Winter 2020

OBTAINING A REFERENCE FOR CALIBRATING BROADBAND MULTIBEAM SEABED BACKSCATTER

IVAN BODRA GUIMARAES
University of New Hampshire, Durham

OBTAINING A REFERENCE FOR CALIBRATING BROADBAND MULTIBEAM SEABED BACKSCATTER

by

Ivan Bodra Guimaraes

THESIS

Submitted to the University of New Hampshire

in Partial Fulfillment of

the Requirements for the Degree of

Master of Sciences

in

Earth Science: Ocean Mapping

December, 2020

This thesis was examined and approved in partial fulfillment of the requirements for the degree of Master of Sciences in Earth Science: Ocean Mapping by:

Thesis Director, John Hughes Clarke, Professor (Seabed Mapping)

Brian Calder, Research Professor (Seabed Mapping)

Thomas Weber, Associate Professor (Underwater Acoustics)

On October 8, 2020

Approval signatures are on file with the University of New Hampshire Graduate School.

ACKNOWLEDGEMENTS

Thanks to the Brazilian Navy for the investment in this qualifying program that is so enriching academically, where I had the opportunity to earn my first degree outside the military, remarkably thanks to my fellow Officer Peçanha that showed support beyond any expectation and helped me sailing in these unfamiliar waters by charting the most important steps to take on the master's program at UNH that he graduated in 2016.

To my advisors, professors Hughes Clarke, Weber, and Calder, it has been an honor and a privilege to be under the supervision of such brilliant minds. Special thanks to John who has a long-term relationship with the Brazilian Navy and shared his experience along with amazing breakfasts in Squamish and Sidney, Canada, before each survey day and all the support during the development of the thesis and the master's program.

Thanks to CCOM and UNH for the opportunity and the friendly environment that made me feel at home in your country.

Thanks to the Canadian Hydrographic Service for the unrestricted support for collecting the data in BC, and to SIMRAD-Kongsberg that shared the expertise and most of the equipment used to collect the data and the MATLAB codes that ignited the data processing developed in this thesis

With love and caring, thanks to my wife, Jeci, and to my family that have always been my safe harbor and to the technology that allowed us to be closer mainly during the pandemic crisis.

To my long-term friend that became more like a family of my own to me, Leo, his wife Nane and their lovely smiley son Arthur.

Jennifer for the friendship and camaraderie of one of the most empathic persons I have ever met, and to Liz, Alex and all the other UNH colleagues that offered their expertise unconditionally.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Standard calibration procedures for multibeam sonars currently only address the fidelity of the bathymetric data. Equivalent effort is needed to ensure that the acquired seabed backscatter strength measurements are referenced to a similarly precise level. This thesis presents an operational method for acquiring reference seabed backscatter data utilizing multiple pre-calibrated split beam echo sounders covering a wide range (45-450 kHz) of frequencies. This is needed to cover the full range of frequencies utilized by multi-sector multibeam systems operating in continental shelf depths.

The method considers both the frequency and the angle of incidence dependence of the backscatter strength of a homogenous seafloor region. By using a mechanically rotated plate, the split beam transducers, once calibrated, are able to collect the absolute angular response curve of the seafloor for any frequency within the bandwidth of interest. This thesis addresses the design, implementation and required processing to deliver the curves of selected areas. Although not part of this research, the next step would be to calibrate the desired multibeam echosounder for backscatter by comparing the results obtained by the systems over the same seafloor area.

The results obtained, reveal one of the most complete pictures of the continuous variation of the seabed backscatter angular response from 45 to 400 kHz. Significantly, this extends well above the 100 kHz level that normally defines the upper end of surface scattering model fidelity. As the chosen sites cover the main range of expected marine sediment

types (gravel to mud), trends in both frequency and grazing angle are apparent that might impact the choice of frequency used in multi-spectral backscatter imaging.

1. INTRODUCTION

Knowing the physical characteristics of the material on the seafloor can be a requirement for different reasons, such as identifying areas with soft sediments for safe anchorage, monitoring sediment type for dredge operations, establishing geotechnical properties for structure emplacement, mapping biological habitats and geological sediment distribution, among others.

The classical and unambiguous way to identify the seafloor is to obtain a physical sample of the material (usually termed "ground truth"). This, however, demands significant ship time to deploy and to retrieve the sampling equipment. An unavoidable byproduct of the discrete nature of physical sampling is that the data will consist only of sparsely sampled points distributed across the whole area. Therefore, to get around this efficiency limitation, it would be preferable to analyze the backscatter information from a multibeam echosounder (MBES), which are already being routinely operated by mapping agencies to provide complete seafloor coverage.

The use of the seabed acoustic backscatter as a classification tool is based on the well-established observation that different types of seafloor will present a particular acoustic signature (ICES, 2007) and thus can potentially be used to attempt seabed characterization. Such characterization can be based on physical models (Jackson et al., 1986) or an empirical approach through comparison of the acoustic response against sparse co-located ground truth.

Successful characterization, however, assumes that the acoustic signature for the same seafloor is at least reproducible. It has been repeatedly demonstrated, however, that from sonar to sonar, even of the same model, that signature can vary significantly (Hughes Clarke et al., 2008, Lamarche and Lurton, 2018), implying that the output is uncalibrated. Two such uncalibrated systems will provide unrelated responses for the same seafloor, which will prevent further comparative studies. As an intermediate solution, overlapping data from different systems are often adjusted to produce the same result (Hughes Clarke et al. 2008, Roche et al., 2018 and Weber and Smith, 2018), but even then, neither system is providing an absolute measure.

There is thus a pressing need to come up with a robust method of obtaining an absolute response. The advantage of having an absolute response of the seafloor is that it will be possible to both attempt regional classification using multiple systems as well as compare data at the same location from different systems collected at different times, thereby looking at seafloor changes.

The complexity of calibrating a MBES for backscatter is the main reason why this thesis presents an alternative method, based on an extension of the concept describe by Weber and Ward, (2015) and Eleftherakis, et al., (2018). The most well-known approach, standard within the fisheries community is to suspend a reference sphere of precisely known target strength under the sonar (Foote et al. 1987, Demer et al., 2015). For a vertically-mounted wide (7+ deg) beam system, the logistics of suspending, moving and locating the sphere within the single beam are manageable in the field (Foote et al., 1987), and could be done for narrower multibeam beams in a tightly controlled tank environment (Foote et al. 2005). The logistics, however, of precisely navigating a sphere under a ship-mounted MBES in open waters within an obliquely oriented beam as narrow as 1^o, is very unlikely to be done successfully. It has been attempted for narrow sector (only ±40 deg), wider beam multibeam systems such as the ME70 (Ona et al. ,2009) and at 200kHz for a Reson 7125 (Lanzoni and Weber, 2011). Note that an ME70 receiver is split both fore-aft and across track. An important limitation is that most MBES receivers are split only across track and thus the within-beam fore-aft location of a suspended target cannot be identified. To address this, Lanzoni and Weber (2011) presented a method using a split beam sonar together with the MBES to determine the position of the calibrating sphere. Additionally, for an MBES, the calibration needs to be individually performed for each receiver beamforming channel (typically several hundred today), from the nadir up to 65^o, or more, to port and to starboard and thus the target has to be moved well out to the side (e.g., Ona et al., 2009).

Removing the MBES from a ship so that it can be calibrated in a tank (as demonstrated by Lanzoni, 2011) is possible but has significant limitations, two of which are the far-field problem for low frequencies; and possible changes due to radiation pattern differences between the sonar mounted in the tank and on the bottom of a hull. The installation of such complex system is usually not done in a detachable fashion.

The method of calibrating split beam echosounders (SBES) with beam width of 7º are well understood by the fisheries community (Foote at al., 1987) where they need to estimate the amount and size of the fishes they are observing as well as tracking the variation of those numbers along the seasons. To achieve this, they need their systems to be calibrated to ensure consistent responses. They use spheres with known target strength (TS) and compare the values obtained by the SBES, so they can estimate the beam pattern and the correction for the maximum response axis (MRA) of their systems.

As a logical extension of the fisheries approach, by now deploying the pre-calibrated SBES to ensonify the seafloor and rotating the system through a range of grazing angles, it is possible to have the absolute angular response curve of that seafloor at the frequency used on the SBES (the first documented application was Urick, 1954). This approach is expanded on in this thesis.

Considering that the most common multibeam systems owned by the Brazilian Navy as well as the U.S. Naval Oceanographic Office and the N.O.A.A. Office of Coast Survey fleet operate in the frequency range from 45kHz up to 400 kHz, this project selected four SBES transducers with combined available bandwidth covering the range from 45kHz up to 450 kHz. By deploying these systems 10-60m above the seabed and rotating them in elevation angle and azimuth, the absolute seabed backscatter strength angular response may be derived across this frequency range for the local seabed.

That absolute response of the seafloor can then be used to calibrate the MBES by comparing the results at the same seafloor. The specific application of the calibration to the MBES would be the next logical step of the project but is beyond the scope of this thesis. This thesis will focus on the design, implementation, and proper data reduction needed to obtain the absolute angular response curve of a seafloor for the frequency range from 45kHz up to 450 kHz. As such a broad high-frequency range has rarely been attempted, especially with continuous frequency coverage and extending over the full range of grazing angles, the results, and particularly limitations of the method will be described, together with recommendations for future research.

PROBLEM STATEMENT:

The seabed backscatter strength can be used to estimate specific physical properties of the sediment water interface (as will be presented in Chapter 3), which may be used to infer seabed parameters of economic and environmental value to society. Examples include information on non-living natural resources to be exploited (e.g., aggregate or placer minerals, Roche, 2002) or as indicators of specific benthic habitat of value to commercial fisheries (Kenny et al., 2003). Therefore, measuring the backscatter response properly, in this case by ensuring calibration, is an essential precursor to identifying the seafloor.

As mentioned in the introduction, many nation's hydrographic services routinely use shelf-depth multibeam sonars within the frequency range from 40 kHz – 400 kHz that standardly deliver two products from the same survey: bathymetry and backscatter mosaics. There is usually no additional operational cost to collect backscatter data, as it is within the standard survey procedures already. The only additional effort involves archiving, processing, and presentation. Unlike the bathymetric processing sequence, for backscatter processing, calibration has not standardly been undertaken. Note that for deep waters (deeper than 500m), to allow for the greater path lengths, lower frequencies, in the range 12 kHz to 30 kHz, are commonly used. Although this thesis will not address those lower frequencies (mainly due to the prohibitive size and weight of the infrastructure required), the approach presented should be equally applicable.

As the backscatter strength output from these systems are not routinely calibrated, to ensure a usefully interpretable value, techniques have been developed to provide at least a stable and repeatable relative response. In this manner, areas surveyed by a single system will be classified according to their signature relative to each other. To provide a useful reference, representative physical seabed samples ("ground truth") are collected from areas that appear similar. Should a different uncalibrated system, however, be used it will provide an offset response for the same area, even if they operate with the same frequency and parameters. As a result, even though bathymetrically, two separate systems show equivalent data, the resulting backscattering map produced by those systems would be unrelatable, therefore, from the point of view of backscatter mapping, there would be no use to have a large area surveyed by more than one system.

If, however, a practical method of calibrating multiple systems to a common absolute reference were performed, two or more survey systems could then be used together and could be related to each other and compiled to cover a larger area. As an extension of this concept, with both absolute measurements and reference ground truth, a library might be developed to automatically identify the type of seafloor, and thus there could be no need to collect further ground truth anymore.

To facilitate future automated seabed characterization, the problem addressed in this thesis will be to design, implement and assess a practical method for calibrating MBES backscatter strength estimates over a wide range of frequencies using split beam echo sounders (SBES).

2. BACKGROUND

2.1. CONTRIBUTIONS TO THE RECEIVED INTENSITY

Multibeam and single beam echosounders are primarily used to determine the depth of the water column based on the time it takes for the sound wave to travel from the transducer to the seafloor and back. In addition, however, some of these sonars can provide the strength of the received acoustic intensity. As initially collected, however, this value is not an inherent property of the seafloor. Rather, its level is a result of a combination of radiometric, geometric, and environmental factors (one of which is the seafloor), each of which influence the resulting intensity. Only by unravelling the individual or combined effect of the non-seabed related factors can we obtain a measure of the bottom backscatter strength.

2.2. DEFINITION OF SEABED BACKSCATTER STRENGTH

To characterize the seafloor acoustically, one needs a measure that is an inherent property of the seafloor.

The dimensionless quantity called the scattering cross section per unit area of a scattering seabed surface (σ) is defined as

$$\sigma = \frac{r^2 I_s}{I_0 A} \qquad (1)$$

where r is the range, I_s is the scattered intensity, I_i is the incident intensity on the surface, and A is the effective ensonified area (Jackson et al., 1986). The logarithmic form of this, Sb = 10 $\log_{10}$ σ, is referred to as the scattering strength (Urick, 1975) and its value is an ensemble averaged quantity in dB.

A sonar equation is commonly used to describe the levels in each part of the sound travel, and a simplified (logarithmic) version can be designed as:

$$EL = SL - 2TL + TS \qquad (2)$$

Where EL is the received Echo Level, SL, is the transmitted Source Level, 2TL is the two-way (two times) the Transmission Losses caused by spreading and absorption and TS is the Target Strength that contains the S_b

$$TS = S_b + 10\log(A) \quad (3)$$

A being the ensonified Area.

So, S_b can be described as:

$$S_b = EL - SL + 2TL - 10\log(A) \quad (4)$$

While TL and A can be conventionally calculated knowing the geometry and the frequency, the EL and SL terms require calibration. Notably, it is adequate to achieve just their combined calibration (EL-SL, the product of the source level and receiver sensitivity) simplifying the process.

Once these steps are undertaken, finally an estimate of S_b can be attempted. For a given seabed type, S_b varies only with grazing angle (Gr), azimuth, and frequency.

2.3. PHYSICAL CONTROLS ON S_b

The measurement of interest in this thesis is the scattering strength of the sediment-water interface (the seafloor). Common types of seafloor range from soft muds to solid rock. Such a range of wide material types needs to be parameterized in a way that can be usefully input into a physical model (e.g., Jackson et al., 1986). The parameters most usually obtained include:

Interface Roughness (defined by power and exponent of 2D spectra). Even though the mean grazing angle between the beam vector and the normal to the mean seabed surface is estimated here, stereo photography of the seabed (Briggs, 1989) reveals that the sediment water interface is not really a simple plane. For a small area, the height deviations about a mean plane may be described by a probability distribution function (de Moustier, 1986). For a given height distribution variance, however, the horizontal scale over which the heights change can be quite different and thus an alternate way of presenting roughness distribution is a two-dimensional spectrum (Jackson and Briggs, 1992). The most common way of parametrizing such a spectrum is to estimate the mean slope and intercept of that spectrum in log-log space. This is termed the spectral slope and exponent.

As the seabed is a 2D surface, there is always the potential for the roughness to vary with azimuth (for example ripple fields). Most models, however, reduce the 2D spectrum to an equivalent 1D version. Lurton et al. (2018), however, have clearly identified azimuth dependence in some seabeds. For this reason, part of the experimental design is to deliberately rotate the SBES in azimuth (in addition to elevation angle) to see if there is such an azimuthal heterogeneity in the calibration site.

Given that most spectrums have a slope, the amount of roughness clearly varies with length scale and thus, as scattering depends on the scale of the roughness relative to the acoustic wavelength (Ogilvy, 1991), one can expect roughness scattering to be frequency dependent. For this reason, we expect to observe some frequency dependence in the calibration results.

Impedance Contrast. Whatever the interface roughness, the amount of scattered energy will depend on the seabed surface impedance contrast. Impedance is defined as the product of the sound speed and bulk density in the material. Thus, the sound speed and bulk density of the surficial sediment and the overlying water must be known. Note that bulk density is the density of the combined solids and liquids (e.g., sand grains and interstitial fluids). Additionally, to provide the contrast, the sound speed and bulk density of the overlying water mass is required.

Volume Scattering contributions. Given that the surface reflection coefficient is never 1.0, there will always be the potential for energy to refract through the sediment water interface into the volume of the sediment

For low impedance sediments (e.g., fine sands and muds), even with a rough interface, most of the energy will refract into the sediment and thus there is a secondary mechanism for further scattering if there are lateral impedance variations buried in the sediment. Examples of this might include buried pebbles, shells, or burrows. To parametrize this one needs to have an estimate of the length scales (3D spectrum) and magnitude of the buried impedance contrasts (e.g., Tang et al., 2002, Briggs et al., 2010). In practice this is an extremely hard parameter to measure and thus usually a single constant is empirically estimated.

Whatever the volume heterogeneity in the sediment, the scattering strength will depend on how much of the energy arrives at the buried scatterer. Thus, the attenuation in the sediment, which is normally at least 10^3 times larger than in seawater, is important. As acoustic attenuation in the sediment is strongly frequency dependent (Peng et al., 2004), the effective depth into the sediment to which sound penetrates will be correspondingly frequency dependent. So again, as part of the calibration, for soft sediment it would be expected that, if volume scattering is the dominant influence, there will be frequency dependence observed.

Time variability of Scattering Parameters: If a seafloor reference site is intended to be used more than once (see section 3.3), one of the concerns is always that the scattering properties may vary over time. For example, the impedance contrast is also dependent on the impedance of the overlying water and thus, for a given sediment type, the ratio could have seasonal dependences as the overlying oceanography changes.

For the case of interface roughness, common roughness elements at scales approaching the wavelength such as ripples and burrowing can vary with surface wave and or seasonal bioturbation. And for the case of volume scattering, again if bioturbation is a seasonal phenomenon then this may influence the long-term scattering strength. All these factors must be considered in the selection of calibration site.

Dealing with speckle – statistical distribution of scatter strength: For an extended target such as the seafloor, the instantaneous scattering is controlled by the contribution of multiple scattering components all with random phase. The net result is that, for the same seafloor, successive instantaneous estimate of the backscatter strength will vary significantly. The outliers are termed speckle. Only the mean value and the distribution is of interest. Thus, to get a stable mean estimate of the bottom backscatter strength, multiple measurements must be obtained to observe that population. This requirement for multiple observations will be addressed as part of the sampling density considerations in the experimental design.

2.4. SBES CALIBRATION FACTORS

While the ultimate intent of the method being developed is to use it to calibrate survey multibeam echosounders, this thesis is primarily concerned with the extraction of a reference backscatter measurement using a single (split) beam echosounder (SBES). Thus, the study here will be constrained to briefly review those radiometric and geometric factors in the split beam while its calculation is more detailed in the section 4.1.

Note that all of these factors are also common to the MBES, but they are more complex as there are multiple beams, with asymmetrical shapes and variable center frequencies and pulse lengths. Additionally, several of these factors are not a constant if the MBES operational settings are changed. Such changes are common as the depth varies.

Range effects – as part of the sonar equation, the transmission losses (TL) are due to the spherical spreading and the attenuation effects in the two-way travel, making TL dependent on the range. It is worth noting that the attenuation effects are frequency dependent. For this adequate knowledge of the temperature and salinity (and strictly pH) variations with depth must be known (Carvalho and Hughes Clarke, 2012).

Beam pattern – Both the transmit and receive beam pattern for a split beam echo-sounder are frequency dependent. For the purposes of this study, just their combined product is important. Any point target located within that pattern will have its apparent target strength modulated by that pattern. The calibration process deliberately involves moving the reference target across the beam to quantify this (see section 4.1). This combined pattern will, of course be frequency dependent.

The beam pattern observed by the sphere calibration is then used to determine an effective beam width. This is the equivalent solid angle, at normal incidence, over which the power at the MRA would be spread to provide the equivalent power that result from integration of the real beam pattern over the whole sphere. This equivalent angle dimension is then used to estimate the ensonified area.

Ensonified area – for the sphere calibration, the ensonified area is not considered, as the method compares the TS with a model sphere. For a bottom ensonification, however, the instantaneous ensonified area is required to normalize the S_b response (see section 3.8). The slant range, the grazing angle, and the two-way equivalent beam width defined by the beam pattern are the predominant factors that influence the area calculation. Figure 1 illustrates how the ensonified area for an oblique incidence is approximated by the rectangular area of $S_1 x S_2$ limited by the pulse length (for CW case) or the length of the Matched-Filtered extract (for the FM case used herein) on the across-track dimension, referred to as PL on the figure, and by the effective fore-aft beam width on the along-track. For a normal incidence, however, both the across and along-track dimensions would correspond to their respective equivalent beam widths.

For the Figure 1, consider:

ψ_{ac} – Two-way equivalent beam angle across-track.

ψ_{al} – Two-way equivalent beam angle along-track.

θ – Grazing angle.

PL – Pulse Length (for a CW pulse) or the length of the Matched-Filtered extract (for the FM case used herein)

R – Range

S_1 and S_2 – correspondent sides of a rectangular area calculated by the simplification explained herein.

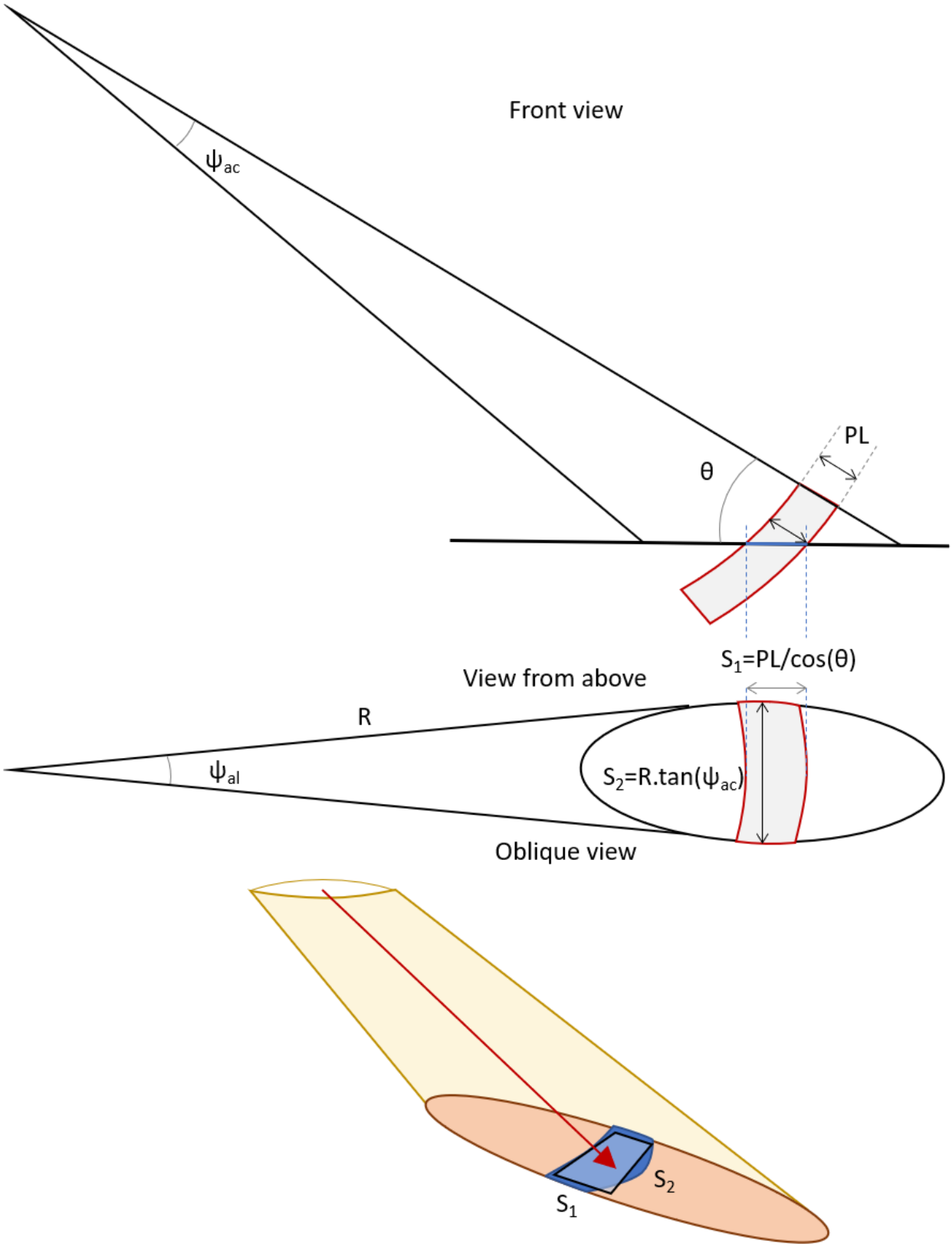


Figure 1 - Sketch of the ensonified area of an oblique incidence.

2.5. PREVIOUS WORKS

Preexisting experiments measuring frequency and grazing angle dependence of backscatter

The first studies mentioning the frequency and angular dependence of the seabed backscatter response date from the period of the Second World War (NDRC, 1946). In that study, bottom backscatter strength was first reported, and an estimate of the frequency dependence was made. Notably the authors stated, "An irregular variation of (the backscattering coefficient) with frequency was noted (…), but this variation is less than the estimate error of calibration." There was thus already the recognition that transducer calibration would be a limiting factor in our ability to obtain useful seabed backscatter strength measurements.

Some years later, Urick, in 1954, performed a study on four different areas ("from a hard rocky bottom to a soft muddy one") using a transducer with a range of frequencies varying from 10 kHz to 60 kHz installed on a structure that allowed tilting the equipment to compare the response at different grazing angles. He also used a mine as a "reference sphere," "to obtain a rough calibration." He used a continuous wave (CW) pulse and, knowing that duration together with width of the beam pattern and the approximate grazing angle, estimated the ensonified area. This was the first detailed description of the method that this thesis builds on. Specifically, a mechanically rotated single beam of known beam pattern, source level, pulse length and receiver sensitivity.

In the 1980s, as part of a program supported by the Office of Naval Research a number of other authors (Boheme et al.,1985, Boehme and Chotiros, 1988, Stanic et al., 1988, Stanic et al., 1989 among others) published articles exploring important details about what contributes to the backscatter response, such as grazing angle, frequency dependence, azimuth dependence, roughness of the seafloor and ensonified area.

The ensonified area, in particular, is important to properly normalize the backscatter response, but it needs to have the grazing angle properly assessed, therefore the raytracing and the seafloor slope play an important role here as shown by the Figure 2 (Boheme et al., 1985). In contrast to the approach used in 1985, for the method developed here, the grazing angle was determined without the need for raytracing (see section 3.8).

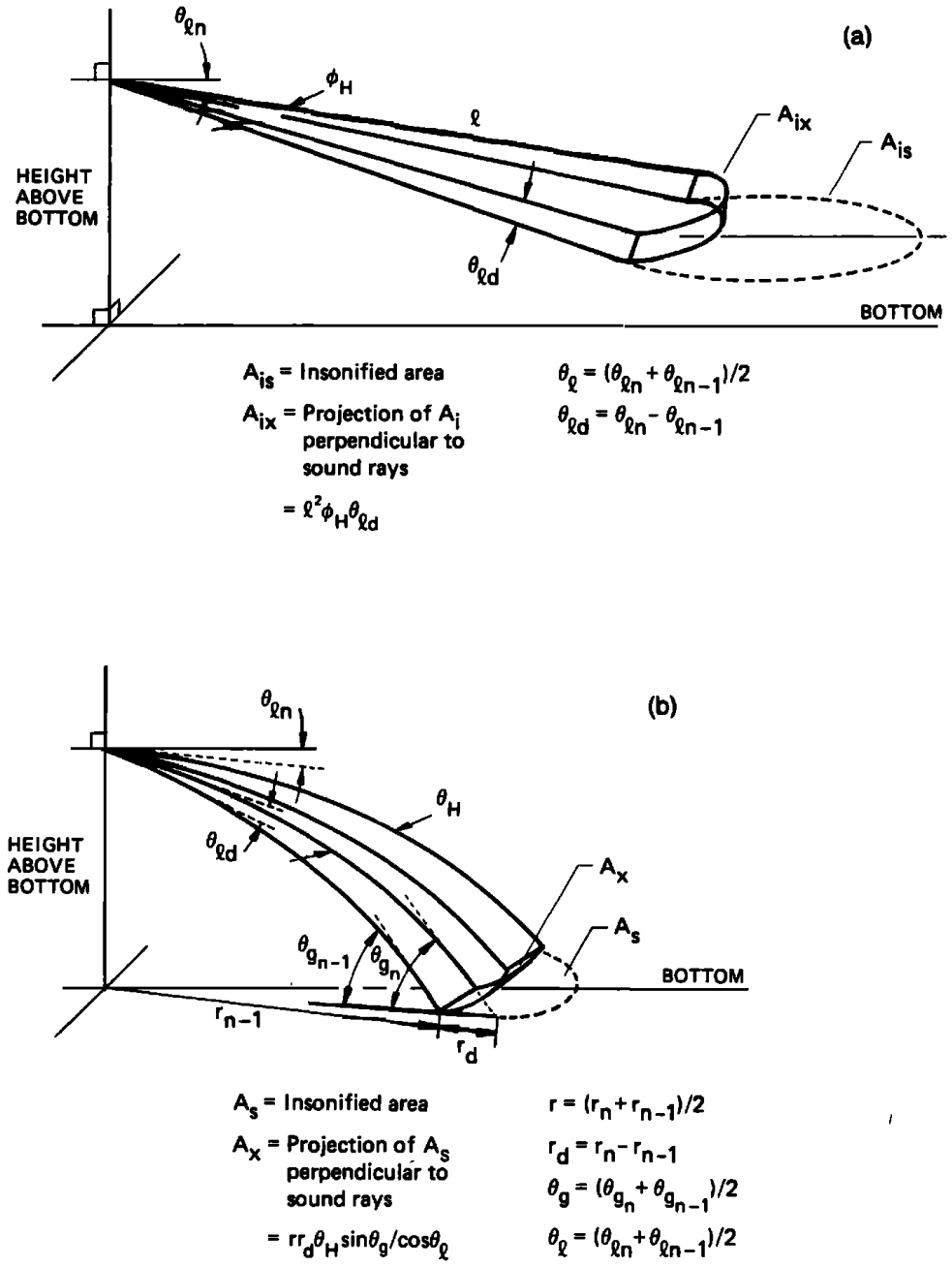


Figure 2 - Spreading of rays with range "l," launch angle "θ_l." (a) Ideal case, (b) with ray tracing (Boheme et al., 1985).

The acoustic measurements performed on sandy bottoms near San Diego, CA by Boehme et al., (1985) were "made using transducers mounted on a tripod assembly about 4m tall rested on the bottom" pinging at a range of grazing angles as low as 2^o - 10^o and a range of frequencies of 30 - 95 kHz (see Figure 3) with CW and FM pulses of small bandwidth (1-4kHz) varying the pulse length. Boehme and Chotiros, (1988) reproduced the experiment with similar parameters over a frequency range of 30 - 80 kHz at low grazing angles in a sand bottom in Charleston, SC in 20m deep waters. The published paper included a comparison of the results (see Figure 4, on the left-hand side plot) between the fixed platform method with a different approach, a moving sonar platform over a long track (tens of kilometers). A plot comparing the two regions showing a frequency dependence at the low grazing angles analyzed in the experiments was presented by Boehme and Chotiros, (1988), and is reproduced here on Figure 4, on the right-hand side plot.

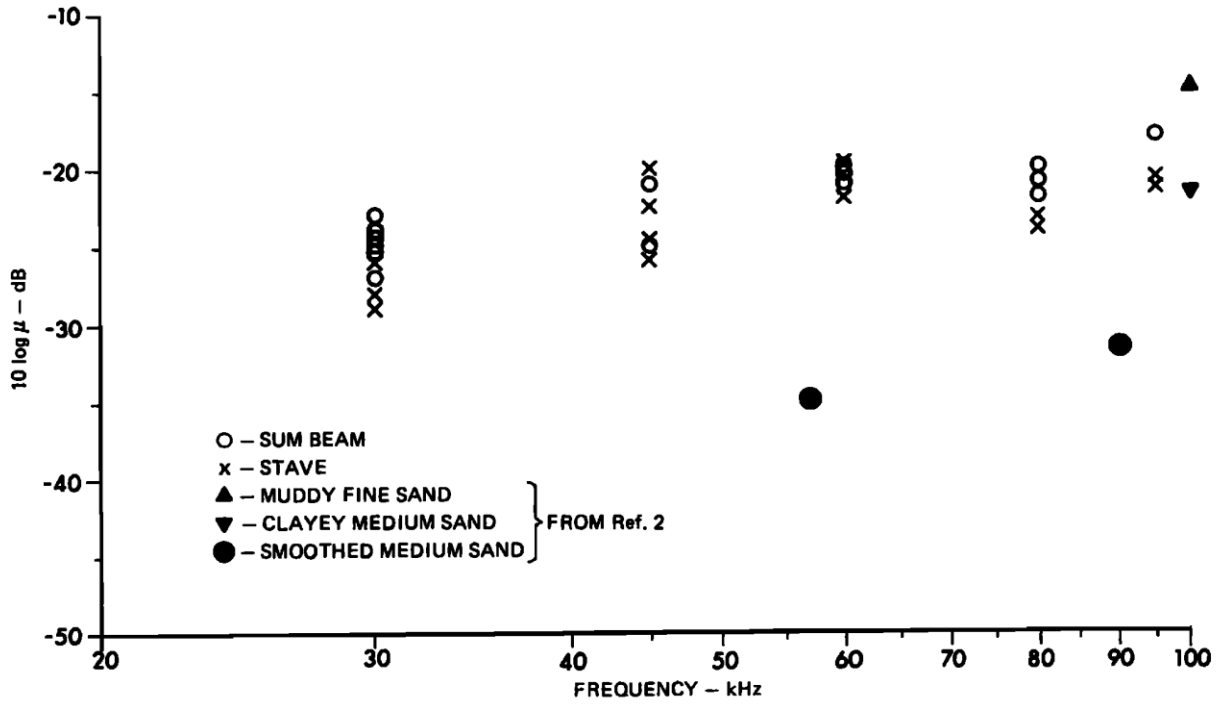


Figure 3 – "Estimated values of the bottom backscattering characteristic 10 log(μ) versus frequency for the fine sand bottom region near San Diego" (Boehme et al., 1985).

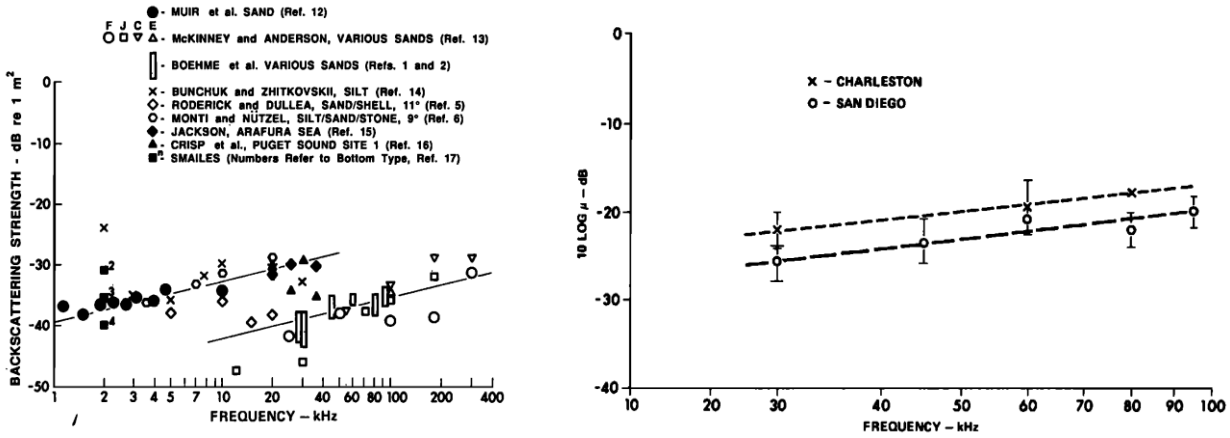


Figure 4 - On the left side, the frequency response at grazing angle of 10º from various sources. On the right side, a comparison between (Boehme et al., 1985) and (Boehme and Chotiros, 1988).

In another study, also in sandy area, south of Panama City, FL, Stanic et al., (1988) collected bottom backscattering measurements as a function of frequency (20-180 kHz), grazing angle (5º-30º), azimuthal angle, and CW pulse of length 5μs – 10ms using a support tower. The experiment conducted in Jacksonville, FL (Stanic et al., 1989) detailed the results from a shelly bottom with the same acoustic tower within the same frequency and grazing angle ranges. This tower consists in "a twin-hull catamaran design with a vertical tank supporting an instrument chamber, a triaxial positioner, and a two-dimensional array mount." Considerable logistics were thus required involving towing to the predetermined position, on site instrument assembly and then deployed on the ocean bottom (see Figure 6). Also, divers had to be employed to connect various cables between the tower and the support ship that must be moored nearby. Both sources and the acoustic receiving system (an array of a dozen of hydrophones) were laboratory calibrated.

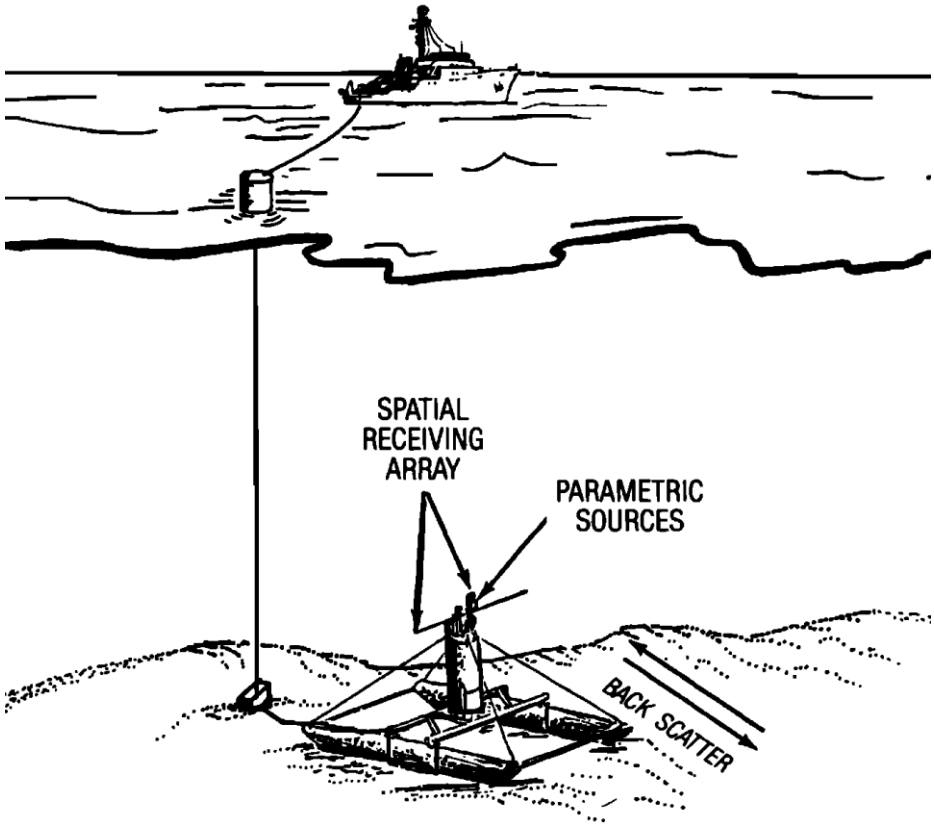


Figure 6 – The acoustic tower configuration (Stanic et al.,1988).

The angular response curves (ARC) for Panama City and Jacksonville are reproduced on Figure 5 and Figure 7, respectively.

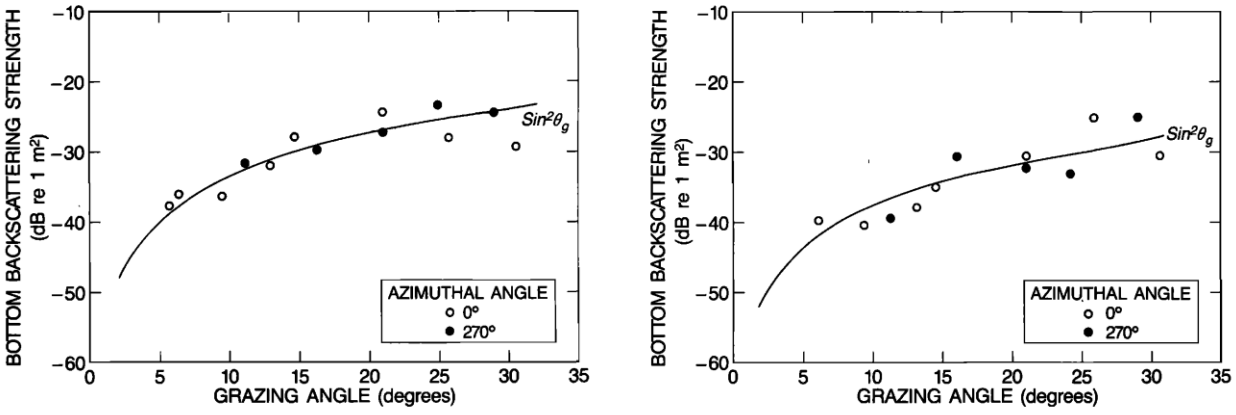


Figure 5- Angular response curves at 40kHz (left) and at 150kHz (right) in Panama City (Stanic et al., 1988).

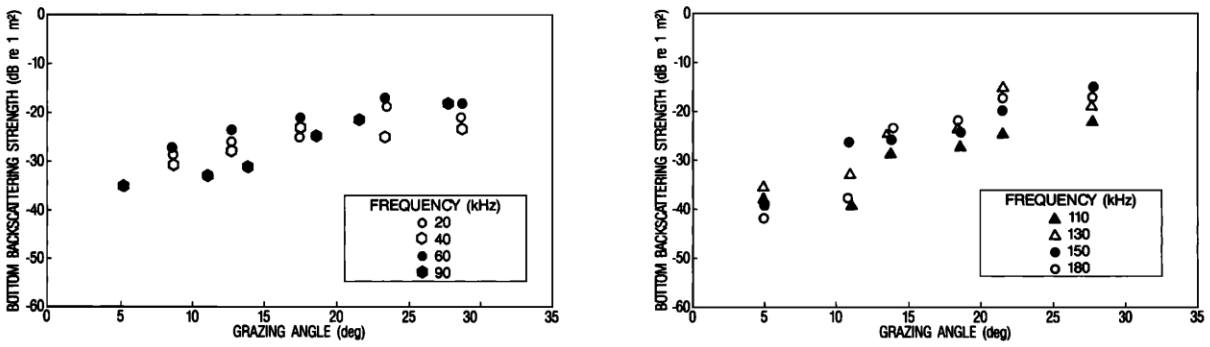


Figure 7 - ARC for lower frequencies (left) and higher frequencies (right) in Jacksonville (Stanic et al., 1989).

From these 1980s studies, a comparison between the CW and FM pulses of short bandwidths produced similar results. Although Boehme et al., (1985) stated that "an examination of the frequency dependence reported for bottom backscattering strength results (was) inconclusive by virtue of the large variations in reported results, even for similar bottom type," other studies suggested that there could be inferred some frequency dependence (see Figure 8 reproducing Stanic et al., 1989).

Gensane, (1989) developed a device called a "Reverberometer" which consisted of a parametric array as the transmitter capable of transmitting CW and FM pulses varying from 8kHz to 40 kHz and a hydrophone as the receiver capable of digitizing the received signal at 100 kHz. The Reverberometer was lowered near the bottom in shallow waters on several different areas of sand, gravel and clay and the grazing angles varied from 4^o to 90^o. The results showed the expected angular response curves but no frequency dependence between 8 and 40 kHz except near nadir where it was observed a decrease in the response along the frequency.

As can be seen, unlike the experiment in this thesis, most of these studies did not observe larger grazing angles and usually only sparsely sampled in the frequency domain and were usually limited to less than 200 kHz. These experiments also required difficult logistics, including a moored ship and divers to deploy the fixed bottom. This thesis intends to replicate an anchored version of the bottom ensonification in a much simpler manner covering a wide range of frequency and all the practical grazing angles.

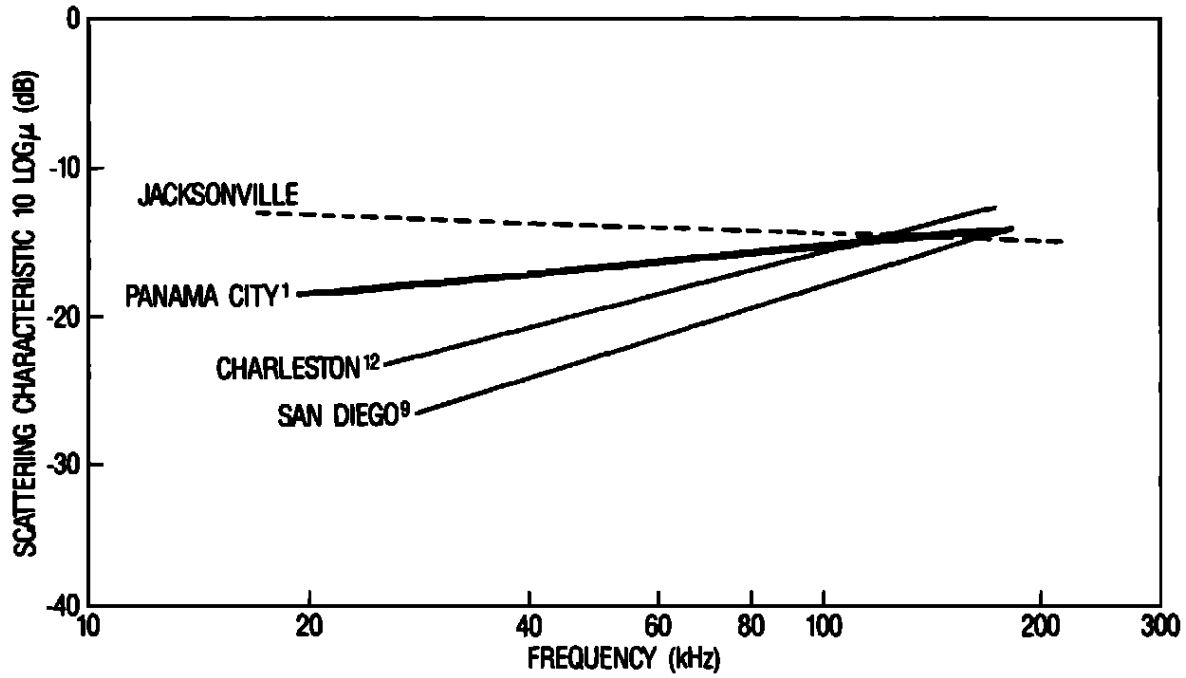


Figure 8– Comparison of frequency dependence for the areas from the 1980s studies (Stanic et al., 1989).

Recent developments in acquiring Reference S_b

Up to the end of the 90's most of the seabed backscatter experiments used relatively narrow band CW pulses and thus, the backscatter strength was only valid for a specific center frequency (even though multiple transducers were used to obtain stepped results at discrete offset frequencies). Of interest though for this project is the continuous frequency variation as modern multi-sector systems, employing broad band transducers, have multiple relatively tightly spaced center frequencies for each sector.

For any single transducer there is an available bandwidth which could be exploited, either by progressively switching center frequency for subsequent CW pulses or, more efficiently, by utilizing a swept FM pulse covering the full available bandwidth. In 2015, Weber and Ward were the first to demonstrate this approach when they performed an experiment using an FM pulse (swept from 170 kHz to 250 kHz) pointing at just a single fixed grazing angle (45 degrees).

The calibration involved comparing the spectrum of the observed TS of a sphere against a model (see later discussion in section 3.4). Once calibrated, to overcome the randomness of the backscatter response and obtain a stable seabed backscatter strength estimate, they averaged a number of pings along a survey line while over a seabed with assumed homogenous material type. This was acquired from an underway vessel.

More recently, the French National Institute for Ocean Science, *Institut français de recherche pour l'exploitation de la mer* (IFREMER), used two calibrated split-beam transducers using CW pulse centered at 200 kHz and 333kHz and performed a cross calibration with MBES in the field. The SBES were calibrated using the method of sphere calibration.

In summary, these previous pioneering studies describe the preceding work that has led to the concept developed in this thesis and provide substantial background on the evolution of the calibration method proposed here. Herein, the swept frequency approach of Weber and Ward (2015) will be combined with mechanically rotated geometry of Eleftherakis et al. (2018).

3. METHODOLOGY

3.1. OVERVIEW

The objective is to have a reference seabed backscatter data against which one can calibrate multibeam backscatter. The method proposed in this thesis is based on a combination of the Weber and Ward (2015) and IFREMER approach (Eleftherakis et al., 2018) and adopts the same first step in calibrating the reference SBES with the well-known method of the model sphere widely adopted in the fisheries community.

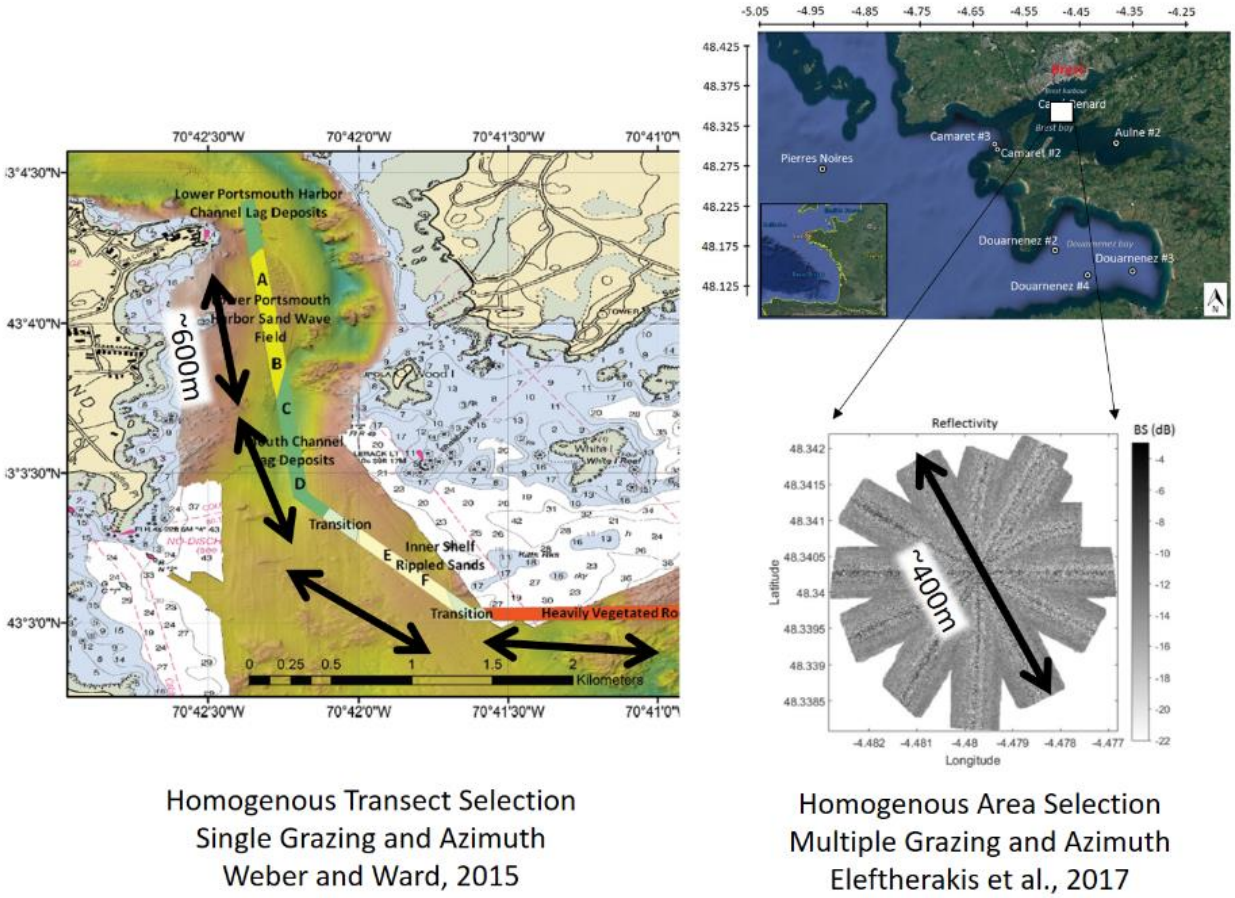
Target strength values of reference spheres were calculated by equations presented in MacLennan (1981) and properties of tungsten carbide and copper spheres were defined in MacLennan and Dunn (1984) and Foote, et al. (1981), respectively.

With the SBES calibrated, a suitable site to collect the reference data needs to be chosen. That area is selected bearing in mind the following considerations: it should be flat, homogeneous, isotropic, low traffic, protected from weather and currents, and as far from sediment discharge and turbidity currents (to avoid seabed changes), as possible.

Previous Implementations

For comparison, Weber and Ward (2015) used the "NEWBEX" reference line which was a long corridor with a few well-known seabed types (Figure 9). Prior geological knowledge (Ward and Birch, 1999) indicated that these sediment types did not change with time. Measurements were averaged over elongate transects ~ 600m long in which the seabed was known to be all the same. Along that transect the azimuth was always the same (orthogonal to the ship track). The stability of this corridor has now been evaluated over several years of collection (Weber et al., 2018). Notably, the corridor is in a high energy environment and thus mainly consists of coarser sands and gravel.

Eleftherakis et al., (2018) in contrast picked a few local areas in and around the Rade de Brest (see Figure 9) that they believed might be stable. The reference and MB data were deliberately collected along short (~400m long) transects within this area at a full range of azimuths. A few sites were subsequent recognized to not be sufficiently temporally stable and/or have a strong azimuth dependence (Lurton et al., 2018)



Homogenous Transect Selection
Single Grazing and Azimuth
Weber and Ward, 2015

Homogenous Area Selection
Multiple Grazing and Azimuth
Eleftherakis et al., 2017

Figure 9 – Comparison of the philosophy of selecting homogenous areas between Weber and Ward, 2015 and Eleftherakis et al., 2018.

Implemented approach to collecting reference data.

For this thesis, a slightly different approach to obtaining reference data was utilized. Rather than collecting the SBES data from a moving vessel with a rigid, potentially rotatable frame, the decision was made to be stationary, and have a suspended frame that could easily be rotated in azimuth and elevation angle (similar in concept to Gensane, 1989). Interestingly, although stationary, from a 20-50m altitude above the seafloor, the resulting samples are still spread out over an area with a diameter of several hundred meters, comparable to the transect scale used by the previous experiments. The only difference is that the high grazing angle data is preferentially restricted to a much smaller area. This choice was in part a financial decision and allowed much simpler and less complex deployments.

For both the previous experiments, and this thesis, the SBES sonars are held close to the surface so that the slant range to the seabed will grow with depth. For depths less than 30m, this is manageable as even the highest frequency echoes are just discernable at those slant ranges. For deeper depths, however, this would not be practical as was found for two of the areas (see results section 3.8).

To address the potential azimuthal complications identified by Lurton, herein the SBES was deliberately manipulated to sample the seafloor in different azimuths to test for any variation in the angular response with heading. At each azimuth, in turn, it was rotated through all grazing angles from normal incidence to as low as is practical (~5-10 deg grazing). And the data was acquired across the full range of frequencies that the non-calibrated MBES tested operates (EM2040 180-400 kHz and EM710 MkII, 40-100 kHz). Of particular concern is to obtain enough redundant measurements across the full range of grazing angles so that we can obtain valid statistical measures of the mean and variance.

This last point addresses the fact that any instantaneous measurement of scattering intensity has a strong speckle component (Ogilvy, 1991). Most scattering models expect a high variance (standard deviation equivalent to the mean for true Rayleigh scattering). Thus, only by making a large number of observations that can be averaged (see section 3.8), can a stable mean angular response curve be extracted.

Once the SBES is calibrated (see section 3.4), the response obtained, after considering all the losses and normalizing by the area (see details in section 3.8), is our best estimate of the absolute response curve of that seafloor which should be obtained by any calibrated system.

Although not a part of this thesis, the next step would be, on the same area of the seafloor, for the MBES to collect backscatter measurements covering all the different grazing angles and azimuths. If the MBES has a range of operating modes (combinations of pulse lengths, beam widths, center frequencies) the data would need to be collected separately for each mode. The response obtained by the MBES will not be equal the absolute response curve as it is not calibrated yet. For a given grazing angle at the center frequency used by the MBES, the difference between the reference and the MBES backscatter strength estimate is the calibration that should be applied to the MBES (e.g., Figure 10 of Eleftherakis et al., 2018). In practice, the simple difference is not actually a sufficient corrector, as that instantaneous difference is often a reflection of sector beam patterns which may rotate as the vessel rolls and thus corrupt the measurement at a particular grazing angle differently depending on the vessel roll (Hiroji and Hughes Clarke, 2016 and 2017).

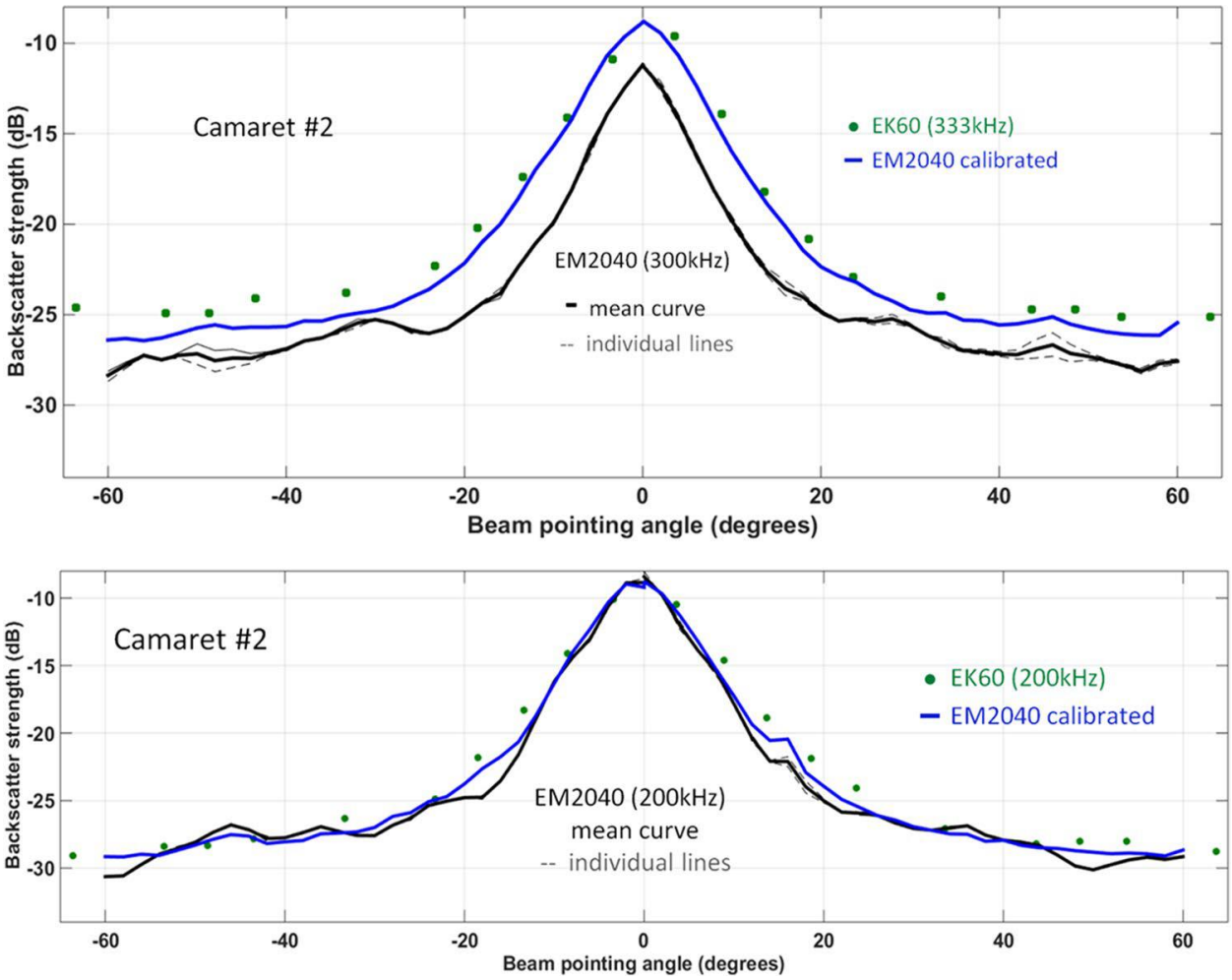


Figure 10 - Angular response curves of calibrated SBES EK60 (200kHz on top and 333kHz on bottom) and an uncalibrated MBES EM2040 (from Eleftherakis et al., 2018).

3.2. EQUIPMENT

The equipment employed in this thesis was composed of:

Table 1 - List of Transducers

Owner	Model	Freq-mode FM up	PL	Power
SIMRAD	ES70-7C	45kHz to 95kHz	2.048ms	150W
SIMRAD	ES120-7C	90 kHz to 170 kHz	2.048ms	150W
SIMRAD	ES200-7C	160 kHz to 260 kHz	2.048ms	100W
CCOM	ES333-7C	280 kHz to 450 kHz	2.048ms	50W

Table 2 - List of other equipment

Equipment	Owner	Model	Parameter
IMU	UNB	MRU-6	rate:100Hz
Rotatable plate	CCOM	---	---
Sphere	CCOM	WC	Diam: 38.1mm

- **Split beam echosounders (SBES), model EK-80, manufactured by SIMRAD**: They (or their predecessor the EK60) are one of the most commonly used by the fisheries because of the possibility of localizing the targets within its 7^{o} of beam width. As part of the system, there were four wide-band transceivers (WBT) and four transducers, consisting of one set provided by the Center of Coastal and Ocean Mapping (CCOM), the ES-333-7CD (operating frequency from 280 kHz to 450 kHz), and three sets provided by SIMRAD, the ES70-7C (45kHz to 95kHz), ES120-7C (90 kHz to 170 kHz), and ES200-7C (160 kHz to 260 kHz).

- **An Inertial Measurement Unit, MRU-6,** that could calculate the pitch, roll magnetic heading and heave, provided by the University of New Brunswick (UNB). This outputs a serial string (EM300 format) of orientation at 50 Hz into the WBT for inclusion in the EK80 data stream.

- **A mechanically rotatable plate**: manufactured by CCOM, for mounting the transducers and the MRU-6. The rotation mechanism merely consisted of ropes to the surface where an operator was free to tilt and scan with the plate.

- **A tungsten-carbide (WC) sphere of 38.1mm of diameter** as the reference sphere for calibrating the SBES, provided by CCOM.

The facilities and research vessel available for the project were:

- The Centre for Ocean Engineering's Tank was used for lab tests and to calibrate the ES70-7C, ES120-7C and ES200-7C transducers.

- **The CCOM's R/V Gulf Surveyor** for field tests in Portsmouth, NH.

- **The Canadian Hydrographic Service's CSL Heron,** a hydrographic/geophysical/ oceanographic survey launch, with length of 10m and draft of 1.15m, for collecting the data in BC, Canada that was analyzed in this thesis.

3.3. SELECTION OF SUITABLE AREA

As mentioned, the reference area should fulfill some requirements as detailed in the sequence:

Flat, or at least **planar** - to avoid errors in calculating the grazing angle due to rapidly changing slopes. The greatest concern exists mainly near nadir where it is not possible to derive the grazing angle from the phase ramp (see section 3.8).

Roughness – strictly this factor is frequency dependent as relief at scales shorter than the wavelength will be irrelevant. The roughness scale of concern here is at wavelengths that fit within the projected ~5-degree beam footprint thereby impacting the estimation of the local grazing angle. As a result, the seafloor should be as smooth as possible to avoid errors in the grazing angle estimation and rapid fluctuations in the response due to unresolved slope changes. Also, some features such as sand ripples which would have roughness that is azimuth dependent would therefore be undesirable.

Homogeneous – as the plate is rotated to achieve lower grazing angles, the horizontal distance of the covered area can be ten times the depth for grazing angle up to 5.7 degrees which can lead to a large area of coverage. To have an angular response curve of the seafloor, the sediment on the whole covered area (e.g., 500m diameter for 50m altitude) should be the same.

At the longest scales there should thus be no regional sediment type variations across the ~200-400m diameter testing site. At the shortest scales, the minimum length scale over which homogeneity is apparent needs to be qualified. Many seafloors, if viewed at optical scales have local patchiness due to, for example, presence/absence of shell debris or burrow spacing. The idea here is that this heterogeneity should take place at scales small with respect to the size of the instantaneous ensonified area. Typical footprint dimensions are a function of the range as discussed further in this section (Impact of water depth on site suitability). Bottom photography did indeed demonstrate that there is heterogeneity at small scale in almost all the areas.

Navigational restrictions – each government, institution or research group has its safety procedures and should be aware of the traffic as well as the predominant weather conditions mainly regarding the sea state and the tides and currents around the selected area. For example, traffic lanes or active fishing zones should be avoided. Similarly, exposed water is to be avoided as the suspended plate would tend to rotate too rapidly and the safety of the operator becomes of concern. Also, strong tidal streams are detrimental as the resulting flow past the plate prevents it from being directed at all azimuths.

Stationarity – when the composition of the sediments in the area is demonstrated to be stationary over time, with no discharge of sediments, no presence of turbidity currents, erosion or other geological variations, then such area can be used as a reference area for calibrating any sonar with the same absolute reference curve established with calibrated SBES at any time until it is noticed there has been some variation on the seafloor. As part of this, human disturbances such as anchorage areas of trawled areas should be avoided.

As a practical procedure, a previous survey with an uncalibrated MBES would provide an ideal means of selecting the best area to perform this method of calibration.

The areas chosen in this thesis are in British Columbia, Canada as we had logistical support of the Canadian Hydrographic Service (CHS) who made one of their ships available for this research, the CSL Heron.

Each area was selected based on 1990-2005 vintage 300 kHz multibeam data (EM3000 or EM3002) collected, and kindly made available, by the CHS. This allowed the regional delineation of backscatter variability and gross seabed roughness. Areas of patchy seabeds, strong slopes or obvious rock outcrops, were avoided. Figure 11 illustrates a 4 x 2 km area around the sand site revealing the variation in bathymetric roughness and backscatter variability.

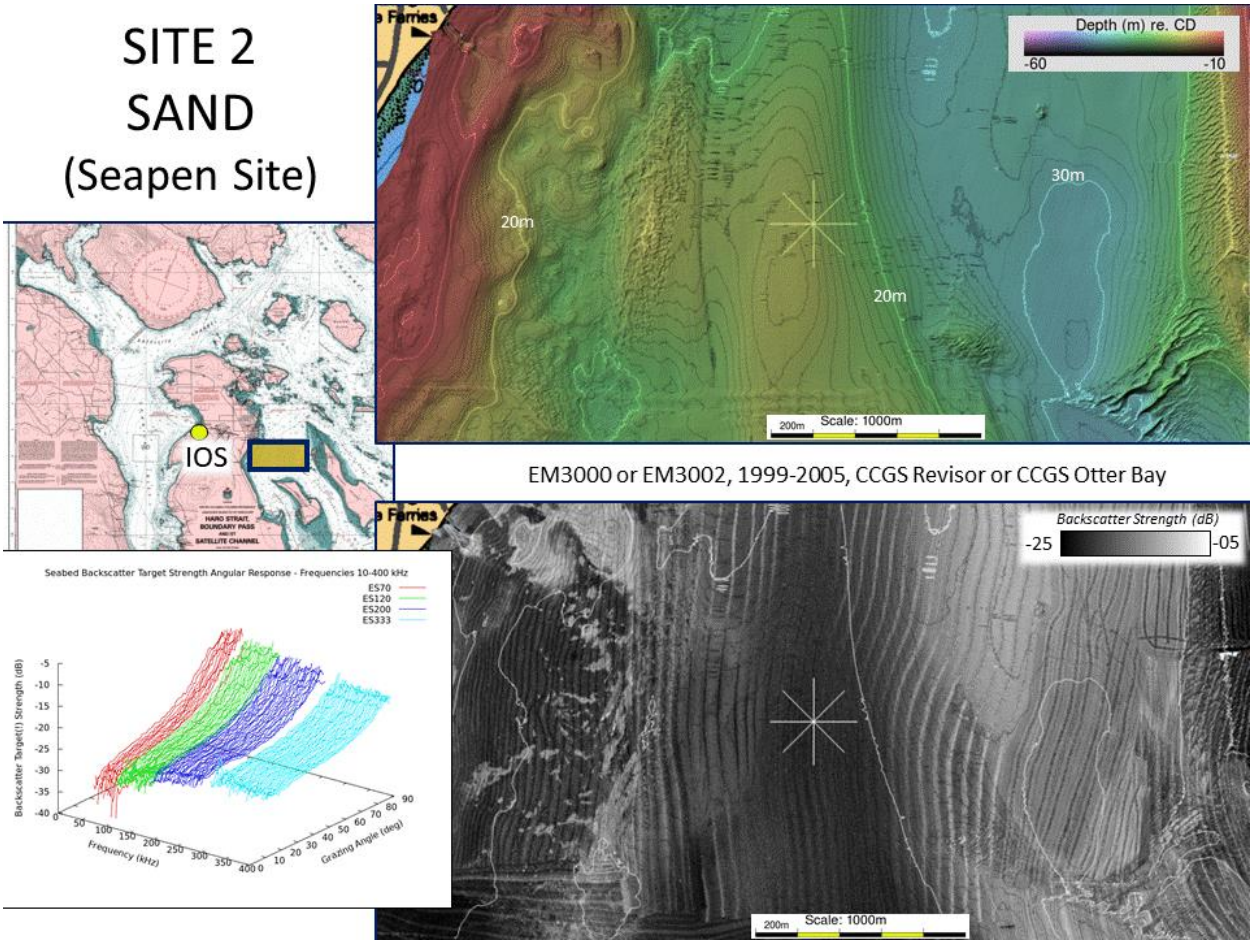


Figure 11 - Archived multibeam bathymetry and backscatter for the sand site (Sidney Approaches). Star pattern identifies the location of the test area. Data courtesy of the Canadian Hydrographic Service. Processed using swathed.

Based on the CHS data, five homogenous sites were identified with as wide a range of mean backscatter strength as possible. EM710 and EM2040 bathymetry and backscatter acquired over these sites at the time of the calibration data collection are presented in Figure 12. These indicate that the area within the EK80 data collection radius (indicated by the yellow circle) was completely homogenous.

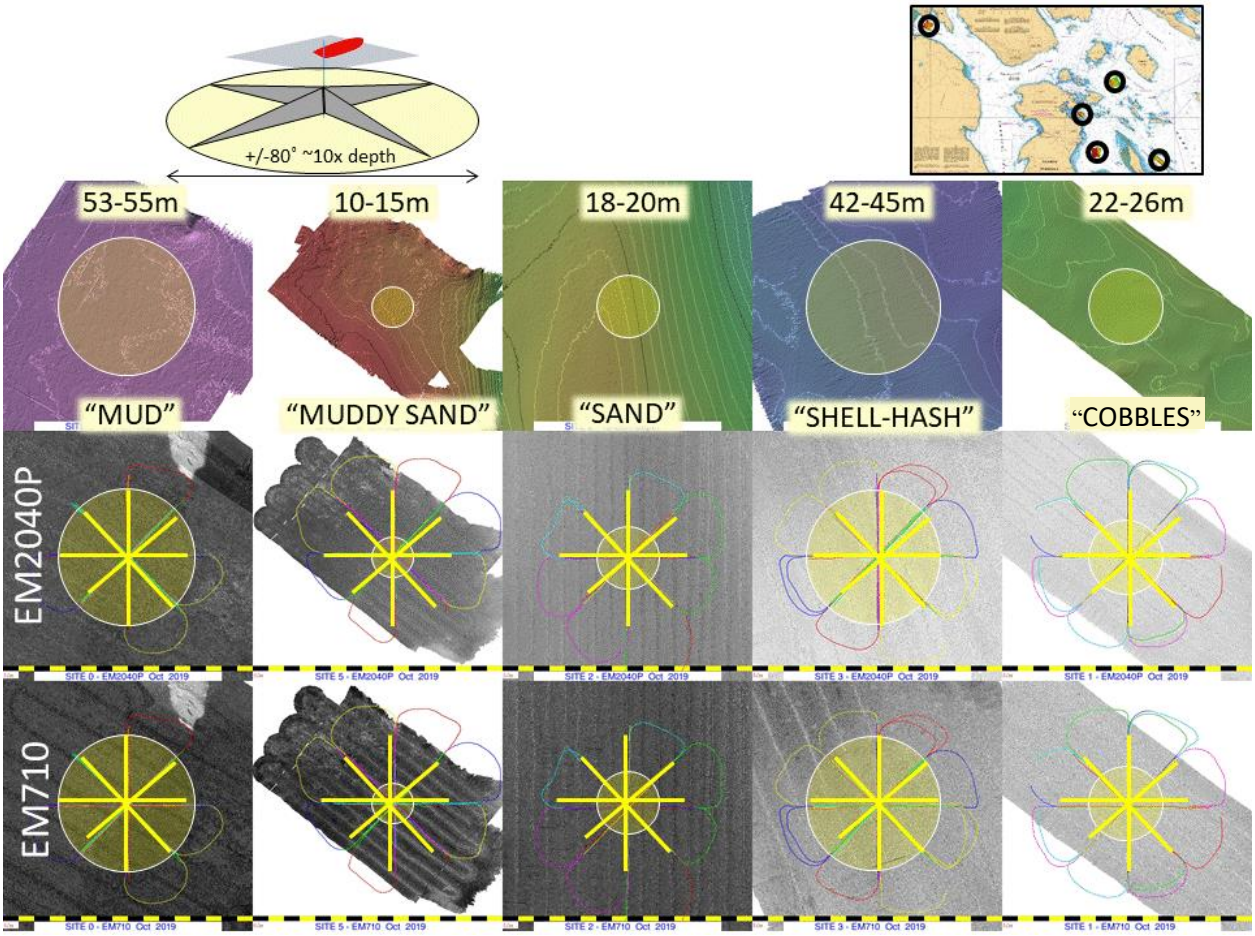


Figure 12 - Bathymetry and backscatter acquired using EM710 and EM2040 over the five calibration sites. The sites are ordered from lowest backscatter (left) to highest backscatter (right). The yellow circles indicate the radius of the area within which calibrated backscatter was collected. Processed using swathed

The Table 3 summarizes the characteristics of each area, and for the sediment type, the samples are pictured in Appendix I.

Table 3 - Characteristics of each area

Area	Collection date	Sediment type	Average depth	Average Sound speed	Temperature
A	09JUN2019	Mud	55m	1491 m/s	12ºC
B	10JUN2019	Shell hash	44m	1487 m/s	11ºC
C	12JUN2019	Muddy Sand	14m	1487 m/s	11ºC
D	08JUN2019	Sand	17m	1491 m/s	12ºC
E	10JUN2019	Cobbles	25m	1487 m/s	11ºC
Calibration	12JUN2019	xxx	xxx	1489 m/s	11ºC

All the areas are protected by the surrounding islands (see Figure 13 below), ensuring a smooth sea state. Additionally, the areas are out of the lanes of traffic and the multibeam bathymetry indicated that they were flat and sufficiently smooth, with minimal indication of different sediments within the same area.

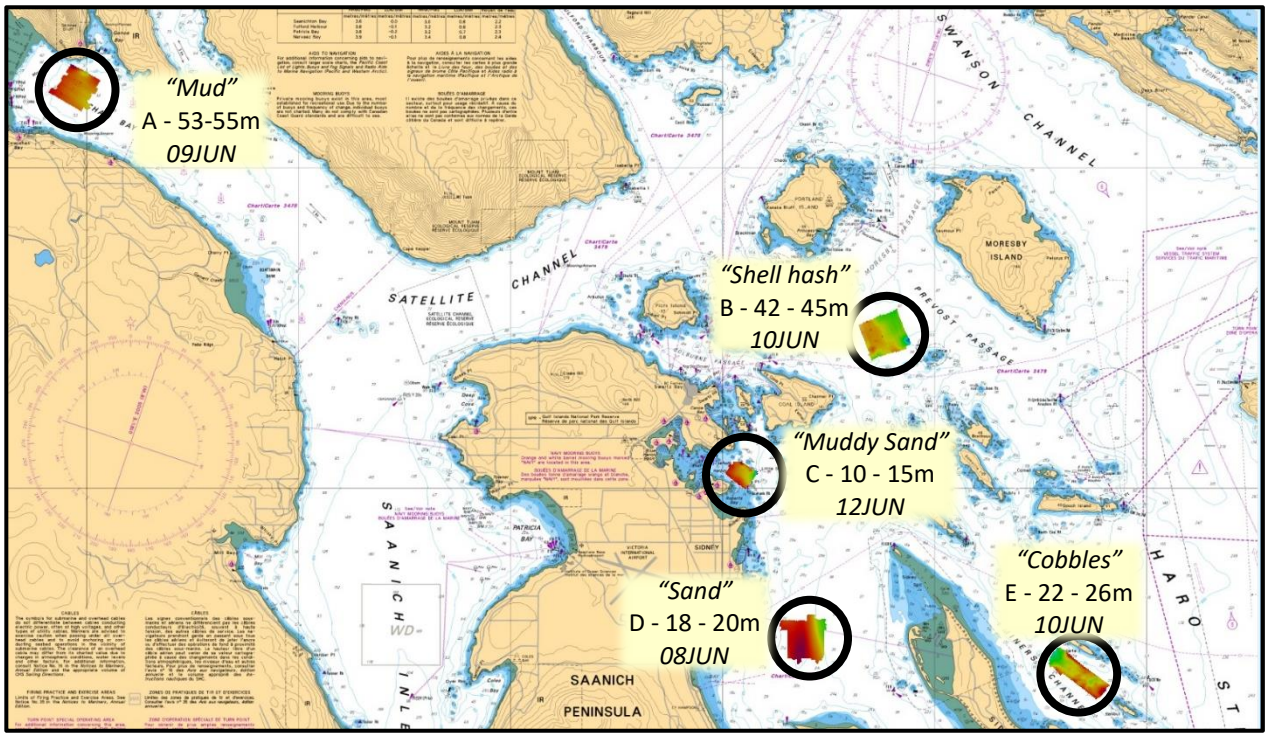


Figure 13 – The location of each area on the Nautical Chart CHS 3441 - Haro Strait Boundary Pass and Satellite Channel

Impact of water depth on site suitability:

Although not appreciated when the sites were first selected, it turned out that the water depth had an impact on the ability to collect low grazing data at the highest frequencies. Most notably, Area "A" had the softest sediment type, mud, and the greatest depth, 55m, which resulted in the received signal to noise ratio (SNR) being too weak to collect useful data from the transducer with the higher frequency (the ES333-7CD, centered in 333kHz) at grazing angles beyond 40 degrees. Although the depth of Area "B" was also relatively deep (~44m), the sediment type present in the area had a stronger backscatter strength, so the ES333-7CD was able to process the received signal even at lower grazing angles.

Figure 14 demonstrates the slant range variations with altitude and grazing angle. This reveals that for a flat seafloor, the slant range at grazing angle of 10^{o} can be only 50m for a plate elevated 10m from the bottom whereas it can be 300m if the plate is 50m above the seafloor.

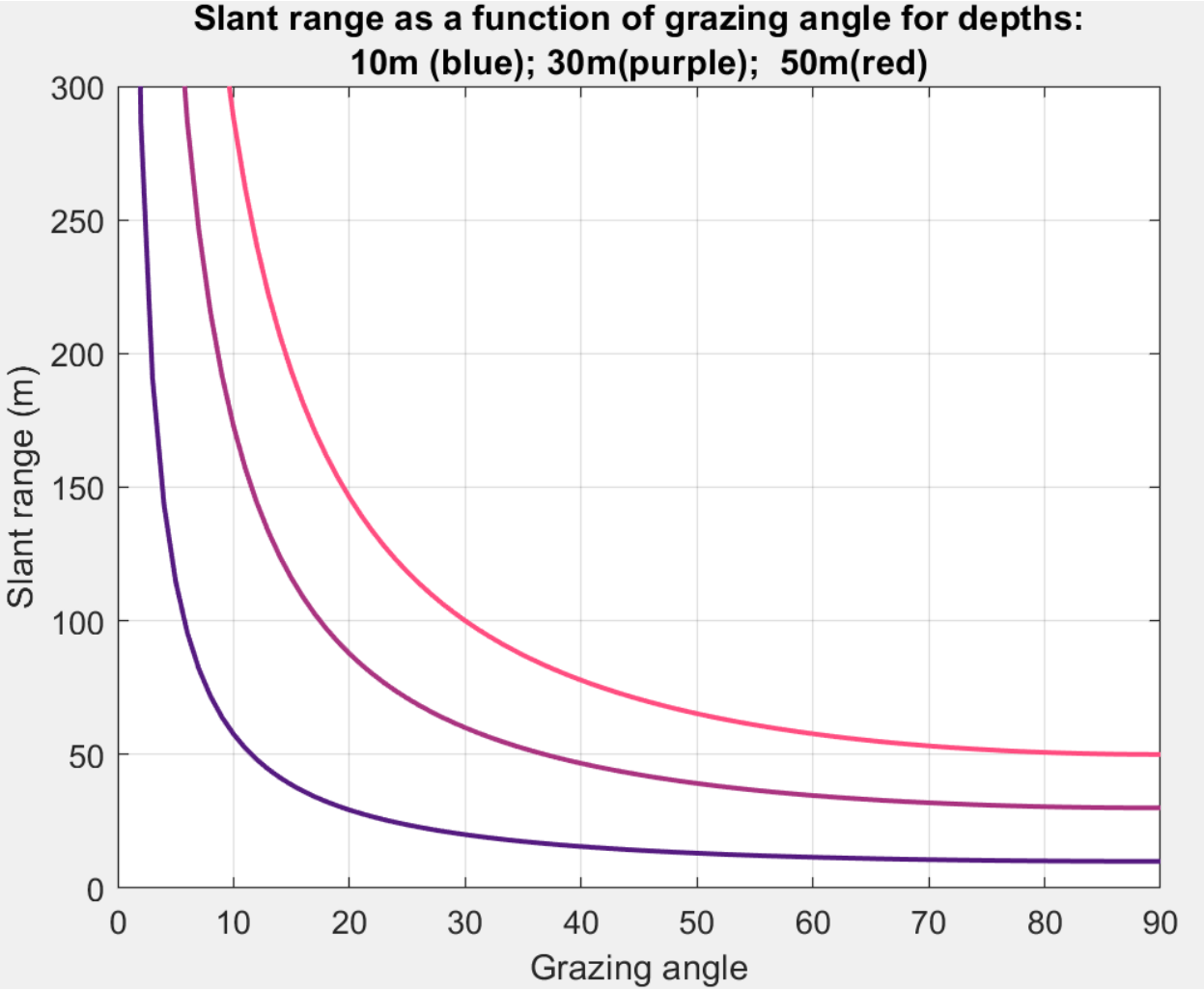


Figure 14 - Slant range as a function of grazing angle for a flat seafloor at different depths, 10m (blue), 30m (purple), and 50m (red).

Oceanographic Environmental Characterization:

To properly reduce the backscatter data, the transmission loss terms that are frequency dependent need to be precisely estimated. This is primarily controlled by the local oceanography. Fortunately, there were no significant variations in temperature nor in the harmonic sound speed during the few days of observation around the areas. For calculating the absorption losses shown in Figure 15, a pH of 8 and a salinity of 30 PSU were assumed.

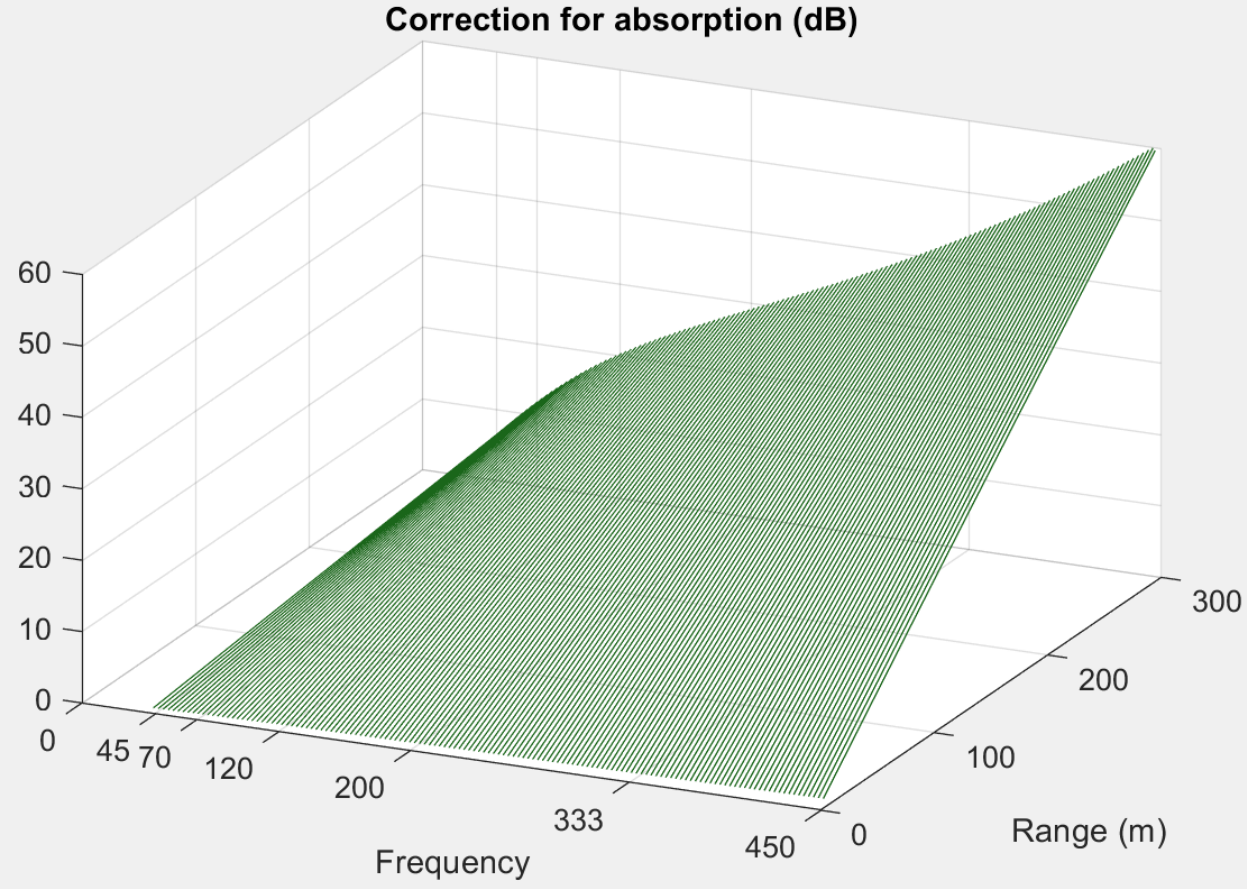


Figure 15 - Correction for absorption to be applied as a function of range and frequency, with environmental parameters of BC.

The data acquired from area "E" was collected during strong currents, tilting the plate significantly, which prevented collecting response from the nadir. As a result, its angular response curve has its grazing angle limits between 5 and 85 degrees. Also, as the plate acts as a kite when suspended in the tidal flow, it was not possible to align the beam at all azimuths. Rather only azimuths facing up or down stream were possible. This demonstrates the importance of selecting areas with reduced current conditions.

The site chosen for the SBES calibration was well protected from waves and currents and was deep enough to have the sphere deployed in the far field as well as being distant from the seafloor to avoid any confusion in target detection. The sediment type is irrelevant in this matter. The only other concern is that there should not be local scatterers in the water such as fish or algae.

3.4. CALIBRATION OF SBES WITH SPHERE MODEL

The approach used herein is the one most commonly used by the fisheries community utilizing the target sphere method to calibrate their sonars for the purpose of detecting and characterizing biomass in the water column. The sphere is a point target whose dimension is small with respect to the projected solid angle of the beam. As part of the calibration, the apparent target scattering strength (TS) can be estimated. To do this, in addition to the standard transmission loss calculations, the only correction required is to estimate where, within the two-way beam pattern, the target lies. The resulting TS estimate can then be compared with the expected model and the difference being the required calibration (the combined source level and receiver sensitivity product). The additional step beyond the fisheries approach that is used in this thesis is that, having achieved the calibration, we are now considering the ensonification of an extended target (the seafloor surface). This thus implies that the dimensions of the ensonified area must be considered.

The reference sphere is usually made of tungsten carbide or copper, with known parameters such as density, compression and shear wave speed and diameter. As noted by the National Oceanic and Atmospheric Administration (NOAA), the environmental conditions are important to determine the impedance contrast (NOAA, 2020); therefore, the temperature and salinity (and thereby sound speed) of the water are considered to calculate the expected TS according to the frequency. The TS values are calculated using equations in MacLennan (1981). The properties of the WC and Cu spheres are from MacLennan and Dunn (1984) and Foote et al. (1981), respectively.

The sound wave reaches the sphere and is partially scattered omnidirectionally, partially revolves around the sphere as the contributions of compression and shear waves. Depending on the diameter of the sphere, the wave that goes around the sphere can be in opposite phase with the reflected wave, therefore, certain frequencies will get none or weak response due to destructive interference (see minima in Figure 16), that are referred to as “nulls.” Close to the location in frequency of these nulls, no calibration is possible and thus interpolation is required.

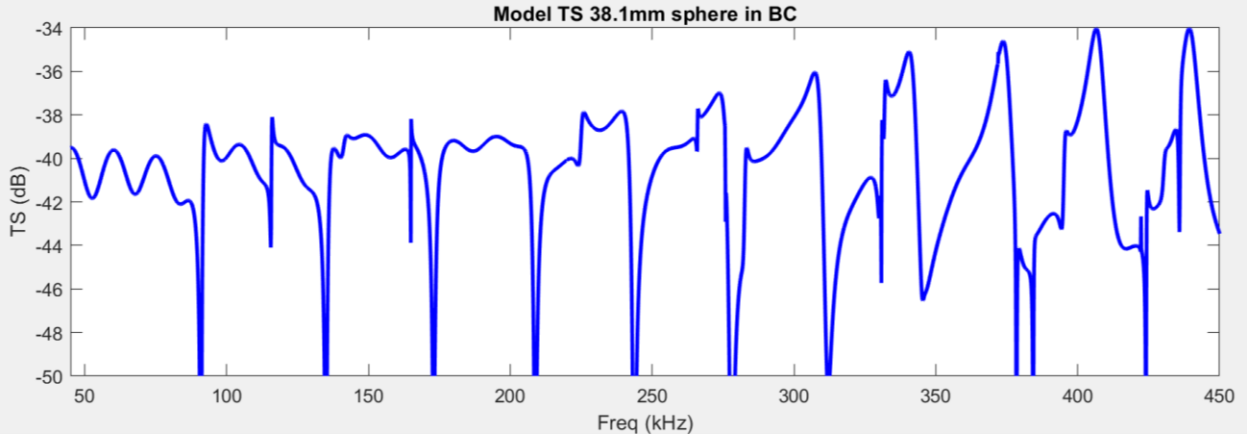


Figure 16 – TS model of a WC 38.1m sphere with the environmental parameters found in BC

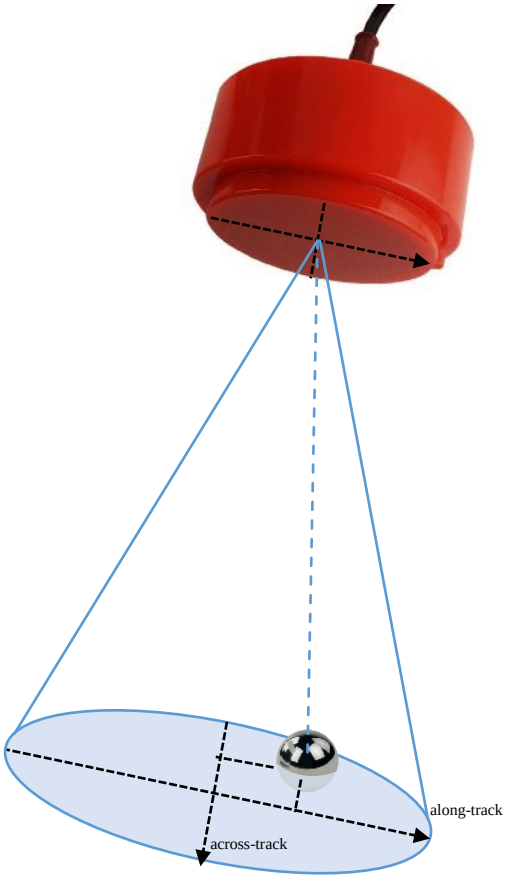
In this thesis, a tungsten carbide (WC) sphere with 38.1 mm of diameter was used. Due to irregularities on the construction of the sphere, when comparing to the model, the actual location in frequency of the nulls may be offset from the predicted model position. Thus, the immediately surrounding frequencies close to the expected null position are not considered to avoid errors and the adjacent values are interpolated. To fill the nulls within the bandwidth of one sphere, another one with a different diameter could be used. For this experiment, however, only one sphere was used, and the nulls were replaced by interpolation and the transducer calibration was expected to vary smoothly across the bandwidth.

An additional problem with the choice of 38.1mm sphere for the 333 kHz transducer was that, between the nulls, at the higher end of the frequency range, the TS varies quite steeply with frequency (see Figure 16). Thus, if the actual sphere response is slightly shifted due to imperfectly assumed properties or dimension, biases may result in the calibration. At the lower end (40-~250 kHz) the inter-null TS variability was much flatter.

The net result is that the quality of the calibrations obtained for the ES70, 120 and 200 transducers appears much more robust than the ES333 (see section 4.1).

Locating reference target within beam pattern

An important characteristic of the split beam echosounders is that the transducer has adjacent sectors that can be used to calculate the angle of the target with respect to the boresight based on the difference in phase between the echo at each of the sectors. The sketch in Figure 17 presents the transducer split into four sectors, represented by dashed lines, which enable the localization of a target in terms of along-track and across-track angles within the beam.



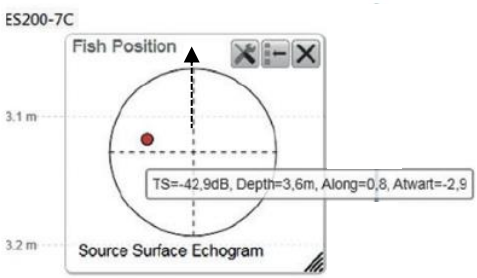


Figure 17 - Sketch of the split beam sectors.

It is important to have multiple soundings on the sphere in different positions in all quadrants within the beam, near the MRA as well as by the edge of detection. The variation of the apparent TS response reveals the two-way beam pattern of that transducer and if the transducer had been perfectly constructed, the beam pattern would show an axial symmetry and the MRA would be located at the maximum response axis (MRA).

The sphere calibration should be performed for one transducer at a time. The response obtained from the sphere by the transducer must be corrected to account for the two-way transmission losses that results from spherical spreading and absorption. The spherical spreading loss is a simple geometric function of range whereas the absorption loss is not only a function of range but also is dependent on many environmental properties of the sea water, such as temperature, salinity, pressure, pH and, particularly important in this thesis that is using FM pulses, it is dependent on frequency. Therefore, the spherical spreading loss is applied in the time domain and the absorption one is applied in the frequency domain.

3.5. FREQUENCY DOMAIN PARTICULARITIES

The ultimate aim of this project is to obtain a measure of the backscatter strength over the full available bandwidth of the transducers. For efficiency, rather than using successive narrowband CW pulses each progressively stepping through the frequency range, one ping at a time, FM pulses were used of 2.048ms of duration. Each transducer's FM pulse swept linearly through the full available bandwidth. The pulses were chosen to be closer to a rectangular shape ("fast ramp") rather than tapered. This was so that the full power was maintained at the ends of the available bandwidth. The price for using rectangular pulses is that there are stronger time sidelobes in the autocorrelation of the pulse. This would matter if range resolution was the main requirement. But for the case of this experiment, it is the frequency content that is paramount. This is also important to know in selecting an appropriate window around the main target echo as these sidelobes contain valid energy both before and after the main target echo.

The original received signal (Rx) for a point target has an elongated envelope the same length as the pulse length (~2ms) and when correlated with the transmitted pulse (Tx), generates a narrow Matched-Filtered (MF) result (see Figure 18), the width of the peak of which reflects the bandwidth.

Each signal Rx, Tx or MF contains all the frequency information within it. The Fourier-Transform is the means of examining it. In the case of the signal that is digitally sampled, therefore discrete in time, the Fast-Fourier-Transform (FFT) is then applied.

Because the matched filtered echo will have preceding and trailing sidelobes, as part of the pulse autocorrelation function, a finite window before and after the target needs to be included in estimating the TS. Furthermore, because the diametrically re-radiating echoes occur at times after the leading edge of the echo, to calculate the full spectrum of the target strength, the window must be extended several decimeters beyond the position of the first target echo.

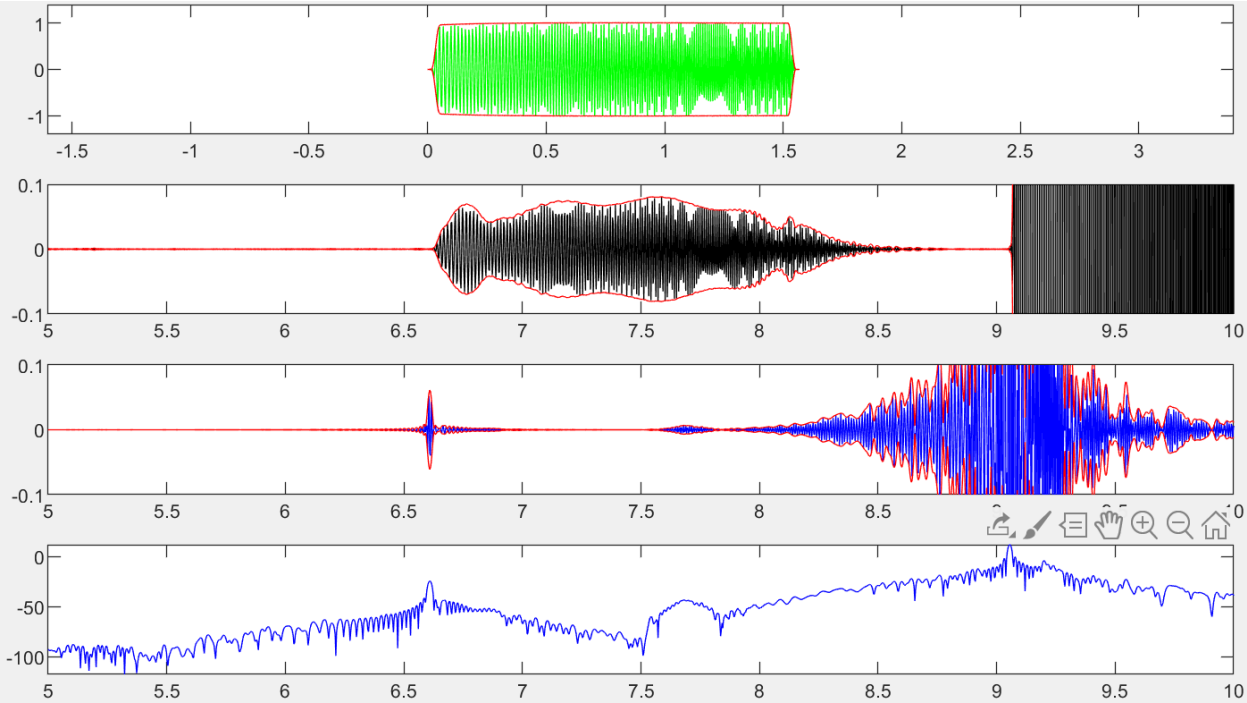


Figure 18 - From top to bottom: Transmitted signal (Tx); Received signal (Rx) with a sphere as first target and the seafloor as the second and strongest target; MF signal (convolution of the Tx and Rx); and level in dB of the MF signal. X-axes are in meters. The signals are represented by its real part with a red line evolving the echo envelope based on the absolute value of each signal.

The length of the extract of the signal used to apply the FFT is referred to here as the "subset" and was a significant matter of consideration in this thesis as it has implications in the resolution in terms of frequency. The larger the subset, the greater is the frequency resolution (see Figure 19). The choice impacts the results at both the calibration and the bottom echo analysis stage.

For the calibration, the length has to be at least as long as all the echo contributions (trailing circumferential waves and sidelobes in pulse autocorrelation), but not too long to include other spurious contributions such as knots in the line or natural suspended targets. For the calibrations utilized, a window 30 cm before and 30cm after the peak of the target was used, corresponding to approximately 200 samples. At a 250 kHz sampling rate, this corresponds to a frequency resolution of 1.25 kHz.

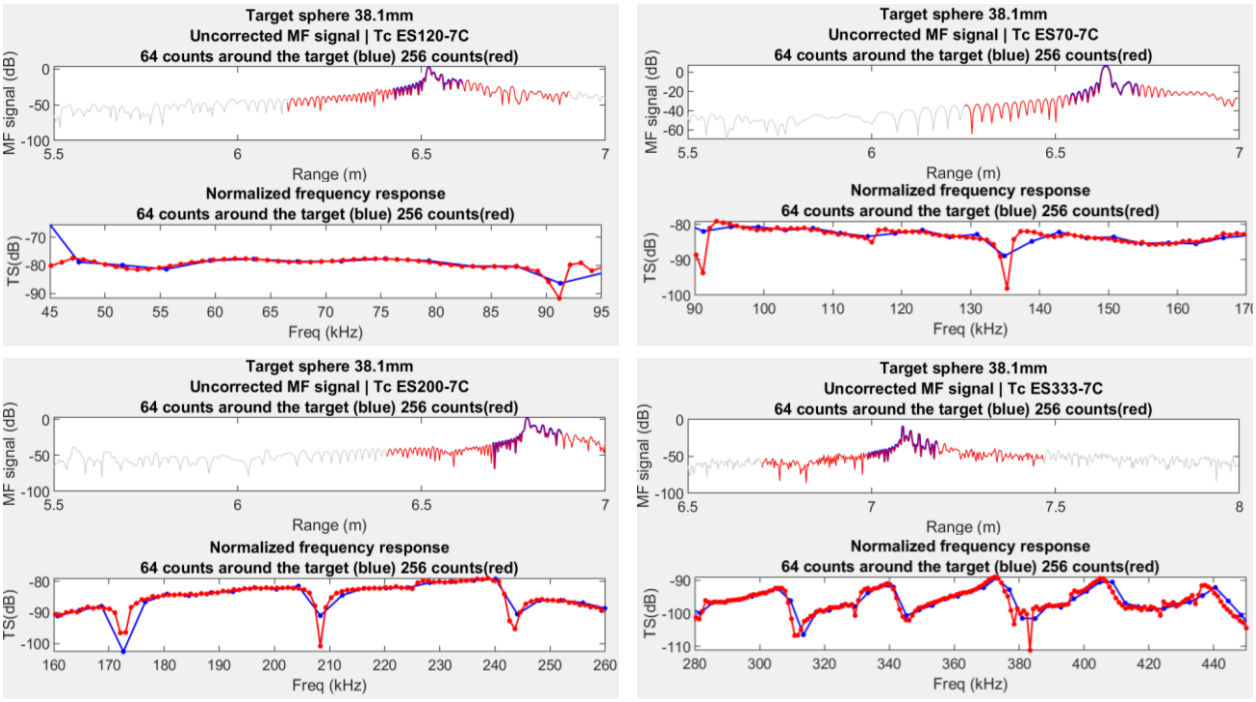


Figure 19 - Comparison of FFT of different sample sizes, 64 (blue) and 256 (red) for a 38.1mm sphere detection for each transceiver.

For the seabed echo, the dominant factor affecting the choice of subset length is primarily a geometric one. The length impacts the results in two ways: for high grazing angles it controls our ability to discriminate in angle, and at lower grazing angles it affects how we compensate for the across track beam pattern projection (see section 3.8).

Whilst the time sampling defines the Nyquist frequency and is inherited by the transceiver, the selected size of the subset sample will define the resolution in the frequency domain. As will be discussed further later, the chosen seabed subset length was 64 samples (19cm) which, with a 250 kHz sampling rate resulted in a 3.9 kHz spectral resolution. Note that this is significantly larger than the theoretical range resolution which is controlled by the FM pulse bandwidth (ranging from 50 to 200 kHz or 1.5 to 0.4 cm). As we do not expect to see rapid changes in the bottom backscatter strength over a few kHz (as the seabed consists of a wide range of scattering scales unlike the solitary reference target), this was considered adequate. The resulting seabed backscatter frequency trends (see section 4.3) bear this assumption out.

For the work done herein, the length of calibration FFT, which controls the frequency resolution of the calibration, was not the same as the length chosen for the seabed FFT. This results in complications in applying the finer calibration coefficients to the coarser seabed spectra. This will be discussed later in section 3.8.

For the case of a signal that is a continuous wave, the length of the subset is indifferent, providing the same result, as long as it has the minimal resolution to define the signal. For the case of the reference target echo, however, where there is a peak with only noise before and after the peak, the longer the length of the subset, the more noise is incorporated on the signal to calculate the FFT, which yields a compromised response. A proper normalization of the signal is then needed to compensate this factor.

The way SIMRAD, which produces the transducers used in this project, normalize the FFT of the MF signal is presented below (Demer et al., 2015):

$$U_{target}(f) = \frac{U_r(f)}{U_t(f)} \qquad (5)$$

The frequency response of the target ($U_{target}(f)$) is the frequency response of the subset that includes the target ($U_r(f)$), normalized by the frequency response of the autocorrelated transmitted signal with the same length as the subset ($U_t(f)$). The autocorrelated transmitted signal has twice the length of the transmitted pulse. So, the maximum possible resolution in frequency is the correspondent resolution achieved with an FFT of twice the length of the transmitted pulse.

As will be seen in section 4.1, the beam pattern also has a frequency dependence; therefore, a unique beam-pattern correction, based on the observed location of the target in the beam should be applied to each point in the FFT.

3.6. SPHERE CALIBRATION SUMMARY

The sequence to process the data for the sphere calibration was:

1. Convolve Rx with the Source Function to arrive at the MF.
2. Add the geometric spreading component of the transmission loss.
3. Apply the FFT.
4. Normalize by the FFT of the autocorrelated Tx.
5. Add the transmission loss due to absorption as a function of frequency.
6. Locate the sphere in terms of angles within the beam by using the across and along-track phases.
7. Using multiple pings, with enough data to cover the whole beam-pattern, calculate the pattern as a function of frequency.
8. Obtain and correct all pings for the beam pattern as a function of frequency, based on the across and along track angles, so all the pings would correspond to an equivalent MRA response.
9. Calculate the average in a linear scale, not in a logarithmic scale.
10. Compare the result with the model disregarding the neighborhood around the nulls (identified as areas with slopes greater than 0.1 dB/kHz.
11. Interpolate the response of the frequencies surrounding the nulls (done linearly).
12. The comparison is the calibration that should be applied to the transducers together with correction for the beam pattern according to the angle of the response off the boresight.

3.7. LOGISTICS OF BOTTOM TS DATA COLLECTION

This section describes the logistics of acquiring Sb data from an anchored vessel with the suspended plate. The ship should be steady at a position at the center of the selected area and avoid drifting to collect data only within the area with the same type of sediment. Anchoring is recommended.

The plate containing the transducers of the SBES and the motion sensor unit is submerged a few meters to allow controlled rotation and prevent surface interference. As mentioned, moments of strong current should be avoided when possible.

Since the transducers overlap in frequency, it is necessary to ping one at a time to avoid interference. As the maximum slant range was < 250m a 0.33s ping repetition rate was used. The plate should be steady or rotating slowly enough during the ping, so the angle of transmission and reception are nearly the same. Therefore, the rotation rate should be as slow as the range demands to collect each sequence of pings, bearing in mind that the range can be about ten times the altitude with grazing angle as low as approximately six degrees. Considering this, the rotation at lower grazing angles should be less than 1 degree of rotation within the ping duration, so, less than 3 degrees per second.

The plate is gradually rotated up to about 85 degrees of roll, with no intentional variation in pitch, and it is then rotated back through nadir and up to -85 degrees in the reverse azimuth. The rotation is repeated several times at different headings to account for the azimuthal dependence.

In reality, there will always be some pitch rotation. What this implies is that the beam elevation vector is not adequately described just by the roll. For large rolls, a few degrees of pitch do not significantly impact the grazing angle. But as the roll decreases (and the grazing angle increases) the sensitivity increases.

Figure 20 below demonstrates the impact of the pitch in the grazing angle with the roll. On top, the grazing angle as a function of roll for pitch=0 º, 5 º, 10 º, 15 º and 20º. On bottom, the difference between the grazing angle and the complementary angle of the roll, for a pitch = 0º, the grazing angle and the complementary roll are equivalent. The dashed red line indicates when the difference exceeds 1 degree. For a pitch = 10º, for example, the difference becomes less than 1 degree for roll greater than 50º, whilst for a pitch = 5º, the difference becomes less than 1 degree for roll greater than 12º.

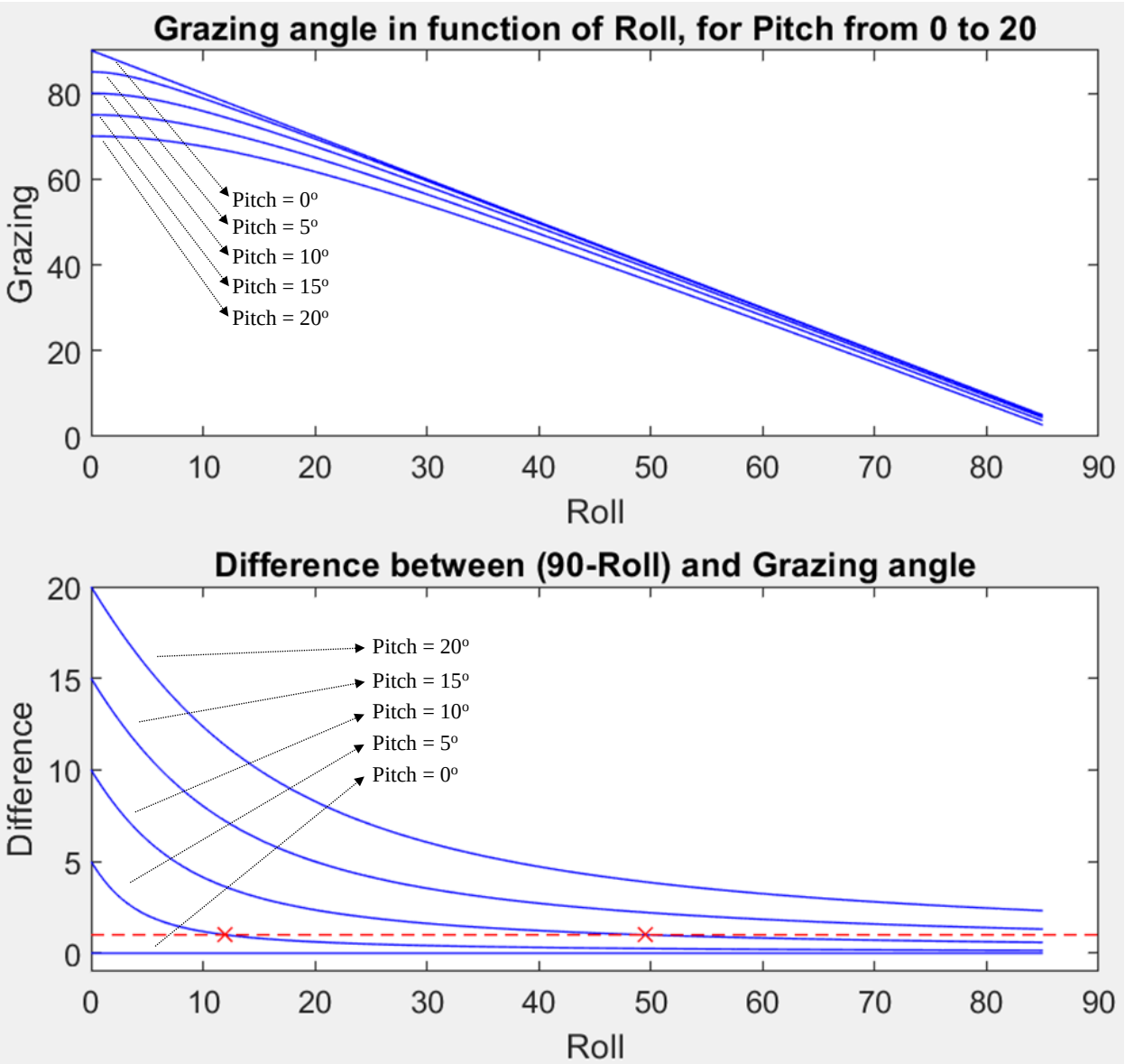


Figure 20 - Impact of Pitch in the grazing angle.

3.8. BOTTOM TS DATA PROCESSING

The data is processed following the steps detailed below:

Echo Envelope Identification and MRA Strike-Point Location

STEP 1 - Amplitude Detection:

After the matched filtering, for every ping, the first step is to locate the two-way-travel-time (TWTT) of the maximum response, considering just the spreading component of the transmission loss (see Figure 21).

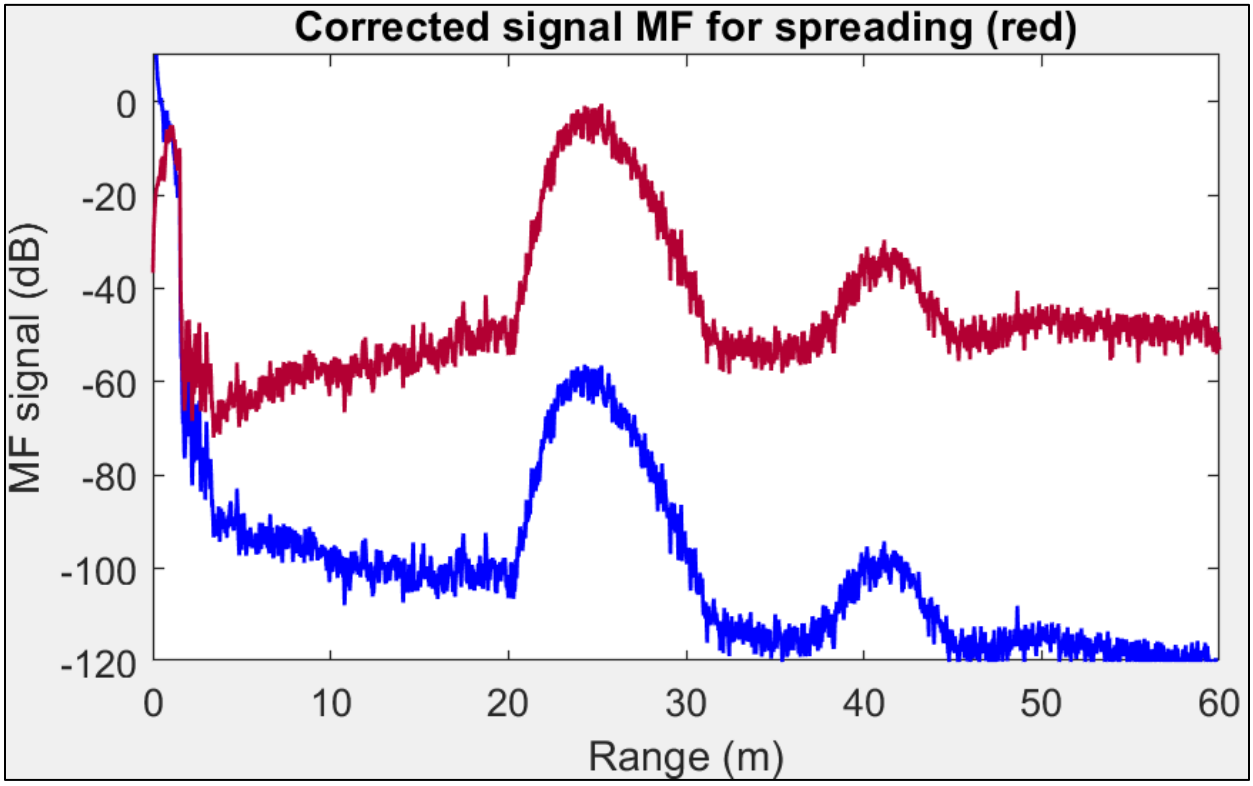


Figure 21 - Raw MF signal in blue in comparison with the signal compensated by spreading (40log(r)) in red.

The absorption loss was not considered to find the TWTT because the signal-to-noise ratio (SNR) would be affected, mainly at higher frequencies and longer ranges. A more robust code which uses the average absorption coefficient (as a broad band pulse) should prevent this from happening and use all the corrections to find the TWTT. However, it will only have a minor difference on the final result, since this is only a preliminary estimate of the approximate center of the echo envelope.

The Figure 22 reveals the peak response, where the TWTT was found for that ping.

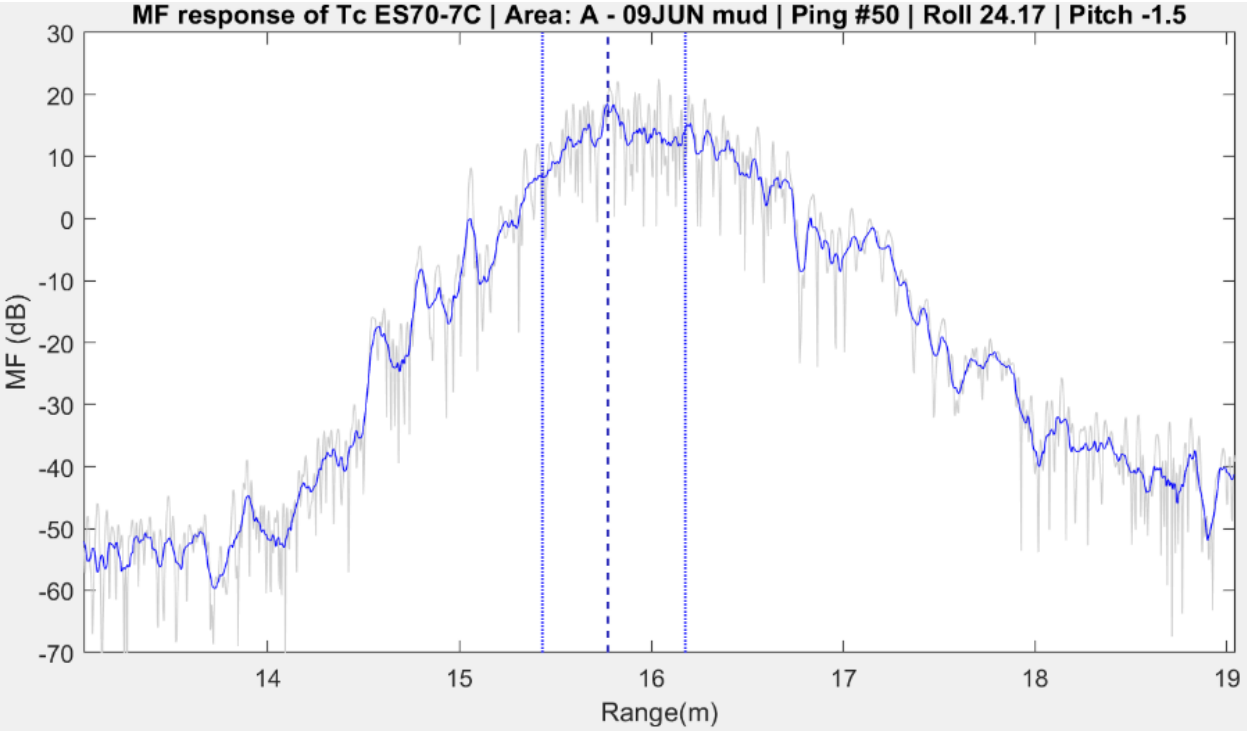


Figure 22 - Finding the Maximum response: MF signal with TL (TS) in grey, smoothed TS in blue line, Maximum TS in vertical dashed dark blue, boundaries in vertical dotted blue.

Based on the TWTT of the amplitude detection, the data is subset by windowing according to the expected length of the echo accounting for the elevation angle measured by the MRU-6 and the nadir depth. On this step, the data was narrowed using ±3 degrees of elevation from the estimated TWTT point. This reflects the average two-way 3dB widths of ±2.5 degrees. The idea being to utilize just the echo envelope within the main lobe.

$$R_i = R_t \sin(Gr) / \sin(Gr + 3^o) \qquad (6)$$

$$R_f = R_t \sin(Gr) / \sin(Gr - 3^o) \qquad (7)$$

Where:

R_i – Initial range of the extract to be observed

R_t – Range of the target by amplitude detection

R_f – Final range of the extract to be observed

Gr – Grazing angle

This allows a better search for the zero-crossing on the next step.

As part of the following step, a series of subset of points will be analyzed rather than only one point and the error will be within the range of the subset.

STEP 2: attempted Phase detection (and limitations)

This step will search for the zero-crossing on the across-track phase ramp, constrained within the range delimited by the previous step. When it cannot identify a clear zero crossing in the phase, there will be only amplitude detection. This could occur if the number of phase samples within the range window is too few or a regression through that phase data shows too high a residual. In practice, amplitude detections were forced within ~ 4 degrees of normal incidence.

As the phase data is inherently noisy, vector averaging of the phase differences was conducted before seeking the zero crossing (Figure 23). The impact of this phase smoothing on subsequent estimation of the grazing angle is discussed later.

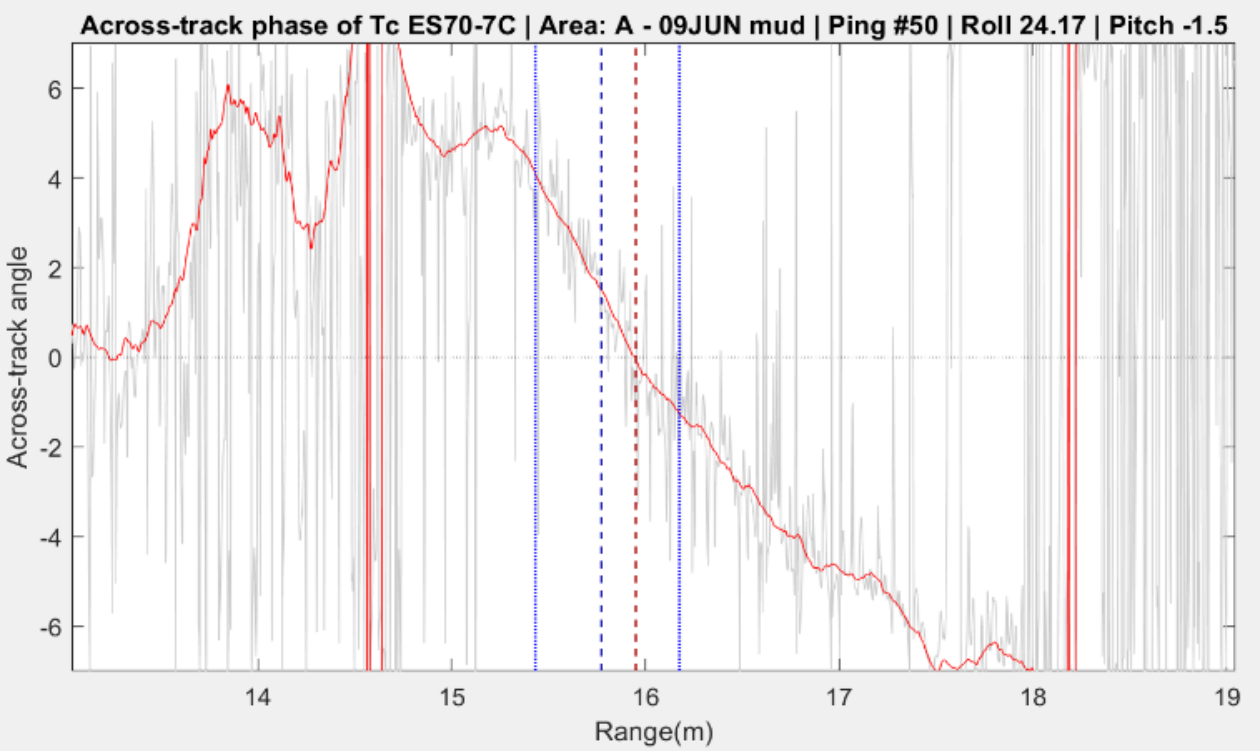


Figure 23 - Searching for the zero crossing within the boundaries defined by the previous step: Across-track angle (phase-across) in grey, smoothed phase-across in red line, Maximum TS in vertical dashed dark blue, boundaries in vertical dotted blue and Zero-crossing in vertical dashed dark red.

The across-track phase signal is obtained by the difference of phase between the two sectors on the port and the two sectors on the starboard. The slope of the phase difference indicates the sonar-relative across-track seafloor slope which, for a low relief seafloor, will be mainly due to the roll of the plate. The electric phase is then converted into physical angle according to the manufacturer's indication of sensitivity for each transducer. The corresponding along-track phase is obtained from the difference between two sectors facing forward and the two sectors facing aft. For large values of pitch and small values of roll, the slope of this phase difference will indicate the pitch displacement.

For the typical situation in this experiment, however, while the roll varies through ±85 degrees, the pitch is typically small (< 10 degrees). Under these circumstances, the geometry of the annulus constrained within the beam footprint needs to be considered as under large roll and small pitch geometries, that annulus extends across the full fore-aft (along-track) dimension of the footprint. As other researchers have noted, for such an extended target, as the annulus will include parts of the surface at all the different along track angles, their complex scattered sum cannot be used to indicate any one along-track seafloor slope.

As a result, only the across-track seafloor slope can be discerned and thus the grazing angle estimate is imperfect. This aspect is expanded more in the grazing angle estimation section.

Assuming that an improved TWTT estimate is obtained using phase, the window for analysis is now shifted and slightly expanded. Based on the same concept of narrowing the sample of the time series of each ping used in the previous step (equations 7 and 8), the data analysis window is defined by applying a new constraint within ±5 degrees of elevation from the zero-crossing point.

$$R_i = R_p \sin(Gr) / \sin(Gr + 5^o) \qquad (8)$$

$$R_f = R_p \sin(Gr) / \sin(Gr - 5^o) \qquad (9)$$

Where:

R_i – Initial range of the extract to be observed

R_p – Range of the target by phase detection

R_f – Final range of the extract to be observed

Gr – Grazing angle

The new boundaries defined by the phase ramp (see Figure 24 below), when a zero-crossing was found, will define the limits of the phase ramp to be used on the next steps.

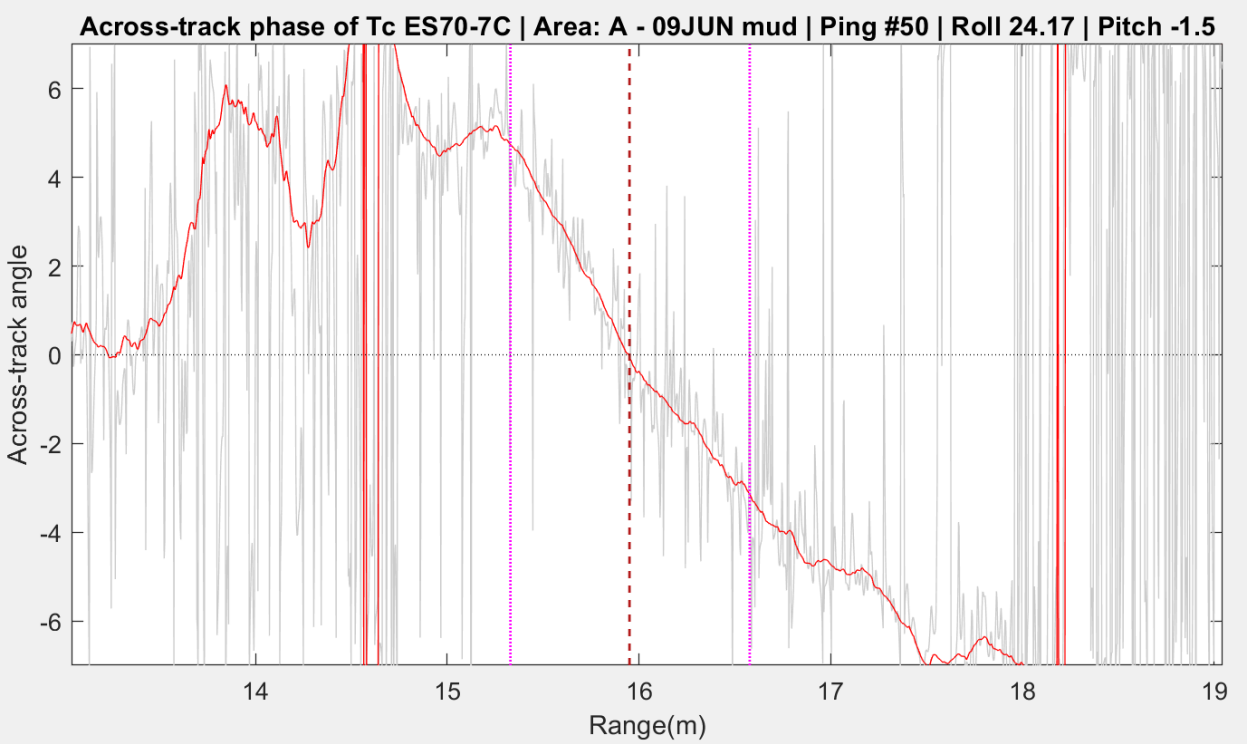


Figure 24 - Redefining the range of selected data: Across-track angle (phase-across) in grey, smoothed phase-across in red line, Zero-crossing in vertical dashed dark red, new boundaries in vertical dotted magenta.

Considerations about the selection of subsets within the echo envelope

Once the echo envelope has been identified, previous studies (Weber and Ward, 2015 and Eleftherakis et al., 2018) had both restricted their analysis to just the peak of the envelope, providing a single bottom backscatter strength for that ping cycle. Given, however, that such instantaneous estimates can be overprinted by speckle, it was considered advantageous to obtain multiple estimates to get a more stable average. Without this step, getting sufficient samples requires covering a larger area (for an underway vessel as was the case for Weber and Ward, 2015 and Eleftherakis et al., 2018) or more transects over the same area.

In this thesis, the decision was made to use multiple points within the echo envelope so that more independent estimates of the bottom backscatter strength (each at slightly different angles) could be obtained from each ping cycle. To do so, however, requires compensation for the across track beam pattern.

Also, the number of independent samples (herein termed subsets) will depend on the choice of subset length relative to the echo envelope length. The envelope length of course, varies with both sonar altitude and grazing angle.

In this project the time sampling interval of all four transducers were the same, 4μs, which corresponds to 250 kHz of bandwidth in the frequency domain. The number of samples selected in the time domain will be the same in the frequency transformation, equally distributed within the bandwidth. The sample size can be set as convenient, the more samples, the better resolution in frequency. However, the longer the sample, the worse the resolution in grazing angle.

Figure 26 shows a MF signal being extracted and the corresponding FFT with large and small subsets to illustrate the difference in frequency resolution between using a short or long subset length. As can be seen, while the longer FFT gives better frequency resolution, the echo envelope variation due to the across-track beam pattern becomes significant over the subset length. Thus, there is an additional complication with long subset lengths in that the beam pattern contribution is changing throughout the FFT window, complicating the beam pattern compensation.

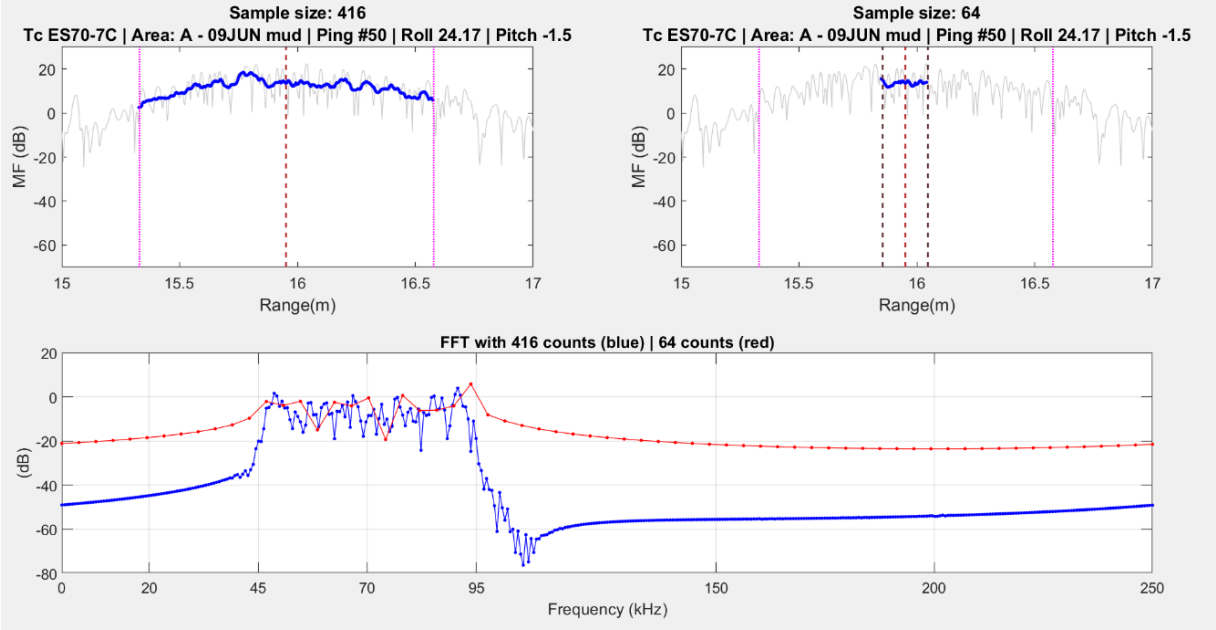


Figure 25 - The two plots on the top show the same extract of the MF signal with different sample sizes and the bottom plot shows the FFT response of the seafloor with the correspondent sample sizes. The zero-crossing occurrence is shown in the dashed dark red line.

Based on inspection of the length of typical high-grazing angle echo envelopes which were seen to be as short as a few decimeters, a subset length of 19 cm (64 samples) was chosen. If the echo envelope at the 3 dB points was longer than this, more than one subset would be used. And to get more estimates, each subset would overlap its neighbors by half of the subset length. For the lower grazing angle footprints which extend several meters, this resulted in ~60 subsets for the lowest grazing angles (see Figure 26).

For a better computing performance and to avoid being out of memory on MATLAB, the length of the MF signal that was extracted was limited to 2001 samples (~6m in range) around the peak of the MF, which implied in a limitation of 61 subsets of 64 samples. For the pings with a high grazing angle, the echo envelope would require fewer subsets, but for lower grazing angles and in deeper waters, the echo envelope would be greater than 6m, demanding more than what was provided herein. A further development on the codes would be necessary to allow more data in the analysis.

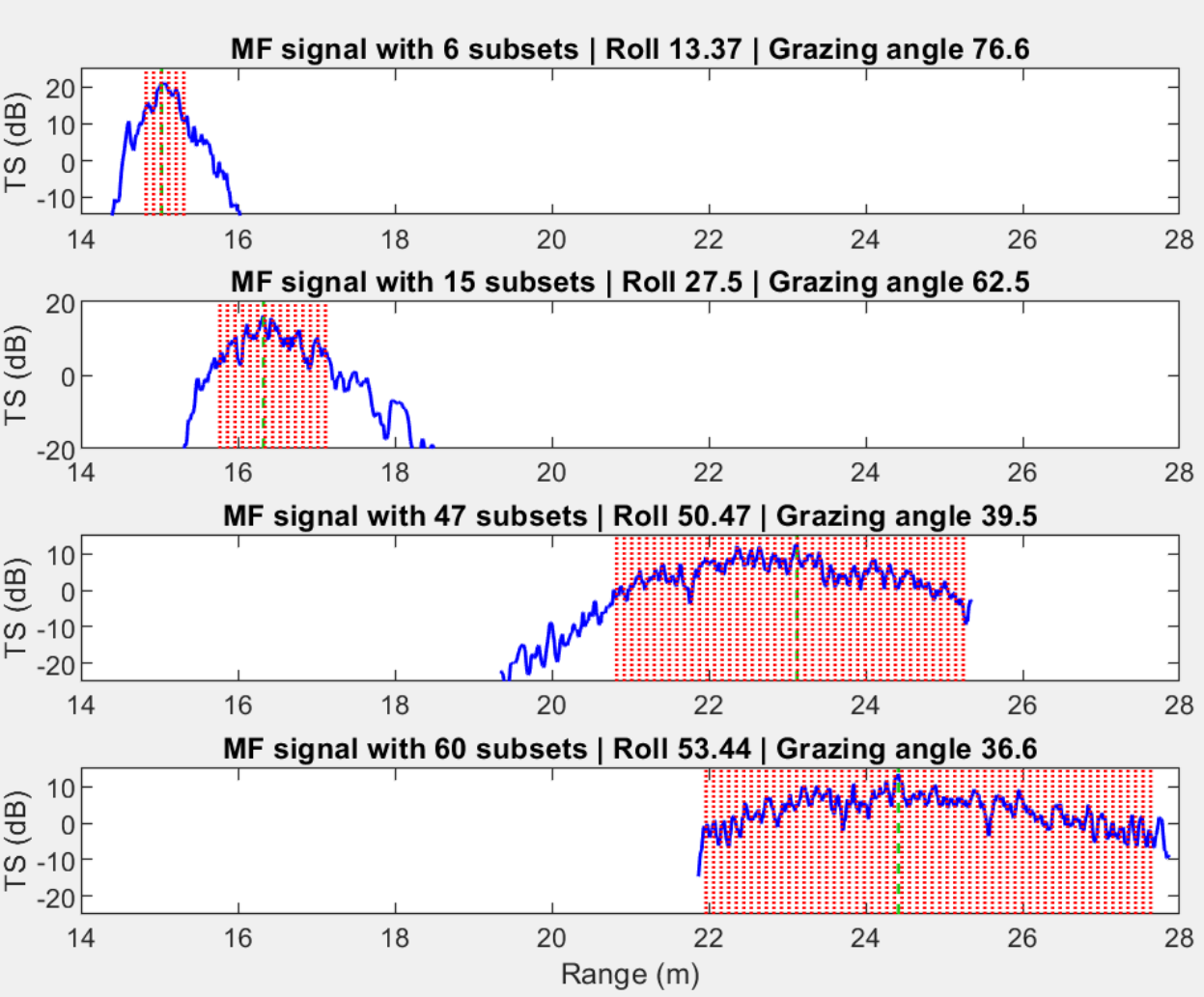


Figure 26 - Increasingly number of subsets as the grazing angle is lower.

The net result of this choice was that the frequency resolution was 250 kHz/64 = 3.9kHz/bin, which provided 12 bins between 45kHz and 95kHz for the TC ES70-7C, 20 bins between 90 kHz and 170 kHz for the TC ES120-7C, 25 bins between 160 kHz and 260 kHz for the TC ES200-7C and 43 bins between 280 kHz and 450 kHz for the TC ES333-7CD.

Grazing Angle Estimation

Ultimately, for each subset, a grazing angle needs to be associated. Within a single beam footprint, the ray path depression angle will be expected to vary by at least 5 degrees reflecting the swing from inboard to outboard edges of the beam pattern. In addition to the change in ray vector, the slope of the seafloor below will not be exactly zero.

To try to estimate that grazing angle, two strategies were implemented depending on how close to normal incidence the beam was.

Method 1 - Grazing angle from MRU-6

The Inertial Motion Unit MRU-6 provides Pitch and Roll, among other measurements, that can be utilized to calculate the approximate angle that the TC is pinging at with respect to the Nadir. Note that this assumes that the MRU is perfectly aligned with the boresight of each of the transducers. With this assumption, if the seafloor is horizontal and flat and the sound travels with no refraction, then the grazing angle (Gr) can be directly obtained by the MRU-6:

$$Gr = \sqrt{\theta^2 + \varphi^2} \qquad (10)$$

Being θ and φ the pitch and roll, respectively.

In reality, as shown by Figure 27, the alignment is not perfect, and the seafloor will likely present some relief. Additionally, although not critical at high grazing angles, there will be a ray tracing along the water column.

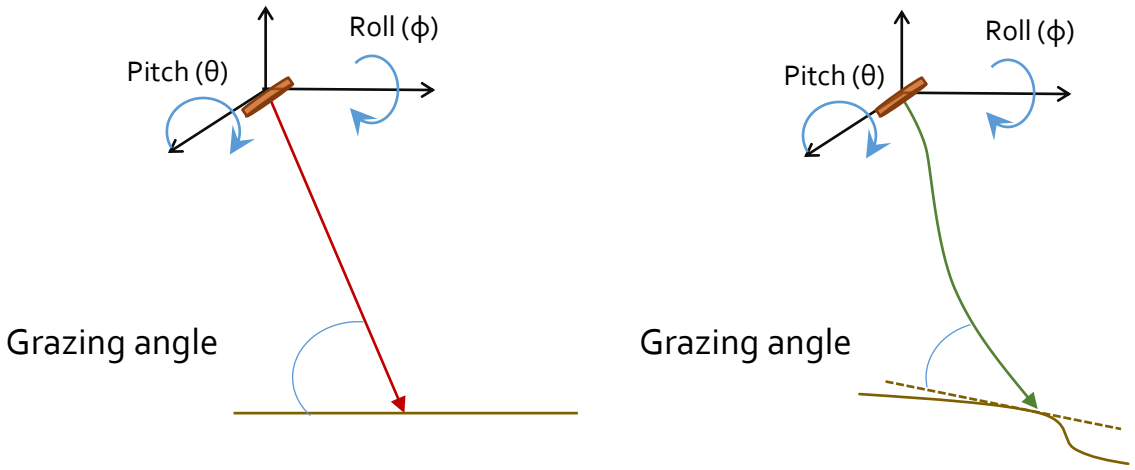


Figure 27 - On the left, an ideal propagation of the beam where the grazing angle can be directly calculated by the ship's attitude. On the left a sketch of a more realistic case that demands other technics to infer the actual grazing angle.

The advantage of using the MRU orientation directly is that both the pitch and roll can be combined to get a 3D grazing angle. This turns out to be significant as with the freely suspended configuration used, there was often pitch present. The result of this is that, if only the roll were used, one might erroneously assume that normal incidence had been achieved when the roll was zero. As, for some smooth sediments, the specular peak in the angular response curve (ARC) can be steep, this could corrupt the reference estimate of the ARC.

Although not implemented herein, an imperfect but practical approach to partly address the case of a non-flat seafloor, would be to estimate the mean regional slope (dip and azimuth) from the chart (or accompanying multibeam survey). This might be adequate as the site is already deliberately chosen to not contain significant slope variations. This would better constrain the true location of the specular peak echo.

Method 2 - Grazing angle from phase ramp

Given all the limitations of the assumptions in the MRU approach to getting grazing angle, a more direct method would be preferred. To get the actual grazing angle, one should know the ray tracing and the seafloor slope. In this thesis an alternative method, based on the phase ramp was adopted, and its limitations (and advantages) are explained here.

The phase ramp across-track indicates the angle within the beam where the backscatter is coming from and the difference of this angle at two different times represents one angle of a triangle (Figure 28 below) calculated by the equation:

$$Gr = \sin^{-1} \frac{r_1 \,.\, \sin\beta}{\sqrt{r_1^2 + r_0^2 - 2 \,.\, r_1 \,.\, r_0 \,.\, cos\,\beta}} \qquad (11)$$

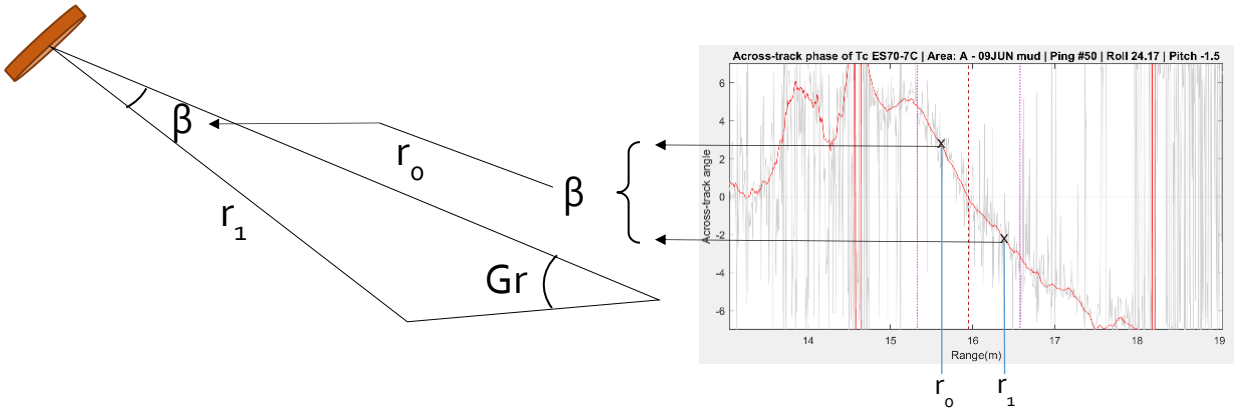


Figure 28 - Geometric calculation of the grazing angle based on information from the across-track phase.

As each time corresponds to a known range, and all the other measurements of such triangle can be calculated, one of them being the grazing angle (considering the limitations of this method).

One notable advantage of this method is that, because both ray paths have almost the same refracted distortion (just a fraction of a degree apart), the grazing angle calculated reflects the actual seabed impinging angle irrespective of the refraction. At low grazing angles, especially where there are strong refraction distortions, the difference between the launch angle and the arriving angle can be over a degree.

Because, for low grazing angles, we have multiple phase samples across the echo envelope, potentially the algorithm outlined above could give us a grazing angle for every pair of phase angles. In reality, of course, the instantaneous phase estimates are noisy. Thus, some phase smoothing is required. Herein, it was decided to get the average MRA relative angle at the center of each subset. Thus, between each pair of subsets, the grazing angle can be calculated and used to generate the figure of the bottom as illustrated. As can be seen, this allows us to recognize both the average across track seafloor slope over the whole echo envelope and identify if there are shorter wavelength slope variations within the beam footprint.

Figure 29 illustrates the seafloor across-track relief calculated by the phase of one ping.

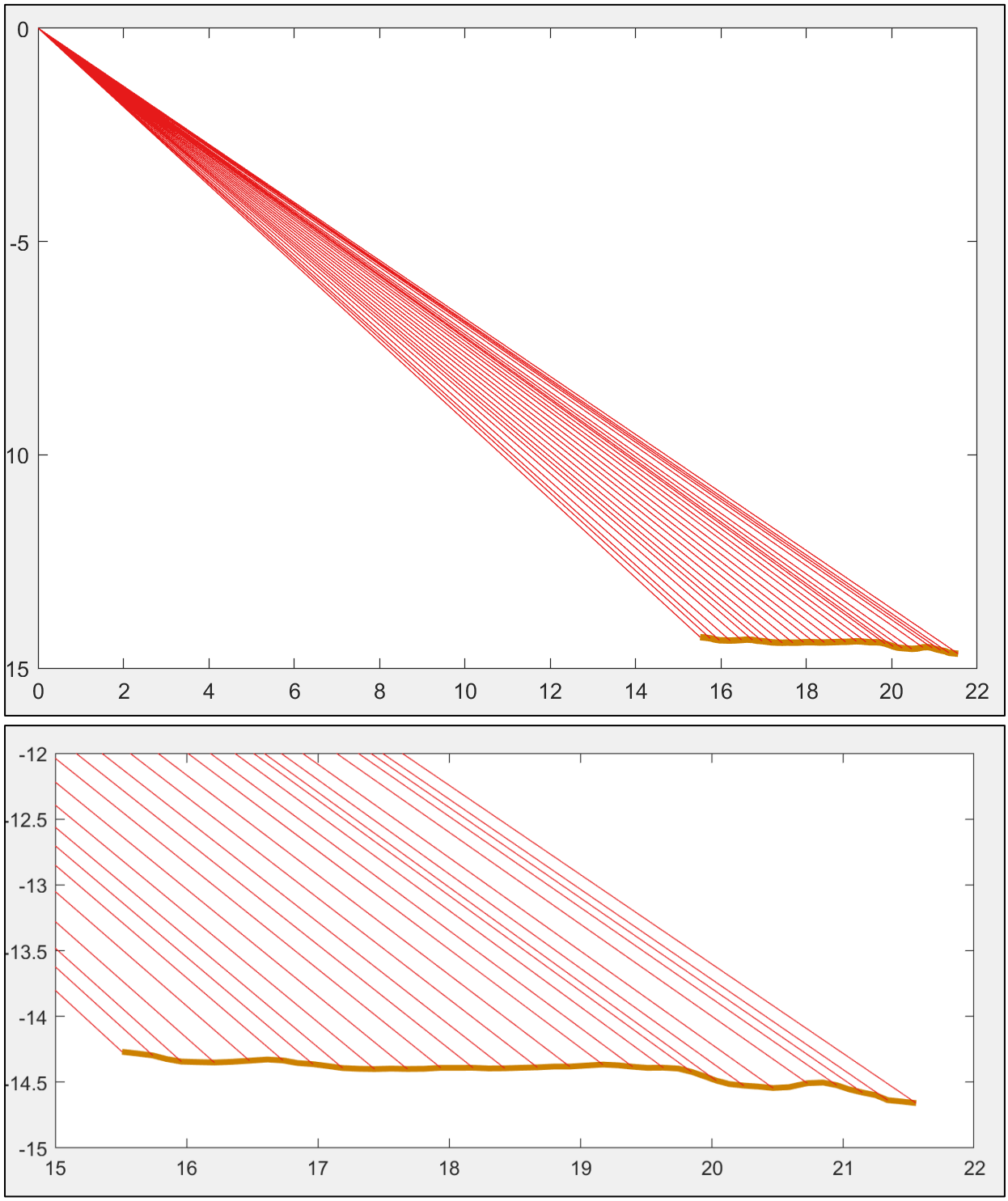


Figure 29 - Demonstration of one ping with multiple subsets that combined show the relief of the terrain on the across-track direction. The bottom plot is a zoom in the bottom.

The net result is that, for each bottom backscatter strength estimate from a subset, there is a unique associated grazing angle. Each echo envelope can thus potentially provide Sb estimates over a range of up to 5 degrees in grazing angle. As will be expanded upon in the results in section 4.2, the Sb value were sorted by 1-degree bins. And as the plate is being rotated no more than ~1 degree between pings, in a single sweep, five estimates of the Sb for a single bin may therefore be obtained.

As outlined in the discussion in the bottom detection approach, phase detection is only possible for beam vectors more than ~2 deg off the apparent normal incidence. Thus, within that range the simpler MRU approach has been defaulted to.

An additional concern is whether noise in the phase ramp results in false grazing angle estimation, thereby distorting the measured backscatter angular response. To minimize this the angle from the phase was calculated based on the mean of the moving average of the phase within the subset. Increasing the averaging length smooths out real across-track seafloor roughness, but potentially prevents false angle noise. The phase noise impacts how detailed one is able to calculate the micro reliefs. As each subset window represents an average of the relief along-track over a distance equal to the fore-aft beam width (~ 5 degrees), micro relief at shorter scales than this would be averaged out anyway.

In this discussion, it is important to acknowledge that only the across-track phase is considered as the along track phase is not useful for the usual high roll, low pitch situation, when the annulus covers an extended surface fore-aft. By not being able to use the along track phase there is a lack of consideration of the combined pitch of the transducer and any slope in the orthogonal direction (see Figure 20 above on section 3.7, quantifying the angular error if pitch is present). Thus, for this method to perform better, a flat seafloor and a good control of the plate to avoid pitching is mostly desired.

Although not implemented here, it is recognized that, for the special case of no roll and some pitch, the along track differential phase could in fact provide a valid grazing angle. Indeed, if there is about equal pitch and roll, a diagonal phase difference could also be considered.

To illustrate this effect, Figure 30 shows the mechanical angle derived from the difference of phase between fore/aft, (the "along-track phase"). Two examples are presented, one close to normal incidence and one very oblique (large roll). For both cases, the pitch is around the same magnitude, -7.7º on the left and -9.9 º on the right. the roll, on the other hand, is, on the left, just a couple of degrees, whereas on the right the roll is -61.6 º. The along track phase ramp is clear when the roll is close to zero, allowing the localization of the ensonification on the along-track, but for bigger rolls, the along-track phase becomes just noise.

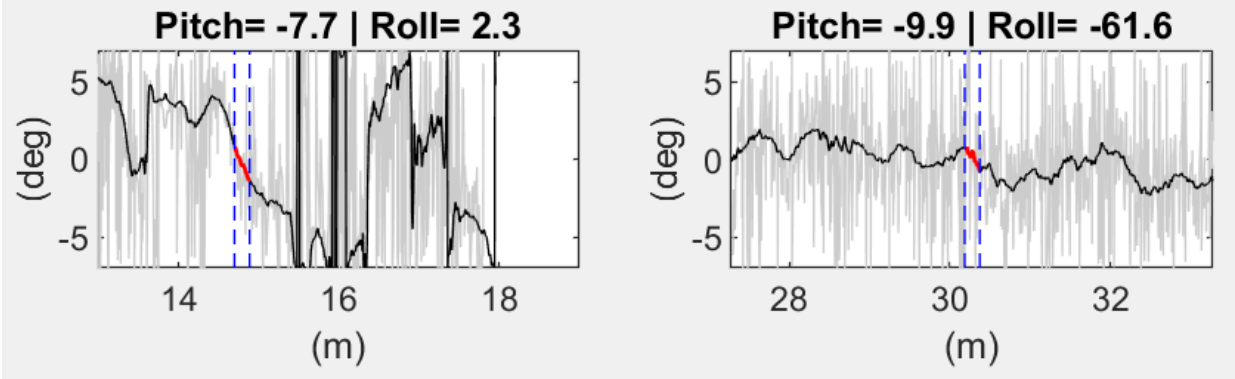


Figure 30 - Along-track phase.

In reality, by the time you get within a few degrees of normal incidence, the annulus can no longer be reasonably approximated by a linear swath either along or across the beam footprint. Rather it would appear as a circular subset. A more detailed examination of estimating grazing angle within a few degrees of normal incidence would be a useful future avenue of investigation.

Transmission Loss Spreading Corrections to the subsets

The range associated with each subset, that will be used to determine some corrections and to estimate the grazing angle, is calculated by a simple multiplication of the TWTT and the mean sound speed obtained by the MVP. This is then used for the transmission loss terms. Note that the subset is a finite range over which the TL does vary slightly. For the calculations performed herein, just the average TL at the center of the subset is applied.

The transmission loss (TL) term is composed of the two-way spreading and the absorption components, the first one being only range dependent, which is in dB calculated by:

40 log(R), or 20 log(R^2)

and should be added to the MF signal. In linear scale, the correction to the linear pressure term would be the product between the MF signal (MF_{linear}) and R^2, which corresponds in dB as:

20 log(abs(MF_{linear})) + 20 log(R^2)

Figure 31 shows the application of the range losses.

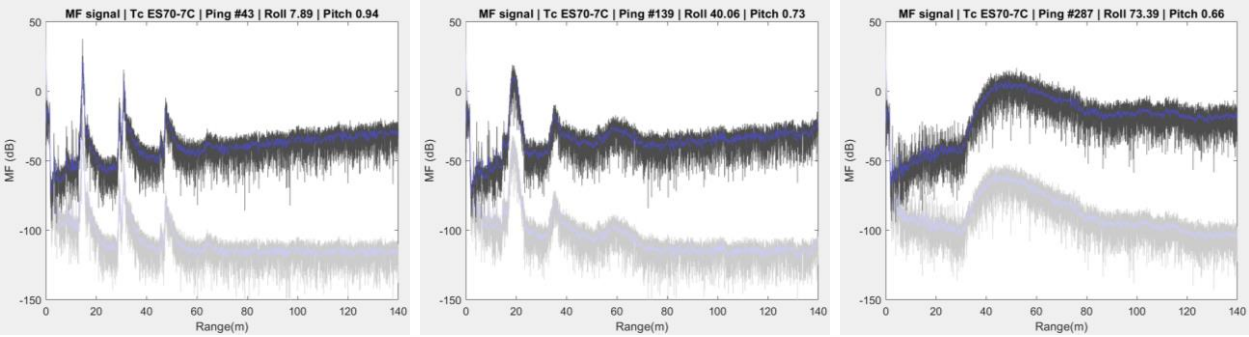


Figure 31 - Three plots of the MF signal without the geometric component of the transmission loss correction in light gray and with the 40LogR correction in dark gray, the blue lines represent the moving average of the MF. Each plot has a different roll (and thus grazing angle), changing the shape and the range of the peak response.

FFT with Normalization

In the same manner as applied in the sphere calibration, a FFT is applied to the (sub) extract of the MF signal (referred to here as the subset) normalized by the FFT of the autocorrelated transmitted signal with the same size as the subset. As justified previously, as part of this experiment, the FFT size was chosen to be 64 samples in order to obtain several independent solutions within a typical echo envelope.

In this manner, each subset will generate a frequency response associated with the mean slant range of the subset, a single grazing angle, and (for the cases of a pulse/FFT limited ensonified area) an angle within the beam with respect to the boresight. For a low grazing angle echo envelope this typically results in multiple solutions over about a 5-degree range of grazing angles.

The FFT results are then subjected to a sequential application of frequency dependent, beam pattern dependent, range dependent and area dependent corrections in order to arrive at a calibrated backscatter strength estimate for that grazing angle. These corrections are outlined below, except for the geometric spreading that is only range dependent which has already been discussed above.

Beam pattern correction

The maximum response axis (MRA) corresponds to the aiming direction of the transducer and any echo contribution for an angle away from the MRA will return a weaker signal according to the beam-pattern, as measured by the sphere calibration. As discussed in section 3.4, previously, that calibration is defined using the fore-aft (along) and port-starboard (across) angle derived from the differential phase. Unlike a point target such as the reference sphere, however, which can be considered to lie at a unique along and across angle, the instantaneously ensonified area potentially extends across a range of both along and across angles. For the pulse/FFT-length limited ensonified area case, only the across track angle can be identified. Thus, the beam pattern correction is applied differently in the across and along track direction.

For the along track direction, the equivalent beam width is estimated (close to the 3dB width) for each frequency bin. The resulting dimension due to that angle, projected over the slant range, is used as part of the ensonified area correction (see section 3.8).

For the across track direction, the port-starboard angle within the beam with respect to the MRA is calculated from the mean angle within the subset in the smoothed across-track phase, as explained previously. As an alternate to a simple mean, a fitting curve could also be utilized and may be subject of future study. The angle, then, is used to derive the across-track beam-pattern correction. That beam pattern correction is frequency dependent and thus needs to be applied for each subset across the frequency domain.

Figure 32 demonstrates the frequency dependence of the beam-pattern and its application on the frequency response of the MF signal. Top-left, there is a 3D view of the beam-pattern correction as a function of frequency and angle (with respect to the boresight), on bottom-left, the frontal view to highlight how a higher frequency has a narrower beam-pattern than a lower frequency. On the right-hand side, a 3D-plot of the frequency response of each subset along one ping to demonstrate how curved the response is before the correction for beam pattern (the dashed lines were used to better guide the viewer). Also, the MF signal in red was plotted out of scale on the background to show the pulse envelope.

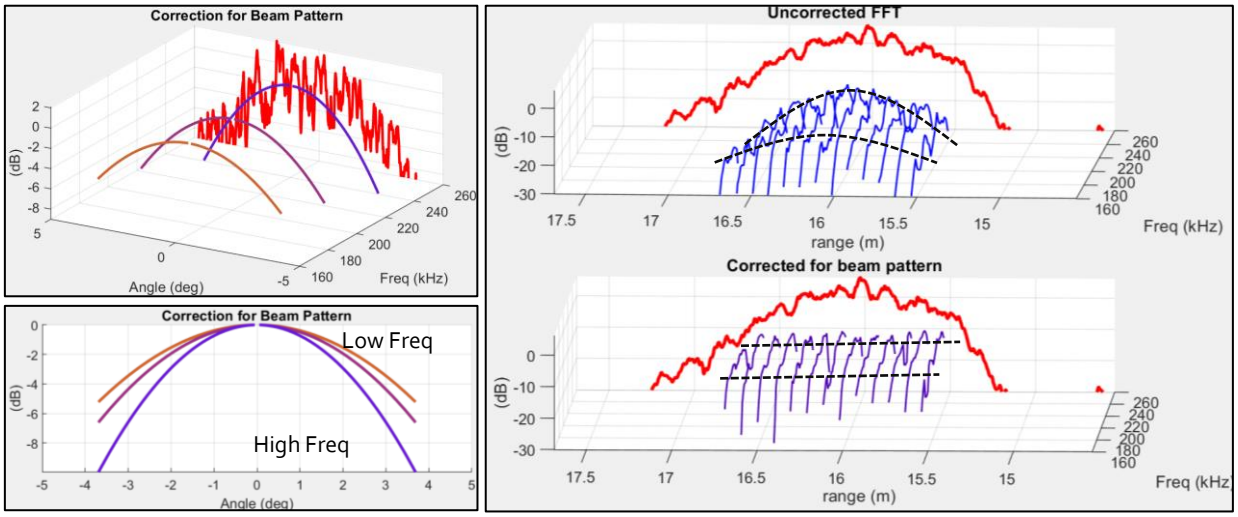


Figure 32 – Plots of beam-pattern and its application.

Calibration for MRA correction

The other factor derivable from the sphere calibration is the calibration curve for the MRA that should be applied after the beam-pattern correction, which is also frequency dependent. It is, however, invariant in range and is thus applied to all subsets within the beam regardless of the angle. As a result, the calibration is equally added to all subsets in the frequency domain. The Figure 33 shows the correction for the MRA of the ES70-7C. All corrections will be expanded in Chapter 4.

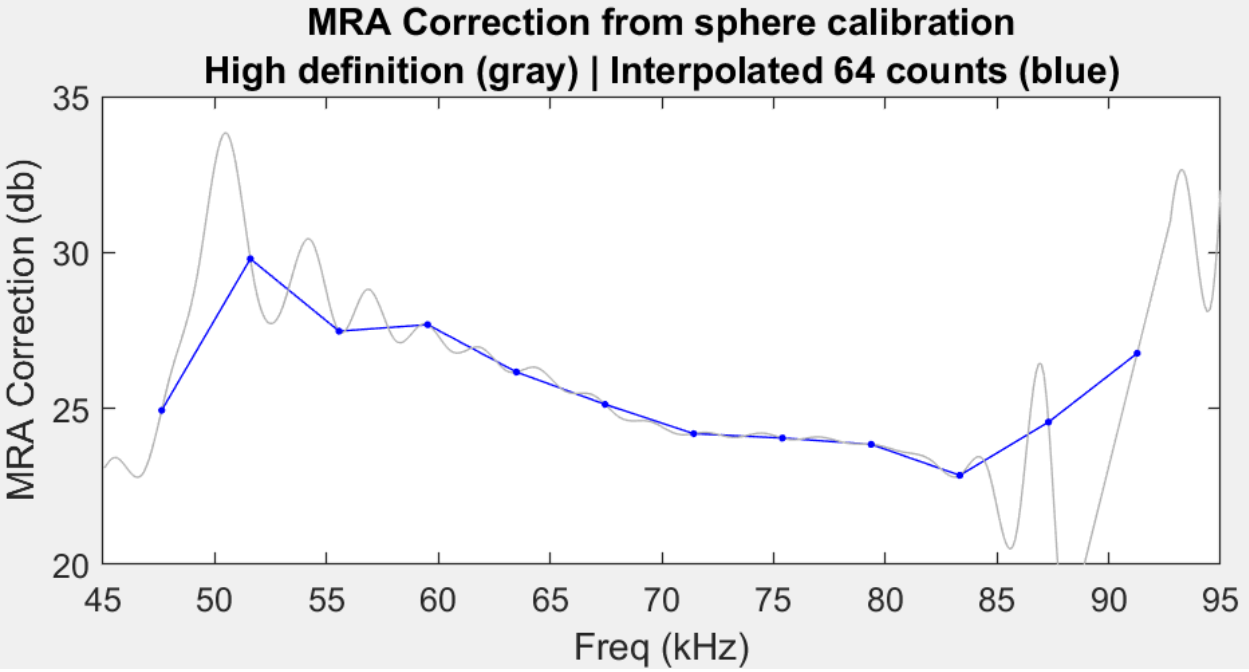


Figure 33 - Correction for the MRA of the ES70-7C.

Absorption correction

As mentioned in the paragraph about the spreading correction, the absorption is part of the TL and is frequency dependent, and thus this term is only applied in the frequency domain after the FFT, rather than together with the spherical spreading term in the time domain.

The absorption is a function of several factors such as slant range, frequency, temperature, salinity, pressure, pH and depth. Although strictly the absorption coefficient will vary with depth due to pressure and oceanographic stratification, for each frequency bin in the FFT, a single mean coefficient is used herein and applied uniquely for each subset as a function of slant range. The corresponding frequency dependence as a function of slant range is illustrated in Figure 34.

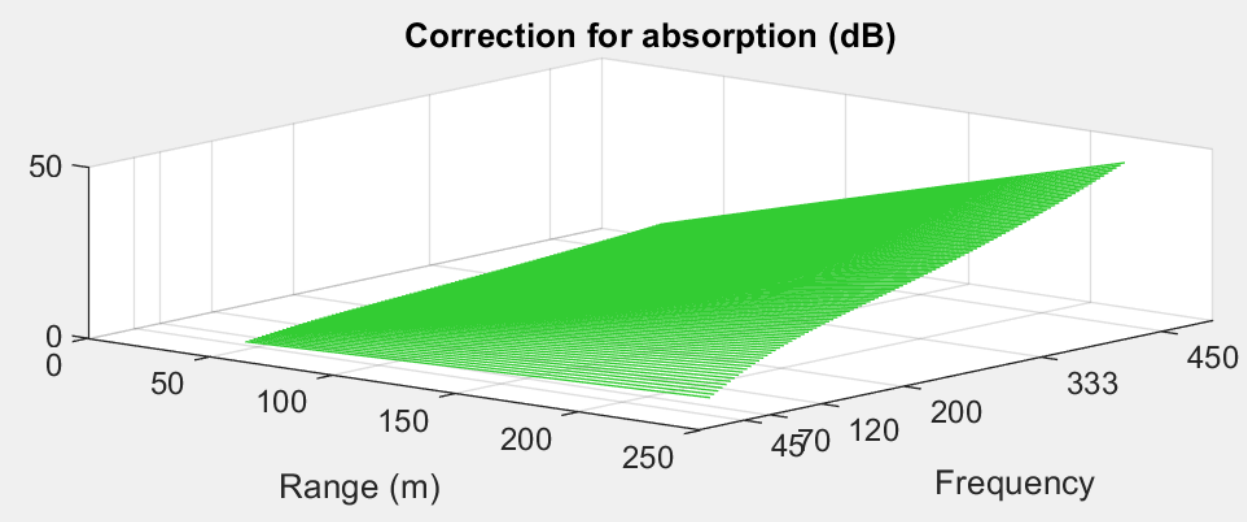


Figure 34 - Absorption loss as a function of frequency and range in dB with estimated environmental parameters.

Area normalization

To go from the Target Strength to a Bottom Backscatter Strength, the instantaneous ensonified area needs to be estimated. This area corresponds to that subset of the projected annulus (with a radial width controlled by the FFT length) which lies within the projected beam pattern.

While the beam pattern is strictly a continuous function over the entire sphere (all azimuths and all elevation angles), practically one can limit it to within the main lobe and define that limit by using an equivalent solid angle centered about the MRA, within which the peak power is radiated. For narrow beam ensonified areas such as multibeam footprints, commonly the 3dB beam width equivalent is used. As demonstrated, however, by Weber (2013), when integrating the total power over all solid angles, a better typical approximation is about 1.15 x the 3dB limit. That approximation was used herein.

Given the constraint of an equivalent beam width, there are two discrete cases where the ensonified area is defined differently. The choice of case depends on whether the projected radial (across track) dimension of the annulus is larger than (the beam-limited case) or smaller than (the pulse/FFT-limited case) the projected across-track bean width. This geometry, first defined in Urick (1954), is illustrated in Figure 35, where a long pulse has a circular ensonified area near nadir, whilst with a short pulse, the ensonification becomes an annulus instead of a full circle, and at oblique incidences a section of the annulus is the geometry of the ensonification and is represented at different times of the same ping, before the boresight, at the boresight and after the boresight.

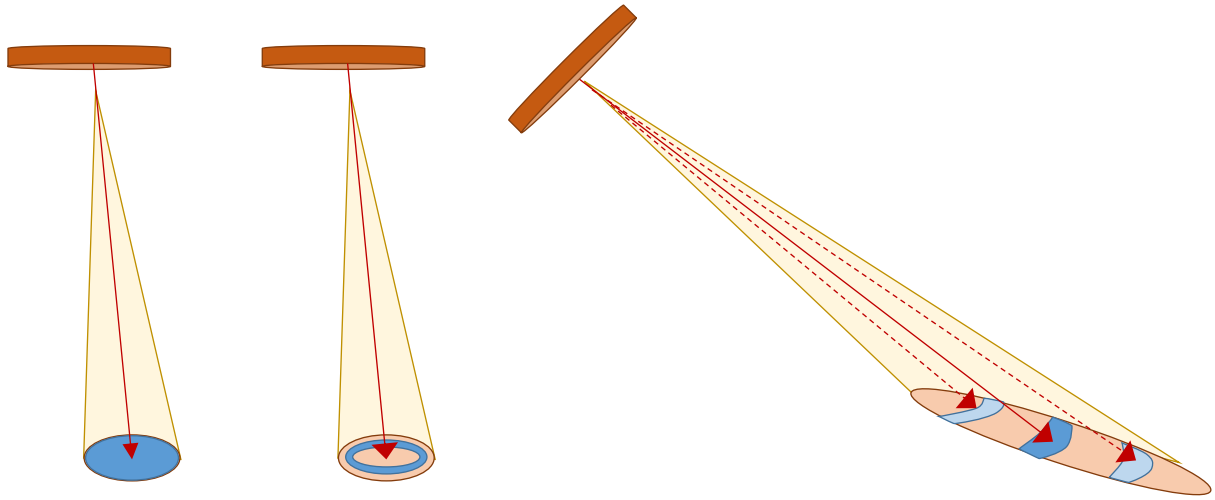


Figure 35 - Sketches of ensonified area, two near nadir with long pulse and short pulse and one with oblique incidence.

Commonly those two cases are managed with simplifications:

For the beam-limited case, the area is approximated as a rectangle with along and across track sides defined by the projected along and across equivalent beam widths:

$$\mathbf{A_0} = \psi_{al} \ \psi_{ac} \ R^2 / \sin(Gr) \qquad (12)$$

Where ψ_{al} (Along-track) and ψ_{ac} (Across-track) are the two-way equivalent beam angles in radians, varying with frequency, R is the range to the center of the subset ang Gr is the grazing angle of the subset.

For the pulse/FFT-limited case, the area is commonly approximated by a rectangle of dimension of the along track projected equivalent beam width and the across track by the projected pulse/FFT-length as defined in:

$$\mathbf{A} = R \ \psi_{al} \ c \ \mathbf{T} / 2 \cos(Gr) \qquad (13)$$

Where c is the average sound speed and T is the time-length of the FFT for a MF signal (Weber and Ward, 2015), whereas for a CW pulse this term would be the pulse length.

Herein these approximations are used. Strictly though, as the beam width gets larger, they are not ideal, as at the highest grazing angles, the curvature of the annulus within the projected beam footprint becomes significant. This becomes more acute as the FFT length used gets shorter so that pulse-length limited case is maintained at higher grazing angles. To do a more accurate calculation, however, would only be justified if the grazing angle were precisely known. As described in the grazing angle estimation section, at these highest grazing angles, the across track phase cannot be used and thus the pitch and roll of the sonar is substituted.

The along and the across track equivalent beam angles are usually defined separately to accommodate the case of Mills Cross geometry where the transmitter and receiver arrays are of differing lengths. For the piston source used here the calibration of the transducers demonstrated that the beam-pattern is approximately axisymmetric whereby the along-track and across-track angle for the beam-pattern have the same value, only varying with frequency.

It is acknowledged though, that errors from the calculation of the grazing angle, the beam-pattern, and the range will lead to frequency-dependent biased error in the area estimation.

Add all corrections

As the sonar equation states in dB, the Sb is:

$$Sb = EL + 40 \log R + 2 \alpha r - SL - 10 \log A \qquad (14)$$

EL – SL is represented by the MF signal and includes the calibration.

“40 log R” was implemented as mentioned by the “spreading correction” paragraph.

"2 α R" is added as calculated by the "absorption correction" paragraph.

"10 log A" is subtracted based on the term "A" as estimated by the "Area normalization" paragraph.

The calibration terms of "beam-pattern" and "the MRA correction" are also considered according to their respective paragraphs.

Thus, the final term is the actual S_b result in terms of frequency and grazing angle for all subsets and all pings.

Compilation of S_b observations into Bins of 1 degree and 4kHz (64 counts)

With all the corrections added to the seafloor response, in the time and frequency domain, it is possible to compile the resulting measured grazing and frequency dependence of the S_b. The subsets, however, produce instantaneous values with pronounced random components (speckle) which, as discussed previously in this section, can have a standard deviation approaching the mean value. Thus, averaging is required to reveal the mean response.

The frequency domain is already discrete based on the size of the subset as mentioned previously, at about 3.9kHz for a selected FFT window of 64 samples. In order not to smear the frequency dependence further, no additional averaging in frequency space was attempted.

The acquired data provided apparent grazing angles in the range 5 to 90 degrees and each subset has a corresponding value. When considering the choice of a suitable grazing angle bin size, the first factor is uncertainty in the grazing angle measurement itself. The second factor is the number of available samples within the chosen bin size to average. To obtain enough samples to average, four aspects were controllable:

- the ping rate for a given transducer,
- the rotation rate of the transducer,
- the size of subsets within each echo envelope and
- the amount of time spent pinging.

A typical resulting histogram distribution by grazing angle is illustrated in Figure 36.

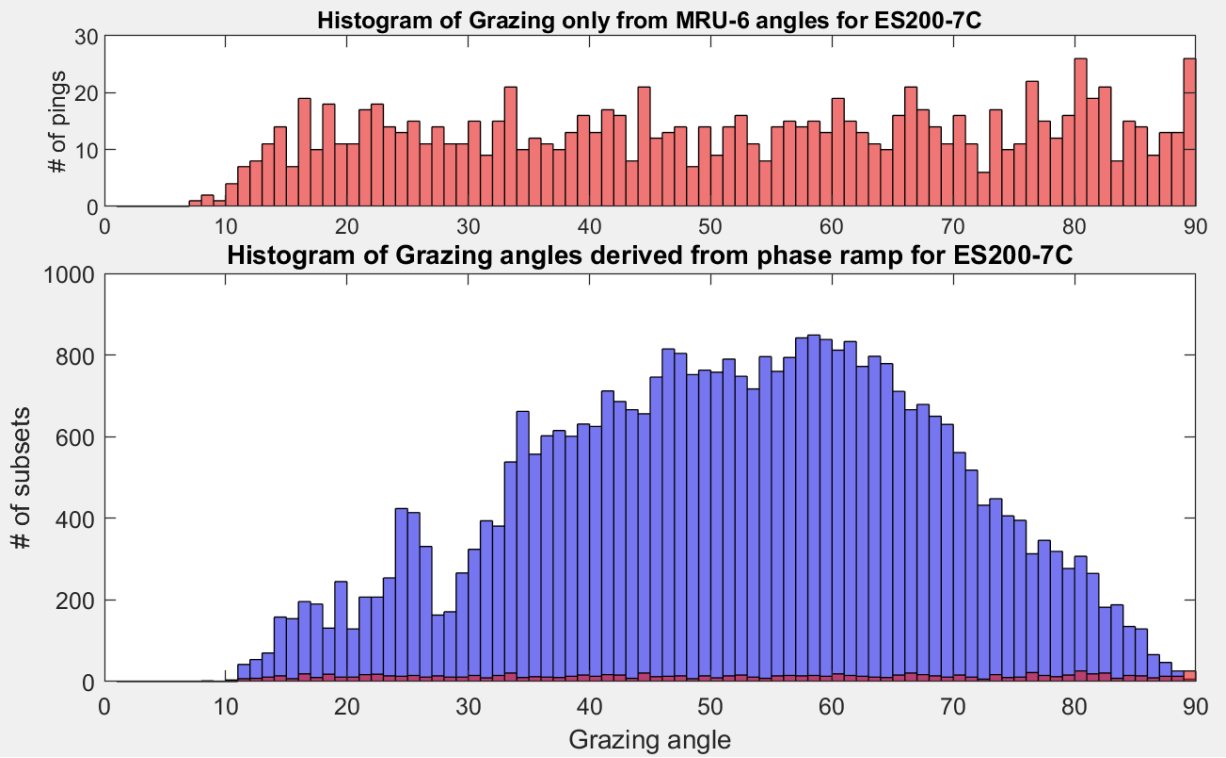


Figure 36 - Histogram of samples of the correspondent grazing angle (within 1-degree) calculated only by the MRU-6 on top (notice the scale) and a comparison with the grazing angle calculated by the phase on bottom.

As can be seen, from the nadir to about 60º of grazing angle, the observation density increases with decreasing grazing angle, remaining approximately the same amount up to 40º, and decaying up to the lowest grazing angle. This is a result of the number of subsets possible within the echo envelope increasing with lowering grazing angle. The curve flattens and decline as a result of the limitation imposed to the extract of the MF used to compile the subsets, which implied in a limitation of 61 subsets of 64 samples within the extract of 2001 samples (see Figure 37).

This limitation would not be required if the project worked with unlimited extract of the MF data, which would increase even more the number of subsets for low grazing angles. The lowest density unavoidably occurs at the highest grazing angles close to normal incidence where the steepest variation with grazing angle occurs. The choice was made in this thesis to average the responses within one-degree bins. The net effect is that for the results presented, each bin is the average response of 1 degree and ~4kHz.

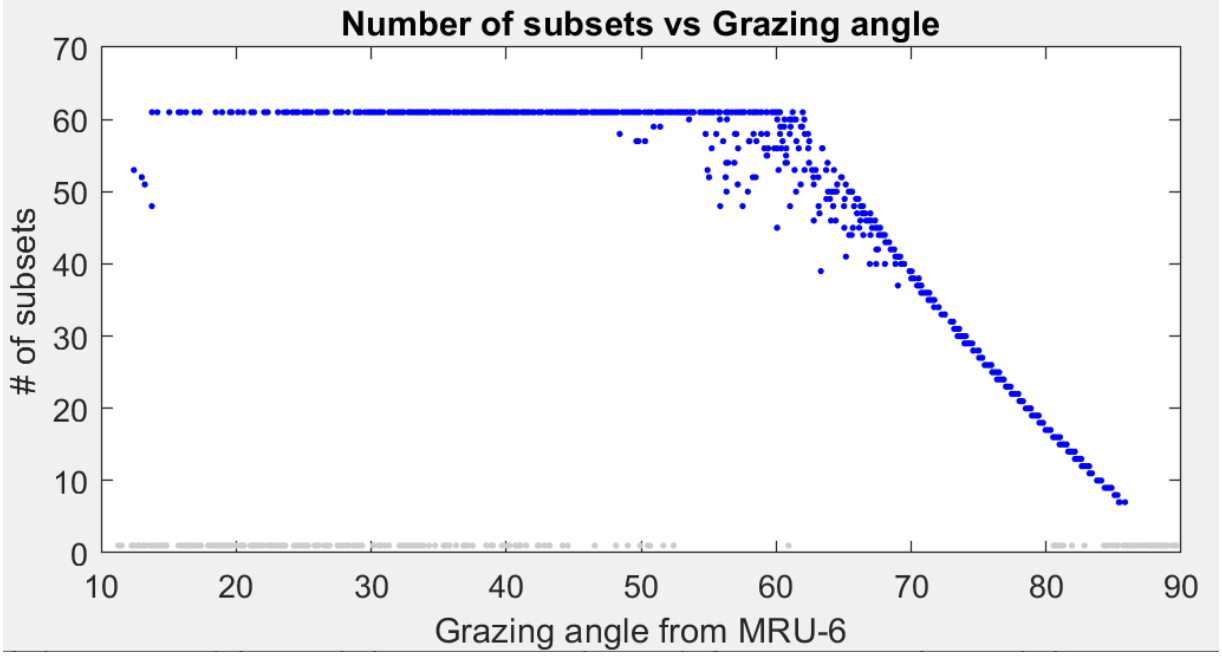


Figure 37 - Number of subsets in function of Grazing angle for the ES200-7C in Area C (14m deep) The grey dots represent the pings where there was only amplitude detection and therefore, only one subset.

Average S_b in linear scale for each bin

The average is calculated for each set of S_b responses that are within the one-degree range from 0-90 degrees. It is possible to overlap results by taking the average of the same range of one-degree varying from 0.5-89.5 degrees, but it was not done in this thesis.

It is important to note that the S_b is in dB, but the average should be calculated in linear intensity units. That way, the S_b is transformed into linear Intensity to have the average calculated and then is transformed back to dB.

$$S_b(f, gg) = 10 \log_{10} \left[\frac{1}{N} \sum_{i=1}^{N} 10^{BS_i(f,\ gg)/10} \right],$$

for $(0 > gg \leq 1), (1 > gg \leq 2), \ldots, (89 > gg \leq 90)$ (15)

STD of S_b in linear scale for each bin

The same logic of the previous paragraph is applied to calculate the standard deviation.

$$\text{STD}(f, gg) = 10 \log \left[\sqrt{\frac{1}{N-1} \sum_{i=1}^{N} 10^{BS_i(f,gg)/10} - 10^{BS(f,gg)/10}} \right],$$

for $(0 > gg \leq 1),\ldots,(89 > gg \leq 90)$ (16)

Final absolute response curve

The final absolute response curve is a shortened dataset with one value of S_b in dB for each bin of one degree by ~4kHz within the range of frequency operated by each transducer.

Since the corrections are applied by each transducer independently, there will be overlapping results for the bins at the edges of the frequency range of the transducers, except between the ES200-7C and the ES333-7CD because their frequency range do not overlap. Those overlapping bins from different transducers should ideally have the same S_b value.

The S_b curve in terms of frequency and grazing angle is then the reference that any system should be able to compare when collecting S_b data of the same seafloor and any difference should be considered to calibrate the system.

4. RESULTS

4.1. SPHERE CALIBRATION RESULTS

Before attempting seabed backscatter estimation, the first requirement is to calibrate the transducer/transceiver with respect to a reference target. This section described that calibration process. The sphere calibration was performed in three different sites with different environment conditions, but with all parameters properly considered in the method, each site should deliver the same correction values (assuming no degradation in the transducer/transceiver hardware).

The first attempt was performed in May 2019 at the lab tank in fresh water, with the water temperature around 21^{o}C and the sound speed in the water about 1486m/s. The Tc ES333-7CD was not available yet, so, the calibration was performed only with the other three transceivers.

The sphere calibration results that were applied in this thesis were obtained in BC, Canada, on 12th June 2019, in salt water with temperature around 11^{o}C, a salinity of 30 PSU and the sound speed in the water about 1489m/s

Another sphere calibration procedure was performed also in salt water in Portsmouth, NH in August 2019 where the water temperature was around 17^{o}C, the salinity was 30 PSU and the sound speed in the water was about 1492m/s and its results are displayed for comparison.

Comparison between Tank, BC and NH

The calibration curves for each transceiver in three different environments should be ideally the same.

The first plot (Figure 38) shows the result using a high resolution in frequency (longer FFT), where the apparent oscillation in the calibration is clearly present. When applied, however using a shorter FFT window, for example 64 samples equivalent to that utilized for the seabed subsets, the oscillations will not be present. As the calibration, by necessity must cover a window of around 60 cm to capture the full target characteristics, the difference in frequency resolution has to be accounted for in the application of the calibration.

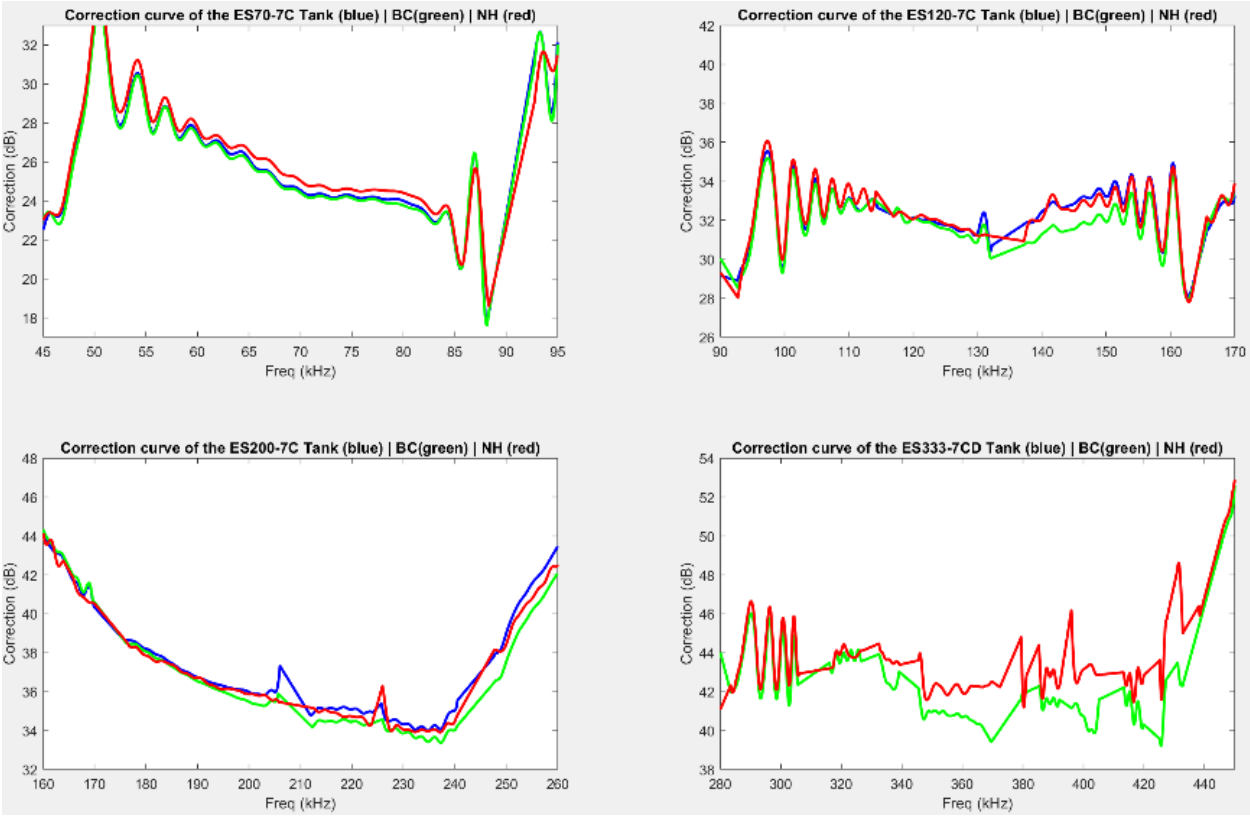


Figure 38 - Calibration curves for each Tc at three different environments, CCOM's fresh-water lab tank in blue (the Tc ES333-7CD was not calibrated in the tank), the calibration in BC in green and a calibration procedure performed in Portsmouth, NH in red.

To try to handle this, a smoothing was applied to the calibration curves (Figure 39) before application to the seabed subset spectra. As has become apparent when examining the results (see below), for all except the 200 kHz transducer, there remain small but clear systematic frequency-dependent calibration residuals. These are probably a result of improper handling of these calibration oscillations. This is further complicated by the necessary interpolation over the nulls, where those oscillations cannot be accurately reproduced.

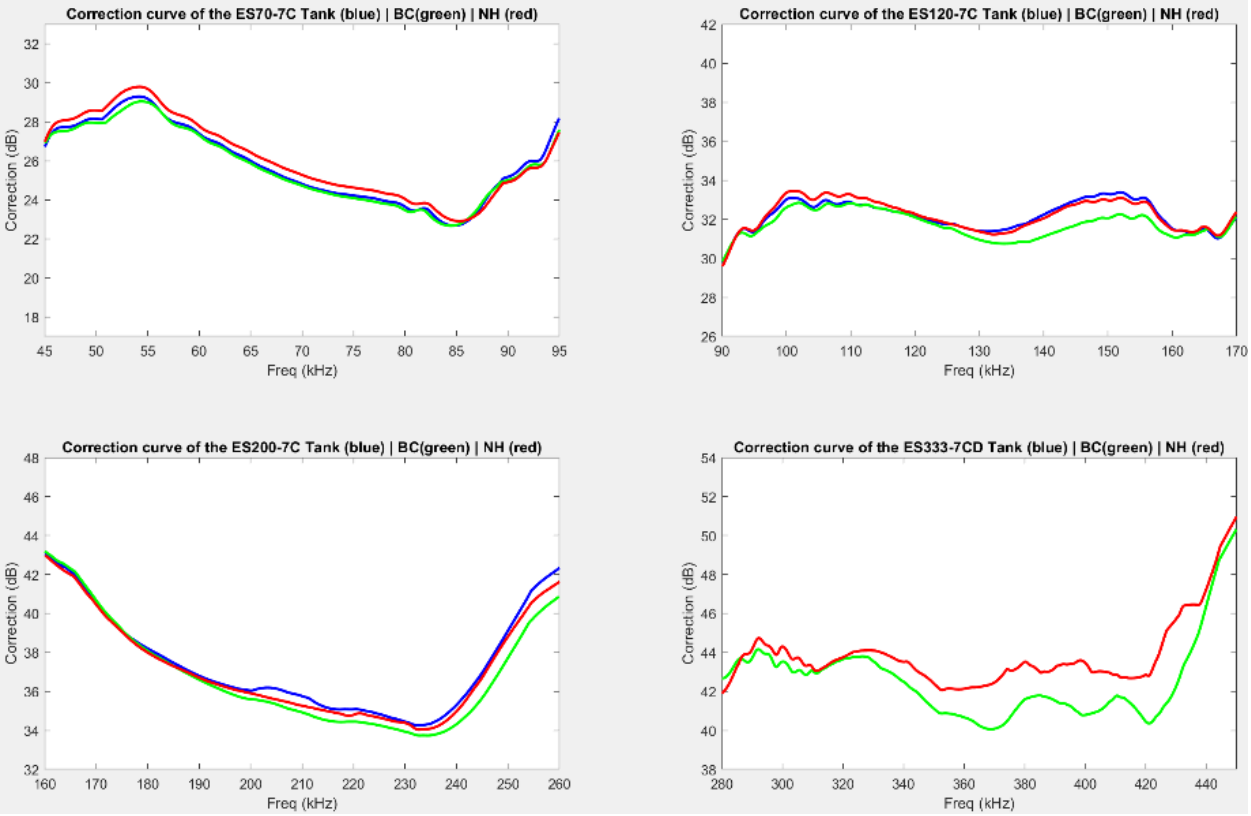


Figure 39 - Smoothed calibration curves for each Tc at three different environments, CCOM's freshwater lab tank in blue (the Tc ES333-7CD was not calibrated in the tank), the calibration in BC in green and a calibration procedure performed in Portsmouth, NH in red.

Figure 40 illustrates the comparison between the calibration results obtained at the three areas and indicated that the calibration differences were within about ±1.5dB. In each case more than 300 pings for each TC were statistically independently collected to get to this result.

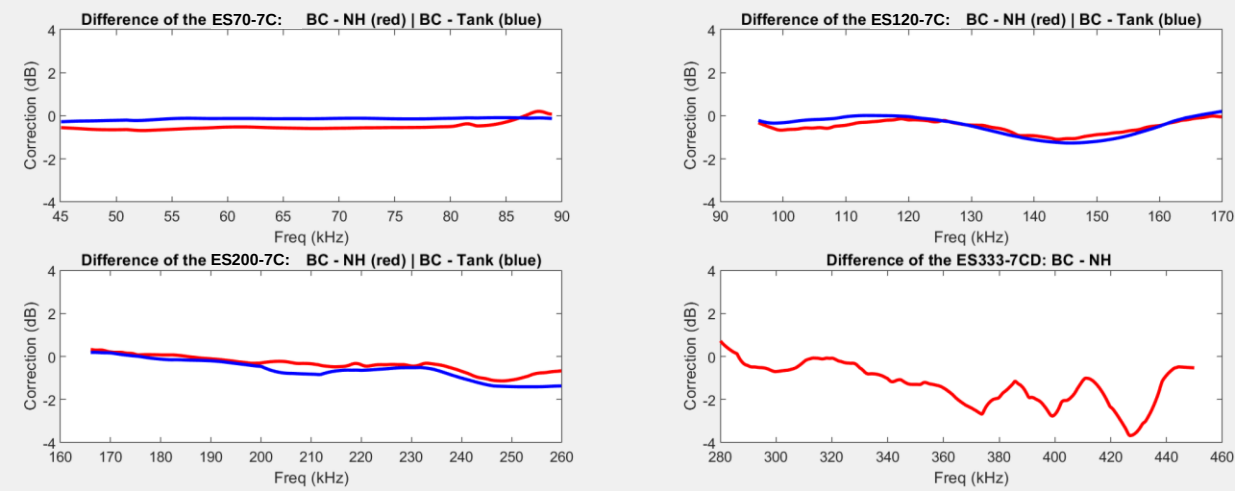


Figure 40 - Difference of calibration results from BC, NH and CCOM's tank for each transceiver.

The notable exception was the ES333-7CD whose calibration was seen to be more divergent for frequencies higher than 330 kHz. A further research should address the reasons for that the 333 kHz mismatch, but, as will be shown on the next section, the excessive number of nulls that the sphere utilized by this thesis present at higher frequency suggests that a sphere of a different diameter should be more appropriate for ES333-7CD.

Despite the application of this smoothing, the final seafloor results (see section 4.3) indicate that there remain small systematic frequency dependent residuals in the frequency domain which are clearly not real. To address this, they could be subsequently removed by fitting a trend to those results (see Figure 41). This step, while resulting in a more believable smooth frequency dependence, does at this time leave a question about the absolute fidelity of the resulting reference data.

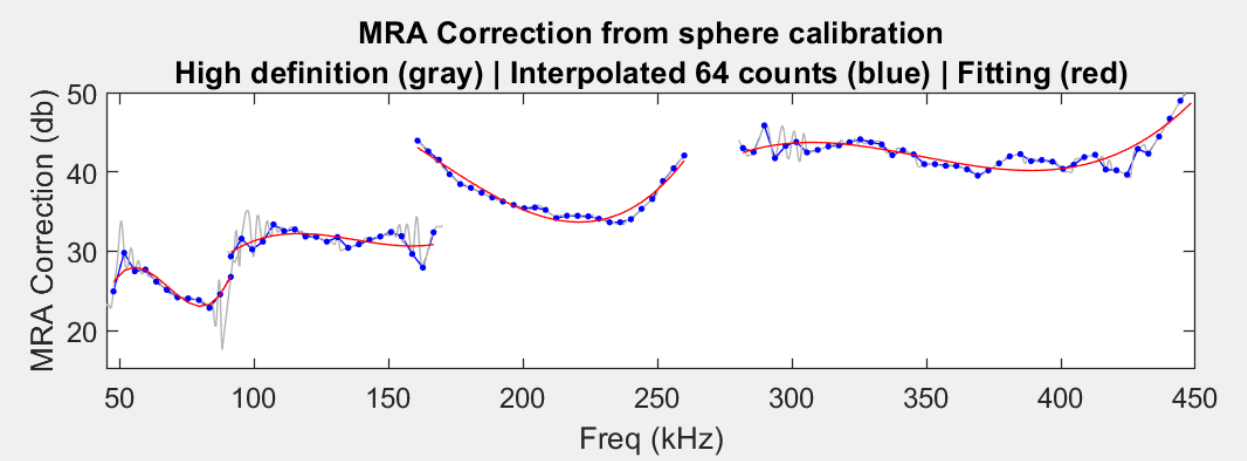


Figure 41 - MRA corrections from sphere calibration for all transceivers. High frequency resolution (gray), downsampled to 64 counts (blue),and correspondent fitting curve (red).

Comparison of the observed reference target nulls with the model

A comparison between the average response obtained from each transducer and the ideal TS for the model sphere (Figure 42) indicates that the exact null locations were often slightly offset. As a result, the frequency range in the vicinity of the nulls should not be considered. More spheres of different diameters that have nulls at different positions in the frequency range could be used to have a more complete response, however in this thesis, a linear interpolation across the null windows was done when necessary. The nulls and their vicinity were identified using the slope of the model curve. Regions with > 0.1 dB/kHz were discarded.

As mentioned in the previous section, the number of occurrences of null is greater as the frequency increases, to the point that for the ES333-7CD, a large number of gaps need to be interpolated, leading to more errors, as shown by the Figure 42.

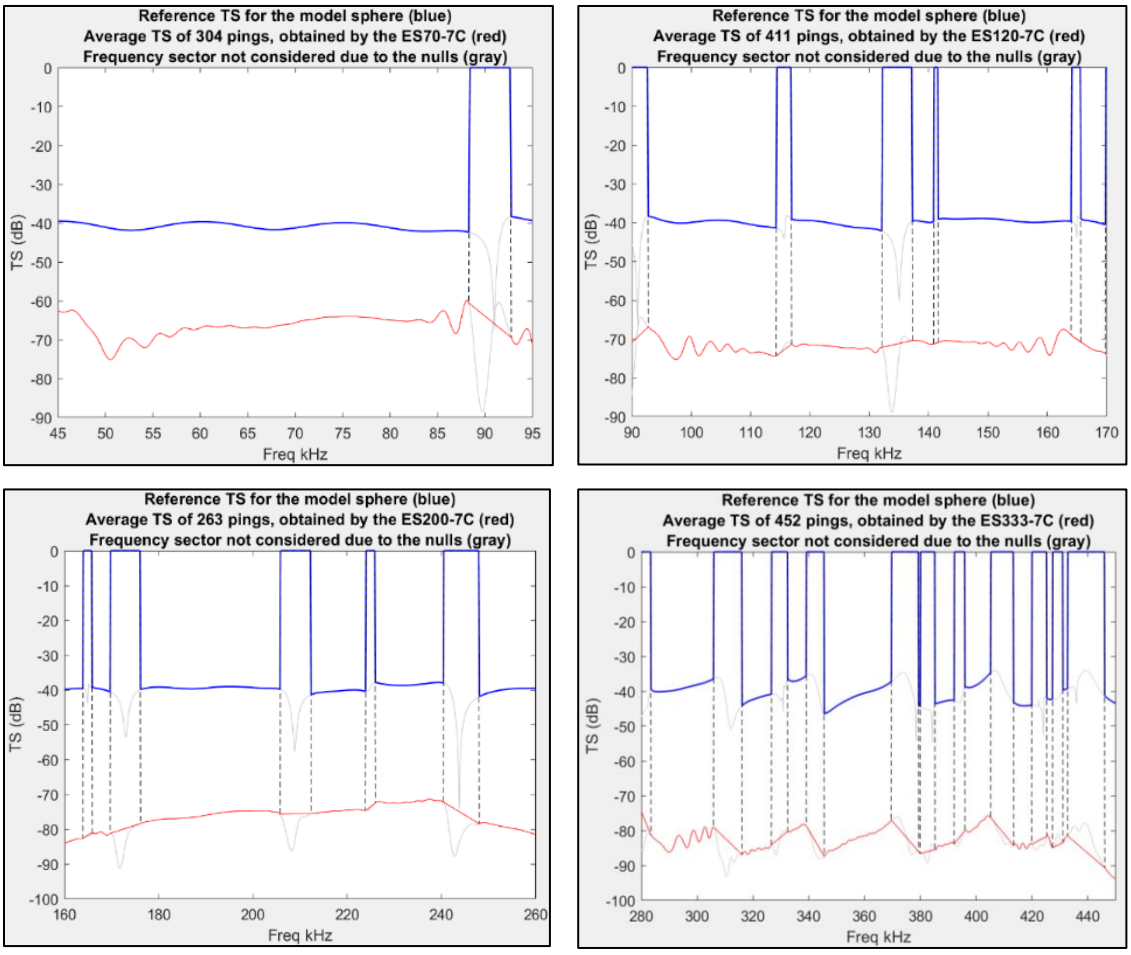


Figure 42 - Plots of the ideal TS response from the reference sphere in blue, the mean response from each TC in red and the frequency sector not considered due to the nulls in gray.

Additionally, in the inter-null regions, as the frequency increases, the slope of the TS curve gets steeper. Thus, any inaccuracy in the estimate of the sphere diameter and/or P and S wave velocity that might shift the curve in frequency, will result in larger mis-calibrations.

Beam pattern estimation

The combined transmit-receive beam pattern (i.e., two-way) derived from the sphere calibration indicates how the intensity of the signal drops as the angle increases away from the MRA and how it varies with frequency. These results (Figure 43) will be applied as a correction for each frequency bin in two ways. Firstly, the equivalent along–track beam width (1.15x 3dB limits) is used as part of the ensonified area calculation.

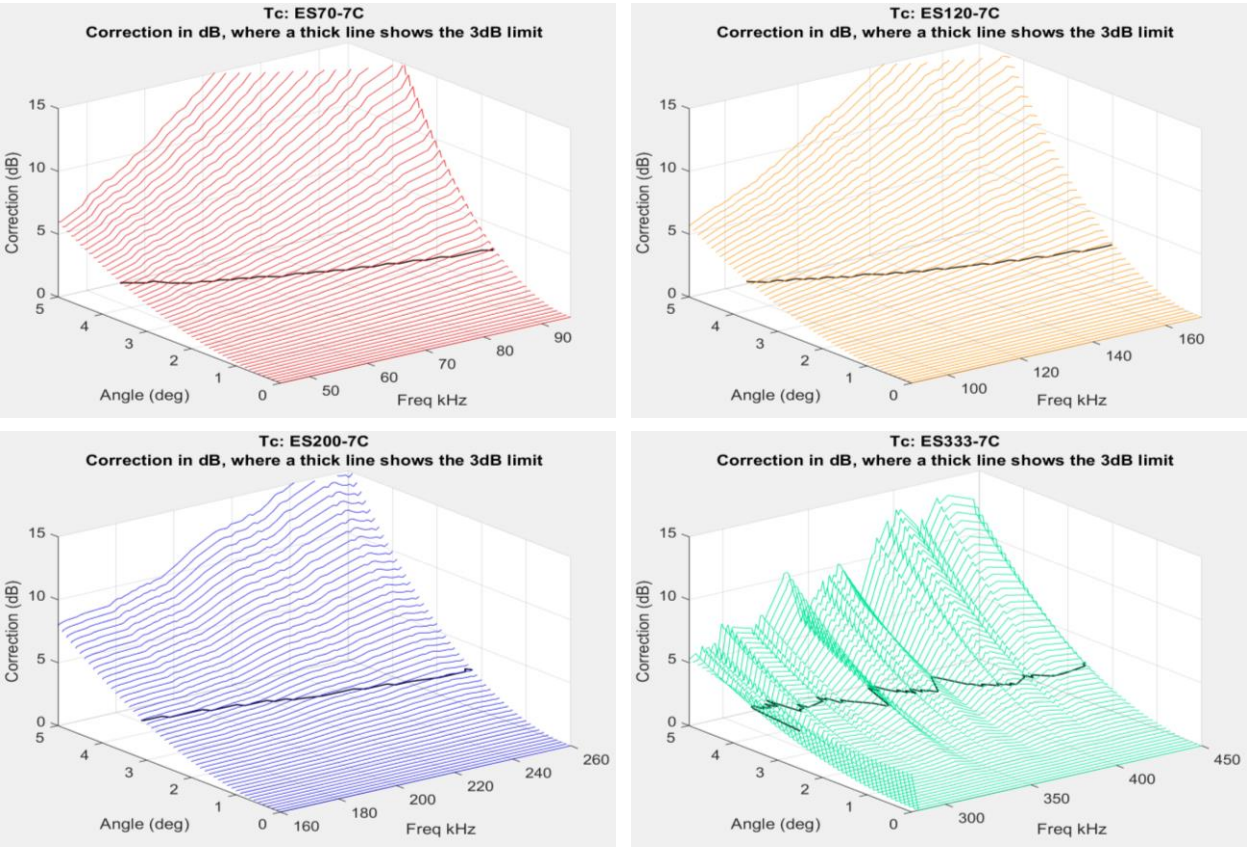


Figure 43 - Beam pattern of each transducer obtained by the sphere calibration with a highlighted line of the 3dB correction value.

Secondly, the across- track beam pattern is used in one of two possible calculations: for the beam-width-limited case, it defines the equivalent across track beam width, or for the pulse-length-limited case, it controls how each subset is gained up depending on where its center lies relative to the MRA. At the MRA, angle = 0, there is no correction.

To support the equivalent beam width estimation, the angle at which the intensity is half of the value at the MRA, or -3dB, is calculated (Figure 44) which, of course, also varies with frequency. Note that this is expected to vary smoothly with frequency and does indeed do so for the 70, 120 and 200 kHz transducers. An irregular pattern, however, is observed for the 333 kHz transducer suggesting imperfect calibration, again possibly related to a poor choice of sphere.

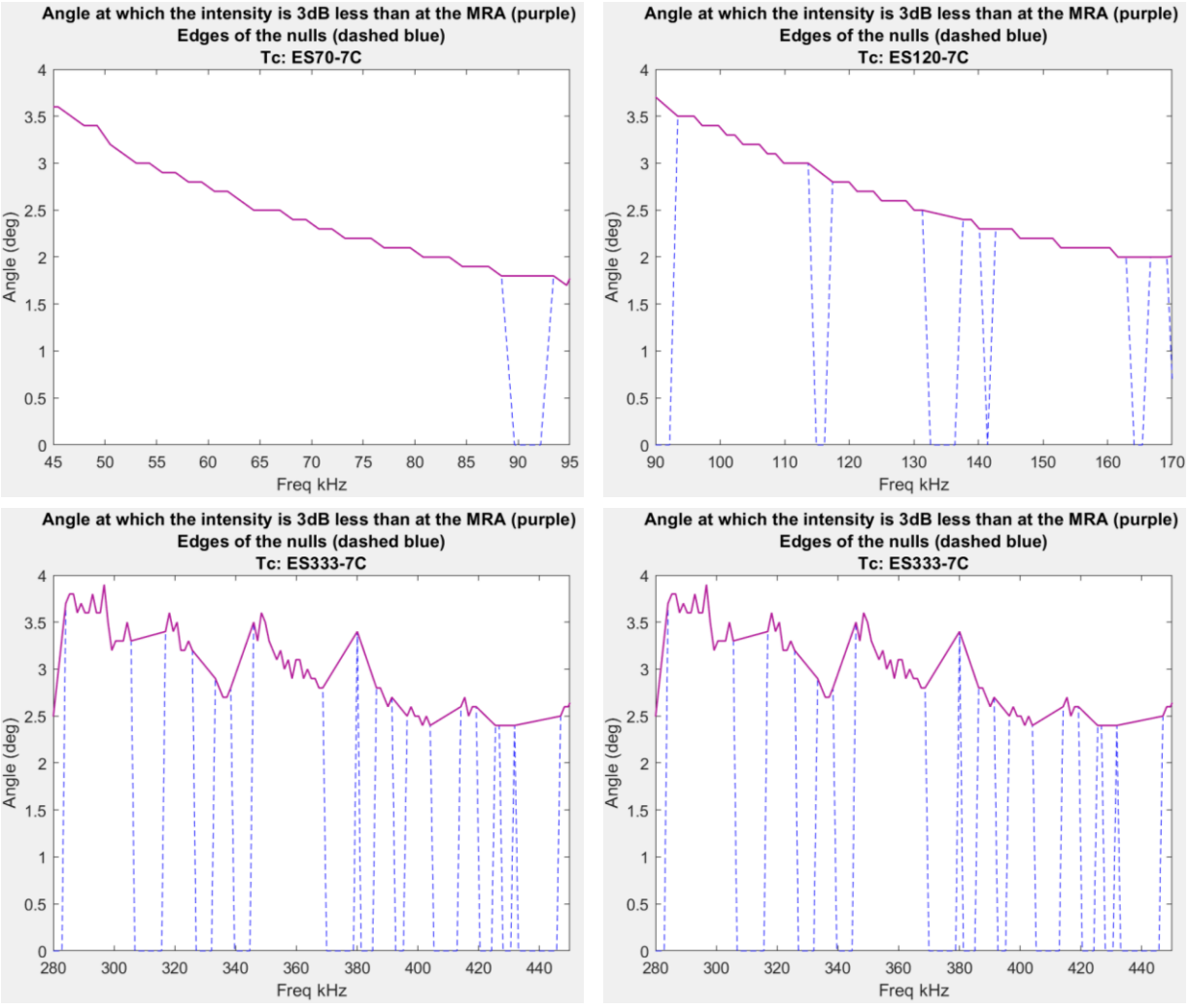


Figure 44 - Angle of intensity 3dB less than the MRA in function of frequency for each Tc, with the edges of the nulls in dashed blue.

4.2. BOTTOM BACKSCATTER RESULTS (BY FREQUENCY AND GRAZING ANGLE)

As described earlier, five areas in BC, Canada were selected due to their homogeneous characteristics, their flatness, and other qualities addressed in section 3.3. The five areas were also deliberately chosen to be widely different seafloor types covering much of the expected range of sediments from muds to gravels.

All four transducers were set to ping sequentially an FM pulse to get a complete response over the full bandwidth investigated. The sequential pinging was set up to avoid leakage between overlapping sections of bandwidth. The plate on which they were mounted was mechanically rotated, in a smooth way, several times, to obtain the response from any grazing angle from 90, nadir, to near 5 degrees. Also, the plate was rotated in the Z-axis to investigate any azimuth dependence. Figure 45 illustrates a typical view of the resulting distribution of beam footprint centers from above. As can be seen a reasonably even distribution was normally achieved. Poorer distribution was normally a result of anchoring in a tidal stream which forced the plate to preferentially align across the flow, resulting in measurements predominantly just upstream or downstream. The results are detailed in this section.

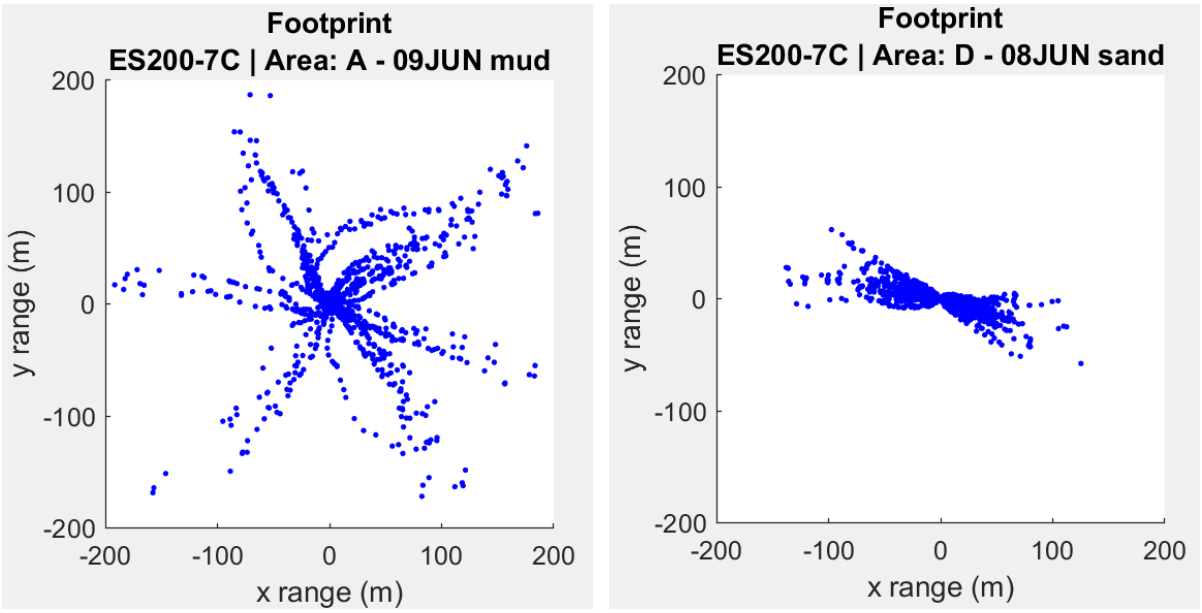


Figure 45 - Footprint of Area A (55m deep) on left and of Area D (17m deep), with strong tide currents on right.

Footprint Location and Bottom Detection Failures.

As previously described in section 3.3 (Figure 14), the slant range depends on the altitude of the transducer and the grazing angle. For a given slant range and source level, the ability to achieve sufficient SNR depends on the slant range, the bottom type and frequency, the two main varying factors that define this being the TL, and the Sb. In the event of insufficient SNR, the bottom detection process was not successful, and that ping was recorded as rejected.

In this thesis, the areas chosen were between 14m (Area "C") and 55m (Area "A") deep. Area "A" was not only the deepest site but also had the sediment with lowest Sb, contributing to the lack of received signal at higher frequencies and lower grazing angle. Figure 46 and Figure 47 plot the position XYZ in meters of each subset with a dot, based on the heading, pitch, roll and range (heave was neglected), to show that the seafloor was sampled in different azimuths. It can be seen that the number of rejected soundings in "Area A" is larger than in "Area C" due to the depth and the type of seafloor. The ES333-7CD has more gaps due to its lower SNR, mainly on "Area A", where it is deeper. One way to avoid that in the future, and also to undertake this procedure in even deeper waters, would be to deploy the equipment closer to the seafloor by suspending them on a cable below the vessel (as was done by Gensane, 1989). This of course comes with additional complications about how to achieve the elevation and azimuth rotations. Another advantage of being closer to the seafloor is that the horizontal distance would be shorter, reducing the required radius of the area needed with the same sediments.

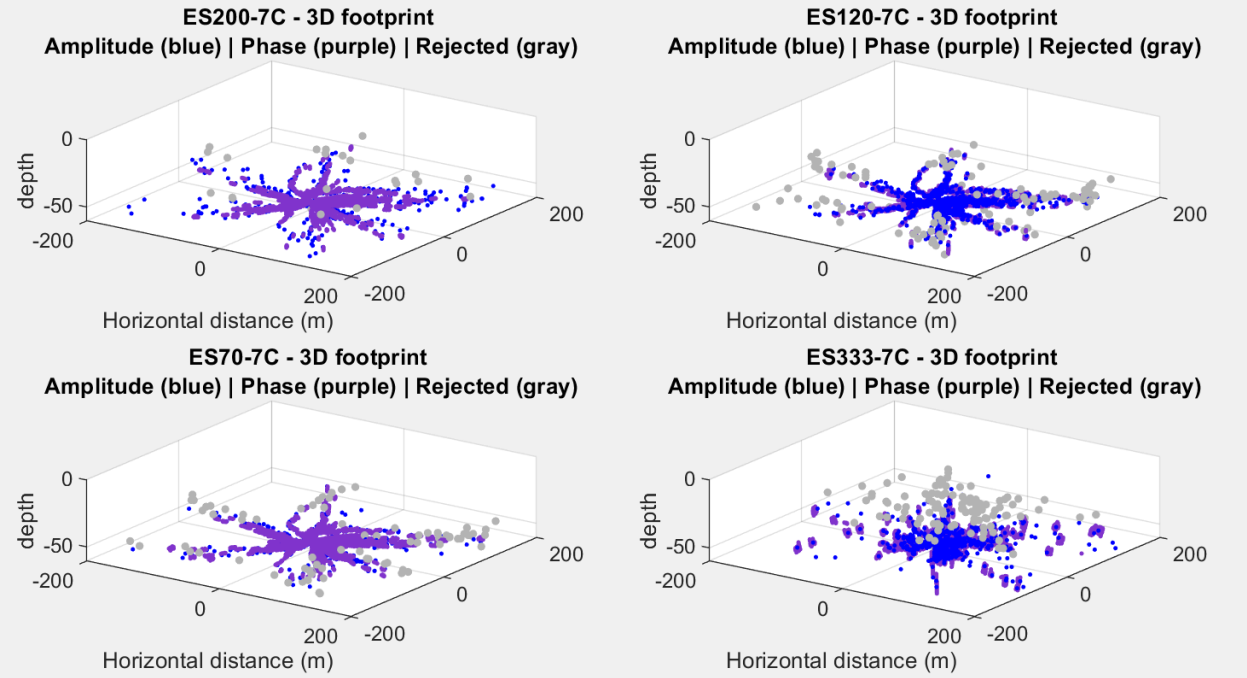


Figure 46 - 3D plot for each Tc with the position in meters of each subset of data according to its grazing angle, range, and azimuth for "Area A", with an average depth of 55m. All rejected pings are in gray

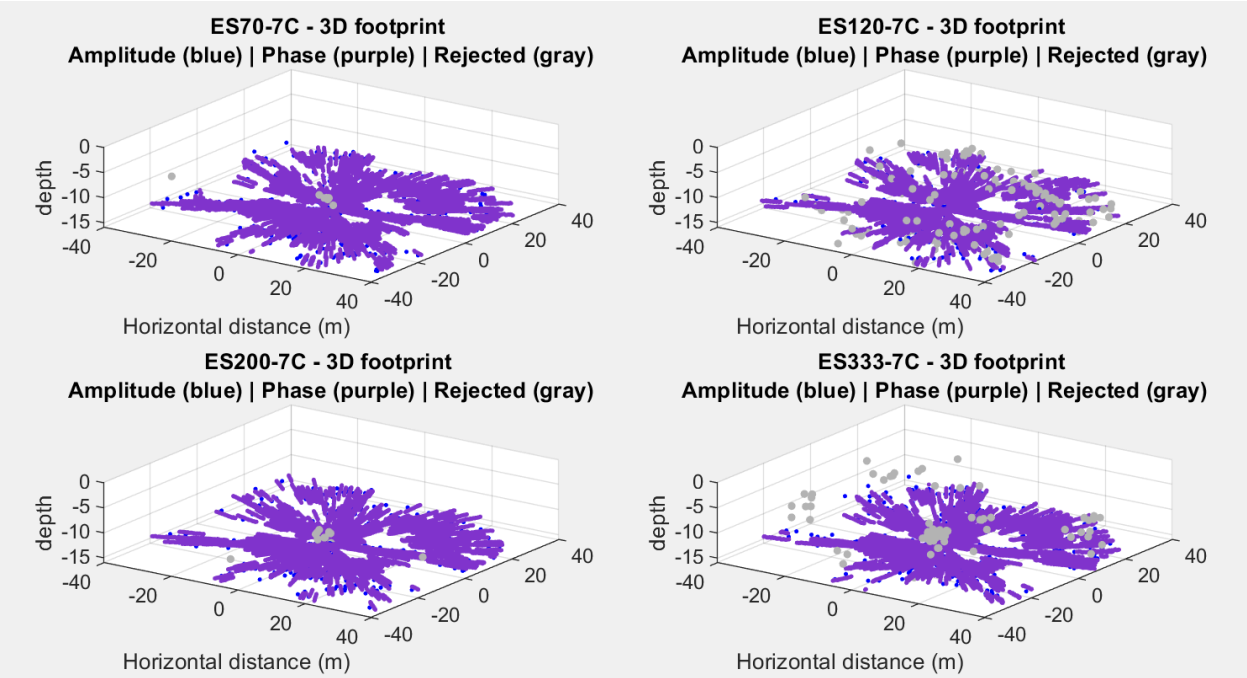


Figure 47 - 3D plot for each Tc with the position in meters of each subset of data according to its grazing angle, range, and azimuth for "Area C", with an average depth of 14m. All rejected pings are in gray

Figure 48 illustrates a zoomed in subset of the figures above which demonstrates that from each ping, by using the phase ramp, rather than just one sounding, several subsets can be acquired that represent more observation points of the seafloor.

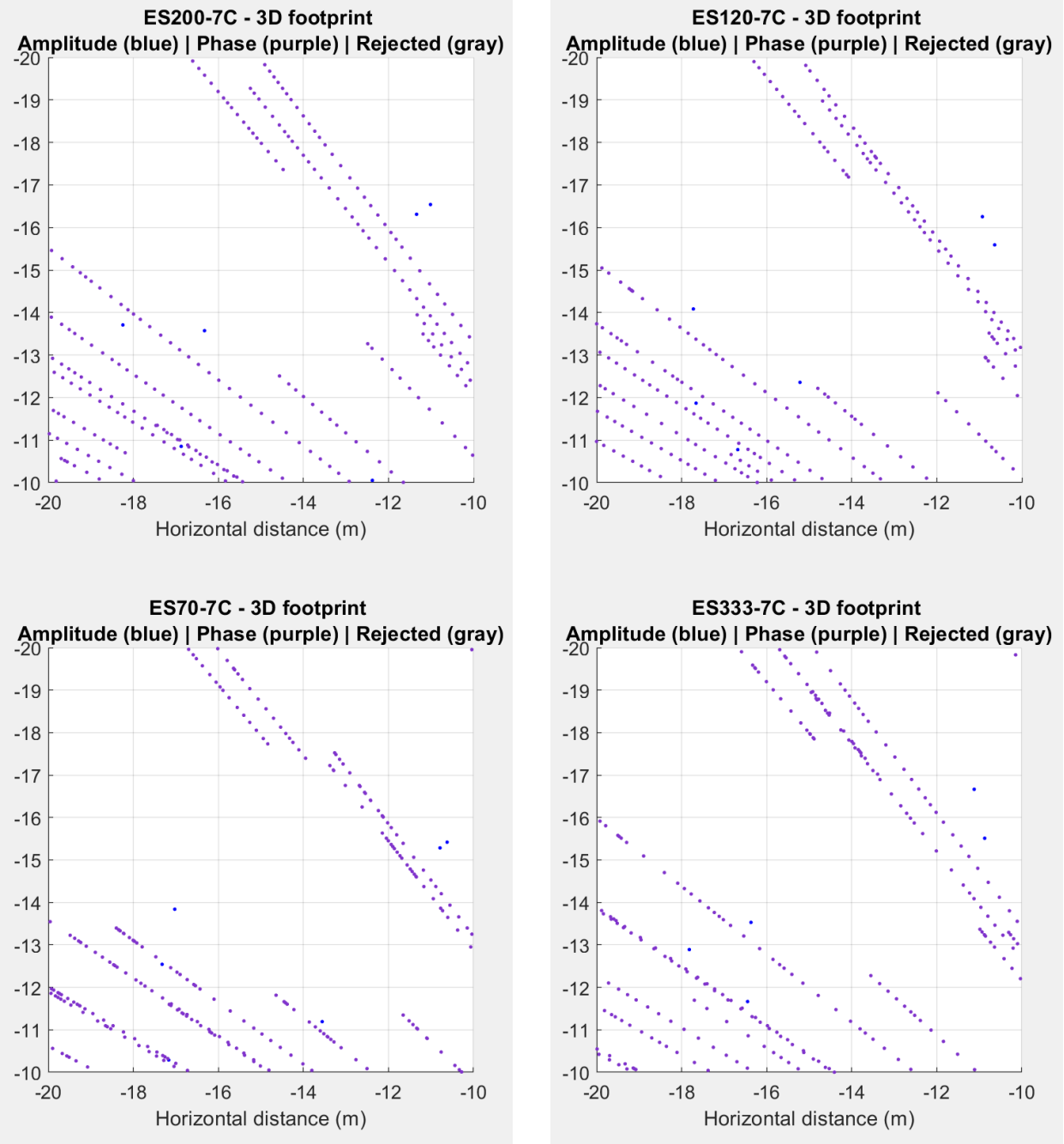


Figure 48 – View from above, on "Area A", zoomed in 10x10m.

Extraction of Echo Envelope from the Echogram

As first recorded, the data is viewed as a conventional echogram with successive traces plotted sequentially. The seabed echo will vary from a minimum slant range equivalent to the local depth, to a maximum of ~ 5x that minimum range at the lowest grazing angles.

As part of the first step in the bottom detection process, a window is extracted around the initial estimate of the location of the center of the beam. As discussed earlier, that window length reflected the approximate expected length of the echo assuming a beam width of ±5 degrees. This was done for memory management reasons. Should the initial detection estimate be faulty (as was often the case at longest ranges with the ES333), the phase information would not indicate the expected ramp and that ping was thus rejected. However, the full echogram can still be reproduced if intended, since the raw data is preserved.

As the plate is rotated, with an increment of roll, the grazing angle gets lower and the range of the seafloor increases. The echograms illustrated in Figure 50 and Figure 49 show the detected bottom of “Area A”, the muddy site 55m deep, with its range varying with the roll, and it is noticeable how much noisier that range estimate gets at larger ranges. Notice that the color bar represents the scale of the MF signal corrected by the range, in dB, and that the transceiver with higher frequencies have weaker signal response, to the point that the ES333-7CD cannot detect the bottom correctly at longer range.

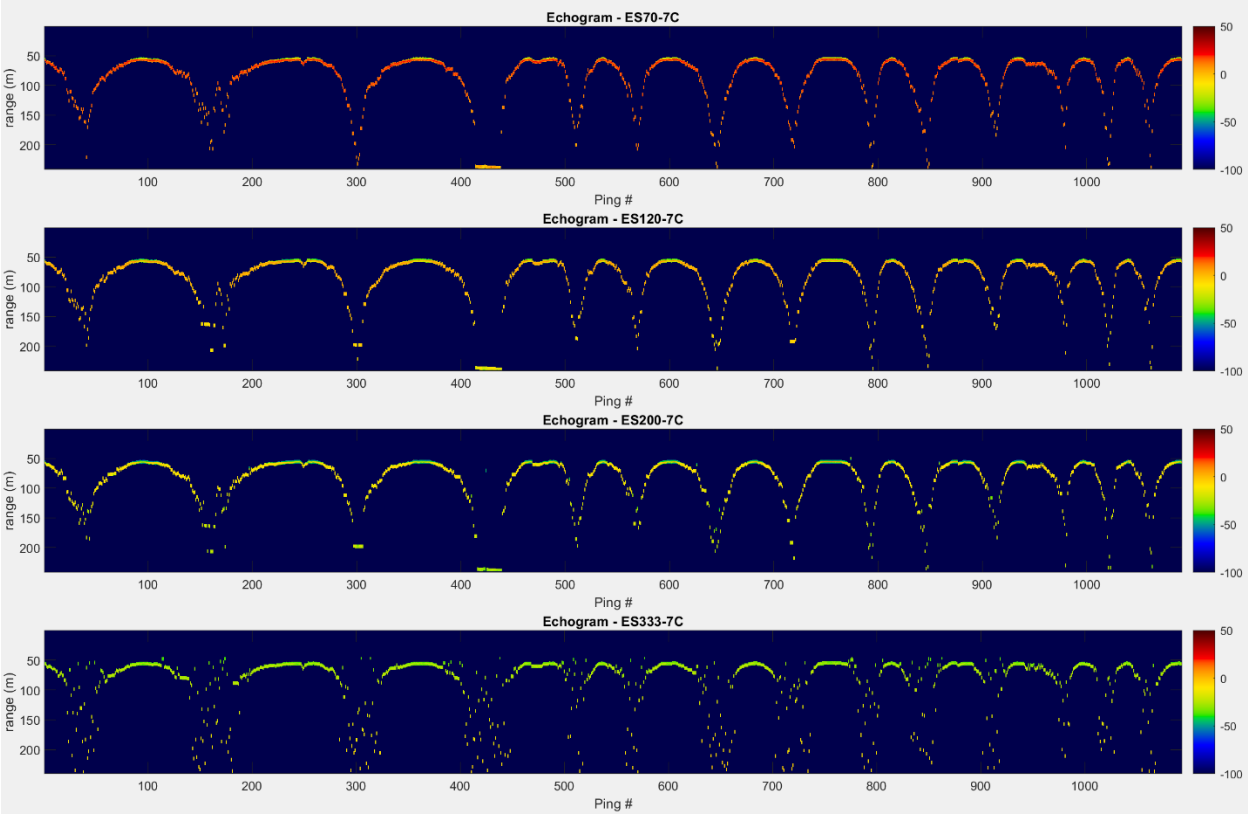


Figure 49 - Echogram of Area A, average depth 55m.

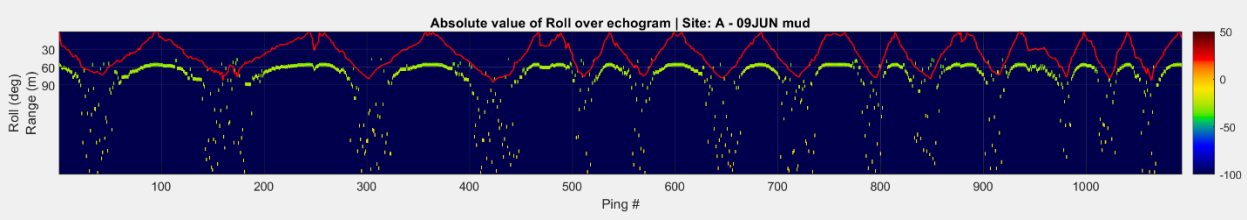


Figure 50 - Echogram of ES333-7CD at Area A with the correspondent Roll in red for each ping. Notice that increasing the roll, the range also increases.

As a contrast, the echograms of Area C, with average depth of 14m, and a muddy sand bottom, are presented in Figure 51 and Figure 52. As can be seen, the bottom is easily detected for all transducers and the ES333-7CD does not show as much noise as in Area A.

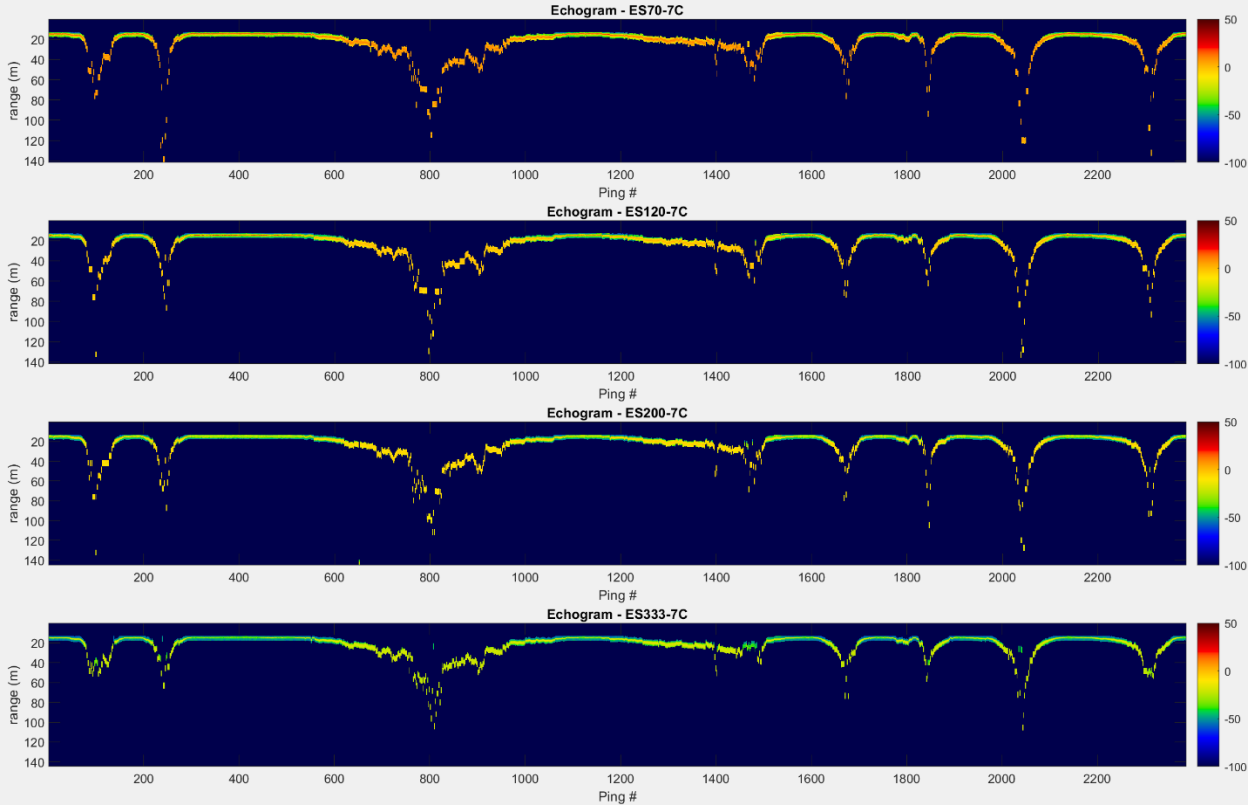


Figure 51 - Echogram of Area C, average depth 14m.

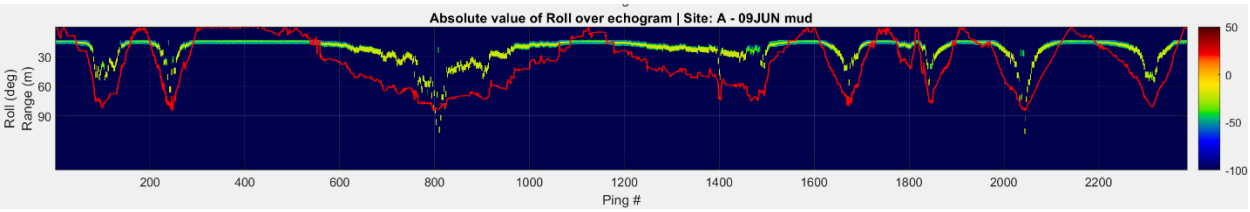


Figure 52 - Echogram of ES333-7CD at Area C with the correspondent Roll in red for each ping. Notice that increasing the roll, the range also increases.

Implementation of each applied correction.

The corrections added to the MF signal that will provide the final absolute Sb response are described below.

- Corrections for Transmission Loss

This can be separated into that component due just to spreading (Figure 53 left), which is not frequency dependent and absorption which is frequency and environment (T and S) dependent (Figure 53 right).

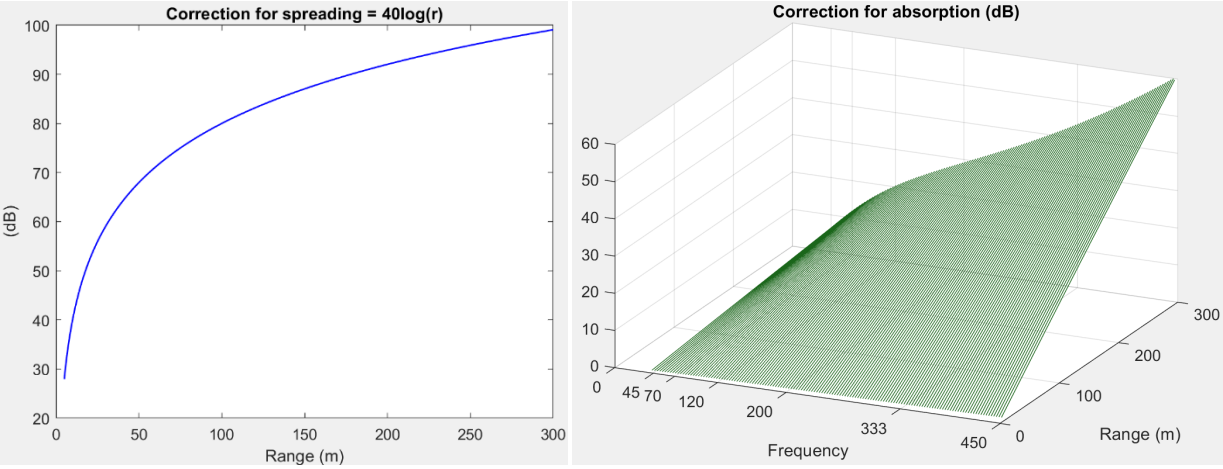


Figure 53 - On the left-hand side, a 2D plot of the losses by spreading with range. On the right-hand side, the 3D plot of absorption loss by frequency and range, with the environmental parameters of "Area A." The other sites would have similar curves for absorption since approximately the same parameters were observed.

- Correction from sphere calibration for MRA.

Based on the procedures outlined in section 4.1 (see Figure 41), calibration corrections for each of the four transducers are applied in frequency to the FFT of the subset of the matched filter output.

- Correction for beam pattern for subsets off the boresight

Based on the previously described calibration the correction for echoes at angles way from the MRA were applied by transducer and frequency. This was used in two ways: to estimate the effective along track beam width, and to correct for the across track signature superimposed response on each of the subsets.

As can be seen in Figure 54, as expected, for each transducer, the beam width narrows with increasing frequency. This should be a smooth variation, but for the ES333 the calibration shows a slight irregularity as noted earlier.

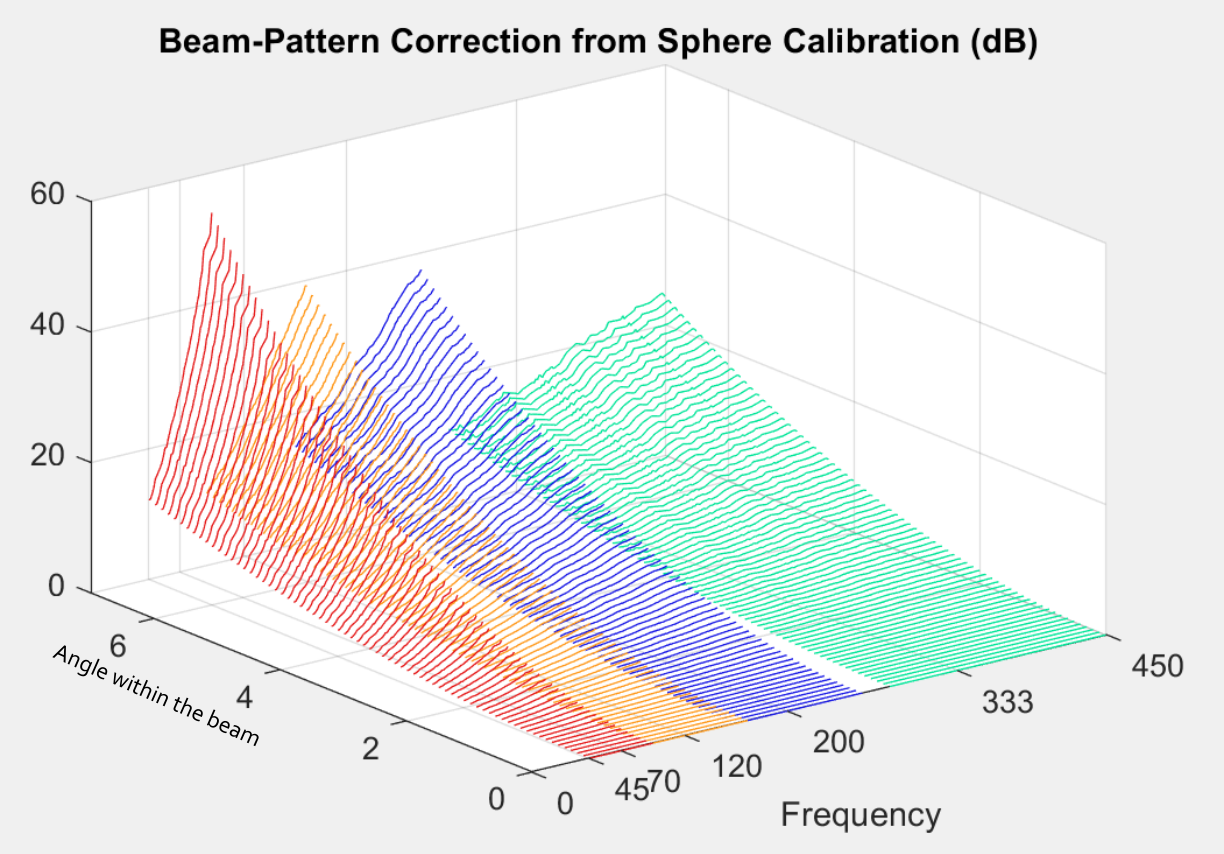


Figure 54 - Beam pattern for each transducer based on the sphere calibration.

Ensonified Area Correction:

The correction for Area is calculated for each subset as a function of frequency dependent beam pattern, grazing angle, range, and FFT length as explained previously. To illustrate the typical frequency and grazing angle dependence, Figure 55 represents a model of the values which was made based on the actual beam pattern obtained by the sphere calibration, but in a flat seafloor 14m deep. Actual values for area correction are noisier since the grazing angle and range are not perfectly correspondent.

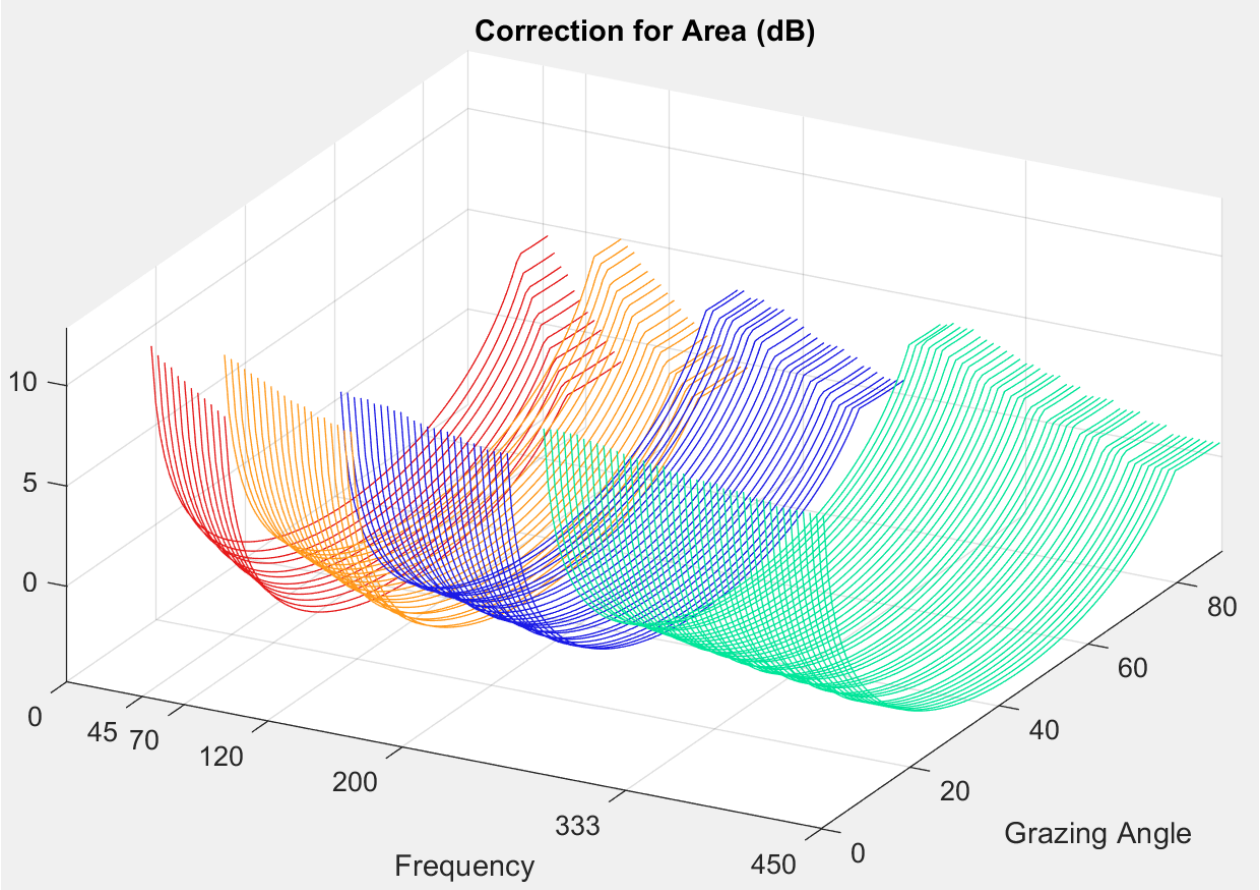


Figure 55 - Values of 10log(A), being "A" the ensonified area, considering the beam pattern calculated on the sphere calibration modeled in a flat bottom with 14m of depth.

Note the abrupt step at high grazing angles. This represents the estimated shift from a beam width limited to a pulse/FFT-length limited ensonified area. And the gradual drop and then rise in the ensonified area illustrates the two competing aspects of the growing along track footprint and the shortening projected FFT length (the width of the annulus). For each transducer, the ensonified area systematically shrinks with frequency as the beam widths (both along and across) shrink.

Comparing response, before and after applied corrections:

For each ping, one or more subsets of the matched filtered output are extracted from around the estimated beam center. Each subset provides a unique spectrum for that transducer and grazing angle uncorrected for the frequency dependent variables (attenuation, beam pattern, calibration, and area normalization). Before binning and averaging, these raw observations include the incoherent speckle. The uncorrected results are illustrated on Figure 56-top and show abrupt transducer to transducer offsets as well as gradual variations within a single transducer reflecting all the frequency dependent terms. The corrections outlined above are then combined and added to those and presented in Figure 56-bottom. If the corrections and calibration are adequate, the resulting data should now show a common trend. The next step is to average these data to provide a mean (and variance) estimate for each frequency and grazing angle bin.

After narrowing all the data in bins and cleaning out the responses that seemed to be incorrect (bottom mis-tracking or low SNR), the plots on Figure 57 are representative of the angular and frequency response curves of the absolute Sb for each site.

These data are now available to use as an absolute reference, against which the backscatter strength estimates from any multibeam sonar can the compared. For a typical narrow band system (most simply a single sector MBES with a CW pulse), the closest corresponding frequency bin can be extracted.

Note that, while five different seabed responses are provided, only one would typically be needed to do the calibration. If the same multibeam were to acquire Sb data over two or more of these sites, the calibration result (the mismatch between the MBES Sb and the reference) should be identical.

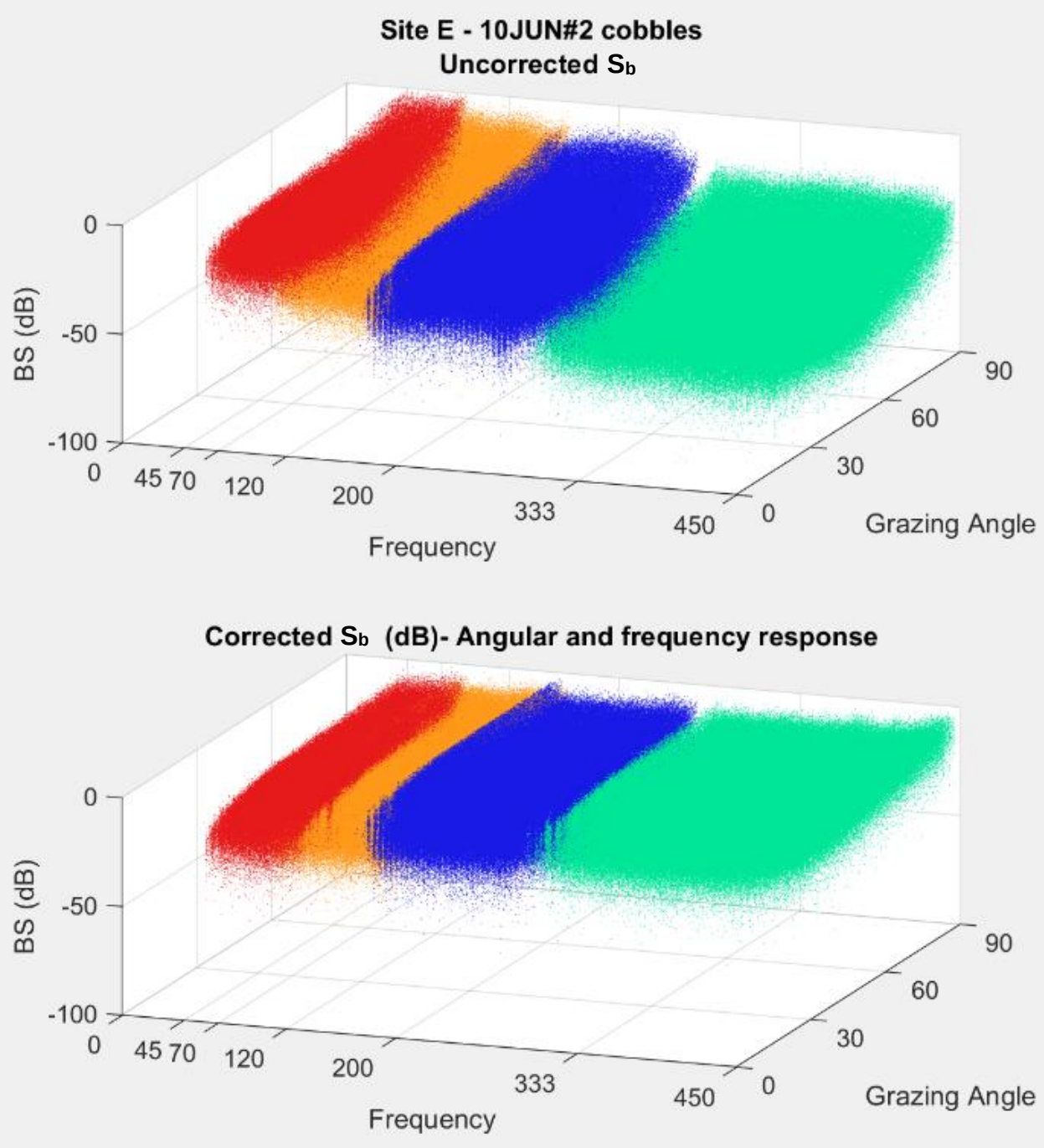
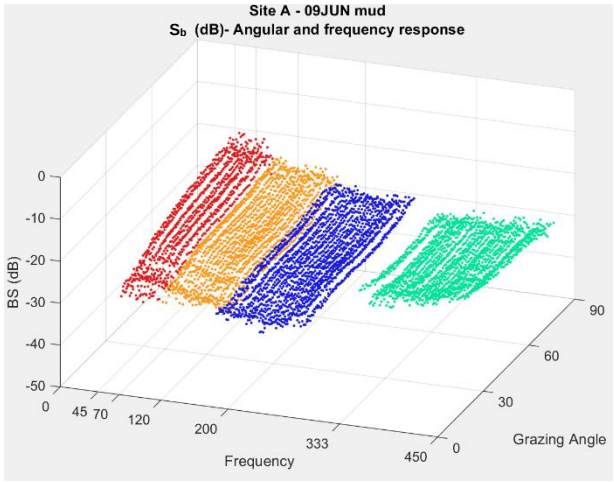
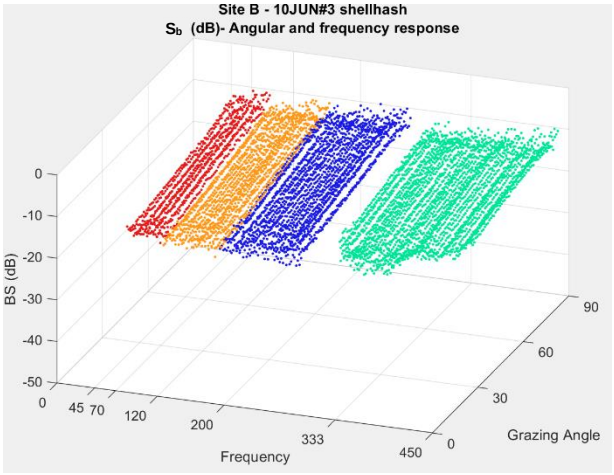
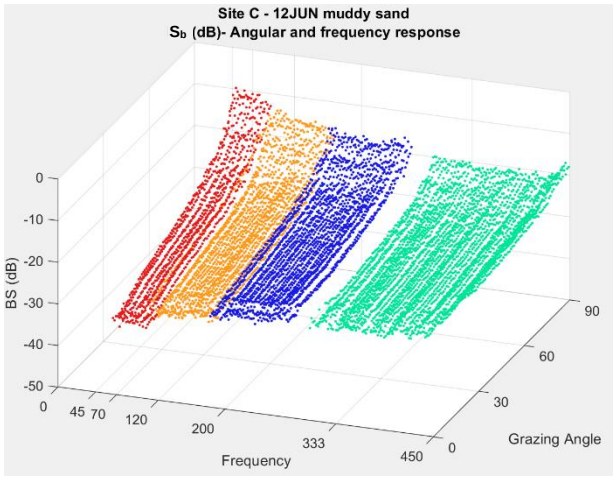
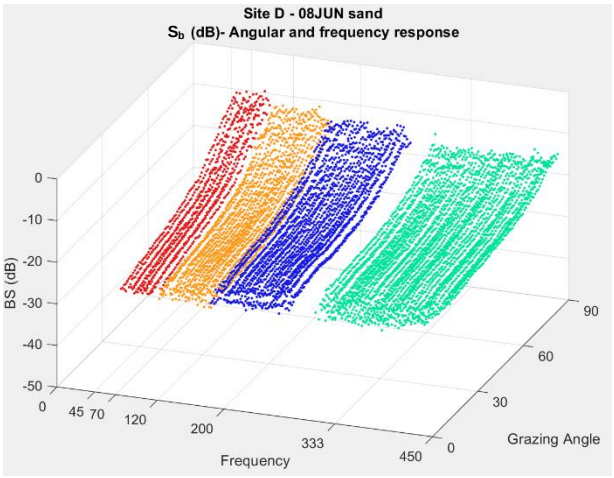


Figure 56 - Cloud of points of BS for each subset within each frequency slot for Area E before and after the corrections.









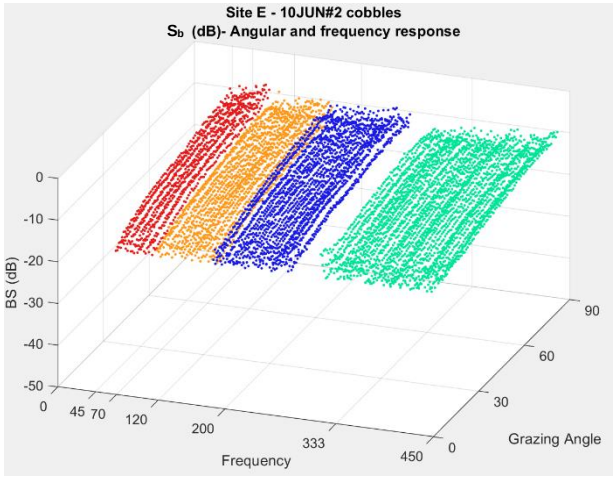


Figure 57 - Absolute BS of each site

4.3. OBSERVATIONS ON SEABED RESULTS

Trends in frequency

Although the main aim of this research is to provide a reference against which a specific multibeam sonar's backscatter can be calibrated, a byproduct was the measurement of the actual seabed response over a remarkably wide range of frequencies. As early as McKinney and Anderson (1964), it was recognized that there was a frequency dependence. Those earliest studies similarly covered a wide range of sediment type (muds to rock) and employed multiple transducers, in that case all calibrated with reference to the TS of a 28-inch sphere. The frequency dependence, however, was only obtained using CW pulses at discrete frequencies (12, 25, 30, 45, 50, 70, 100, 180, 290 kHz) spaced more widely at the upper end of the frequency range. The data acquired herein, utilizing FM pulses, allows a continuous assessment of that frequency dependence. And a similarly wide range of sediment types were examined (mud to cobbles). Grain size estimation is currently only by eye (due to COVID closure of grain size laboratory facilities). Nevertheless, notable trends exist and are reported here.

From the data stored in bins, it is possible to create other plots to better describe the characteristics within the data. The trend in frequency can be seen at different grazing angles for each site. Figure 58 illustrates those trends at four representative grazing angles.

Superimposed over the average frequency trend is an apparent rapid fluctuation. While this could be inherent noise in the data, the high level of averaging allowed through the use of subsets, would suggest another cause. Figure 59 illustrates the use of smoothing applied to each transducer separately. One can see that, although offset, each of the five sediment types exhibit a near identical rapid variation with frequency. This suggests that imperfections in the calibration rather than a physical phenomenon.

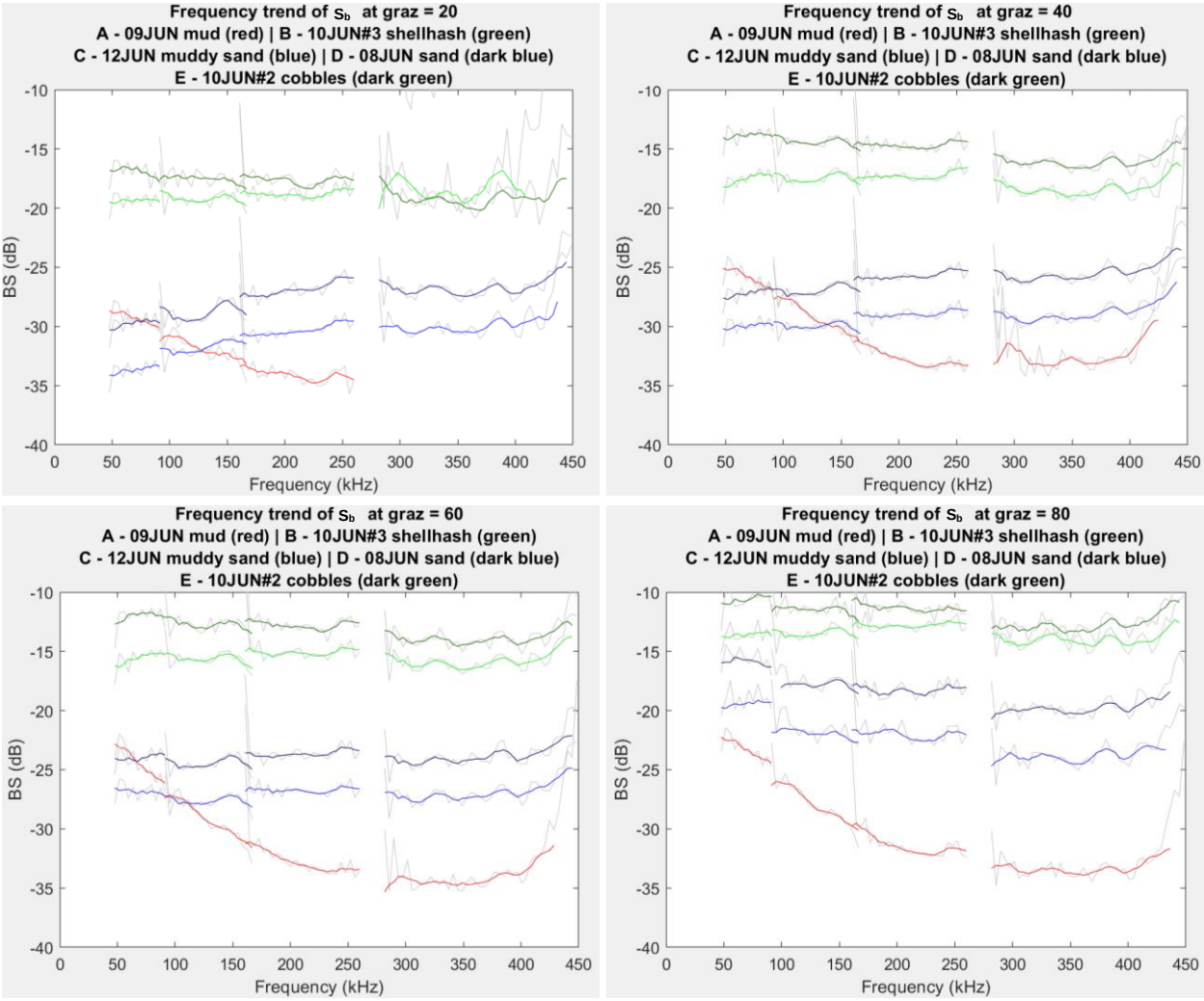


Figure 58 - Frequency trend of BS for all sites at grazing angles of (rotating clockwise)20, 40, 60 and 80 degrees. The original data from the bins are in gray and the smoothed response are colored.

Additionally, between each transducer's results there appear slight offsets, again indicating a slight calibration remnant. To address this, in Figure 59, a curve fit is applied across all transducer results. This allows the visualization of trend (and permits interpolation across the ES200-ES333 frequency gap).

A third aspect that needs to be addressed is the high noise content in the muddy sediment results above 200 kHz. As discussed previously this is an unavoidable result of the highest slant ranges due to the 55m transducer elevation, combined with the lowest seabed backscatter strength measurements. To deal with this, the mud curve has been manually culled at the lower grazing angles.

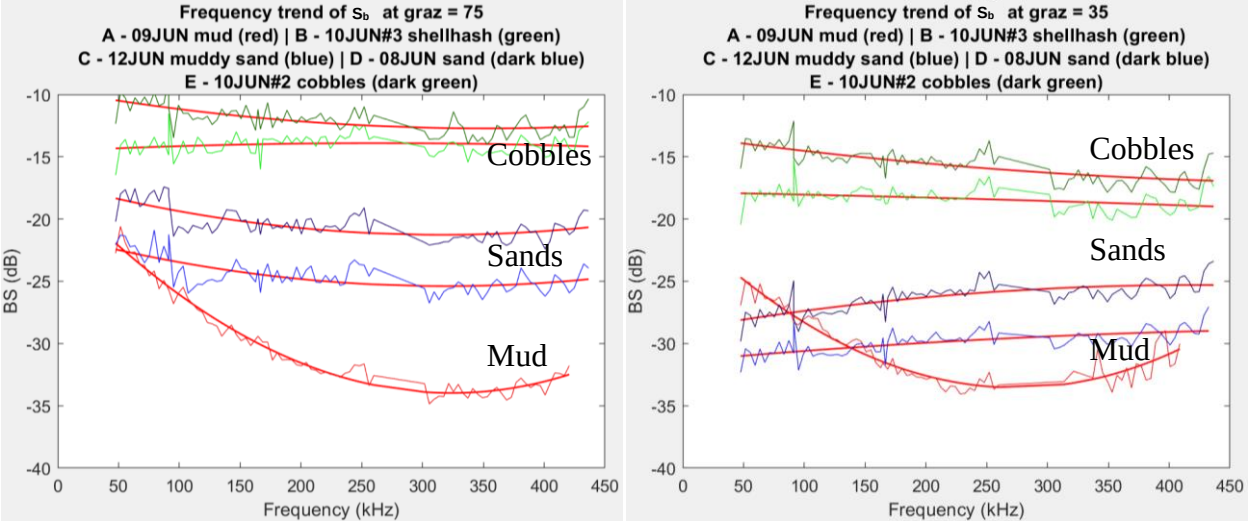


Figure 59 - Frequency trend at grazing angles of 75º (left) and 35º (right) with fitting curves to better visualize the trend.

Figure 59 uses cleaned data to determine a curve fitting to the frequency response across the full 45-400 kHz range. To illustrate the changing frequency trends with grazing angles two representative grazing angles of 75 and 35 degrees are used as illustration. Three distinct frequency trends are most noticeable, associated with the coarser/higher impedance sediments (Areas B and E), the sandy sediments (areas C and D) and the muddy site (Area A).

Whilst the muddy site preserves the same strong drop in Sb with frequency at different grazing angles (the reversal in the trend is probably suspect above ~ 200 kHz), the frequency trend in the sandy sites change significantly, decreasing with frequency at the higher grazing angle to increasing with frequency at lower grazing angles. Finally, the coarsest sites show the flattest frequency response at both grazing angles, exhibiting just a slightly drop with frequency, with different slope at different grazing angles. For all sediment types, the frequency dependence notably flattens with increasing frequency.

The frequency trends seen herein are similar to those reported previously although few high grazing angle comparisons are available. For example, for sands, McKinney and Anderson (1964) saw a positive trend with frequency at lower (10˚) grazing angle for. It is harder to compare these results with the 10-30˚ data from four sandy sites performed in the 1980s (Boehme et al., 1985, Boehme et al., 1988, Stanic et al., 1988, Stanic et al., 1989) as the frequency trends there were weak and the individual frequency results showed no consistent trend. For the coarser sediments, McKinney and Anderson reported an almost flat response for solid rock and Jackson et al. (1986b) show a weak positive trend with frequency.

All these previous results, however, involved sparse frequency sampling and rarely addressed near normal incidence. The results reported herein, have a much more continuous view of the frequency dependence and always extend to normal incidence. As such, these results may guide future efforts in multi-spectral backscatter classification. To date these efforts (Hughes Clarke, 2015, Brown et al., 2019, Gaida et al., 2020) have taken advantage of available bandwidth in shelf-mapping sonars. As the frequency dependence generally is weakest at the highest frequencies, less information is probably gained by collecting 200 v. 400 kHz rather than <=100 kHz v. ~300 kHz.

Trends in angle

While the frequency dependence is illuminating, most multibeam systems only operate within a narrow frequency band. Within that band, the calibration has to be undertaken over the full range of grazing angles. If properly calibrated, the shape of that angular response is the primary output of the mapping system and it can directly be used to attempt sediment classification (Fonseca et al., 2007). Additionally, however, for the purpose of mosaic (plan view spatial maps of backscatter) construction one of the main required image processing steps is to flatten the angular response. Any variation in that shape with sediment type complicates this. To help the reader visualize the variation in that curve shape as a function of sediment type and frequency, Figure 60 presents the classical visualization of the angular response curve for three different frequencies for each transducer.

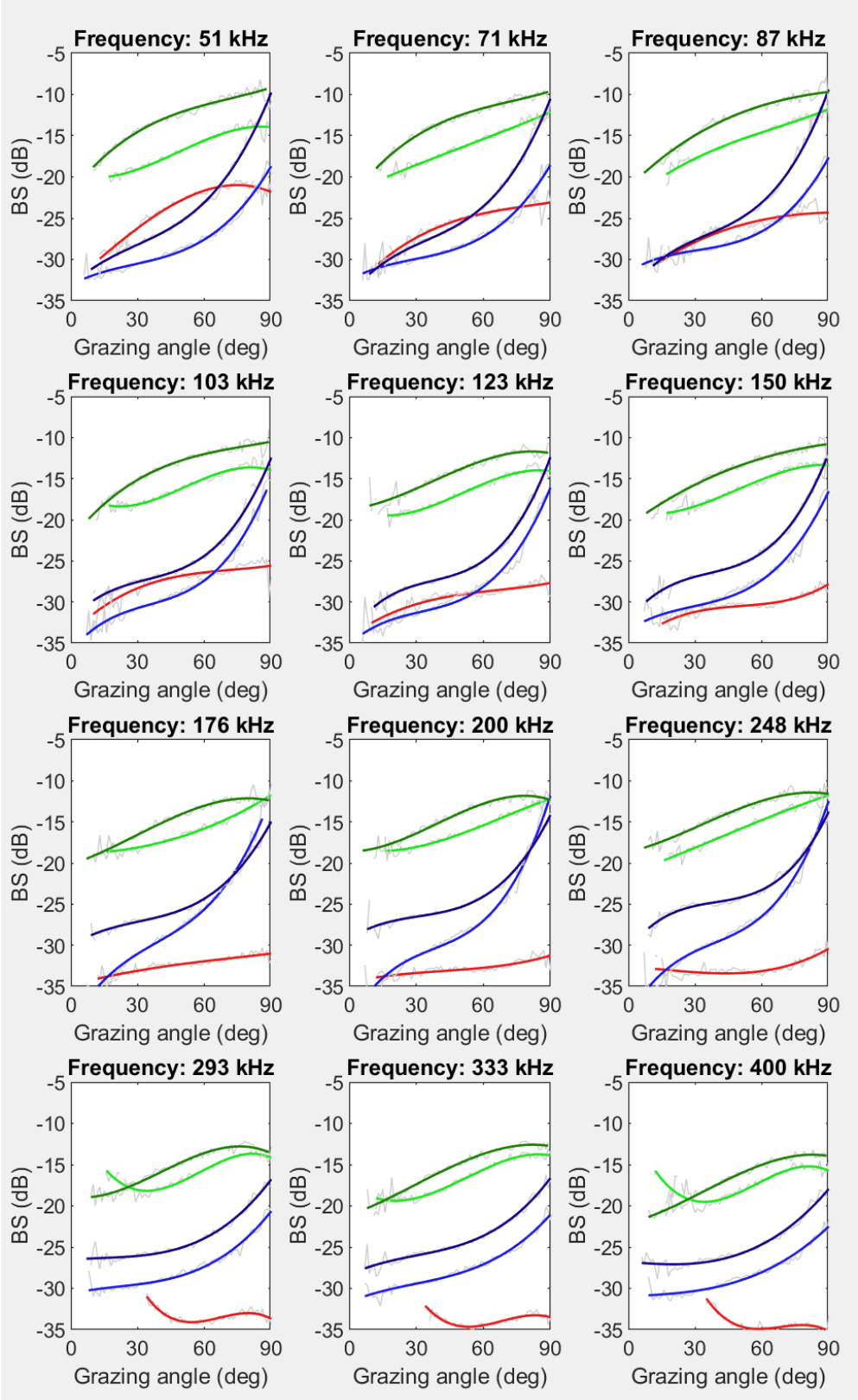


Figure 60 - Angular response curves of different frequencies with color code for Areas: A - Mud (red), B - Shell hash (green), C - Muddy Sand (blue), D - Sand (dark blue), and E - Cobbles (dark green)

Depending on the frequency, the angular response of two different sites can vary between similar to strongly contrasting. Take for example, the response of the sites Area C ("Muddy Sand") and D ("Sand") to observe how similar their curves are at 123kHz and how different they become at 200 kHz.

In Figure 60, a curve fit is attempted to illustrate the trends. As can be seen, however, above 200 kHz for the mud curves where noise is contaminating the lowest grazing angles, the rise in Sb at the lowest grazing angles is almost definitely an artefact. Similarly, the shell hash curves as seen through the ES333 transducer, even though a stronger average Sb, has problems at the lowest grazing angles as the site was deep (47m).

Investigating the ripple artifact in the frequency trend

While the general frequency trends are clear, superimposed on top of all the curves is a small magnitude (generally < +/-2dB) rapid variation in frequency that appears systematic as it is present largely irrespective of sediment type and grazing angle. In order to analyze it, that trend has been extracted and is presented in figures 61 and 62.

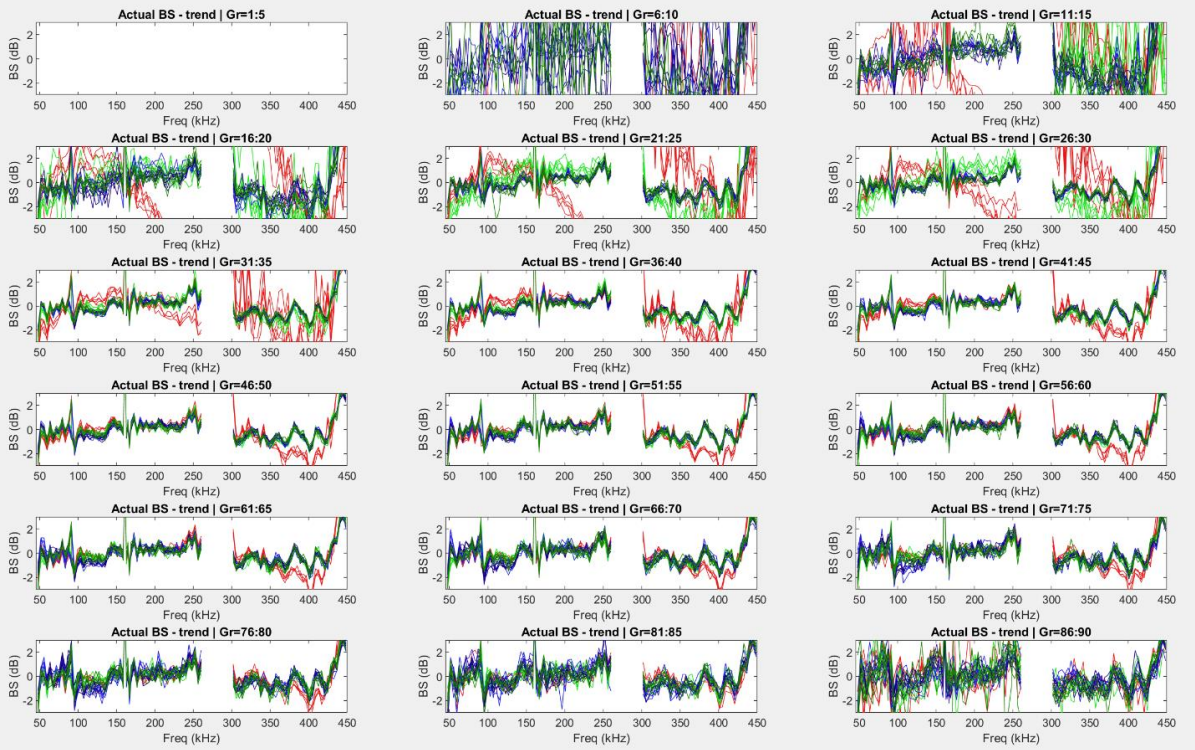


Figure 61 - Difference of the bottom response to the fitting curve for the frequency trend at steps of 5º of grazing angle.

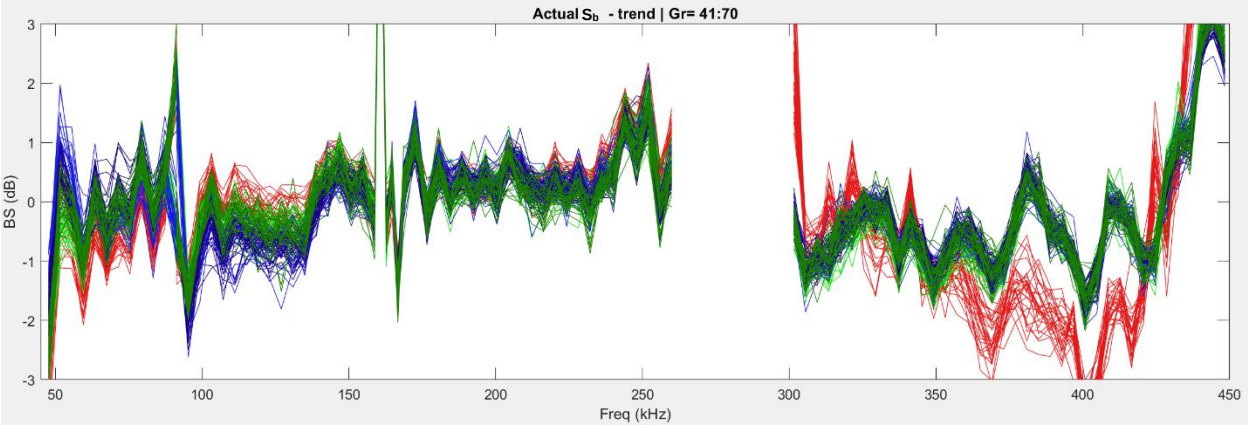


Figure 62 - Difference of the bottom response to the fitting curve for the frequency trend for grazing angle from 41º-70º.

The extracted residuals are derived by subtracting a smooth curve fit to the 45-450 kHz trend in Sb within a narrow range of grazing angles from the corresponding observations. As can be seen, irrespective of either the grazing angle or the sediment type (all 5 sediment type data are plotted here), the frequency trend always shows the same rapid fluctuations with frequency. This strongly suggests that this is a result of imperfect compensation for frequency dependent transducer calibrations rather than a seafloor response. To investigate this, two aspects were considered, MRA calibration and beam pattern response.

As noted in the sphere calibration, the apparent MRA corrections for all transducers except the 200 kHz exhibited a characteristic ripple on the outer edges of their bandwidth (see figure 38). One possible explanation for the residuals is that the normalization of the sphere response by the magnitude of the autocorrelation of this design pulse (see formula at end of section 3.5) was not applied correctly. The attempt at compensating for this by smoothing (see figure 39) was only partially successful. To test this hypothesis, the difference between the smoothed and original calibration was derived (Figure 63) and then plotted against the residuals (Figure 64). This reveals that there is indeed some correlation.

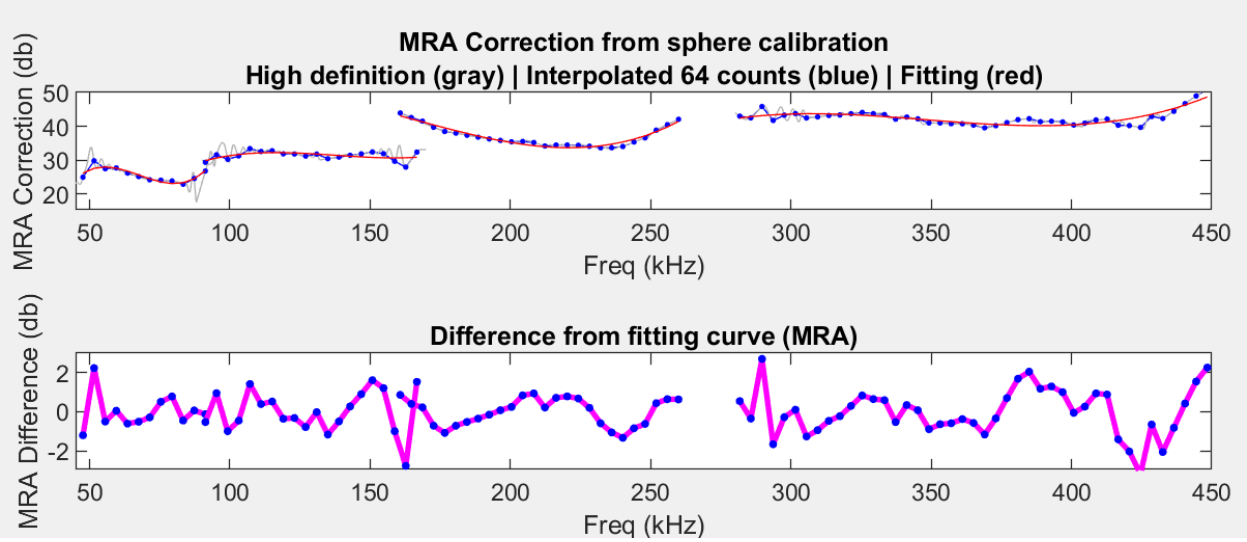


Figure 63 - Difference of the sphere calibration response to the fitting curve in the frequency domain.

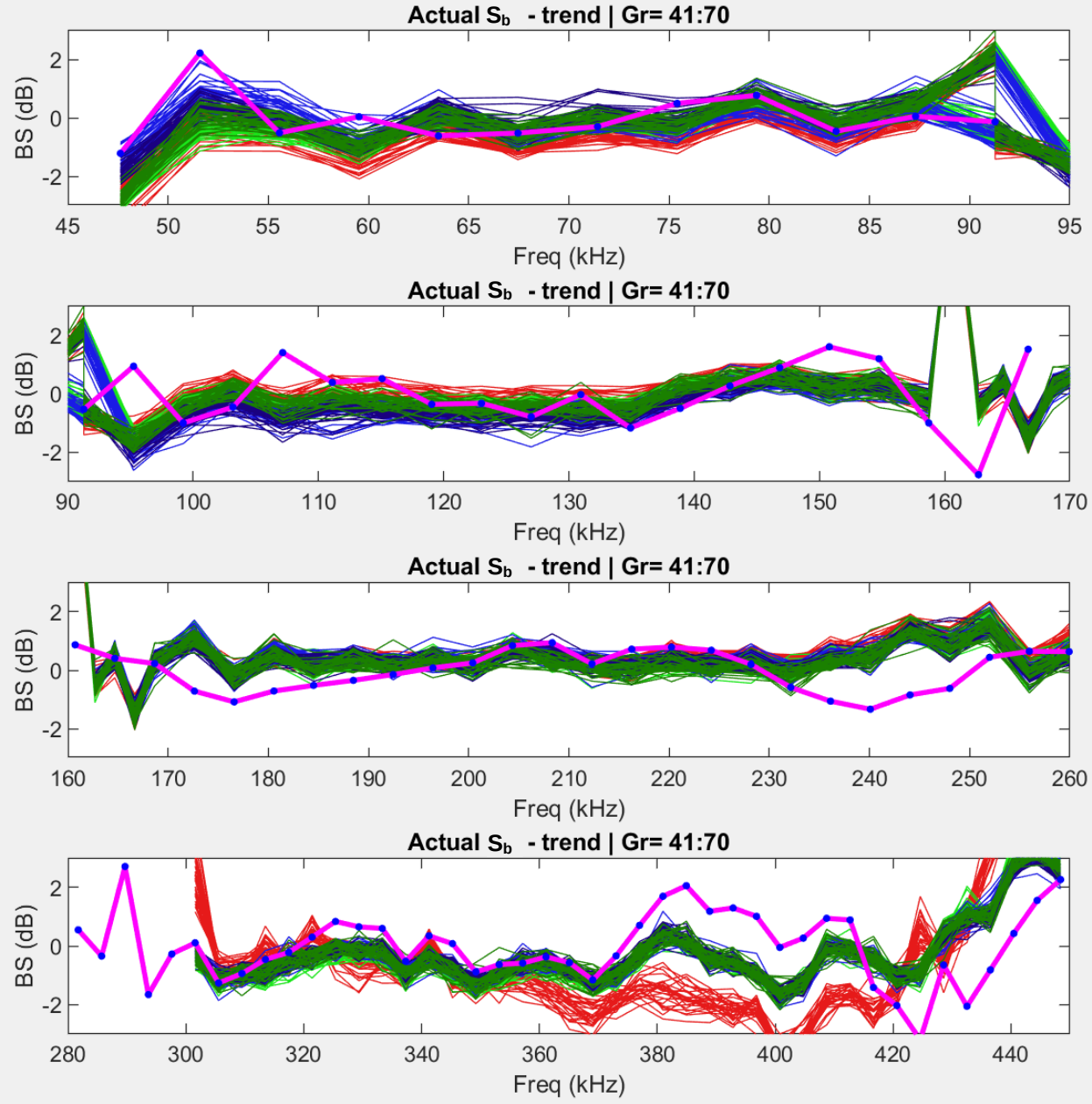


Figure 64 - Comparison between the bias presented on the bottom response (color coded by each area) with the bias inherited from the sphere calibration (magenta).

The alternate possibility is that the residuals come from imperfect beam pattern compensation which should vary only smoothly with frequency. Figure 65, however, illustrates the difference between the applied beam pattern corrections and a smoothed fit. The residuals do not appear either significant or correlated with the seafloor residuals suggesting that this is not the source of the problem.

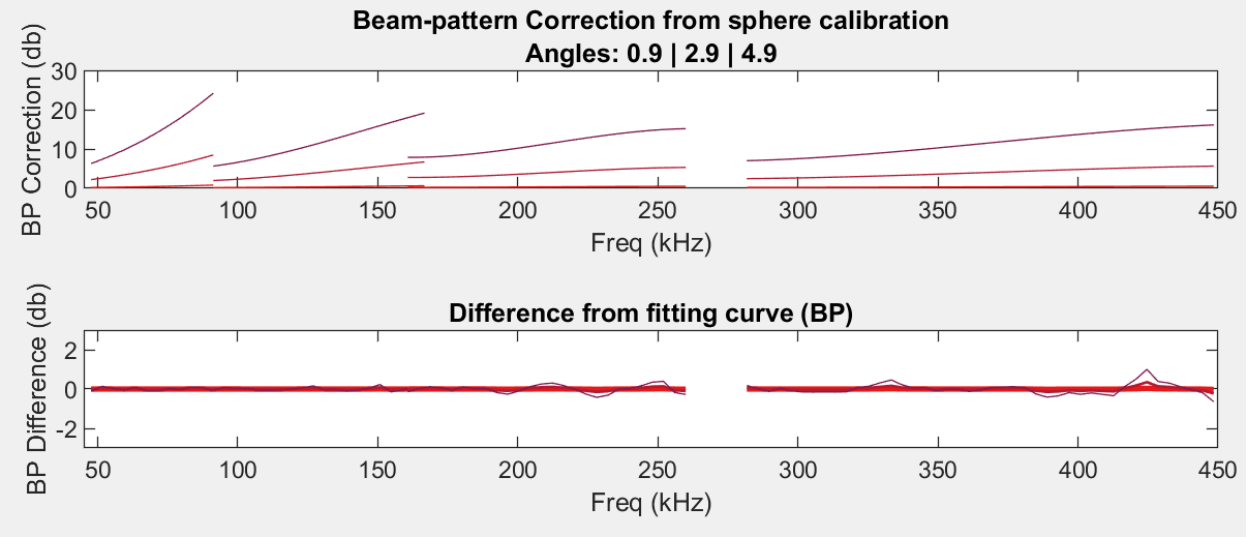


Figure 65 – Minimal difference of the beam pattern response to the fitting curve in the frequency

Although the ripple in the sphere MRA calibration does not match exactly the ripples in the final bottom response (Figure 64), there are some considerable similarities suggesting that this might be the source of the problem. This may be exacerbated by the mismatch in the size of FFT length for the sphere calibration and the seafloor subset. One possible solution, not tested, to avoid this effect might be using the same FFT length for the sphere calibration as for the subsets of the bottom. In summary, a deeper investigation is clearly needed to properly address what might be causing these residual ripples.

5. CONCLUSION

The method presented in this thesis is based on an extension of previous techniques that have been developed progressively over time since the 1950s in an attempt to acquire absolute backscatter response from the seafloor.

As part of this thesis, a new method has been developed and tested for acquiring calibrated seabed backscatter strength over a wide range of frequencies (45-450 kHz) the full range of grazing angles (90 to < 5) for a wide range of seabed types (muds to cobbles). These data were collected for the purpose of providing a reference data set, against which future backscatter observation from multibeam sonar systems can be cross-calibrated.

To achieve the wide total bandwidth, the experiment employed four broad band split beam transducers and transceivers, each of which utilized FM pulses that swept through the available bandwidth within each transducer. Each transducer was calibrated against the modelled target strength of a precisely machined sphere in the standard manner reported by Foote et al. (1987). The sphere calibration provides the beam pattern and the corrections for the Maximum Response Axis (MRA) as a function of frequency for each transducer.

The transducers were rigidly mounted on a suspended plate with a co-located orientation sensor. The plate was then deployed over a homogenous area of the seabed and the transducers activated in turn while being manually rotated over the full range of grazing and azimuth angles multiple times to have a good statistical collection of seabed backscatter intensity observations.

The data collected is processed applying the sphere calibration parameters along with the compensations of the Transmission Losses (TL) and Ensonified Area to produce the reference data of the bottom of each site.

The approach implemented was an extension and combination of the FM work of Weber and Ward (2015), and the mechanically rotated CW approach of Eleftherakis et al., (2018). A notable difference in the deployment geometry was the use of an anchored, rather than moving vessel, allowing simpler manual rotations. Successful application of this approach relies on the chosen site being suitably stable in time and homogenous in space.

Sites were chosen based on archived multibeam bathymetry and backscatter data provided by the CHS. This was reprocessed and analyzed to identify suitable sites. To confirm the stability of the sites, reconnaissance EM2040 and EM710 bathymetry and backscatter data were acquired over the immediate calibration area. These could be utilized in a follow-on project to attempt full calibration.

The net result of the described processing sequence was a measurement of the absolute bottom Sb response over a wide range of frequencies and grazing angles, suitable for use as a reference for cross calibration. The next step would be to collect the backscatter response of the same seafloor with an MBES, apply the compensations for TL and ensonified area and compare with the reference in the range of frequency operated by the MBES. The difference between the response obtained by the uncalibrated system and the reference will be the calibration parameters for such system.

Although this thesis presented references for five different sites, a practical operating method would neither require multiple areas nor a known seafloor type as long as it is demonstrated to be homogeneous. Rather, a strategic analysis conducted by the interested survey institute would be crucial to determine how many and where the sites should be to benefit most for the calibration of the sonar systems of the fleet.

The method seems to be reliable as four independent transducers had applied independent sphere calibration parameters and produced final results that revealed a smooth and continuous transition on the edge of the frequency range of each transducer in comparison to its neighbor indicating that the response is mostly consistent.

Recognized Limitations

While the approach did produce usable results, some weaknesses were identified in the applied method that would benefit from future analysis. Additionally, a number of operational limitations were recognized that might be improved upon in future deployments.

Weaknesses and limitations include:

Imperfect handling of rapid frequency oscillations in the calibration.

Driven by the need for a long FTT window for the calibration and the desire for a shorter FFT window for the seabed data processing, the mismatch in spectral resolution between the calibration and seabed data appears to have resulted in a small (~±1 dB) systematic ripple in the frequency trend in all the data collected.

Sub-optimal choice of calibration sphere for the 333 kHz transducer

This resulted in both calibration offsets and imperfect beam pattern measurement. It is believed to be a result of the high level of nulls and sloped inter-nulls in the modelled TS response of the 38.1 mm WC sphere. In hindsight, a smaller diameter sphere might have improved this.

Limiting SNR for the altitudes employed.

As part of the search for suitable sites, depths ranged from 15 to 55m. Subsequent analysis of the data indicate that for the lowest grazing angles and lowest backscatter sediment types, the data were compromised. Further investments in this method should consider the possibility of developing a plate that can be deployed closer to the seafloor both to avoid excessive TL and to narrow the required area to be homogeneous, mostly in areas with soft sediment type.

Imperfect calculation of the ensonified area.

The typical simplifications used in narrower beam multibeam ensonified area calculations was employed herein. These are most sensitive at the highest grazing angles as the transition from pulse to beam-width limited area occurs. Apparent step-like features in the observed angular response in these regions suggest that a more sophisticated calculation of the true arcuate ensonified area might be useful. This is probably compounded by the difficulty of obtaining precise grazing angles at these near-normal-incidence geometries.

Imperfect measurement of grazing angle near normal incidence

Because of the limitations in phase detection near normal incidence, grazing angles under those geometries were calculated using just the transducer frame orientation. While this does allow full correction for both pitch and roll rotations, it does not account for seafloor slope. Inversely, for the lower grazing angle data, which utilized phase data, while the across-track slope is well defined (and inherently included the refracted ray path), it fails to properly account for any along track slope.

An additional complication is that near nadir, the instantaneous ensonified area occupies a wide range of grazing angles (~5 degrees) both along and across-track, thereby unavoidably smearing the angular responses within the beam width. A higher definition is only possible for angles of incidence at which the echo envelope is long enough to distinguish angles within the phase that permit the creation of multiple subsets.

6. LIST OF REFERENCES

Boehme, H., Chotiros, N. P., Rolleigh, L. D., Pitt, S. P., Garcia, A. L., Goldsberry, T. G., and Lamb, R. A. (1985) "Acoustic backscattering at low grazing Angles from the ocean bottom. Part I. Bottom backscattering strength." J. Acoust. Soc. Am. 77, 962-979

Boehme, H. and Chotiros, N.P. (1988) "Acoustic Backscattering at low grazing angles from the ocean bottom." J. Acoust. Soc. Am. 84, 1018, doi: 10.1121/1.396738

Brown, C.J.; Beaudoin, J.; Brissette, M.; Gazzola, V. (2019) "Multispectral Multibeam Echo Sounder Backscatter as a Tool for Improved Seafloor Characterization." Geosciences, 9, 126.

Briggs, K. B. (1989) "Micro topographical roughness of shallow-water continental shelves." IEEE J. Oceanic Eng. 14, 360-367.

Briggs, K. B., Reed, A. H., Jackson, D. R. and Tang, D. (2010) "Fine-Scale Volume Heterogeneity in a Mixed Sand/Mud Sediment off Fort Walton Beach, FL." in IEEE Journal of Oceanic Engineering, vol. 35, no. 3, pp. 471-487, July, doi: 10.1109/JOE.2010.2041834.

Carvalho, R. and Hughes Clarke, J. E. (2012) "Proper environmental reduction for attenuation for multi-sectors sonars." Proceedings of the Canadian Hydrographic Conference, 15pp.

Demer, D.A., Berger, L., Bernasconi, M., Bethke, E., Boswell, K., Chu, D., Domokos, R., et al. (2015) "Calibration of acoustic instruments." ICES Cooperative Research Report No. 326. 136 pp.

Eleftherakis, D., Berger, L., Le Bouffant, N. et al. (2018) "Backscatter calibration of high-frequency multibeam echosounder using a reference single-beam system, on natural seafloor." Mar Geophys Res 39, 55–73. https://doi.org/10.1007/s11001-018-9348-5

Fonseca, L. and Mayer L., (2007) “Remote Estimation of Surficial Seafloor Properties through the Application Angular Range Analysis to Multibeam Sonar Data.” Marine Geophysical Researches, v. 28, p.119–126

Foote, K. G., Knudsen, H. P., Vestnes, G., MacLennan, D. N., and Simmonds, E. J., (1987) “Calibration of acoustic instruments for fish density estimation: A practical guide.” International Council for the Exploration of the Sea. Cooperative Research Report n^{o} 144, 1-69.

Foote, K. G., Chu, D., Hammar, T. R., Baldwin, K. C., Mayer, L. A., Hufnagle, L. C., and Jech, J. M. (2005) “Protocols for calibrating multibeam sonar.” Journal of the Acoustical Society of America, 117: 2013–2027.

Gaida, T.C.; Mohammadloo, T.H.; Snellen, M.; Simons, D.G. (2020) “Mapping the Seabed and Shallow Subsurface with Multi-Frequency Multibeam Echosounders.” Remote Sens. 12, 52.

Gensane, M. (1989) “A statistical study of acoustic signals backscattered from the sea bottom.” in /IEEE Journal of Oceanic Engineering/, vol. 14, no. 1, pp. 84-93, Jan., doi: 10.1109/48.16818.

Hughes Clarke, J.E., Iwanowska, K.K., Parrott, R., Duffy, G., Lamplugh, M. and Griffin, J., (2008) “Inter-calibrating multi-source, multi-platform backscatter data sets to assist in compiling regional sediment type maps: Bay of Fundy.” Proceedings of the Canadian Hydrographic Conference and National Surveyors Conference, Victoria, BC. Paper 8-2, 22pp.

Hughes Clarke, J.E. (2015) “Multispectral acoustic backscatter from multibeam, improved classification potential.” In Proceedings of the United States Hydrographic Conference, National Harbor, MD, USA, 15–19 March

ICES (2007) “Acoustic seabed classification of marine physical and biological landscapes.” ICES Cooperative Research Report No. 286. 183 pp.

Jackson, D. R., Winebrenner, D. P., and Ishimaru, A. (1986a). “Application of the composite roughness model to high-frequency bottom backscattering.” J. Acoust. Soc. Am. 79(5), 1410–1422.

Jackson, D. R., Baird, A. M., Crisp, J. J., and Thomson, P. A. (1986b). “High-frequency bottom backscatter measurements in shallow water.” J. Acoust. Soc. Am. 80(4), 1188–1199.

Jackson, D. R. & Briggs, K. (1992) “High-frequency bottom backscattering: Roughness versus sediment volume scattering.” Journal of The Acoustical Society of America - J ACOUST SOC AMER. 92. 962-977. 10.1121/1.403966.

Kenny AJ, Cato I, Desprez M, Fader G, Schüttenhelm RTE, Side J (2003) “An overview of seabed-mapping technologies in the context of marine habitat classification.” ICES J Mar Sci 60:411–418

Lamarche, G., Lurton, X. (2018) “Recommendations for improved and coherent acquisition and processing of backscatter data from seafloor-mapping sonars.” Mar Geophys Res 39, 5–22. https://doi.org/10.1007/s11001-017-9315-6

Lanzoni, J. C., (2011) “Field calibration methodology for a multibeam echo sounder using a split beam sonar system and a standard target.” Master's Thesis and Capstones. 687. https://scholars.unh.edu/thesis/687

Lanzoni, J. C., and Weber, T., (2011) “A method for field calibration of a multibeam echo sounder.” OCEANS'11 MTS/IEEE KONA, Waikoloa, HI, 2011, pp. 1-7, doi: 10.23919/OCEANS.2011.6107075

Lurton, X., Eleftherakis, D. & Augustin, J. (2018) “Analysis of seafloor backscatter strength dependence on the survey azimuth using multibeam echosounder data.” Mar Geophys Res 39, 183–203. https://doi.org/10.1007/s11001-017-9318-3

McKinney, C. M., and Anderson, C. D. (1964) “Measurements of backscattering of sound from the ocean bottom.” J. Acoust. Soc. Am. 36(1), 158–163.

Mitchell, N. C. and Somers, M. L. (1989) “Quantitative backscatter measurements with a long-range side-scan sonar.” in IEEE Journal of Oceanic Engineering, vol. 14, no. 4, pp. 368-374, Oct., doi: 10.1109/48.35987.

de Moustier, C. (1986) “Beyond bathymetry: Mapping acoustic backscattering from the deep seafloor with Sea Beam.” J. Acoust. Soc. Am. 79, 3 IG33 I

NOAA (2020) “Standard Sphere Target Strength Calculator.” https://swfscdata.nmfs.noaa.gov/AST/SphereTS/ Accessed in September 2020.

Ogilvy, J. A. (1991) “Theory of wave scattering from random rough surfaces.” publisher, A. Hilger, Bristol, England, Philadelphia

Ona, E., Mazauric, V., and Andersen, L. N. (2009) “Calibration methods for two scientific multibeam systems.” – ICES Journal of Marine Science, 66: 1326–1334.

Peng, Z., Zhou, J. X., Dahl, P. H. and Zhang, R. (2004) “Sea-bed acoustic parameters from dispersion analysis and transmission loss in the East China Sea.” in IEEE Journal of Oceanic Engineering, vol. 29, no. 4, pp. 1038-1045, Oct., doi: 10.1109/JOE.2004.836400

Roche, M. (2002) “Utilisation du sonar multifaisceaux pour la classification acoustique des sédiments et son application à la cartographie de la zone de concession 2 de la mer territoriale et du Plateau Continental Belge: etude de faisabilité.” Ministère des Affaires Économiques, Administration Qualité et Sécurité, Brussel. 1–67. http://www.vliz.be/en/imis?module=ref&refid=60184 Accessed 26 Aug. 2020

Roche, M., Degrendele, K., Vrignaud, C. et al. (2018) “Control of the repeatability of high frequency multibeam echosounder backscatter by using natural reference areas.” Mar Geophys Res 39, 89–104. https://doi.org/10.1007/s11001-018-9343-x

Stanic, S., Briggs, K. B., Fleischer, P., Ray, R. I. and Sawyer, W. B. (1988) “Shallow-Water high-frequency bottom scattering off Panama City, Florida.” J. Acoust. Soc. Am. 83, 2134-2144

Stanic, S., Briggs, K. B., Fleischer, P., Sawyer, W. B. and Ray, R. I. (1989) “High-frequency acoustic backscattering from a coarse shell ocean bottom.” J. Acoust. Soc. Am. 85, 125, doi: 10.1121/1.397720

Tang, D., (1997) ‘‘Small scale volumetric inhomogeneities of shallow water sediments: Measurements and discussion in High Frequency Acoustics in Shallow Water.” CP-45, pp. 539–546, Lerici, Italy, July 1997. NATO SACLANT Undersea Research Centre.

Tang, D., Briggs, K. B., Williams, K. L., Jackson, D. R., Thorsos, E. I. and Percival, D. B. (2002) “Fine-scale volume heterogeneity measurements in sand.” in IEEE Journal of Oceanic Engineering, vol. 27, no. 3, pp. 546-560, July, doi: 10.1109/JOE.2002.1040937.

Ward, L. G. and Birch, F. S. (1999) “Sedimentology of the New Hampshire Inner Continental Shelf Based on Subbottom Seismics, Side Scan Sonar, Bathymetry and Bottom Samples.” Marine Georesources and Geotechnology, vol. 17. Taylor & Francis, pp. 165-172,

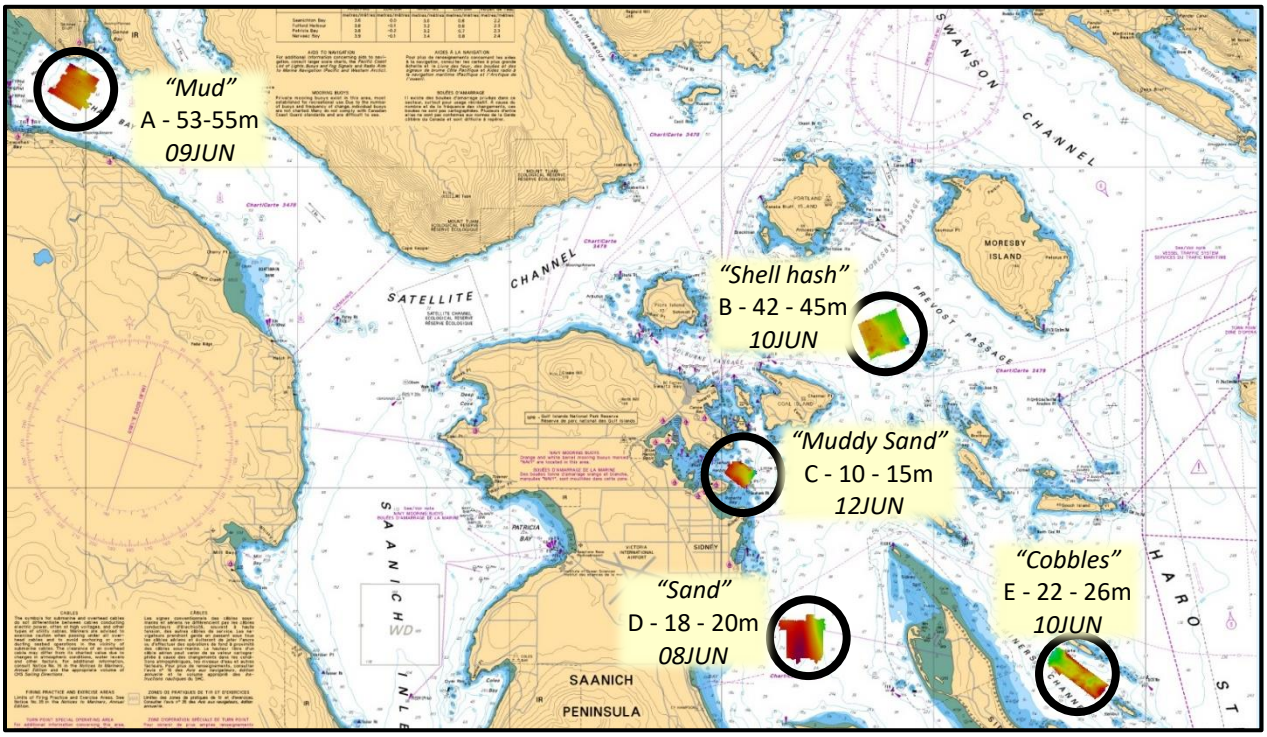
Weber, T., (2013) “Acoustic Seabed Interaction Theory: Multibeam Training Course.” Lecture 19, Niteroi. Lecture Notes.

Weber, T., and Ward, L. G. (2015) “Observations of backscatter from sand and gravel seafloors between 170 and 250 kHz.” The Journal of the Acoustical Society of America 138, 2169 (2015); https://doi-org.unh.idm.oclc.org/10.1121/1.4930185

Weber, T. C., Rice, G. and Smith, M. (2018) “Toward a standard line for use in multibeam echo sounder calibration.” Mar Geophys Res 39, 75–87. https://doi.org/10.1007/s11001-017-9334-3

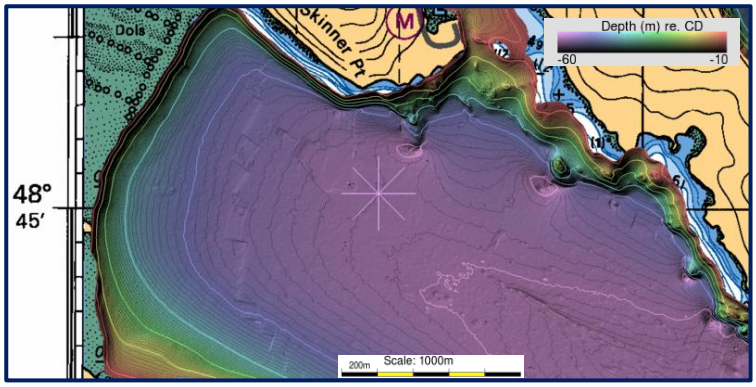
APPENDIX I

Pictures of the bottom samples for each area.

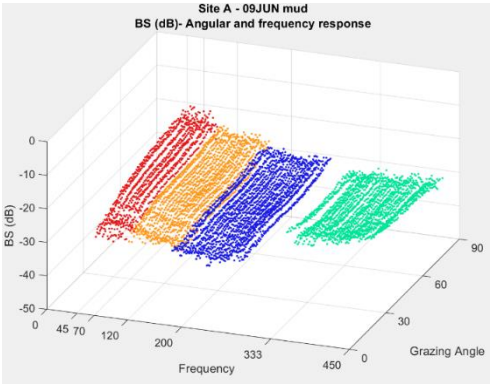


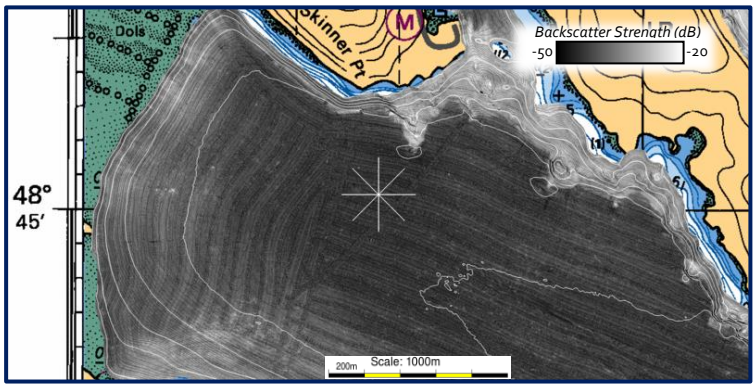
Nautical Chart CHS 3441 - Haro Strait Boundary Pass and Satellite Channel with the location of each area.

AREA A (Cowicham) – Mud

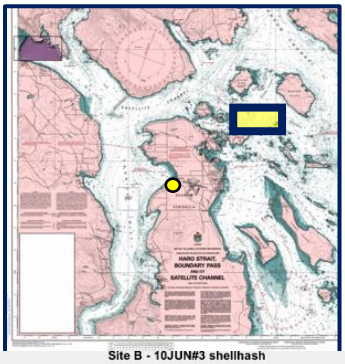


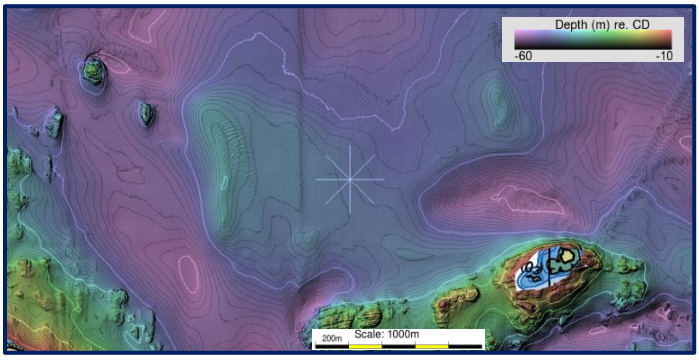
EM3000 or EM3002, 1999-2005, CCGS Revisor or CCGS Otter Bay



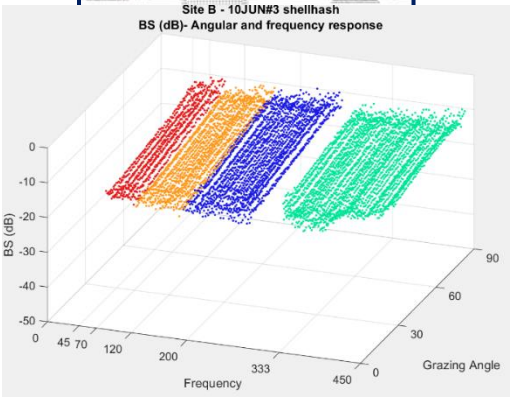


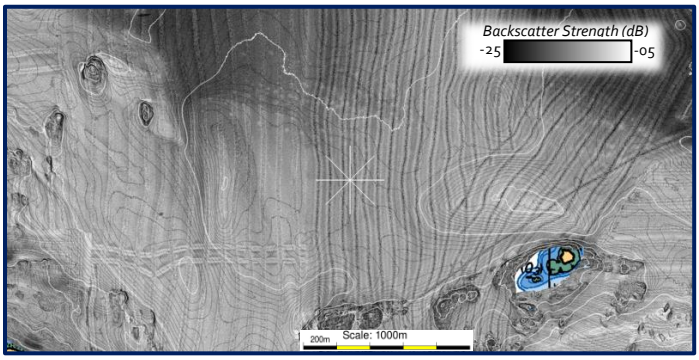
AREA B (Prevost Passage) – Shell hash

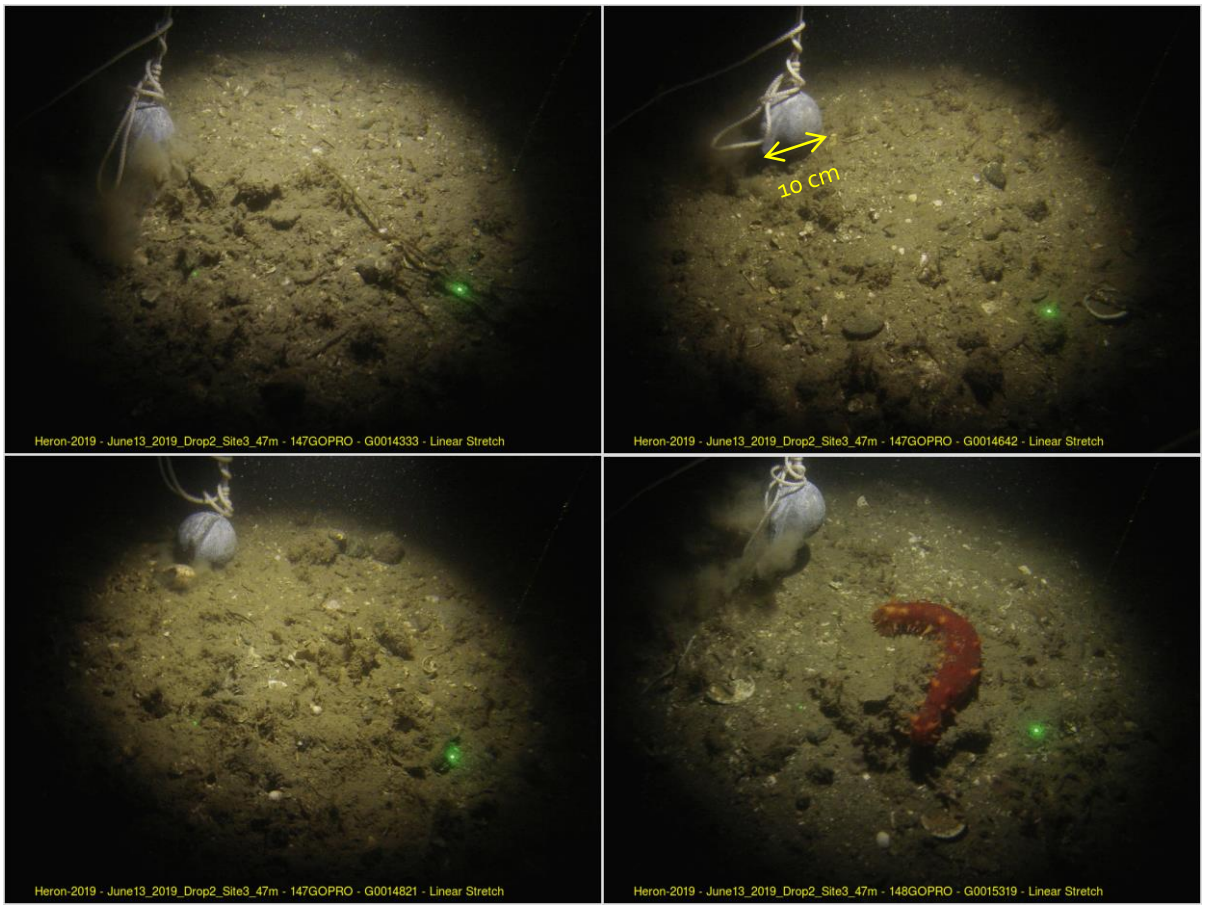




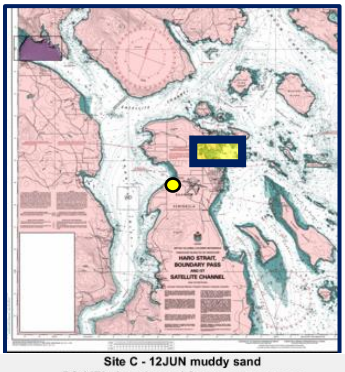
EM3000 or EM3002, 1999-2005, CCGS Revisor or CCGS Otter Bay

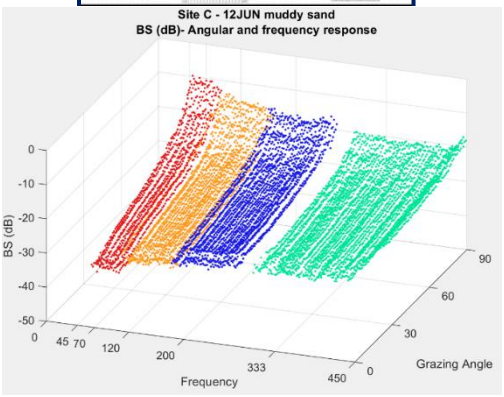


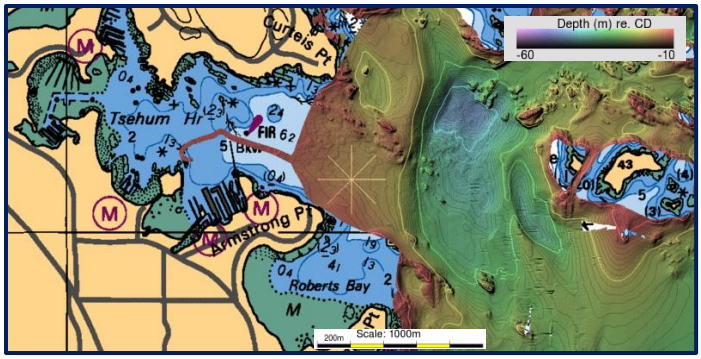



Heron-2019 - June13_2019_Drop2_Site3_47m - 147GOPRO - G0014333 - Linear Stretch
10 cm
Heron-2019 - June13_2019_Drop2_Site3_47m - 147GOPRO - G0014642 - Linear Stretch
Heron-2019 - June13_2019_Drop2_Site3_47m - 147GOPRO - G0014821 - Linear Stretch
Heron-2019 - June13_2019_Drop2_Site3_47m - 148GOPRO - G0015319 - Linear Stretch

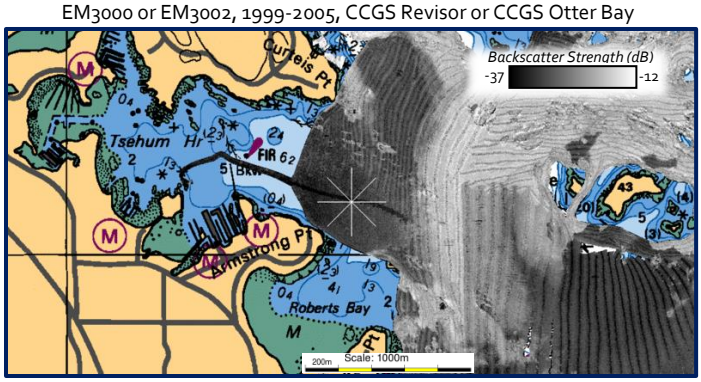
AREA C (Van Isle) – Muddy sand

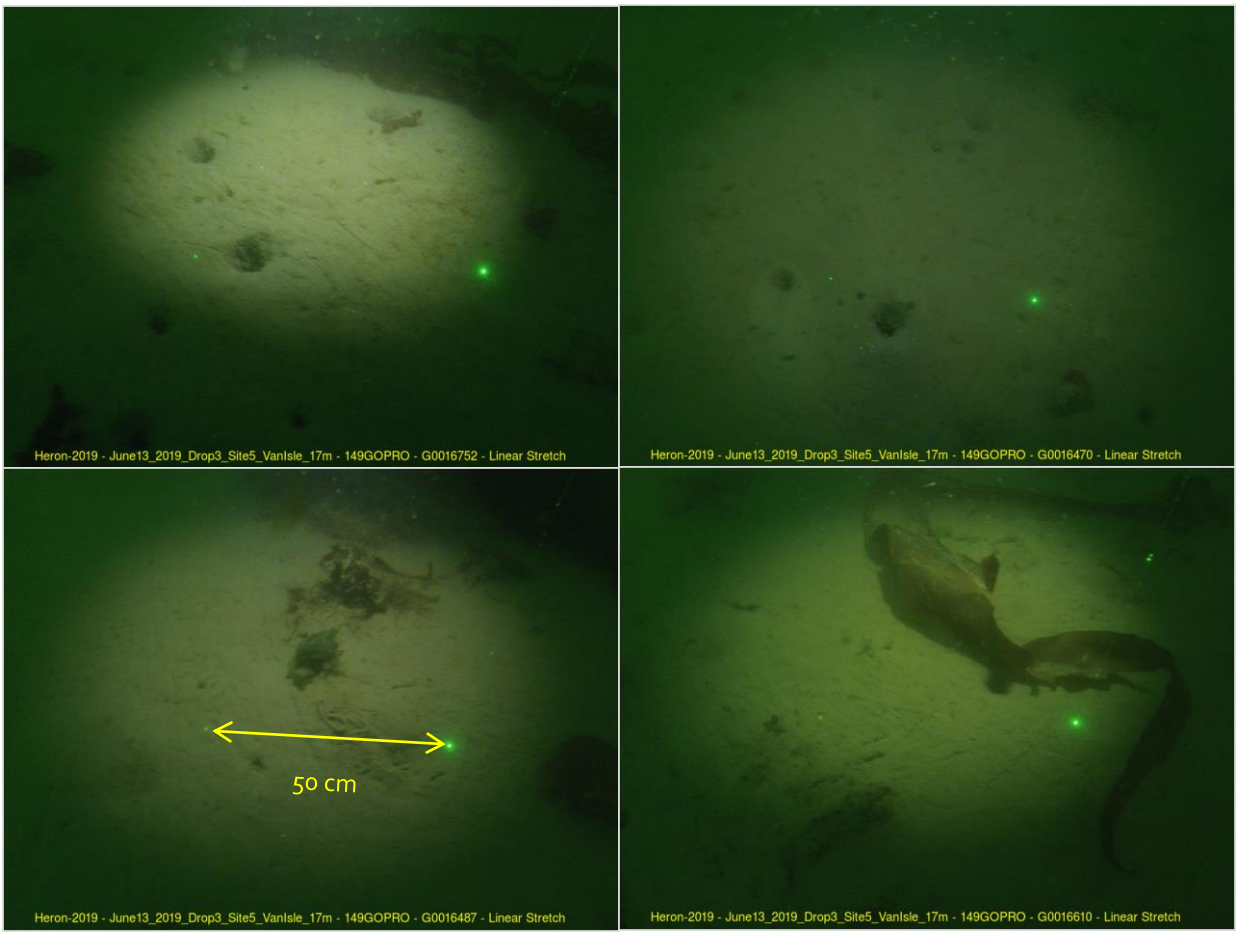




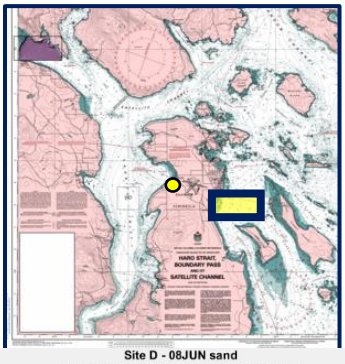


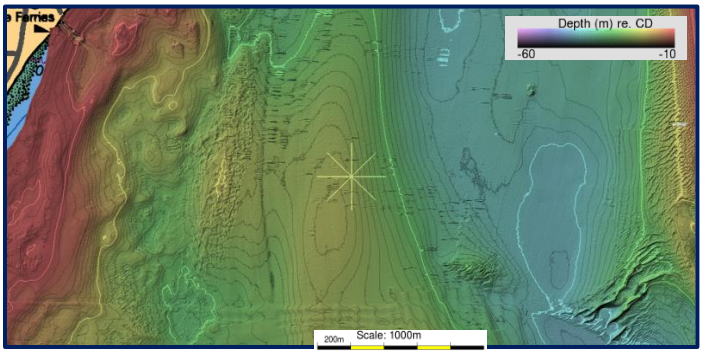
EM3000 or EM3002, 1999-2005, CCGS Revisor or CCGS Otter Bay

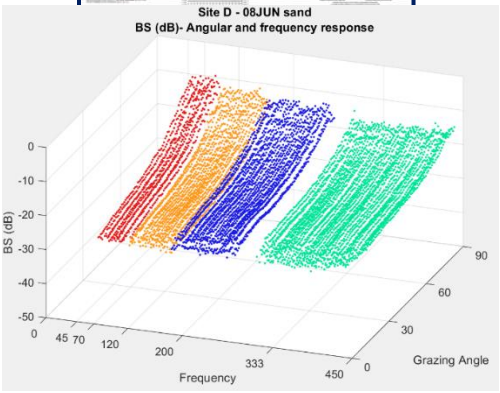



Heron-2019 - June13_2019_Drop3_Site5_VanIsle_17m - 149GOPRO - G0016752 - Linear Stretch
Heron-2019 - June13_2019_Drop3_Site5_VanIsle_17m - 149GOPRO - G0016470 - Linear Stretch
50 cm
Heron-2019 - June13_2019_Drop3_Site5_VanIsle_17m - 149GOPRO - G0016487 - Linear Stretch
Heron-2019 - June13_2019_Drop3_Site5_VanIsle_17m - 149GOPRO - G0016610 - Linear Stretch

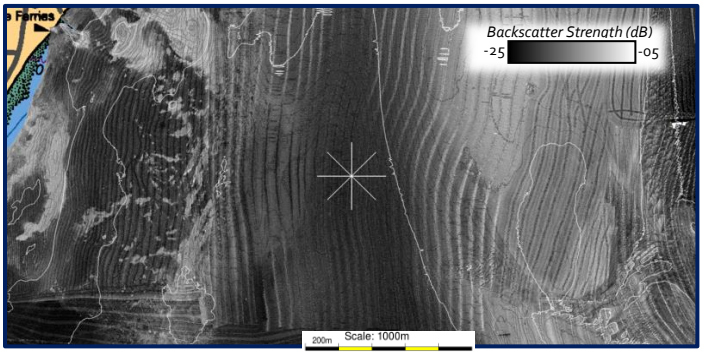
AREA D (Seapen Site) – Sand





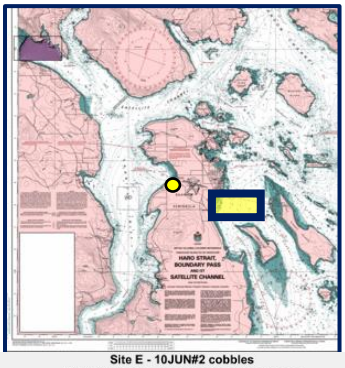


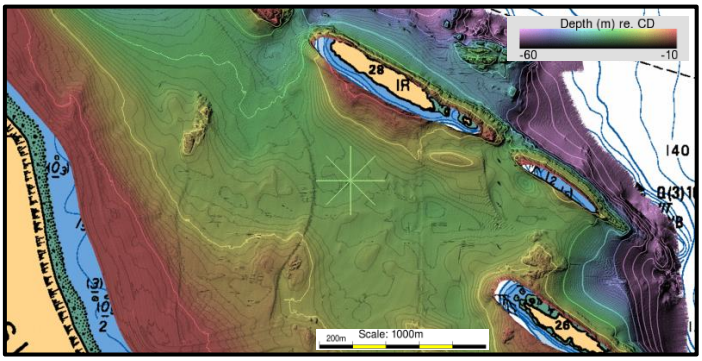
EM3000 or EM3002, 1999-2005, CCGS Revisor or CCGS Otter Bay

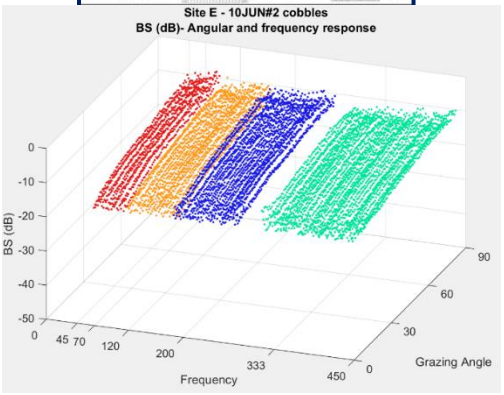



Heron-2019 - June13_2019_Drop1_Site2_20m - 144GOPRO - G0090089 - Linear Stretch
Heron-2019 - June13_2019_Drop1_Site2_20m - 144GOPRO - G0090063 - Linear Stretch
Heron-2019 - June13_2019_Drop1_Site2_20m - 144GOPRO - G0089945 - Linear Stretch
Heron-2019 - June13_2019_Drop1_Site2_20m - 144GOPRO - G0090178 - Linear Stretch
Heron-2019 - June13_2019_Drop1_Site2_20m - 144GOPRO - G0090223 - Linear Stretch

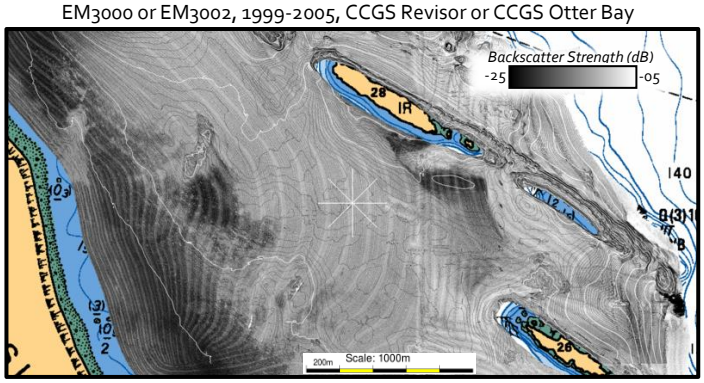
AREA E (Miners Channel) – Cobbles



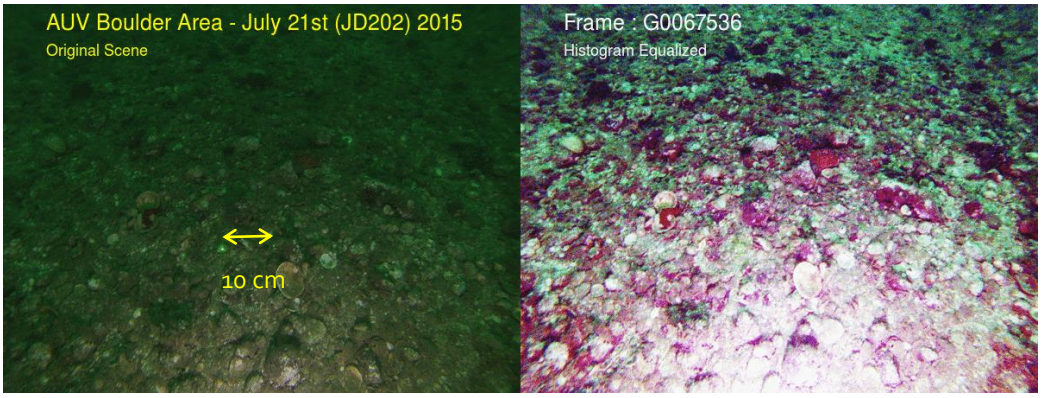




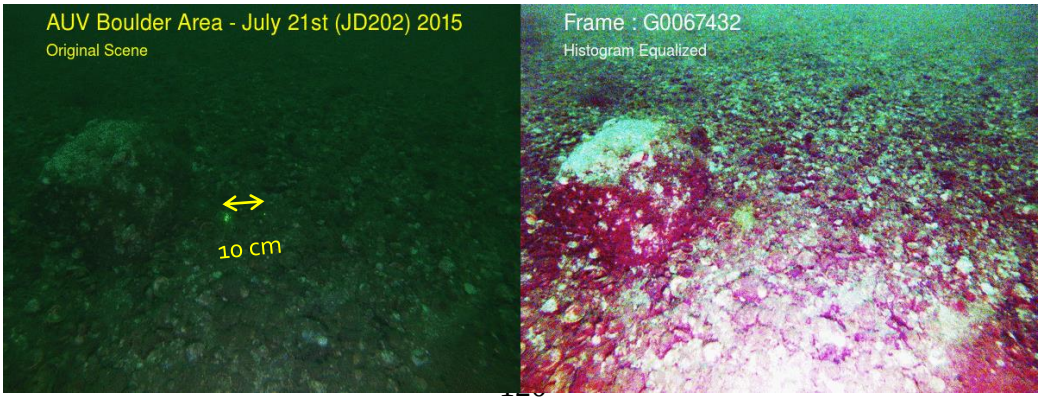
EM3000 or EM3002, 1999-2005, CCGS Revisor or CCGS Otter Bay



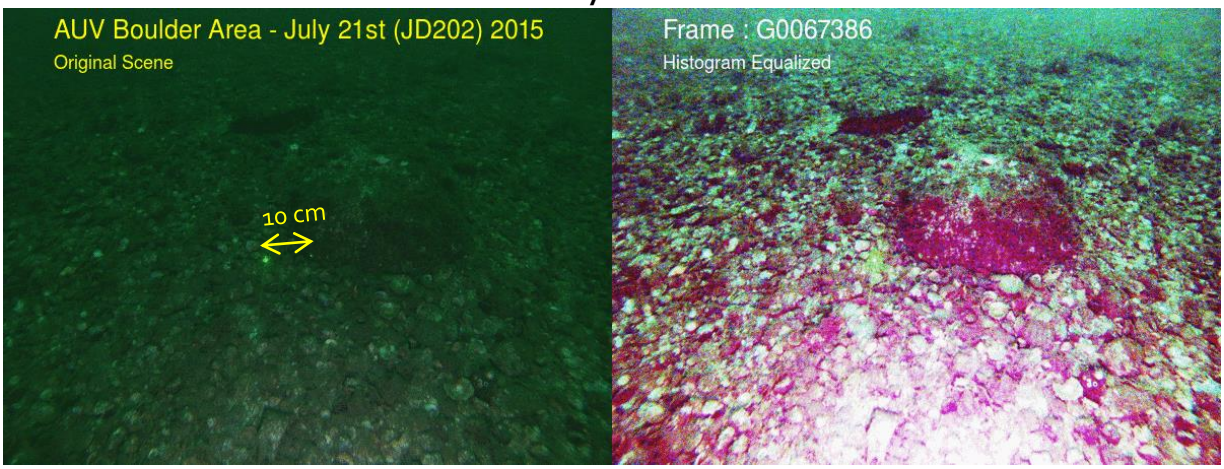
mollusk colonized gravel pavement



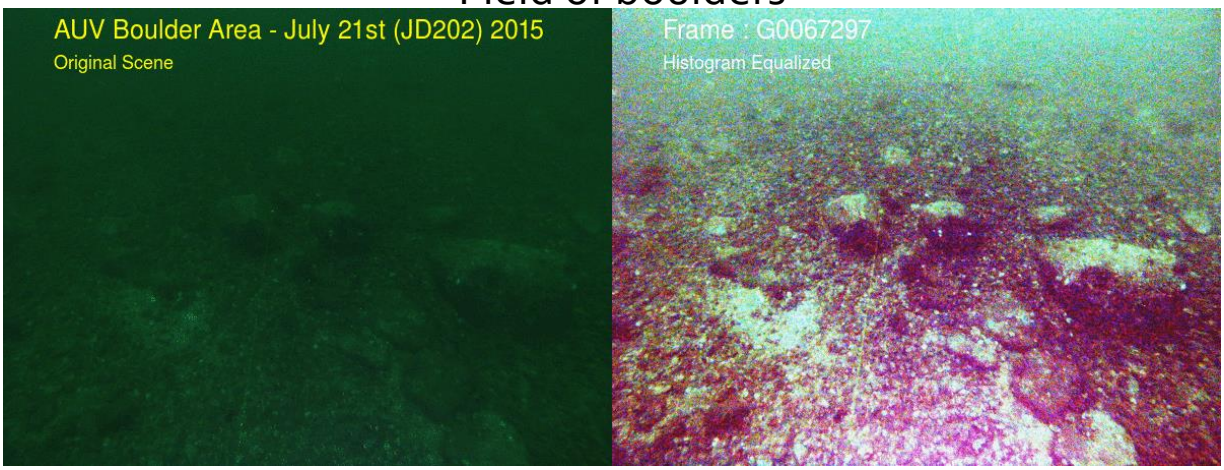
Solitary



Solitary boulder



Field of boulders



APPENDIX II

MATLAB codes.

EK80data_processor_app.m

```
classdef EK80data_processor_app < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                        matlab.ui.Figure
        Panel                           matlab.ui.container.Panel
        TransducerusedButtonGroup       matlab.ui.container.ButtonGroup
        AlltransducersButton            matlab.ui.control.RadioButton
        kHzButton                       matlab.ui.control.RadioButton
        kHzButton_2                     matlab.ui.control.RadioButton
        kHzButton_3                     matlab.ui.control.RadioButton
        kHzButton_4                     matlab.ui.control.RadioButton
        OperationLabel                  matlab.ui.control.Label
        AngledistinctionEditFieldLabel  matlab.ui.control.Label
        AngledistinctionEditField       matlab.ui.control.NumericEditField
        MaximumDepthmEditFieldLabel     matlab.ui.control.Label
        MaximumDepthmEditField          matlab.ui.control.NumericEditField
        MinimumDepthmEditFieldLabel     matlab.ui.control.Label
        MinimumDepthmEditField          matlab.ui.control.NumericEditField
        AciditypHEditFieldLabel         matlab.ui.control.Label
        AciditypHEditField              matlab.ui.control.NumericEditField
        SoundspeedmsEditFieldLabel      matlab.ui.control.Label
        SoundspeedmsEditField           matlab.ui.control.NumericEditField
        TemperatureCEditFieldLabel      matlab.ui.control.Label
        TemperatureCEditField           matlab.ui.control.NumericEditField
        SalinityppmEditFieldLabel       matlab.ui.control.Label
        SalinityppmEditField            matlab.ui.control.NumericEditField
        InputParametersLabel            matlab.ui.control.Label
        SelectfilesButton               matlab.ui.control.Button
        Switch2                         matlab.ui.control.ToggleSwitch
        Panel_2                         matlab.ui.container.Panel
        PlotButton                      matlab.ui.control.Button

        BeforeTargetEditFieldLabel      matlab.ui.control.Label
        BeforeTargetEditField           matlab.ui.control.NumericEditField
        AfterTargetEditFieldLabel       matlab.ui.control.Label
        AfterTargetEditField            matlab.ui.control.NumericEditField

        % Variable EK80_data
        EK80_data;
     end
```

```
    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Position = [100 100 549 434];
            app.UIFigure.Name = 'UI Figure';

            % Create Panel
            app.Panel = uipanel(app.UIFigure);
            app.Panel.Position = [5 15 260 410];

            % Create Switch2
            app.Switch2 = uiswitch(app.Panel, 'toggle');
            app.Switch2.Items = {'Bottom', 'Sphere'};
            app.Switch2.Orientation = 'horizontal';
            app.Switch2.FontSize = 14;
            app.Switch2.FontWeight = 'bold';
            app.Switch2.FontColor = [0.851 0.3255 0.098];
            app.Switch2.Position = [107 308 45 20];
            app.Switch2.Value = 'Bottom';

            % Create TransducerusedButtonGroup
            app.TransducerusedButtonGroup = uibuttongroup(app.Panel);
            app.TransducerusedButtonGroup.AutoResizeChildren = 'off';
            app.TransducerusedButtonGroup.Title = 'Transducer used';
            app.TransducerusedButtonGroup.FontSize = 14;
            app.TransducerusedButtonGroup.Position = [64 106 131 178];

            % Create AlltransducersButton
            app.AlltransducersButton =
uiradiobutton(app.TransducerusedButtonGroup);
            app.AlltransducersButton.Text = 'All transducers';
            app.AlltransducersButton.Position = [11 131 102 22];
            app.AlltransducersButton.Value = true;

            % Create kHzButton
            app.kHzButton = uiradiobutton(app.TransducerusedButtonGroup);
            app.kHzButton.Text = '#1';
            app.kHzButton.Position = [11 101 66 22];

            % Create kHzButton_2
            app.kHzButton_2 = uiradiobutton(app.TransducerusedButtonGroup);
            app.kHzButton_2.Text = '#2';
            app.kHzButton_2.Position = [11 71 66 22];

            % Create kHzButton_3
            app.kHzButton_3 = uiradiobutton(app.TransducerusedButtonGroup);
            app.kHzButton_3.Text = '#3';
            app.kHzButton_3.Position = [11 41 66 22];

            % Create kHzButton_4
            app.kHzButton_4 = uiradiobutton(app.TransducerusedButtonGroup);
```

```
app.kHzButton_4.Text = '#4';
app.kHzButton_4.Position = [11 11 60 22];

% Create OperationLabel
app.OperationLabel = uilabel(app.Panel);
app.OperationLabel.HorizontalAlignment = 'center';
app.OperationLabel.FontSize = 16;
app.OperationLabel.FontWeight = 'bold';
app.OperationLabel.Position = [42 361 176 22];
app.OperationLabel.Text = 'Operation';

% Create Panel_2
app.Panel_2 = uipanel(app.UIFigure);
app.Panel_2.Position = [270 15 250 410];

% Create SalinityppmEditFieldLabel
app.SalinityppmEditFieldLabel = uilabel(app.Panel_2);
app.SalinityppmEditFieldLabel.HorizontalAlignment = 'right';
app.SalinityppmEditFieldLabel.Position = [42 322 79 22];
app.SalinityppmEditFieldLabel.Text = 'Salinity (ppm)';

% Create SalinityppmEditField
app.SalinityppmEditField = uieditfield(app.Panel_2, 'numeric');
app.SalinityppmEditField.Position = [177 322 40 22];
app.SalinityppmEditField.Value = 30;

% Create TemperatureCEditFieldLabel
app.TemperatureCEditFieldLabel = uilabel(app.Panel_2);
app.TemperatureCEditFieldLabel.HorizontalAlignment = 'right';
app.TemperatureCEditFieldLabel.Position = [42 292 98 22];
app.TemperatureCEditFieldLabel.Text = 'Temperature (°C)';

% Create TemperatureCEditField
app.TemperatureCEditField = uieditfield(app.Panel_2, 'numeric');
app.TemperatureCEditField.Position = [177 292 40 22];
app.TemperatureCEditField.Value = 11;

% Create SoundspeedmsEditFieldLabel
app.SoundspeedmsEditFieldLabel = uilabel(app.Panel_2);
app.SoundspeedmsEditFieldLabel.HorizontalAlignment = 'right';
app.SoundspeedmsEditFieldLabel.Position = [42 262 107 22];
app.SoundspeedmsEditFieldLabel.Text = 'Sound speed (m/s)';

% Create SoundspeedmsEditField
app.SoundspeedmsEditField = uieditfield(app.Panel_2, 'numeric');
app.SoundspeedmsEditField.Position = [177 262 40 22];
app.SoundspeedmsEditField.Value = 1490;

% Create AciditypHEditFieldLabel
app.AciditypHEditFieldLabel = uilabel(app.Panel_2);
app.AciditypHEditFieldLabel.HorizontalAlignment = 'right';
app.AciditypHEditFieldLabel.Position = [42 232 68 22];
app.AciditypHEditFieldLabel.Text = 'Acidity (pH)';

% Create AciditypHEditField
app.AciditypHEditField = uieditfield(app.Panel_2, 'numeric');
```

```
            app.AcidityPHEditField.Position = [177 232 40 22];
            app.AcidityPHEditField.Value = 8;

            % Create MinimumDepthmEditFieldLabel
            app.MinimumDepthmEditFieldLabel = uilabel(app.Panel_2);
            app.MinimumDepthmEditFieldLabel.HorizontalAlignment = 'right';
            app.MinimumDepthmEditFieldLabel.Position = [42 202 114 22];
            app.MinimumDepthmEditFieldLabel.Text = 'Minimum Depth  (m)';

            % Create MinimumDepthmEditField
            app.MinimumDepthmEditField = uieditfield(app.Panel_2, 'numeric');
            app.MinimumDepthmEditField.Position = [177 202 40 22];
            app.MinimumDepthmEditField.Value = 5;

            % Create MaximumDepthmEditFieldLabel
            app.MaximumDepthmEditFieldLabel = uilabel(app.Panel_2);
            app.MaximumDepthmEditFieldLabel.HorizontalAlignment = 'right';
            app.MaximumDepthmEditFieldLabel.Position = [42 172 111 22];
            app.MaximumDepthmEditFieldLabel.Text = 'MaximumDepth (m)';

            % Create MaximumDepthmEditField
            app.MaximumDepthmEditField = uieditfield(app.Panel_2, 'numeric');
            app.MaximumDepthmEditField.Position = [177 172 40 22];
            app.MaximumDepthmEditField.Value = 10;

             % Create BeforeTargetFieldLabel
            app.BeforeTargetEditFieldLabel = uilabel(app.Panel_2);
            app.BeforeTargetEditFieldLabel.HorizontalAlignment = 'right';
            app.BeforeTargetEditFieldLabel.Position = [42 142 99 22];
            app.BeforeTargetEditFieldLabel.Text = 'Before Target (m)';

            % Create BeforeTargetEditField
            app.BeforeTargetEditField = uieditfield(app.Panel_2, 'numeric');
            app.BeforeTargetEditField.Position = [177 142 40 22];
            app.BeforeTargetEditField.Value = 1.7;

            % Create AfterTargetFieldLabel
            app.AfterTargetEditFieldLabel = uilabel(app.Panel_2);
            app.AfterTargetEditFieldLabel.HorizontalAlignment = 'right';
            app.AfterTargetEditFieldLabel.Position = [42 112 88 22];
            app.AfterTargetEditFieldLabel.Text = 'After Target (m)';

            % Create AfterTargetEditField
            app.AfterTargetEditField = uieditfield(app.Panel_2, 'numeric');
            app.AfterTargetEditField.Position = [177 112 40 22];
            app.AfterTargetEditField.Value = 1;

            % Create AngledistinctionEditFieldLabel
            app.AngledistinctionEditFieldLabel = uilabel(app.Panel_2);
            app.AngledistinctionEditFieldLabel.HorizontalAlignment = 'right';
            app.AngledistinctionEditFieldLabel.Position = [42 82 109 22];
            app.AngledistinctionEditFieldLabel.Text = 'Angle distinction
(°)';

            % Create AngledistinctionEditField
```

```
            app.AngledistinctionEditField = uieditfield(app.Panel_2,
'numeric');
            app.AngledistinctionEditField.Position = [177 82 40 22];
            app.AngledistinctionEditField.Value = 10;

            % Create InputParametersLabel
            app.InputParametersLabel = uilabel(app.Panel_2);
            app.InputParametersLabel.HorizontalAlignment = 'center';
            app.InputParametersLabel.FontSize = 15;
            app.InputParametersLabel.FontWeight = 'bold';
            app.InputParametersLabel.Position = [42 361 176 22];
            app.InputParametersLabel.Text = 'Input Parameters';

            % Create SelectfilesButton
            app.SelectfilesButton = uibutton(app.Panel, 'push');
            app.SelectfilesButton.ButtonPushedFcn = createCallbackFcn(app,
@SelectfilesButtonPushed, true);
            app.SelectfilesButton.BackgroundColor = [0.5 0.8 0.3];
            app.SelectfilesButton.FontSize = 14;
            app.SelectfilesButton.FontWeight = 'bold';
            app.SelectfilesButton.Position = [80 46 100 24];
            app.SelectfilesButton.Text = 'Select files';

            % Create PlotButton
            app.PlotButton = uibutton(app.Panel_2, 'push');
            app.PlotButton.Text = 'Plot  >';
            app.PlotButton.BackgroundColor = [1 0.8 0.5];
            app.PlotButton.FontSize = 14;
            app.PlotButton.FontWeight = 'bold';
            app.PlotButton.Position = [101 45 58 24];
            app.PlotButton.ButtonPushedFcn = createCallbackFcn(app,
@PlotButtonPushed, true);

             % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end

    % Callbacks that handle component events
    methods (Access = private)

        % Button pushed function: SelectfilesButton
        function [EK80_data]=SelectfilesButtonPushed(app, event)
                app.EK80_data=Pressed_select_files(app,event);
        end

        % Button pushed function: PlotplotButton
        function PlotButtonPushed(app, event)
            Pressed_plot
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
```

```
        function app = EK80data_processor_app

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```

Pressed_select_files.m

```
% to be used with the app EK80data_processor_app
function [EK80_data]=Pressed_select_files(app,~)
%% get files

filelocation = 'F:\Thesis\Data\Field_BC\12JUN\Calibration';
[fname,fpath,~] = uigetfile(strcat(filelocation,'\*.raw'),'Select raw
files','MultiSelect','on');

if iscell(fname)
    numfiles = size(fname,2);
else
    numfiles = 1;
    if fname==0
        disp('No files selected')
        EK80_data=[];
        return
    end
end

%% Parse data
if strcmp(app.Switch2.Value,'Bottom')
    if app.AlltransducersButton.Value % if All tc were selected
        profile on
        for Channel=1:4
            depth_ini=app.MinimumDepthmEditField.Value;
            depth_end=app.MaximumDepthmEditField.Value;
            [EK80_data(Channel),~] =
ParseEK80_data_extracts(fpath,fname,Channel,numfiles,depth_ini,depth_end);
        end
    else
        Channel = [app.kHzButton.Value, app.kHzButton_2.Value,
app.kHzButton_3.Value, app.kHzButton_4.Value];
        [~,Channel]=max(Channel);
        depth_ini=app.MinimumDepthmEditField.Value;
        depth_end=app.MaximumDepthmEditField.Value;
        EK80_data =
ParseEK80_1tc_extracts(fpath,fname,Channel,numfiles,depth_ini,depth_end);
    end
else % if app.Switch2.Value = 'Sphere';
    if app.AlltransducersButton.Value % if All tc were selected
        disp('Bad attempt on Calibrating with ALL the transducers at the same
time. Choose one at a time.')

    else
        Channel = [app.kHzButton.Value, app.kHzButton_2.Value,
app.kHzButton_3.Value, app.kHzButton_4.Value];
        [~,Channel]=max(Channel);
        depth_ini=app.MinimumDepthmEditField.Value;
        depth_end=app.MaximumDepthmEditField.Value;
        EK80_data =
ParseEK80_1tc_extracts(fpath,fname,Channel,numfiles,depth_ini,depth_end);
        %         EK80_data = ParseEK80(fpath,fname,Channel,numfiles);
```

```
    end
end

% save EK80_data variable to a file
% save('C:\Users\Public\Documents\Simrad\EK80\Data\Field_BC\Mat
files\05JUN_CH4.mat','EK80_data');

display('Insert the parameters and press buttom to Plot')
end
```

ParseEK80_data_extracts.m

```
function [EK80Data,TWTT] =
ParseEK80_data_extracts(fpath,fname,Channels,numfiles,depth_ini,depth_end)
% Read and parse EK80 file
%   Input:
%           filename                = location of .RAW file
%   Output:
%           NMEA                    = position strings
%           configdata              = EK80 configuration data
%           echogram_SP             = approximate sound pressure
%           echogram_time           = time stamps
%           echogram_MF            = match filtered data
%           echogram_phi_along      = along track EPA
%           echogram_phi_across     = across track EPA
%           echogram_phi_analysis   = radial EPA
%           fs                      = sampling frequency


npingsmax = 1e6;
if numfiles > 1
    pings=zeros(1,numfiles); % prealocating
    sampledata = cell(1,numfiles);
    NMEA_concat = cell(1,numfiles);
    for i = 1:numfiles

        filename = [fpath fname{1,i}];

        [configdata,filterdatavec,sampledatamat,NMEA] =
readrawEK80(filename,npingsmax,i,Channels); % ChannelID);
        pings(i) = size(sampledatamat,2);
        sampledata(i) = {sampledatamat(Channels,:)};
        NMEA_concat(i) = {NMEA};
    end

    sampledatamat = [sampledata{:}];
    NMEA = [NMEA_concat{:}];
    clear sampledata; clear NMEA_concat;
else
    filename = [fpath fname];
    [configdata,filterdatavec,sampledatamat,NMEA] =
readrawEK80(filename,npingsmax,1,Channels); % ChannelID);
    sampledatamat=sampledatamat(Channels,:); % included on 19SEP2019
    pings = size(sampledatamat,2);
end

%% Prealocate variables
npings = size(sampledatamat,2);
totalpingno = 0;
notempty=0;
columns=2001;

for pingno = 1:npings
```

```
          if (~isempty(sampledatamat(1,pingno).complexsamples)) % is there
data in current ping?
              notempty=notempty+1;
          end
      end

      echogram_time = zeros(1,notempty);
      echogram_MF = zeros(notempty,columns);
      echogram_phi_along = zeros(notempty,columns);
      echogram_phi_across = zeros(notempty,columns);
      echogram_idx_ini = zeros(1,notempty);
      heave = zeros(1,notempty);
      roll = zeros(1,notempty);
      pitch = zeros(1,notempty);
      heading = zeros(1,notempty);

      %% read in complex data, transducer and environment data for current
ping

      for pingno = 1:npings
          clc
          home
          disp(['Parsing Ping no: ' int2str(pingno) ' / ' int2str(npings)]);
          disp(['Transducer: '
configdata.transceivers(Channels).channels.transducer.TransducerName]); %
ChannelID]) ;

          if (~isempty(sampledatamat(1,pingno).complexsamples)) % is there
data in current ping?

              % read in/get transceiver data
              transceiver =
gettransceiver(configdata,filterdatavec,sampledatamat,1,Channels,pingno,1);

              clear yrx

              % current ping data
              sampledata  = sampledatamat(1,pingno);

              % sum of quandrants for current ping
              yrx = sum(sampledata.complexsamples,2);

              % split aperature calculation
              y_fore = sum(sampledata.complexsamples(:,3:4),2)/2;
              y_aft = sum(sampledata.complexsamples(:,1:2),2)/2;
              y_star = (sampledata.complexsamples(:,1) +
sampledata.complexsamples(:,4))/2;
              y_port = sum(sampledata.complexsamples(:,2:3),2)/2;

              clear sampledata.complexsamples

              % create tx signal
              [~,ytx]=createtx(transceiver);

              %% match filter
```

```
                if (transceiver.iscw)
                    yc = yrx;
                else

                    % apply match filter
                    yrx_match = conv(yrx,flipud(conj(ytx)))/norm(ytx)^2;
                    y_fore_match = conv(y_fore,flipud(conj(ytx)))/norm(ytx)^2;
                    y_aft_match = conv(y_aft,flipud(conj(ytx)))/norm(ytx)^2;
                    y_star_match = conv(y_star,flipud(conj(ytx)))/norm(ytx)^2;
                    y_port_match = conv(y_port,flipud(conj(ytx)))/norm(ytx)^2;

                    % filter delay
                    delay_match = length(ytx);

                    % remove filter delay from data
                    yc = yrx_match(delay_match:end);
                    yf = y_fore_match(delay_match:end);
                    ya = y_aft_match(delay_match:end);
                    ys = y_star_match(delay_match:end);
                    yp = y_port_match(delay_match:end);

                    clear yrx_match y_fore_match y_aft_match y_star_match
y_port_match
                    clear y_fore y_aft  y_star y_port
                end

                %% apply secondary filter

                %[B,A] = butter(11,[45/62.5 90/62.5]);
                %yc = filtfilt(B,A,double(yc));

                %% calculate split-beam angles
                phi_along =
angle(yf.*conj(ya))*180/pi/(transceiver.alongfactor*transceiver.fnom/transcei
ver.fc) ;% 14FEB2020 % must be verified
                phi_across =
angle(ys.*conj(yp))*180/pi/(transceiver.athwfactor*transceiver.fnom/transceiv
er.fc) ;% 14FEB2020

                %% bottom detection
                if
strcmp(configdata.transceivers(Channels).channels.transducer.TransducerName,'
ES333-7C')
                    TS = movmean(20*log10(abs(yc)),[10 10],2,'omitnan'); % TS -
2TL = EL - SL
                else
                    TS = movmean(20*log10(abs(yc)),[10
10],2,'omitnan')+40*log10((1:length(yc))'*750/transceiver.fsdec); % TS - 2TL
= EL - SL
                end
                [~,TWTT]=
max(TS(max(1,round(depth_ini/750*transceiver.fsdec)):min(round(depth_end/750*
transceiver.fsdec),size(TS,1))));

                %% extract
```

```
            extract_ini =
max(round(depth_ini/750*transceiver.fsdec),TWTT+round(depth_ini/750*transceiv
er.fsdec)-floor(columns/2));
            extract_end =
min(size(TS,1),min(round(depth_end/750*transceiver.fsdec),TWTT+round(depth_in
i/750*transceiver.fsdec)+floor(columns/2)));
            yc = [yc(extract_ini:extract_end);NaN(columns-
extract_end+extract_ini-1,1)];% pad with NaN if extract limit is less than
2001

            %% save data
            length_yc=length(yc);
            totalpingno = totalpingno+1;
            configdata.pulse_length(totalpingno)=sampledata.pulselength; %
included on 31/Jan/2020
            echogram_time(totalpingno) = sampledata.time;
            echogram_MF(totalpingno,1:length_yc) = yc;
            echogram_idx_ini(totalpingno) = extract_ini;
            clear yc
            clear yrx
            echogram_phi_along(totalpingno,1:length_yc) =
[phi_along(extract_ini:extract_end);NaN(length_yc-extract_end+extract_ini-
1,1)];
            clear phi_along
            echogram_phi_across(totalpingno,1:length_yc) =
[phi_across(extract_ini:extract_end);NaN(length_yc-extract_end+extract_ini-
1,1)];
            clear phi_across
            heave(totalpingno) = sampledatamat(1,pingno).heave;
            roll(totalpingno) = sampledatamat(1,pingno).roll;
            pitch(totalpingno) = sampledatamat(1,pingno).pitch;
            heading(totalpingno) = sampledatamat(1,pingno).heading;
        end
    end
    if totalpingno > 0
        fs = 1/sampledata.sampleinterval;

        %% Clear exceeded data
        del_c=sum(echogram_MF)==0;
        echogram_MF(:,del_c) = [];
        echogram_phi_along(:,del_c) = [];
        echogram_phi_across(:,del_c) = [];

        del_l = sum(echogram_MF,2)==0;
        echogram_time(:,del_l) = [];
        echogram_MF(del_l,:) = [];
        echogram_phi_along(del_l,:) = [];
        echogram_phi_across(del_l,:) = [];
        echogram_idx_ini(:,del_l) = [];
        heave(:,del_l) = [];
        roll(:,del_l) = [];
        pitch(:,del_l) = [];
        heading(:,del_l) = [];

        %% Create EK80Data variable
        EK80Data.fs = fs;
```

```
            EK80Data.NMEA = NMEA;
            EK80Data.configdata = configdata;
            EK80Data.configdata.angle_sensitivity =
transceiver.alongfactor*transceiver.fc/transceiver.fnom;
            EK80Data.echogram_time = echogram_time;
            EK80Data.echogram_MF = echogram_MF;
            EK80Data.idx_ini = echogram_idx_ini;

            clear echogram_MF
            EK80Data.echogram_phi_along = echogram_phi_along;
            clear echogram_phi_along
            EK80Data.echogram_phi_across = echogram_phi_across;
            clear echogram_phi_across
            EK80Data.motion.heave = heave;
            EK80Data.motion.roll = roll;
            EK80Data.motion.pitch = pitch;
            EK80Data.motion.grazing = 90-sqrt(roll.^2+pitch.^2);
            EK80Data.motion.heading = heading;
            EK80Data.file_info.pings = pings;
            EK80Data.auto_tx= (conv(ytx,flipud(conj(ytx))));

      end
      end
```

ParseEK80_1tc_extracts.m

```
      function [EK80Data] =
ParseEK80_1tc_extracts(fpath,fname,Channels,numfiles,depth_ini,depth_end)
      % there is no actual difference from the ParseEK80_data_extracts except
when calling
      % "sampledata(channels,pingno)". Here it is called
"sampledata(1,pingno)", instead.

      % Read and parse EK80 file
      %   Input:
      %           filename                = location of .RAW file
      %   Output:
      %           NMEA                    = position strings
      %           configdata              = EK80 configuration data
      %           echogram_SP             = approximate sound pressure
      %           echogram_time           = time stamps
      %           echogram_MF            = match filtered data
      %           echogram_phi_along      = along track EPA
      %           echogram_phi_across     = across track EPA
      %           echogram_phi_analysis   = radial EPA
      %           fs                      = sampling frequency


      npingsmax = 1e6;

      if numfiles > 1
          sampledata = [];
          NMEA_concat = [];
          pings=zeros(1,numfiles); % prealocating
          for i = 1:numfiles

              filename = [fpath fname{1,i}];
              [configdata,filterdatavec,sampledatamat,NMEA] =
readrawEK80(filename,npingsmax,i,Channels);
              pings(i) = size(sampledatamat,2);
              sampledata = [sampledata sampledatamat(Channels,:)];
              NMEA_concat = [NMEA_concat NMEA];

          end
          sampledatamat = sampledata;
          NMEA = NMEA_concat;
          clear sampledata; clear NMEA_concat;
      else
          filename = [fpath fname];
          [configdata,filterdatavec,sampledatamat,NMEA] =
readrawEK80(filename,npingsmax,1,Channels); % ChannelID);
          sampledatamat=sampledatamat(Channels,:);
          pings = size(sampledatamat,2);
      end
      %% Prealocate variables
      npings = size(sampledatamat,2);
      totalpingno = 0;
      % Channels=1;
```

```
      notempty=0;
      % columns=zeros(1,npings);
      columns=2001;

      for pingno = 1:npings

          if (~isempty(sampledatamat(1,pingno).complexsamples)) % is there
data in current ping?
              notempty=notempty+1;
          end
      end
      echogram_time = zeros(1,npings);
      echogram_MF = zeros(npings,columns);
      echogram_idx_ini = zeros(1,npings);
      echogram_phi_along = zeros(npings,columns);
      echogram_phi_across = zeros(npings,columns);
      echogram_phi_analysis = zeros(npings,columns);
      heave = zeros(1,npings);
      roll = zeros(1,npings);
      pitch = zeros(1,npings);
      heading = zeros(1,npings);
      %% read in complex data, transducer and environment data for current
ping

      for pingno = 1:npings
          if (~isempty(sampledatamat(1,pingno).complexsamples)) % is there
data in current ping?

              % read in/get transceiver data
              transceiver =
gettransceiver(configdata,filterdatavec,sampledatamat,1,Channels,pingno,1);

              clear yrx

              % current ping data
              sampledata  = sampledatamat(1,pingno);

              % current environment data
              %         envdata = sampledata.env;

              % sum of quandrants for current ping
              yrx = sum(sampledata.complexsamples,2);

              % split aperature calculation
              y_fore = sum(sampledata.complexsamples(:,3:4),2)/2;
              y_aft = sum(sampledata.complexsamples(:,1:2),2)/2;
              y_star = (sampledata.complexsamples(:,1) +
sampledata.complexsamples(:,4))/2;
              y_port = sum(sampledata.complexsamples(:,2:3),2)/2;

              % complex samples for coherance
              % complex = sampledata.complexsamples;
              clear sampledata.complexsamples

              % create tx signal
              [~,ytx]=createtx(transceiver);
```

```
            %% match filter

            if (transceiver.iscw)
                yc = yrx;
            else
                % apply match filter
                yrx_match = conv(yrx,flipud(conj(ytx)))/norm(ytx)^2; %
yrx_match = conv(yrx,(conj(ytx)))/norm(ytx)^2;
                y_fore_match = conv(y_fore,flipud(conj(ytx)))/norm(ytx)^2;
                y_aft_match = conv(y_aft,flipud(conj(ytx)))/norm(ytx)^2;
                y_star_match = conv(y_star,flipud(conj(ytx)))/norm(ytx)^2;
                y_port_match = conv(y_port,flipud(conj(ytx)))/norm(ytx)^2;

                % filter delay
                delay_match = length(ytx);

                % remove filter delay from data
                yc = yrx_match(delay_match:end);
                yf = y_fore_match(delay_match:end);
                ya = y_aft_match(delay_match:end);
                ys = y_star_match(delay_match:end);
                yp = y_port_match(delay_match:end);
                clear yrx_match(delay_match:end)
y_fore_match(delay_match:end) y_aft_match(delay_match:end)
y_star_match(delay_match:end) y_port_match(delay_match:end)
                clear y_fore y_aft y_port y_star
            end

            %% apply secondary filter

            %[B,A] = butter(11,[45/62.5 90/62.5]);
            %yc = filtfilt(B,A,double(yc));

            %% calculate split-beam angles
            phi_along =
angle(yf.*conj(ya))*180/pi/(transceiver.alongfactor*transceiver.fnom/transcei
ver.fc) ;% 14FEB2020 % must be verified
            phi_across =
angle(ys.*conj(yp))*180/pi/(transceiver.athwfactor*transceiver.fnom/transceiv
er.fc) ;% 14FEB2020

            %% combine angles
            phi_analysis = sqrt(phi_along.^2+phi_across.^2);

            %% bottom detection
            if
strcmp(configdata.transceivers(Channels).channels.transducer.TransducerName,'
ES333-7C')
                TS = movmean(20*log10(abs(yc)),[10 10],2,'omitnan'); % TS -
2TL = EL - SL
            else
                TS = movmean(20*log10(abs(yc)),[10
10],2,'omitnan')+40*log10((1:length(yc))'*750/transceiver.fsdec); % TS - 2TL
= EL - SL
            end
```

```
            [~,TWTT]=
max(TS(round(depth_ini/750*transceiver.fsdec):min(round(depth_end/750*transce
iver.fsdec))));

            %% extract
            extract_ini =
max(round(depth_ini/750*transceiver.fsdec),TWTT+round(depth_ini/750*transceiv
er.fsdec)-1000);
            extract_end =
min(round(depth_end/750*transceiver.fsdec),TWTT+round(depth_ini/750*transceiv
er.fsdec)+1000);
            yc = [yc(extract_ini:extract_end);NaN(columns-
extract_end+extract_ini-1,1)];% pad with NaN if extract limit is less than
2001

            %% save data

            totalpingno = totalpingno+1;
            echogram_time(totalpingno) = sampledata.time;
            echogram_MF(totalpingno,1:length(yc)) = yc;
            echogram_idx_ini(totalpingno) = extract_ini;

            echogram_phi_along(totalpingno,1:length(yc)) =
[phi_along(extract_ini:extract_end);NaN(length(yc)-extract_end+extract_ini-
1,1)];;
            echogram_phi_across(totalpingno,1:length(yc)) =
[phi_across(extract_ini:extract_end);NaN(length(yc)-extract_end+extract_ini-
1,1)];;
            echogram_phi_analysis(totalpingno,1:length(yc)) =
[phi_analysis(extract_ini:extract_end);NaN(length(yc)-
extract_end+extract_ini-1,1)];;
            heave(totalpingno) = sampledatamat(1,pingno).heave;
            roll(totalpingno) = sampledatamat(1,pingno).roll;
            pitch(totalpingno) = sampledatamat(1,pingno).pitch;
            heading(totalpingno) = sampledatamat(1,pingno).heading;
        end
    end
    if totalpingno > 0
        fs = 1/sampledata.sampleinterval;
        %% Clear exceeded data
        del_c=sum(echogram_MF)==0;
        echogram_MF(:,del_c) = [];
        echogram_phi_along(:,del_c) = [];
        echogram_phi_across(:,del_c) = [];
        echogram_phi_analysis(:,del_c) = [];

        del_l = sum(echogram_MF,2)==0;
        echogram_time(:,del_l) = [];
        echogram_MF(del_l,:) = [];
        echogram_idx_ini(:,del_l) = [];
        echogram_phi_along(del_l,:) = [];
        echogram_phi_across(del_l,:) = [];
        echogram_phi_analysis(del_l,:) = [];
        heave(:,del_l) = [];
        roll(:,del_l) = [];
        pitch(:,del_l) = [];
```

```
            heading(:,del_l) = [];

            %% Create EK80Data variable
            EK80Data.fs = fs;
            EK80Data.NMEA = NMEA;
            EK80Data.configdata = configdata;
            EK80Data.configdata.angle_sensitivity =
transceiver.alongfactor*transceiver.fc/transceiver.fnom;
            EK80Data.echogram_time = echogram_time;
            EK80Data.echogram_MF = echogram_MF;
            clear echogram_MF
            EK80Data.idx_ini = echogram_idx_ini;
            EK80Data.echogram_phi_along = echogram_phi_along;
            clear echogram_phi_along
            EK80Data.echogram_phi_across = echogram_phi_across;
            clear echogram_phi_across
            EK80Data.echogram_phi_analysis = echogram_phi_analysis;
            clear echogram_phi_analysis
            EK80Data.motion.heave = heave;
            EK80Data.motion.roll = roll;
            EK80Data.motion.pitch = pitch;
            EK80Data.motion.grazing = 90-sqrt(roll.^2+pitch.^2);
            EK80Data.motion.heading = heading;
            EK80Data.file_info.pings = pings;
            EK80Data.auto_tx= (conv(ytx,flipud(conj(ytx))));
        end
        end
```

Pressed_plot.m

```
      %% import variables

      sal = app.SalinityppmEditField.Value;
      temp = app.TemperatureCEditField.Value;
      ss = app.SoundspeedmsEditField.Value;
      acid = app.AciditypHEditField.Value;
      depth_ini=app.MinimumDepthmEditField.Value;
      depth_end=app.MaximumDepthmEditField.Value;
      width=app.AngledistinctionEditField.Value;
      plate_Off=[0,0,0]; % offset of the plate

      if isprop(app,'EK80_data') % verify if EK80_data is already created
      EK80_data=app.EK80_data;%app.EK80_data
      end
      if isempty(EK80_data)
          filelocation = 'C:\Users\ivanb\OneDrive\Documentos\MATLAB\EK80_data
Parsed
files';%'C:\Users\Public\Documents\Simrad\EK80\Data\Field_BC\12JUN\Calibratio
n'; %\05JUN';
      load(uigetfile(strcat(filelocation,'\*.mat'),'Select compiled EK80_data
file','MultiSelect','off'));
      end

      if strcmp(app.Switch2.Value,'Bottom')
          if app.AlltransducersButton.Value % if All tc were selected
              Plot_bottom
          else
              Channels = [app.kHzButton.Value, app.kHzButton_2.Value,
app.kHzButton_3.Value, app.kHzButton_4.Value];
              [~,Channels]=max(Channels);
      %         Plot_1Tc % not created
          end
      else % if app.Switch2.Value = 'Sphere';
          if app.AlltransducersButton.Value % if All tc were selected
              disp('Bad attempt on Calibrating with ALL the transducers at the
same time. Choose one at a time.')
          else
              Channels = [app.kHzButton.Value, app.kHzButton_2.Value,
app.kHzButton_3.Value, app.kHzButton_4.Value];
              [~,Channels]=max(Channels);
              Plot_sphere
          end
      end
```

Plot_bottom.m

```
for Channels=1:4
    %% preliminary variables
    win_obs = 2^6-1; % so the FFT will have 64 (2^6) counts of data
    allow_HD = true;% true; % True for High Definition (subsets) | False for
MRA only
    gr_from_phase = true; % True to use grazing from phase ramp | False to
use from MRU + beam
    Gr_diflim = 4; % Limit of difference between Grazing from phase and Gr
from MRU+beam
    roll_min = 4.1; % minimum roll to look for a zero crossing
    cut_low_Gr=5; % when GR from MRU and phase are too different, consider
only MRU
    sites={'A - 09JUN mud';'B - 10JUN#3 shellhash';'C - 12JUN muddy sand';'D
- 08JUN sand';'E - 10JUN#2 cobbles';...
        'NH - 27AUG - Mouth';'NH - 27AUG - Maine';'NH - 29AUG - J1
Lighthouse';'NH - 29AUG - J2';'NH - 29AUG - J3 low BS';'NH - 29AUG - J4 high
BS'};
    % 1- 09JUN mud 55m ss=1491 t=12
    % 2- 10JUN#3 shellhash 44m ss=1487 t=11
    % 3- 12JUN muddy sand 14m ss=1487 t=12
    % 4- 08JUN sand 17m ss=1491 t=12
    % 5- 10JUN#2 cobbles 25m ss=1487 t=11
    site = sites{5};
    correct_beam_angle=false; % created to analyse errors on the beam angle
    refine_beam_pattern = true; % do not use beam pattern correction near
nadir (n_fft<3)
%     plots=zeros(1,20);
    plots=ones(1,20); % select if all or none plots should be created
    %     plots(18:20)=1; % select which plots shoud be created

    clear FFT_bottom fft_bottom_ping ranges grazing_actual
    EK80_data(Channels).echogram_MF(EK80_data(Channels).echogram_MF==0)=NaN;
    depth_ini_idx = EK80_data(Channels).idx_ini'-1;
    numpings = size(EK80_data(Channels).echogram_MF,1);
    numsamples = size(EK80_data(Channels).echogram_MF,2)+max(depth_ini_idx);
    sensitivity = EK80_data(Channels).configdata.angle_sensitivity;

    % correct TL for plotting
    r = (1:numsamples)/EK80_data(Channels).fs*750;
    r_tvg = ((0:size(EK80_data(Channels).echogram_MF,2)-
1)+repmat(depth_ini_idx,1,size(EK80_data(Channels).echogram_MF,2)))/EK80_data
(Channels).fs*750;
    phase_angle = 5; % define the angle to select the extract of the phase
ramp

    if
strcmp(EK80_data(1).configdata.transceivers(1).channels.transducer.Transducer
Name,'ES70-7C')
        color_plot=[.9,.1,.1 ; 1,.6,.1 ; .1,.1,.9 ; 0,.9,.6];
    else
        color_plot=[.1,.1,.9 ; 1,.6,.1 ; .9,.1,.1 ; 0,.9,.6];
    end
```

```
    TVG_40 = 40*log10(r_tvg);% + 2*absorp.*r; % 2*TL
    TS = movmean(20*log10(abs(EK80_data(Channels).echogram_MF)),[10
10],2,'omitnan'); % TS - 2TL = EL - SL
    TS = TS + TVG_40;% TL will be applied after bottom detection
    %% Select the calibration files
    % use the following samples to test the script
    %     load('SAMPLE_Calib_12_06_ES200-7C.mat')
    % load the files created after the sphere calibration
    if
strcmp(EK80_data(Channels).configdata.transceivers(Channels).channels.transdu
cer.TransducerName,'ES70-7C')
        load('autotxNormal_Calib_MF_12JUN_ES70-7C.mat')
        fstart = 45; fstop = 95;
    elseif
strcmp(EK80_data(Channels).configdata.transceivers(Channels).channels.transdu
cer.TransducerName,'ES120-7C')
        load('autotxNormal_Calib_MF_12JUN_ES120-7C.mat')
        fstart = 90; fstop = 170;
    elseif
strcmp(EK80_data(Channels).configdata.transceivers(Channels).channels.transdu
cer.TransducerName,'ES200-7C')
        load('autotxNormal_Calib_MF_12JUN_ES200-7C.mat')
        fstart = 160; fstop = 260;
    elseif
strcmp(EK80_data(Channels).configdata.transceivers(Channels).channels.transdu
cer.TransducerName,'ES333-7C')
        load('autotxNormal_Calib_MF_12JUN_ES333-7C.mat')
        fstart = 280; fstop = 450;
    end
    %% Bottom detection
    mag_dev = 0*pi/180;
    heading=EK80_data(Channels).motion.heading*pi/180 + mag_dev; % including
a magnetic declination
    pitch=EK80_data(Channels).motion.pitch*pi/180;
    roll=EK80_data(Channels).motion.roll*pi/180;
    grazing = 90- atan(sqrt(tan(roll).^2+tan(pitch).^2))*180/pi;

    [TS_bottom,TWTT]= max(TS,[],2);

    range=zeros(numpings,3);
    range(:,3)=(TWTT+depth_ini_idx)/EK80_data(Channels).fs*750;
    footprint = repmat(plate_Off,numpings,1)+range;
    foot_print=zeros(3,numpings);

    %% Rotation matrix - to find the N and E coordinates and depths

    for nn=1:numpings
        rotz = [ cos(heading(nn))  -sin(heading(nn))   0 ; ...
            sin(heading(nn))  cos(heading(nn))  0 ; ...
            0              0          1 ] ;

        roty = [ cos(pitch(nn))  0  sin(pitch(nn)) ; ...
            0        1       0       ; ...
            -sin(pitch(nn))   0  cos(pitch(nn)) ] ;
```

```
        rotx = [  1           0                  0          ; ...
            0   cos(roll(nn))  sin(roll(nn))  ; ...
            0   -sin(roll(nn))  cos(roll(nn)) ] ;

        rot3d = rotz*roty*rotx ;

        foot_print(:,nn)=rot3d*footprint(nn,:)';
    end
    %% Reduce noise
    keep = (foot_print(3,:)>mean(foot_print(3,:))-
1*std(foot_print(3,:))&foot_print(3,:)<mean(foot_print(3,:))+1*std(foot_print
(3,:)))'; % Keeping only 68.27% of depths
    keep(grazing==90)=0;
    pings_selected = 1:numpings;
    pings_rejected = pings_selected(~keep);
    pings_selected=pings_selected(keep);
    %% Phase Detection
    target_idx = TWTT+depth_ini_idx;

    range_ini_bottom =
max(depth_ini_idx'+1,floor(r(target_idx).*sin((grazing)*pi/180)./sin(min(90,(
grazing+3))*pi/180)/750*EK80_data(Channels).fs))-depth_ini_idx';
    range_end_bottom =
min(depth_ini_idx'+size(EK80_data(Channels).echogram_MF,2)-
1,floor(r(target_idx).*sin((grazing)*pi/180)./sin(max(0.1,(grazing-
3))*pi/180)/750*EK80_data(Channels).fs))-depth_ini_idx';

    range_ini_phase = ones(1,numpings);
    range_end_phase = ones(1,numpings);
    depth_bottom = foot_print(3,:);
    depth_phase = zeros(1,numpings);
    zero_phase_m = zeros(1,numpings);
    n_fft = zeros(1,numpings);
    FFT_bottom = cell(size(pings_selected));
    FFT_bottom_uncor = cell(size(pings_selected));
    beam_angles = cell(size(pings_selected));
    for ii=pings_selected
        % for each ping:
        % - smooth the phase across and find zero-crossing
        % - find the window and the subsets to be analysed
        % - find the grazing angle associated to the zero-crossing
        % - when there is no zero-crossing (near nadir), use grazing from MRU
        % - FFT of the subsets
        clc;disp(['Calculating ping #' int2str(ii) '/' int2str(numpings)])
        if grazing(ii)<70
            apurar=1;
        end

smooth_phi=atan2(movmean(sind(EK80_data(Channels).echogram_phi_across(ii,:)*s
ensitivity),[100],'omitnan'),...

movmean(cosd(EK80_data(Channels).echogram_phi_across(ii,:)*sensitivity),[100]
,'omitnan'))*180/pi/sensitivity; %(23^2/sensitivity);
        % weighted version (didn't show much difference, hence it is not
used):
```

```
        %
smooth_phi=atan2(movmean(sind(EK80_data(Channels).echogram_phi_across(ii,:)*s
ensitivity).*abs(EK80_data(Channels).echogram_Rx(ii,:)),[200],'omitnan'),...
        %
movmean(cosd(EK80_data(Channels).echogram_phi_across(ii,:)*sensitivity).*abs(
EK80_data(Channels).echogram_Rx(ii,:)),[200],'omitnan'))*180/pi/(23^2/sensiti
vity);
        %
smooth_phi_MF=atan2(movmean(sind(EK80_data(Channels).echogram_phi_across(ii,:
)*sensitivity).*abs(EK80_data(Channels).echogram_MF(ii,:)),[100],'omitnan'),.
..
        %
movmean(cosd(EK80_data(Channels).echogram_phi_across(ii,:)*sensitivity).*abs(
EK80_data(Channels).echogram_MF(ii,:)),[100],'omitnan'))*180/pi/(23^2/sensiti
vity);
        if EK80_data(Channels).motion.roll(ii)>roll_min &&
smooth_phi(range_ini_bottom(ii))>0

[~,zero_phase_idx]=find(smooth_phi(range_ini_bottom(ii):range_end_bottom(ii))
<0,1);
        elseif EK80_data(Channels).motion.roll(ii)<-roll_min &&
smooth_phi(range_ini_bottom(ii))<0

[~,zero_phase_idx]=find(smooth_phi(range_ini_bottom(ii):range_end_bottom(ii))
>0,1);
        else % if there is no phase detection
            zero_phase_idx = find(1>2); % just to create an empty response
        end

        if ~isempty(zero_phase_idx)&allow_HD % if there is zero-crossing %
use 1==0 to force only amplitude detection
            win_shift = floor((win_obs+1)/2);
            range_ini_phase(ii) =
max(1,min(length(r),floor((r(depth_ini_idx(ii)+zero_phase_idx+range_ini_botto
m(ii)-
1).*sin((grazing(ii))*pi/180)./sin((grazing(ii)+phase_angle)*pi/180))/750*EK8
0_data(Channels).fs)-depth_ini_idx(ii))-floor(.4*win_obs)); %24MAY/2020
included offset (.4 * win_obs) to get the peak at high grazings
            range_end_phase(ii) =
max(range_ini_phase(ii)+win_obs+1,min(size(EK80_data(Channels).echogram_MF,2)
,floor((r(depth_ini_idx(ii)+zero_phase_idx+range_ini_bottom(ii)-
1).*sin((grazing(ii))*pi/180)./sin((grazing(ii)-
phase_angle)*pi/180))/750*EK80_data(Channels).fs)-depth_ini_idx(ii)));
            zero_phase_m(ii) = r(zero_phase_idx+range_ini_bottom(ii)-
1+depth_ini_idx(ii));
            depth_phase(ii) = -zero_phase_m(ii) * sin (grazing(ii)*pi/180);

            if
~isempty(find(isnan(EK80_data(Channels).echogram_phi_across(ii,range_ini_phas
e(ii)+1:end)),1))

nan_idx=find(isnan(EK80_data(Channels).echogram_phi_across(ii,range_ini_phase
(ii)+1:end)),1)+range_ini_phase(ii)-1;
                range_end_phase(ii) = min(range_end_phase(ii),nan_idx);
            end
```

```
            n_fft(ii) = floor((range_end_phase(ii) - win_shift -
range_ini_phase(ii))./win_shift);
            range_end_phase(ii) = range_ini_phase(ii) + win_shift + n_fft(ii)
* win_shift ;

            %% More efficient FFT (previous approach has been deleted)
            subset_ini = range_ini_phase(ii)+win_shift*((1:n_fft(ii))-1);
            subset_end = subset_ini+win_obs;
            subset_idx = zeros(win_obs+1,n_fft(ii));
            for sub=1:n_fft(ii)
                subset_idx(:,sub) =
(subset_ini(sub):subset_ini(sub)+win_obs);
            end
            beam_angle =
mean(smooth_phi(subset_idx),(n_fft(ii)==1)+1,'omitnan'); % (n_fft(ii)==1)+1
is used to go through columns or rows based on the numbers of subsets
            if correct_beam_angle
                beam_angle = beam_angle*1.05;
            end

            corr_r = r(subset_idx+depth_ini_idx(ii)).^2;
            MF = EK80_data(Channels).echogram_MF(ii,:);
            range_subset =
r(range_ini_phase(ii)+win_shift*(1:n_fft(ii))+depth_ini_idx(ii));
            Fdomain = (0:win_obs)*EK80_data(Channels).fs/win_obs;

            % Tc 70 kHz
            if
EK80_data(Channels).configdata.transceivers(Channels).channels.transducer.Fre
quency==70000
                fft_bottom_ping = (fft(MF(subset_idx).*corr_r));
                fstart = 45; fstop = 95;
                FFT_auto =
fft(EK80_data(Channels).auto_tx(floor(length(EK80_data(Channels).auto_tx)/2)-
floor(win_obs/2):win_obs+floor(length(EK80_data(Channels).auto_tx)/2)-
floor(win_obs/2)));

                % for 120kHz
            elseif
EK80_data(Channels).configdata.transceivers(Channels).channels.transducer.Fre
quency==120000
                fft_bottom_ping =          (fft(MF(subset_idx).*corr_r));
                fstart = 90; fstop = 170;
                FFT_auto =
fft(EK80_data(Channels).auto_tx(floor(length(EK80_data(Channels).auto_tx)/2)-
floor(win_obs/2):win_obs+floor(length(EK80_data(Channels).auto_tx)/2)-
floor(win_obs/2)));

                % Tc 200kHz
            elseif
EK80_data(Channels).configdata.transceivers(Channels).channels.transducer.Fre
quency==200000
                fft_bottom_ping = fftshift(fft(MF(subset_idx).*corr_r),2-
(n_fft(ii)>1));
                Fdomain = Fdomain +125000;
                fstart = 160; fstop = 260;
```

```
                FFT_auto =
fftshift(fft(EK80_data(Channels).auto_tx(floor(length(EK80_data(Channels).aut
o_tx)/2)-
floor(win_obs/2):win_obs+floor(length(EK80_data(Channels).auto_tx)/2)-
floor(win_obs/2))),1);

                % 333 kHz
            elseif
EK80_data(Channels).configdata.transceivers(Channels).channels.transducer.Fre
quency==333000
                fft_bottom_ping = (fft(MF(subset_idx).*corr_r));
                Fdomain=Fdomain+250000;
                fstart = 280; fstop = 450;
                FFT_auto =
fft(EK80_data(Channels).auto_tx(floor(length(EK80_data(Channels).auto_tx)/2)-
floor(win_obs/2):win_obs+floor(length(EK80_data(Channels).auto_tx)/2)-
floor(win_obs/2)));
            end
            %% Correct FFT for beam pattern
            [~,b] =
max(calib_file.beam_pattern.beam_angle>=min(max(calib_file.beam_pattern.beam_
angle),abs(beam_angle')),[],2);
            beam_pattern = calib_file.beam_pattern.correction(:,b);
            beam_pattern =
interp1(calib_file.beam_pattern.freq_domain,beam_pattern,Fdomain,'linear');
            if refine_beam_pattern & n_fft(ii)<3
                beam_pattern = zeros(size(fft_bottom_ping));
            end%             beam_pattern = 0;

            if n_fft(ii)==1

fft_bottom_ping_uncor=20*log10(abs(fft_bottom_ping'./FFT_auto));

fft_bottom_ping=20*log10(abs(fft_bottom_ping'./FFT_auto))+beam_pattern';
            else

fft_bottom_ping_uncor=20*log10(abs(fft_bottom_ping./FFT_auto));

fft_bottom_ping=20*log10(abs(fft_bottom_ping./FFT_auto))+beam_pattern;
            end
            %% Find the actual grazing angles, considering the slope of the
seafloor
            if beam_angle(1)>0
                th1 = find(beam_angle<beam_angle(1)-1.5,1);
            else
                th1 = find(beam_angle>beam_angle(1)+1.5,1);
            end
            r0 = range_subset(th1:end);
            r1 = range_subset(1:end-th1+1);
            th = abs(beam_angle(th1:end) - beam_angle(1:end-th1+1));

            if ~isempty(th)
                actual_gr = [zeros(1,th1-1) , asin(r0 .* sind(th) ./
sqrt(r1.^2 + r0.^2 - 2.* r1 .* r0 .*cosd(th)))*180/pi];
```

```
                actual_gr(1:min(th1-1,length(beam_angle)-th1+1)) =
actual_gr(th1:min(th1-1,length(beam_angle)-th1+1)+th1-1)+th(1:min(th1-
1,length(beam_angle)-th1+1));
                actual_gr(actual_gr==0)= actual_gr(th1)-
beam_angle(actual_gr==0)+beam_angle(th1);

                if roll(ii)>0
                    gr_MRU = grazing(ii)+beam_angle;
                else % roll(ii)<0
                    gr_MRU = grazing(ii)-beam_angle;
                end
            else
                actual_gr = grazing(ii)-beam_angle;

                if roll(ii)>0
                    gr_MRU = grazing(ii)+beam_angle;
                else
                    gr_MRU = grazing(ii)-beam_angle;
                end
            end

            if ~exist('grazing_actual','var')
                grazing_actual = actual_gr;
                MRU_gr = gr_MRU;
                flag_amp = 1==0;
                ranges = range_subset;
            else
                grazing_actual = [grazing_actual actual_gr];
                MRU_gr = [MRU_gr gr_MRU];
                flag_amp = [flag_amp 1==0];
                ranges = [ranges range_subset];
            end
            if n_fft(ii) == 1
                apurar=1;
            end
        else % if there is no zero-crossing
            %% FFT for Amplitude detection
            actual_gr = grazing(ii);
            n_fft(ii) = 1;
            if ~exist('grazing_actual','var')
                grazing_actual = actual_gr;
                ranges = r(target_idx(ii));
                MRU_gr = actual_gr;
                flag_amp = 1==1;
            else
                grazing_actual = [grazing_actual actual_gr];
                ranges = [ranges r(target_idx(ii))];
                MRU_gr = [MRU_gr actual_gr];
                flag_amp = [flag_amp 1==1];
            end

            Fdomain = (0:win_obs)*EK80_data(Channels).fs/win_obs;

            % Tc 70 kHz
```

```
            if
EK80_data(Channels).configdata.transceivers(Channels).channels.transducer.Fre
quency==70000
                corr_r = r(target_idx(ii)-
round(win_obs/2):target_idx(ii)+floor(win_obs/2)).^2;
                fft_bottom_ping =
fft(EK80_data(Channels).echogram_MF(ii,max(1,min(2001-win_obs,TWTT(ii)-
round(win_obs/2))):max(win_obs+1,min(TWTT(ii)+floor(win_obs/2),size(EK80_data
(Channels).echogram_MF,2)))).*corr_r)';
                fstart = 45; fstop = 95;
                FFT_auto =
fft(EK80_data(Channels).auto_tx(floor(length(EK80_data(Channels).auto_tx)/2)-
floor(win_obs/2):win_obs+floor(length(EK80_data(Channels).auto_tx)/2)-
floor(win_obs/2)));

                % for 120kHz
            elseif
EK80_data(Channels).configdata.transceivers(Channels).channels.transducer.Fre
quency==120000
                corr_r = r(target_idx(ii)-
round(win_obs/2):target_idx(ii)+floor(win_obs/2)).^2;
                fft_bottom_ping =
fft(EK80_data(Channels).echogram_MF(ii,max(1,min(2001-win_obs,TWTT(ii)-
round(win_obs/2))):max(win_obs+1,min(TWTT(ii)+floor(win_obs/2),size(EK80_data
(Channels).echogram_MF,2)))).*corr_r)';
                fstart = 90; fstop = 170;
                FFT_auto =
fft(EK80_data(Channels).auto_tx(floor(length(EK80_data(Channels).auto_tx)/2)-
floor(win_obs/2):win_obs+floor(length(EK80_data(Channels).auto_tx)/2)-
floor(win_obs/2)));

                % Tc 200kHz
            elseif
EK80_data(Channels).configdata.transceivers(Channels).channels.transducer.Fre
quency==200000
                corr_r = r(target_idx(ii)-
round(win_obs/2):target_idx(ii)+floor(win_obs/2)).^2;
                fft_bottom_ping =
fftshift(fft(EK80_data(Channels).echogram_MF(ii,max(1,min(2001-
win_obs,TWTT(ii)-
round(win_obs/2))):max(win_obs+1,min(TWTT(ii)+floor(win_obs/2),size(EK80_data
(Channels).echogram_MF,2)))).*corr_r),2)';
                Fdomain = Fdomain +125000;
                fstart = 160; fstop = 260;
                FFT_auto =
fftshift(fft(EK80_data(Channels).auto_tx(floor(length(EK80_data(Channels).aut
o_tx)/2)-
floor(win_obs/2):win_obs+floor(length(EK80_data(Channels).auto_tx)/2)-
floor(win_obs/2))));

                % 333 kHz
            elseif
EK80_data(Channels).configdata.transceivers(Channels).channels.transducer.Fre
quency==333000
                corr_r = r(target_idx(ii)-
round(win_obs/2):target_idx(ii)+floor(win_obs/2)).^2;
```

```
                fft_bottom_ping =
fft(EK80_data(Channels).echogram_MF(ii,max(1,min(2001-win_obs,TWTT(ii)-
round(win_obs/2))):max(win_obs+1,min(TWTT(ii)+floor(win_obs/2),size(EK80_data
(Channels).echogram_MF,2)))).*corr_r)';
                Fdomain=Fdomain+250000;
                fstart = 280; fstop = 450;
                FFT_auto =
fft(EK80_data(Channels).auto_tx(floor(length(EK80_data(Channels).auto_tx)/2)-
floor(win_obs/2):win_obs+floor(length(EK80_data(Channels).auto_tx)/2)-
floor(win_obs/2)));
            end

            fft_bottom_ping=20*log10(abs(fft_bottom_ping./FFT_auto));
            fft_bottom_ping_uncor=fft_bottom_ping;

            beam_angle=0;
        end
        %% FFT bottom
        FFT_bottom{ii} = fft_bottom_ping;
        FFT_bottom_uncor{ii} = fft_bottom_ping_uncor;
        beam_angles{ii} = beam_angle;
    end
    FFT_bottom = [FFT_bottom{:}];
    FFT_bottom_uncor = [FFT_bottom_uncor{:}];
    beam_angles = [beam_angles{:}];
    %% Absorption (frequency dependent)
    absorp_r = (abscoef_freq(temp,sal,ranges',acid,ss,Fdomain).*ranges')';

    %% Correction for Scattered Area
    theta_3dB = calib_file.beam_pattern.bp_3dB*pi/180;
    theta_3dB(theta_3dB==0)=max(theta_3dB);
    theta_3dB =
interp1(calib_file.beam_pattern.freq_domain,theta_3dB,Fdomain,'spline');

    % normal incidence, long pulse
    % Area_normal = pi*tand(theta_3dB').^2 * ranges.^2 ;

    % normal incidence, short pulse
    % Area = pi* c * PL * r

    % Oblique incidence
    % Area_slant = 2*beam_pattern * r * c * PL / (2* sin(grazing)) % the
beam_pattern
    % is multiplied by 2 because beam_pattern was defined as the angle from
the MRA to the -3dB

    PL = (win_obs+1)/EK80_data(Channels).fs* ss; % PL is the length of the
subset. It will be divided by 2 to calculate the area.
    if ~gr_from_phase
        grazing_actual = MRU_gr; % when there is too much noise in the actual
grazing calculation
    else
        grazing_actual((grazing_actual - MRU_gr)>Gr_diflim) =
MRU_gr((grazing_actual - MRU_gr)>Gr_diflim)+Gr_diflim;
        grazing_actual((grazing_actual - MRU_gr)<-Gr_diflim) =
MRU_gr((grazing_actual - MRU_gr)<-Gr_diflim)-Gr_diflim;
```

```
    end

    Area_normal = (tan(2*theta_3dB') * ranges).^2 ;
    Area_slant = tan(theta_3dB') * (2 * ranges * PL/2 ./
cosd(grazing_actual)); % PL is divided by 2 due to TWTT

    Area = min(Area_slant,Area_normal);

    %% Interpolation of Frequency domains and aplication of calibration

F_calib=calib_file.calibration.freq_domain(find(calib_file.calibration.freq_d
omain>=fstart,1):find(calib_file.calibration.freq_domain>=fstop,1));
    calibration =
movmean(calib_file.calibration.correction(find(calib_file.calibration.freq_do
main>=fstart,1)-1:find(calib_file.calibration.freq_domain>=fstop-5,1)),[50]);
    calibration = [calibration repmat(calibration(end),1,length(F_calib)-
length(calibration))];
    calibration = interp1(F_calib,calibration,Fdomain/1000,'spline');

    FFT_bottom = FFT_bottom +... % logarithmic scale
        2*absorp_r - ...

EK80_data(Channels).configdata.transceivers(Channels).channels.transducer.Gai
n(1)...
        + calibration' - 10*log10(Area);

    %% Echograms
    if plots(1)==1
        % mounting the echogram again
        echogram =
NaN(floor(max(r_tvg,[],'all')/750*EK80_data(Channels).fs)+1,numpings);
        for ii=1:numpings
            echogram(depth_ini_idx(ii):depth_ini_idx(ii)-
1+size(EK80_data(Channels).echogram_MF,2),ii) = TS(ii,:);
        end

        figure(1)
        subplot(4,1,Channels)
        imagesc(1:numpings,r,echogram)
        hold on
        %         plot(range(:,3),'c.','MarkerSize',4.5)
        hold off

        map1 = [zeros(101,1),   zeros(101,1),  (.3:.7/100:1)';...
            zeros(101,1),   (0:.9/100:.9)',  (1:-.01:0)'.^.5;...
            (0:.01:1)'.^.5,  ones(101,1)*.9,   zeros(101,1);...
            ones(101,1),    (.9^2:-.9^2/100:0)'.^.5,   zeros(101,1);...
            (1:-.7/100:.3)',  zeros(101,1),    zeros(101,1)  ];

        colormap(map1)

        colorbar('eastoutside')
        ylabel('range (m)')
        xlabel('Ping #')
        title(['Echogram - ', ...
```

```
EK80_data(1).configdata.transceivers(Channels).channels.transducer.Transducer
Name])
        set(gca,'Ydir','reverse')
        clear echogram
        caxis([-100 50])
    end

    %% TS
    if plots(3)==1
        figure(3)
        if Channels==1
            clf;
        end
        subplot(2,4,Channels)
        plot(1:numpings,TS_bottom,'Color',[.8 .8 .8])
        hold on
        TS_select = TS_bottom;
        TS_select(~keep) = NaN;
        plot((1:numpings),TS_select,'b')
        plot([1 numpings],[min(TS_select) min(TS_select)],'Color',[.8 .8
.8],'LineStyle',':')
        title(['TS at Fc-
',num2str(EK80_data(Channels).configdata.transceivers(Channels).channels.tran
sducer.Frequency/1000),' kHz'])
        axis([1 numpings max(TS_bottom)-40 max(TS_bottom)+5])
        hold off
        %% Depth
        subplot(2,4,Channels+4)
        plot(pings_rejected,-foot_print(3,~keep),'.','Color',[.7 .7
.7],'MarkerSize',12)
        hold on
        plot(pings_selected,-foot_print(3,keep),'b.')
        plot(depth_phase,'r.')
        xlabel('ping number')
        ylabel('Depth - Amplitude (blue) | Phase (red)')
        axis([1 numpings -round(mean(foot_print(3,:))/5)*5-5 0])
        hold off
    end
    %% All depths
    cum_nfft=cumsum(n_fft)+floor(n_fft/2);
    cum_nfft_real=cumsum(n_fft);
    depth_phase(depth_phase==0)=NaN;

    if plots(4)==1
        figure(4)
        if Channels==1
            clf;
        end
        subplot(4,1,Channels)
        plot(-ranges.*sind(MRU_gr),'.','Color',[.5 .2 .8])
        hold on
        plot(cum_nfft,depth_phase,'r.','MarkerSize',20)
        plot(cum_nfft,-foot_print(3,:),'b.')
        plot(cum_nfft(pings_rejected),-foot_print(3,~keep),'.','Color',[.7 .7
.7],'MarkerSize',12)
```

```
        amp_detect = -foot_print(3,pings_selected).*flag_amp;
        amp_detect(amp_detect==0)=NaN;

        plot(cum_nfft(pings_selected),amp_detect,'b+')
        xlabel('Subset number')
        title('Depth - Amplitude (blue) | Phase (MRA red and purple) - blue
crosses indicate there was only amplitude detection')
        axis([1 cum_nfft(end) mean(depth_phase(depth_phase>-100))*1.1 0])
        hold off
    end
    %% Bottom 2D
    if plots(5)==1
        figure(5)
        if Channels==1
            clf;
        end
        subplot(2,2,Channels)
        plot(ranges.*cosd(MRU_gr), -ranges.*sind(MRU_gr),'.','Color',[.5 .2
.8]) % all phase detection subsets
        hold on
        plot(r(target_idx).*cosd(grazing),-r(target_idx).*sind(grazing),'b.')
%amplitude detection
        plot(r(target_idx(pings_rejected)).*cosd(grazing(pings_rejected)),-
r(target_idx(pings_rejected)).*sind(grazing(pings_rejected)),'.','Color',[.7
.7 .7],'MarkerSize',12) % rejected amplitude detection
        plot(zero_phase_m.*cosd(grazing),depth_phase,'r.') % phase detection
        ylabel('depth')
        xlabel('Horizontal distance')
        title('2D Bottom - Amplitude (blue) | Phase (MRA red and purple)')
        axis([0 -2*mean(depth_phase(depth_phase>-100))
mean(depth_phase(depth_phase>-100))*1.1 mean(depth_phase(depth_phase>-
100))*.9])
        hold off
    end
    %% Footprint and Attitude
    % plot the points on the seafloor to see how close to a MBES swath, the
tilt on the plate was performed
    if plots(6)==1
        figure(6)
        if Channels==1
            clf;
        end
        subplot(2,4,Channels)
        scatter(foot_print(1,:),foot_print(2,:),'b.')
        title({['Footprint'];...

[EK80_data(1).configdata.transceivers(Channels).channels.transducer.Transduce
rName, ' | Area: ',site]})
        axis('equal')
        axis([-200 200 -200 200])
        xlabel('x range (m)')
        ylabel('y range (m)')

        %% plot Attitude
        % roll
```

```
        subplot(6,4,Channels+12)
        plot(1:numpings,EK80_data(Channels).motion.roll)
        ylabel('Roll')
        axis([1 numpings -90 90])
        yticks(-90:30:90)

        % pitch
        subplot(6,4,Channels+16)
        plot(1:numpings,EK80_data(Channels).motion.pitch)
        ylabel('Pitch')
        axis([1 numpings -90 90])
        yticks(-90:30:90)

        % heading
        subplot(6,4,Channels+20)
        plot(1:numpings,EK80_data(Channels).motion.heading)
        ylabel('Heading')
        xlabel('ping number')
        axis([1 numpings 0 360])
        yticks(0:60:360)
    end

    %% 3D footprint
    heads = repelem((heading(pings_selected)*180/pi),n_fft(pings_selected));
    flag_phases = repelem(~flag_amp,n_fft(pings_selected));
    heads = heads(flag_phases);
    rolls =
repelem((roll(pings_selected)./abs(roll(pings_selected))).*(~flag_amp),n_fft(
pings_selected));
    rolls_ping = roll./abs(roll);
    if plots(7)==1
        figure(7)
        if Channels==1
            clf;
        end
        subplot(2,2,Channels)
        plot3(foot_print(1,~keep),foot_print(2,~keep),-
foot_print(3,~keep),'.','Color',[.7 .7 .7],'MarkerSize',12)
        hold on
        plot3(foot_print(1,pings_selected),foot_print(2,pings_selected),-
foot_print(3,pings_selected),'b.')
        plot3(-
ranges(flag_phases).*cosd(MRU_gr(flag_phases)).*sind(heads).*rolls(flag_phase
s),ranges(flag_phases).*cosd(MRU_gr(flag_phases)).*cosd(heads).*rolls(flag_ph
ases), -ranges(flag_phases).*sind(MRU_gr(flag_phases)),'.','Color',[.5 .2
.8]) % all phase detection subsets
        zlabel('depth')
        xlabel('Horizontal distance (m)')

title({[EK80_data(Channels).configdata.transceivers(Channels).channels.transd
ucer.TransducerName,' - 3D footprint '];['Amplitude (blue) | Phase (purple) |
Rejected (gray)']})% | Zero-crossing (red)']})
        axis([-40 40 -40 40 mean(depth_phase(depth_phase>-100))*1.1 0])
        grid on
        view(35,50)
        hold off
```

```
    end

    %% Plot Phase
    % Plot_phase % not developed

    %% Analyse FFT of the bottom
    % Need to adopt a proper method to get the mean and std of Intensity
    % (Rayleig distribution)
    [grazing_sorted,sort_idx]= sort(grazing_actual);
    FFT_BS = zeros(win_obs+1,90);
    FFT_BS_unc = zeros(win_obs+1,90);
    pos_std_BS = zeros(win_obs+1,90);
    std_BS = zeros(win_obs+1,90);
    neg_std_BS = zeros(win_obs+1,90);

    for gg=1:90
        if sum(grazing_actual>(gg-1)&grazing_actual<=gg)>0
            FFT_BS(:,gg) =
10*log10(mean(10.^((FFT_bottom(:,grazing_actual>(gg-
1)&grazing_actual<=gg)/10)),2,'omitnan'));
            pos_std_BS(:,gg) =
10*log10(+1*std(10.^((FFT_bottom(:,grazing_actual>(gg-
1)&grazing_actual<=gg)/10)),0,2,'omitnan')+mean(10.^((FFT_bottom(:,grazing_ac
tual>(gg-1)&grazing_actual<=gg)/10)),2,'omitnan'));
            std_BS(:,gg) =  std(10.^((FFT_bottom(:,grazing_actual>(gg-
1)&grazing_actual<=gg)/10)),0,2,'omitnan');
            neg_std_BS(:,gg) =  10*log10(-
1*std(10.^((FFT_bottom(:,grazing_actual>(gg-
1)&grazing_actual<=gg)/10)),0,2,'omitnan')+mean(10.^((FFT_bottom(:,grazing_ac
tual>(gg-1)&grazing_actual<=gg)/10)),2,'omitnan'));
            FFT_BS_unc(:,gg) =
10*log10(mean(10.^(FFT_bottom_uncor(:,grazing_actual>(gg-
1)&grazing_actual<=gg)/10),2,'omitnan'));
        else
            FFT_BS(:,gg) = NaN;
            pos_std_BS(:,gg) = NaN;
            neg_std_BS(:,gg) =  NaN;
            FFT_BS_unc(:,gg) = NaN;
            std_BS(:,gg) = NaN;
        end
    end

    FFT_BS(:,1:cut_low_Gr)=NaN;
    pos_std_BS(:,1:cut_low_Gr)=NaN;
    neg_std_BS(:,1:cut_low_Gr)=NaN;
    FFT_BS_unc(:,1:cut_low_Gr)=NaN;
    absorp_r = absorp_r(:,sort_idx);
    %% Final plot of the BOTTOM BS
    if plots(8)==1
        figure(8)
        if Channels==1
            clf;
        end
        hold on
        plot3((1:90),...
```

```
repmat(Fdomain(:,floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,
1,'last')))/1000,90,1),...

movmean(FFT_BS(floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,
'last')),:),[1],2,'omitnan'),'.','MarkerEdgeColor',color_plot(Channels,:))%,'
Linewidth',2)
        hold on
        grid on
        xlabel('Grazing Angle')
        ylabel('Frequency')
        zlabel('BS (dB)')
        title({['Site ',site];['BS (dB)- Angular and frequency response']})
        if Channels == 1
            axis([0 90 0 450 -inf inf])
            max_BS =
max(FFT_BS(floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,'las
t')),:),[],'all');
            zlim([-50 0])
        end
        yticks([0,45,70,120,200,333,450])
        xticks([0 30 60 90])
        view(110,35)
        set(gca,'xdir','reverse')
    end
    %% Final plot of the BOTTOM BS - Freq trend
    if plots(8)==11
        figure(28)
        if Channels==1
            clf;
        end
        hold on


plot3(Fdomain(:,floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1
,'last')))/1000,...

repmat((1:90),size(Fdomain(:,floor(find(Fdomain/1000>=fstart,1):find(Fdomain/
1000<=fstop,1,'last')))/1000,2),1),...

movmean(FFT_BS(floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,
'last')),:),[1],2,'omitnan'),'Color',color_plot(Channels,:))%,'Linewidth',2)
        hold on
        grid on
        ylabel('Grazing Angle')
        xlabel('Frequency')
        zlabel('BS (dB)')
        title('BS (dB)- Frequency trend')
        if Channels == 1
            axis([0 450 0 90 -inf inf])
            max_BS =
max(FFT_BS(floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,'las
t')),:),[],'all');
            zlim([(floor(max_BS/5)+1)*5-25 (floor(max_BS/5)+1)*5])
        end
```

```
        xticks([0,45,70,120,200,333,450])
        yticks([0 30 60 90])
        view(100+90,30)
        set(gca,'ydir','reverse')
        set(gca,'xdir','reverse')
    end
    %% Plot uncorrected FFT
    if plots(9)==1
        figure(9)
        if Channels==1
            clf
        end
        subplot(211)
        FFT_bottom_uncor_sort = FFT_bottom_uncor(:,sort_idx);
        plot3(grazing_sorted,...

repmat(Fdomain(:,floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,
1,'last')))/1000,length(grazing_sorted),1),...

movmean(FFT_bottom_uncor_sort(floor(find(Fdomain/1000>=fstart,1):find(Fdomain
/1000<=fstop,1,'last')),:),[1],2,'omitnan'),'.','Color',color_plot(Channels,:
),'MarkerSize',1)%,'Linewidth',2)
        hold on
        grid on
        xlabel('Grazing Angle')
        ylabel('Frequency')
        zlabel('BS (dB)')
        title({['Site ',site];['Uncorrected BS']})

        if Channels == 1
            axis([0 90 0 450 -inf inf])
            max_BS =
max(FFT_BS(floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,'las
t')),:),[],'all');
            zlim([-100 0])
        end

        yticks([0,45,70,120,200,333,450])
        xticks([0 30 60 90])
        view(110,35)
        set(gca,'xdir','reverse')

        % corrected BS
        subplot(212)
        FFT_bottom_sort = FFT_bottom(:,sort_idx);
        plot3(grazing_sorted,...

repmat(Fdomain(:,floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,
1,'last')))/1000,length(grazing_sorted),1),...

movmean(FFT_bottom_sort(floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<
=fstop,1,'last')),:),[1],2,'omitnan'),'.','Color',color_plot(Channels,:),'Mar
kerSize',1)%,'Linewidth',2)
        hold on
        grid on
        xlabel('Grazing Angle')
```

```
        ylabel('Frequency')
        zlabel('BS (dB)')
        title('Corrected BS (dB)- Angular and frequency response')
        yticks([0,45,70,120,200,333,450])
        xticks([0 30 60 90])
        if Channels == 1
            axis([0 90 0 450 -inf inf])
            max_BS =
max(FFT_BS(floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,'las
t')),:),[],'all');
            zlim([-100 0])
        end

        view(110,35)
        set(gca,'xdir','reverse')
    end
    %% uncorrected FTT (only corrected for range and beam pattern)    if
plots(4)==1
    if plots(10)==1
        figure(10)
        if Channels==1
            clf;
        end

        subplot(221)
        plot3((1:90),...

repmat(Fdomain(:,floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,
1,'last')))/1000,90,1),...

movmean(FFT_BS_unc(floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fsto
p,1,'last')),:),[1],2,'omitnan'),'Color',color_plot(Channels,:))%,'Linewidth'
,2)
        hold on
        grid on
        xlabel('Grazing Angle')
        ylabel('Frequency')
        title('UNCORRECTED TS - Angular and frequency response')
        axis([0 90 0 450 -inf inf])
        yticks([0,45,70,120,200,333,450])
        set(gca,'xdir','reverse')
        if Channels == 1
            max_BS =
max(FFT_BS(floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,'las
t')),:),[],'all');
            zlim([(floor(max_BS/5)+1)*5-25 (floor(max_BS/5)+1)*5])
        end
        view(100,30)

        subplot(222)
        plot3(grazing_sorted,...

repmat(Fdomain(:,floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,
1,'last')))/1000,size(grazing_sorted,2),1),...
```

```
10*log10(Area(floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,'
last')),sort_idx)),'Color',color_plot(Channels,:))%,'Linewidth',2)
        hold on
        grid on
        xlabel('Grazing Angle')
        ylabel('Frequency')
        title('Correction for Area (dB)')
        set(gca,'ydir','reverse')
        axis([0 90 0 450 -inf inf])
        yticks([0,45,70,120,200,333,450])

        subplot(223)
        plot(calib_file.calibration.freq_domain,...
            calib_file.calibration.correction,'Color',[.8 .8
.8])%,'Linewidth',2)
        hold on

plot(Fdomain(:,floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,
'last')))/1000,...

calibration(floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,'la
st'))),'Color',color_plot(Channels,:),'Linewidth',2)
        grid on
        ylabel('Calibration')
        xlabel('Frequency')
        title('MRA - Correction from Sphere Calibration (dB)')
        axis([0 450 0 50])
        xticks([0,45,70,120,200,333,450])

        subplot(224)

plot3(calib_file.beam_pattern.freq_domain(find(calib_file.beam_pattern.freq_d
omain/1000>=fstart,1):find(calib_file.beam_pattern.freq_domain/1000<=fstop,1,
'last'))/1000,...

repmat(calib_file.beam_pattern.beam_angle,size(calib_file.beam_pattern.freq_d
omain(find(calib_file.beam_pattern.freq_domain/1000>=fstart,1):find(calib_fil
e.beam_pattern.freq_domain/1000<=fstop,1,'last')),2),1),...

calib_file.beam_pattern.correction(find(calib_file.beam_pattern.freq_domain/1
000>=fstart,1):find(calib_file.beam_pattern.freq_domain/1000<=fstop,1,'last')
,:),...
            'Color',color_plot(Channels,:))
        hold on
        grid on
        ylabel('Angle within the Beam')
        xlabel('Frequency')
        title('Beam-Pattern Correction from Sphere Calibration (dB)')
        axis([0 450 -inf inf])
        xticks([0,45,70,120,200,333,450])
    end
    %% Reverse angle x freq
    if plots(11)==1
        figure(11)
        if Channels==1
```

```
            clf;
        end

        subplot(211)
        plot3(grazing_sorted,...

repmat(Fdomain(:,floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,
1,'last')))/1000,size(grazing_sorted,2),1),...

2*absorp_r(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,'last'),:)
,'Color',color_plot(Channels,:))
        hold on
        grid on
        xlabel('Grazing Angle')
        ylabel('Frequency')
        title('Correction for absorption (dB)')
        axis([0 90 0 450 -inf inf])
        yticks([0,45,70,120,200,333,450])
        set(gca,'ydir','reverse')
        set(gca,'xdir','reverse')

        subplot(212)

abs_plot=(abscoef_freq(temp,sal,(min(ranges):5:max(ranges))',acid,ss,(45:4:45
0)*1000).*(min(ranges):5:max(ranges))')';

plot3((45:4:450),repmat((min(ranges):5:max(ranges)),size((45:4:450),2),1),2*a
bs_plot,'Color',[.2 .8 .2])
        ylabel('Range (m)')
        xlabel('Frequency')
        title('Correction for absorption (dB)')
        view(40,40)
        xticks([45 70 120 200 333 450])
        grid on
    end
    %% Histogram
    if plots(12)==1
        figure(12)
        if Channels==1
            clf;
        end

        subplot(4,1,Channels)
        histogram(grazing_sorted,(1:90))
        hold on
        histogram(grazing,(1:90))
        hold off
        title(['Histogram of Grazing angles for ',...

EK80_data(1).configdata.transceivers(Channels).channels.transducer.Transducer
Name])
        axis([0 90 0 30])
    end
    %% Standard deviation of BS
    std_BS_dB =  pos_std_BS-FFT_BS;
    if plots(13)==1
```

```
        figure(13)
        if Channels==1
            clf;
        end

        subplot(211)
        plot3((1:90),...

repmat(Fdomain(:,floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,
1,'last')))/1000,90,1),...

movmean(FFT_BS(floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,
'last')),:),[2 2],2,'omitnan'),'Color',[.8 .8 .8])%,'Linewidth',2)
[1/Channels Channels/4 abs(sin(Channels))]
        hold on
        plot3((1:90),...

repmat(Fdomain(:,floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,
1,'last')))/1000,90,1),...
            ...
movmean(2*std_BS(floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,
1,'last')),:)+FFT_BS(floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fs
top,1,'last')),:),[2 2],2,'omitnan')','--
','Color',color_plot(Channels,:))%,'Linewidth',2)

movmean(pos_std_BS(floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fsto
p,1,'last')),:),[2 2],2,'omitnan'),'Color',min([1 1
1],color_plot(Channels,:)+.2),'Linewidth',2)
        hold on
        plot3((1:90),...

repmat(Fdomain(:,floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,
1,'last')))/1000,90,1),...
            ...       movmean(-
2*std_BS(floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,'last'
)),:)+FFT_BS(floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,'l
ast')),:),[2 2],2,'omitnan'),'--
','Color',color_plot(Channels,:))%,'Linewidth',2)

movmean(neg_std_BS(floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fsto
p,1,'last')),:),[2 2],2,'omitnan'),'Color',max([0 0
0],color_plot(Channels,:)-.4),'Linewidth',2)

        grid on
        xlabel('Grazing Angle')
        ylabel('Frequency')
        zlabel('BS (dB)')
        title('BS (dB)- 1std confidence interval')
        axis([0 90 0 450 -inf inf])
        yticks([0,45,70,120,200,260])
        yticks([0,45,70,120,200,333,450])
        xticks([0 30 60 90])
        view(100,30)
        set(gca,'xdir','reverse')

        % only STD
```

```
        subplot(212)
        %     if Channels==1
        %         clf;
        %     end
        plot3((1:90),...

repmat(Fdomain(:,floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,
1,'last')))/1000,90,1),...

std_BS_dB(floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,'last
')),:),'Color',color_plot(Channels,:),'Linewidth',1)
        hold on
        grid on
        xlabel('Grazing Angle')
        ylabel('Frequency')
        zlabel('dB')
        title('10*log[ (1*std+mean) / mean]')
        axis([0 90 0 450 -inf inf])
        yticks([0,45,70,120,200,260])
        yticks([0,45,70,120,200,333,450])
        xticks([0 30 60 90])
        view(100,30)
        zlim([0 7])
        set(gca,'xdir','reverse')
    end
    %% Compare grazing from MRU x phase
    if plots(14)==1
        figure(14)
        if Channels==1
            clf;
        end
        subplot(4,1,Channels)
        plot(grazing_actual,'b')
        hold on
        plot(MRU_gr,'r')

plot(cum_nfft(pings_selected),grazing(pings_selected),'+','MarkerSize',4,'Col
or',[.8 .3 .3])
        plot(cum_nfft,90-abs(roll)*180/pi,'.','Color',[.2 .1
.8],'LineWidth',1)
        axis([1 cum_nfft(end) 0 90])
        title('Grazing by: Phase (blue)  |  MRU + beam (red)  |  MRU at MRA
(cross)  |  90-roll (dark blue)')
        hold off
        xlabel('Ping#')
        ylabel('Grazing')
    end
    %%
    if plots(15)==1
        figure(15)
        subplot(2,2,Channels)
        plot(ranges,grazing_actual,'b.','MarkerSize',1)
        hold on
        plot(ranges,MRU_gr,'r.','MarkerSize',1)
        hold off
```

```
        title(['Grazing by: Phase ramp (blue)  |  MRU (red) |  ', ...

EK80_data(1).configdata.transceivers(Channels).channels.transducer.Transducer
Name])
        xlabel('Target Range')
    end
    %%
    if plots(16)==1
        figure(16)
        subplot(2,2,Channels)
        plot(MRU_gr,MRU_gr,'b',MRU_gr,grazing_actual,'r.','MarkerSize',1)
        xlabel('Gr from MRU+beam')
        ylabel('Gr from Phase ramp')

title(EK80_data(Channels).configdata.transceivers(Channels).channels.transduc
er.TransducerName)
    end
    %% Comparing BS with win_obs =  win_obs, 128, 256 counts
    if plots(17)==1
        figure(17)
        plot3((1:90),...

repmat(Fdomain(:,floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,
1,'last')))/1000,90,1),...

movmean(FFT_BS(floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,
'last')),:),[2 2],2,'omitnan'),...
            'Color',[(256-win_obs)/256 0.5 63/win_obs])%,'Linewidth',2)
        hold on
        grid on
        xlabel('Grazing Angle')
        ylabel('Frequency')
        zlabel('BS (dB)')
        title(['BS (dB) site: ',site])%darker for larger win-obs')

        if Channels == 1
            axis([0 90 0 450 -inf inf])
            max_BS =
max(FFT_BS(floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,'las
t')),:),[],'all');
            zlim([(floor(max_BS/5)+1)*5-25 (floor(max_BS/5)+1)*5])
        end
        yticks([0,45,70,120,200,333,450])
        xticks([0 30 60 90])
        view(100,30)
        set(gca,'xdir','reverse')
    end
    %% Plot freq response for Grazing = 20, 40, 60, 89
    if plots(18)==1
        figure(18)

plot(Fdomain(:,floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,
'last')))/1000,...

(FFT_BS(floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,'last')
```

```
),20)),'Color',[(256-win_obs)/256 0
63/win_obs],'LineStyle',':')%,'Linewidth',2)
        hold on

plot(Fdomain(:,floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,
'last')))/1000,...

(FFT_BS(floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,'last')
),40)),'Color',[(256-win_obs)/256 0 63/win_obs],'LineStyle','-
.')%,'Linewidth',2)

plot(Fdomain(:,floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,
'last')))/1000,...

(FFT_BS(floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,'last')
),60)),'Color',[(256-win_obs)/256 0 63/win_obs],'LineStyle','--
','Linewidth',1)

plot(Fdomain(:,floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,
'last')))/1000,...

(FFT_BS(floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,'last')
),89)),'Color',[(256-win_obs)/256 0 63/win_obs],'Linewidth',2)

        xlabel('Frequency')
        zlabel('BS (dB)')
        title('BS (dB)- Frequency response for Gr = [20, 40, 60, 89] -
thinner to thicker lines')
        axis([45 450 -inf inf])
        xticks([0,45,70,120,200,333,450])
        if Channels == 1
            max_BS =
max(FFT_BS(floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,'las
t')),:),[],'all');
            zlim([(floor(max_BS/5)+1)*5-25 (floor(max_BS/5)+1)*5])
        end
    end
    %% plot several freq response curves
    if plots(19)==1
        figure(19)
        kk=floor((0:19)*89/19)+1;
        for kk1=1:length(kk)
            subplot(4,5,kk1)

plot(Fdomain(:,floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,
'last')))/1000,...

(FFT_BS(floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,'last')
),kk(kk1))),'.','Color',[(256-win_obs)/256 0 63/win_obs])%,'Linewidth',2)
            hold on

errorbar(Fdomain(:,floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fsto
p,1,'last')))/1000,...

(FFT_BS(floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,'last')
),kk(kk1))),...
```

```
std_BS_dB(floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,'last
')),kk(kk1)),'.','Color',[(256-win_obs)/256 0 63/win_obs])%,'Linewidth',2)
            if kk1==1
                xlabel('kHz')
                ylabel('dB')
            end
            title(['Gr=',num2str(kk(kk1))])
            xlim([45 450])
            ylim([-50 0])
        end
    end
    %% plot Mean BS with error bar for several angular response curves
    if plots(20)==1
        figure(20)

kk2=floor(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,'last'));
        for kk=1:length(kk2)
            %     subplot(floor(length(kk2)/5)+1,5,kk)
            subplot(4,length(kk2),(Channels-1)*length(kk2)+kk)
            plot((1:90),...
                (FFT_BS(kk2(kk),:)),'.','Color',[(256-win_obs)/256 0
63/win_obs])%,'Linewidth',2)
            errorbar((1:90),...
                (FFT_BS(kk2(kk),:)),...
                std_BS_dB(kk2(kk),:),'.','Color',[(256-win_obs)/256 0
63/win_obs])%,'Linewidth',2)
            set(gca,'xdir','reverse')
            if kk==1
                xlabel('Gr')
                ylabel('dB')
            end
            title(['f = ',num2str(round(Fdomain(kk2(kk))/1000)),'kHz'])
            ylim([-50 0])
            xlim([0 90])

        end
    end
    %% Store absolute response of seafloor
    Bottom_BS(Channels).site = site;
    Bottom_BS(Channels).Tc_name =
EK80_data(Channels).configdata.transceivers(Channels).channels.transducer.Tra
nsducerName;
    Bottom_BS(Channels).BS =
FFT_BS(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,'last'),:);
    Bottom_BS(Channels).Fdomain =
Fdomain(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,'last'));
    Bottom_BS(Channels).Grazing = (1:90);
    Bottom_BS(Channels).std =
std_BS_dB(find(Fdomain/1000>=fstart,1):find(Fdomain/1000<=fstop,1,'last'),:);
    Bottom_BS(Channels).param.win_obs = win_obs;
    Bottom_BS(Channels).param.allow_HD = allow_HD;
    Bottom_BS(Channels).param.gr_from_phase = gr_from_phase;
    Bottom_BS(Channels).param.Gr_diflim = Gr_diflim;
    Bottom_BS(Channels).param.roll_min = roll_min;
    Bottom_BS(Channels).param.cut_low_Gr = cut_low_Gr;
```

```
end

%% Save absolute response of seafloor
% remove the comment to save the file
% save(site,'Bottom_BS') % save the whole variable Bottom_BS with the name of
the indicated site
```

Plot_sphere.m

```
      %% preliminary coeff
      numpings = size(EK80_data(1).echogram_MF,1);
      depth_ini_idx = EK80_data.idx_ini'-1;
      numsamples = size(EK80_data(1).echogram_MF,2)+max(depth_ini_idx);
      % correct TL for plotting
      r = (1:numsamples)/EK80_data(1).fs*750;
      fc =
EK80_data(1).configdata.transceivers(Channels).channels.transducer.Frequency;
% 200000;%22000; % center frequency
      r_tvg = ((0:size(EK80_data(1).echogram_MF,2)-
1)+repmat(depth_ini_idx,1,size(EK80_data(1).echogram_MF,2)))/EK80_data.fs*750
;

      absorp = zeros(numpings,size(EK80_data(1).echogram_MF,2));
      for ii=1:numpings
          absorp(numpings,:) =
abscoef(temp,sal,r_tvg(ii,:),acid,ss,fc).*r_tvg(ii,:);
      end
      TL_2x = 40*log10(r_tvg) + 2*absorp;
      TS_semi_cor = 20*log10(abs(EK80_data(1).echogram_MF)) + TL_2x;
      %% Plot
      figure(1)

      % echogram
      subplot(211)
      r_tvg2=r_tvg';
      normal_TS = ((TS_semi_cor-
mean(TS_semi_cor,'all','omitnan'))/(max(TS_semi_cor,[],'all')-
min(TS_semi_cor,[],'all')))';
      scatter(repelem((1:numpings),1,size(r_tvg,2)),r_tvg2(:),[],normal_TS(:),
'filled','s')
      colorbar
      map1 = [zeros(101,1),   zeros(101,1),  (.3:.7/100:1)';...
          zeros(101,1),   (0:.9/100:.9)',  (1:-.01:0)'.^.5;...
          (0:.01:1)'.^.5,  ones(101,1)*.9,   zeros(101,1);...
          ones(101,1),    (.9^2:-.9^2/100:0)'.^.5,   zeros(101,1);...
          (1:-.7/100:.3)',  zeros(101,1),    zeros(101,1)  ];

      colormap(map1)
      xlabel('Ping #')
      ylabel('Range (m)')
      axis([1 numpings 1 min(3*depth_end,max(r_tvg2,[],'all'))])
%size(TS_uncor,2)
      title(['Sphere detection - ', ...

EK80_data(1).configdata.transceivers(Channels).channels.transducer.Transducer
Name])
      set(gca,'Ydir','reverse','Color',[.1 0 .3])
      hold on

      [TS_sphere,TWTTsphere]= max(TS_semi_cor,[],2);
      [~,TWTTsphere_linear]= max(TS_semi_cor,[],2,'linear');
```

```
keep = (TS_sphere>max(TS_sphere)-10);
keep = keep&(abs(TS_sphere-mean(TS_sphere))<2*std(TS_sphere));
pings_selected = 1:numpings;
pings_selected=pings_selected(keep);
range=zeros(3,numpings);
range(3,:)=(TWTTsphere+depth_ini_idx)/EK80_data.fs*750;
plot(pings_selected,range(3,keep),'c')
hold off

% targets xy
T_along=EK80_data.echogram_phi_along(TWTTsphere_linear);
T_across=EK80_data.echogram_phi_across(TWTTsphere_linear);
T_phi=EK80_data.echogram_phi_analysis(TWTTsphere_linear);

XY_TS_phi=[T_along(keep),T_across(keep),TS_sphere(keep),T_phi(keep)];

subplot(234)
scatter(XY_TS_phi(:,1),... % along track - axis X
    XY_TS_phi(:,2),... % across track - axis Y
    20,... %size of the circle on the plot
    XY_TS_phi(:,3),'LineWidth',2) % colorcode based on Target Strength
colorbar('southoutside')
xlabel('Along-Track')
ylabel('Across-Track')
axis([-5 5 -5 5])
axis('equal')

% smoothed version of beam forming
fit_beam=fit(XY_TS_phi(:,4),XY_TS_phi(:,3),'poly2');
max_fit=feval(fit_beam,roots(polyder(coeffvalues(fit_beam))));

fit_beam_3db=fit(XY_TS_phi(:,4),XY_TS_phi(:,3)-max_fit+3,'poly2'); % fit
-3dB
std_beam=predint(fit_beam,XY_TS_phi(:,4),0.95,'observation','off');
[sorted_X,sort_idx]=sort(XY_TS_phi(:,4));
std_beam=std_beam(sort_idx,:); % reorganizing the order

% 2D beam pattern collapsed in elevation angles
subplot(2,3,5)
scatter(XY_TS_phi(:,4),... % elevation angle (phi) - axis X
    XY_TS_phi(:,3),... % beam strength - axis Y
    2,... %size of the circle on the plot
    XY_TS_phi(:,3),'LineWidth',.5) % colorcode based on Target Strength
colorbar('southoutside')
hold on
plot([0  3.5],[max_fit max_fit],'r') % max of fit
plot([0  3.5],[max_fit-3 max_fit-3],'r') % -3dB of max of fit
hold off
ylabel('TS')
xlabel('Elevation angle')
legend off

% histogram for elevation angle
subplot(2,3,6)
histogram(XY_TS_phi(:,4),50)
```

```
xlabel('Elevation angle')
ylabel('Histogram')
axis([0 max(XY_TS_phi(:,4)) 0 25])

%% Plot beam pattern of FM pulse
figure(2)
plot(fit([XY_TS_phi(:,1),XY_TS_phi(:,2)],...
    XY_TS_phi(:,3),'poly23')) % fit surface
alpha(0.5)
shading interp
hold on
% plot together with the points
plot3(XY_TS_phi(:,1),... % along track - axis X
    XY_TS_phi(:,2),... % across track - axis Y
    XY_TS_phi(:,3),'.')
hold off
title('Beam pattern of Chirp')
xlabel('Along-Track')
ylabel('Across-Track')
zlabel('Uncor TS (dB)')
axis equal
view(2)
caxis([max(XY_TS_phi(:,3))-3 max(XY_TS_phi(:,3))])
%% Call Plot_fft_sphere cript
Plot_fft_sphere
```

Plot_fft_sphere.m

```
% Fast Fourier Transform of the MF signal of the sphere
% Script called by Plot_sphere

%% New FFT Sphere using original scripts (EK80_data)
% load the correspondent model TS for the site
load('Model_June12.mat')
% load('Model_May16.mat')

before_target = floor(app.BeforeTargetEditField.Value
/750*EK80_data.fs);
after_target = floor(app.AfterTargetEditField.Value /750*EK80_data.fs);
win_obs = min(length(EK80_data.auto_tx),after_target+before_target);

FFT_sphere=zeros(win_obs+1,numpings);

beam_angle = zeros(1,numpings);

Fdomain = (0:win_obs)*EK80_data.fs/win_obs;

% 70 kHz
if
EK80_data.configdata.transceivers(Channels).channels.transducer.Frequency==70
000
    for ii=1:numpings
        corr_r = ((TWTTsphere(ii)-before_target + depth_ini_idx(ii) :
TWTTsphere(ii)+ depth_ini_idx(ii)+after_target)*750/EK80_data.fs)'.^2;
        FFT_sphere(:,ii) =
(fft(EK80_data.echogram_MF(ii,max(1,min(size(EK80_data.echogram_MF,2)-
win_obs,TWTTsphere(ii)-
before_target)):max(win_obs+1,min(TWTTsphere(ii)+after_target,size(EK80_data.
echogram_MF,2))))'.*corr_r));
        beam_angle(ii) =
EK80_data.echogram_phi_analysis(ii,TWTTsphere(ii));
    end
    FFT_sphere = flip(FFT_sphere);
    fstart = 45; fstop = 95;
    FFT_auto = fft(EK80_data.auto_tx(floor(length(EK80_data.auto_tx)/2)-
floor(win_obs/2):win_obs+floor(length(EK80_data.auto_tx)/2)-
floor(win_obs/2)));

    % for 120kHz
elseif
EK80_data.configdata.transceivers(Channels).channels.transducer.Frequency==12
0000
    for ii=1:numpings
        corr_r = ((TWTTsphere(ii)-before_target + depth_ini_idx(ii) :
TWTTsphere(ii)+ depth_ini_idx(ii)+after_target)*750/EK80_data.fs)'.^2;
        FFT_sphere(:,ii) =
(fft(EK80_data.echogram_MF(ii,max(1,min(size(EK80_data.echogram_MF,2)-
win_obs,TWTTsphere(ii)-
before_target)):max(win_obs+1,min(TWTTsphere(ii)+after_target,size(EK80_data.
echogram_MF,2))))'.*corr_r));
```

```
            beam_angle(ii) =
EK80_data.echogram_phi_analysis(ii,TWTTsphere(ii));
        end
        FFT_sphere = flip(FFT_sphere);%/length(FFT_sphere);
        fstart = 90; fstop = 170;
        FFT_auto = fft(EK80_data.auto_tx(floor(length(EK80_data.auto_tx)/2)-
floor(win_obs/2):win_obs+floor(length(EK80_data.auto_tx)/2)-
floor(win_obs/2)));

        % for tc 200kHz
    elseif
EK80_data.configdata.transceivers(Channels).channels.transducer.Frequency==20
0000
        for ii=1:numpings
            corr_r = ((TWTTsphere(ii)-before_target + depth_ini_idx(ii) :
TWTTsphere(ii)+ depth_ini_idx(ii)+after_target)*750/EK80_data.fs)'.^2; %
correction for range in linear domain, not dB
            FFT_sphere(:,ii) =
fftshift(fft(EK80_data.echogram_MF(ii,max(1,min(size(EK80_data.echogram_MF,2)
-win_obs,TWTTsphere(ii)-
before_target)):max(win_obs+1,min(TWTTsphere(ii)+after_target,size(EK80_data.
echogram_MF,2))))'.*corr_r));
            beam_angle(ii) =
EK80_data.echogram_phi_analysis(ii,TWTTsphere(ii));
        end
        FFT_sphere = flip(FFT_sphere);%/length(FFT_sphere);
        Fdomain = Fdomain + Fdomain(end)/2;
        fstart = 160; fstop = 260;
        FFT_auto =
fftshift(fft(EK80_data.auto_tx(floor(length(EK80_data.auto_tx)/2)-
floor(win_obs/2):win_obs+floor(length(EK80_data.auto_tx)/2)-
floor(win_obs/2))));

        % 333 kHz
    elseif
EK80_data.configdata.transceivers(Channels).channels.transducer.Frequency==33
3000
        for ii=1:numpings
            corr_r = ((TWTTsphere(ii)-before_target + depth_ini_idx(ii) :
TWTTsphere(ii)+ depth_ini_idx(ii)+after_target)*750/EK80_data.fs)'.^2;
            FFT_sphere(:,ii) =
(fft(EK80_data.echogram_MF(ii,max(1,min(size(EK80_data.echogram_MF,2)-
win_obs,TWTTsphere(ii)-
before_target)):max(win_obs+1,min(TWTTsphere(ii)+after_target,size(EK80_data.
echogram_MF,2))))'.*corr_r));
            beam_angle(ii) =
EK80_data.echogram_phi_analysis(ii,TWTTsphere(ii));
        end
        Fdomain=Fdomain+Fdomain(end);
        FFT_sphere = flip(FFT_sphere);
        fstart = 280; fstop = 450;
        FFT_auto = fft(EK80_data.auto_tx(floor(length(EK80_data.auto_tx)/2)-
floor(win_obs/2):win_obs+floor(length(EK80_data.auto_tx)/2)-
floor(win_obs/2)));
    end
```

```
      absorp_r=zeros(win_obs+1,numpings);
      for ii=1:numpings
          absorp_r(:,ii) = abscoef_freq(temp,sal,(TWTTsphere(ii)+
depth_ini_idx(ii))*750/EK80_data.fs,acid,ss,Fdomain)*(TWTTsphere(ii)+
depth_ini_idx(ii))*750/EK80_data.fs;
      end

      % FFT_sphere is now Backscatter Strength in dB
      % BS_dB = EL - SL + 2TL - A_dB
      % BS_dB = 20log10(MF) - Gain + 2*(20*log10(r) + abs*r) - 10*log10(A) ;

      FFT_sphere = 20*log10(abs(FFT_sphere./FFT_auto)) +... % logarithmic
scale
          2*absorp_r - ...

EK80_data.configdata.transceivers(Channels).channels.transducer.Gain(1);

      %% Narrowing the sample to the selected pings only
      % variable keep is a boolean defined on mother script
      % Before/After target: ES70(-1.5:1.5) ES120(-0.5:1) ES200(-.15:.7)
      keep_ftt =
(std(FFT_sphere,1,'omitnan')<3*std(std(FFT_sphere,1,'omitnan'),'omitnan')+mea
n(std(FFT_sphere,1,'omitnan'),'omitnan')|std(FFT_sphere,1,'omitnan')>-
3*std(std(FFT_sphere,1,'omitnan'),'omitnan')+mean(std(FFT_sphere,1,'omitnan')
,'omitnan'));
      keep = logical(keep.*keep_ftt');
      FFT_sphere = FFT_sphere(:,keep);
      beam_angle = beam_angle(keep);
      %% Plot the absortion in terms of Freq
      figure(30)
      plot(Fdomain/1000,absorp_r(:,1),'Linewidth',2)
      title('Correction for (Absortion times range) as a function of
frequency')
      xlabel('Frequency')
      ylabel('Correction (dB)')

      %% Plot FFT
      figure(31)
      plot3(Fdomain/1000,...
          repmat(beam_angle,size(Fdomain,2),1),...
          FFT_sphere)
      hold on

      plot3(Model.out.freq,... % freq domain of Model
          zeros(1,size(Model.out.TS,2)),... % boresight angle = 0
          Model.out.TS,'b','LineWidth',1)
      hold off

      title(['FFT from -',num2str(app.BeforeTargetEditField.Value),'m to +'
num2str(app.AfterTargetEditField.Value), 'm of the target, all pings'])
      xlabel('Freq kHz')
      ylabel('Beam angle wrt Boresight')
      zlabel('FFT')
      xlim([fstart fstop])
      view(0,0)
```

```
      %% Plot MF extract
      figure(32)
      for ii=pings_selected((1:50)*floor(length(pings_selected)/50))
          clc;disp(ii)

          plot3((1 :1 +  before_target+after_target)*750/EK80_data.fs,...

repmat(EK80_data.echogram_phi_analysis(ii,TWTTsphere(ii)),size(Fdomain,2),1),
...

smooth(20*log10(abs(EK80_data.echogram_MF(ii,max(1,min(size(EK80_data.echogra
m_MF,2)-win_obs,TWTTsphere(ii)-
before_target)):max(win_obs+1,min(TWTTsphere(ii)+after_target,size(EK80_data.
echogram_MF,2))))))),20))
          hold on
      end
      hold off
      title('Smooth MF extract of a few pings')
      xlabel('Range')
      ylabel('Beam angle')
      zlabel('dB')

      %% Beam pattern
      [sorted_angle,sort_idx]= sort(beam_angle);
      sorted_FFT_sphere = FFT_sphere(:,sort_idx);
      sorted_FFT_sphere = [flip(sorted_FFT_sphere,2) sorted_FFT_sphere];
      sorted_angle2 = [-flip(sorted_angle,2) sorted_angle];
      angles = (0:0.1:7);
      y_fit = zeros(length(Fdomain),length(angles));

      % Correction for beam pattern
      pattern_corr = zeros(length(Fdomain),length(angles)); % correction to
take effect only for the FFT
      beam_pattern_3dB = zeros(1,length(Fdomain));
      beam_pattern_4dB = zeros(1,length(Fdomain));

      for ii=1:length(Fdomain)
          FFT_fit = polyfit(sorted_angle2',sorted_FFT_sphere(ii,:)'...%,2);
              +max(movmean(sorted_FFT_sphere,[10 10],1),[],'all')-
max(movmean(sorted_FFT_sphere(ii,:),[10 10],1)),2);
          y_fit(ii,:) = polyval(FFT_fit,angles);
          pattern_corr(ii,:) = max(y_fit(ii,:))-y_fit(ii,:); %
numpings+1:end);
          if ~isempty(angles(find(pattern_corr(ii,:)>3,1)))
              beam_pattern_3dB(ii) = angles(find(pattern_corr(ii,:)>3,1));
          end

          if ~isempty(angles(find(pattern_corr(ii,:)>4,1)))
              beam_pattern_4dB(ii) = angles(find(pattern_corr(ii,:)>4,1));
          end
      end

      %% correction of the beam pattern
      pattern_corr(isnan(pattern_corr))=0;
      FFT_sphere(:,sort_idx) = FFT_sphere(:,sort_idx) +
interp1(angles,pattern_corr',sorted_angle,'spline')';
```

```
%% Interpolation of Frequency domains
FFT_sphere2 = interp1(Fdomain/1000,FFT_sphere,Model.out.freq,'spline');
figure(34)
plot3(Model.out.freq,...
    repmat(beam_angle,size(Model.out.freq,2),1),...
    FFT_sphere2)
hold on

plot3(Model.out.freq,... % freq domain of Model
    zeros(1,size(Model.out.TS,2)),... % boresight angle = 0
    Model.out.TS,'b','LineWidth',1)
hold off
title('Interpolated corrected for beam pattern FFT')
xlabel('Freq kHz')
ylabel('Beam angle wrt Boresight')
zlabel('FFT')
xlim([fstart fstop])
view(0,0)

%% Annalysis of nulls of the model
figure(35)
plot(Model.out.freq(1:end-1),abs((Model.out.TS(2:end)-
Model.out.TS(1:end-1))))

% smooth the nulls data
nulls_model = [abs(Model.out.TS(2:end)-Model.out.TS(1:end-1))<.1,0==1];
nulls_model(1:end-1)=nulls_model(1:end-1).*nulls_model(2:end);
nulls_model(1:end-1)=nulls_model(1:end-1).*nulls_model(2:end);
nulls_model(2:end)=nulls_model(1:end-1).*nulls_model(2:end);
nulls_model(2:end)=nulls_model(1:end-1).*nulls_model(2:end);
if Channels==3 % null from 382:384 kHz for the ES333
    nulls_model(1:end-1)=nulls_model(1:end-1).*nulls_model(2:end);
    nulls_model(2:end)=nulls_model(1:end-1).*nulls_model(2:end);
    nulls_model(2:end)=nulls_model(1:end-1).*nulls_model(2:end);
else
    if Channels==4 % null from 382:384 kHz for the ES333
        nulls_model(2938:2955)=logical(false);
        for nn=1:17
            nulls_model(1:end-1)=nulls_model(1:end-
1).*nulls_model(2:end);
        end
    end
end
hold on
plot(Model.out.freq,Model.out.TS,'Linewidth',2)
plot(Model.out.freq,Model.out.TS.*(nulls_model))
hold off
title('Nulls that will be not considered to calibrate')
xlabel('Freq kHz')
ylabel('Ideal TS')

%% Difference from the model
figure(36)
angle = 3.5; % only responses from elevation angle within will be
averaged
```

```
      subplot(311)
      % hold on
      plot(Model.out.freq,... % plot the mean

mean(FFT_sphere2(:,beam_angle<angle),2),'r',Model.out.freq,Model.out.TS,'b')
      hold on
      % plot the std
      % std is normal (index 1) when there is more than 30 observations,
      % it considers N-1 when less than 30 obs (index 0)
      plot(Model.out.freq, mean(FFT_sphere2(:,beam_angle<angle),2)...

+1.98*std(FFT_sphere2(:,beam_angle<angle),min(1,floor(sum(beam_angle<angle)/3
0)),2),'r-.',...%'--')%...
          Model.out.freq, mean(FFT_sphere2(:,beam_angle<angle),2)...
          -
1.98*std(FFT_sphere2(:,beam_angle<angle),min(1,floor(sum(beam_angle<angle)/30
)),2),'r-.')
      hold off
      title(['Mean response of the MRA (<',num2str(angle),'deg), ',
num2str(sum(beam_angle<angle)) ' pings used']) %, (std =
',num2str(mean(std(FFT_sphere2(find(fstart*1000<Fdomain,1):find(fstop*1000<Fd
omain,1),beam_angle<angle),min(1,floor(sum(beam_angle<angle)/30)),2))),')'])
      xlabel('Freq (kHz)')
      ylabel('Uncalibrated BS (dB)')
      xlim([fstart fstop])

      calib_BS = (Model.out.TS'-
10*log10(mean(10.^(FFT_sphere2(:,beam_angle<angle)/10),2,'omitnan'))).*nulls_
model';
      edges = logical(nulls_model - [1==1,nulls_model(1:end-
1).*nulls_model(2:end)].*[nulls_model(1:end-1).*nulls_model(2:end),1==1]); %
used to find the edges to interpolate
      edges(1)= true; % to assure the inicial and final values will be
interpolated
      edges(end)= true;
      calib_BS =
~nulls_model.*interp1(Model.out.freq(edges),calib_BS(edges),Model.out.freq)+c
alib_BS'; % interpolation of the gaps
      nulls_model2 = logical(interp1(Model.out.freq,nulls_model+1-
1,Fdomain/1000,'linear','extrap'));
      edges2 = logical(nulls_model2 - [1==1,nulls_model2(1:end-
1).*nulls_model2(2:end)].*[nulls_model2(1:end-1).*nulls_model2(2:end),1==1]);
% used to find the edges to interpolate
      pattern_corr = pattern_corr.*nulls_model2';
      pattern_interp =
~nulls_model2'.*interp1(Fdomain(edges2),pattern_corr(edges2,:),Fdomain);%
interpolation of the gaps
      pattern_interp(isnan(pattern_interp))=0;
      pattern_corr=pattern_interp+pattern_corr;

      beam_pattern_3dB = beam_pattern_3dB.*nulls_model2;
      pattern_interp =
~nulls_model2.*interp1(Fdomain(edges2),beam_pattern_3dB(edges2),Fdomain);%
interpolation of the gaps
      pattern_interp(isnan(pattern_interp))=0;
```

```
      beam_pattern_3dB=pattern_interp+beam_pattern_3dB;
      beam_pattern_4dB = beam_pattern_4dB.*nulls_model2;
      pattern_interp =
~nulls_model2.*interp1(Fdomain(edges2),beam_pattern_4dB(edges2),Fdomain);%
interpolation of the gaps
      pattern_interp(isnan(pattern_interp))=0;
      beam_pattern_4dB=pattern_interp+beam_pattern_4dB;

      subplot(312)
      plot(Model.out.freq,calib_BS,'r')
      hold on
      plot(Model.out.freq,calib_BS,'r.')
      title('Difference to be used to calibrate')
      xlabel('Freq (kHz)')
      xlim([fstart fstop])
      ylabel('Calibration factor (dB)')

      subplot(313)
      plot(Model.out.freq,FFT_sphere2+calib_BS','r')
      hold on
      plot(Model.out.freq,Model.out.TS,'b','Linewidth',2)
      hold off
      title('All calibrated responses')
      xlabel('Freq (kHz)')
      ylabel('Calibrated BS (dB)')
      xlim([fstart fstop])

      %% Beam pattern plot
      figure(33)
      plot3(angles,repmat(Fdomain/1000,size(angles,2),1), ...
          -pattern_corr)
      legend off
      title('Beam Pattern of different frequencies')
      xlabel('Beam angle wrt Boresight')
      ylabel('Freq kHz')
      zlabel('FFT')
      axis([angles(1) angles(end) fstart fstop -inf inf])
      set(gca,'Ydir','reverse')
      view(0,0)
      %% Save calibration file
      calib_file.tc =
EK80_data.configdata.transceivers(Channels).channels.transducer.TransducerNam
e;
      calib_file.fstart = fstart*1000;
      calib_file.fstop = fstop*1000;
      calib_file.calibration.freq_domain = Model.out.freq;
      calib_file.calibration.correction = calib_BS;
      calib_file.beam_pattern.beam_angle = angles;
      calib_file.beam_pattern.freq_domain = Fdomain ;
      calib_file.beam_pattern.correction = pattern_corr;
      calib_file.beam_pattern.bp_3dB = beam_pattern_3dB;
      calib_file.beam_pattern.bp_4dB = beam_pattern_4dB;

      %% Save calib files
      % save(['Sample_Calib_MF_',EK80_data.NMEA(2).text.text(18:19),'_',
EK80_data.NMEA(2).text.text(21:22),'_',...
```

```
      %
EK80_data.configdata.transceivers(Channels).channels.transducer.TransducerNam
e,'.mat'],...
      %     'calib_file')

      % Tank files
      % save(['Tank_autotxNormal_Calib_MF_16MAY_',...
      %
EK80_data.configdata.transceivers(Channels).channels.transducer.TransducerNam
e,'.mat'],...
      %     'calib_file')
```

readrawEK80.m

```
% Reading EK80 raw data file
% Simrad, Lars Nonboe Andersen, 10/10-13
% clear all; close all;
% addpath(genpath('C:\Users\pscor\OneDrive\Documents\MATLAB\EK80'))
% filelocation = 'Z:\users\weidner.elizabeth\EK80';
% [fname,fpath,~] = uigetfile(strcat(filelocation,'\*.raw'),'Select raw
files');
% fname = [fpath fname];
% npingsmax = 1e6;

function [configdata,filterdatavec,sampledatamat,NMEA] =
readrawEK80(fname,npings,file,ChannelID)

headerlength = 12; % Bytes in datagram header
pingno = 0;
pingtime = 0;
nmeaidx = 0; % added EW 6/13

fid = fopen(fname,'r');
if (fid==-1)
    error('Could not open file');
else
    pingno = 0;

    while(1)

        dglength = fread(fid,1,'int32');

        if (feof(fid))
            break
        end

        dgheader = readdgheader(fid);
        %       dgheader.type % LIZ
        switch (dgheader.type)

            case 'XML0' % XML datagram
                xmldata = readxmldata(fid,dglength-headerlength);
                xmldata = parseXML(xmldata);
                %               xmldata.Name %LIZ
                switch (xmldata.Name)

                    case 'Configuration' % Configuration XML data
                        configdata = parseconfxmlstruct(xmldata);
                        %                           nchannels =
length(configdata.transceivers);

                    case 'Environment' % Environment XML data
                        envdata = parseenvxmlstruct(xmldata);

                    case 'Parameter' % Sampledata parameter data
                        if (dgheader.datetime~=pingtime)
```

```
                                    pingtime = dgheader.datetime;
                                    pingno = pingno+1;

                                    clc
                                    home
                                    disp(['Reading file: ' int2str(file) ' from
' configdata.transceivers(ChannelID).channels.transducer.TransducerName   ]);
% ChannelID]) ;
                                    disp(fname);
                                    disp(['Ping no: ' int2str(pingno)]);
                                    %                               disp(['Time: '
datestr(dgheader.datetime)]);
                                end

                                prmdata = parseprmxmlstruct(xmldata);

                            end

                        case 'FIL1' % Filter datagram
                            filterdata = readfilterdata(fid);
                            if ~exist('filterdatavec','var')
                                filterdatavec(1,filterdata.stage) = filterdata;
                            else
                                idx =
find(strcmp(filterdata.channelid,{filterdatavec(:,1).channelid}));
                                if isempty(idx)

filterdatavec(size(filterdatavec,1)+1,filterdata.stage) = filterdata;
                                else
                                    filterdatavec(idx,filterdata.stage) =
filterdata;
                                end
                            end

                        case 'NME0' % NMEA datagram
                            nmea = readtextdata(fid,dglength-headerlength);
                            nmeaidx = nmeaidx + 1; %added EW 6/13
                            NMEA(nmeaidx).text = nmea; % added EW 6/13
                            NMEA(nmeaidx).time = dgheader.datetime; % added EW 6/13

                        case 'TAG0' % Annotation datagram
                            annotation = readtextdata(fid,dglength-headerlength);

                        case 'MRU0' % Motion data
                            motiondata = readmotiondata(fid);

                        case {'RAW0','RAW3'} % Sample datagram

                            if (dgheader.datetime~=pingtime)
                                pingtime = dgheader.datetime;
                                pingno = pingno+1;
                            end

                            sampledata = readsampledata(fid,dgheader.type);
                            if (dgheader.type=='RAW0')
                                channelno = sampledata.channelno;
```

```
                    else
                        channels = [configdata.transceivers(:).channels];
                        channelno   =
find(strcmp(deblank(sampledata.channelid),{channels.ChannelId}));

                        if (~exist('motiondata'))
                            motiondata.heave    = 0;
                            motiondata.roll     = 0;
                            motiondata.pitch    = 0;
                            motiondata.heading  = 0;
                        end

                        sampledata  =
mergesampledata(envdata,motiondata,prmdata,sampledata);

                    end
                    sampledata.time                 = dgheader.datetime;

                    sampledatamat(channelno,pingno) = sampledata;

                otherwise
                    error(strcat('Unknown datagram ''',dgheader.type,''' in
file'));
            end

            dglength = fread(fid,1,'int32');

            if (pingno>npings)
                break
            end
        end

        fclose(fid);
    end

    disp(strcat('Finished reading file'));

    end
```

gettransceiver.m

```
      function transceiver =
gettransceiver(configdata,filterdatavec,sampledatamat,sampledatamatnum,transc
eiverno,pingno,chan)
      %        transceiver =
gettransceiver(configdata,filterdatavec,sampledatamat,1       ,
Channels,     pingno,1);

      %
      sampledata                  = sampledatamat(sampledatamatnum,pingno);

      transceiver.iscw            = ~logical(sampledata.pulseform);

      if transceiver.iscw
          transceiver.fstart          =
configdata.transceivers(transceiverno).channels.transducer.Frequency;
          transceiver.bw              = 0;
      else
          transceiver.fstart          = sampledata.frequency;
          transceiver.bw              =
sampledata.sweep*sampledata.pulselength;
      end

      transceiver.pulselength     = sampledata.pulselength;
      transceiver.txpower         = sampledata.transmitpower;
      transceiver.soundspeed      =
sspeed(sampledata.env.temp,sampledata.env.sal,sampledata.env.depth);

      transceiver.slope           = sampledata.slope;%/100;
      transceiver.gain            =
configdata.transceivers(transceiverno).channels(chan).transducer.Gain;
      transceiver.gain            = transceiver.gain(end);
      transceiver.fnom            =
configdata.transceivers(transceiverno).channels(chan).transducer.Frequency;
      transceiver.alongnom        =
configdata.transceivers(transceiverno).channels(chan).transducer.BeamWidthAlo
ngship;
      transceiver.athwnom         =
configdata.transceivers(transceiverno).channels(chan).transducer.BeamWidthAth
wartship;
      transceiver.psinom          =
configdata.transceivers(transceiverno).channels(chan).transducer.EquivalentBe
amAngle;
      transceiver.alongfactor     =
configdata.transceivers(transceiverno).channels(chan).transducer.AngleSensiti
vityAlongship; % included on 17SEP2019
      transceiver.athwfactor     =
configdata.transceivers(transceiverno).channels(chan).transducer.AngleSensiti
vityAthwartship; % included on 17SEP2019

      transceiver.fstop           = transceiver.fstart+transceiver.bw;
      transceiver.fc              = (transceiver.fstart+transceiver.fstop)/2;
```

```
      transceiver.rximpedance     = 1e3;%5e3; %5.4e3
      transceiver.ztrd            = 75;
      transceiver.fs              = 1.5e6; % determine the sample rate to
build the Tx signal
      % transceiver.fs             = 1e6; % testing different frequency
samples

      % transceiver.stage1         = filterdatavec(sampledatamatnum,1).coeff;
      % transceiver.stage2         = filterdatavec(sampledatamatnum,2).coeff;
      transceiver.stage1          = filterdatavec(transceiverno,1).coeff;
      transceiver.stage2          = filterdatavec(transceiverno,2).coeff;

      % transceiver.dec_fpga       = filterdatavec(sampledatamatnum,1).dec;
      % transceiver.dec_pc         = filterdatavec(sampledatamatnum,2).dec;
      transceiver.dec_fpga        = filterdatavec(transceiverno,1).dec;
      transceiver.dec_pc          = filterdatavec(transceiverno,2).dec;

      transceiver.fsdec           =
round(transceiver.fs/transceiver.dec_fpga/transceiver.dec_pc);
      transceiver.sampleinterval  = 1/transceiver.fsdec;

      transceiver.vrsplit         =
transceiver.rximpedance/(transceiver.rximpedance+transceiver.ztrd);


      end
```

createtx.m

```
function [tx,ytx] = createtx(transceiver)

wbt.fs = transceiver.fs;
wbt.dt = 1/wbt.fs;

ztrd = transceiver.ztrd;

% Create tx signal

tx.amp  = sqrt((transceiver.txpower/4)*(2*ztrd));

tx.t  = (0:wbt.dt:transceiver.pulselength-wbt.dt)';
tx.nt = length(tx.t);

nwtx    = 2*floor(transceiver.slope*tx.nt);
wtxtmp  = hann(nwtx);
nwtxh   = round(nwtx/2);
wtx     = [wtxtmp(1:nwtxh); ones(tx.nt-nwtx,1); wtxtmp(nwtxh+1:end)];

tx.sig =
tx.amp*chirp(tx.t,transceiver.fstart,transceiver.pulselength,transceiver.fsto
p).*wtx;

tx.df   = (transceiver.fstop-transceiver.fstart)/tx.nt;
tx.f    = (0:tx.nt-1)*tx.df + transceiver.fstart;

% Stage 1 - FPGA filter

b_fpga      = transceiver.stage1;

tx.filfpga      = conv(tx.sig/max(abs(tx.sig)),b_fpga);
tx.filfpgadec   = downsample(tx.filfpga,transceiver.dec_fpga);
% tx.filfpgadec   = downsample(tx.filfpga,1); % keep the same signal

% Stage 2 - PC filter

b_pc = transceiver.stage2;

tx.filpc        = conv(tx.filfpgadec,b_pc);
% tx.filpcdec     = downsample(tx.filpc,transceiver.dec_pc);
tx.filpcdec     = downsample(tx.filpc,1); % keep the same signal


% ytx_org = tx.sig;
ytx = tx.filpcdec;
end
```

sspeed.m

```
function c = sspeed(t,s,d)
% c = sspeed(t,s,d);
% Calculates sound speed in m/s as a function of temperature, t,
% salinity, s, and depth, d.

if (s==0)
    % Freshwater
    c = 1402.388 + 5.03711*t - 0.0580852*t^2 + 0.3342e-3*t^3 - 0.1478e-
5*t^4 + 0.315e-8*t^5;
else
    % Salt water
    c = 1448.96 + 4.591*t - 0.05304*t.^2 + 2.374e-4*t.^3 ...
        + 1.34*(s-35) + 0.0163.*d + 1.675e-7*d.^2 ...
        - 0.01025*t.*(s-35) - 7.139e-13*t.*d.^3;
end
```

xmlreadstring.m

```
function [parseResult,p] = xmlreadstring(stringToParse,varargin)
%XMLREADSTRING Modified XMLREAD function to read XML data from a string.
% Author: Luis Cantero.
% The MathWorks.

p = locGetParser(varargin);
locSetEntityResolver(p,varargin);
locSetErrorHandler(p,varargin);

% Parse and return.
parseStringBuffer = java.io.StringBufferInputStream(stringToParse);
parseResult = p.parse(parseStringBuffer);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function p = locGetParser(args)

p = [];
for i=1:length(args)
    if isa(args{i},'javax.xml.parsers.DocumentBuilderFactory')
        javaMethod('setValidating',args{i},locIsValidating(args));
        p = javaMethod('newDocumentBuilder',args{i});
        break;
    elseif isa(args{i},'javax.xml.parsers.DocumentBuilder')
        p = args{i};
        break;
    end
end

if isempty(p)
    parserFactory = javaMethod('newInstance',...
        'javax.xml.parsers.DocumentBuilderFactory');

    javaMethod('setValidating',parserFactory,locIsValidating(args));
    %javaMethod('setIgnoringElementContentWhitespace',parserFactory,1);
    %ignorable whitespace requires a validating parser and a content
model
    p = javaMethod('newDocumentBuilder',parserFactory);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function tf=locIsValidating(args)

tf=any(strcmp(args,'-validating'));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function locSetEntityResolver(p,args)

for i=1:length(args)
    if isa(args{i},'org.xml.sax.EntityResolver')
        p.setEntityResolver(args{i});
        break;
```

```
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function locSetErrorHandler(p,args)

for i=1:length(args)
    if isa(args{i},'org.xml.sax.ErrorHandler')
        p.setErrorHandler(args{i});
        break;
    end
end
```

readxmldata.m

```
% Reading EK80 raw data file XML data
% Simrad, Lars Nonboe Andersen, 10/10-10
%
% modified by TW: 1/6/2015
%     getting this error:[Fatal Error] :13:17: Content is not allowed in
trailing section.
%     modifying this to get rid of content after the last >
%       added the following two lines:
%           idx = strfind(textdata,'>');
%           textdata = textdata(1:idx(end));

function xmldata = readxmldata(fid,length)
% length in bytes

textdata = char(fread(fid,length,'char')');

idx = strfind(textdata,'>');
textdata = textdata(1:idx(end));

[parseResult,p] = xmlreadstring(textdata);

xmldata = parseResult;
```

readtextdata.m

```
% Reading EK80 raw data file text data
% Simrad, Lars Nonboe Andersen, 10/10-13

function textdata = readtextdata(fid,length)
% length in bytes

textdata.text = char(fread(fid,length,'char')');
```

readsampledata.m

```
% Reading EK80 raw data file sample data
% Simrad, Lars Nonboe Andersen, 10/10-13

function sampledata = readsampledata(fid,rawx)

if strcmp(rawx,'RAW0')

    sampledata.channelno                = fread(fid,1,'int16');
    sampledata.mode_low                 = fread(fid,1,'int8');
    sampledata.mode_high                = fread(fid,1,'int8');
    sampledata.mode                     = 256*sampledata.mode_high +
sampledata.mode_low;
    sampledata.transducerdepth          = fread(fid,1,'float32');
    sampledata.frequency                = fread(fid,1,'float32');
    sampledata.transmitpower            = fread(fid,1,'float32');
    sampledata.pulselength              = fread(fid,1,'float32');
    sampledata.bandwidth                = fread(fid,1,'float32');
    sampledata.sampleinterval           = fread(fid,1,'float32');
    sampledata.soundspeed               = fread(fid,1,'float32');
    sampledata.absorptioncoefficient    = fread(fid,1,'float32');
    sampledata.heave                    = fread(fid,1,'float32');
    sampledata.roll                     = fread(fid,1,'float32');
    sampledata.pitch                    = fread(fid,1,'float32');
    sampledata.temperature              = fread(fid,1,'float32');
    sampledata.heading                  = fread(fid,1,'float32');
    sampledata.transmitmode             = fread(fid,1,'int8');
    sampledata.pulseform                = fread(fid,1,'int8');
    sampledata.slope                    = fread(fid,1,'int8');
    sampledata.spare1                   = char(fread(fid,1,'char')');
    sampledata.sweep                    = fread(fid,1,'float32');

elseif strcmp(rawx,'RAW3')
    sampledata.channelid                = char(fread(fid,128,'char')');
    sampledata.mode_low                 = fread(fid,1,'int8');
    sampledata.mode_high                = fread(fid,1,'int8');
    sampledata.mode                     = 256*sampledata.mode_high +
sampledata.mode_low;
    sampledata.spare1                   = char(fread(fid,2,'char')');
else

    error(strcat('Unknown sample datagram ''',rawx,''' in file'));

end

sampledata.offset                   = fread(fid,1,'int32');
sampledata.count                    = fread(fid,1,'int32');
if (sampledata.mode_low<4)
    power                           =
fread(fid,sampledata.count,'int16');
    sampledata.complexsamples.power                 =
power*10*log10(2)/256;          %LIZ
    if (sampledata.mode_low==3)
```

```
            angle                           = fread(fid,[2
sampledata.count],'int8');
            sampledata.complexsamples.angle           = angle(1,:) +
angle(2,:)*256;   %LIZ
            sampledata.complexsamples.alongship       = angle(2,:)';
%LIZ
            sampledata.complexsamples.athwartship     = angle(1,:)';
%LIZ
        end
    elseif (sampledata.mode_low==8)
        ncomplex                        = sampledata.mode_high;
        complexsamples                  =
fread(fid,2*ncomplex*sampledata.count,'float32=>single');
        complexsamples                  = reshape(complexsamples,[2 ncomplex
sampledata.count]);
        sampledata.complexsamples       =
(squeeze(complex(complexsamples(1,:,:),complexsamples(2,:,:)))).'; % is there
a way to improve this line?
    else
        error('Unknown sample mode');
    end
```

readmotiondata.m

```
% Read EK80 raw data file motion datagram
% Simrad, Lars Nonboe Andersen, 10/10-13

function motiondata = readmotiondata(fid)

motiondata.heave    = fread(fid,1,'float32');
motiondata.roll     = fread(fid,1,'float32');
motiondata.pitch    = fread(fid,1,'float32');
motiondata.heading  = fread(fid,1,'float32');

end
```

readfilterdata.m

```
% Read EK80 raw data file filter datagram
% Simrad, Lars Nonboe Andersen, 10/10-13

function filterdata = readfilterdata(fid)

filterdata.stage = fread(fid,1,'int16');

filterdata.channelno = fread(fid,1,'int16');

filterdata.channelid = char(fread(fid,128,'char')');

filterdata.ncoeff = fread(fid,1,'uint16');
filterdata.dec = fread(fid,1,'uint16');

coeff = fread(fid,2*filterdata.ncoeff,'float32=>single');
coeff = reshape(coeff,[2 filterdata.ncoeff]);
coeff = (squeeze(complex(coeff(1,:),coeff(2,:)))).';

filterdata.coeff = coeff;

end
```

readdgheader.m

```
% Reading EK80 raw data file datagram header
% Simrad, Lars Nonboe Andersen, 10/10-13

function dgheader = readdgheader(fid)

dgheader.type = char(fread(fid,4,'char')');

dgheader.lowdatetime = fread(fid,1,'uint32');
dgheader.highdatetime = fread(fid,1,'uint32');

dgheader.datetime = NTTime2Mlab(dgheader.highdatetime*2^32 +
dgheader.lowdatetime);
```

parseXML.m

```
function theStruct = parseXML(tree)
% PARSEXML Convert XML DOM to a MATLAB structure.

% Recurse over child nodes. This could run into problems
% with very deeply nested trees.
try
    removeIndentNodes( tree.getChildNodes );
    theStruct = parseChildNodes(tree);
catch
    error('Unable to parse XML DOM');
end


% ----- Subfunction PARSECHILDNODES -----
function children = parseChildNodes(theNode)
% Recurse over node children.
children = [];
if theNode.hasChildNodes
    childNodes = theNode.getChildNodes;
    numChildNodes = childNodes.getLength;
    allocCell = cell(1, numChildNodes);

    children = struct(             ...
        'Name', allocCell, 'Attributes', allocCell,    ...
        'Data', allocCell, 'Children', allocCell);

    for count = 1:numChildNodes
        theChild = childNodes.item(count-1);
        children(count) = makeStructFromNode(theChild);
    end
end

% ----- Subfunction MAKESTRUCTFROMNODE -----
function nodeStruct = makeStructFromNode(theNode)
% Create structure of node info.

nodeStruct = struct(                        ...
    'Name', char(theNode.getNodeName),       ...
    'Attributes', parseAttributes(theNode),  ...
    'Data', '',                              ...
    'Children', parseChildNodes(theNode));

if any(strcmp(methods(theNode), 'getData'))
    nodeStruct.Data = char(theNode.getData);
else
    nodeStruct.Data = '';
end

% ----- Subfunction PARSEATTRIBUTES -----
function attributes = parseAttributes(theNode)
% Create attributes structure.
```

```
attributes = [];
if theNode.hasAttributes
    theAttributes = theNode.getAttributes;
    numAttributes = theAttributes.getLength;
    allocCell = cell(1, numAttributes);
    attributes = struct('Name', allocCell, 'Value', ...
        allocCell);

    for count = 1:numAttributes
        attrib = theAttributes.item(count-1);
        attributes(count).Name = char(attrib.getName);
        attributes(count).Value = char(attrib.getValue);
    end
end

% ----- Subfunction REMOVEINDENTNODES -----
function removeIndentNodes( childNodes )

numNodes = childNodes.getLength;
remList = [];
for i = numNodes:-1:1
    theChild = childNodes.item(i-1);
    if (theChild.hasChildNodes)
        removeIndentNodes(theChild.getChildNodes);
    else
        if ( theChild.getNodeType == theChild.TEXT_NODE && ...
                ~isempty(char(theChild.getData()))         && ...
                all(isspace(char(theChild.getData()))))
            remList(end+1) = i-1; % java indexing
        end
    end
end
for i = 1:length(remList)
    childNodes.removeChild(childNodes.item(remList(i)));
end
```

parseprmxmlstruct.m

```
% Parse Parameter XML data to a Parameter Matlab structure
% Simrad, Lars Nonboe Andersen, 10/10-13

function prm = parseprmxmlstruct(xmldata)

nattributes = length(xmldata.Children(1).Attributes);

for i = 1:nattributes
    prmxml.(xmldata.Children(1).Attributes(i).Name) =
xmldata.Children(1).Attributes(i).Value;
end

prm.channelid       = prmxml.ChannelID;
prm.channelmode     = str2double(prmxml.ChannelMode);
prm.txpower         = str2double(prmxml.TransmitPower);
prm.pulseform       = str2double(prmxml.PulseForm);
prm.pulselength     = str2double(prmxml.PulseDuration);
prm.slope           = str2double(prmxml.Slope);
if prm.slope == 0
    prm.fstart          = str2double(prmxml.Frequency);
    prm.fstop           = str2double(prmxml.Frequency);
else
    %prm.bandwidth       = str2double(prmxml.BandWidth);
    prm.fstart          = str2double(prmxml.FrequencyStart);
    prm.fstop           = str2double(prmxml.FrequencyEnd);
end
prm.sampleinterval  = str2double(prmxml.SampleInterval);
prm.transducerdepth = 4.42;
```

parseenvxmlstruct.m

```
% Parse Environment XML data to an Environment Matlab structure
% Simrad, Lars Nonboe Andersen, 10/10-13

function env = parseenvxmlstruct(xmldata)

nattributes = length(xmldata.Attributes);

for i = 1:nattributes,
    envxml.(xmldata.Attributes(i).Name) = xmldata.Attributes(i).Value;
end

%env.temp    = str2num(envxml.Temperature);
env.temp    = 20;
env.sal     = 35;
env.depth   = 35;
env.acid    =8;
```

parseconfxmlstruct.m

```
% Parse Configuration XML data to a Configuration Matlab structure
% Simrad, Lars Nonboe Andersen, 10/10-13

function config = parseconfxmlstruct(xmldata)

% Header

headeridx = find(strcmp({xmldata.Children.Name},'Header'));
headerxml = xmldata.Children(headeridx);

nattributes = length(headerxml.Attributes);

for i = 1:nattributes,
    header.(headerxml.Attributes(i).Name) =
headerxml.Attributes(i).Value;
end

transceiversidx = find(strcmp({xmldata.Children.Name},'Transceivers'));
transceiversxml = xmldata.Children(transceiversidx);

ntransceivers = length(transceiversxml.Children);

for i = 1:ntransceivers,
    transceiverxml = transceiversxml.Children(i);
    transceivers(i).id =  transceiverxml.Name;
    nattributes = length(transceiverxml.Attributes);
    for j = 1:nattributes,
        transceivers(i).(transceiverxml.Attributes(j).Name) =
transceiverxml.Attributes(j).Value;
    end

    channelsxml = transceiverxml.Children;
    nchannels = length(channelsxml.Children);
    channels = [];
    for j = 1:nchannels,
        channelxml = channelsxml.Children(j);
        channels(j).Name = channelxml.Name;
        nattributes = length(channelxml.Attributes);
        for k = 1:nattributes,
            channels(j).(channelxml.Attributes(k).Name) =
channelxml.Attributes(k).Value;
        end

        transducer = [];
        transducerxml = channelxml.Children;
        nattributes = length(transducerxml.Attributes);
        for m = 1:nattributes,
            transducer.(transducerxml.Attributes(m).Name) =
transducerxml.Attributes(m).Value;
        end
        channels(j).transducer = transducer;
    end
```

```
            transceivers(i).channels = channels;
        end

        configxml.header = header;
        configxml.transceivers = transceivers;

        % Header
        config.header.TimeBias = str2num(configxml.header.TimeBias);

        % Transceivers
        for i = 1:ntransceivers,
            config.transceivers(i).SerialNumber =
str2num(configxml.transceivers(i).SerialNumber);
            nchannels = length(configxml.transceivers(i).channels);
            channels = [];
            for j = 1:nchannels,
                channels(j).ChannelId =
configxml.transceivers(i).channels(j).ChannelID;
                %        channels(j).ChannelId =
configxml.transceivers(i).channels(j).ChannelIdLong;
                transducer = [];
                transducer.AngleSensitivityAlongship    =
str2num(configxml.transceivers(i).channels(j).transducer.AngleSensitivityAlon
gship);
                transducer.AngleSensitivityAthwartship  =
str2num(configxml.transceivers(i).channels(j).transducer.AngleSensitivityAthw
artship);
                transducer.BeamWidthAlongship           =
str2num(configxml.transceivers(i).channels(j).transducer.BeamWidthAlongship);
                transducer.BeamWidthAthwartship         =
str2num(configxml.transceivers(i).channels(j).transducer.BeamWidthAthwartship
);
                transducer.EquivalentBeamAngle          =
str2num(configxml.transceivers(i).channels(j).transducer.EquivalentBeamAngle)
;
                transducer.Frequency                    =
str2num(configxml.transceivers(i).channels(j).transducer.Frequency);
                transducer.Gain                         =
str2num(configxml.transceivers(i).channels(j).transducer.Gain);
                transducer.TransducerName               =
configxml.transceivers(i).channels(j).transducer.TransducerName;
                channels(j).transducer = transducer;
            end
            config.transceivers(i).channels = channels;
        end
```

NTTime2Mlab.m

```
      function mtime=NTTime2Mlab(NTTime)
      % NTTime2Mlab - converts from NT time to matlab serial time as returned
from datenum
      %   mtime=NTTime2Mlab(NTTime) - Converts the NTTime vector containing
time in NT format
      %                               to matlab serial time.Output can be used
directly into datestr
      % Ruben Patel IMR


      import java.util.GregorianCalendar;
      import java.util.TimeZone;
      import java.text.SimpleDateFormat;
      import java.sql.Timestamp;

      cal = GregorianCalendar;

      NT_START_CAL=GregorianCalendar(1601, 0, 1);
      GMT = TimeZone.getTimeZone('GMT');
      NT_START_CAL.setTimeZone(GMT);
      NT_START_DATE= -
11644473600000.0;%NT_START_CAL.get(NT_START_CAL.MILLISECOND);
      OUT_DATE_FORMAT = SimpleDateFormat('dd-MMM-yyyy HH:mm:ss');
      OUT_DATE_FORMAT.setTimeZone(GMT);
      for i=1:length(NTTime)
          mlabMilli=(NTTime(i)/10000.0)+NT_START_DATE;
          date = Timestamp(mlabMilli);
          cal.setTime(date);

mtime(i)=datenum(cal.get(cal.YEAR),cal.get(cal.MONTH)+1,cal.get(cal.DAY_OF_MO
NTH),cal.get(cal.HOUR_OF_DAY),cal.get(cal.MINUTE),cal.get(cal.SECOND)+date.ge
tNanos*1e-9);
      end
```

mergesampledata.m

```
function sampledata  = mergesampledata(env,motion,prm,sampledata)

% sampledata.temp     = env.temp;
% sampledata.sal      = env.sal;
% sampledata.depth    = env.depth;
% sampledata.acid     = env.acid;
sampledata.env        = env;

sampledata.heave    = motion.heave;
sampledata.roll     = motion.roll;
sampledata.pitch    = motion.pitch;
sampledata.heading  = motion.heading;

sampledata.channelid        = prm.channelid;
sampledata.transmitmode     = prm.channelmode;
sampledata.transmitpower    = prm.txpower;
sampledata.pulseform        = prm.pulseform;
sampledata.pulselength      = prm.pulselength;
sampledata.slope            = prm.slope;
sampledata.sweep            = (prm.fstop-prm.fstart)/prm.pulselength;
sampledata.frequency        = prm.fstart;
sampledata.sampleinterval   = prm.sampleinterval;
sampledata.transducerdepth  = prm.transducerdepth;
```

abscoef_freq.m

```
function a = abscoef_freq(t,s,d,ph,c,Fdomain)
% Gives the absortion coefficient for each frequency of the Fdomain
% Calculates absorption coefficient in dB/m as a function of
temperature, t,
% salinity, s, depth, d, acidity, p, and sound speed, c.

% Absorption coefficient

Fdomain = Fdomain/1e3;

a1 = (8.86./c)*10.^(0.78*ph-5);
p1 = 1;
f1 = 2.8*(s/35).^0.5.*10.^(4-1245./(t+273));

a2 = 21.44.*(s./c).*(1+0.025*t);
p2 = 1 - 1.37e-4*d + 6.62e-9.*d.^2;
f2 = 8.17*10.^(8-1990./(t+273))./(1+0.0018*(s-35));

p3 = 1 - 3.83e-5*d + 4.9e-10*d.^2;

a3l = 4.937e-4 - 2.59e-5*t + 9.11e-7*t.^2 - 1.5e-8*t.^3;
a3h = 3.964e-4 - 1.146e-5*t + 1.45e-7*t.^2 - 6.5e-10*t.^3;
a3 = a3l.*(t<=20) + a3h.*(t>20);

a = Fdomain.^2.*( a1.*p1.*f1./(f1.^2+Fdomain.^2) +
a2.*p2.*f2./(f2.^2+Fdomain.^2) + a3.*p3 );

a = a/1e3;
```

abscoef.m

```
function a = abscoef(t,s,d,ph,c,f)
% Calculates absorption coefficient in dB/m as a function of
temperature, t,
% salinity, s, depth, d, acidity, p, and sound speed, c.

% Absorption coefficient

f = f/1e3;

a1 = (8.86./c)*10.^(0.78*ph-5);
p1 = 1;
f1 = 2.8*(s/35).^0.5.*10.^(4-1245./(t+273));

a2 = 21.44.*(s./c).*(1+0.025*t);
p2 = 1 - 1.37e-4*d + 6.62e-9.*d.^2;
f2 = 8.17*10.^(8-1990./(t+273))./(1+0.0018*(s-35));

p3 = 1 - 3.83e-5*d + 4.9e-10*d.^2;

a3l = 4.937e-4 - 2.59e-5*t + 9.11e-7*t.^2 - 1.5e-8*t.^3;
a3h = 3.964e-4 - 1.146e-5*t + 1.45e-7*t.^2 - 6.5e-10*t.^3;
a3 = a3l.*(t<=20) + a3h.*(t>20);

a = f.^2.*( a1.*p1.*f1./(f1.^2+f.^2) + a2.*p2.*f2./(f2.^2+f.^2) + a3.*p3
);

a = a/1e3;
```