containing the all range cells we
%                   wish to obtain prediction coefficients for
%azimuth            The unbroken vector of all azimuth indexes of interest
%scan               The unbroken vector containing the scan numbers we are
%                   interested in in our dataset
%Surf               The 2-element vector containing the start and end range
%                   index of the surf zone ("branding")
%MaxR               The maximum index width that the range sector for
%                   which we calculate the prediction coefficients
%MaxA               The maximum index width that the azimuth sector for
%                   which we calculate the prediction coefficients
%Ysignals           the vector containing the predictor cells from the
%                   available choice described above (any numbers between 1
%                   and 66, starting from top left in Scan-1, going down)
%Output variables:
%PCM                A cell array containing the prediction coefficients
%                   for the respective range and azimuth sectors [R-by-A]
%RSM                A matrix containing the start and end cell for dry land,
%                   the surf and open sea respectively [R-by-2]
%ASM                A matrix containing the start and end cell of the
%                   azimuth sectors [A-by-2]
%(where R and A signify the calculated number of range and azimuth sectors)
allR1=ismember(range,Surf);%R1 is the surf zone
R1first=find(allR1,1,'first');
R1end=find(allR1,1,'last');
R1sectornum=ceil(length(R1first:R1end)/MaxR);
RSM1=NaN(R1sectornum,2);
R1length=length(R1first:R1end)/R1sectornum;
for i=1:R1sectornum
    RSM1(i,1)=range(R1first)+floor(R1length*(i-1));
    RSM1(i,2)=range(R1first)-1+floor(R1length*(i));
end
if R1sectornum>0
RSM1(R1sectornum,2)=range(R1end); %in case of rounding errors
end

allR2=find(range>Surf(end)+1);%R2 is the open sea
R2sectornum=ceil(length(allR2)/MaxR);%num of range sectors in the open sea
RSM2=NaN(R2sectornum,2);
R2length=length(allR2)/R2sectornum;
for i=1:R2sectornum
    RSM2(i,1)=range(R1end+1)+floor(R2length*(i-1));
```

```
    RSM2(i,2)=range(R1end+1)-1+floor(R2length*(i));
end
if R2sectornum>0
RSM2(R2sectornum,2)=range(end); %in case of rounding errors
end

Rsectornum=R1sectornum+R2sectornum;
RSM=NaN(Rsectornum,2);
RSM(1:R1sectornum,:)=RSM1;
RSM(R1sectornum+1:end,:)=RSM2;
Asectornum=ceil(length(azimuth)/MaxA);%num of azimuth sectors in all sea
ASM=NaN(Asectornum,2);
Alength=length(azimuth)/Asectornum;

for i=1:Asectornum
    ASM(i,1)=azimuth(1)+floor(Alength*(i-1));
    ASM(i,2)=azimuth(1)-1+floor(Alength*(i));
end
PCM=cell(Rsectornum,Asectornum);
for D=doppler
    for R=1:Rsectornum
        for A=1:Asectornum
            PCM{R,A}=LP_coefficients_single(ArrayData, D, RSM(R,1):RSM(R,2), ...
                ASM(A,1):ASM(A,2), scan, Ysignals);
        end
    end
end
```

B.13 *LP_PredictedArrayData*

```
function[PredictedArrayData] = LP_PredictedArrayData(ArrayData, doppler, range...
    , azimuth, scan, PCM, RSM, ASM, Ysignals)
%[PredictedArrayData] = LP_PredictedArrayData(ArrayData, doppler, range, ...
    azimuth, scan, PCM, RSM, ASM, Ysignals)
%This program calculates the residue signal, the difference between the
%predicted and actual dB values based on the prediction coefficients from
%the program "LP_coefficients_multiple". The output PredictedArrayData can
%subsequently be plotted via radarplotVideo.
%Input variables
%ArrayData                  This is the input 4 dimensional data containing
%                           the complex signal values (in order D, R, A, S).
%                           Should be the waves including the target
%doppler                    This is the scalar representing the actual
%                           Doppler of all cells under test X
%range                      The unbroken vector containing the range cells
%                           of the cells under test. Start from at least the
%                           start of the surf zone.
%azimuth                    The unbroken vector containing the azimuth
%                           indexes of the cells under test
%scan                       The scalar of the scan numbers of the cells
%                           under test
%PCM                        The PredictionCoefficientMatrix. A cell array
%                           containing the pred coefficients for the
%                           respective range and azimuth sectors. Comes from
%                           program LP_coefficients_multiple
%RSM                        The RangeSectorMatrix. A matrix containing the
```

```
%                           start and end cell for dry land, the surf and
%                           open sea respectively [3-by-2 matrix]. Comes
%                           from program LP_coefficients_multiple
%ASM                        The AzimuthSectorMatrix. A matrix containing the
%                           start and end cell of the azimuth sectors. Comes
%                           from program LP_coefficients_multiple
%Ysignals                   The choice of predictor cells [vector]
%Output variables
%PredictedArrayData         This is the ArrayData array, which contains the
%                           absolute difference between predicted and actual
%                           signal values in dB [Size is equal to input
%                           ArrayData minus 2 scans due to predictor shift]
ArrayData=10*log10(abs(ArrayData).^2);
ArrayData=ArrayData-mean(mean(mean(mean(ArrayData(doppler,range,azimuth,scan...
    (2:(end-1)))))));

PredictedArrayData=zeros(size(ArrayData)-[0,0,0,2]);%1st and last scan trimmed...
     due to predictor shift

Asectornum=length(ASM);
Rsectornum=length(RSM);

for D=doppler;
    for R=range;
        for A=azimuth;
            for S=scan(2:(end-1));
                for Di=-2:2
                    for Dj=-2:2
                        for Dk=-1:1
                            Ymatrix(Di+3, Dj+3, Dk+2)=ArrayData(D,R+Di,A+Dj,S+...
                                Dk);
                        end
                    end
                end
                Ysurrounding=Ymatrix([1:25 26:30 31 35 36 40 41 45 46:50 ...
                    51:75]);%66 predictor cells surrounding X

                for Ri=1:Rsectornum
                    if R>=RSM(Ri,1) & R<=RSM(Ri,2);
                    CurrentR=Ri;
                    end
                end
                for Ai=1:Asectornum
                    if A>=ASM(Ai,1) & A<=ASM(Ai,2);
                    CurrentA=Ai;
                    end
                end
                coefficients=cell2mat(PCM(CurrentR,CurrentA));
                predicted=Ysurrounding(Ysignals)*coefficients;
                actual=ArrayData(D,R,A,S);
                absdiff=abs(predicted-actual);
                PredictedArrayData(D,R,A,S-1)=absdiff;
            end
        end
    end
end
```

B.14 *LP_AbsDiff_2dim*

```
function[ResidualMatrix] = LP_AbsDiff_2dim(ArrayData, ActDoppler, ActRange, ...
    ActAzimuths, ActScannumbers,Prediction_coefficients)
%[AbsDiffPlotMatrix] = LP_AbsDiff_2dim(ArrayData, ActDoppler, ActRange, ...
    ActAzimuths, ActScannumbers,Prediction_coefficients)
%This program calculates the difference in predicted and actual values
%calculated and plots their absolute difference in a colour-plot. This is
%for two dimensions, where the 2 dimensions are elected automatically by
%which two that contain multiple elements. The 2 dimensions with one single
%value are fixed at that value
%ArrayData                  This is the input 4 dimensional data containing
%                           the complex signal values
%ActDoppler                 This is the scalar representing the actual
%                           Doppler of our target signal X
%ActRange                   The unbroken vector containing the rangecells
%                           that we wish to examine
%ActAzimuths                The unbroken vector containing the azimuth
%                           indexes that we wish to examine
%ActScannumbers             The scalar of the scannumber that we deal with
%Prediction_coefficients    This is the 16-element standing vector containing ...
    the
%                           coefficients for our linear prediction. This
%                           comes from program "LP_coefficients_..."
%Output variables
%ResidualMatrix             The 4 dimensional data containing the residual
%                           signal values
ArrayData=10*log10(abs(ArrayData).^2);

DLength=length(ActDoppler);
RLength=length(ActRange);
ALength=length(ActAzimuths);
SLength=length(ActScannumbers);

ActualAD=zeros(DLength, RLength, ALength, SLength);
PredictedAD=zeros(DLength, RLength, ALength, SLength);
PredictedADSize=size(PredictedAD);
DimensionofInterest=find(PredictedADSize>1);
AllDimNames=[{'Doppler indexes','Range indexes','Azimuth indexes','Scan ...
    indexes'}];
DimNames=AllDimNames([DimensionofInterest]);

for D=ActDoppler;
    for R=ActRange;
        for A=ActAzimuths;
            for S=ActScannumbers(2:(end-1));
                for Di=-2:2
                    for Dj=-2:2
                        for Dk=-1:1
                        Ymatrix(Di+3, Dj+3, Dk+2)=ArrayData(D,R+Di,A+Dj,S+Dk);
                        end
                    end
                end
                Ysurrounding=Ymatrix([1:25 26:30 31 35 36 40 41 45 46:50 ...
                    51:75]);

                PredictedValue=Ysurrounding*Prediction_coefficients;
```

```
                ActualValue=ArrayData(D,R,A,S);
                AbsDiff=abs(PredictedValue-ActualValue);
                ActualAD(D-ActDoppler(1)+1,R-ActRange(1)+1,A-ActAzimuths(1)+1,...
                    S-ActScannumbers(1)+1)=ActualValue;
                PredictedAD(D-ActDoppler(1)+1,R-ActRange(1)+1,A-ActAzimuths(1)...
                    +1,S-ActScannumbers(1)+1)=AbsDiff;
            end
        end
    end
end

DataPlotMatrix=squeeze(ActualAD);
DataPlotMatrixSize=size(DataPlotMatrix);
DataPlotMatrix(DataPlotMatrixSize(1)+1,:)=0;
DataPlotMatrix(:,DataPlotMatrixSize(2)+1)=0;
ResidualMatrix=squeeze(PredictedAD);
ResidueMatrixSize=size(ResidualMatrix);
ResidualMatrix(ResidueMatrixSize(1)+1,:)=0;
ResidualMatrix(:,ResidueMatrixSize(2)+1)=0;

subplot(121)
pcolor(1:DataPlotMatrixSize(2)+1,1:DataPlotMatrixSize(1)+1,double(...
    DataPlotMatrix));colorbar;title(colorbar,'[dB]');shading('flat');
            axis([1,DataPlotMatrixSize(2)+1,1,DataPlotMatrixSize(1)+1]);
            title('Original Signal');
            xlabel([ DimNames(2) ]); ylabel([ DimNames(1) ])
subplot(122)
pcolor(1:ResidueMatrixSize(2)+1,1:ResidueMatrixSize(1)+1,double(ResidualMatrix...
    ));colorbar;title(colorbar,'[dB]');shading('flat');
            axis([1,ResidueMatrixSize(2)+1,1,ResidueMatrixSize(1)+1]);
            title('Residue Signal');
            xlabel([ DimNames(2) ]); ylabel([ DimNames(1) ])
```

B.15 *SD_coefficients_single*

```
function [Yn,a, Rcorr] = SD_coefficients_single(Rbig,Rsmalldims)
%[Yn,a, Rcorr] = SD_coefficients_single5(Rbig,Rsmalldims,n)
%Applies decorrelation steps until the correlation of all remaining
%predictor cells decreases to magnitude 0.1
%Input variables
%Rbig               The covariance matrix, comes from program
%                   "Grandcovariance"
%Rsmalldims         This is the size of the 'inner' covariance matrix,
%                   comes from the program "Grandcovariance"
%Output variables
%Yn                 The matrix of predictor cell locations [n-by-2], for n
%                   decorrelation steps made
%a                  The standing vector of predictor coefficients, each
%                   row matching to the row of the predictor cell locations
%                   [n-by-1], for n decorrelation steps made
%Rcorr              The correlation coefficient matrix for the input
%                   covariance matrix [same size as Rsmalldims]
Rbigcentre=(size(Rbig)+1)/2;
centresmall=Rsmalldims/2+0.5;
[Rcorr,centreindex] =Grandcov2corrmatrix(Rbig,Rsmalldims);
[D]=findmaxcorrelation(Rcorr,1);
```

```
i=0;

while abs(Rcorr(D(1),D(2)))>0.1
    i=i+1;
    [Rcorr,centreindex] =Grandcov2corrmatrix(Rbig,Rsmalldims);
    [D]=findmaxcorrelation(Rcorr,1);
    Yn(i,:)=D;
    Dindex=sub2ind(Rsmalldims,D(1),D(2));
    a(i)=Rbig(Dindex,Rbigcentre(1))/Rbig(Dindex,Dindex);
    [C,Rbig2] = SD_1step(Rbig,D-centresmall,Rsmalldims);
    Rbig=Rbig2;
     for k=1:length(Rbig)
         if (abs(Rbig(k,k))<0.001)
             Rbig(k,k)=0;
         end
     end
end
a=a';
```

B.16 *SD_coefficients_multiple*

```
function[anM, YnM, RSM, ASM] = SD_coefficients_multiple(ArrayData, doppler, ...
    range, azimuth, scan, maxd, Surf, MaxR, MaxA)
%[anM, YnM, RSM, ASM] = SD_coefficients_multiple(ArrayData, doppler, range, ...
    azimuth, scan, maxd, Surf, MaxR, MaxA)
%This program reads the ArrayData matrix of waves only and splits it into
%various range azimuth sectors. Then via the follow up program
%"SD_coefficients_single", it calculates the predictor cells and
%coefficients for each sector, with enough decorrelation steps to bring
%correlations below magnitude 0.1.
%Input variables:
%ArrayData          This is our 4 dimensional array containing the complex
%                   signal values of waves only (in order D,R,A,S)
%doppler            This is the scalar Doppler we wish to examine
%range              The unbroken vector containing the all range cells we
%                   wish to obtain prediction coefficients for
%azimuth            The unbroken vector of all azimuth indexes of interest
%scan               The unbroken vector containing the scan numbers we are
%                   interested in in our dataset
%maxd               The maximum size of the covariance matrix, in range and
%                   azimuth [2-element vector]
%Surf               The 2-element vector containing the start and end of
%                   surf zone ("branding")
%MaxR               The maximum index width that the range sector for
%                   which we calculate the prediction coefficients
%MaxA               The maximum index width that the azimuth sector for
%                   which we calculate the prediction coefficients
%Output variables:
%anM                A cell array containing the prediction coefficients
%                   for the respective range and azimuth sectors [R-by-A]
%YnM                A cell array containing the predictor cells for the
%                   respective range and azimuth sectors [R-by-A]. The
%                   cell is from top left, where the 3 scans are placed in
%                   a row (scan -1, 0, 1)
%RSM                A matrix containing the start and end cell for dry land,
%                   the surf and open sea respectively [R-by-2]
```

```
%ASM                A matrix containing the start and end cell of the
%                   azimuth sectors [A-by-2]
%(where R and A signify the calculated number of range and azimuth sectors)
ArrayData=20*log10(abs(ArrayData));

allR1=ismember(range,Surf);
R1first=find(allR1,1,'first');
R1end=find(allR1,1,'last');
R1sectornum=floor((length(R1first:R1end)-2*maxd(1))/(MaxR+1));
RSM1=NaN(R1sectornum,2);
R1length=MaxR+1;
for i=1:R1sectornum
    RSM1(i,1)=maxd(1)+(MaxR/2)+range(R1first)+(R1length*(i-1));
    RSM1(i,2)=maxd(1)+(MaxR/2)+range(R1first)-1+(R1length*(i));
end
shift=Surf(end)-RSM1(R1sectornum,2);
RSM1=RSM1+shift;
allR2=find(range>Surf(end)+1);
R2sectornum=floor((length(allR2)-2*maxd(1))/(MaxR+1));
RSM2=NaN(R2sectornum,2);
R2length=MaxR+1;
for i=1:R2sectornum
    RSM2(i,1)=range(R1end+1)+(R2length*(i-1));
    RSM2(i,2)=range(R1end+1)-1+(R2length*(i));
end
Rsectornum=R1sectornum+R2sectornum;
RSM=NaN(Rsectornum,2);
RSM(1:R1sectornum,:)=RSM1;
RSM(R1sectornum+1:end,:)=RSM2;
Asectornum=floor((length(azimuth)-2*maxd(2)-MaxA)/(MaxA+1));
ASM=NaN(Asectornum,2);
Alength=MaxA+1;
for i=1:Asectornum
    ASM(i,1)=maxd(2)+MaxA/2+azimuth(1)+(Alength*(i-1));
    ASM(i,2)=maxd(2)+MaxA/2+azimuth(1)-1+(Alength*(i));
end
anM=cell(Rsectornum,Asectornum);
YnM=cell(Rsectornum,Asectornum);
for D=doppler
    for R=1:Rsectornum
        for A=1:Asectornum
            CovR=RSM(R,:);
            CovA=ASM(A,:);
            ranges=(CovR(1)-maxd(1)):(CovR(2)+maxd(1));
            azimuths=(CovA(1)-maxd(2)):(CovA(2)+maxd(2));
            Y=squeeze(ArrayData(D,ranges,azimuths,scan));
            [Rbig,centre,Rsmalldims] =Grandcovariance(Y,maxd);
            [Yn, a]=SD_coefficients_single(Rbig, [Rsmalldims(1),Rsmalldims(2)...
                ]);
            anM{R,A}=a;
            YnM{R,A}=Yn;
        end
    end
end
```

B.17 ***SD_PredictedArrayData*** (uses B.17a, 17b and 17c)

```
function[PredictedArrayData] = SD_PredictedArrayData(ArrayData, doppler, range...
    , azimuth, scan, anM, YnM, RSM, ASM, maxd)
%[PredictedArrayData] = SD_PredictedArrayData(ArrayData, doppler, range, ...
    azimuth, scan, anM, YnM, RSM, ASM, maxd)
%This program performs sequentially decorrelation on the data containing
%waves with injected targets based on the previously calculated
%coefficients from program "SD_coefficients_multiple". The residue signals,
%difference between predicted and actual dB values are the output which can
%subsequently be plotted in radarplot
%Input variables
%ArrayData                  This is the input 4 dimensional data containing
%                           the complex signal values (in order D,R,A,S)
%doppler                    This is the scalar representing the actual
%                           Doppler of all cells-under-test X
%range                      The unbroken vector containing the range cells
%                           of the cells under test. Start from at least the
%                           start of the surf zone.
%azimuth                    The unbroken vector containing the azimuth
%                           indexes of the cells under test
%scan                       The scalar of the scan numbers of the cells
%                           under test
%anM                        A cell array containing the prediction coefficients
%                           for the respective range and azimuth sectors [R-by-...
    A]
%                           Comes from "SD_coefficients_multiple"
%YnM                        A cell array containing the predictor cells for the
%                           respective range and azimuth sectors [R-by-A]. The
%                           cell is from top left, where the 3 scans are placed...
     in
%                           a row (scan -1, 0, 1). Comes from
%                           "SD_coefficients_multiple"
%RSM                        A matrix containing the start and end cell for
%                           dry land, the surf and open sea respectively
%                           [R-by-2]. Comes from "SD_coefficients_multiple"
%ASM                        A matrix containing the start and end cell of the
%                           azimuth sectors [A-by-2]. Comes from
%                           "SD_coefficients_multiple"
%maxd                       The maximum size of the covariance matrix, in
%                           range and azimuth [2-element vector]
%Output variables
%PredictedArrayData         This is the array, which contains the dB residue
%                           signal, the absolute difference between
%                           predicted and actual signal [Size equal to input
%                           ArrayData minus 2 scans due to time shifted
%                           predictor cells]
%
%H.Krause November 7th, 2013

PredictedArrayData=zeros(size(ArrayData)-[0,0,0,2]);

ArrayData=20*log10(abs(ArrayData));

ArrayData=ArrayData-mean(mean(mean(mean(ArrayData(doppler,:,1:end-1,:)))));

Asectornum=length(ASM);
Rsectornum=length(RSM);
```

```
D=doppler;

for S=scan(2:end-1);
    for Ri=1:Rsectornum
        R=RSM(Ri,1):RSM(Ri,2);
        for Ai=1:Asectornum
            A=ASM(Ai,1):ASM(Ai,2);

            a=cell2mat(anM(Ri,Ai));
            Yn=cell2mat(YnM(Ri,Ai));

            Y=squeeze(ArrayData(D,(R(1)-maxd(1)):(R(end)+maxd(1)),(A(1)-maxd...
                (2)):(A(end)+maxd(2)),(S-1):(S+1)));

            PredY = SD_pred(Y,Yn,a,maxd);
            Yinner=squeeze(ArrayData(D,R(1):R(end),A(1):A(end),S));
            PredMatrix=abs(PredY-Yinner);

            PredictedArrayData(D,R,A,S-1)=PredMatrix;
        end
    end
end
```

B.17a *SD_1step*

```
function [C,Rbig2] = SD_1step(Rbig,Yindex,Rsmalldims)
%[C,Rbig2] = SD_1step(Rbig,Yindex,Rsmalldims)
%applies 1 decorrelation step with respect to the 2D covariance matrix
%Input variables
%Rbig           The covariance matrix
%Yindex         The predictor cells
%Rsmalldims     The size of the covariance matrix
%Output variables
%C              The resulting predictor coefficient
%Rbig2          The new covariance matrix after decorrelation
dim=size(Rbig);             %dimensions of the Covariance matrix
centre=(dim+1)/2;       %position of the centre element (variance of Y)
centresmall=Rsmalldims/2+0.5;

Y1indexsmall=sub2ind(Rsmalldims,Yindex(1)+centresmall(1),Yindex(2)+centresmall...
    (2));
maxd=dim-centre;        %max distance in each direction for which there is ...
    covariance data
maxdsmall=Rsmalldims-centresmall;
C=NaN(Rsmalldims);          %coefficient for decorrelation
Rbig2=NaN(size(Rbig));
power=Rbig(Y1indexsmall,Y1indexsmall);    %signal power

for i=-maxdsmall(1):maxdsmall(1);          %loop over first dimension
    for j=-maxdsmall(2):maxdsmall(2);      %loop over second dimension
        covindexsmall=sub2ind(Rsmalldims,centresmall(1)+i,centresmall(2)+j);
        C(covindexsmall)=Rbig(covindexsmall,Y1indexsmall)/power;
    end
end
%calculate new Covariance matrix
for i=-maxd(1):maxd(1);        %loop over first dimension
```

```
    for j=-maxd(2):maxd(2);    %loop over second dimension

        Rbig2(i+centre(1),j+centre(2))=Rbig(i+centre(1),j+centre(2))-Rbig(i+...
            centre(1),Y1indexsmall)*Rbig(Y1indexsmall,j+centre(2))/power;
    end
end
```

B.17b *SD_pred*

```
function[Ypredsignals,SS] = SD_pred(Ystartsignals,Yn,a,maxd)
%[Ypredsignals] = SD_pred5(Ystartsignals,Yn,a,maxd)
%This program calculates the difference in predicted and actual values
%based on the prediction coefficients from the program
%"LP_coefficients_multiple". The output PredictedArrayData can subsequently
%be plotted via radarplotVideo
%Input variables
%Ystartsignals              This is the matrix containing the original
%                           real dB signal values
%Yn                         The predictor cells
%a                          The predictor coefficients
%maxd                       The size of the covariance matrix
%Output variables
%Ypredsignals               The matrix containing the predicted signal
%                           dB values
%SS                         Scan shift
Firstmean=mean(mean(mean(abs(Ystartsignals(:,:,2)))));
dim=size(Ystartsignals);
Yinner=Ystartsignals(((maxd(1)+1):(end-maxd(1))),((maxd(2)+1):(end-maxd(2)))...
    ,1);
diminner=size(Yinner);
centreinner=(diminner+1)/2;

Ypredsignals=zeros(diminner(1),diminner(2));
Yndim=size(Yn);

for i=1:diminner(1)
    for j=1:diminner(2)
        for k=1:Yndim(1)
            for scan=1:3
                if (1+(maxd(2)*2+1)*(scan-1))<=Yn(k,2) && Yn(k,2)<=((maxd(2)...
                    *2+1)*scan);
                    ScanShift=scan-2;
                end
            end
            SS(k)=ScanShift;
            signal(k)=Ystartsignals(i+Yn(k,1)-1,j+Yn(k,2)-(maxd(2)*2+1)*(...
                ScanShift+1)-1,ScanShift+2);
        end
        Ypredsignals(i,j)=(signal)*a;
    end
end
Secondmean=mean(mean(abs(Ypredsignals)));

Ypredsignals=Ypredsignals*Firstmean/Secondmean;
```

B.17c *findmaxcorrelation*

```
function [D]=findmaxcorrelation(Rcorr,k)
%[D]=findmaxcorrelation(Rcorr,k)
%find the dimensions of the point containing the kth largest correlation
%(positive or negative) but a point at least 2 cells from the centre
%Input variables
%Rcorr      multidimensional matrix with correlation data
%k          the kth largest correlation in the data
%Output variables
%D          the row and column of the maximum covariance
centre=(size(Rcorr)+1)/2;
Rcorr=abs(Rcorr);

for j=1:numel(Rcorr)
maxvalue2=max(max(Rcorr));
index=find(Rcorr==maxvalue2);
Rcorr(index)=-inf;

[D11, D22]=ind2sub(size(Rcorr),index(1));
D=[D11, D22];
if sqrt((D11-centre(1))^2+(D22-centre(2))^2)>sqrt(2)
    if k==1
        return
    else
        k=k-1;
    end
end
end
```

B.18 *ArrayData_setabovethreshold*

```
function [ArrayData_abovethreshold]=ArrayData_setabovethreshold(ArrayData,...
    threshold)
%[ArrayData_abovethreshold, Moviegetframe]=ArrayData_abovethreshold(ArrayData,...
    threshold)
%This program searches through an ArrayData array, finds the values that
%are above a given threshold (for example 20) and sets them 50, where
%they can subsequently be highlighted via the program radarplotVideo (where
%they will appear green).
%Input variables
%ArrayData      This is the 4-dimensional array which we wish to examine
%               [4 dimensional array]
%threshold      This is the scalar quantity which we wish to set as a
%               minimum
%Output variables
%ArrayData_abovethreshold   This is the altered 4 dimensional array of same
%                           size as the input ArrayData
ArrayData_abovethreshold=zeros(size(ArrayData));
AboveMinimumValues=find(ArrayData>threshold);
ArrayData_abovethreshold(AboveMinimumValues)=35;
%(35 will appear yellow on 0 to 60 or 80dB scale)
```

B.19 *PAD_ROC*

```
function [tpfp, AUC, maxaccvalue, maxaccthreshold] = PAD_ROC(...
    PredictedArrayData, ArrayData_onlyobjects, maxtargetsignal, Doppler, range,...
     azimuth, scan)
%[PredictedArrayData, threshold, FAR] = SD_wholeprocess(ArrayData, ...
    ArrayData_withobjects, doppler, range, azimuth, scan, anM, YnM, RSM, ASM, ...
    maxd)
%This program calculates the ROC curve and also calculates the AUC values,
%maximum accuracy value, the corresponding maximum threshold value.
%Input variables
%ArrayData                  This is the input 4 dimensional data containing
%                           the predicted data (residual dB signal values)
%ArrayData_onlyobjects      This is the input 4 dimensional data containing
%                           the complex signal values of only objects
%maxtargetsignal            Scalar value of the maximum target dB signal
%Doppler                    This is the scalar representing the Doppler
%range                      The unbroken vector containing the range cells
%azimuth                    The unbroken vector of azimuth indexes
%scan                       The scalar of the scan numbers of the cells
%Output variables
%tpfp                       2-by-100 matrix containing the true positive vs
%                           false values
%AUC                        The scalar AUC value
%maxaccvalue                The maximum accuracy value
%maxaccthreshold            The corresponding maximum threshold value.
ArrayData_onlyobjects=20*log10(abs(ArrayData_onlyobjects));
dim=size(ArrayData_onlyobjects);
ArrayData_positive=zeros(dim);

for d=1:dim(1)
    for r=1:dim(2)
        for a=1:dim(3)
            for s=1:dim(4)
                if ArrayData_onlyobjects(d,r,a,s)>0.85*maxtargetsignal
                    ArrayData_positive(d,r,a,s)=1;
                end
            end
        end
    end
end

P=length(find(ArrayData_positive(Doppler,range,azimuth,scan)==1));
N=length(Doppler)*length(range)*length(azimuth)*length(scan)-P;
tpfp=zeros(2,101);

for X=0:1:100
    [PAD_detections]=ArrayData_setabovethreshold(PredictedArrayData,100-X);%...
        100-X is the threshold which decreases steadily
    TP=0;FP=0;TN=0;FN=0;
    for D=Doppler
        for R=range
            for A=azimuth
                for S=scan
                    if PAD_detections(D,R,A,S)>0 && ArrayData_positive(D,R,A,S...
                        )>0%positive detection among TARGET
                        TP=TP+1;
                    end
                    if PAD_detections(D,R,A,S)>0 && ArrayData_positive(D,R,A,S...
```

```
                        )==0%positive detection among NO TARGET
                        FP=FP+1;
                    end
                    if PAD_detections(D,R,A,S)==0 && ArrayData_positive(D,R,A,...
                        S)==0%negative detection among NO TARGET
                        TN=TN+1;
                    end
                    if PAD_detections(D,R,A,S)==0 && ArrayData_positive(D,R,A,...
                        S)>0%negative detection among TARGET
                        FN=FN+1;
                    end
                end
            end
        end
    end
    fprate=FP/(FP+TN);
    tprate=TP/(TP+FN);
    tpfp(1,X+1)=fprate;
    tpfp(2,X+1)=tprate;
    acc(X+1)=(TP+TN)/(TP+FP+TN+FN);
end

AUC=trapz(tpfp(1,:), tpfp(2,:));
[maxaccvalue, accloc] = max(acc(:));
maxaccthreshold=100-accloc+1;
```

B.20 *ArrayData_trailfunction*

```
function [ArrayData_onlytrails] = ArrayData_trailfunction(ArrayData)
%[ArrayData_onlytrails] = ArrayData_trailfunction(ArrayData)
%This program adds the signals from all scans and places them into a single
%scan, creating a trail of either targets or detections (raw data or
%residue signals will not be useful here).
%Input variables
%ArrayData      This is the 4-dimensional array which we wish to examine
%               [in order D, R, A, S]
%Output variables
%ArrayData_onlytrails   This is the altered 4 dimensional array of same
%                       size as the input ArrayData, but with only one scan
dim=size(ArrayData);
ArrayData_onlytrails=zeros(dim(1),dim(2),dim(3),1);
for s=1:dim(4)
    ArrayData_onlytrails=ArrayData_onlytrails+squeeze(ArrayData(:,:,:,s));
end
```

B.21 *CorrelationPlot_4dim*

```
function[Correlation_Matrix, AmountOfCorrData] = CorrelationPlot_4dim(...
    ArrayData, AzimuthValues, ScanCheckVector, Dopplerbin, Rangecells, ...
    Azimuthbounds, Scannumbers, DeltaVec)
%[Correlation_Matrix, AmountOfCorrData] = CorrelationPlot_4dim(ArrayData, ...
    AzimuthValues, ScanCheckVector, Dopplerbin, Rangecells, Azimuthbounds, ...
    Scannumbers, DeltaVec)
%This program reads the correlation coefficients between 4 dimensions
```

```
%and plots them in a graph with the various delta changes on the axes.
%
%ArrayData          This is our 4D-array which contains the complex signal
%                   values in dimensions order (D, R, A, S). Comes from
%                   leesdata6, 7 or 8
%AzimuthValues      The matrix containing the raw data from azmiuth values
%                   and scannumbers. Also comes from leesdata6, 7 or 8
%ScanCheckVector    A vector giving the scannumbers that were read, also
%                   coming from leesdata6, 7 or 8
%Dopplerbin         This is one scalar Dopplerbin we wish to set. Must be a
%                   positive integer within the Dopplers of our ArrayData ...
    array.
%Rangecells         The unbroken vector with the rangecell indexes we take
%                   to later calculate our correlation coefficients along
%Azimuthbounds      The 2-element vector containing the lower and upper bound
%                   of the azimuth values along which we calculate the
%                   correlation coefficients. The azimuthwidth is added onto
%                   each azimuth index of the first value.
%Scannumbers        The unbroken vector of the correct scannumbers - in ...
    ascending order
%Azimuthwidth       The scalar quantity of the number of azimuthindexes that
%                   will be read. Note the width of one azimuthindex is
%                   approximately 0.5degrees, depending on the radar used.
%                   Set to a maximum of 20 for reasons of accuracy.
%DeltaVec           This is a 2-element vector containing two scalars (the
%                   dimension-number 1,2,3,4 for D,R,A,S respectively) that
%                   we want in our graph. The first scalar for the x-axis,
%                   the second for the y-axis.
%For the program leesdata8, we use the input variable fid (file-identifier)
%which comes from program openlog.
%Output variable
%Correlation_Matrix    this is a matrix giving the correlation coefficient
%                      values for the 2 dimensions along the respective
%                      number of displacements for the dimensions. The size
%                      of the matrix depends on the dimensions chosen.
%AmountOfCorrData      a scalar of the quantity of data from which we are
%                      calculating the correlation coefficients.
DimensionInitials='DRAS';
Azimuthwidth=10;

AmountOfCorrData=length(Rangecells)*(Azimuthwidth)*length(Scannumbers);
Dopplerbin_Original=Dopplerbin;
DopplerbinLongVector=[1:24 1:24 1:24];

Azimuthindexes=find(Azimuthbounds(1)<=AzimuthValues(:,1) & Azimuthbounds(end)...
    >=AzimuthValues(:,1));
TotalDimVec=[3,2;2,4;3,4;1,2;1,3;1,4];

DeltaVec4D=zeros(4,2);
DeltaVec4D(1,1:2)=[-DeltaVec(1),DeltaVec(1)];
DeltaVec4D(2,1:2)=[-DeltaVec(2),DeltaVec(2)];
DeltaVec4D(3,1:2)=[-DeltaVec(3),DeltaVec(3)];
DeltaVec4D(4,1:2)=[-DeltaVec(4),DeltaVec(4)];

for plot=1:6
    DimVec=TotalDimVec(plot,:);
    Correlation_Matrix=zeros(DeltaVec4D(DimVec(2),2)-DeltaVec4D(DimVec(2),1)...
```

```
        +2,DeltaVec4D(DimVec(1),2)-DeltaVec4D(DimVec(1),1)+2);
    DispVec=[0,0,0,0];
for m=DeltaVec4D(DimVec(2),1):DeltaVec4D(DimVec(2),2)
    DispVec(DimVec(2))=m;
    for n=DeltaVec4D(DimVec(1),1):DeltaVec4D(DimVec(1),2)
        DispVec(DimVec(1))=n;
        for j=Azimuthindexes(1):Azimuthindexes(end)
            Azimuths_Original=      j:j+Azimuthwidth-1;
            Scanindexes_Original=   Scannumbers-ScanCheckVector(1)+1;
            Dopplerbin_New=         DopplerbinLongVector(Dopplerbin+24+DispVec...
                (1));
            Rangecells_New=         Rangecells(1)+DispVec(2):Rangecells(end)+...
                DispVec(2);
            Azimuths_New=           j+DispVec(3):j+Azimuthwidth-1+DispVec(3);
            Scanindexes_New=        Scannumbers-ScanCheckVector(1)+1+DispVec...
                (4);

            Original_Signal_Values= 10*log10( abs( squeeze(ArrayData(...
                Dopplerbin_Original,Rangecells,Azimuths_Original,...
                Scanindexes_Original))).^2 );
            New_Signal_Values=      10*log10( abs( squeeze(ArrayData(...
                Dopplerbin_New,Rangecells_New,Azimuths_New,Scanindexes_New))).....
                ^2 );
            %the above give us our original vector and a displaced vector (...
                displaced
            %according to our input)
            Vector_Original_Signal_Values=Original_Signal_Values(:)';
            Vector_New_Signal_Values=New_Signal_Values(:)';
            CCValues(j-Azimuthindexes(1)+1)= mean( (...
                Vector_Original_Signal_Values - mean(...
                Vector_Original_Signal_Values)).*(Vector_New_Signal_Values - ...
                mean(Vector_New_Signal_Values))) / (std(...
                Vector_Original_Signal_Values,1)*std(Vector_New_Signal_Values...
                ,1));
        end
        Correlation_Matrix(m+1-DeltaVec4D(DimVec(2),1),n+1-DeltaVec4D(DimVec...
            (1),1))=mean(CCValues);
    end
end

subplot(230+plot)
pcolor(DeltaVec4D(DimVec(1),1)-0.5:DeltaVec4D(DimVec(1),2)+1-0.5,DeltaVec4D(...
    DimVec(2),1)-0.5:DeltaVec4D(DimVec(2),2)+1-0.5,double(Correlation_Matrix));...
    colorbar;shading('flat');caxis([-1,1])
                axis([DeltaVec4D(DimVec(1),1)-0.5,DeltaVec4D(DimVec(1),2)+0.5,...
                    DeltaVec4D(DimVec(2),1)-0.5,DeltaVec4D(DimVec(2),2)+0.5]);
                xlabel(['\Delta in ' num2str(DimensionInitials(DimVec(1))) ' ...
                    dimension']);ylabel(['\Delta in ' num2str(DimensionInitials...
                    (DimVec(2))) ' dimension'])
end
subplot(232),title(['Correlation coefficient plot for Scout MK3 data (Thales ...
    Ref: BU-RS/PJV/13-0002)' 10 'Dopplerbin ' num2str(Dopplerbin) ', range ...
    indices ' num2str(Rangecells(1)) ':' num2str(Rangecells(end)),' (or ' ...
    num2str(round(Rangecells(1)*5.428)) ':' num2str(round(Rangecells(end)*5.428...
    )) ' m), azimuth values ' num2str(Azimuthbounds(1)) ':' num2str(...
    Azimuthbounds(end)) ' rad' 10 'at azimuth width ' num2str(Azimuthwidth) ' ...
    indices (~6^{\circ}) and scan numbers 1:21']);% num2str(Scannumbers(1)) ':'...
```

```
    num2str(Scannumbers(end))]);
return
end
```

B.22 *ArrayData_boundaryfunction*

```
function [ArrayData_withboundary] = ArrayData_boundaryfunction(ArrayData, ...
    range, azimuth)
%[ArrayData_withboundary] = ArrayData_boundaryfunction(ArrayData, range, ...
    azimuth)
%This goes through an ArrayData array, and marks white NaN boundaries in
%range and azimuth for all dopplers and scans. This is useful when marking
%perimeters around cells of interest for LP and SD prediction models,
%separating them from invalid cells of interest.
%Input variables
%ArrayData      This is the 4-dimensional array which we wish to examine
%               [4 dimensional array]
%range          The 2 element vector containing the first and second
%               index range boundary
%azimuth        The 2 element vector containing the first and second
%               index azimuth boundary
%Output variables
%ArrayData_withboundary   The altered 4 dimensional array of same size as
%                         the input ArrayData, now containing white
%                         boundaries
dim=size(ArrayData);
if length(dim)==3
    dim(4)=1;
end
ArrayData_withboundary=ArrayData;
for d=1:dim(1)
    for r=[range(1),range(2)];
        for a=azimuth(1):azimuth(2)
            for s=1:dim(4)
                ArrayData_withboundary(d,r,a,s)=NaN;
            end
        end
    end
end

for d=1:dim(1)
    for r=range(1):range(2)
        for a=[azimuth(1),azimuth(2)]
            for s=1:dim(4)
                ArrayData_withboundary(d,r,a,s)=NaN;
            end
        end
    end
end
```

Bibliography

[1] D. K. Barton and S. A. Leonov, *Radar Technology Encyclopedia*, Artech House, 1998

[2] P. S. Bell, "Shallow water bathymetry derived from analysis of x-band marine radar images of waves", *Coastal Eng*, 1999

[3] G. E. P. Box and N. R. Draper, *Empirical model-building and response surfaces*, ser. Wiley series in probability and mathematical statistics. Applied probability and statistics. New York, Chichester: J. Wiley, 1987, includes indexes. [Online]. Available: http://opac.inria.fr/record=b1081708

[4] G. Brooker, *Sensors and Signals*, Australian Centre for Field Robotics, 2006

[5] A. Buda and A. Jarynowski, Life-time of correlations and its applications vol.1, Wydawnictwo Niezalezne: 5-21, 2010

[6] G. Bulian, A. Francescutto, and C. Lugni "Theoretical, numerical and experimental study on the problem of ergodicity and 'practical ergodicity' with an application to parametric roll in longitudinal long crested irregular sea", *DINMA*, 2005

[7] J. V. Candy, E. F. Breitfeller "Receiver operating characteristic (ROC) curves: an analysis tool for detection performance", *Lawrence Livermore National Laboratory*, 2013

[8] COMET®Website at http://meted.ucar.edu/ of the University Corporation for Atmospheric Research (UCAR), sponsored in part through cooperative agreement(s) with the National Oceanic and Atmospheric Administration (NOAA), U.S. Department of Commerce (DOC). ©1997-2014 University Corporation for Atmospheric Research. All Rights Reserved.

[9] N.R Cook "Use and misuse of the receiver operating characteristic curve in risk prediction", 2007

[10] H. Dankert, "Retrieval of surface-current fields and bathymetries using radar-image sequences", *IEEE*, 2003

[11] H. Dankert and W. Rosenthal, "Retrieval of ocean surface wave fields using marine radar-image sequences", *IEEE*, 2004

[12] H. Dankert, J. Horstmann and W. Rosenthal, "Ocean surface winds retrieved from marine radar-image sequences", *IEEE*, 2004

[13] F. M. Dekking, C. Kraaikamp, H. P. Lapuhaa, L. E. Meester "A modern introduction to probability and statistics", *Springer*, 2005

[14] A. Farina, *Antenna based signal processing techniques for radar systems*, Artech House, 1992

[15] T. Fawcett "An introduction to ROC analysis", *Pattern Recognition Letters*, 2005

[16] D. R. Fuhrmann "Complex Random Variables and Stochastic Processes", The Digital Signal Processing Handbook (pp. 60-1, 61-1), *CRC Press*, 1997

[17] C. T. Han, L. K. Boon and T. K. Gaik "Solving non-linear equation by newton-raphson method using built-in derivative function in casio fx-570ES calculator"

[18] L. Henry, "A study of ocean wave statistical properties using nonlinear, directional, phase-resolved ocean wave-field simulations", *Massachusetts Institute of Technology*, 2010

[19] D. Kidner, M. Dorey and D. Smith, "What's the point? Interpolation and extrapolation with a regular grid DEM", *Geocomputation 99*, 1999

[20] E. W. Kang "Radar system", *Artech house*, 2008

[21] T. Lai "Evaluating probability forecasts", *Ann. Statist.*, Vol. 39, No. 5 (2011), 2356-2382, 2010

[22] B. R. Mahafza, *Radar signal analysis and processing using MATLAB*, Chapman and Hall, 2010

[23] H. Meikle, *Modern Radar Systems*, ser. Artech House Radar Library. Artech House, 2001

[24] B. Moran "Maths of radar", 20th Century Harmonic Analysis – a Celebration, *NATO Advanced Study Institute*, 2000

[25] J. Olkkone]n and H. Olkkonen, "Complex hilbert transform filter", *Journal of Signal and Information Processing*, Vol. 2 No. 2, 2011, pp.112-116

[26] J. Rice, *Mathematical statistics and data analysis* (3rd ed.), Thomson, 2007

[27] A. M. Simundic, "Measures of diagnostic accuracy: basic definitions", *eJIFCC*, 2008, http://www.ifcc.org/ifccfiles/ docs/190404200805.pdf

[28] J. Tessendorf, "Simulating ocean water", *SIGGRAPH 2001 Course notes*, 2001

[29] A. Thomson "A target simultion for studies of radar detection in clutter", *Defense R&D Canada*, 2002

[30] S. Vaseghi, "Advanced Digital Signal Processing and Noise Reduction", *John Wiley & Sons Ltd*, 2000

[31] W. B. Wu and H. Xiao "Covariance matrix estimation in time series", *Ann. Statist., 2012*, Vol. 40, No. 1, 466-493, 2011