In Situ Sensing to Enable the 2010 Thermodynamic Equation of Seawater

by

Del Thomas Dakin
BSc, Royal Roads Military College, 1985

A Dissertation Submitted in Partial Fulfillment
of the Requirements for the Degree of

Doctor of Philosophy

in the School of Earth and Ocean Sciences

© Del Thomas Dakin, 2016
University of Victoria

All rights reserved. This dissertation may not be reproduced in whole or in part, by photocopy or other means, without the permission of the author.

Supervisory Committee

In Situ Sensing to Enable the 2010 Thermodynamic Equation of Seawater

by

Del Thomas Dakin
BSc, Royal Roads Military College, 1985

Supervisory Committee

Dr. Stan E. Dosso, School of Earth and Ocean Sciences
Co-Supervisor

Dr. Svein Vagle, School of Earth and Ocean Sciences
Co-Supervisor

Dr. Jay T. Cullen, School of Earth and Ocean Sciences
Departmental Member

Dr. Adam Zielinski, Department of Electrical and Computer Engineering
Outside Member

Abstract

The thermodynamic equation of seawater - 2010 (TEOS-10) is hampered by the inability to measure absolute salinity or density in situ. No new advances for in situ salinity or density measurement have taken place since the adoption of the practical salinity scale in 1978. In this thesis three possible technologies for in situ measurements are developed and assessed: phased conductivity, an in situ density sensor and sound speed sensors. Of these, only sound speed sensors showed the potential for an in situ TEOS-10 measurement solution. To be implemented, sensor response times need to be matched and the sound speed sensor accuracy must be improved. Sound speed sensor accuracy is primarily limited by the calibration reference, pure water. Test results indicate the TEOS-10 sound speed coefficients may also need to be improved. A calibration system to improve sound speed sensor accuracy and verify the TEOS-10 coefficients is discussed.

Keywords: seawater density, water density, acoustic impedance, absolute salinity, sound speed, sound speed standard, sound speed calibration, phased conductivity, TEOS-10

Table of Contents

List of Tables

List of Figures

Acknowledgments

The research on the sound speed standard and phased conductivity portions of the work described herein were partially funded by AML Oceanographic, 2071 Malaview Ave. W, Sidney, BC, Canada, V8L 5X6. The remainder of the research was funded by the author.

Chapter 1

Introduction

1.1 Requirement for a new physical oceanography sensing technique

A fundamental property in physical oceanography is the density of seawater. Seawater density differences, under the force of gravity, cause the movement of one parcel of seawater relative to another. Denser seawater sinks, less-dense seawater floats. This process governs local and global ocean circulation. Thus density is a primary forcing mechanism for heat transport, saline transport, nutrient transport and sediment transport.

Presently, *in situ* density data are derived from conductivity, temperature and depth (CTD) instrument measurements. CTD measurements are used to derive thermodynamic properties (such as density, sound speed, specific heat, etc.) using the, empirically derived, Equations of State of seawater 1980 (EOS-80) (Fofonoff and Millard, 1983). The fundamental parameter calculated for EOS-80 is practical salinity S_P. Practical salinity is a ratio of the seawater electrical conductivity to the conductivity of a standard solution of KCl. Due to the inherent limitations of practical salinity and the EOS-80 equations, the United Nations Educational, Scientific and Cultural Organization (UNESCO) released a new equation, the Thermodynamic Equations of State 2010 (TEOS-10) (IOC, SCOR and IAPSO, 2010). TEOS-10 solves many problems associated with EOS-80 but has a serious *in situ* sensing shortcoming since TEOS-10 relies on absolute salinity, S_A, instead of practical salinity. Absolute salinity is defined as the mass fraction of dissolved material in seawater. There is presently no *in situ* sensor capable of providing absolute salinity measurements, which makes the practical implementation of TEOS-10 problematic.

1.2 Requirement for TEOS-10 and absolute salinity

In 1981 UNESCO adopted the EOS-80 as the metrology standard for measurements in marine sciences. EOS-80 is a set of empirical equations relating several thermodynamic parameters of seawater. Even at the time of adoption there were several known inconsistencies with the EOS-80 methodology. These inconsistencies include:

- There were multiple empirical equations available to compute the same parameters; for example, the Naval Research Laboratory (NRL) II (Del Grosso 1974) and Chen and Millero (1977) sound speed equations. In some cases, the UNESCO equation choice was not universally considered to be the most accurate under all conditions (Dushaw, 1993).
- The empirical equations were not reversible, i.e. the use of a calculated parameter to recalculate the initial conditions introduced errors.
- The adoption of the practical salinity scale, based on conductivity, temperature and pressure (CTD) measurements, was based on the assumption of a constant ratio of salts in seawater and estuaries worldwide, which is not the case.

The UNESCO Thermodynamic Equation of State for seawater 2010 was developed to address these inconsistencies. TEOS-10 is a Gibbs free energy function (McDougall et al., 2009). At a given absolute salinity S_A, temperature T, and pressure p, the specific Gibbs free energy $g(S_A,T,p)$ relates the specific enthalpy h and specific entropy s of seawater in the form, $g = h(S_A,T,p) - (273.15 + T)\, s(S_A,T,p)$. The seawater Gibbs function is the sum of the pure water Gibbs function and the saline Gibbs function $g(S_A,T,p) = g^W(T,p) + g^S(S_A,T,p)$. TEOS-10 is a single equation that can be mathematically manipulated, via partial differentiation, to yield a variety of thermodynamic parameters such as specific volume, sound speed, absolute salinity, temperature, thermal expansion coefficient, boiling point, freezing point, etc. The single

equation is reversible (i.e., one can compute a parameter, such as sound speed *c*, from some starting conditions S_A, temperature *T* and pressure *p*, then use the computed parameter in a computation to reproduce the original starting conditions) whereas the EOS-80 equations are not.

TEOS-10 uses absolute salinity S_A, instead of practical salinity S_P. This distinction is important since it resolves the majority of errors resulting from the assumption of constant salt ratios in seawater worldwide, but it comes at the cost of ease of measurement. Presently, oceanographers have no way to measure the absolute salinity of seawater *in situ*. As an interim approach, the TEOS-10 Manual (pp 13, 14) recommends the following methodology:

- take CTD measurements in the ocean,
- take a water sample,
- use a laboratory densimeter to determine the absolute salinity of the water sample,
- compute the density anomaly of the CTD computed salinity relative to the sample,
- record the CTD salinity and the density anomaly.

This method is labour intensive, requires extreme care in sample handling, and is expensive. These additional resources are only required if the seawater salinity is not of Reference Composition (Millero et al., 2008), such as the North Pacific, enclosed seas and estuarine waters. Millero et al. (2008) define the Reference Composition of sea water as the exact mole fractions given in Table 1.1. Reference Composition is representative of filtered, surface, north Atlantic sea water. The mass fraction column shows the relative contribution to S_A by the constituents. The mole fraction multiplied by the valence column provides an approximate relative impact on the seawater conductivity and shows the Na^+ and Cl^- ions dominate the impact on conductivity.

Table 1.1 *Reference Composition of sea water.*

Solute	Valence	Molar mass g mol^{-1}	Mole fraction x 10^{-7}	Mole fraction x valence x 10^{-7}	Mass fraction
Na^+	1	22.990	4188071.5	4188071	0.306596
$Mg2^+$	2	24.305	471677.6	943356	0.036506
Ca_2^+	2	40.078	91822.9	183646	0.011719
K^+	1	39.098	91158.8	91159	0.011349
Sr_2^+	2	87.62	809.6	1620	0.000226
Cl^-	-1	35.453	4874838.9	-4874839	0.550340
SO_4^{2-}	-2	96.063	252152.4	-504304	0.077132
HCO_3^-	-1	61.017	15340.4	-15340	0.002981
Br^-	-1	79.904	7520.1	-7520	0.001913
CO_3^{2-}	-2	60.009	2133.6	-4268	0.000408
$B(OH)_4^-$	-1	78.844	899.8	-900	0.000226
F^-	-1	18.998	610.2	-610	0.000037
OH^-	-1	17.007	71.2	-71	0.000004
$B(OH)_3$	0	61.833	2806.5	0	0.000553
CO_2	0	44.010	86	0	0.000012
Sum			10000000	0	1

Note: Modified from Millero et al. (2008, table 3).

For many seawater measurements the CTD estimate of salinity is accurate and could be used provided the assumption of Reference Composition can be verified, thus saving resources. A sensor capable of verifying the Reference Composition would therefore be very valuable until a sensor is developed which can measure or allow the computation of absolute salinity *in situ*.

An *in situ* absolute salinity measurement would enable the TEOS-10 equation to accurately and immediately provide the full suite of thermodynamic parameters it was developed to address. The TEOS-10 Manual UNESCO (page 141) mentions four potential measurements (here in bold) to the practical implementation of TEOS-10:

> *"A way out of this practical dilemma is the measurement of a different seawater quantity that is traceable to SI standards and possesses the demanded small uncertainty, and from which the salinity can be computed via an empirical relation that is very precisely known (Seitz et al. (2010b)). Among the potential candidates for this purpose are the* ***sound speed****, the* ***refractive***

> ***index***, ***chemical analysis*** *(e.g. by mass spectroscopy) of the sea salt constituents, in particular chlorine, and direct* ***density*** *measurements."*

1.3 Definition of the problem

Absolute salinity is the mass fraction of dissolved material to seawater. The pure water component of seawater is well characterized (IAPWS-09) with respect to molar mass, density, temperature and pressure. However, the dissolved material component of seawater is an unknown quantity since it can have a variable ratio of constituents. The absolute salinity can be determined indirectly by measuring any three of the thermodynamic properties developed in TEOS-10; some examples being temperature, pressure, density, specific heat, compressibility, thermal expansion, and sound speed. This conservative nature of the TEOS-10 equation is a useful step forward with regards to the equations of state for seawater. Oceanographers can choose which measurement parameters will provide the best measurement resolution and accuracy to characterize the seawater of interest.

Two parameters, temperature and pressure, are well established for the purpose. Both parameters have mature sensor technologies, standards, and established calibration facilities. Both parameters also have good resolution and accuracy as shown in table 1.2. However, as will be shown in section 4.1.1, the present temperature sensor technology does limit the vertical profiling speed for *in situ* measurements.

Table 1.2 *Temperature and pressure resolution and accuracy for CTD measurements.*

	Parameter	Resolution	[ppm]	Accuracy	[ppm]
Typical CTD	Temperature	0.0002°C	5	0.005°C	125
	Pressure (strain)	0.08 dbar at 4000dbar	20	4 dbar at 4000dbar	1000
State of the art	Temperature	0.0002°C	5	0.001°C	25
	Pressure (quartz)	4E−5 dbar at 4000dbar	0.01	0.4 dbar at 4000dbar	100

Note: These represent values at time of calibration and were taken from the specifications sheets of three commercial instruments. The typical values are taken from an SBE-19 CTD. The state of the art temperature values are taken from an SBE-911 CTD. The state of the art pressure values are taken from an RBR bottom pressure recorder.

Choosing the third parameter should be done with the accuracy of the generally used EOS-80 methodology in mind. The new measurement parameter should, when fully developed, meet the typical CTD accuracy in Reference Composition seawater and improve upon the density accuracy provided by a typical CTD in non-standard seawater. A 4000 m profiling CTD has the density errors given in table 1.3 at time of calibration. More typically, the errors are closer to those given in table 1.4 which includes 1 year of sensor drift. The total error shown in both tables is the root sum of squares (RSS) combination of the five error sources.

Table 1.3 *CTD density accuracy at time of calibration*.

Error source	Measurement error	Error in density [kg m^{-3}]	Error [ppm]
Conductivity measurement	0.015 mS cm^{-1}	0.011	10
Temperature measurement	0.005°C	0.005	4
Pressure measurement	6 dbar	0.026	26
Salinity calculation, PSS-78	0.02	0.015	15
Density calculation, EOS-80	0.0036 kg m^{-3}	0.0036	4
Total EOS-80 error, RSS		0.033	32

Note: These values are for a SBE-19 CTD at time of calibration. The salinity measurement error is in practical salinity units (psu), which is a dimensionless quantity. The reference conditions are conductivity = 42.914 mS/cm, temperature = 15°C, and pressure = 0 dbar.

Table 1.4 *Typical profiling CTD density accuracy one year after calibration*.

Error source	Measurement error	Error in density [kg m^{-3}]	Error [ppm]
Conductivity measurement	0.041 mS cm^{-1}	0.029	28
Temperature measurement	0.007°C	0.007	6
Pressure measurement	14 dbar	0.062	60
Salinity calculation, PSS-78	0.02	0.015	15
Density calculation, EOS-80	0.0036 kg m^{-3}	0.0036	4
Total EOS-80 error, RSS		0.070	69

Note: These values are for a typical SBE-19 CTD one year after calibration. The salinity measurement error is in practical salinity units (psu), which is a dimensionless quantity.

The choice of a new sensor should be made with the goal of improving upon the 0.033 kg m^{-3} density error of the EOS-80 method shown in table 1.3. The EOS-80 method is mature, and while there is some room for improvement in the conductivity accuracy, there is no ability to improve the salinity calculation error due to the erroneous assumption of constant salt ratios. Therefore, any new density sensing method should have the ability to achieve density errors less than 0.033 kg m^{-3} before being used as an oceanographic tool and should have room to achieve better accuracy as the sensing technology matures. The best sensor choices would be those sensors that have reasonable technological improvement paths resulting in a 5 to 10 ppm density accuracy, which would put the instrument on par with present laboratory density measurements, such as the Anton Parr DSA-5000M.

1.4 Overview of work in this thesis

Since the development of the CTD (Hamon, 1955) and the practical salinity scale (Lewis and Perkin, 1978), the *in situ* oceanographic measurements for salinity and density have remained essentially unchanged. The new TEOS-10 equation provides an opportunity to improve the accuracy of *in situ* measurements of salinity and density by removing the assumption of constant salt ratios in sea water.

The work presented in this thesis considers several possible methods to improve *in situ* measurements for salinity and density. It was recognized at the outset that this is a challenging task for the oceanographic community, and the work here is intentionally exploratory in nature. This thesis was initiated with the expectation that developing a new functional measurement technology for would be a high-risk endeavor. Although the ultimate goal is not achieved here, several potentially promising approaches are developed and examined in detail, and much useful progress is achieved to guide future work.

For this research study, two general approaches to an interim solution of verifying the Reference Composition (Millero et al., 2008) assumption for the practical salinity S_P measurement are considered. The first approach considered here involves measuring the sound speed in addition to the conductivity to provide a verification. However, this approach was ultimately determined to be unsuitable as a conductivity verification for three reasons. First, if the sound speed measurement was accurate enough to verify the S_P measurement with respect to S_A, then sound speed could act as the primary S_A sensor and a conductivity measurement verification would not be required for the S_A measurement. Second, the sound speed sensor calibration reference used by the sensor manufacturer's is either the NRL II equation (Del Grosso, 1974), or Bilaniuk and Wong's (1996) equation, both of which are accurate to 0.02 m s^{-1} in pure water. This error is equivalent to a salinity error of 0.017 g kg^{-1}, which is on par with the salinity error (table 1.4) that the verification is attempting to eliminate. Third, there is no salt water reference with sufficient accuracy for verification. Since no sound speed standard exists, the reference must be a sound speed equation. The top three sound speed equations, TEOS-10 (2010), Chen, Millero and Li (Millero and Li, 1994) and NRL II (Del Grosso, 1974), are based in whole or in part on the same salt water data set (Del Grosso and Mader, 1972, RMS accuracy

0.05 m s^{-1}). This data set, while likely accurate, remains un-validated after 4 decades. Given the requirement for a sound speed accuracy less than 0.02 m/s, the three equations do not track each other well, as shown in figure 1.1. Without a sound speed standard it is difficult to assess which equation is the best reference. As a result, the sound speed approach was initially discounted.

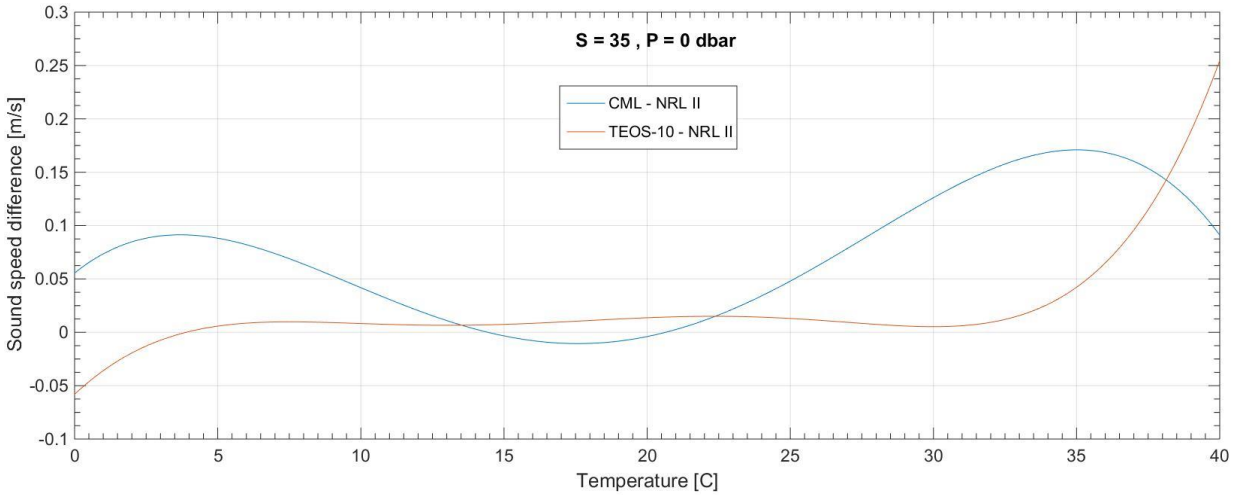


Figure 1.1 *A comparison of the top three sound speed equations. This plot uses the NRL II equation as the reference. The blue plot is the Chen, Millero and Li equation difference from the NRL II equation and the red plot is the TEOS-10 sound speed difference from the NRL II equation. The plots are presented for a practical salinity of 35 and a pressure of 0 dbar.*

Chapter 2 of this thesis describes the second approach considered for the sea water Reference Composition validation. The approach is based on Gurriana et al.'s (2005) paper on the phase shift between the voltage and current measurement for different saline solutions. The phase shift measured for solutions of NaCl and KCl, at the same conductivity, differed by a factor of two. Hypothesizing that the phase shift is due to molecular relaxation (Liebermann, 1949) of certain salts in seawater, then any deviation of the phase measurement compared to the expected phase based on the practical salinity measurement would indicate an error in the assumption of Reference Composition from an ion perspective. The effect on electrical phase due to the addition of non-ionic dissolved matter such as silica, SiO_2, would need to be assessed

empirically. Unfortunately, the phase shift as a bulk property of the saline solution is proven to be incorrect in this thesis, and the phased conductivity approach was abandoned.

Chapter 3 of this thesis describes the first *in situ* measurement method to enable TEOS-10, which is based on density measurement. The method developed here utilizes the acoustic reflection coefficient at an interface between a reference material and the seawater. Prior art for acoustic reflection coefficient laboratory measurements is briefly discussed and theory is developed to improve the accuracy, eliminate the requirement for an accurate sound speed measurement, and allow for profiling the sea water density from the surface to the sea floor. Unfortunately, the test results of the final prototype show the method is overly sensitive to microscopic conditions on the measurement surfaces and hence does not provide a practical sensor. However, incidental sound speed measurements from the prototype provided good estimates of the absolute salinity and density, indicating that sound speed measurement is a feasible method for TEOS-10 absolute salinity determination even without a sound speed standard.

Chapter 4 of this thesis describes the second *in situ* measurement method to enable TEOS-10, which is based on sound speed measurement. A sound speed sensor technology, previously developed by the author (Eaton and Dakin, 1996), was tested for precision and accuracy. The limitations of the sound speed method are described and solutions proposed.

One major limitation to the sound speed method is the lack of a sound speed calibration reference. Although funding limitations prevented the fabrication of a sound speed standard, the design of such a device is presented in Chapter 5.

While a complete, implementable, *in situ* measurement method to enable TEOS-10 is not developed in this thesis, the research showed that accurate sound speed measurement provides a likely basis for such a system. Chapter 6 summarizes the research results, identifies the impediments to the use of sound speed for TEOS-10 and proposes solutions to the significant sources of error.

To recap, each of the methods introduced above is described in a separate chapter:

Chapter 2 **Phased Conductivity**, testing the validity of the constant salt ratio assumption.

Chapter 3 **Density Sensor**, using the acoustic reflection coefficient to determine density.

Chapter 4 **Sound Speed Sensor**, using sound speed to determine absolute salinity.

Chapter 5 **Sound Speed Standard**, specification of a system required to improve the accuracy of the TEOS-10 sound speed coefficients and the calibration accuracy of sound speed sensors.

Chapter 6 **Summary and Discussion**, this chapter summarizes the findings of the various research themes and presents a roadmap for enabling *in situ* measurements to support the TEOS-10 equation.

Chapter 2

Phased conductivity

Eliminating the Reference Composition error would require a new sensor, as discussed in the Introduction (chapter 1). However, if the Reference Composition could be verified for the water sample then the CTD density measurement accuracy could be improved by removing the 0.02 g kg^{-1} uncertainty in the salinity measurement (TEOS-10, 2010).

Gurriana et al. (2005) discuss the discovery of a variable phase shift between the voltage applied and the current developed in saline solutions during conductivity measurements. Gurriana et al. suggest that the phase shift in the sample could be used to differentiate between types of salts.

Gurriana et al. also suggest that the phase shift, electrically modelled as a change in capacitance, could be used to improve the accuracy of seawater conductivity measurements. This could slightly improve the conductivity measurements in table 1.3, but only at very low salinities since the phase shift is proportional to the electrical time constant $\tau = R\,C$, where R is the resistance and C is the capacitance of the conductivity cell. The electrical time constant for the Gurriana cell has been computed from the NaCl plots given in Gurriana et al. (2005). The resulting time constants, plotted in figure 2.1, show the phase shift will vary significantly only for solutions with a practical salinity less than 0.1. Note that the conversion from seawater conductivity to practical salinity, used by Gurriana et al., is an approximation since the NaCl solution is not Reference Composition sea water. The difference is approximately −0.6% of the practical salinity based on table 1.1 values for NaCl.

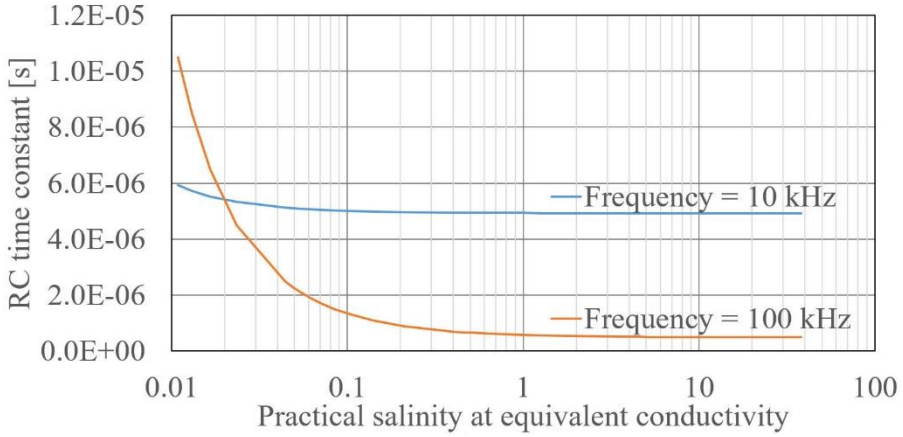


Figure 2.1 *Resistance and capacitance time constant for Gurriana et al. (2005) conductivity cell. Time constant results for an NaCl solution excited at two frequencies. The results are computed from the resistance and capacitance plots of Gurriana et al. (2005). Note that the practical salinity is an approximation since the solution was not of Reference Composition.*

Since there were measurable voltage and current phase shifts discovered by Gurriana et al. there must be a physical mechanism causing these shifts. Stogryn (1971) reports the dielectric properties of saline solutions of various salts found in seawater do not markedly change based on the type of salt. If the electrical permittivity of the saline solution is not the source of the variations in the phase shift, then two other mechanisms seem possible. First and most likely, the phase shift could be due to a boundary layer developing on the electrodes of the conductivity cell. If that is the mechanism, then the phase shift would be susceptible to changes in flow, salt composition and non-ionic dissolved material. The resulting variability in the capacitance of the boundary layer would not provide useful *in situ* improvements to the conductivity accuracy or Reference Composition validation. This could be tested by utilizing an inductive based conductivity sensor instead of an electrode based conductivity sensor. A second, more useful, but less likely hypothesis is that the phase shift is due to molecular relaxation, an important component of sound absorption in the ocean (Mellan et al., 1987). Molecular relaxation occurs when certain salts, $MgSO_4$ and boric acid, associate during the compression phase of an acoustic

wave and dissociate during the rarefaction phase of the wave. This pulls energy out of the compression regime and releases it in the rarefaction regime, 180° later in the acoustic wave. If the electric field of the conductivity sensor caused a similar association and dissociation of the salts this could be the mechanism causing the phase shifts measured by Gurriana et al. and would be measurable over the full range of sea water salinities.

Molecular relaxation due to pressure is known to be frequency limited (Mellan et al., 1987) and the upper cut off frequency is different for each salt. If this limitation applies to electrically induced molecular relaxation as well, the process could be exploited to assess the phase shift contribution of both $MgSO_4$ and boric acid individually.

If the phase shift is caused by molecular relaxation in the presence of an electric field, then the amount of energy absorbed or released during the alternating field cycle should be proportional to the concentration of the participating ions. As a result, for a given conductivity, if the measured phase is not the same as the phase expected for Reference Composition seawater then the Reference Composition assumption is false. While the measurement of correct phase, for a given S_P is not a definitive measurement of all salts being in the proper ratios, an incorrect phase would indicate an error in the S_P measurement.

While there were many ‘ifs’ which could lead to the failure of the phased conductivity approach, the ease with which it could be tested and the potential benefit of the approach warranted the effort to test the hypothesis.

2.1 Phased conductivity- Methods

Testing the hypothesis of molecular relaxation as the driving mechanism behind the phase shift is relatively straightforward. Fresh water and NaCl solutions are not known to exhibit molecular relaxation but $MgSO_4$ solutions, boric acid, and sea water are known to exhibit this effect. Therefore, measuring the phase shift in these solutions should show changes in phase between these solutions at similar conductivities. The use of an inductive based conductivity cell would eliminate the electrodes and therefore eliminate any electrode boundary layer from developing.

The test setup, shown in figure 2.2A, included a temperature controlled bath, solution bath with mixer, a temperature sensor, and a conductivity sensor. The conductivity sensor utilized an inductive conductivity cell, shown in figure 2.2B, with the primary coil excited by an arbitrary function generator and operational amplifier to allow the excitation frequency to be varied. Both the excitation and received signals were captured using a custom built two channel, 16 bit resolution, digitizer operating at 40 MS/s (for clarity throughout this thesis, samples per second, S/s, will be used to indicate digitizing rates and Hertz (Hz), representing cycles per second, will be used to indicate the frequency of the induced or measured parameter). The high digitizing rate was chosen to improve phase resolution. Data were collected at various temperatures and excitation frequencies for each sample liquid.

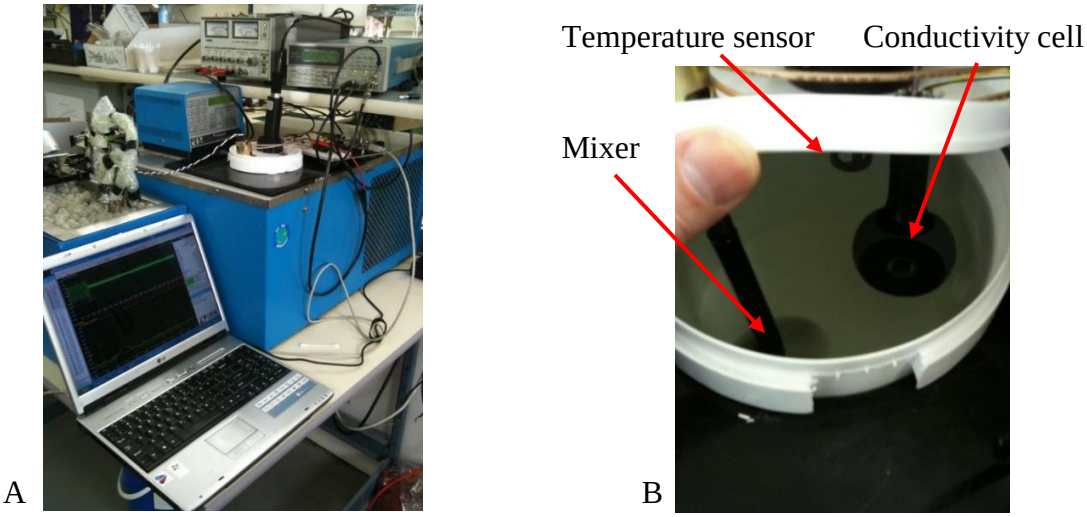


Figure 2.2 *Phased conductivity test setup.*
A) Phased conductivity test setup. B) Conductivity cell, mixer and temperature sensor in sample.

Distilled water, NaCl (table salt), MgSO4 (Epsom salt), and seawater are all easily available so these solutions were chosen for the initial test. To verify that there were no capacitance influences due to ionic layers, the tests were also run using a wire loop through the inductive cell with a series resistance standard (Vishay resistors) as shown in figure 2.3.

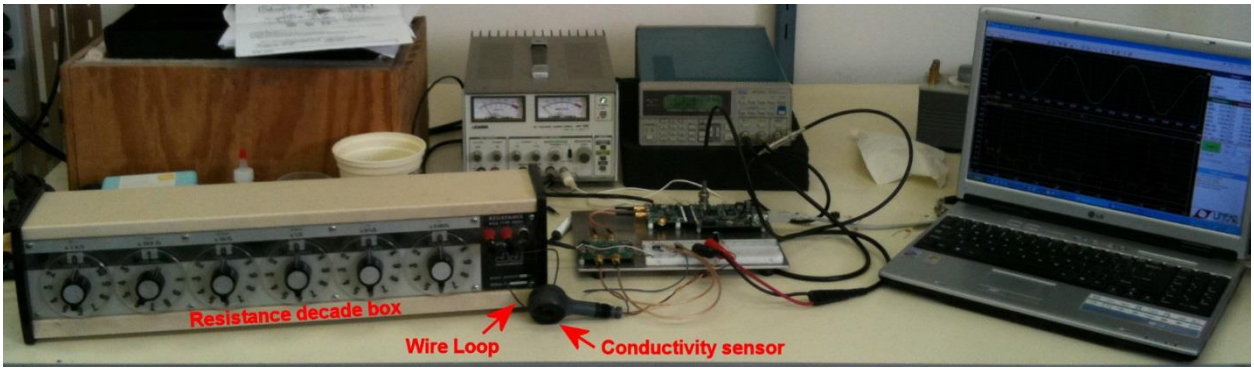


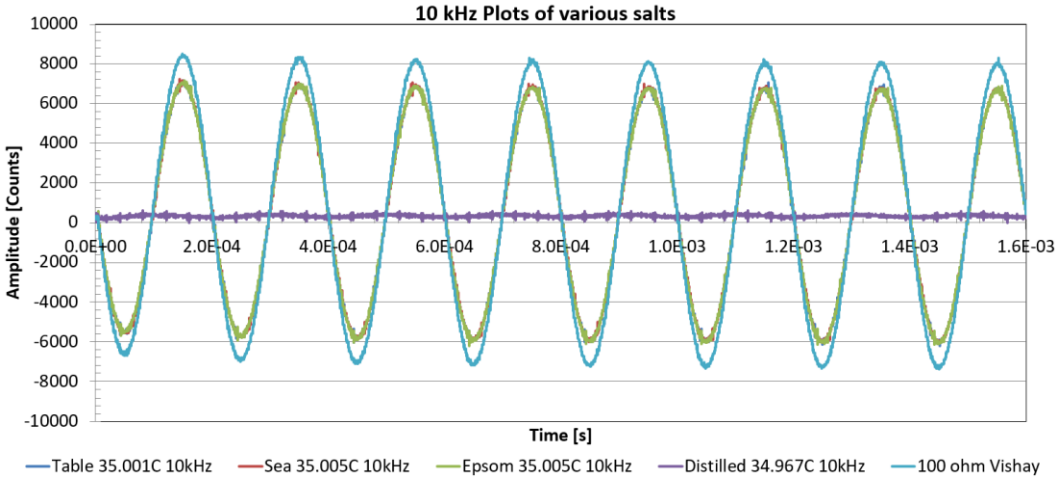
Figure 2.3 *Phased conductivity resistor setup.*

To compare the phase shift between different salt solutions at the same conductivity, the seawater sample was measured first. The table salt and Epsom salt solutions were then prepared by adding salt until the conductivity cell amplitude (proportional to conductivity) matched the seawater amplitude.

2.2 Phased conductivity- Results

Figure 2.4 shows the conductivity cell output at 10 kHz and 200 kHz excitation respectively. Examining the falling zero crossings of each cycle in figure 2.4, one can see there is no phase shift between the saline solutions. Nor is there a phase shift between the solutions and the wired resistance loop. This indicates that there is no measurable phase shift based on the salts in the solution. The only phase shift occurred in distilled water (violet line), which indicates that the phase shift is limited to low conductivities (high resistances).

A



B

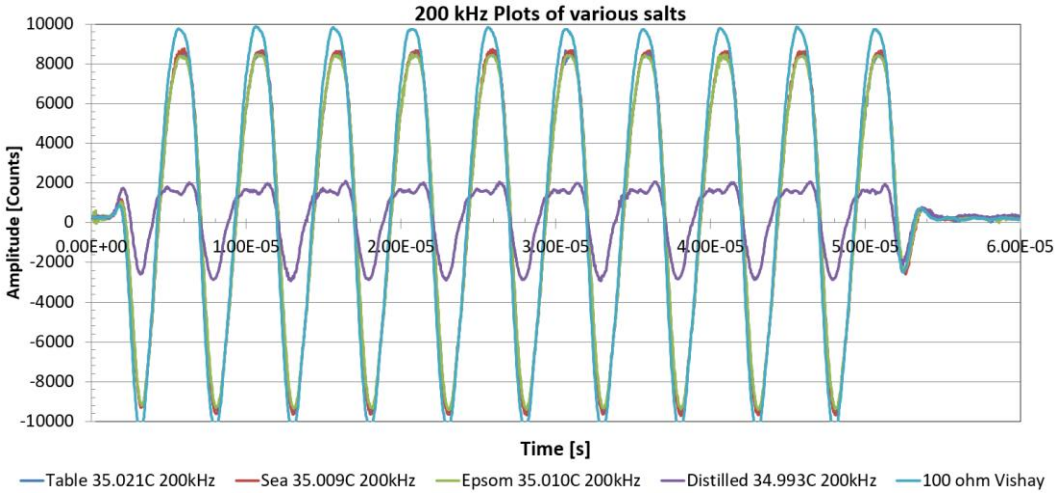


Figure 2.4 *Conductivity cell outputs for various samples.*
Results for: A) 10 kHz excitation, and B) 200 kHz. Blue is table salt, red is sea water, green is Epsom salt, violet is distilled water and cyan is the 100 ohm Vishay resistor loop. The table salt plot is hidden by the seawater and Epsom plots.

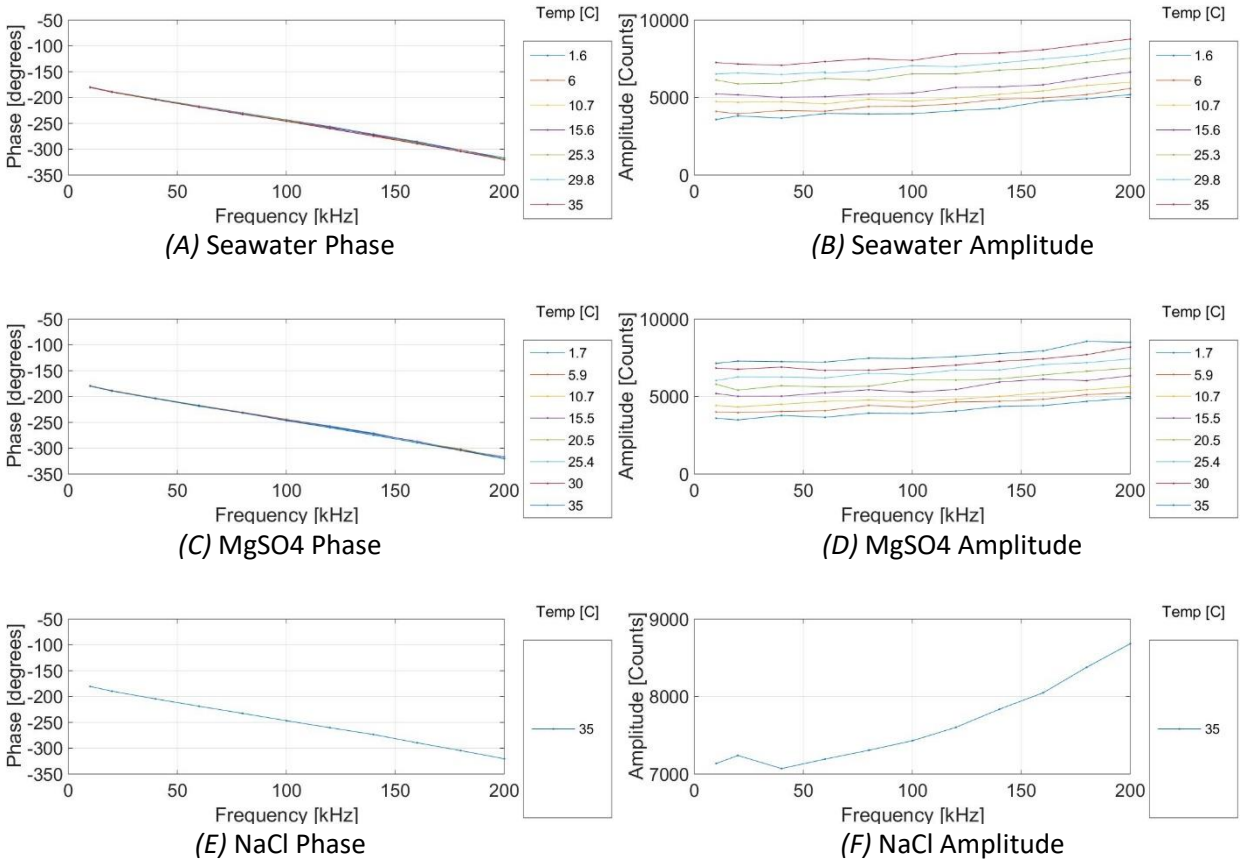


Figure 2.5 *Conductivity phase and amplitude plots for various solutions.*

In an effort to further characterize the phase relationships, the amplitudes and phases of each solution were measured at various temperatures. Changing the temperature varies the conductivity for aqueous solutions. The seawater, $MgSO_4$ and NaCl solution results are shown in figure 2.5. The NaCl solution was only run at 35°C. To simulate similar results with the wire resistance loop, the resistance was varied. Figure 2.6 shows the wire resistance loop results as well as the results for distilled water. The implications of these plots are discussed in section 2.3.

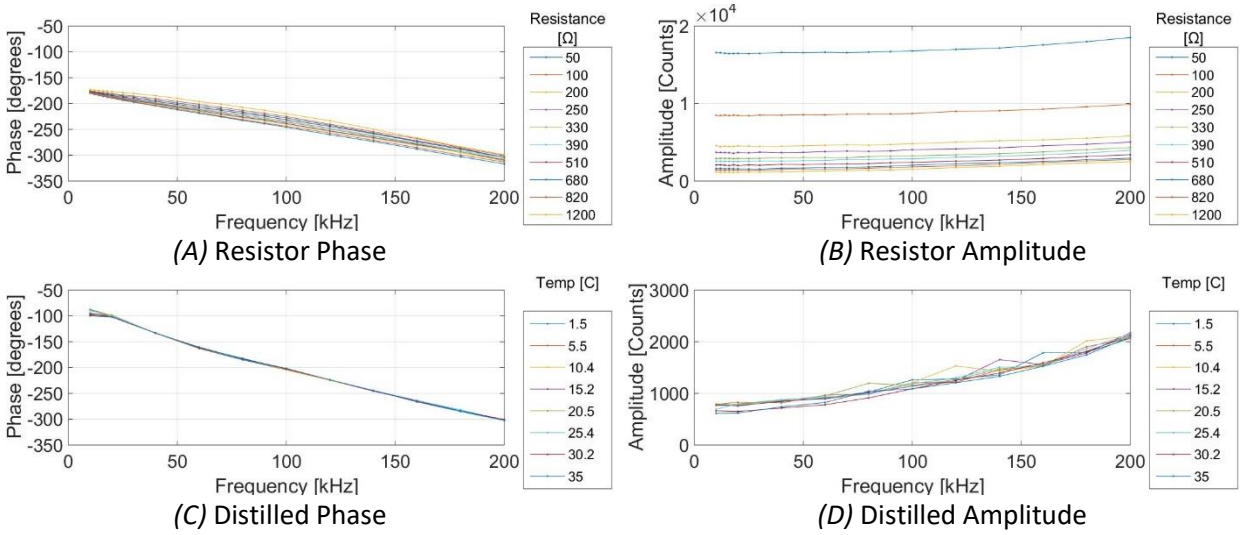


Figure 2.6 *Conductivity phase and amplitude plots for resistance loop and distilled water.*

2.3 Phased conductivity- Discussion

Figure 2.4 shows that at constant conductivity (equal amplitude) there is no phase change as would be expected if the molecular relaxation hypothesis was valid. Figures 2.5 (A), (C) and (E) show almost identical phase plots confirming there is no phase change with or without molecular relaxation capable salts. Figure 2.6 (A) also shows a similar phase plot with an entirely resistive load on the conductivity cell (i.e. no capacitance). This indicates that the phase is dependent only on the load conductivity and excitation frequency. The phase shift is therefore a function of the electrical properties of the conductivity cell and its circuitry rather than a function of the solution constituents.

The failure of the hypothesis means that phased conductivity cannot be used to validate seawater Reference Composition.

Gurriana et al. (2005) did show phase changes for their tests and suggest that the phase can be used to both improve the accuracy of the conductivity measurement and possibly to

distinguish between salts. It is possible that since they used an electrode based conductivity cell that ionic boundary layers were being built up on the electrodes of the cell. This could explain the phase delay as the layer thickness would take time to build as the electric field cycles. This would also create the apparent capacitance that was postulated as part of the electric model of the sensor.

Unfortunately, if the phase shift is a result of boundary layers on the electrodes, then assessing the accuracy of the salt ratio assumption will be difficult. For a given electric field strength, the boundary layer thickness is a function of all the ions present. Since the majority of the ions in seawater are Na and Cl, the phase will be primarily dependent on these ions. The phase sensitivity to other salts will likely be low. This would make the phase measurement unsuitable for Reference Composition validation. The boundary layer will also be susceptible to flow rate and therefore the expected conductivity accuracy improvement may not be realized in ocean profiling applications.

Since the phase shift measured by Gurriana et al. (2005) was proven not to be a bulk property of the solution, further efforts to test the boundary layer hypothesis would not help with the objective of this thesis. The boundary layer hypothesis was therefore not tested.

2.4 Phased conductivity- Conclusion

An inductive conductivity cell was used for the phase measurements to remove any effects of an electrode boundary layer within the cell. The saline dependent phase shift measured by Gurriana et al. (2005) could not be replicated using an inductive cell. This means that the phase

shift is not a result of molecular relaxation. As a result, phased conductivity cannot be used to validate the seawater Reference Composition.

Chapter 3

Density sensor

In an attempt to solve the TEOS-10 sensing problem it was necessary to re-examine the four recommended TEOS-10 potential solutions based on sound speed, mass spectrometry, refractive index, and density measurements. The first solution, sound speed, was initially rejected due to lack of a suitable calibration standard required to meet SI unit traceability, although sound speed calibration is revisited in Chapter 5. The second solution, mass spectrometry, was rejected since the author had previously been involved with the development of the AML Oceanographic Inspectr, an underwater mass spectrometer (Short et al. 1999, 2001). The Inspectr was expensive, labour intensive and the water separation membrane introduced difficulties in isolating the salts from the water for analysis. The third solution, refractometry, is showing progress via recent work in optical refractometry (Wu et al., 2011) but the salinity resolution, 1.5 g kg^{-1}, remains unacceptable for TEOS-10 purposes. The refractive index could be measured acoustically using Snell's law, $\sin(\phi_0)/c_0 = \sin(\phi_1)/c_1$, at the boundary between two media, where c_0 and c_1 are the sound speeds and ϕ_0 and ϕ_1 are the incident and refracted angles. However, this would require a large sensor in order to achieve sufficient angular resolution to make the sensor useful. Acoustic metamaterials may solve the size issue in the future (Haberman and Guild, 2016). However, the refractive index method would also require an accurate sound speed measurement which had already been rejected.

The fourth solution, density, is considered in this chapter by measuring the density acoustically using the acoustic reflection coefficient. This could be accomplished with a small sensor which would allow accurate water column profiling, be easy to handle and deploy, and be

similar in cost to a conductivity sensor. It would seem that this method would also require measurements of sound speed in both media, but a methodology is developed here to eliminate this requirement. An *in situ* density sensor was therefore chosen as the second attempt at solving the TEOS-10 sensing problem.

3.1 Density sensor- Development

The *in situ* density sensor (ISDS) concept is based on measuring the acoustic travel times and the acoustic reflection coefficient between a reference material and seawater. These measurements would then be calibrated to density at various temperatures and pressures. The advantage of this approach is that a high accuracy sound speed reference is not required; the density calibration can be made from high resolution time and acoustic pressure measurements only.

It will become apparent in the following theory that it is necessary to isolate single wave trains in time to obtain uncontaminated acoustic pressure amplitude measurements. It is therefore desirable to reduce the number of waves propagating in the sensor head. One way to do this is to eliminate shear waves by keeping all interface surfaces normal to the acoustic pressure gradient, as shown in figure 3.1. Assuming normal incidence and specular reflection, the solid/water pressure reflection coefficient (Clark, 2011) reduces to

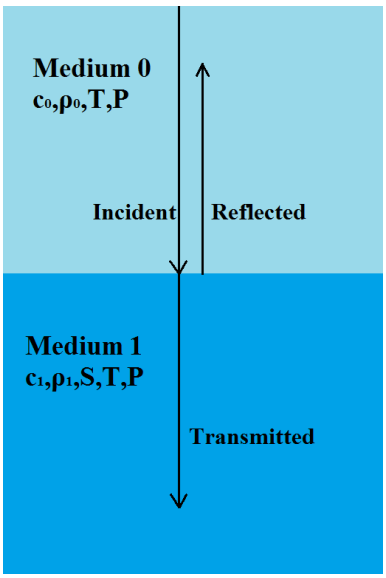


Figure 3.1 *Acoustic reflection at a boundary.*

$$R = \frac{p_r}{p_i} = \frac{z_1 - z_0}{z_1 + z_0}, \tag{3.1}$$

where p_r and p_i are the reflected and incident acoustic pressures, respectively, and z_0 and z_1 are the characteristic acoustic impedances of media 0 and 1, respectively. The characteristic acoustic impedance is equal to the product of the density, ρ, and the sound speed, c, of the medium. Therefore, the density of medium 1 can be solved for as a function of four parameters,

$$\rho_1 = \rho_0 \frac{c_0}{c_1} \frac{(1+R)}{(1-R)}. \tag{3.2}$$

Examining equation (3.2) in a simplistic manner, if the incident medium is a reference material with known density and sound speed, then $R = f(\rho_1, c_1)$. Since sound speed is a function of distance and time, if the acoustic path in medium 1 is held constant, then the sound speed measurement is a function only of the acoustic propagation time. Thus the density, ρ_1, in medium 1 becomes a function of the reflection coefficient and the acoustic propagation time, t, in the water sample,

$$\rho_1 = A \cdot t_1 \cdot \frac{(1+R)}{(1-R)} \quad , \tag{3.3}$$

where A is a sensor specific dimensional constant determined by calibration.

Both quantities, t_1 and R, are measurable and two-dimensional calibrations are often used in ocean sensors (temperature compensated pressure sensors, for example). Therefore, from a simplistic point of view, it would seem that the concept is viable.

3.1.1 Prior art

The patent by Difoggio (2006) used a similar approach for a laboratory based (rather than sea going) density instrument. A cylindrical tube was filled with a liquid sample and the reflection coefficient was measured between the tube and the sample, as shown in figure 3.2(A). There

were issues related to amplitude resolution, diffraction (flat transducer coupled to a round tube), temperature related multipaths (a result of the internal transducer reflections and the coupling material reflections), a very large impedance mismatch between sample and tube, and the wide range of densities to be measured. In spite of these shortcomings, the experimental results look promising, as shown in figure 3.2(B), with a density measurement accuracy of approximately 2% of full scale.

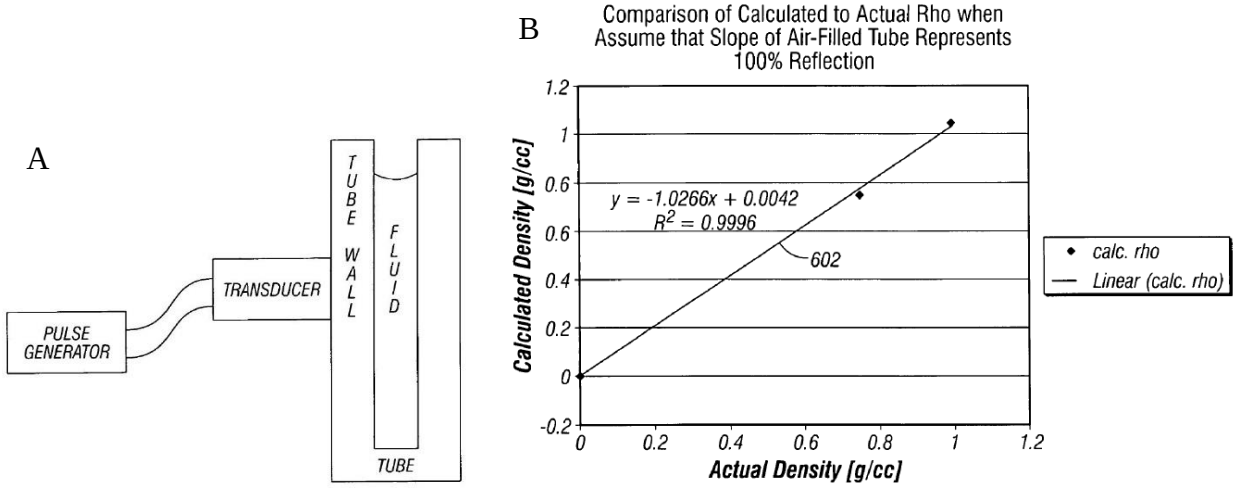


Figure 3.2 *Difoggio (2006) laboratory instrument to measure seawater density via reflection coefficient. Taken from the Difoggio 2006 patent, A) apparatus, B) density accuracy plot. Despite major shortcomings the accuracy looks promising.*

In another work, Bjorndal and Froysa (2008) considered the measurement of the pressure reflection coefficient, not density, however, the concepts are directly applicable. Bjorndal and Froysa explore several different measurement concepts which provide substantial improvements with respect to diffraction, internal reflections and mode conversion as compared to Difoggio's (2006) concept. Bjorndal and Froysa's (2008) apparatus is shown in figure 3.3. This device emits a 5 cycle acoustic wave at 5 MHz from a 2.5 cm diameter transducer face. This wave travels from transducer A through a large aluminum buffer rod, 20.5 cm in diameter and 8 cm thick. A portion of the wave energy is reflected back towards transducer A and a portion enters

the liquid sample, 20.5 cm in diameter and 0.24 or 0.57 cm thick. The energy that enters the liquid sample will reflect back and forth within the liquid sample each time the wave encounters a buffer rod face. At each reflection, a portion of the wave energy is transmitted into the buffer and propagates to transducer A or B as appropriate for the reflecting face. The amplitudes of the signals A_1, A_2, $A_1^*$and A_2^* are used to compute the reflection coefficient. The sequence can be reversed, where transducer B is the transmitter.

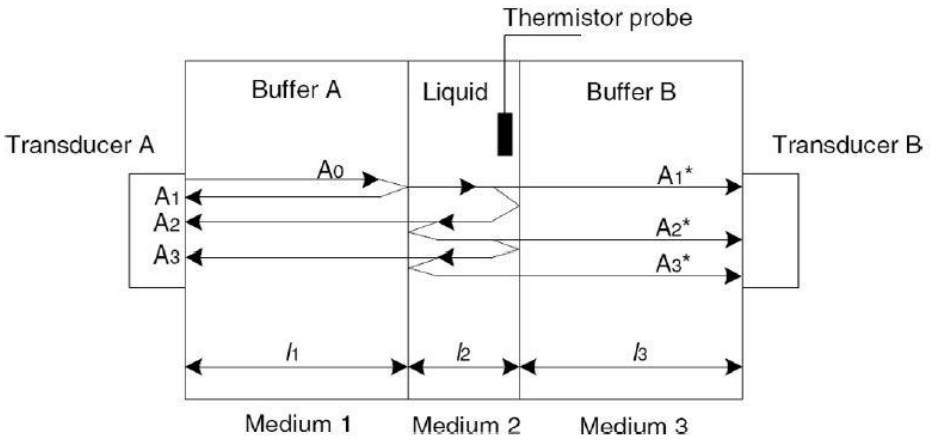


Figure 3.3 *Bjorndal and Froysa (2008) reflection coefficient measurement apparatus.*

The Bjorndal and Froysa (2008) approach measures a pressure reflection coefficient to an accuracy of 300 ppm. However, there are still issues which could be improved, especially for *in situ* measurements, including:

- all boundaries are operating in the acoustic near field, which creates amplitude measurement variations with temperature, pressure and liquid sound speed;
- the device uses a diffraction amplitude correction factor for both diffraction and spreading loss which is independent of temperature, pressure, and liquid sound speed;
- the accuracy degrades linearly with the acoustic attenuation of the sample (i.e. in the presence of suspended solids or bubbles during *in situ* use);

- the very large thermal mass of the sensor will affect the liquid sample making the device unsuitable for ocean profiling;
- the 12 bit (300 ppm) amplitude resolution is a limiting factor, though this could be easily improved with an updated digitizer;
- the impact of internal, longitudinal and shear wave, reflections within the transducers are not accounted for since the internal structure of the commercial transducer is unknown; and
- reducing the density measurement range to the oceanographic density range would improve the resolution.

3.1.2 Measurement resolution.

In practice, with varying pressure and temperature, the density and sound speed of both media in figure 3.1 would vary, as would the critical physical dimensions of the sensor, the electronic and physical propagation delays and the timing clock. Thus the reference (or buffer) material parameters are temperature and pressure dependent, and the sample medium is temperature, pressure and composition dependent. Therefore, the density equation (3.2) becomes a more complicated function in practice,

$$\rho_1 = \rho_0(T, p_h)\frac{c_0(T, p_h)}{c_1(T, p_h, S_A)}\frac{\left(1+R(T, p_h, S_A)\right)}{\left(1-R(T, p_h, S_A)\right)}, \tag{3.4}$$

where T is the temperature, p_h is the hydrostatic pressure and S_A is the absolute salinity.

Batch compositional variability of the reference material and dimensional variability from sensor to sensor precludes a precise knowledge of the functional dependencies of any of the terms in equation (3.4). It may be possible to calibrate the reference material density against

temperature and pressure for each individual sensor with sufficient accuracy, but it would be extremely difficult to calibrate the reference material sound speed on small individual sensor parts with the required accuracy. The target density accuracy, from table 1.2, is 0.033 kg m^{-3}, or 32 ppm. To ensure the instrument accuracy is not resolution limited, most instrumentation manufacturers initially design for a resolution ten times smaller than the target accuracy for the sensor, so the measurement resolution target should be in the 3 ppm range for the density sensor.

For a 0.08 m water path (used in prototype #1 for this research) this resolution requirement equates to a timing resolution requirement of 160 ps. For a 12 mm reference material path the timing requirement is about 8 ps. For cross-correlations the timing resolution is one period of the digitizer rate; which equates to sample rates of 6.3 and 124 GHz, respectively. The latter sample rate is not practical for an *in situ* instrument. However, if the reference material sound speed and thermal expansion varied negligibly with temperature (for example, the glass ceramic ZeroDur has a thermal expansion of only 0.02 ppm C^{-1}), then the time measurement for the reference material would be thermally invariant and could become a calibration constant rather than a critical temperature dependent measurement parameter.

For an acoustic impedance reason, which is explained in section (3.1.3), the reference material of choice is carbon vitreous. To test the thermal variability of carbon vitreous, two test samples were purchased and tested for length and sound speed at 10 and 30°C. The measurements were stable with temperature for both samples within the measurement resolution of the micrometer (1 µm at 25 mm) and oscilloscope (8 m s^{-1} at 4454 m s^{-1} for sound speed). The stable thermal response of carbon vitreous meant it might be a viable material for improving

the time measurement resolution problem. The thermal stability would also be beneficial for sensor thermal response time which would make the sensor accurate when profiling.

For distance measurements of a 0.08 m water path (used in prototype #1) the resolution requirement is 0.12 μm. For a folded path of 12 mm in the solid reference material, the 6 mm thickness resolution requirement is 18 nm. These resolutions are not achievable on a production basis.

Another practical problem is the voltage resolution of the analogue to digital converter (ADC). Digitizers at the time of the instrument design could sample at 2 GHz with 8 bit resolution (3900 ppm). At 160 MHz sample rates it was possible to obtain 16 bit digitizers (15 ppm). The first option is unacceptable and the second option is poor.

Thus measurement resolution is problematic for distance, time and amplitude (voltage) measurements.

3.1.3 Examining the technique for resolution solutions

It may be possible to construct a sensor such that the measurement obstacles are eliminated or reduced.

The basic sound speed equation is

$$c(S_A, T, p_h) = \frac{d(T, p_h)}{t(S_A, T, p_h)}. \tag{3.5}$$

Given the linear hydrostatic compressibility $\alpha_{p_L} = \frac{1}{d}\frac{\partial d}{\partial p_h}$ and the coefficient of thermal expansion (CTE) $\alpha_T = \frac{1}{d}\frac{\partial d}{\partial T}$, the path length in equation (3.5) can be approximated by $d(T, p_h) = d_{nom}\left(1 + \alpha_{p_L}\left(p_h - p_{nom}\right) + \alpha_T\left(T - T_{nom}\right)\right)$, where d_{nom}, p_{nom} and T_{nom} are the path length, pressure and temperature at a nominal condition, such as mid-range for temperature and pressure. Given this first order approximation, the sound speed ratio in equation (3.4) provides an opportunity. If the same material is used for both the reference material and the material that fixes the water path length then the $\left(1 + \alpha_{p_L}\left(p_h - p_{nom}\right) + \alpha_T\left(T - T_{nom}\right)\right)$ pressure and thermal factor is the same for both c_0 and c_1. Since equation (3.4) uses the sound speed as a ratio, the pressure and thermal factors reduce to 1. Thus the sound speed ratio in equation (3.4) becomes $\frac{c_0(T, p_h)}{c_1(S_A, T, p_h)} = \frac{d_{0nom}}{d_{1nom}}\frac{t_1(S_A, T, p_h)}{t_0(T, p_h)}$ and $\frac{d_{0nom}}{d_{1nom}}$ is a sensor specific constant. The $\frac{d_{0nom}}{d_{1nom}}$ constant can be determined by the density calibration which removes the requirement for high resolution sound speed or length measurements. Equation (3.4) then becomes

$$\rho_1 = \rho_0(T, p_h)\frac{d_{0nom}}{d_{1nom}}\frac{t_1(S_A, T, p_h)}{t_0(T, p_h)}\frac{\left(1 + R(S_A, T, p_h)\right)}{\left(1 - R(S_A, T, p_h)\right)} \quad . \tag{3.6}$$

Since the time measurements are a ratio, the temperature effects on the clock will cancel out if the same clock is used for both measurements and if the temperature of the oscillator remains the same during the two measurements. The remaining S_A, T, and p_h, and variabilities in the time measurements are the objectives for the time measurements t_1 and t_0.

Assuming the reference material changes density linearly with pressure, $\alpha_{\rho_{ph}} = \frac{1}{\rho}\frac{\partial \rho}{\partial p_h}$, and temperature, $\alpha_{\rho_T} = \frac{1}{\rho}\frac{\partial \rho}{\partial T}$, the reference density can be approximated by $\rho_0(T, p_h) = \rho_{0nom}\left(1 + \alpha_{\rho_{0ph}}(p_h - p_{nom}) + \alpha_{\rho_{0T}}(T - T_{nom})\right)$; thus, equation (3.6) becomes

$$\rho_1 = \rho_{0nom}\left(1 + \alpha_{\rho_{0ph}}(p_h - p_{nom}) + \alpha_{\rho_{0T}}(T - T_{nom})\right)\frac{d_{0nom}}{d_{1nom}}\frac{t_1(S_A, T, p_h)}{t_0(T, p_h)}\frac{\left(1 + R(S_A, T, p_h)\right)}{\left(1 - R(S_A, T, p_h)\right)}. \qquad (3.7)$$

Since the pressure reflection coefficient $R = p_r / p_i$, there is another ratiometric opportunity to remove variables. If p_r and p_i are measured with the same transducer, the same gain settings, the same analog to digital converter (ADC), and the ADC is linear over the measurement range, then the amplitude scaling factors in both pressure measurements are equal. Within the pressure ratio the scaling factors cancel out and $R = p_r / p_i = V_r / V_i$, where V is the converter voltage count. The remaining S_A, T, and p_h function dependencies for the time and voltage measurements are the desired measurement (the time and voltage dependencies are hereafter omitted for simplicity). Equation (3.7) then becomes

$$\rho_1 = \rho_{0nom}\left(1 + \alpha_{\rho_{0ph}}(p_h - p_{nom}) + \alpha_{\rho_{0T}}(T - T_{nom})\right)\frac{d_{0nom}}{d_{1nom}}\frac{t_1}{t_0}\frac{(V_i + V_r)}{(V_i - V_r)}. \qquad (3.8)$$

Rearranging equation (3.8) and examining the various terms yields

$$\rho_1 = \underbrace{\underbrace{\frac{d_{0nom}}{d_{1nom}}}_{\text{dimension scaling}}\ \underbrace{\rho_{0nom}}_{\text{reference density}}\ \underbrace{\left(1 + \alpha_{\rho_{0ph}}(p_h - p_{nom}) + \alpha_{\rho_{0T}}(T - T_{nom})\right)}_{\text{pressure and temperature compensation}}}_{\text{calibration}}\ \underbrace{\underbrace{\frac{t_1}{t_0}}_{\text{measured times}}\ \underbrace{\frac{(V_i + V_r)}{(V_i - V_r)}}_{\text{measured voltages}}}_{\textit{measurements}}. \qquad (3.9)$$

The reference density temperature compensation $\alpha_{\rho_{0T}}$, in equation (3.9), is material specific. There are materials such as ultra-low expansion glass ceramics that could remove the necessity to account for this term in the calibration. However, the high acoustic impedance mismatch to water would increase the reflection coefficient but reduce the reflection coefficient change due to

water density changes, and may therefore preclude the use of such materials. If this is the case, then the sensor must be temperature compensated. Temperature compensation for a material with a CTE of 3 ppm/°C would require temperature measurement with an accuracy of 0.5°C to achieve 3 ppm accuracy in the density measurement. This accuracy is easily achievable with present off-the-shelf sensors.

The reference density pressure compensation $\alpha_{\rho_{0\,p_h}}$ is material specific. Even materials with a relatively high bulk modulus such as ceramics have $\alpha_{\rho_{0\,p_h}} \approx 1$ ppm bar^{-1}. Therefore, at pressures of 1000 bar, pressure compensation will be required regardless of the reference material chosen. At $\alpha_{\rho_{0\,p_h}} = 10$ ppm bar^{-1}, the pressure would have to be measured to 1 dbar accuracy which is achievable with standard oceanographic strain gauge sensors to depths of 6000 m.

The dimension scaling and reference density constants combine to form a single sensor specific calibration constant. Equation (3.9) then becomes

$$\rho_1 = \underbrace{A\underbrace{\left(1+\alpha_{\rho_{0\,p_h}}\left(p_h - p_{nom}\right)+\alpha_{\rho_{0T}}\left(T-T_{nom}\right)\right)}_{\text{pressure and temperature compensation}}}_{\text{calibration}} \underbrace{\frac{t_1}{t_0} \quad \underbrace{\frac{\left(V_i+V_r\right)}{\left(V_i-V_r\right)}}_{\text{measured voltages}}}_{\text{measured times}} \quad \text{measurements} \quad . \tag{3.10}$$

From equation (3.10), the limiting factors to the density resolution and accuracy are the time and voltage measurement resolution and accuracy. Both of these measurements are limited by the analog to digital converter. There is an inverse relationship between the voltage and time resolution in the converter. If a single converter is used to digitize the time series for both the time and voltage measurements this becomes a limiting factor with this sensor technique.

The incident signal voltage, V_i, is the largest voltage to be measured so fixing the voltage range close to V_i will provide the maximum V_i resolution for a given converter.

The reflected signal voltage, V_r, is a smaller voltage so its resolution will be proportionally lower. It may be possible to improve the V_r resolution by measuring V_r^2, V_r^4, etc. This can be accomplished by looking at subsequent internal reflections within the reference material. The steeper the slope of the cumulative pressure reduction curves, shown in figure 3.4, the better the V_r measurement resolution. The figure shows the best performance, i.e., the steepest cumulative pressure reduction slope, is achieved using a material that provides a high reflection coefficient and using many reflections. In practice, scattering and shear waves (the origin of the shear waves is described in the diffraction section 3.1.6) within the reference material limit the number of internal reflections to three or four at best. Therefore, a significant slope increase for the cumulative pressure reduction is only possible for reflection coefficients above 0.7, as shown for the green curve in figure 3.4.

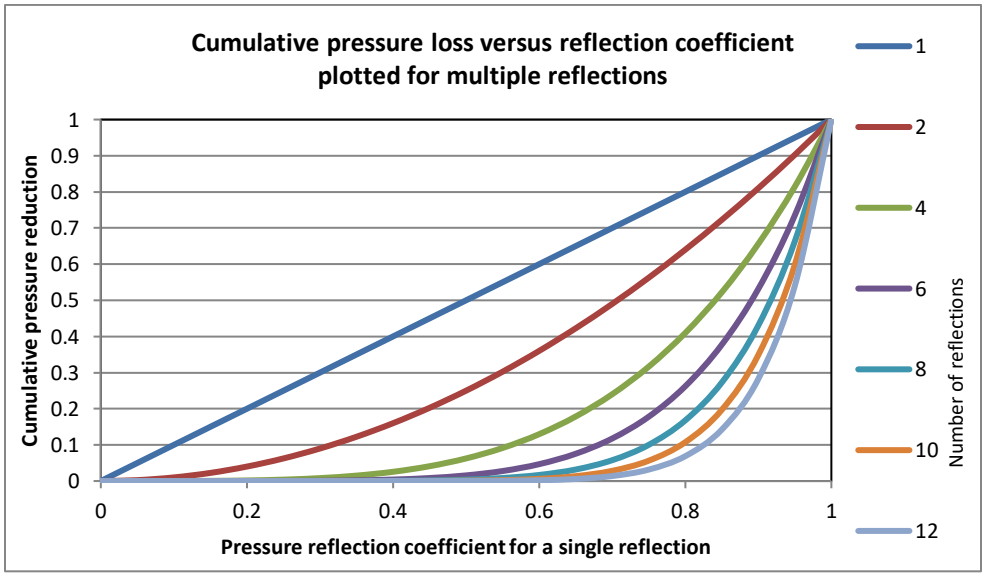


Figure 3.4 *Pressure versus reflection coefficient for various numbers of reflections. The slope steepness determines resolution.*

Unfortunately, at high reflection coefficients a relatively large change in the water density results in a very small change in the reflection coefficient. As a result, the hoped for improvement is not fully realized. This can be shown by comparing the acoustic pressure change from distilled water to sea water for reflections from reference materials with different reflection coefficients, for example, titanium with a reflection coefficient of approximately 0.9 in water and carbon vitreous with a reflection coefficient of approximately 0.65 in water. Titanium provides a change in acoustic pressure of 0.75% with a single reflection, and 2.9% with four reflections. Carbon vitreous provides a change in acoustic pressure of 3.3% for a single reflection, and 12.5% with four reflections. As a result, the amplitude resolution turns out to be better with a lower impedance reference material and the maximum practical number of internal reflections.

The time measurements can also take advantage of multiple internal reflections within each material. This will artificially double, triple, etc. the path length in each material and thus improve the time resolution.

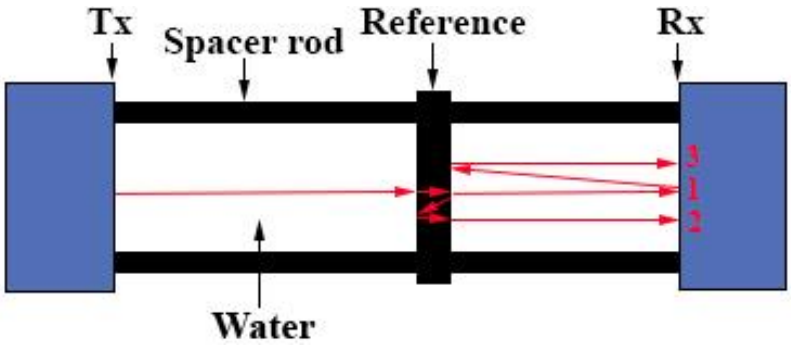


Figure 3.5 *An acoustic density sensor concept.*
Signal 1 is the direct path. Signal 2 is the reference material internal reflection path. Signal 3 is the water chamber internal reflection path. This figure depicts the thin buffer layer sensor concept which uses a very thin layer to separate the piezoelectric element from the water. The buffer layer and piezoelectric element are too thin to show on this figure.

One possible acoustic sensor head is conceptually depicted in figure 3.5. To assess the ability of the sensor to provide the resolutions required, the acoustic travel times should be

examined. If the water chamber between the reference and the receiver is 4 cm long (as in prototype #1), then the two-way travel path transit time, t_1, is the difference between the times for signal 1 and signal 3. At a sound speed of 1500 m/s the difference is 53.33 µs. To achieve 3 ppm resolution in the measurement of this time would require a timing resolution of 160 ps. The prototype #1 reference material is a 6 mm thick plate of carbon vitreous. The two-way travel time in the carbon vitreous reference material, t_0, is the time difference between signal 1 and signal 2. With a sound speed in the carbon vitreous of 4454 m/s the time difference is 2.7 µs. The timing resolution required for this signal would be 8 ps. The requirement could be reduced by increasing the number of internal reflections within the reference material. Thus the timing requirement could be improved to 16 or 24 ps for two or three internal reflections. At the time the prototype sensor was constructed ADCs operating at 160 MS/s with 16 bit voltage resolution were available. This digitizer provides a timing resolution of 6.25 ns which is inadequate. The timing resolution would have to be increased by a factor of 260. Digitizers operating at 2 GS/s were available at the time of the assessment; however, the voltage resolution was only 8 bits.

The voltage resolution should also be better than 3 ppm. To allow for signal variability in various ocean conditions, the peak signal should be tuned to 80% of the peak digitizer amplitude. Therefore, the digitizer requires a range of (1/3ppm/0.8•2) = 833 kcounts, which requires a 20 bit converter. At 16 bits, the digitizer used for the prototype is therefore inadequate, providing a peak signal voltage resolution of $1/2^{(16-1)}/0.8 = 38$ ppm.

Each year ADCs are sampling at faster rates and occasionally the bit count improves as well, so at some point in the future ADCs may be adequate for this approach. In the interim there are some techniques which could be applied to improve the performance.

One option to improve the timing resolution is parabolic interpolation of the correlation function. This has been used successfully in earlier work by the author on the commercial MicroSV sound speed sensor to achieve a factor of 130 improvement in timing. A factor of 260 improvement is untested. At a 130 fold interpolation the timing resolution would be 48 ps. This is not far from the 8, 16 or 24 ps requirement and should be adequate for a feasibility study. For two internal reflections this would provide 9 ppm timing resolution.

A second option to improve the timing resolution is to mathematically upsample the waveform using fast Fourier transform (FFT) techniques (zero padding in the frequency domain and inverse transforming). This approach was tested on sound speed sensor signals and produced nearly identical results to the parabolic interpolation but at the cost of significantly more computation time. For battery operated instruments this should be avoided.

A combined (RSS combination of 9 ppm for timing and 38 ppm for amplitude) resolution of 39 ppm would give a density resolution of roughly 0.04 kg/m^3. Assuming an accuracy of 5 to 10 times the resolution, the density accuracy would be about 0.4 to 0.8 $kg\ m^{-3}$. This is not impressive when compared to the CTD based density resolution of 0.001 $kg\ m^{-3}$ and accuracy of 0.033 $kg\ m^{-3}$. Fortunately, the density measurement resolution is limited by the ADC specifications which continue to improve each year. In the meantime, the feasibility of the method can be tested to better than 1 $kg\ m^{-3}$.

There are many other factors which can affect the *in situ* density sensor accuracy including acoustic attenuation in the water, dissolved gases, diffraction, spreading loss, waveform shape, scattering, shear waves, timing accuracy, bubbles, turbidity, and biofouling. Each of these issues are discussed in the following sections.

3.1.4 Acoustic attenuation

Short range acoustic attenuation in water can vary from negligible in pure water to nearly total absorption in water laden with suspended solids or bubbles. The measurement concept described above utilizes multiple reflections within the reference material for the critical pressure amplitude measurements. This technique allows the same water path length for each of the amplitude measurements with concurrent transit times. Thus, the water attenuation effects are applied equally to the two signals and the signal amplitude ratio will remain accurate. The main impact of water attenuation will be on the sensor's resolution. The density measurement resolution will degrade by an amount proportional to the degradation in signal strength; for example, a 50% loss of signal due to attenuation will equate to a 50% reduction in resolution. If the signal strength degrades significantly, signal to noise ratio considerations will impact the density measurement accuracy as well. Utilizing the Francois-Garrison (1982) model for attenuation, at a frequency of 1 MHz and an acoustic path of 0.2 m, the attenuation varies less than 1%. For tests performed by the author using a Smart SV (1 MHz and 0.2 m path), the sound speed sensor amplitude dropped no more than 5% between distilled water and various sediment laden water samples from the Salish sea. A 5% signal amplitude degradation has a negligible effect on accuracy due to resolution loss.

3.1.5 Dissolved gasses

The effect of dissolved air on the density of water is discussed by Harvey et al. (2005). The water density change between degassed and air saturated water varies from -0.002 kg m^{-3} at 30°C to -0.005 kg m^{-3} just above the freezing point. Since this variability is so small, the water density uncertainty due to dissolved gasses was not studied in the work on the ISDS.

3.1.6 Diffraction

To minimize diffraction effects, the sensor dimensions for prototype #1 were chosen so that the components were in the acoustic far field. Utilizing the Fraunhofer distance equation (Balanis, 2005), a transducer active diameter a of 7 mm, a signal frequency f of 2.5 MHz, and a water sound speed c_w of 1500 m s^{-1}, the near field distance is $D_{nf} = a^2 f / (4 c_w) = 20$ mm. The reference material for prototype #1 was positioned 5 cm away from the transmit transducer and the receive transducer is 9.6 cm away from the transmit transducer. Since all components are in the far field it was assumed this would minimize the diffraction errors.

3.1.7 Spreading loss

For fixed geometries (fixed transmit and receive transducer diameters at a fixed range) spreading loss normally varies with sound speed since the beam angle of the transducer is sound speed dependent. The ratiometric approach of the sensor design reduces this variability. The energy loss due to beam angle variations is applied to both signals in the ratio nearly equally, assuming that the receive transducer face is operating well inside the main lobe diameter. The spreading loss then becomes a fixed geometric problem with the additional range caused by the internal reflections within the reference material, as shown in figure 3.6. The 7 mm diameter receive transducer captures acoustic energy over a face area of 38.485 mm^2, and this area is located 96 mm away from the transmitter transducer in the prototype #1 design. If the range is extended to account for an internal reflection within the reference plate, the acoustic energy spreads to an area of 43.521 mm^2. For prototype #1, the spreading loss is a constant and equal to 88.4% for a single internal reflection and 78.9% for a dual internal reflection within the reference

material. Since the losses are fixed constants they can be treated as part of the dimensional calibration coefficient *A* in equation (3.10).

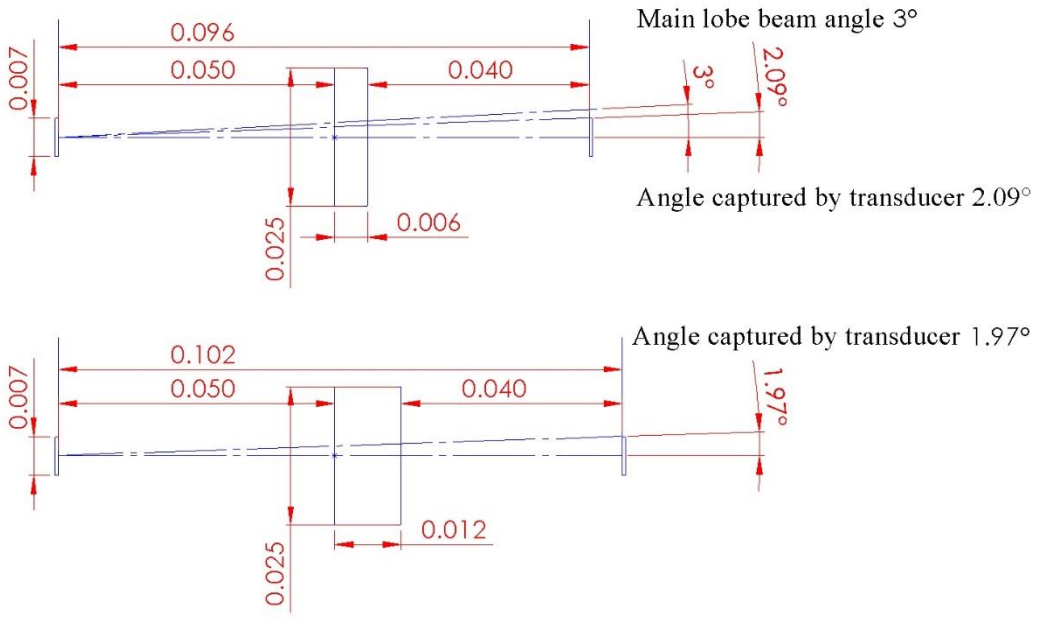


Figure 3.6 *Geometry for spreading loss calculations.*
Top shows direct path. Bottom shows path with one internal reflection in reference material. All distances are given in metres.

3.1.8 Transmitted waveform

The acoustic reflection coefficient method requires measurement of the signal amplitude under various conditions. For linearity in the amplitude measurements across all ambient conditions the transducer transmit and receive response should be constant. This is not the case for present day piezoelectric materials operating in oceanic environments. The piezo material specifications vary with both temperature and pressure. These variable specifications include the piezoelectric charge constants, the voltage constants, the frequency constant, the density, the capacitance, and the electromechanical coupling factors. The variations will affect the output pressure amplitude, thickness and radial resonance frequencies, and electrical impedance of the transducer (which will affect the drive voltage and transmit phase delay). These changes will

produce small variations in the shape of the transmitted acoustic wave. The ratiometric approach to the sensor design will negate these effects since the two signals in the ratio are derived from the same transmit pulse.

Large changes in the transmitted waveform shape are governed by the internal acoustic reflections from the various layers used in the transducer construction. These layers will also affect the receiver transducer and alter the received waveform. The layers affect the ringing of the transducer (bandwidth, burst duration) and this in turn affects the minimum separation time between reflected signals within the reference material. Given that the reference material is limited by the manufacturing process to 6 mm thick with a sound speed of 4454 m s^{-1}, the second signal will follow the first signal with a delay of 2.7 μs. Thus the acoustic signal must decay to less than 0.0002% of the peak amplitude in under 2.7 μs for the sensor to achieve accurate amplitude readings. The result is that the material choices in the design of the transducer will affect the ability to determine the acoustic reflection coefficient accurately.

In an effort to save money and time testing various prototype configurations, an acoustic sensor model, SensorModel, was developed by the author in MatLab to assess the material and dimensional choices. SensorModel is a one-dimensional acoustic propagation model allowing up to 26 material layers. Figure 3.7 shows the SensorModel output for the prototype design and figure 3.8 shows an expanded view of the receiver acoustic pressure as calculated by the model. The signals decay in sufficient time to obtain accurate amplitude measurements. The red plot is the acoustic pressure at the transmit transducer and the blue plot is the acoustic pressure at the receive transducer.

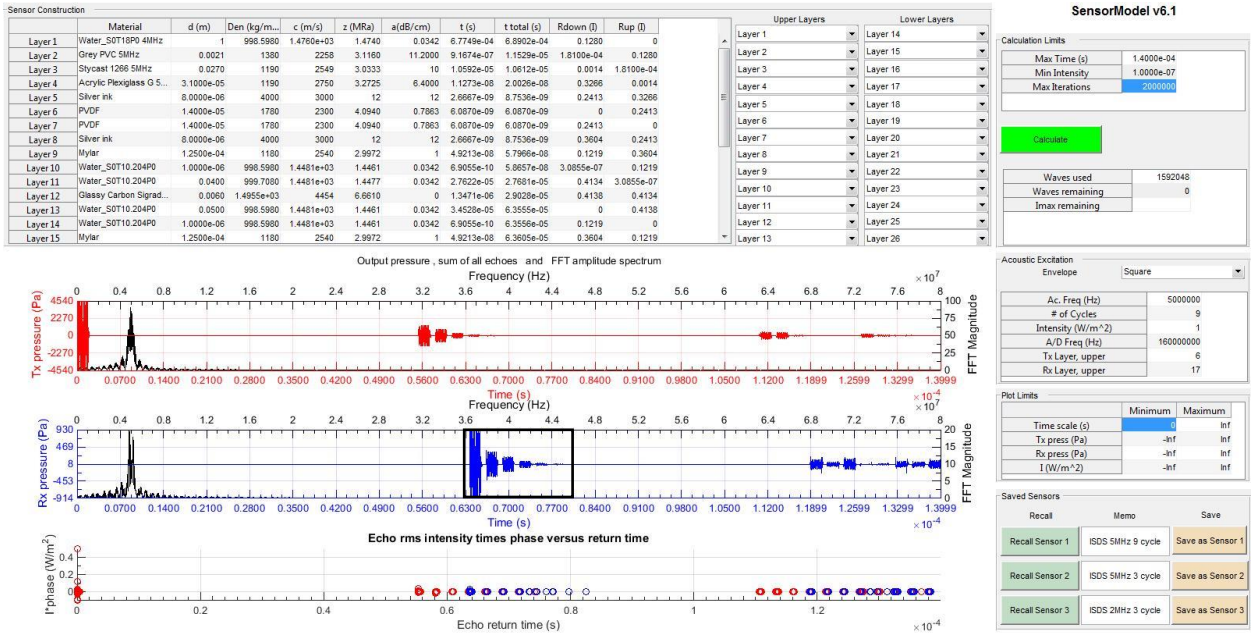


Figure 3.7 *SensorModel display of prototype design.*
The boxed region of the received pressure signal (blue plot) is shown at an expanded scale in figure 3.8.

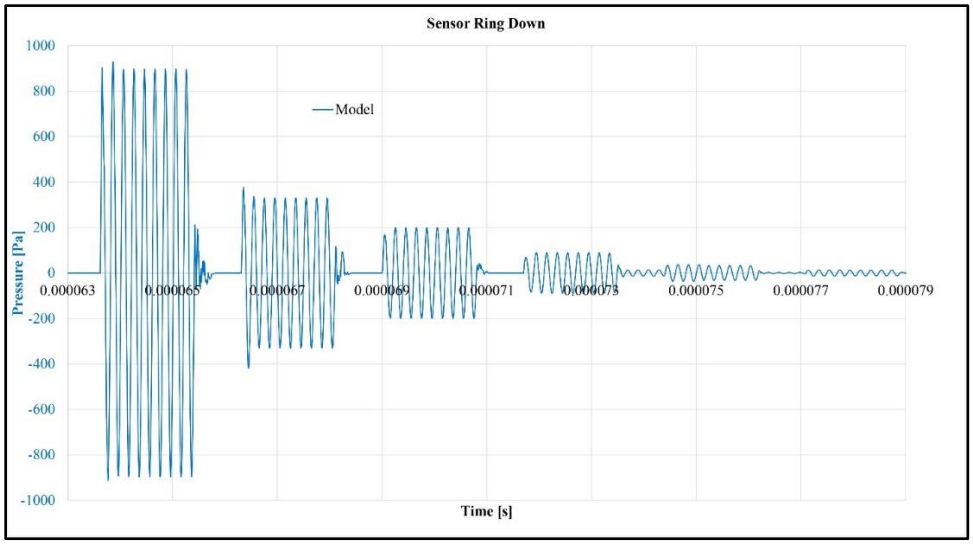


Figure 3.8 *SensorModel receiver acoustic pressure signal from figure 3.7 at expanded scale.*

The transducer layers, such as the electrodes and isolation materials, will vary in thickness with temperature and pressure. The internal reflections will also vary as the water sample acoustic impedance changes. These changes affect both signals in the ratiometric approach of the measurement and therefore cancel out.

3.1.9 Scattering from a rough surface

For specular reflection, the reflector surface roughness should be much smaller than the acoustic wavelength, $r << \lambda$. At 5 MHz in the carbon vitreous resonator, the wavelength is $\lambda_c = c_c/f$ = 4454/5E6 = 891 µm. The wavelength in seawater is 280 to 320 µm depending on the sound speed. These dimensions double for a 2.5 MHz signal.

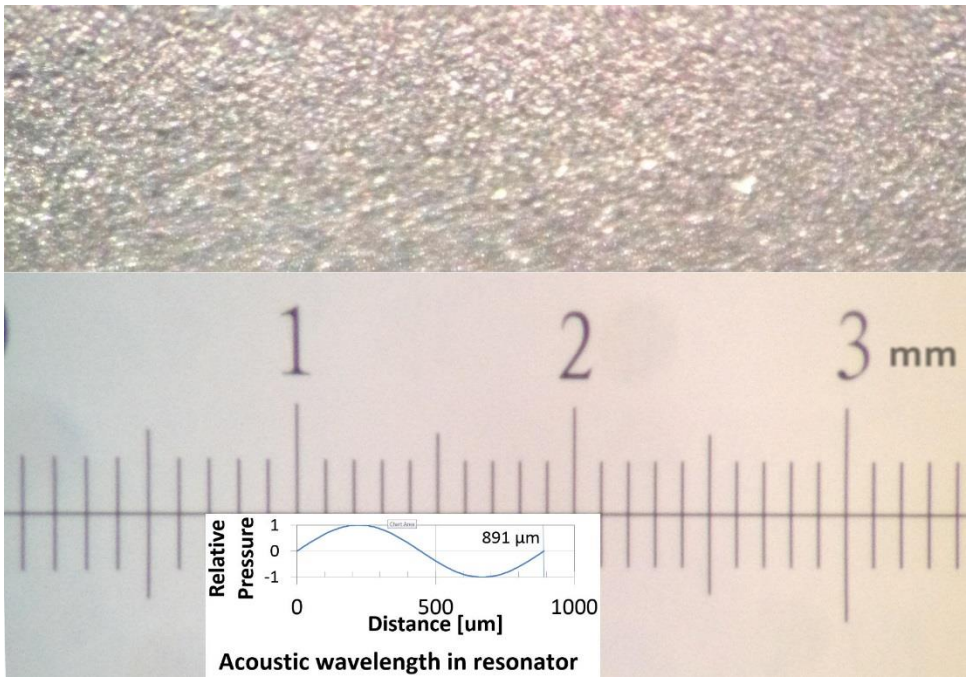


Figure 3.9 *Microscope image of reference surface.*
Overlay shows an acoustic wavelength in the reference at 5 MHz.

Figure 3.9 shows a microscopic view of the surface of the reference disk. The average surface roughness of the resonator surface is estimated as 30 µm. This roughness is about 1/30th of a wavelength at 5 MHz in the carbon vitreous and 1/10th of a wavelength in the water. Therefore, scattering will be small but not negligible. The prototype sensor has been designed to mitigate effects of the scattering of the water borne acoustic waves, since the signals used in the pressure ratio measurement travel through the same water path. However, the scattering of the acoustic waves within the carbon vitreous reference resonator remains an issue. For production

units the surface should be polished to a surface finish (Ra specification) of 3 µm or better. This is an easily achievable finish.

3.1.10 Shear waves

To improve the accuracy of the reflection coefficient measurement it is desirable to limit the variables associated with the measurement. Each variable contributing to the signal amplitude will add an uncertainty to the density measurement. Since the piezoelectric transducer will detect longitudinal pressure (acoustic) waves, shear waves and thermal fluctuations, the measurement sensor should be configured to maximize the pressure wave detection and minimize the other two sensitivities. To minimize the shear wave influence, it is advisable to eliminate their generation rather than try to filter them out. This is accomplished by keeping the acoustic wave propagation perpendicular to each material interface. To accomplish this, the reference material plate must extend beyond the main lobe of the transmitted beam. This reduces the reflection and mode conversion of acoustic energy from the sides of the reference material plate. The main lobe beam width is dependent on the dimensions of the transducer with the largest beam width governed by the narrowest dimension of the active area of the transducer. The active area of the transducer was designed to have a diameter of a = 7 mm. The half beam width angle (beam divergence from the centre line to the 6 dB down point) scaled for degrees and SI units is given by $\theta = \sin^{-1}[1.2c/a/f)]$ which yields a half beam width angle of 3°. The requirements for the transducer face diameter, a, were calculated at the receiver where the maximum spreading occurs. At a range of r = 9.6 cm the face diameter required would be $a = 2r \tan(\theta)$ = 9.9 mm. The minimum face diameter for the prototype design was conservatively chosen as 25 mm.

3.1.11 Timing accuracy

Given that the same digitizer is used for capturing the signals shown in figure 3.5, and the signals are captured within the same 100 µs window, then the timing accuracy is dependent upon the signal correlations and interpolations. Three factors affect the correlations: electrical noise, acoustic noise, and diffraction. Assuming parabolic interpolation, as used in sound speed sensors, the interpolation accuracy is determined by the symmetry of the correlation function at the peak correlation.

Estimating the timing accuracy in advance is difficult given the unknown levels of the three factors involved. Thus, the timing accuracy will have to be estimated based on the empirical measurements.

3.1.12 Bubbles

Bubbles in the water will cause a change in the average water density, sound speed, scattering and absorption of the acoustic energy. Assuming the bubbles are homogenously distributed, including along the reference/water interface, there are several factors to consider with respect to the impact on the density measurement. These factors include the size of the bubbles with respect to the acoustic wavelength, the effect of the volume scattering, the effect on sound speed, and the effect of the bubbles at the reference/water interfaces.

If the bubbles are less than 0.1 of the acoustic wavelength (Caruthers, 2011), few in number and randomly distributed, the scattered acoustic energy will be uncorrelated and small with respect to the insonified cross sectional area. A 30 µm diameter bubble represents 16 ppm in area with respect to the 7 mm diameter transducer face. This will have a small impact on the

waveform amplitude and shape; however, the sensor is designed to ratiometrically cancel out water path amplitude changes. Therefore, the sensor will function properly with respect to the measurement of density for Rayleigh scattering.

If the bubbles are greater than 0.1 of a wavelength, then the bubbles will act as Mie scatterers (Caruthers, 2011) and cause diffraction errors in the density measurement. As the bubble size increases above one wavelength the diffraction errors will become more severe and render the density measurement invalid. Except near the surface, air bubbles in sea water are very small and few since bubbles of larger size rise quickly to the surface (Urick, 1983). At the surface the bubble entrainment depth is roughly equivalent to the significant wave height (Wu, 1994). The open nature of the sensor design will mitigate the entrapment of bubbles in the water sample volume during profiling. However, bubbles could be trapped on the transducer and reference plate surfaces upon immersion or while in the ocean's surface layer. This also happens with other sensors, such as conductivity sensors, and manufacturers recommend pre-wetting, shaking and acclimatization time in the water prior to measurement.

The impact of resonant bubbles (0.65 μm radius at 5 MHz, 1.3 μm at 2.5 MHz, according to Urick, 1983) in the water and especially on the reference plate faces will need to be studied with an operational *in situ* density sensor.

The sound speed change in the water due to the presence of bubbles will affect the density measurement in two ways. The volume effect on sound speed will affect the t_1 measurement of equation (3.10). Bubbles on the reference material face will affect the density of the water sample at the interface. If the bubble and water densities at the reference plate faces do not match

the bulk bubble and water densities, then errors in the density measurement will occur. These errors will need to be studied with an operational *in situ* density sensor.

3.1.13 Turbidity

Turbid water reduces the volume of seawater within a conductivity cell. This appears as a reduction of the conductivity and therefore a reduction of the estimated practical salinity and density of the seawater. The suspended solids typically have a higher density than the seawater, and therefore turbid water has a higher density than seawater alone. This compounds the density error based on CTD density measurements. In contrast to the CTD density measurement, the *in situ* density sensor should treat the turbid waters the same as bubble filled waters for scattering. If the particulates are smaller than 0.1 of a wavelength (Caruthers, 2011) then the ratiometric amplitude measurement in equation (3.10) will be accurate, provided the turbidity scattering does not attenuate the acoustic wave to the point that the signal to noise ratio becomes a problem.

In low flow regimes the larger suspended solids will settle out of the water column. In high flow regimes the effect of suspended solids on the density measurement will need to be studied with an operational *in situ* density sensor.

3.1.14 Biofouling

The ISDS sensor relies on the water/reference interface to make the density measurement. Any biofouling on this interface will therefore affect the measurement accuracy. Macro fouling such as barnacles will render the sensor inoperable. Micro fouling, as a result of biofilms which are invisible to the naked eye, will also affect the measurement.

Biofouling on the transducer faces is less critical. Macro fouling will cause problems due to interference with the acoustic waves and inciting shear waves in the transducers. Biofilms on the transducer faces will not affect the reflection coefficient measurements due to the ratiometric operation of the sensor. However, if a biofilm is allowed to form on the transducer faces macro fouling will develop quickly.

Therefore, biofouling must not be allowed to form on the reference plate or transducer faces. For a profiling application this is not a concern since the sensor will not be in the water long enough to support a biofilm colony. However, for static *in situ* applications biofouling mitigation measures must be taken. Wipers can be used; however, as the wipers foul they will scratch the soft transducer faces and damage the sensor. A better approach would be an ultra-violet (UV-C) antifouling system, although the effect of UV-C on the transducer face and carbon vitreous needs to be considered.

3.1.15 Sensor design and methodology

Based on the assessments above, the sensor concept seemed practical and the prototype design was initiated. The objective for this research was a proof of concept (laboratory prototype) that would be capable of being re-engineered into a device that could be deployed at sea and thus enable the use of TEOS-10.

The sensor head was modelled using SensorModel to test various material and thickness choices to ensure that there were no internal reflection problems within the timing windows required to make the measurements. Two prototype sensor heads were constructed. The details of the two prototypes are described in the section 3.1.15.1. Built into the sensors were two

circuit boards, designed by the author, one for the transmit transducer and one for the receive transducer. These boards contained transmit and receive switches and low noise signal preamplifiers.

For the prototypes, the transmit signal was generated by an arbitrary waveform generator. This allowed the transmit signal frequency, number of cycles and amplitude to be changed easily for testing. The transmit waveform was sent simultaneously to the transmit transducer and the sensor electronics. The sensor electronics used the signal as a synchronization pulse for the data capture. For the *in situ* design this would likely be done via direct digital synthesis from a microprocessor followed by an amplifier.

The custom designed prototype sensor electronics consisted of a variable gain low noise amplifier, a 20 MHz anti-aliasing filter (eventually removed to reduce noise), a 16 bit analog to digital converter operating at 160 MS/s, a USB interface, high and low voltage power supplies, a transmit signal pulser (eventually removed to reduce noise) and an aluminum Faraday cage.

MatLab scripts were used to control the sensor electronics so the data captures could be automated. This allowed between 1 and 4000 samples to be captured as required. Multiple samples were used to allow averaging in an effort to reduce system electrical noise. The same MatLab script was used to perform the signal time correlations, interpolations, RMS amplitude estimations, FFT amplitude estimations and computation of the water sample density. In the *in situ* design this would likely be done in a single field programmable gate array chip.

Calibration was done by measuring the prototype's response to distilled water. Pressure effects were not addressed in this study.

3.1.15.1 Sensor head construction

The first sensor concept utilized a long buffer rod as shown in figure 3.10. If the reference material is carbon vitreous and the transducer diameter is small, the buffer layer cannot be manufactured wide enough to prevent off axis multipath signals from interfering with the desired signals. The manufacturer stated the maximum rod diameter is 10 mm. Additionally, shear waves would develop within the buffer rod and further interfere with the measurements. If the transducer is the same diameter as the buffer, the buffer could act as a planar wave guide.

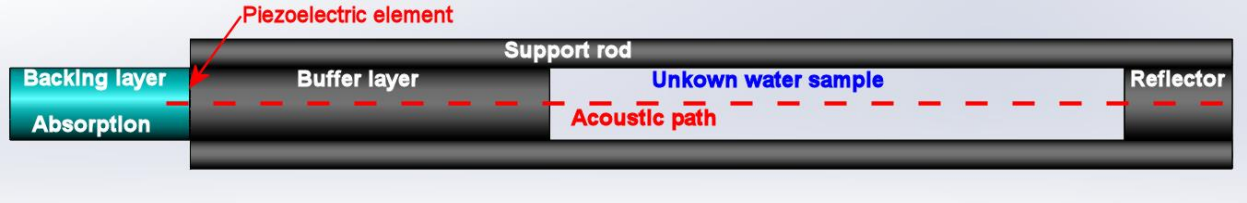


Figure 3.10 *Initial sensor design concept utilizing a long buffer.*

To test the buffer layer planar wave guide approach, Polyvinylidene fluoride (commonly referred to as PVDF) transducers were attached to a 188 mm long 12.4 mm diameter test rod, as shown in figure 3.11, and the signal quality was analyzed. The planar wave guide test was performed using a stainless steel rod and an acrylic rod. In both cases the three cycle transmitted signals suffered from interference and became 23 cycle long signals at the far end of the rod as shown in figure 3.12.

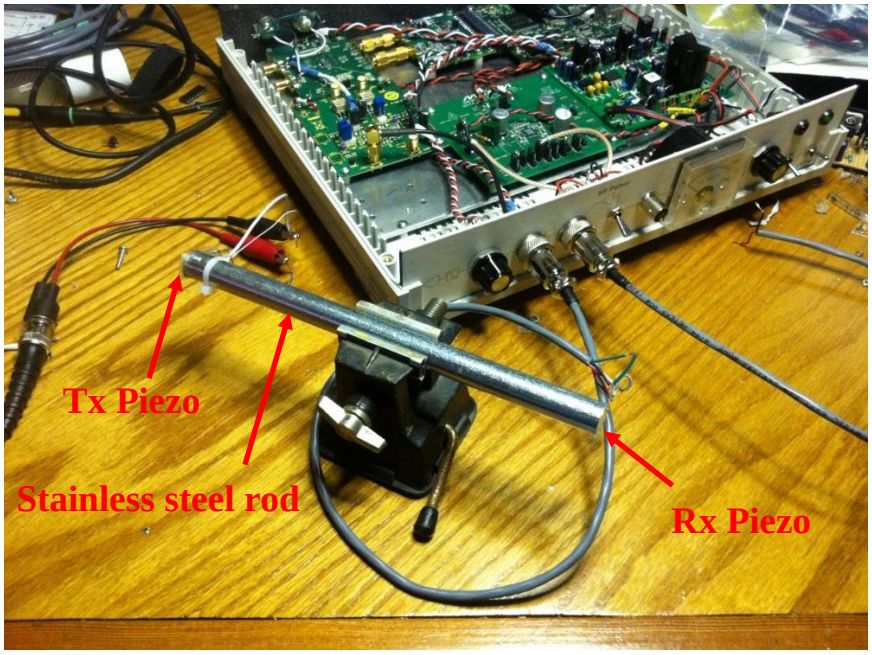


Figure 3.11 *Testing signals propagating in a stainless steel rod.*

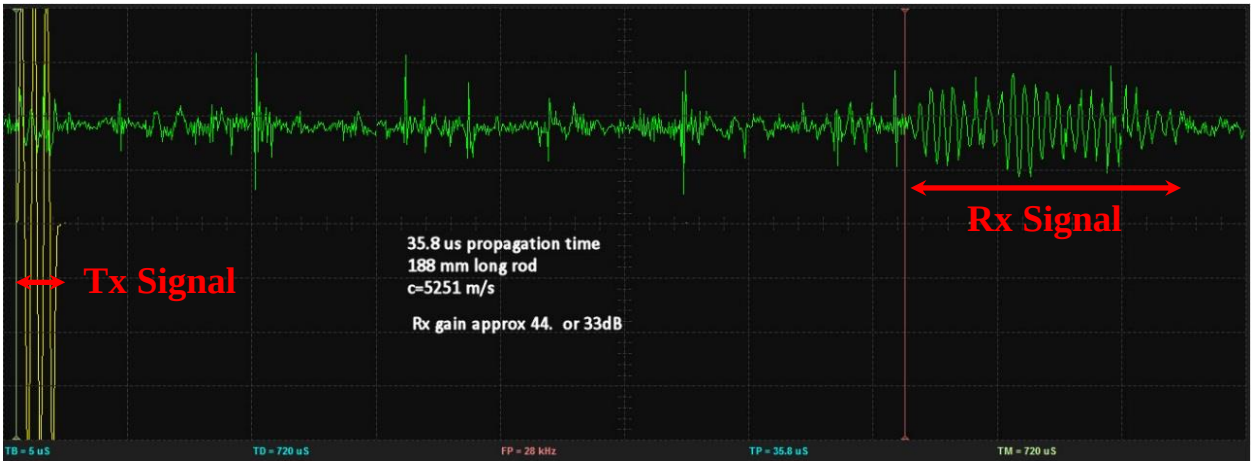


Figure 3.12 *Received signal after transmission through a long rod.*

As a result, the sensor head was redesigned with a thin buffer layer. The concept is shown in figure 3.5. Figure 3.13 shows the signals modelled with SensorModel for this configuration. Both prototype sensors performed similarly to the model as shown in figure 3.14. The signals in figure 3.13 are numbered in the same way as figure 3.5. Signal 1 is the direct path, signal 2 is the path with one internal reflection within the reference disk, and signal 3 is the path with one

internal reflection within the water. The amplitude decay from signal 1 to signal 2 is used to estimate the power reflection coefficient. Since there is a reflection from both faces of the disk the amplitude is proportional to the pressure reflection coefficient squared which is proportional to the power reflection coefficient.

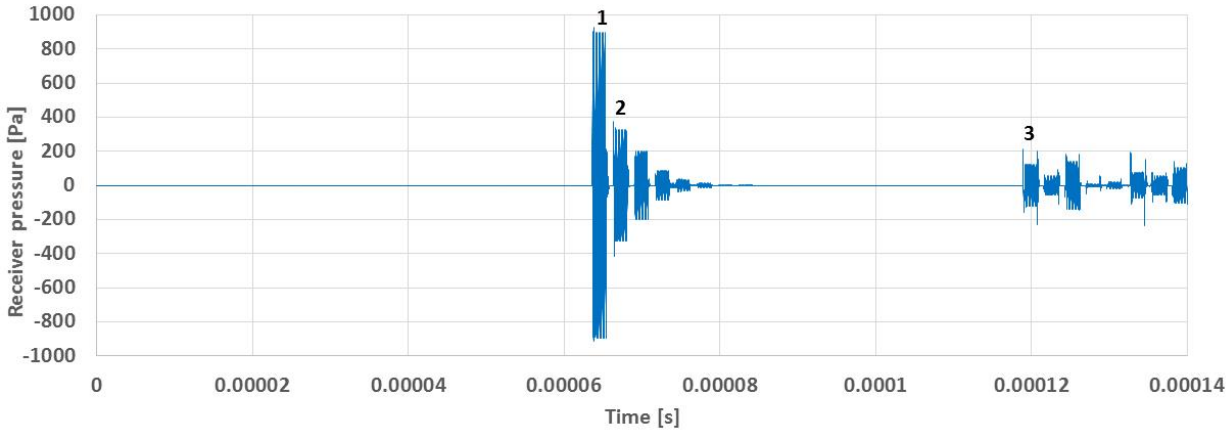


Figure 3.13 *Model waveform for receiver transducer. Numbered signals match the paths shown in figure 3.5.*

Through the use of SensorModel it became apparent that the choices of the piezoelectric material and thickness as well as the electrode material were important considerations in the transducer design. SensorModel also proved helpful later in analyzing the results from the prototype testing. The final sensor model construction parameters chosen are listed in table 3.1.

The objectives in the model transducer design were two-fold. First, to maximize the useable signal duration given the limitation of the 6 mm thick reference disk which required a fast ring down time for the transducer and, second, to choose sensor dimensions to ensure there was no interference with the critical amplitude measurement signals.

Table 3.1 *Sensor as modelled prior to prototype design.*

Material	Thickness [m]	Density [kg m^{-3}]	Sound speed [m s^{-1}]	Acoustic impedance [kg m^{-2} s^{-1}]	Acoustic absorption [dB cm^{-1}]
Water_S0T18P0	1	998.6	1476	1.474	0.034
Grey PVC	0.00207	1380	2258	3.116	11.2
Stycast 1266	0.027	1190	2549	3.03	10
Acrylic Plexiglass G	3.1e-05	1190	2750	3.273	6.4
Silver ink	8.0e-06	4000	3000	12	12
PVDF	2.8e-05	1780	2300	4.094	0.786
Silver ink	8.0e-06	4000	3000	12	12
Mylar	0.000125	1180	2540	2.997	1
Water_S0T18P0	0.040	998.6	1476	1.474	0.034
Carbon vitreous	0.006	1495.5	4454	6.66	0
Water_S0T18P0	0.050	998.6	1476	1.474	0.034
Mylar	0.000125	1180	2540	2.997	1
Silver ink	8.0e-06	4000	3000	12	12
PVDF	2.8e-05	1780	2300	4.094	0.786
Silver ink	8.0e-06	4000	3000	12	12
Acrylic Plexiglass G	3.1e-05	1190	2750	3.273	6.4
Stycast 1266	0.010	1190	2549	3.033	10
Grey PVC	0.00207	1380	2258	3.116	11.2
Water_S0T18P0	1	998.6	1476	1.474	0.034

With respect to the first objective, 6 mm was the thickest carbon vitreous disk the manufacturer could provide, given the nature of the manufacturing process, so this dimension set the acoustic burst duration limit. To provide temporal separation between the signals 1 and 2 from figure 3.5; the ring down time for the transducers was designed to be very short. The ring down time is the signal duration after the transmit signal length, in this case, one cycle. The one cycle ring down time predicted by the model is very close to the actual response of the first prototype. The modelled receiver response is shown compared to the actual signal from the sensor in figure 3.14.

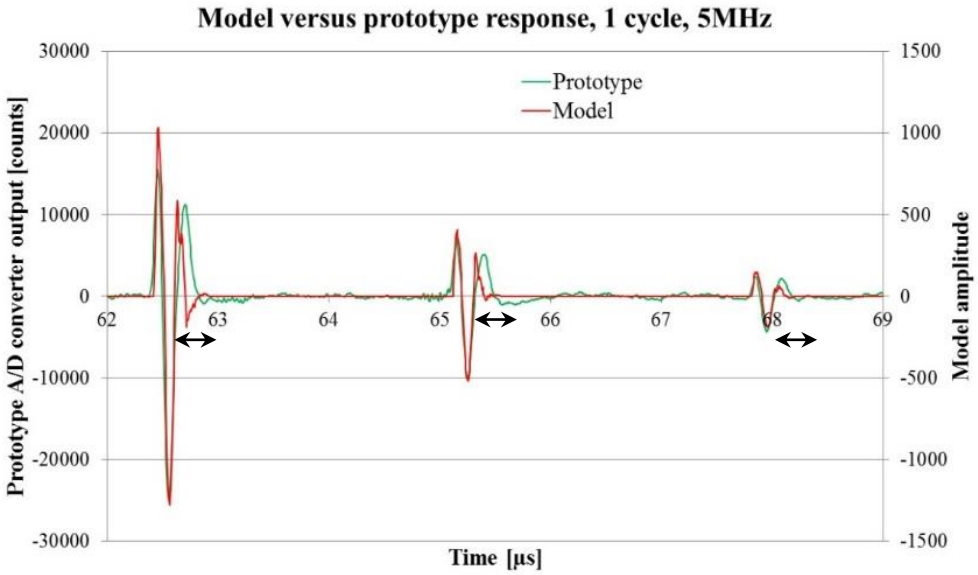


Figure 3.14 *Comparison of signals from SensorModel (red) and actual sensor (green). Ring down times for the signals are comparable from the model to the actual sensor. Ring down time is the signal duration after the transmit signal length, in this case, 1 cycle.*

With respect to the second objective, figure 3.15 shows that the prototype #1 design dimensions chosen provide clean spans for the critical signals.

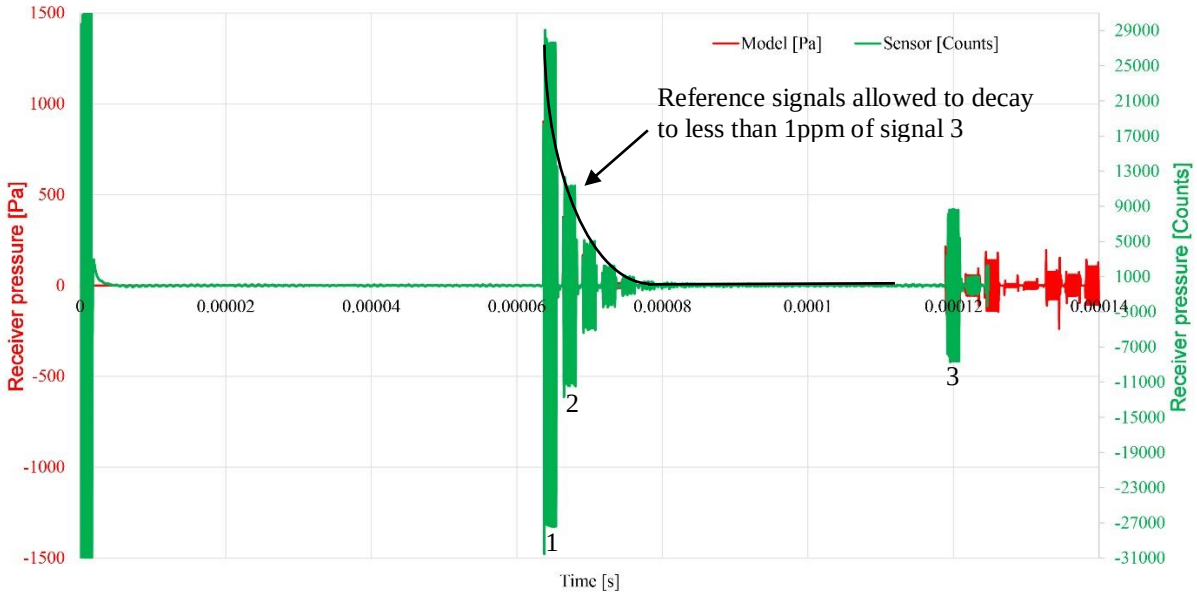


Figure 3.15 *Pulse trains of model and prototype sensor showing signal decay.*

The numerically modelled sensor was mechanically designed using SolidWorks (Dassault Systemes, 2012) prior to fabrication of the component parts, as shown in figure 3.16. For fiscal reasons, one significant compromise was made during the final stages of the design. The 7 mm diameter piezo elements were replaced with the closest off-the-shelf component available, which was a 10 mm by 15 mm rectangular piezo, of the same thickness and mechanical construction as the piezo of the model. The implications of this change are examined in the discussion on diffraction in section 3.2.4.

Figure 3.16 *SolidWorks assembly of the prototype#1 ISDS sensor head.*

During construction of the sensor head it was critical, for shear wave mitigation, to ensure that the transducers and reference disk were parallel so that the acoustic propagation was exactly normal to the faces. Spacer blocks, for use during the sensor head construction, were designed and precisely machined to ensure accurate positioning of the faces, as shown in figure 3.17A.

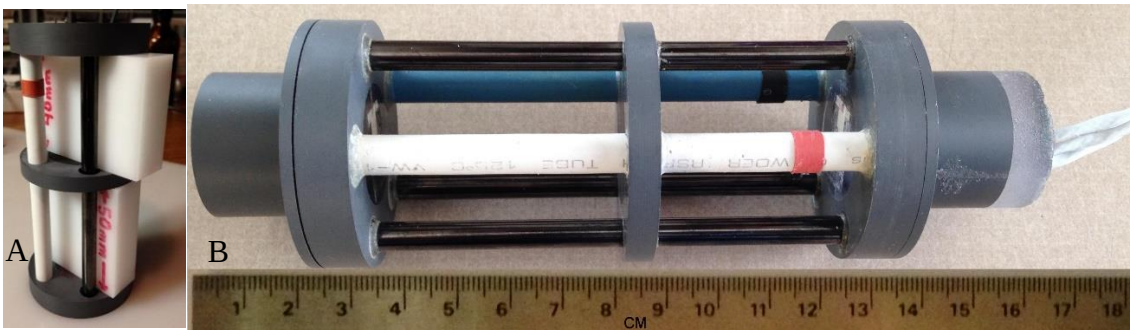


Figure 3.17 *ISDS prototype #1.*
A) Spacer blocks to ensure parallel surfaces and precise dimensions during construction. B) Completed prototype #1 ISDS sensor head.

Testing of prototype #1 revealed problems with the measurement of the signal amplitudes. The signal amplitudes did not track the expected signal amplitudes based on equation (3.10). Diffraction was found to be a significant source of error for signal amplitude; the detailed analysis of the prototype #1 diffraction issue is presented in section 3.2.1. This analysis necessitated a redesign to lengthen the water paths in the construction of prototype #2. Due to the expense of the carbon vitreous, the spacer rods were fabricated using 316 stainless steel for prototype #2, shown in figure 3.18. This was done with the expectation the rods would affect the thermal compensation of prototype #2. The second prototype would determine if the unexpected signal amplitudes of prototype #1 were attributable to diffraction. If diffraction proved to be the problem, the sensor could be rebuilt as a longer version of prototype #1. The final test results in section 3.2 are based on prototype #2 testing.

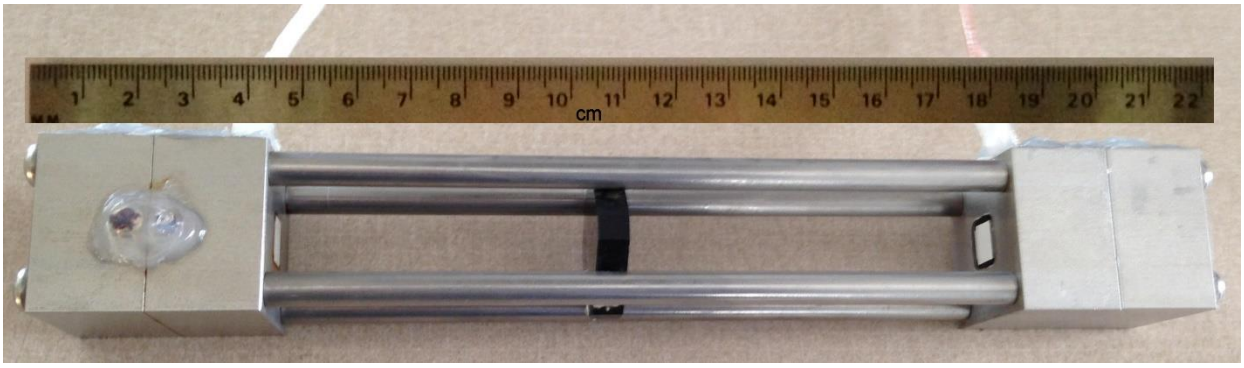


Figure 3.18 *ISDS prototype #2.*

3.1.15.2 Electronics construction

The electronics for the sensor were designed and constructed to the block diagram shown in figure 3.19. The resulting electronics box is shown in figures 3.20 and 3.21.

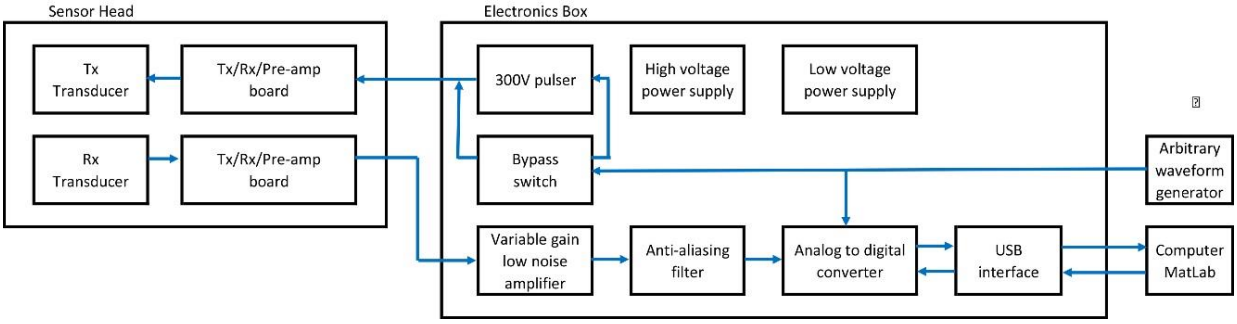


Figure 3.19 *Electronics block diagram.*

Figure 3.20 *Electronics box outside view.*

Figure 3.21 *Electronics box inside view.*

3.2 Density sensor – Results

3.2.1 Diffraction

Initial testing of prototype #1 of the ISDS sensor showed density measurement errors in excess of 100 kg m^{-3}. As a result, the effects of design changes on diffraction were re-examined.

The original prototype #1 sensor design called for a transducer radiating diameter of 7 mm which produced a Fraunhofer near field distance (Balanis, 2005) of 20 mm at 2.5 MHz. A suitable transducer was not available so a commercial off the shelf transducer was chosen. In an effort to maintain the water tight integrity of the transducer, it was decided that the transducer should not be trimmed to size. Since the original design was operating well into the far field, no consideration to the dimension change with respect to the near field was given at the time. A further setback resulted from manufacturing limitations for the carbon vitreous reference material which limited the plate thickness to 6 mm. This necessitated an acoustic frequency change to 5 MHz to allow sufficient time for the acoustic pulse train to come to equilibrium and then decay prior to the arrival of the next internally reflected acoustic wave. The transducer bandwidth with respect to the frequency change was considered, but the diffraction effect was not considered.

For the prototype #1 transducer, the radiating area is 10 mm by 15 mm producing a diagonal of 18 mm. With a wavelength of 298 µm at a sound speed of 1490 m s^{-1} this gives a near field distance of 273 mm. The receiver, at a distance of 96 mm, was therefore well within the near field. The constructive and destructive interference that cause diffraction are dependent on distance, frequency and sound speed. A specific sea water density can occur with a range of sound speeds, dependent on the salinity, temperature and pressure of the water. As a result,

sound speed changes in the water will cause deviations in the acoustic intensity at the receiver that are not a function of the water density.

To assess the effects of diffraction on the propagating wave a diffraction MatLab model was coded, based on Huygens-Fresnel principle (Fresnel, 1816) principle, to simulate the waveform as it approaches any given distance from the transducer. The model, called NearField, is a combined three-dimensional and two-dimensional model. The transmitter is modelled as a planar surface divided into small areas, with each surface radiating in three dimensions. The receiver is only modelled as a line array, to reduce computation time, which is divided into small segments. Each receiver segment sums the energy received at its location from all of the transmitter segments. This is used to calculate the acoustic pressures on a plane normal to the two transducers and parallel to the length axis of the transducers. The model can be run for any plane in the transducer width axis. The plots shown are based on the plane in the centre of the transducers.

Figure 3.22 shows two plots of the modelled normalized pressure amplitude for a cross section through the centre of the sensor coincident with the long axis of the radiating face. The wave is depicted as it is about to hit the receiver transducer of prototype #1. The pressure amplitude is normalized to the pressure emitted at the transmitter transducer face. The so-called edge wave in the field of non-destructive testing (Mair et al., 1987) can be seen in the lower half of figure 3.22A. The interference pattern is called an edge wave because it appears to radiate from the edges of the transducer. The diffraction induced pressure variability along the major crests and troughs of the wave train is seen in the three-dimensional representation in figure 3.22B.

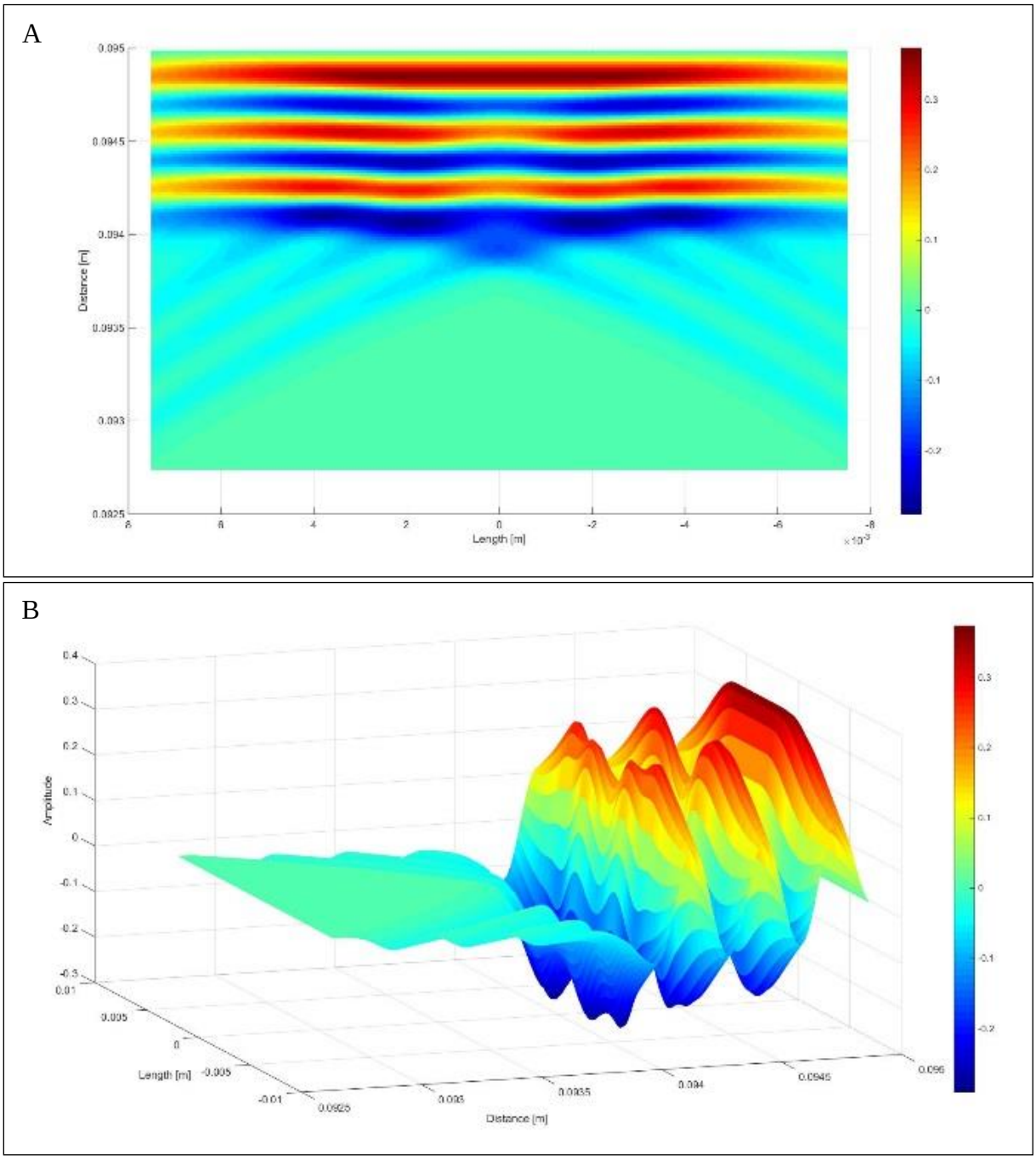


Figure 3.22 *Three cycle normalized pressure waveform at 95 mm from the prototype #1 Tx transducer. A) The edge waves can be seen in the two-dimensional plot. B) The three-dimensional plot clearly shows the variability along the compression and rarefaction fronts, due to diffraction, in all but the leading compression peak.*

Additionally, the reference plate at 50 mm is also in the near field. Within the near field the pressure impinging on the reference face is not isotropic. The resulting pressure gradients will

induce shear waves in the reference plate. Shear waves will further interfere with the acoustic wave and its reflections within the reference plate. This results in fluctuations in the acoustic intensity at the receiver that are not a function of the density of the seawater sample.

Figure 3.23 shows a plot of the normalized pressure amplitude for a cross section through the centre of the sensor coincident with the long axis of the radiating face. The wave is depicted as it is about to hit the reference plate of prototype #1. The pressure amplitude is normalized to the pressure emitted at the transmitter transducer face. The pressure variations along the crests and troughs of all but the first half cycle will induce shear waves in the carbon vitreous reference material. The shear wave speed for carbon vitreous is unknown.

Since prototype #1 had already been built, it was necessary to try to minimize the diffraction induced errors. Figure 3.24 shows the model results using a single cycle at the reference plate where shear wave generation is the primary concern. The first half cycle shows a relatively isobaric distribution across the reference plate face. The second half cycle shows pressure gradients across the reference plate face which will induce shear waves. However, the result is far superior to the 3 cycle model.

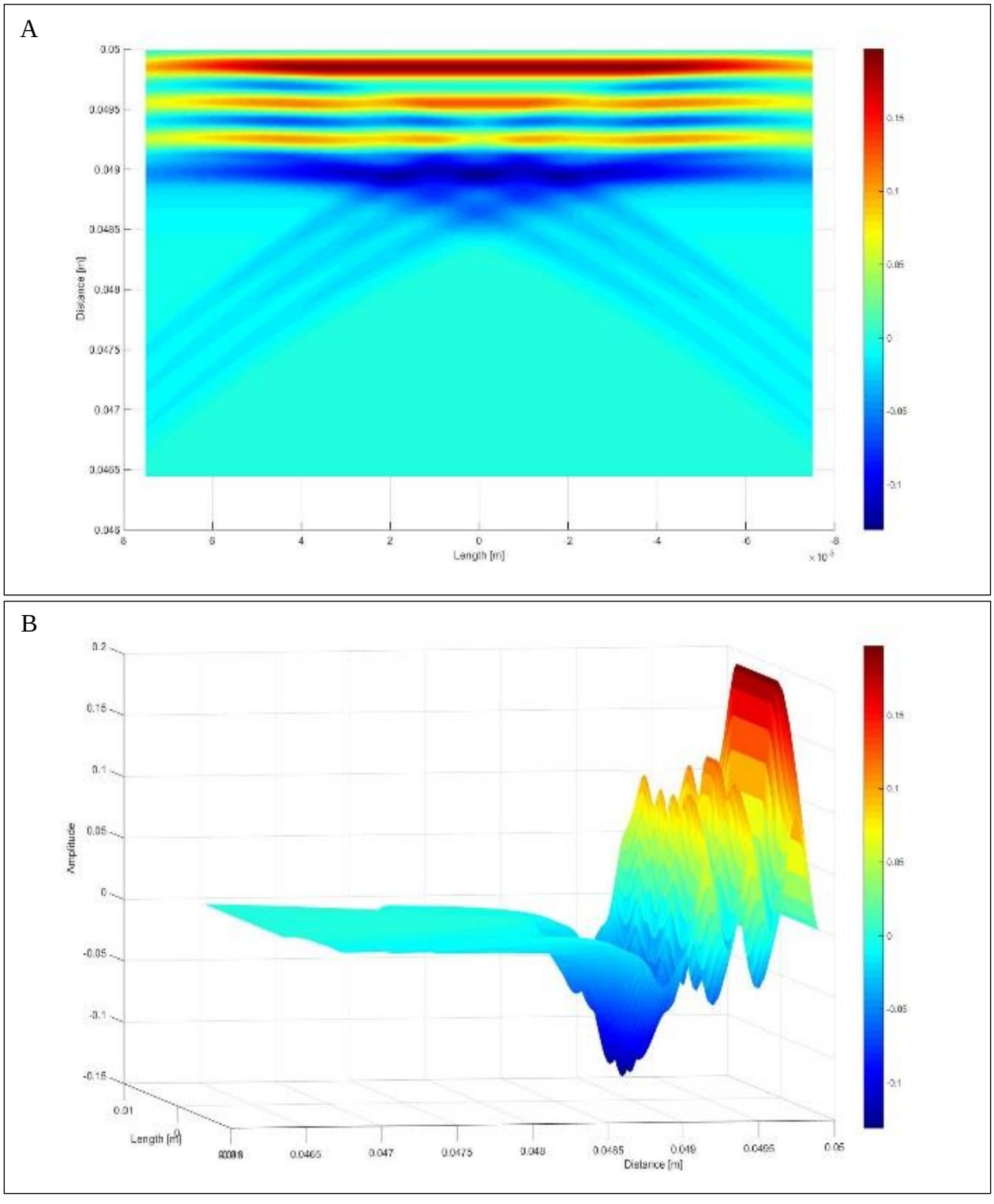


Figure 3.23 *Three cycle normalized pressure waveform at the reference plate face for prototype #1 The pressure variations along the wave peaks and troughs will induce shear waves in the carbon vitreous reference plate.*

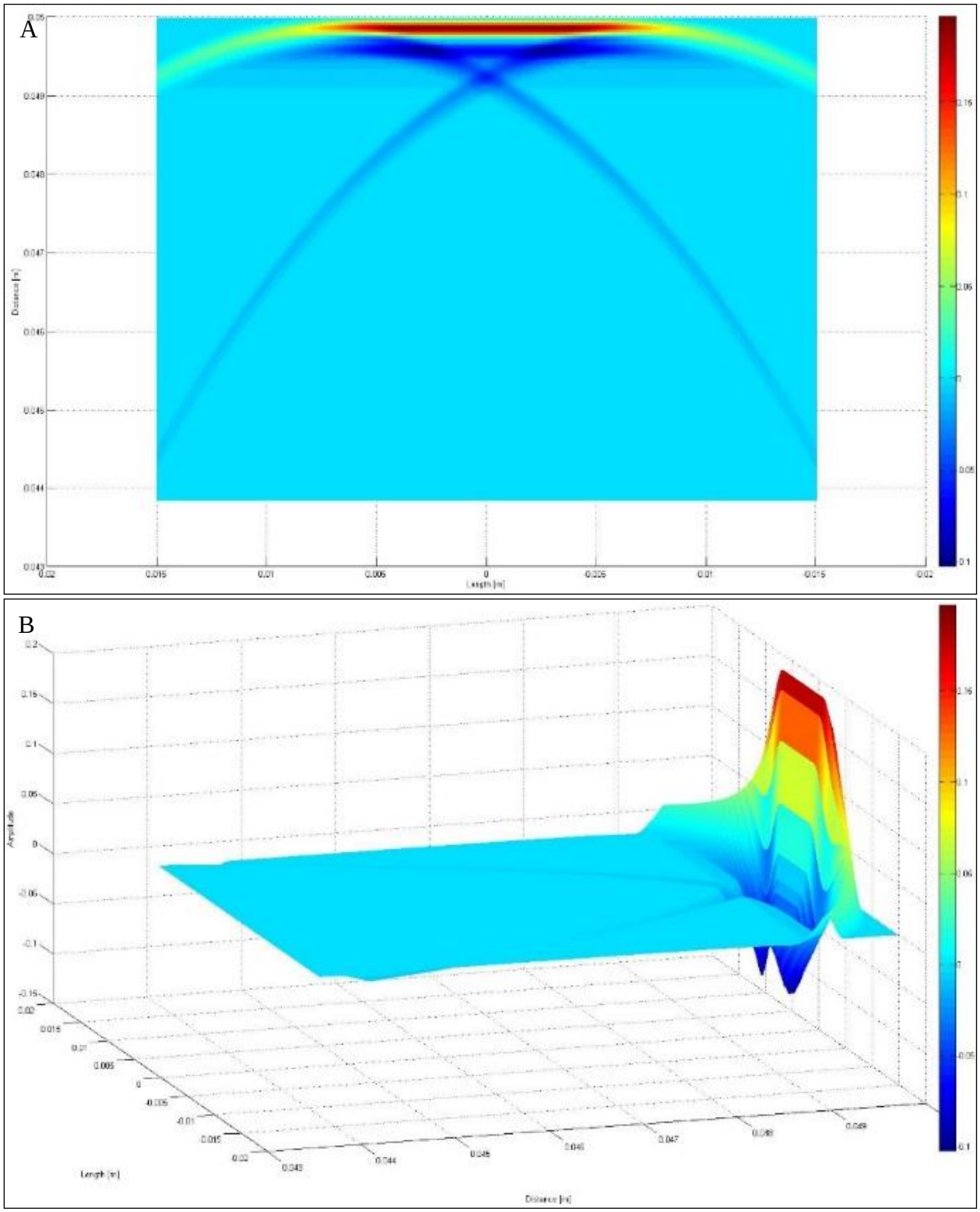


Figure 3.24 *Single cycle normalized pressure waveform at the reference plate face for prototype #1.*

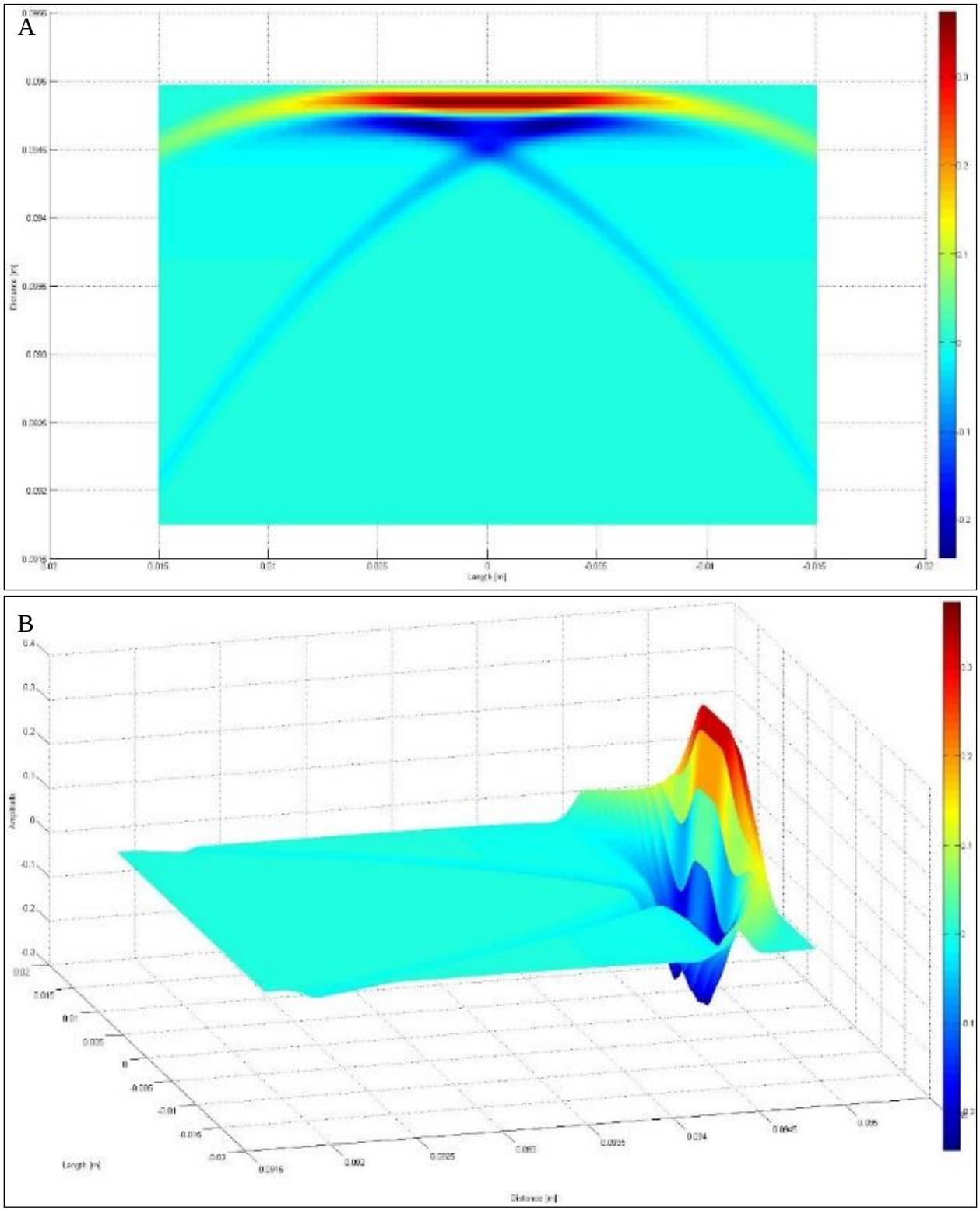


Figure 3.25 *Single cycle normalized pressure waveform at the receiver transducer for prototype #1.*

Figure 3.25 shows the single cycle pressure wave at the receiver face. The second half cycle shows reduced gradients, although they are still significant given the desire to achieve 30 ppm accuracy with the sensor.

To assess the effect of diffraction on the wave amplitude, the 1 cycle model was run at sound speeds of 1410, 1450, 1490, 1530 and 1570 m/s. Since the model uses a 1-dimensional receiver, the pressure can be averaged along the receiver line. Figure 3.26 shows the receiver acoustic pressures at various sound speeds, normalized to the transmitted pressure, averaged along a line through the centre of the receiver. The variations at the peaks are 9% in the leading compression phase and 18% in the rarefaction phase with a change in sound speed of 160 m/s.

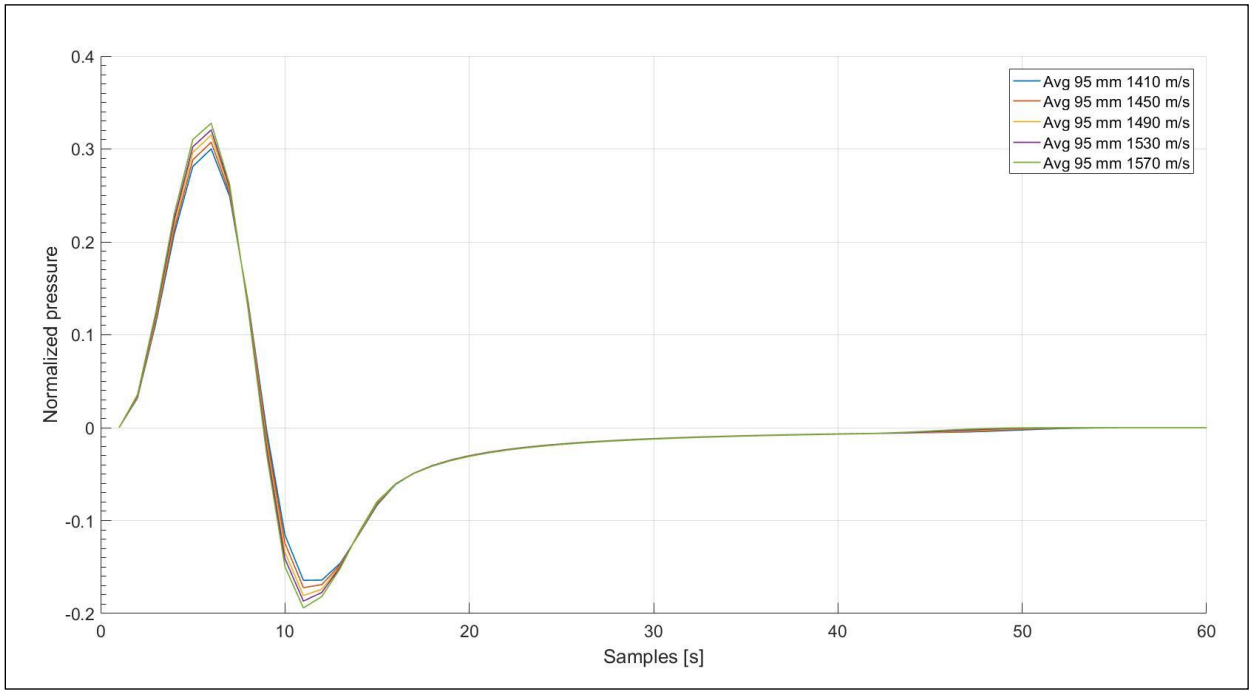


Figure 3.26 *Modelled variability of pressure amplitude due to diffraction for prototype #1. Plots are shown for various sound speeds for a single cycle acoustic wave. Variations are 9% in the leading compression phase, and 18% in the rarefaction phase with a change in sound speed of 160 m/s.*

As a result of these findings, prototype #2 (figure 3.18) was constructed with smaller transducer dimensions and longer water path lengths to mitigate the diffraction problem both at

the receiver and the reference material. As discussed in section 3.1.15, prototype #2 utilized stainless steel spacer rods rather than carbon vitreous rods. This change introduces thermal expansion induced errors; however, it would allow the diffraction problem to be assessed at a lower cost. The same digitizing electronics was used for both prototypes.

3.2.2 Asynchronous clocks

Jitter in the digitized data captures is caused by the lack of synchronization between the transmit clock and the ADC clock. The asynchronous clocks result in signals which can effectively start ± ½ of the ADC sampling period. The jitter is clearly seen in figure 3.27. As a result, there can be up to one sampling period of phase shift between signals in identical conditions. In future instruments this could be solved by synchronizing the ADC and transmit signal generation clocks.

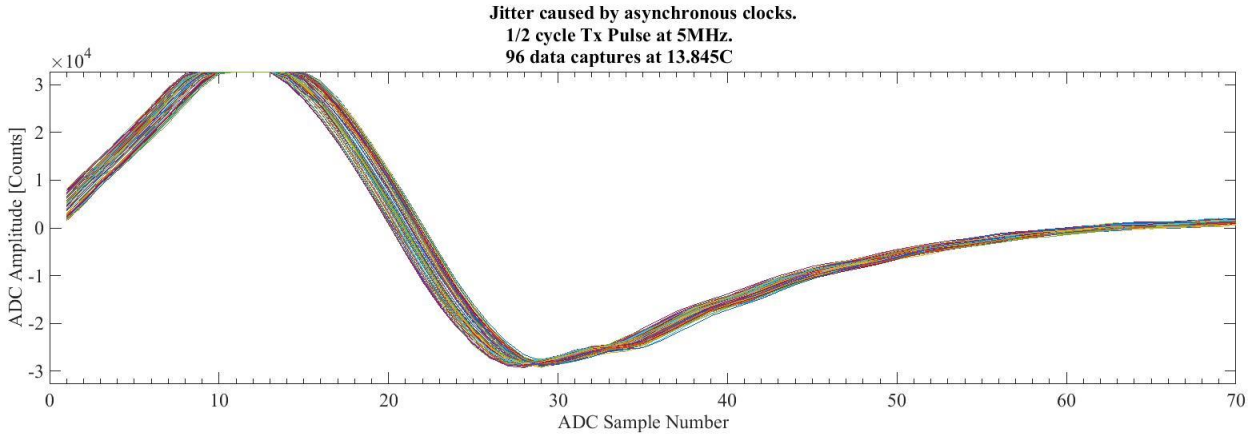


Figure 3.27 *Signal time jitter caused by asynchronous electronic clocks. Results for 96 data captures at a constant temperature.*

If the prototype could use a single data capture, then the signals used in the reflection coefficient and timing calculations would have the same clock error, with no averaging, so the density measurement would be accurate. However, the prototype sensor uses averaging of many

data captures to reduce the electrical noise. Averaging time shifted signals will always produce a low amplitude estimate of the ideal signal, which will introduce an error in the amplitude measurement of the signals used for the pressure ratio calculation. Figure 3.28 shows the modelled RMS amplitude error for time shifted sine wave averaging. For an equally distributed half sample cycle time jitter, averaging can reduce the amplitude error to 1600 ppm. For averages of 200 or more scans the amplitude error is stable and the error is applied nearly equally to both signals used in the pressure ratio calculation. As a result, the jitter induced error is mitigated by averaging 200 or more scans and using a ratiometric measurement of the signals.

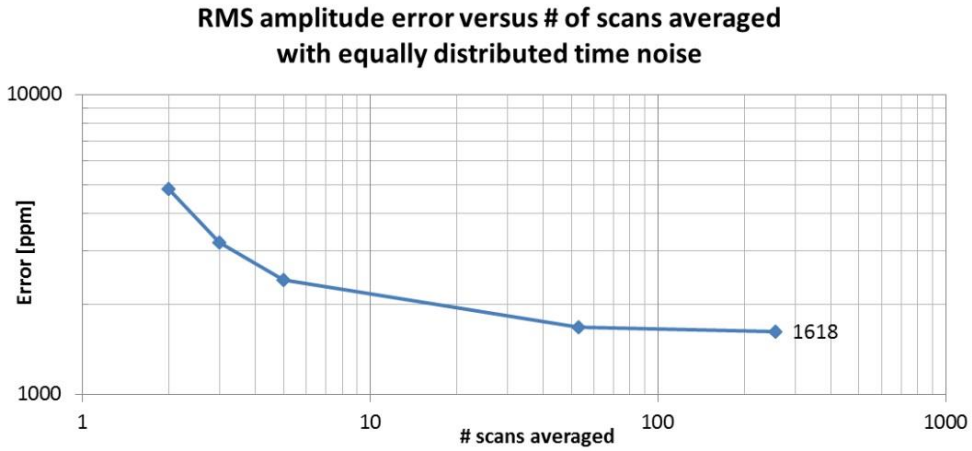


Figure 3.28 *Modelled sinusoidal reduction in amplitude error by averaging. RMS amplitude error versus the number of scans averaged with equally distributed time noise.*

3.2.3 Amplitude accuracy with respect to noise

For prototype #1, the original electronics produced baseline noise with a standard deviation of 515 digitizer counts (approximately 10 mV) with peaks at 3000 counts as shown in figure 3.29. Since the peak signals for the third carbon vitreous signal are only 5000 counts, the electrical noise made the timing and amplitude measurements unreliable.

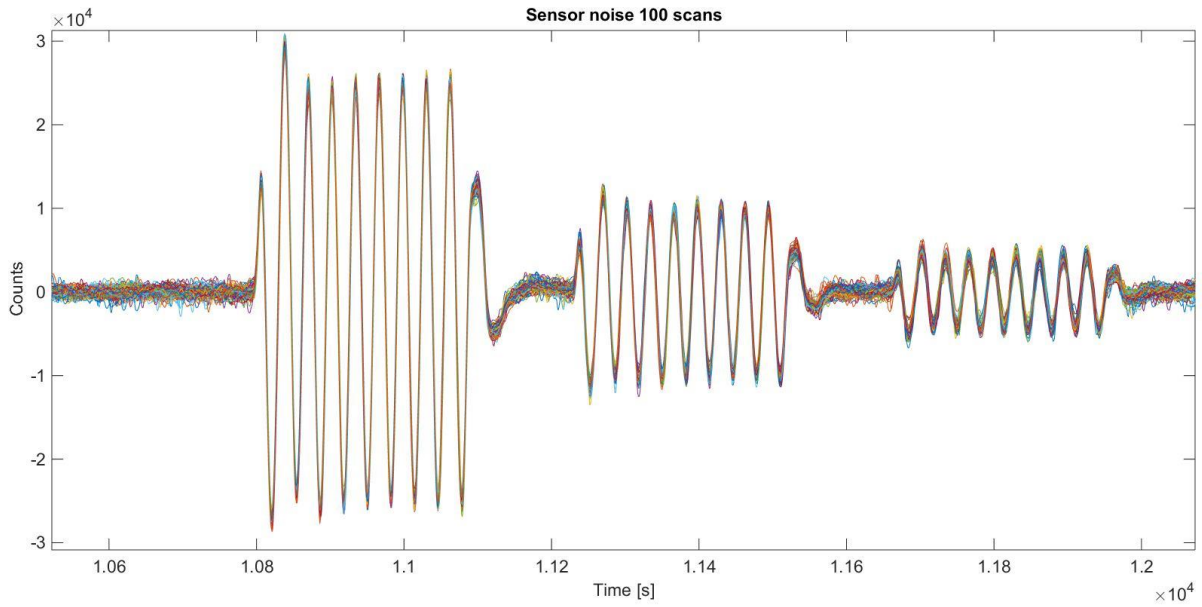


Figure 3.29 *Noise and signal for 100 scans with the original electronics and prototype #1.*

Removing the commercial variable gain amplifier and anti-aliasing boards and redesigning the preamplifier to compensate for the loss of gain reduced the noise to a standard deviation of 93 and a peak of 600, as shown in figure 3.30. The preamplifier change, in addition to diffraction problem discussed in section 3.2.1, required a new prototype sensor head to be constructed, prototype #2. Averaging further reduced the noise to a standard deviation of 9 counts and a peak of 125 counts, which improved the signal to noise ratio as shown in figure 3.31.

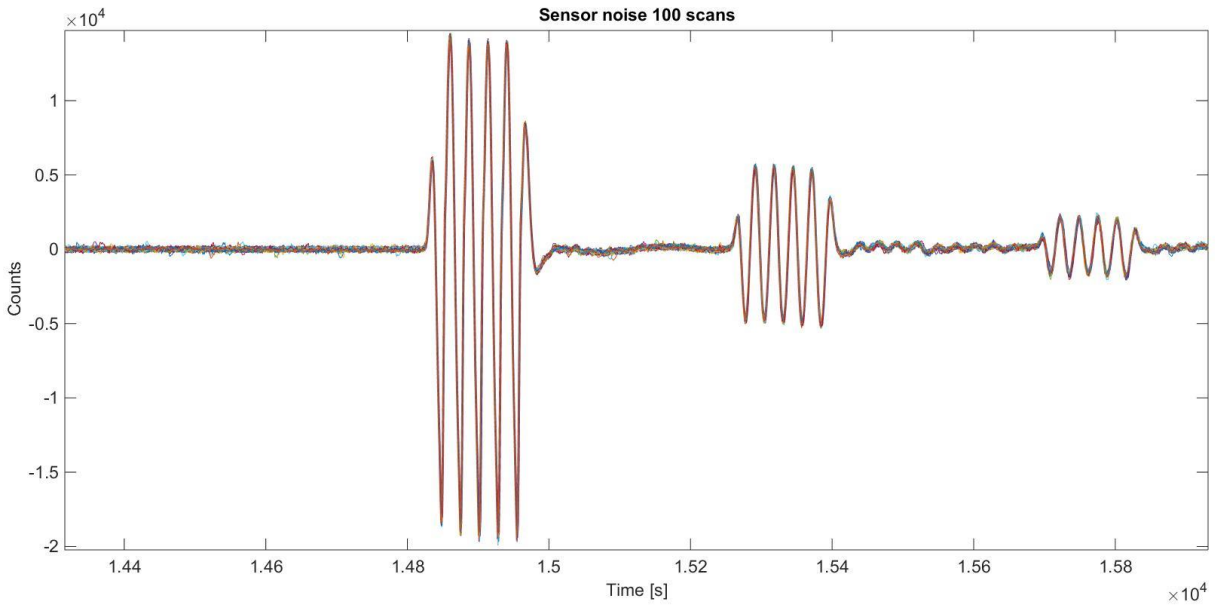


Figure 3.30 *Noise and signal for 100 scans with improved electronics and prototype #2.*

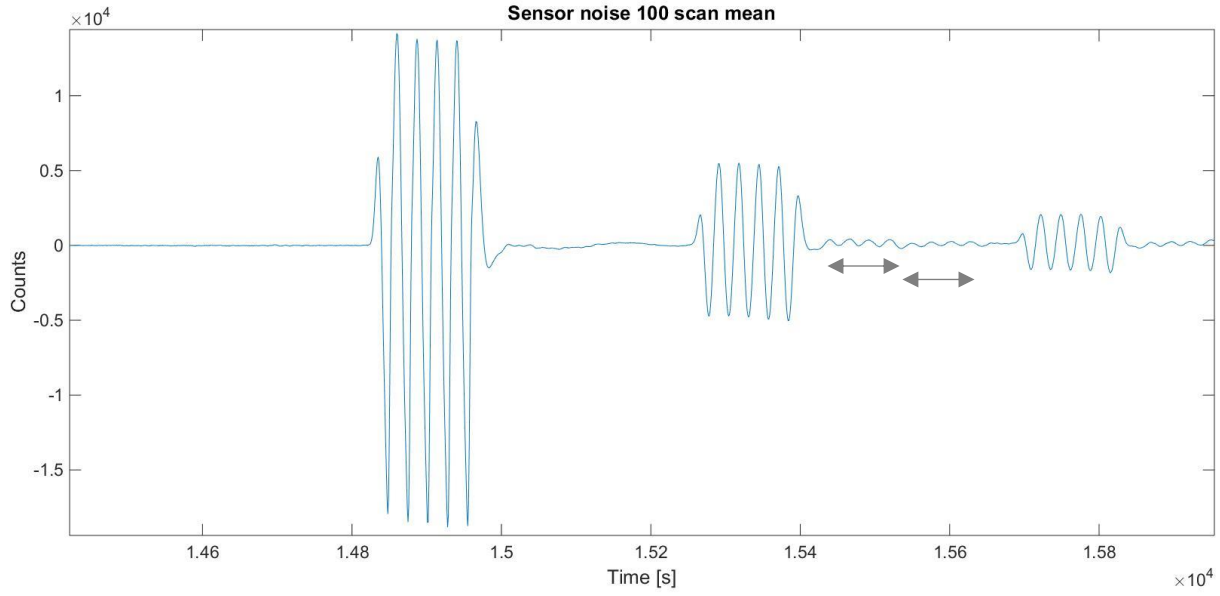


Figure 3.31 *The signal to noise ratio of figure 3.30 is improved by averaging. Note the evidence of two shear waves indicated by the arrows; these are discussed in section 3.2.4.*

Noise in the signal affects the amplitude measurement. Envelope peak measurements will be affected the most due to addition of the noise peaks and the signal peaks. RMS measurements will be significantly affected since they will include noise at all frequencies. FFT amplitude

measurement will be affected but, only by the noise within the frequency bin of the desired signal and any spectral leakage from neighbouring bins.

Table 3.2 shows the theoretical estimated error generated by various noise levels for a 16 bit resolution version of the instrument (applicable to both prototypes). The first echo amplitude is set to 0.8 full scale to avoid saturating the digitizer during operations at various temperatures, pressures, and densities. This has the undesirable effect of reducing the signal to noise ratio. Since the second echo is less than half the amplitude of the first echo, the signal to noise ratio is further degraded, increasing the measurement error for the weaker signal.

Table 3.2 *Error estimates for various peak noise amplitudes.*
Calculated for a 16 bit signal with the maximum signal set to 80% of full scale.

ADC resolution	16	bits
Full Scale	65536	counts
FS Peak	32768	counts
80% FS Peak	26214	counts

Peak Noise [Counts]	Error [ppm]
10	381
20	763
40	1526
60	2289
80	3052
100	3815
150	5722
200	7630
300	11444
400	15259
600	22889
800	30518
1000	38148
1200	45777

3.2.4 Shear waves

There is evidence of two shear waves generated in the carbon vitreous in figure 3.31, at 155 and 156 μs. It is likely these waves are mode conversions at the two carbon vitreous faces. This is probably due to diffraction induced pressure gradients along the faces. As a result, only the first two signals (149 and 153 μs) can be used for the pressure ratio calculation. The diffraction problems preclude the use of higher order internal reflections that could have helped with the resolution and accuracy improvements described in section 3.1.

3.2.5 Timing accuracy

Electrical noise can have a large impact on the timing measurements. This is especially true for comparator triggered timing circuits. However, correlating the signals over several cycles reduces the electrical noise impact.

Diffraction will affect the waveform shape and correlation symmetry, and therefore affect the timing accuracy. The not-quite-planar wave front will add a phase delay to the first half cycle. The following cycles will be subject to further constructive and destructive interference which will create an additional phase shift. This is addressed in the diffraction section 3.2.1.

The timing accuracy requirements for the sensor demand better resolution than the 6.25 ns digitizing rate. Parabolic interpolation between samples provided this improvement. Measurements of the carbon vitreous acoustic travel time were made with a standard deviation of 25 ps and a range of 84 ps. The standard deviation is a 250 times improvement over the digitizing rate. Since the instrument requires relative time, not absolute time, the 25 ps standard deviation can be considered the instrument timing accuracy.

There are some spikes in the carbon vitreous timing data which affect the timing accuracy estimate. Since the reference plate is a solid there is no flow noise. The variability may be due to resonant bubbles on the carbon vitreous faces. This is discussed further in section 3.2.11.

3.2.6 Sound speed in the reference disk

For the method developed in section 3.1.3, the sound speed in the carbon vitreous is not required for the water density measurement. However, the method assumes the reference material plate and the spacer rods have the same thermal and pressure characteristics. There was a difference in the pre-prototype sound speed testing of the carbon vitreous using an oscilloscope. The test samples, discussed in section 3.1.3, had a sound speed of 4454 m s^{-1} and the prototype material had a sound speed of 4471 m s^{-1}. The manufacturer attributes the batch difference to the variability in the heating process during fabrication of the material. Unfortunately, the manufacturer had no data on the variability of the carbon vitreous CTE or sound speed from batch to batch, or with temperature.

Although the carbon vitreous CTE could not be measured by the author, sound speed measurements were possible. Using a path length of 6.030 mm (measured by micrometer), the sound speed was measured with prototype #2 at 4471 m s^{-1} at 20°C. The carbon vitreous sound speed could be measured with two orders of magnitude better resolution using the density sensor electronics as compared to the pre-prototype test set up with the digital oscilloscope. This sound speed precision provided the opportunity to examine the carbon vitreous sound speed temperature response. The results are shown in figure 3.32. The path length was only measured at 20°C so the sound speed is only accurate at that temperature. Without path length thermal compensation, the apparent carbon vitreous sound speed coefficient of thermal expansion (CTE)

is 0.01 m s^{-1} C^{-1} or 2.24 ppm C^{-1}. This indicates the material is relatively stable with temperature and any differences in temperature between the carbon vitreous batch testing could not have caused the difference in observed sound speeds. This confirms the variation in sound speed is batch dependent.

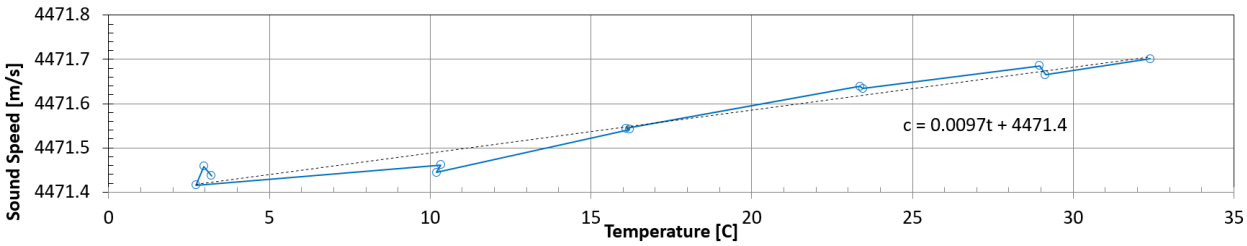


Figure 3.32 *Apparent sound speed for carbon vitreous. The results are based on interpolated timing counts for the internal reflection signal in prototype #1. This plot does not account for thermal expansion of the carbon vitreous. The length of the carbon vitreous was measured at 20°C. Multiple data points were taken at each temperature below 30°C and each data point was computed from a 200 scan average to reduce the effects of clock jitter.*

3.2.7 Path length measurement

As discussed in section 3.1.2, for the density sensor, a mechanical measurement of the water path length cannot be made accurately enough, so a method was developed to mitigate the requirement for an accurate path length measurement for the density sensor. However, since the acoustic time of flight measurement was made in the water path between the reference and the receiving transducer, it was a simple matter to add an estimate of the water sound speed to the instrument output. The water path length was estimated by sound speed and time measurements. A single, 20°C, distilled water, TEOS-10, calibration point was used to estimate water path length. For prototype #2 the estimated water path length was 59.347 mm.

The path length of the reference material, at 20°C, was measured with a micrometer at 5 locations and the average was 6.030 mm. The peak variability of the measurement was 0.001 mm.

3.2.8 Path length thermal expansion

Equation (3.10) was developed to eliminate the thermal expansion issues between the water and the reference. This assumes the CTE is the same for the rods and the resonator. Since the rods and the resonator would likely come from different carbon vitreous batches, and since the carbon vitreous test samples and prototype reference plates had different sound speeds, the CTE could in fact differ between the rods and the resonator. Although carbon vitreous has excellent acoustic properties, the potential variability in the CTE between batches indicates carbon vitreous may be a poor choice for the ISDS. Further testing is needed to determine if plates and rods, formed with the same raw material and fired together, will have the same thermal properties. The traceability requirements for the processed parts could add considerably to the carbon vitreous cost.

3.2.9 Reference density

The carbon vitreous manufacturer specifies the nominal density as 1.42 g cm^{-3}, although this is not measured for each batch. Measuring the test sample density by way of water displacement yielded a density of 1.49 kg m^{-3} at 20°C. This agrees with Yamada and Sato's (1962) reported density range of 1.46 to 1.50 kg m^{-3} for 1300°C fired carbon vitreous. The manufacturer's nominal linear CTE is 2.6 ppm C^{-1} which yields a volume CTE of 7.8 ppm C^{-1}, or −0.012 kg m^{-3} C^{-1}, given constant mass. For the 32 ppm accuracy requirement from table 1.2, the change in

reference material density requires thermal compensation for temperature variations of ± 4°C from the nominal calibration temperature. It should be noted that the manufacturer's CTE was not verified; however, it would require an order of magnitude reduction in the carbon vitreous CTE to mitigate the need to thermally compensate the carbon vitreous density change for ocean temperatures.

3.2.10 Thermal response time of the sensor materials

Thermal response time is only a problem for profiling applications where the temperature can change rapidly. To estimate the necessity for thermal response time compensation, equation (3.7), can be used with the CTE for carbon vitreous and the sensor carbon vitreous sound speed measurements to determine the theoretical water density measurement error due to a step change in temperature. Table 3.3 shows the changes at the sensor due to a 10°C step change using the dimensions for prototype #2. The initial values are given at time zero t_0, and t_∞ gives the values when the sensor has come to equilibrium. The estimated error for the 10°C change is 1.939 kg m^{-3}. Given the thermal conductivity of the carbon vitreous, 6.3 W m^{-1} C^{-1}, is about 40% that of the stainless steel used on older model sound speed sensors, the thermal response time will be several seconds. The combination of a large thermal response error and slow thermal response time renders this sensor unsuitable for high speed profiling. Since accurate density profiles are critical to ocean current models, this is another indication that carbon vitreous may not have been the best material choice for the reference material. From a thermal response time perspective, a low thermal expansion glass ceramic would be a better choice; however, the density resolution issue described in section 3.1.2 would need to be tested.

Table 3.3 *Theoretical thermal response time error for a 10°C step change.*

	at t_0	**at t_∞**	**dt**
Carbon vitreous temperature [C]	20	10	-10
Carbon density [kg m^{-3}]	1490	1490.12	1.20E-01
Carbon path length [m]	0.01206	0.012034	-2.60E-05
Carbon sound speed [m s^{-1}]	4471.59	4471.49	-0.1
Carbon impedance [kg m^{-2} s^{-1}]	6662669	6663057	388
Carbon travel time [s]	2.697027E-06	2.691273E-06	-5.75E-09
Water temperature [C]	10	10	0
Water density [kg m-3]	999.702	999.702	0
Water path length [m]	0.12	1.199740E-01	-2.60E-05
Water sound speed [m s^{-1}]	1447.284	1447.284	0
Water impedance [kg m^{-2} s^{-1}]	1446853	1446853	0
Water travel time [s]	8.291393E-05	8.289596E-05	-1.80E-08
Path length ratio scaling	0.100305	0.100305	0
Pressure reflection Coefficient	-0.64317188	-0.643188933	-1.71E-05
Sensor water density estimate [kg m^{-3}]	**997.763**	**999.702**	**1.939**

3.2.11 Density measurement from developed theory

The reference water density for the ISDS testing was produced by dilution of Atlantic standard seawater and the practical salinities were measured before and after the ISDS tests using a Portasal salinometer. The practical salinity was used to compute the absolute salinity and this was used to compute the density. All computations used the gsw MatLab toolbox (www.TEOS-10.org, 2014).

The sensor and water sample were allowed to stabilize for 4 to 15 hours at a temperature 2ºC higher than the measurement temperature before each test. This allowed the water to off-gas to prevent bubble formation. The bath temperature was then reduced to the measurement temperature.

The density measurement results for prototype #2 are shown in figure 3.33 for distilled water and figure 3.34 for saline water. Here signals 1, 2 and 3 refer to the first three signals arriving at the receiver, as shown in figure 3.31 (not the numbered signals of figure 3.5).

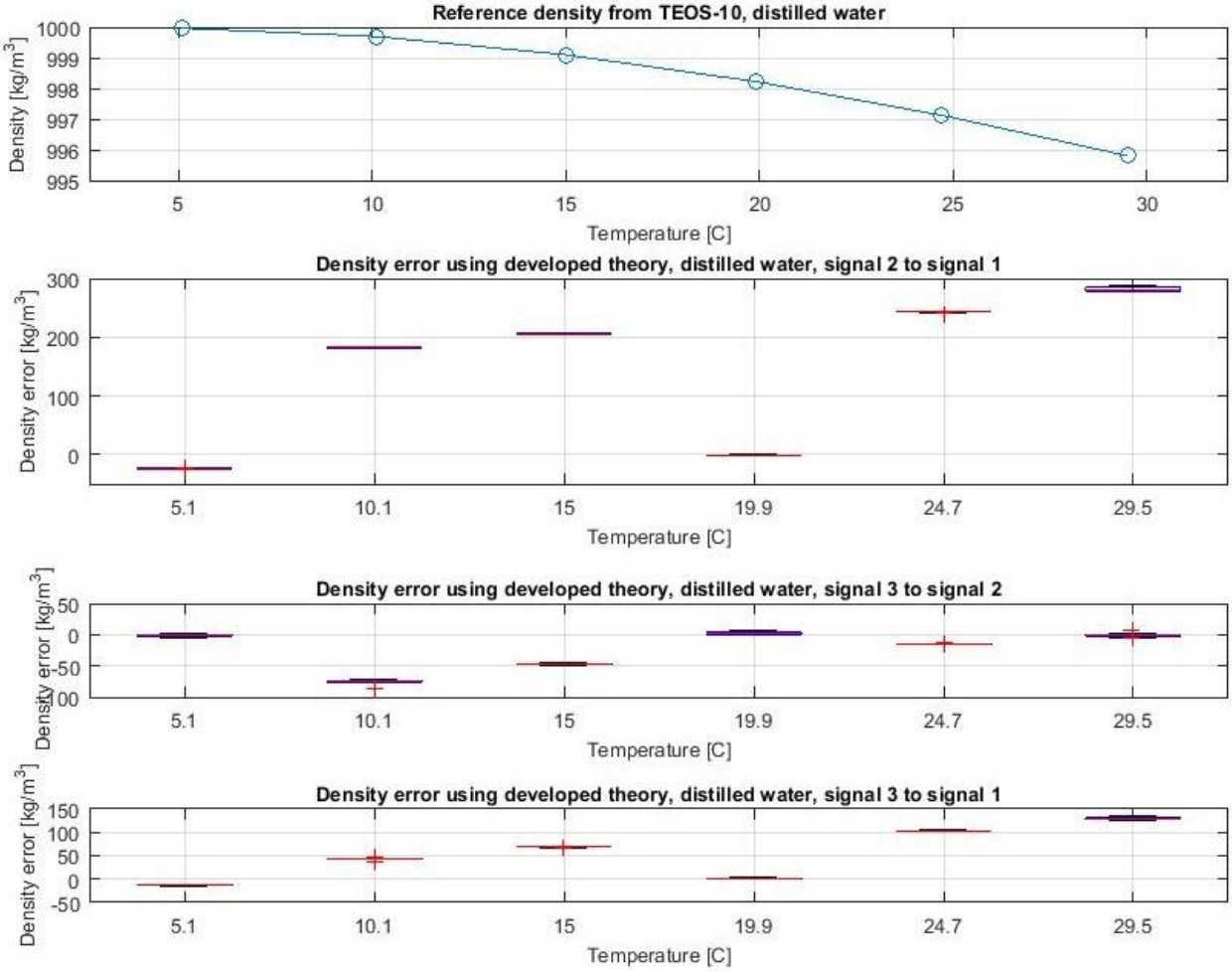


Figure 3.33 *ISDS density measurement errors in distilled water.*

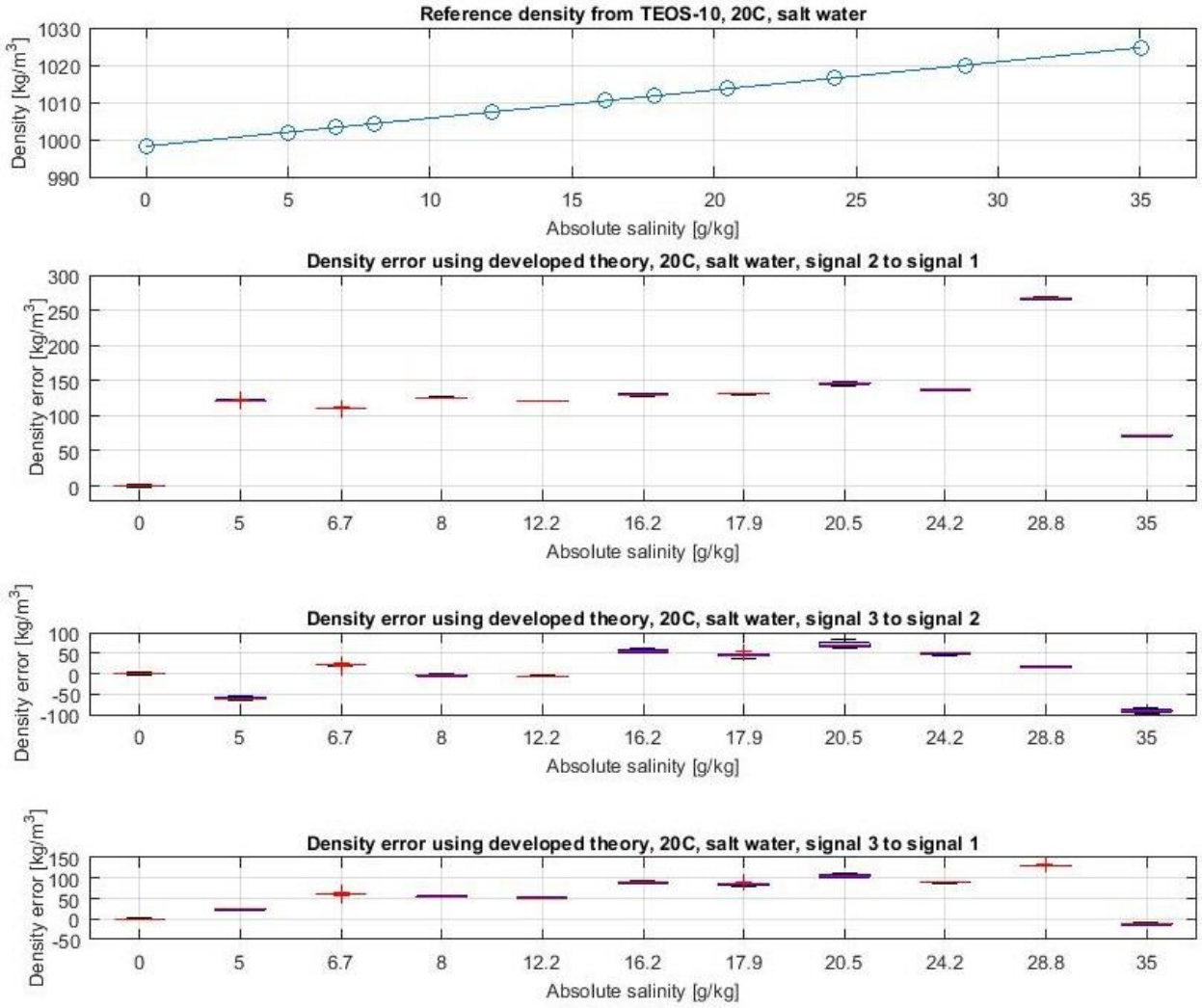


Figure 3.34 *ISDS density measurement errors in saline water.*

The box plots in figures 3.33 and 3.34 of the errors for ISDS prototype #2 were generated from 20 measurements at each water temperature and salinity point. For each measurement 200 acoustic scans were averaged to reduce the noise and clock jitter.

While the reflection coefficient values seemed stable from measurement to measurement, in the tens of seconds time frame, the measurements varied on tens of minutes time scales. The magnitude of the variability was also seen to decrease with time after the water sample reached a new temperature. The least variability occurred when the water was allowed to thermally

stabilize for 7 hours. Additionally, when the sensor was removed and replaced in the water, or when the sensor was wiped underwater, the pressure ratios changed significantly, as shown in figure 3.35. These pressure ratio errors create a density estimate variability of 400 kg m^{-3}. The ISDS errors are four orders of magnitude larger than the target of 0.03 kg m^{-3}, making the sensor as built unsuitable for oceanographic use.

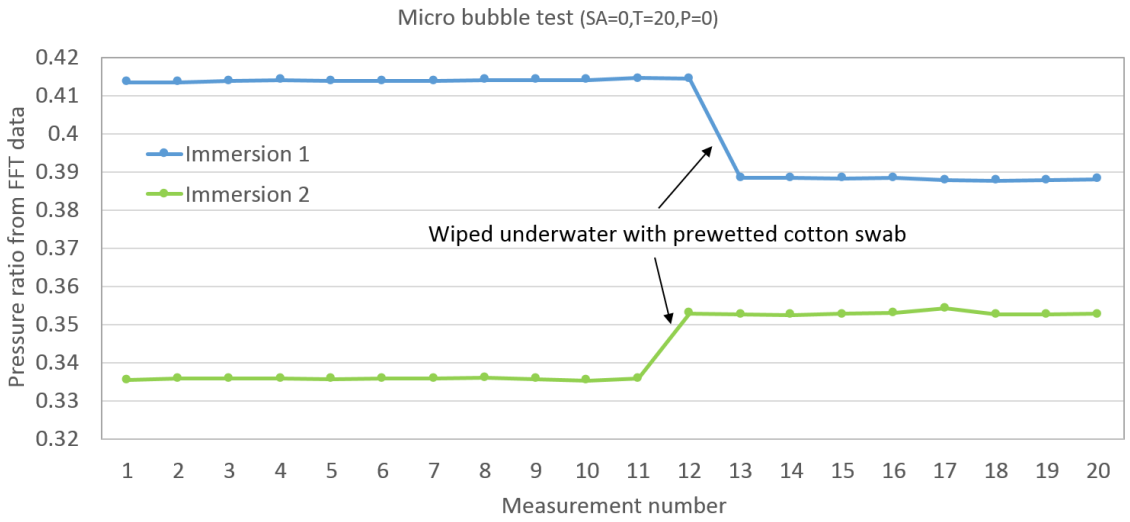


Figure 3.35 *Change of the pressure reflection coefficient with dry immersions and underwater wipes of the reference plate faces, in distilled water at 20°C.*

Unfortunately, dissolved gasses can form microscopic bubbles on surfaces in air saturated water. The huge variability in the reflection coefficients may be due to this formation. The variations shown in figure 3.35 are likely due to a redistribution of bubbles, caught on the reference plate surface micro-structure, during immersions and wiping. Von Rohden et al. (2015) found evidence of micro bubbles affecting the sound speed sensor measurements as well.

The sound speed sensor averages the micro bubble induced error within the water path length, so the errors are small. The ISDS uses an axially microscopic length for the pressure

ratio measurement which magnifies the impact of any contamination. This is likely the reason the pressure reflection coefficient approach to density measurement failed.

It is possible that a rubber squeegee wiper would give better results than the cotton swap for clearing the bubbles. Polishing the mechanically ground reference faces could allow the faces to be more easily wiped free of bubbles. This would also reduce the number of bubble nucleation sites. The mixing pump may have cavitated the water and entrained bubbles which deposited on the reference plate faces. It is possible that many resonant sized bubbles (0.65 μm radius) could have formed in the crevices in the surface finish of the reference plate. Lowering the acoustic frequency to 1 MHz might resolve the resonant bubble issue. Unfortunately addressing these issues will not overcome the hyper sensitivity of the sensor to interface contamination. As a result, the reflection coefficient based density sensor approach was deemed to have a low chance for success as an *in situ* sensor and work on this sensor was halted.

One compelling reason to halt the density sensor research was the incidental sound speed measurements that were computed using the density sensor proved to be very accurate and provided a different avenue to density measurements, as discussed in the following section.

3.2.12 Density measurement from sound speed

Although the ISDS prototypes were not designed to measure sound speed it was a simple matter to compute sound speed from a single point calibration of the path length at 20°C in distilled water. The sound speed was tested to see if the advances developed for the ISDS could improve the accuracy of future sound speed sensors. The sound speed measurements were collected simultaneously with the ISDS prototype #2 density measurements. The distilled water

sound speed results are plotted in figure 3.36, and the salt water sound speed results are plotted in figure 3.37.

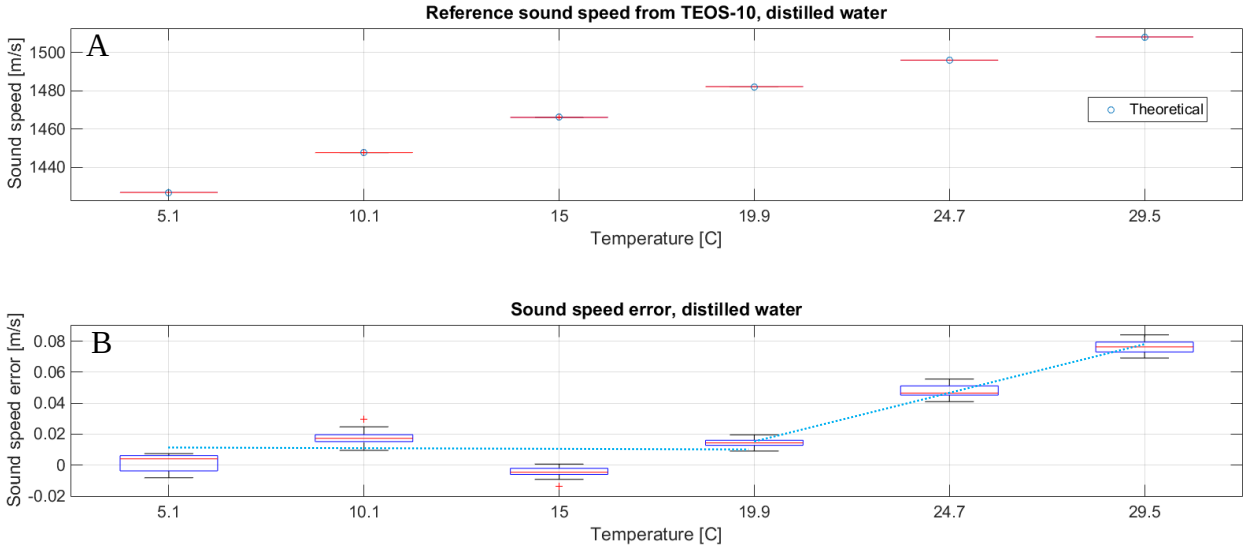


Figure 3.36 *Sound speed measurements in distilled water.*
A) Sound speed measurements in distilled water. B) Sound speed error referenced to TEOS-10. The dotted blue lines indicate a thermal related regime change, this is attributed to a change from tension to compression in the epoxy within the prototype #2 sensor.

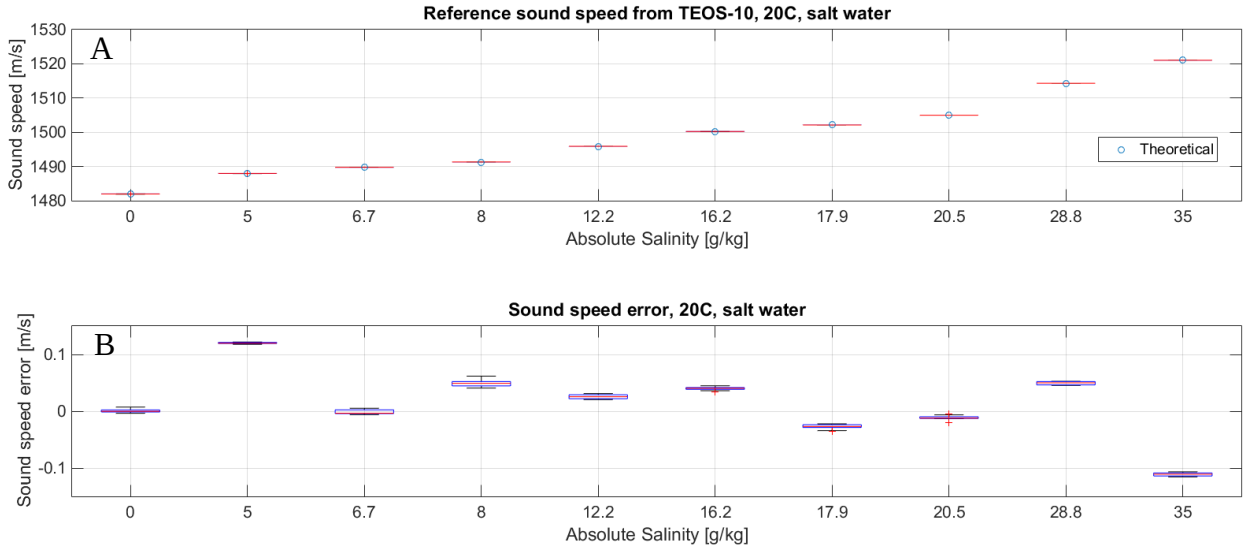


Figure 3.37 *Sound speed measurements in saline water.*
A) Sound speed measurements in saline water. B) Sound speed error referenced to TEOS-10.

The error plot in figure 3.36 B shows a linear sound speed error trend of 0.006 m s^{-1} C^{-1} above 20°C. This is likely a result of the change in strain for the epoxy thermal expansion. The prototype #1 design allowed for thermal expansion of the epoxy; however the prototype #2 design has the transducer epoxy constrained by a stainless steel housing. Below the epoxy cure temperature, 22°C, the epoxy is in tension, above the cure temperature the epoxy is in compression. Above the epoxy cure temperature, the change in path length is equivalent to 4 ppm C^{-1}.

The sound speed results are summarized in table 3.4, along with the associated density computed using TESO-10. To outperform a CTD derived density measurement error of 0.033 kg m^{-3}, from table 1.3, the sound speed measurement would have to be made with an accuracy of 0.029 m s^{-1}. The ISDS sound speed measurement meets this accuracy for distilled water and with an improved thermal strain design in the sensor head, such as used in the prototype #1 design, this could be improved further. It is possible to apply a thermally compensated calibration; however, it is difficult to assess the impact of the compensation on saline waters. This is due to the degraded TEOS-10 sound speed accuracy in saline waters, which is discussed in chapter 4. The TEOS-10 saline reference also impedes the assessment of the ISDS saline accuracy in table 3.4.

Table 3.4 *Accuracy and precision of ISDS sound speed measurements with respect to TEOS-10.*

	Sound speed accuracy m s^{-1}	[ppm]	Sound speed precision m s^{-1}	[ppm]	Density accuracy kg m^{-3}	[ppm]
Distilled	0.028	19	0.003	2	0.017	17
Saline	0.068	45	0.004	3	0.046	45
Total	0.053	35	0.004	3	0.035	34

The precision of the ISDS sound speed measurement is shown in figure 3.38. The overall ISDS sound speed precision of 0.004 m s^{-1} RMS is a factor of two better than the Micro SVX commercial sound speed sensor described in chapter 4. This is likely attributable to the higher bandwidth ISDS transducers which result in a more symmetrical correlation coefficient plot compared to the commercial sensors. This improves the accuracy of the parabolic time interpolation algorithm. Figure 3.39 shows the change in sound speed error and the change in temperature, this narrow plot with respect to temperature (and therefore sound speed) shows the sound speed precision is not a function of the temperature or sound speed, as is the case for the MicroSV sound speed sensor described in chapter 4.

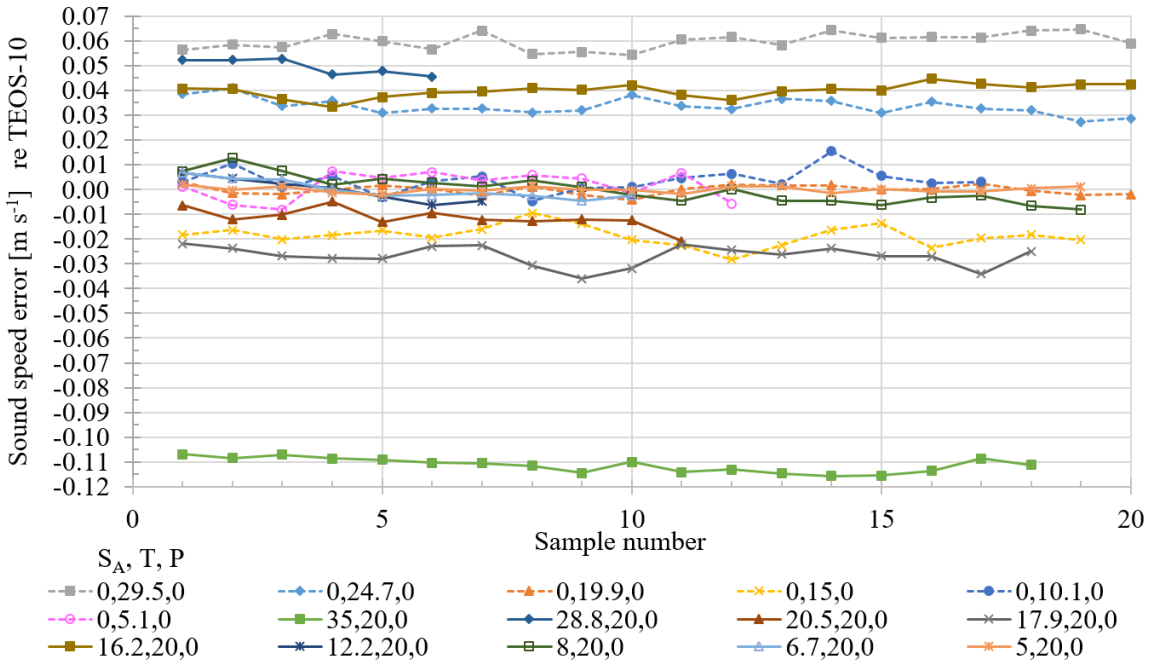


Figure 3.38 *ISDS sound speed measurements in various water types. Dashed lines indicate distilled water measurements and solid lines indicate saline measurements.*

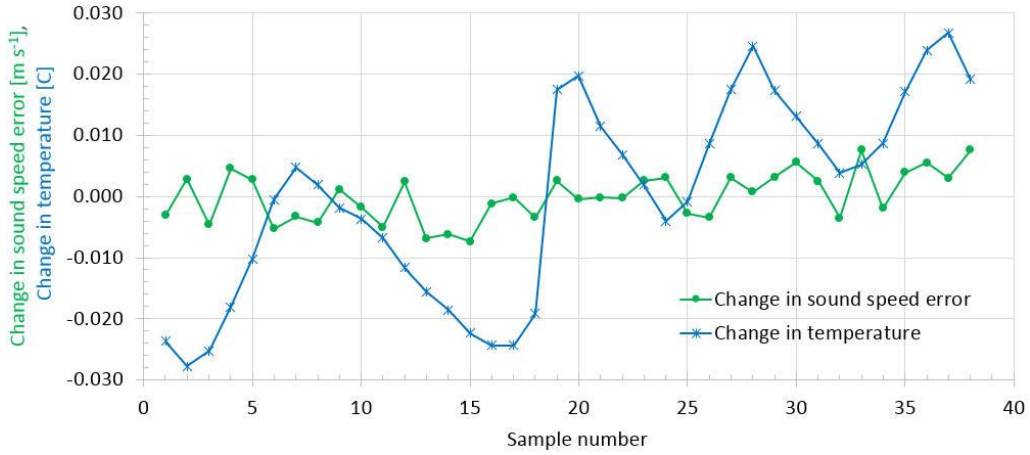


Figure 3.39 *Sound speed measurement error and temperature variability in distilled water at 29.4°C. Error referenced to TEOS-10.*

The overall ISDS sound speed accuracy and precision results are better than the commercial sound speed sensors presented in chapter 4 of this thesis. This is noteworthy since the ISDS sound speed results use a simple single point calibration and the transducers were designed to reduce ringing rather than provide a strong reflecting surface for the seawater sound speed measurement. The ISDS sound speed improvements are likely attributable to three factors: the spacer rods are impermeable to water, the digitizing rate and bit depth are much better than existing sound speed sensors and the high bandwidth transducers produce a more symmetrical correlation peak for the parabolic interpolation algorithm.

The absolute salinity and water density were computed from the simple sound speed measurement data from ISDS prototype #2. As shown in figures 3.40B and 3.41B, the density results are three orders of magnitude better than the ISDS density measurement based on the reflection coefficient shown in figures 3.33 and 3.34. The combined distilled water and salt water RMS error in density for the sound speed derived density is 0.035 kg m^{-3}. This is very close to the table 1.3 water density target of 0.033 kg m^{-3}.

The performance of the sound speed based water density estimates show that sound speed should be re-examined in detail as a potential solution to the TEOS-10 *in situ* measurement problem. This is considered in chapters 4 and 5 of this thesis.

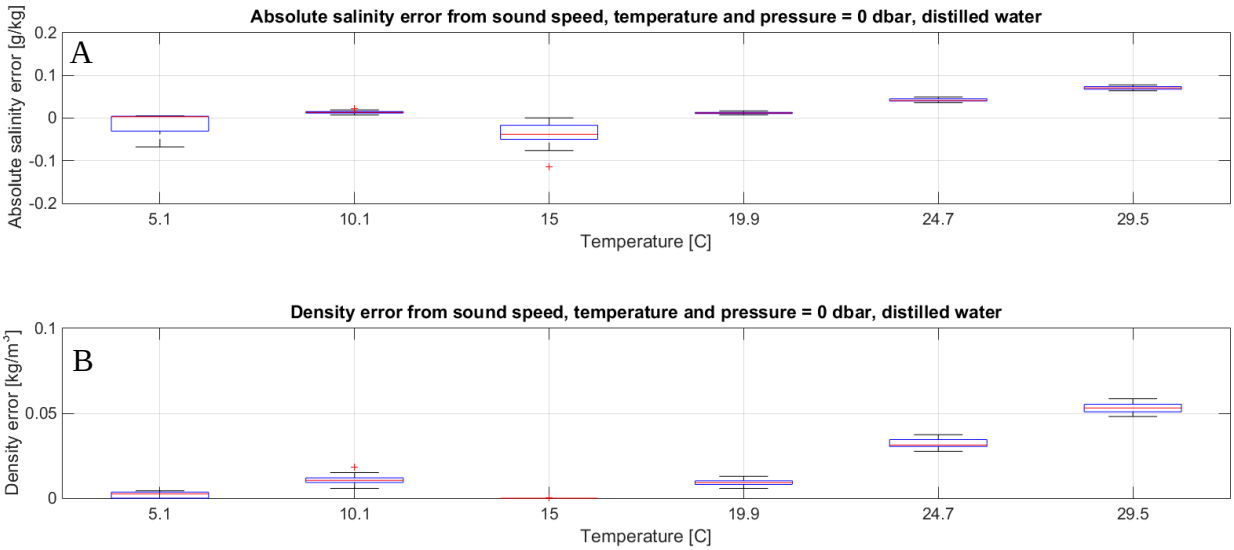


Figure 3.40 *Absolute salinity and density for distilled water.*
Results computed from the rough sound speed measurement from ISDS prototype #2. Note that the TEOS-10 MatLab toolbox will set negative absolute salinities to zero in the computation of density. As a result negative density errors in distilled water are set to zero.

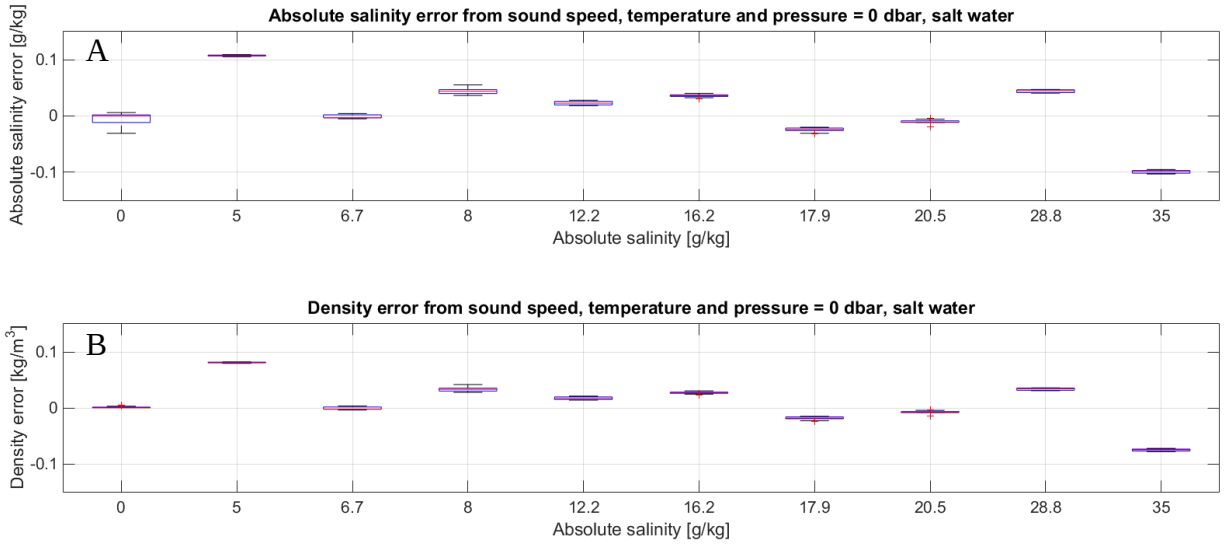


Figure 3.41 *Absolute salinity and density for salt water.*
Results computed from the rough sound speed measurement from ISDS prototype #2.

3.3 Density sensor- Conclusion

In this chapter, a method was developed to utilize the acoustic reflection coefficient to measure the density of seawater. Two prototypes were constructed and tested. The testing revealed several problems which could be mitigated such as: diffraction, asynchronous clocks, noise, interpolation of correlations, and the batch variability of carbon vitreous. The testing also revealed a hyper sensitivity to contamination of the reference plate faces (likely microbubbles), which would be very difficult to mitigate for an *in situ* oceanographic sensor. As a result, the acoustic reflection coefficient based density sensor does not appear to be suitable for *in situ* oceanographic use. However, the density sensor's simple sound speed estimate based on a single calibration point performed very well with a combined distilled and salt water RMS error of 0.035 kg m^{-3}. This water density error is very close to the 0.033 kg m^{-3} CTD based water density accuracy at time of calibration.

Many of the techniques and technologies used for the *in situ* density sensor could be applied to sound speed sensors. These include water impermeable space rods, diffraction modelling, transducer layer modelling, higher speed digitizers, better digitizer voltage resolution, and higher bandwidth transducers to improve the interpolation accuracy. Sound speed is therefore re-examined as a potential TEOS-10 *in situ* measurement solution in chapters 4 and 5.

Chapter 4

Sound speed sensor

4.1 Sound speed sensor – Background

Sound speed is one of the four methods suggested in the TEOS-10 manual (page 141) to enable TEOS-10. Given the results of the sound speed measurements from the ISDS sensor testing in section 3.2.12 of this thesis, the author's prior work on sound speed sensors (Eaton and Dakin, 1996; Dakin, 1999) and the testing performed by Von Rohden et al. (2015), sound speed sensors would seem to be a viable solution to the TEOS-10 *in situ* measurement problem.

In practice, simultaneous measurement of three parameters (most likely sound speed, temperature and pressure) are required to derive absolute salinity and density from TEOS-10. There are two problems hindering this approach: the mismatch in sensor response times, which is discussed in section 4.1.1, and the accuracy of the sound speed measurement, discussed in section 4.1.2. There are three technical developments that need to occur to mitigate these problems. These technical advances include faster temperature sensors (or slower sound speed sensors), improved sound speed sensor accuracy and a new sound speed standard.

4.1.1 Salinity spiking

The first problem, resulting from using multiple sensors with different response times, is sometimes called salinity spiking. Both CTDs and sound speed profilers (SSP) use three sensors, with each of the sensors having a different response time. Therefore, both instruments suffer from salinity spiking errors. Table 4.1 gives the response times of CTD and SSP instruments and the resulting sensor measurement lags as a function of depth that result from profiling at 0.5 m

s^{-1}. At this profiling rate, a 2000 m profile would require the ship to stop for at least 133 minutes, which is expensive. Slower profiling would reduce the spiking errors but increase the time and cost, and reduce the number of profiles that can be done per day. The different depth lags for each sensor result in an error when the three measurements are combined into a salinity estimate which is supposed to be representative of a single depth. Salinity spiking is magnified at high profiling speeds and in areas of rapidly changing water masses, such as thermoclines.

Table 4.1 *Sensor response time effects.*

	Conductivity	Temperature	Pressure	Sound Speed
Response time [ms]	25	100	10	0.03
Depth lag [m]	0.013	0.05	0.005	0.000015

Note: Response times are for CTD and SSP sensors from the same manufacturer, AML Oceanographic. The depth lag is based on a profiling speed of 0.5 m/s.

The mismatch in conductivity and temperature sensor sampling depth is therefore 3.75 cm for the CTD and for the SSP the sound speed and temperature sensor depth mismatch is 5 cm. While this depth mismatch may seem insignificant given the water depths for most ocean profiling operations, this assumption is incorrect. The effects of the mismatch are shown in figure 4.1 for a simulated CTD and in figure 4.2 for a simulated sound speed sensor. The profile speed for the salinity spiking (dashed) curves was 0.5 m s^{-1}. The 'actual' oceanographic profiles are plotted and then modified to apply the simulated response times for each sensor. The equation used for the sensor response (Liptak, 2005) is

$$M_{new} = M_{actual} - \left(M_{actual} - M_{previous}\right) e^{-\left(\frac{t_{new} - t_{previous}}{t_{response}}\right)}, \tag{4.1}$$

where M_{new} is the new measurement value at the new sample time t_{new}, M_{actual} is the actual parameter value in the water at t_{new}, $M_{previous}$ is the sensor measurement value at the previous sample time $t_{previous}$, and $t_{response}$ is the sensor response time constant.

Greater salinity and density spikes are clearly evident in the sound speed profiler data. For the typical 100 ms response time temperature sensor the profile generates a peak salinity spiking error of 0.5 g kg^{-1} for the sound speed profiler and 0.14 g kg^{-1} for the CTD. The peak density errors are 0.42 and 0.15 kg m^{-3} respectively. Thus the CTD profiling accuracy is approximately three times better than the present commercial sound speed profilers.

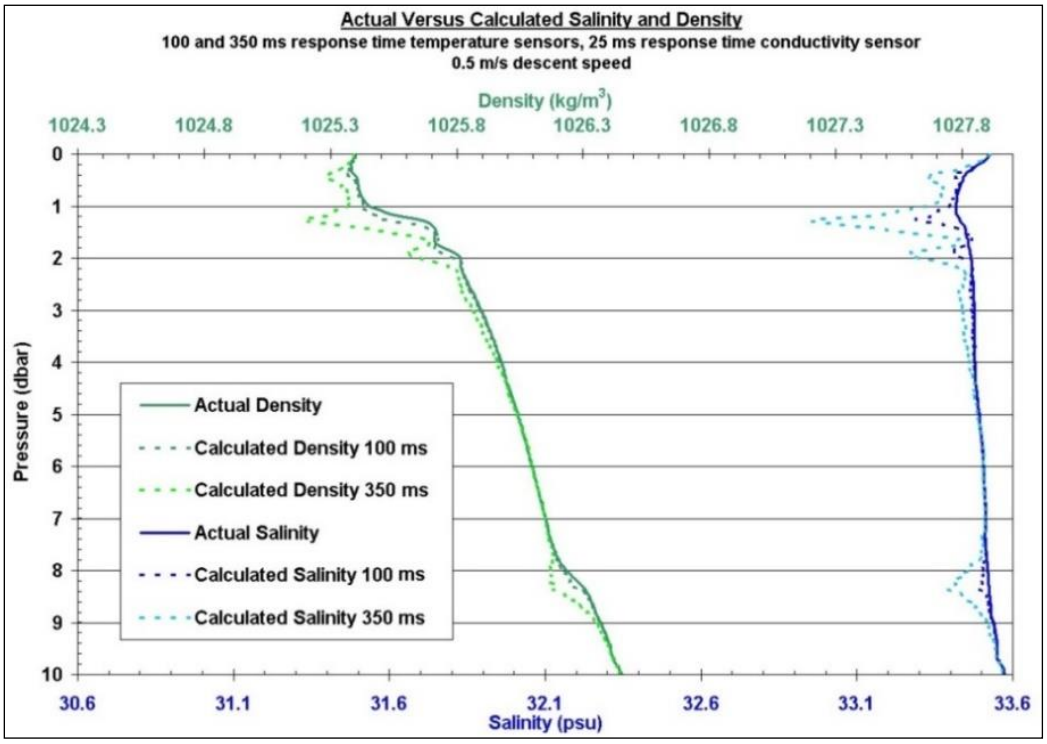


Figure 4.1 *Simulated salinity and density spiking from CTD data. Data generated by applying sensor response time lags to oceanographic profile data.*

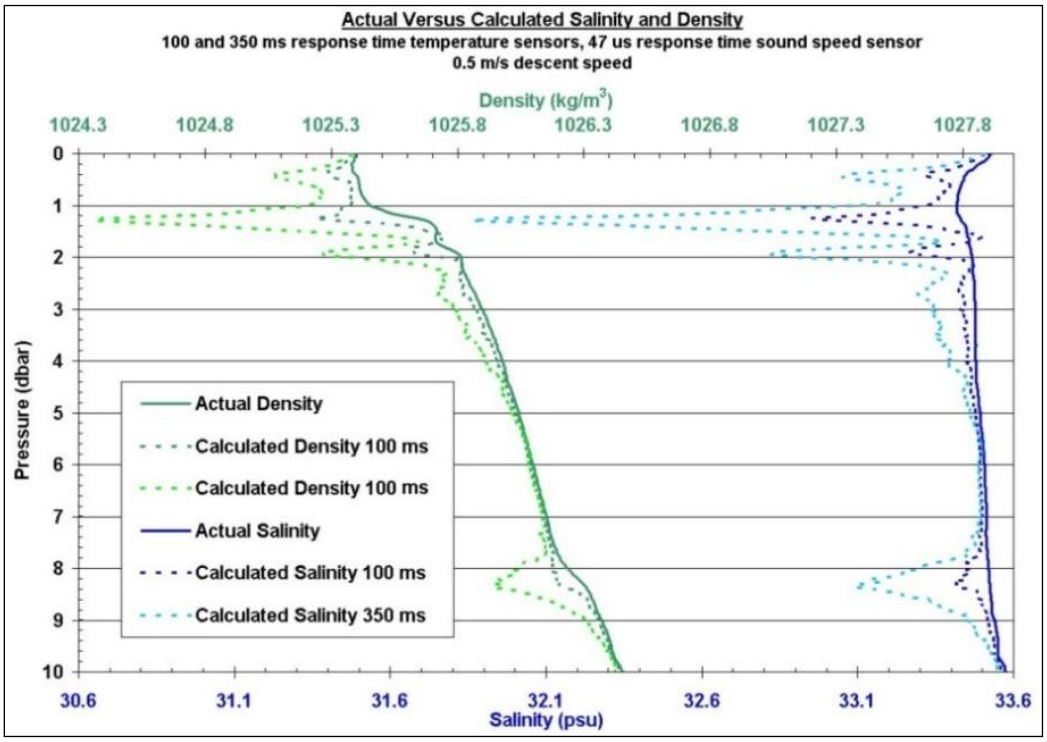


Figure 4.2 *Simulated salinity and density spiking from sound speed profiler data. Data generated by applying sensor response time lags to oceanographic profile data.*

The obvious solution to the salinity spiking problem for both CTDs and SSPs is to develop a temperature sensor with an order of magnitude faster response time. However, this has yet to be accomplished by ocean sensor manufacturers. An alternative is to use the known response time of the temperature sensor to estimate the actual temperature from the sensor rate of change. This was trialed at AML Oceanographic in the 1998 but the method was found to amplify the sensor noise. Given the speed increases for both high resolution digitizers and low power microprocessors; an alternative method may now be possible. By oversampling the sound speed measurement with a sampling period equivalent to, or faster than, the temperature sensor response time; it is possible to artificially slow the sound speed sensor response time to match the temperature sensor response time. Figure 4.3 shows the method applied to a modelled sound speed sensor using a five sample correction. The modified sound speed value

$$c_{\mathrm{modified}} = c_n - (c_n - c_{n-1})e^{-(T_S/\tau)} - (c_{n-1} - c_{n-2})e^{-2(T_S/\tau)} - (c_{n-2} - c_{n-3})e^{-3(T_S/\tau)} - (c_{n-3} - c_{n-4})e^{-4(T_S/\tau)} \quad , \qquad (4.2)$$

where c_{n-1} is the sound speed measurement prior the latest sound speed measurement c_n, T_S is the period of the sound speed measurement, and τ is the temperature sensor response time.

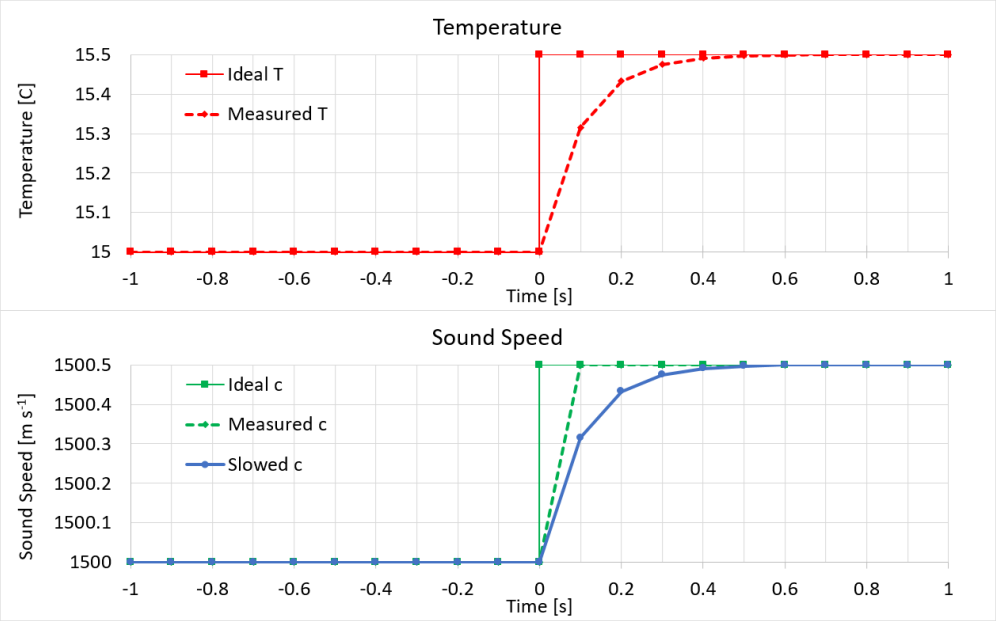


Figure 4.3 *Modified sound speed measurement to approximate temperature sensor response time. The temperature sensor response time for this model is 100 ms.*

This method could also be applied to CTD data to mitigate the salinity spiking error. The author is not aware of any instrumentation in production that uses this technique, so it is an unproven alternative. For static *in situ* measurements salinity spiking is not a problem.

4.1.2 Sound speed accuracy

The second problem with using sound speed sensors, which is applicable to both profiling and static *in situ* measurements, is the accuracy of the sound speed measurements required. As stated in the previous chapter, to match a CTD density error of 0.033 kg m^{-3} the sound speed measurement would have to be made with an accuracy of 0.029 m s^{-1}.

The linearity, precision, and absolute accuracy of sound speed sensors are limited by diffraction, the time measurement method, acoustic path length stability, the buffer layer thickness and material, acoustic mode conversion and calibration reference accuracy. Diffraction errors, which are not specified by the manufacturers, are dependent on the transducer active area, acoustic path length and acoustic frequency. The sound speed sensors currently in production have the necessary resolution and precision (Von Rohden et al., 2015) but the sensor linearity and the calibration standard do not have the required accuracy. Presently the best calibration reference for sound speed sensors is Bilaniuk and Wong's (1993, 1996) pure water equation, with an accuracy of 0.02 m s^{-1}. The best commercial sound speed accuracy specification is 0.017 m s^{-1} RMS at time of calibration. The RSS accuracy for a state of the art sound speed sensor at time of calibration is therefore $(0.02^2 + 0.017^2)^{1/2} = 0.026$ m s^{-1}; however, this accuracy is questionable given the results in section 4.2.1. Table 4.2 shows the combined errors for an SSP instrument at rest in the water. The instrument's measurement accuracy is equivalent to a CTD (table 1.4). With salinity spiking, the error would increase proportional to the rate of temperature change and the error would be higher for the SSP instrument. Unfortunately, there is some concern about the accuracy of the sound speed coefficients in the TEOS-10 equation which were derived, in part, from a sparse Del Grosso and Mader (1972) data set for typical ocean waters. The RMS accuracy of 0.035 m s^{-1} as reported by Marion (TEOS-10 Manual, appendix O) was also based on the sparse data set from Del Grosso and Mader's measurements for typical ocean waters. For waters outside of Del Grosso and Mader's ranges it is possible that the TEOS-10 sound speed errors could be as high as 0.3 m s^{-1} given the differences from Chen, Millero and Li's (1994) equation, the sound speed sensor test results in section 4.2.1, and Von Rohden's (2015) results.

Table 4.2 *Static sound speed profiler density accuracy based on manufacturer's specifications.*

Error source	Measurement error	Error in density [kg m^{-3}]	Error [ppm]
Sound speed measurement	0.17 m s^{-1}	0.013	13
Temperature measurement	0.01°C	0.001	1
Pressure measurement	0.6 dbar	0.028	27
Total sensing error, RSS		0.031	30
Total with TEOS-10 sound speed error of 0.035 m s^{-1}	0.021 kg m^{-3}	0.037	0.036
Total with TEOS-10 sound speed error of 0.3 m s^{-1}	0.189 kg m^{-3}	0.192	188

Note*: Table is based on Valeport listed accuracies at time of calibration. Salinity spiking not included. TEOS-10 errors for sound speed are estimated at 0.035 m s^{-1} for typical oceanographic conditions (Del Grosso, 1974). TEOS-10 errors for sound speed are estimated at 0.3 m s^{-1} for atypical ocean conditions. Density errors were calculated at 35 g kg^{-1}, 4°C and 0 dbar.*

This accuracy problem can be solved in two ways. First, the sound speed sensor linearity and drift rate need to be improved. This is feasible; and can be achieved by a) solving the water absorption problems of the path length rods, b) redesigning the sensor to eliminate diffraction errors and c) improving the time measurement consideration of the buffer layer. Second, the calibration reference accuracy needs to be improved and appropriately ranged. The largest source of error results from extrapolation beyond the calibration range given the non-linearity of the sound speed sensors at present. The calibration reference is therefore as important as the sensor.

There are two possible ways forward for the calibration reference. First is to use the TEOS-10 equation as the calibration reference. This has three advantages: it is quick and inexpensive to implement for the instrument manufacturers, the calibration would make the instrument consistent with TEOS-10 for Del Grosso and Mader's (1972) typical waters, and the calibration could cover the full range of sound speeds. This approach has the disadvantage that it is not

traceable to SI standard units. The second approach is to construct a sound speed standard which is traceable to SI units. This approach is discussed in chapter 5.

4.2 Sound speed sensor – Test results

The testing by Von Rohden et al. (2015) on the AML Oceanographic and Valeport commercial sound speed sensors shows existing commercial sensors have sufficient resolution, 0.001 m s^{-1}, and precision, 0.015 m s^{-1}, to meet the sensor requirements to support TEOS-10. However, the accuracy of the TEOS-10 sound speed coefficients, the pure water calibration method and the validity of the two coefficient calibration equations are questioned by Von Rohden et al.

It is unfortunate that Von Rohden et al. used the wide range, lower accuracy sound speed sensors from AML Oceanographic. For high accuracy sound speed measurements, the sensors to examine are the Valeport 100 mm MiniSVS and the AML standard SVX. Note that table I in Von Rohden et al. contains some minor errors which are corrected in red in table 4.3 in this thesis. The specification errors are small, and will not affect the instrument accuracies, but since the author of this thesis led the design team for three of the five instruments in table 4.3 the specification errors were easily corrected.

Table 4.3 *Sound speed sensor table from Von Rohden et al. (2015).*

Parameter	Units	AML MSV	AML SVX Extended range	VP, VP OEM	AML SVX
Working frequency	MHz	1	4	2.5	4
Acoustic path length	mm	200	66	200	132
Response time	µs	~143	~47	~143	~94
Time resolution	ns	10	~0.02	0.01	0.01
Practical resolution	m s^{-1}	0.2	0.001	0.001	0.001
Range	m s^{-1}	500-2000	1100-2000	1375-1900	1375-1650
Manufacturer's accuracy	m s^{-1}	2.00	0.05	0.017	0.025

Note: The accuracies are the manufacturer's published accuracies (in pure water) not the accuracies as tested by Von Rohden et al. (2015). Corrections are applied in red and the standard AML SVX sensor has been added.

Both the Valeport and AML Oceanographic sensors use a two coefficient calibration equation which cannot account for the diffraction and buffer layer induced variability in the received pressure waves. The Valeport sensors have a larger active acoustic area, lower frequency and longer path length than the AML Oceanographic sensors. The net result is the Valeport sensors are operating in the near field more than the AML SVX sensors. The diffraction effects are examined in section 4.2.2.

The Valeport sensor uses a thin (0.85 mm) polycarbonate buffer layer with a relatively low reflection coefficient with the water (0.25), a sound speed of ~2300 m s^{-1}, a frequency of 2.5 MHz and this design allows 1.8 cycles of uncontaminated signal for the correlation, although it is unclear if the timing algorithm limits the correlation window to the uncontaminated signal. The AML Oceanographic sensor uses a 2 mm thick Macor ceramic buffer layer with a high reflection coefficient (0.80), a sound speed of 5400 m s^{-1}, a 4 MHz signal and this design allows 2.9 cycles of uncontaminated waveform for the correlation, and the correlation window is limited to the uncontaminated portion of the waveform.

The polycarbonate is susceptible to water absorption (0.35% by weight), and to temperature and pressure induced changes in the sound speed and thickness of the layer. The Macor has no water absorption and an order of magnitude better stability with temperature and pressure. These changes will affect the acoustic propagation time and are not accounted for in either manufacturers' calibration coefficients. As a result, the Valeport sensor will be affected by the diffraction and buffer layer contamination to a larger degree than the AML sensor. This error will increase the farther outside the salinity, temperature and pressure calibration range of the instrument the measurement is made.

4.2.1 Commercial sound speed sensor test results

A Valeport 100 mm MiniSVS and an AML Oceanographic SV Xchange were tested in distilled water and sea water. The reference was obtained by applying the TEOS-10 algorithms to water bath data from a Guildline salinometer and a Fluke Blackstack thermometer. A summary of the test results is given in table 4.4. As shown in figure 4.4, both instruments had large errors when extrapolating beyond the calibration range of the instrument. The Valeport sensor was calibrated in distilled water from 2°C (1412 m s^{-1}) to 15°C (1466 m s^{-1}) using Del Grosso's (1974) NRL II equation. The AML sensor was calibrated in distilled water from 2°C (1412 m s^{-1}) to 32°C (1514 m s^{-1}) using Bilaniuk and Wong's (1996) equation. Figure 4.4 used the manufacturer's calibrations.

Table 4.4 *Sound speed sensor test results.*

Parameter	Units	Von Rohden et al.		Dakin		Dakin
		AML SVX	VP MiniSVS	AML SVX	VP MiniSVS	ISDS
Distilled accuracy	m s^{-1}	0.029	0.015	0.010	0.077	0.028
Saline accuracy	m s^{-1}	0.12	0.12	0.093	0.045	0.068
Overall accuracy	m s^{-1}	-	-	0.085	0.104	0.053
Peak error	m s^{-1}	0.3		0.3	0.46	0.122
Precision	m s^{-1}	-	-	0.009	-	0.004

Note: Von Rohden saline accuracies are estimated from figure 7 in Von Rohden et al. (2015). The distilled, saline and overall accuracies were limited to a max temperature of 35°C for consistent comparison of each sensor. The peak error covers the full temperature range tested.

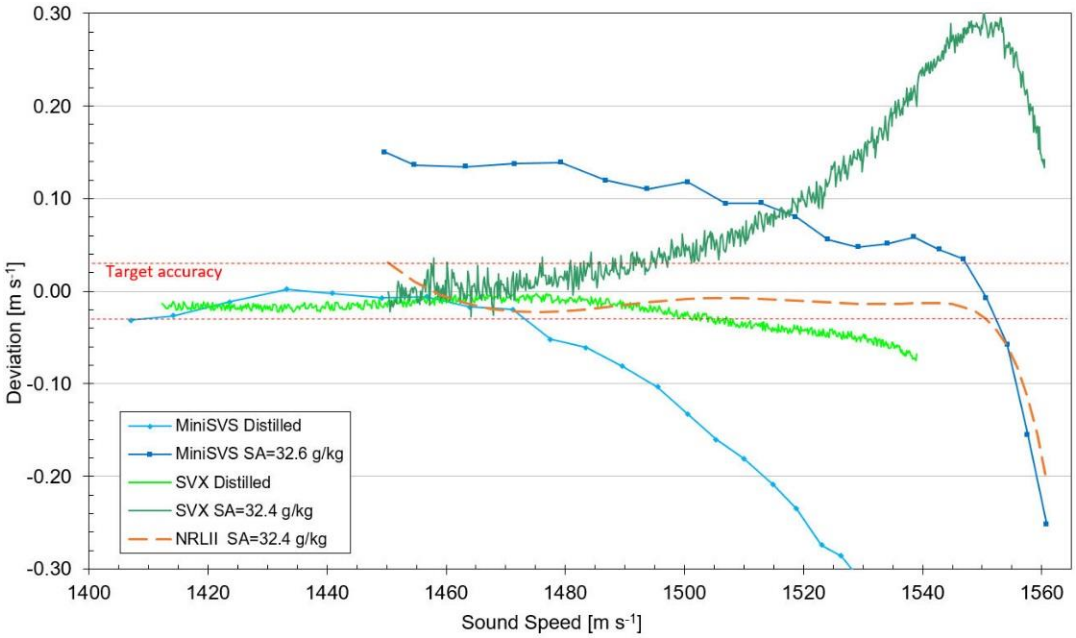


Figure 4.4 *Accuracy of the Valeport 100 mm MiniSVS and AML SV Xchange Deviation with respect to TEOS-10 using the manufacturer's calibration. The orange long dash line shows the deviation of the NRL II equation from TEOS-10.*

In figure 4.4 the downward roll off for both sensors above 1550 m s^{-1} also occurs in the difference between the NRL II equation and TEOS-10 indicating this may be a TEOS-10 induced error which could exceed −0.2 m s^{-1}. If that is the case, the SVX sensor would have a higher error above 1550 m s^{-1} and the MiniSVS would improve.

Focusing on the 1530 m s^{-1} sound speed errors in figure 4.4, there is a deviation between the distilled and salt water sound speed measurements for both instruments, 0.2 m s^{-1} for the SVX and 0.36 m s^{-1} for the MiniSVS. This difference between the salt water and distilled water accuracy is a clue to an error problem. Since the sound speed is the same for the distilled and salt water measurements, the error due to diffraction should be the same for a fixed path length. The error difference must therefore be due to a change in the acoustic path or an inability of the calibration equation to deal with the buffer layer thermal changes.

For the data in figure 4.4 the acoustic path length may have changed in two ways: water absorption in the carbon fiber rods and thermal expansion of the rods. At 1530 m s^{-1} the temperature in distilled water is 40.7°C and the temperature in the 32.6 g kg^{-1} salt water is 24.4°C. The deviation could be explained by a +8.0 ppm C^{-1} CTE for the SVX and a 14.4 ppm C^{-1} CTE for the MiniSVS. These CTEs are possible if the carbon fiber rods have not been wound at the correct angle or the resin formulation has changed. The original design required a 0° or 39° to 45° from longitudinal orientation for the carbon fibers in polyester resin.

An alternative cause for the sound speed difference could have been the buffer layer; however, the required thermal expansion and sound speed changes in the buffer are too large to account for these deviations. The CTE required for the polycarbonate would need to be >2500 ppm C^{-1}, and the >900 ppm C^{-1} for Macor.

A final option is the calibration equation introducing an error in the coefficient which accounts for the buffer layer delay. This could be caused by the diffraction error at the calibration point affecting the estimate of the buffer layer delay.

Water absorption in the carbon fiber rods was tested on a second SVX sensor. Unfortunately, a second MiniSVS was not available, nor was a water bath with temperature rate control. To compensate for the lack of temperature rate control, the bath was programmed for an isothermal period of 40 minutes every 5°C from 30°C to 5°C. This was done to compensate for the slow response time of the temperature sensor (100 ms). The sound speed in distilled water was measured from 32 to 4°C daily. Between measurement runs the distilled bath and sensor were held at 37°C for approximately 15 hours. For each run the bath was taken down to 35°C before the SVX sensor faces were wiped clean of any micro bubbles. The SVX showed a progressive flat offset each day which stabilized at -0.34 m s^{-1} at days 6, 7 and 8. After the sensor stabilized, calibration coefficients were generated referenced to Bilaniuk and Wong's (1996) equation. The SVX output is shown in figure 4.5. The temperature stops are clearly evident in the plots, and should be considered the best estimate of accuracy. Between the temperature stops, while the bath temperature is changing, the lag of the temperature sensor becomes apparent and there is a shift in the error estimate by about -0.03 m s^{-1}. The results after soaking are significantly different from the first SVX sensor tested. The overall linearity has improved though there are still significant errors. It is unfortunate that the MiniSVS was not tested to see if the linearity improved for the MiniSVS as well.

The -0.34 m s^{-1} water absorption shift for the SVX is a significant source of error. Since the SVX uses a titanium reflector plate and a Macor ceramic buffer layer the water absorption is limited to the carbon fiber rods. Since both manufacturers expose the carbon fiber rods directly to the water, it is recommended that they reexamine the path length stability for the sensors. Using Zerodur, with a CTE of 0.01 ppm C^{-1} and zero water absorption, would solve the stability issue.

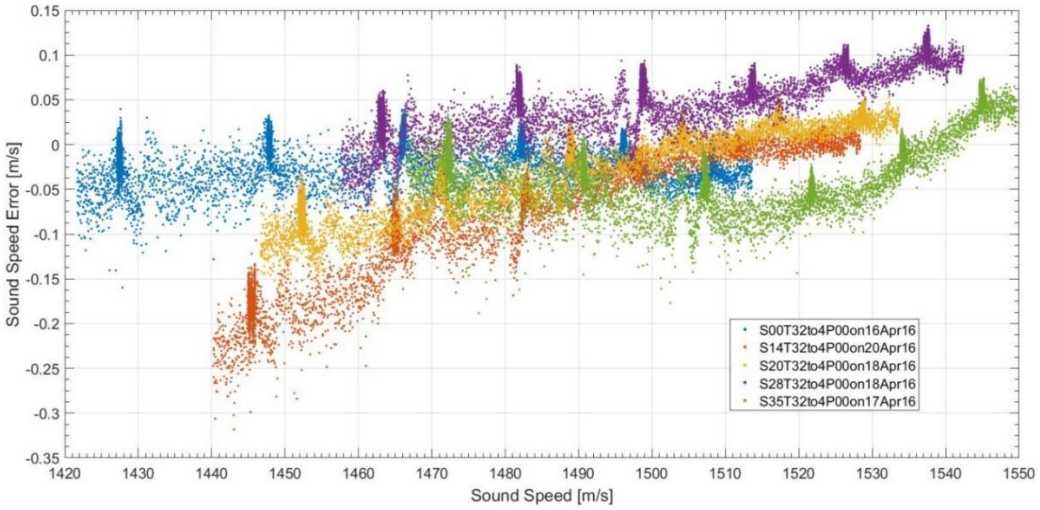


Figure 4.5 *SVX sensor #2 sound speed errors at various salinities. The sensor was recalibrated after an 8-day soak in distilled water. There is a significant improvement at a salinity of 35 g kg^{-1} compared to figure 4.4.*

As an aside: when carbon fiber rods were first introduced to sound speed sensors (on the AML Oceanographic Micro SV sensor developed by the author's R&D team) in 1999, a polyester resin was used with graphite monofilament fibers laid longitudinally. The rods were tubes which free flooded for pressure compensation. The thermal expansion was measured at −0.22 ppm C^{-1} over a range of 2 to 18°C. One of the water absorption tests is shown in figure 4.6.

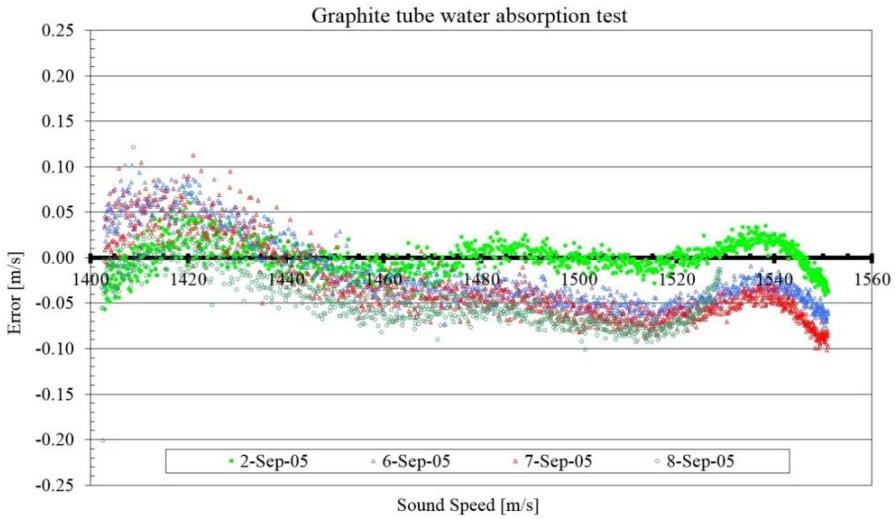


Figure 4.6 *Water absorption test of a carbon-polyester spacer tube. Spacer tube used on an AML oceanographic MicroSV sensor.*

The MiniSVS precision was measured by Von Rohden et al. (2015) and standard deviations of 0.016 and 0.015 were reported for the two instruments. The SVX precision was tested for comparison and the standard deviation was 0.009 m s^{-1} in water with a slow temperature change of −0.019°C per minute. Figure 4.7 shows 2200 measurements over a 2.4°C window and shows the periodic nature of the SVX measurements which contribute to the precision error. When assessing the precision one should consider the precision of the reference temperature sensor used to calculate the TEOS-10 sound speed reference. A MicroT temperature sensor was used for the test. The noise of the MicroT is shown in figure 4.8. The temperature sensor noise has a standard deviation of 0.003°C, a minimum peak of −0.013°C, and a maximum peak of 0.008°C, which are equivalent to a sound speed standard deviation of 0.006 m s^{-1}, and range of −0.027 to 0.017 m s^{-1} at 33°C.

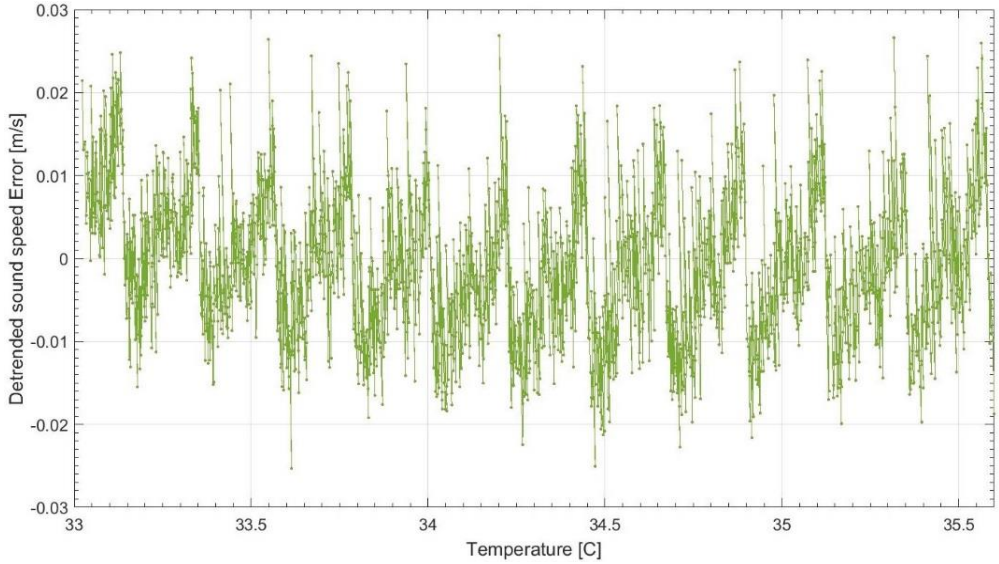


Figure 4.7 *High resolution plot of the detrended sound speed error to show the periodicity of the measurements.*

The period of the ±0.01 m s^{-1} saw tooth pattern in figure 4.7 coincides with the digitization rate, which shows the pattern is due to the interpolation algorithm. The precision could therefore

be improved by switching from the efficient parabolic interpolation to an upsampling interpolation. The parabolic interpolation works most effectively if the correlation coefficient peak is symmetric about the peak, which is not the case for the SVX waveform correlations.

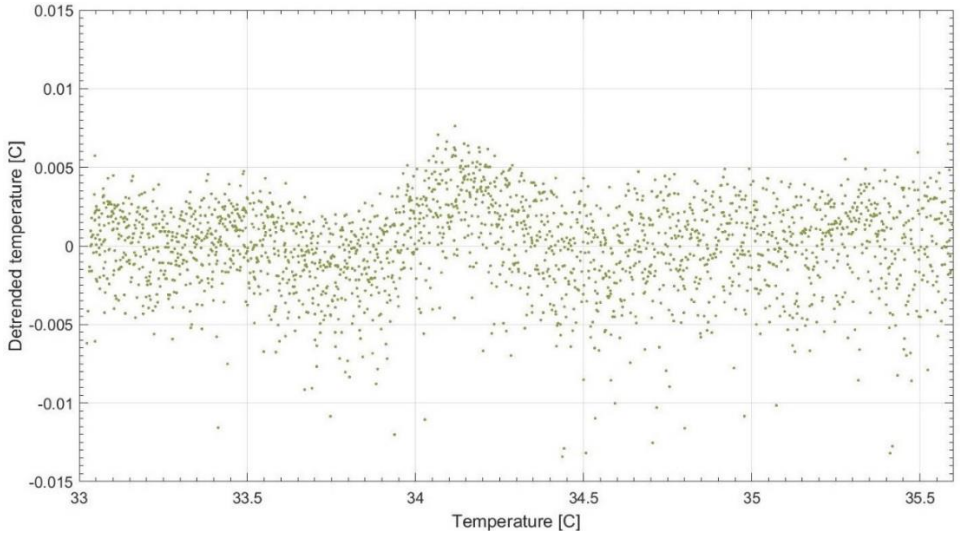


Figure 4.8 *Detrended MicroT sensor temperature data points for the data in figure 4.7. Temperature σ = 0.003°C, which equates to a TEOS-10 sound speed σ = 0.006 m s^{-1}.*

Unfortunately, without a proper sound speed standard, it is impossible to determine if the commercial sensor errors shown in figure 4.4 are due to the sensors or the TEOS-10 equation.

4.2.2 Diffraction modelling

The diffraction for both the Valeport MiniSVS and the AML SVX were modelled using the author's MatLab diffraction model, NearField, previously discussed in section 3.2.1. The model output for the MiniSVS at a sound speed of 1402 m s^{-1} is shown in figure 4.9, the output for a sound speed of 1615 m s^{-1} is shown in figure 4.10. The equivalent plots for the SVX are given in figures 4.11 and 4.12. The MiniSVS differences between figures 4.8 and 4.9 are larger than the SVX differences between figures 4.11 and 4.12; as a result, the diffraction error will have a larger impact on the MiniSVS.

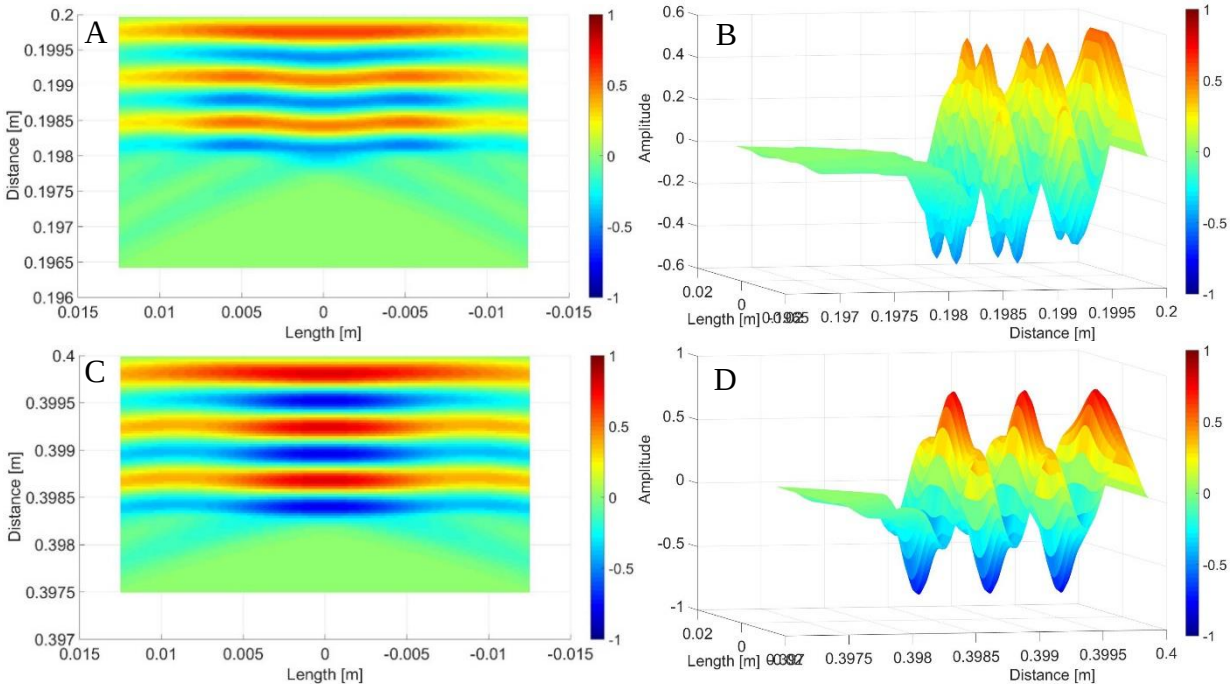


Figure 4.9 *Cross section of the modelled MiniSVS acoustic pressure wave approaching the transducer. The plots illustrate the diffraction effects on the first (A and B) and second (C and D) echoes at a sound speed of 1402 m s^{-1}.*

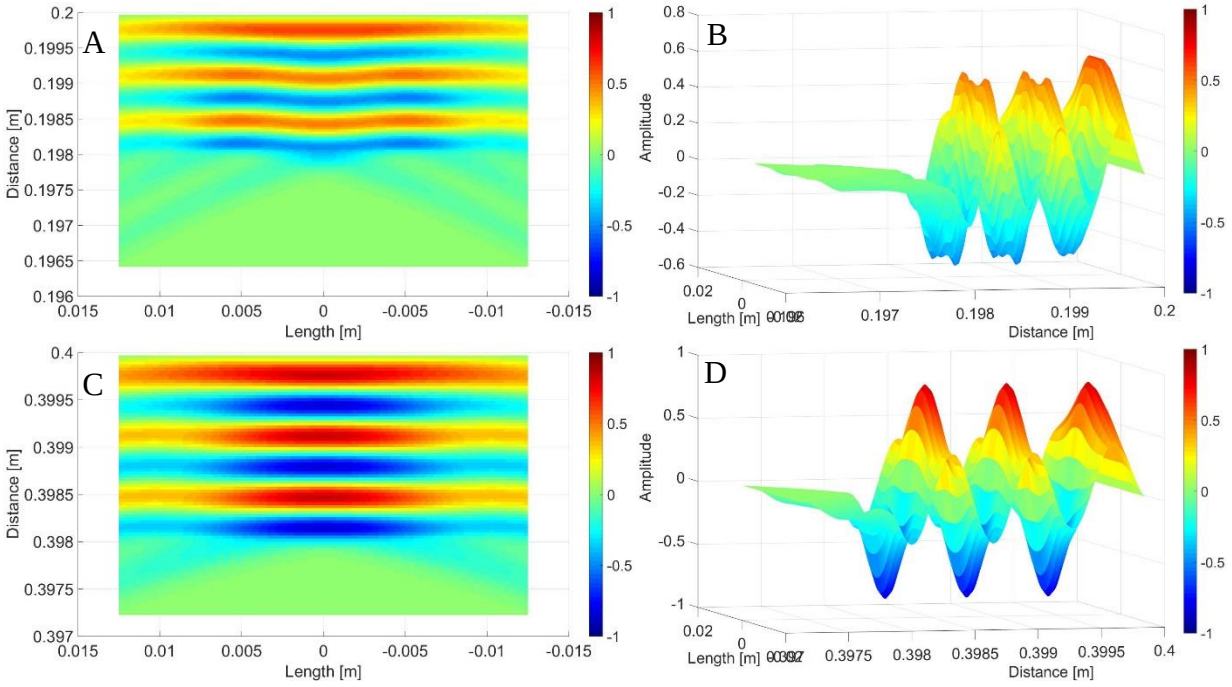


Figure 4.10 *Cross section of the modelled MiniSVS acoustic pressure wave approaching the transducer. The plots illustrate the diffraction effects on the first (A and B) and second (C and D) echoes at a sound speed of 1615 m s^{-1}.*

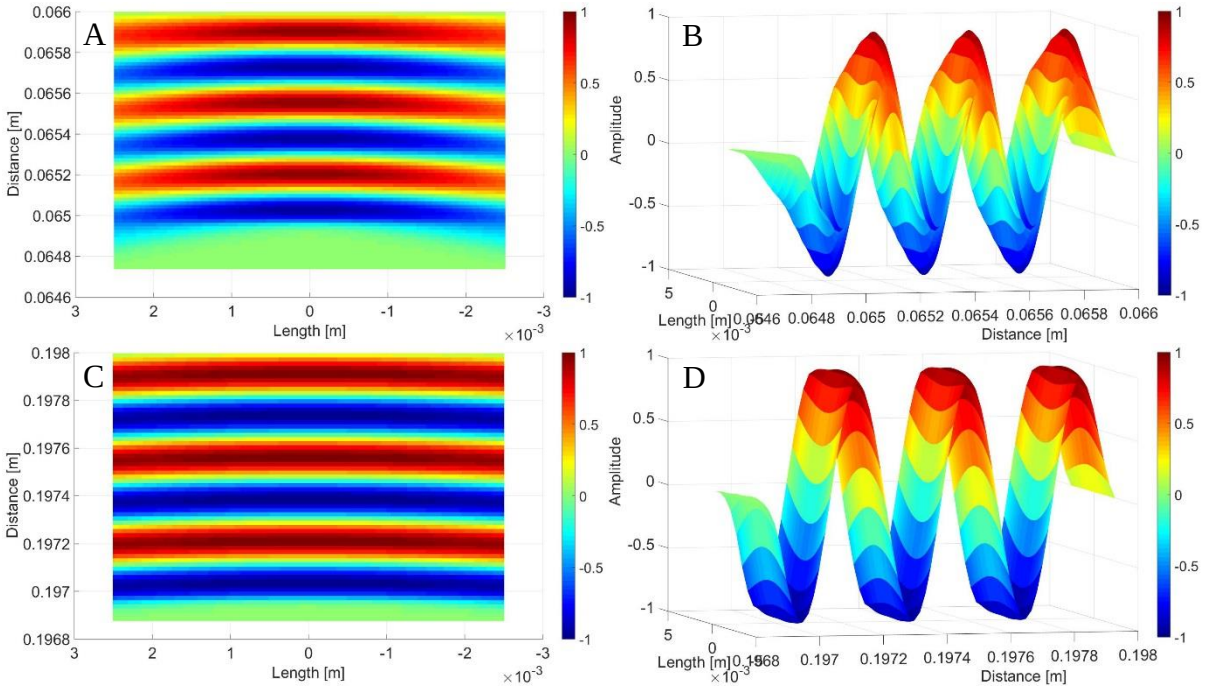


Figure 4.11 *Cross section of the modelled SVX acoustic pressure wave approaching the transducer. The plots illustrate the diffraction effects on the first (A and B) and third (C and D) echoes at a sound speed of 1402 m s^{-1}.*

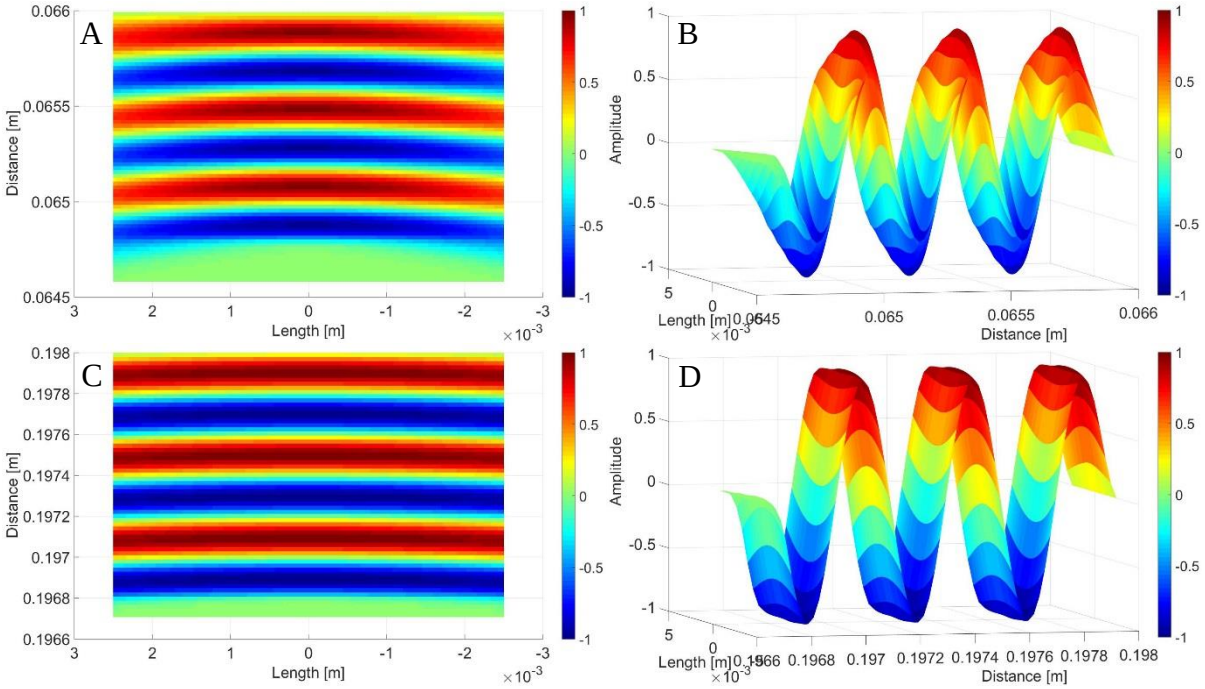
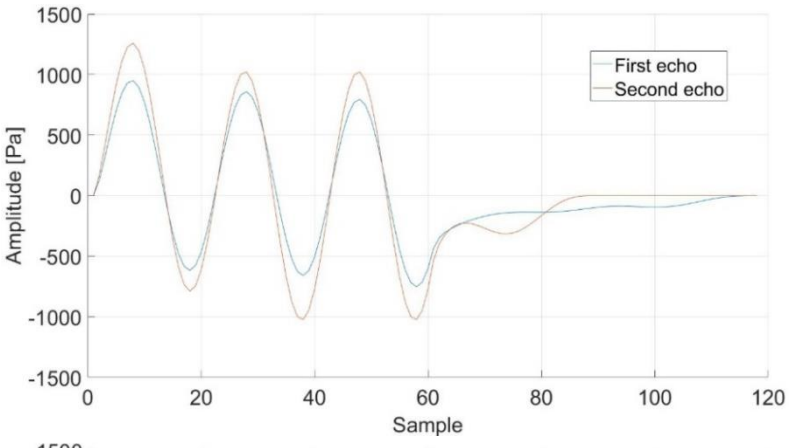


Figure 4.12 *Cross section of the modelled SVX acoustic pressure wave approaching the transducer. The plots illustrate the diffraction effects on the first (A and B) and third (C and D) echoes at a sound speed of 1615 m s^{-1}.*

For figures 4.9 to 4.12 the pressure cross sections shown in plots B and D are polar integrated to produce the total signal for the receiver transducer. The integrated waveforms for the MiniSVS diffraction model are shown in figure 4.13. To highlight the diffraction effects the figure 4.13 amplitudes are scaled to remove spreading loss. The first three cycles of the first and second echoes are cross correlated for each sound speed to determine the magnitude of the diffraction error. For a sound speed of 1402 m s^{-1} the timing error is 1.43 ns generating a sound speed error of 0.014 m s^{-1}. For a sound speed of 1615 m s^{-1} the timing error is 1.62 ns generating a sound speed error of 0.021 m s^{-1}. The diffraction error offset and slope could be mitigated by extending the upper calibration point closer to 1615 m s^{-1}.

A



B

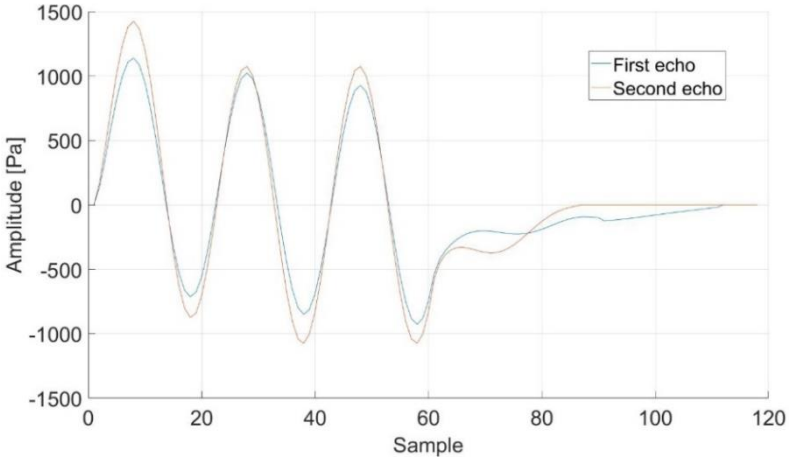


Figure 4.13 *Model polar integrated pressure plots for the MiniSVS diffraction configuration. Amplitude scaled to remove spreading loss. A) Pressure at a sound speed of 1402 m s^{-1}. B) Pressure at a sound speed of 1615 m s^{-1}.*

4.3 Sound speed sensor – Conclusions

The target accuracy for sound speed measurements to enable TEOS-10 is 0.029 m s^{-1}. Present commercial sound speed sensors have excellent resolution (0.001 m s^{-1}), and good precision (0.009 m s^{-1}), but poor accuracy (0.3 m s^{-1}) at specific sound speeds, with respect to enabling TEOS-10. As a result, for static *in situ* measurements, present sound speed sensors cannot meet the accuracy of the CTD based density estimates. Due to the slow temperature sensor response time (100 ms), both CTD and sound speed profilers are subject to salinity spiking errors during profiling operations. However, salinity spiking induced density errors are three times larger for present sound speed profilers when operating at 0.5 m s^{-1} profiling speeds.

Sensor testing and modelling revealed several sources of error for the sound speed instruments. The error sources include: slow temperature sensor response times inducing salinity spiking (±0.5 g kg^{-1} salinity), diffraction induced timing errors (0.014 to 0.021 m s^{-1} sound speed for the MiniSVS), interpolation induced timing errors (±0.01 m s^{-1} sound speed for the SVX), potential buffer layer reflection induced timing errors, water absorption induced path length errors (0.3 m s^{-1}), errors from extrapolating too far beyond the calibration range (0.3 m s^{-1}), the accuracy of the calibration reference (0.02 m s^{-1}), and possible errors in the TEOS-10 sound speed coefficients above 1550 m s^{-1} (which could exceed −0.2 m/s).

Del Grosso and Mader's (1972) sea water sound speed apparatus, decommissioned several decades ago, could be revised and rebuilt to solve the calibration range, calibration accuracy and verify, or correct, the TEOS-10 sound speed coefficients. This is addressed in greater detail in chapter 5 of this thesis. This apparatus would resolve two of the top three static sound speed errors.

The final major uncertainty preventing sound speed from enabling TEOS-10 is water absorption in the spacer rods. This is easily resolved by switching to a low thermal expansion glass ceramic material such as Zerodur.

The diffraction errors can be addressed with smaller diameter transducers, longer path lengths, lower frequencies or using multiple transits through the water sample. The interpolated time errors can be addressed by a higher bandwidth transducer, higher digitizing rates or switching to an upsampling interpolation method. The buffer layer reflection interference issue can be addressed by using thick buffer layers and windowing the cross correlation window to the uncontaminated signal.

The profiling error due to salinity spiking should be addressed by improving the temperature sensor response time to 10 ms or better. However, this is a challenging task. A more practical approach is to oversample the sound speed data and apply a sliding sound speed average to achieve a response time equivalent to the temperature sensor.

All of the listed error mitigation strategies are easily implemented with present technology, with the exception of the sound speed standard. Sound speed could be used to enable the TEOS-10 equation if a sound speed standard is produced and implemented by the sensor manufacturers. This problem is considered in the following chapter.

Chapter 5

Sound speed standard

As discussed in chapter 4, present sound speed sensors have excellent resolution (0.001 m s^{-1}), good precision (0.009 m s^{-1}), but poor accuracy (0.3 m s^{-1}). The primary limitation for the accuracy is the calibration reference. Unfortunately, the manufacturer's sound speed reference is either Del Grosso's (1974) equation or Bilaniuk and Wong's (1993, 1996) pure water equations which have estimated accuracies of 0.02 m s^{-1} RMS. Since the target sound speed sensor accuracy is 0.029 m s^{-1}, it is desirable to have a calibration reference accuracy of 0.005 m s^{-1} or better. The order of magnitude difference in accuracy ensures the *in situ* sensor accuracy is not limited by the calibration reference. The present reference accuracy is compounded by the limited sound speed range of the pure water equations, 1402 to 1530 m s^{-1}. For the sensor to operate accurately in sea water the sensor calibration curve must be extrapolated beyond 1530 m s^{-1} with an associated degradation in sensor accuracy, as shown in figure 4.4. A better solution is to have a sound speed standard which operates over the full range of oceanic sound speeds.

5.1 Description of the prior art

The data set for Del Grosso and Mader's (1972) apparatus, shown in figure 5.1, is the basis for the two salt water sound speed equations used in EOS-80 and the Gibbs equation of TEOS-10. The apparatus, now decommissioned by the Naval Research Labs, has never been replicated or independently verified. However, Dushaw et al. (1993) report the NRL II equation (Del Grosso 1974) based solely on the Del Grosso and Mader (1972) data set matches long range ocean measurements better than modified data sets such as those of Chen and Millero (1977).

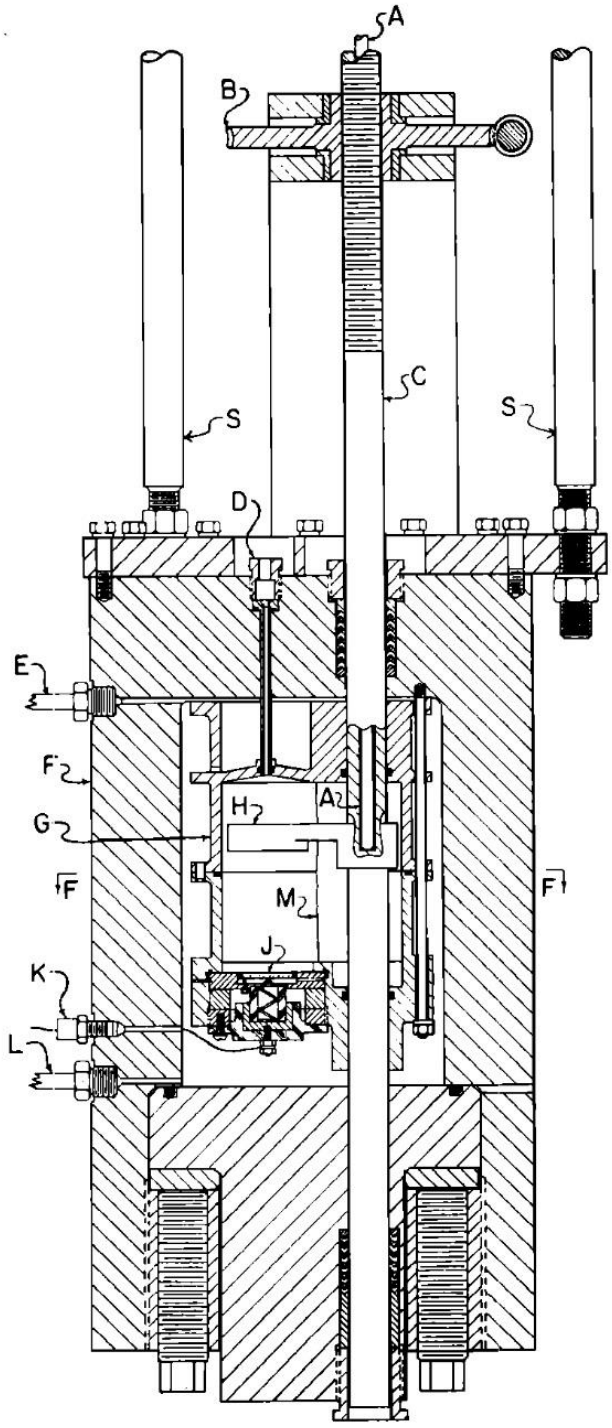


Figure 5.1 *Del Grosso and Mader's pressurized acoustic interferometer schematic. (A) quartz gauge rod (eliminated), (B) drive gear, (C) Reflector shaft, (D) cell air bleed, (E) pressure jacket air bleed, (F) pressure jacket, (G) interferometer cell, (H) reflector, (J) quartz crystal, (K) electrical input; (L) pressure jacket oil inlet, (M) pressure separator, (N) platinum thermometer, (P) test liquid inlet, (Q) quartz thermometer probes, (S) support rods. Taken from Del Grosso and Mader, 1972.*

Del Grosso and Mader's (1972) apparatus, shown in figure 5.1, measured the acoustic path length change required to add a further 300 acoustic wavelengths in the path. A continuous sinusoidal voltage, at 5 MHz, was applied to the piezoelectric element *J*, a continuous acoustic wave was emitted from the element and travelled through the water sample to the reflector plate *H*, where the wave was reflected back to the piezoelectric element *J*. The reflected acoustic wave entering the piezoelectric element affected the impedance of the element, and the impedance varied with the phase difference between the transmitted and reflected acoustic waves. Impedance and phase meters were used to count the acoustic interference cycles as the reflector plate was moved. The impedance was tuned by eye, by slow reflector plate movement, so that the starting impedance matched the final impedance. Wavelength resolution and accuracy were improved by moving the path length through 300 acoustic wavelengths. The laser interferometer fringe count was used to measure the path length displacement. The acoustic frequency was verified with a national time standard pulse. The equation for the Del Grosso and Mader apparatus is $c = \lambda f = (2\delta D / 300) f$, where c is the sound speed in the water, λ is the wavelength in the water, f is the acoustic frequency, and δD is the reflector distance change measured by laser interferometry.

5.2 Sound speed standard proposal

In an attempt to address the present lack of a sound speed reference, a sound speed standard project was initiated by the author while working for AML Oceanographic in 2008. The proposed device was based upon Del Grosso and Mader's (1972) sound speed apparatus so the project risk was low. At the end of the project definition phase the author submitted a project requirements document, cost estimates and concept designs for approval to continue the project. The cost estimates included salaries, a thermally stable and vibration isolated room, fabrication

of a titanium pressure vessel, and traceable references for time, distance, pressure, salinity and temperature measurement, which totaled nearly $0.5 M. This estimate caused the research sponsor, AML Oceanographic, to suspend further efforts on the project since it would be difficult to recover the investment.

In the past 8 years, technology advances have changed the expense requirements for the sound speed standard. Micromachining and lithographic systems have driven advances in nano-positioning devices, increasing the actuator force and lowering the cost of these systems. The proliferation of computer numerically controlled (CNC) milling machines has driven significant improvements in accuracy, stability, size, cost and vibration immunity for laser interferometer distance measurement systems. The vibration immunity alone allows the elimination of the vibration isolated room which was nearly half of the proposal cost. As a result of all the above improvements, the apparatus would be more affordable now. The sound speed standard concept is presented here with the hope that it will stimulate efforts to address the deficiency.

5.2.1 Sound speed standard- Design

The proposed standard is based upon Del Grosso and Mader's (1972) sound speed apparatus. While Del Grosso and Mader measured the differential path length and the number of acoustic wavelengths, the new method measures the differential time and differential path length. The basic premise is the sound speed $c = \delta D / \delta t$, where δD is the change in distance travelled by an acoustic wave, and δt is the associated change in acoustic travel time.

The concept shown in figure 5.2 is based on Del Grosso and Mader's (1972) apparatus, described in section 5.1, with modifications to allow: the introduction of a sound speed sensor to

be calibrated, improved automation, reduced diffraction errors, reduced multipath errors, and improved path length and timing accuracy via new technologies and new signal timing techniques.

As with Del Grosso and Mader's (1972) apparatus, the new design utilizes a variable path length, since it is easier to obtain relative length than absolute length. The operational concept is shown in figure 5.2. The technique relies on high speed digitizing of the reflected acoustic waveforms at two path lengths; the modelled waveforms are shown in figure 5.3. The two waveforms must be captured in rapid succession since the methodology assumes that the sound speed will be the same for both samples. The validity of this assumption will be dependent on the rate of change of the pressure and temperature within the sample chamber. The temperature rate of change is therefore a critical design requirement. This is most easily achieved by submerging the standard in a temperature controlled bath.

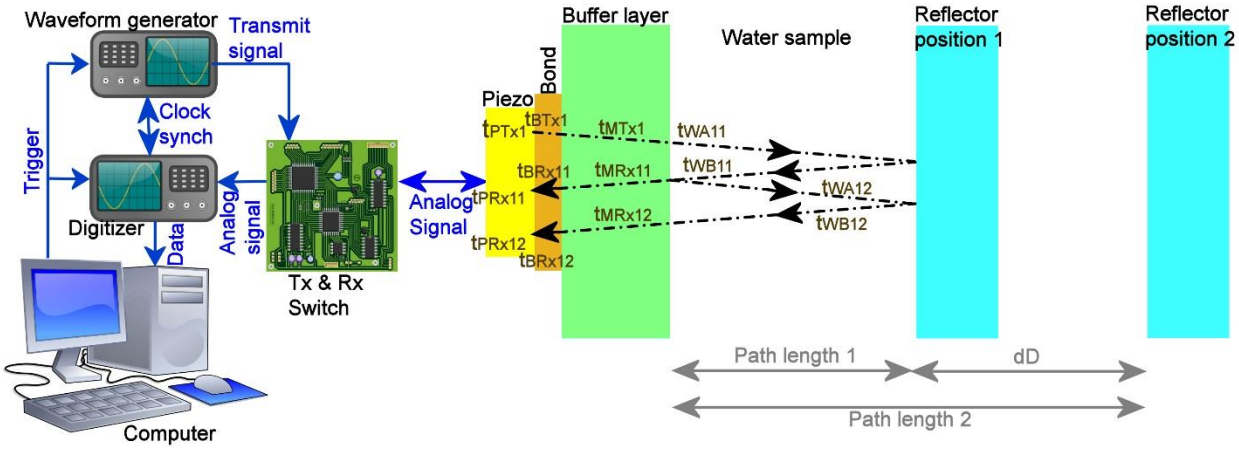
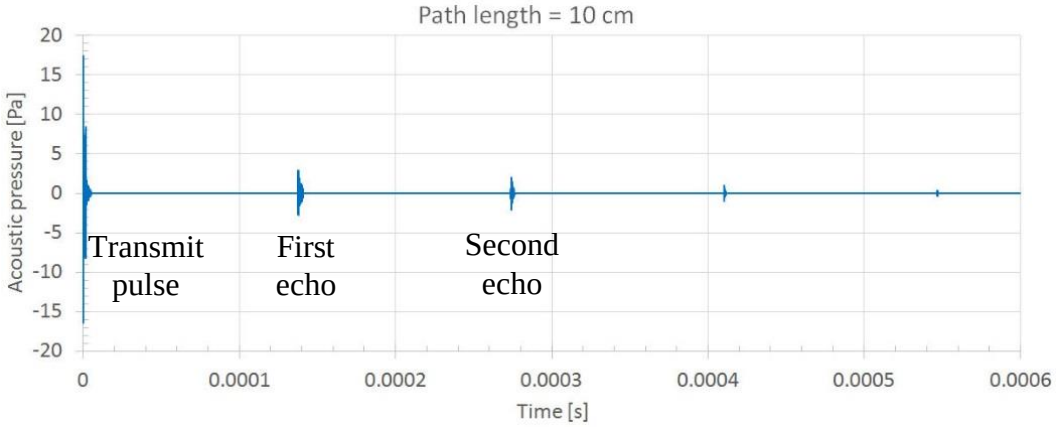


Figure 5.2 *Operational concept for a sound speed standard. The acoustic path from the piezoelectric transducer to the reflector and back to the piezoelectric transducer with four transits through the water is represented by the dashed lines.*

A



B

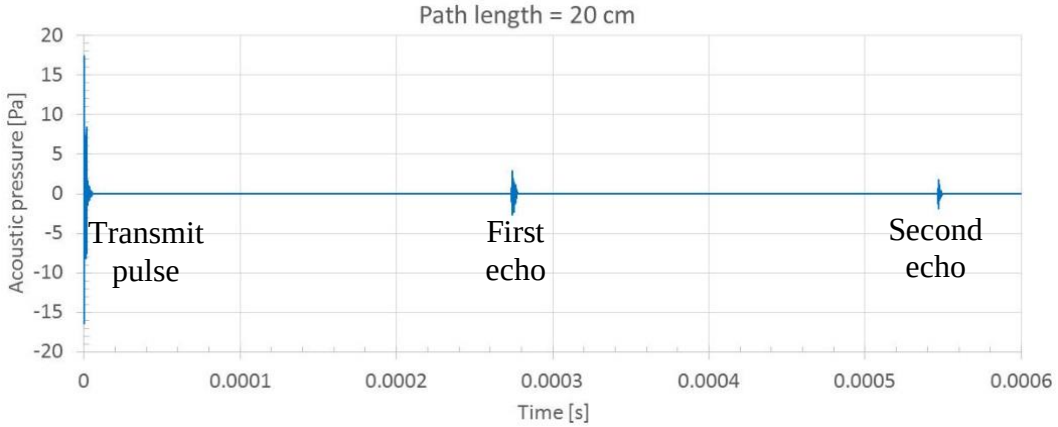


Figure 5.3 Modelled sound speed standard sensor response.
A) for path length 1; B) for path length 2.

Following the sequence shown in figure 5.2, the timing and digitizing operation begins with the computer triggering the waveform generator and the digitizer to start a sample. The digitizer begins to capture the signal at 6 GS/s, or higher, which is achievable with commercial off the shelf technology at 12 bit amplitude resolution (GaGe 12-Bit PCIe Gen3 EON Express). The waveform generator generates an excitation pulse for the piezoelectric transducer. The transducer changes shape, with a piezo transmit time delay t_{PTx1}, creating an acoustic pulse which propagates though the bond line with time t_{BTx1}, buffer layer t_{MTx1}, and into the water t_{WA11}. The acoustic pulse is reflected back through the water t_{WB11}, to the buffer layer. Some energy is transmitted through the buffer layer t_{MRx11}, and bond line t_{BRx11}, to produce a voltage across the

piezoelectric transducer with delay t_{PRx11} and this voltage represents the first echo. Some of the energy is reflected from the buffer layer back into the water following a similar path with propagation times of t_{WA12}, t_{WB12}, t_{MRx12}, t_{BRx12}, and t_{PRx12}. This energy is used to produce the second echo voltage. During the entire sequence the analog voltage from the piezoelectric element is amplified and digitized producing the waveform shown in figure 5.3A. The path length is then changed to position 2 and the process is repeated to generate the waveform shown in figure 5.3B.

The second echo is used to decrease diffraction and increase the differential path length while keeping the instrument dimensions as small as possible. Commercial off-the-shelf laser interferometers (SIOS Model SP 2000) have sufficient range of measurement (10 to 20 cm) and can provide 20 pm resolution for the differential path length. This resolution is much better than required. With a reflector travel of 10 cm and a sound speed of 1615 m s^{-1} a 1 nm path length resolution is equivalent to a 0.00002 m s^{-1} sound speed resolution.

The sound speed resolution is then limited by the timing resolution. With a 6 GHz digitizer sample rate the sound speed resolution is 0.0011 m s^{-1}. This could be improved by interpolation; however, it is limited by the phase to which the acoustic cycle can be resolved. With the ISDS, the interpolation was successful to 1/250 of the digitizing period or 1/8000 of the acoustic period. It is possible the latter is the limiting factor since the peak of the cross correlation is relatively flat at that resolution. Thus the data may become noisy with more than a factor of two interpolation of the 6 GHz digitizer sampling. A factor of two improvement would provide a sound speed resolution ten times better than the desired accuracy of 0.005 m s^{-1}, which is acceptable.

To understand how the method reduces timing errors one must examine the various factors contributing to the complete time solution. For the first capture, at path length 1, the total time for the second echo is

$$t_{12} = t_{ETx1} + t_{DTx1} + t_{PTx1} + t_{BTx1} + t_{MTx1} + t_{WA11} + t_{FA11} + t_{WB11} + t_{FB11} + t_{WA12} + t_{FA12} + t_{WB12} + t_{FB12} + t_{MRx12} + t_{BRx12} + t_{PRx12} + t_{ERx12} + t_{DRX12} + t_{Diff12} + t_{ED12},$$

where:
t_{12} = Transmit to receive time for path length 1 echo 2,
t_{ETx1} = Electronics transmit delay for path length 1,
t_{DTx1} = Digitizer triggering delay for path length 1,
t_{PTx1} = Piezo transmit delay for path length 1,
t_{BTx1} = Bonding layer transmit delay for path length 1,
t_{MTx1} = Buffer layer transmit delay for path length 1,
t_{WA11} = Water transit time piezo to reflector path length 1 echo 1,
t_{WA12} = Water transit time reflector to piezo path length 1 echo 2,
t_{WB11} = Water transit time piezo to reflector path length 1 echo 1,
t_{WB12} = Water transit time reflector to piezo path length 1 echo 2,
t_{FA11} = Flow noise time piezo to reflector path length 1 echo 1,
t_{FA12} = Flow noise transit time reflector to piezo path length 1 echo 2,
t_{FB11} = Flow noise transit time piezo to reflector path length 1 echo 1,
t_{FB12} = Flow noise transit time reflector to piezo path length 1 echo 2,
t_{MRx12} = Buffer layer receive delay for path length 1 echo 2,
t_{BRx12} = Bonding layer receive delay for path length 1 echo 2,
t_{PRx12} = Piezo receive delay for path length 1 echo 2,
t_{ERx12} = Electronics receive delay for path length 1 echo 2,
t_{DRx12} = Digitizer receive delay for path length 1 echo 2, and
t_{Diff12} = Diffraction delay for path length 1 echo 2.

Assuming constant sound speed in the water, then $t_{WA11} = t_{WB11} = t_{WA12} = t_{WB12}$, and the total time the wave spends in the water is $t_{W12} = t_{WA11} + t_{WB11} + t_{WA12} + t_{WB12}$. Since we are using a folded path and a fast propagation time in comparison to the flow rate, the time shift due to flow is equal and opposite in each direction, $t_{FA11} = -t_{FB11} = t_{FA21} = -t_{FB21}$. Then

$$t_{12} = t_{ETx1} + t_{DTx1} + t_{PTx1} + t_{BTx1} + t_{MTx1} + t_{W12} + t_{MRx12} + t_{BRx12} + t_{PRx12} + t_{ERx12} + t_{DRX12} + t_{Diff12}.$$

Similarly, for the second path length time,

$$t_{22} = t_{ETx2} + t_{DTx2} + t_{PTx2} + t_{BTx2} + t_{MTx2} + t_{W22} + t_{MRx22} + t_{BRx22} + t_{PRx22} + t_{ERx22} + t_{DRX22} + t_{Diff22}.$$

Assuming constant temperature and pressure in the bath and constant temperature in the electronics for the path length 1 and 2 measurements the following equalities can be applied: $t_{ETx1} = t_{ETx2}$, $t_{DTx1} = t_{DTx2}$, $t_{PTx1} = t_{PTx2}$, $t_{BTx1} = t_{BTx2}$, $t_{MTx1} = t_{MTx2}$, $t_{MRx1} = t_{MRx2}$, $t_{BRx1} = t_{BRx2}$, $t_{PRx1} = t_{PRx2}$, $t_{ERx1} = t_{ERx2}$, and $t_{DRx1} = t_{DRx2}$,

The differential time between the path length 2 second echo and the path length 1 second echo is $t_{(2-1)2} = t_{22} - t_{12} = t_{W22} - t_{W12} + t_{Cor(2-1)2}$, where $t_{Cor\ (2\text{-}1)\ 2}$ is the time error introduced by correlating the two second echoes. The correlation error is a result of correlating two signals which are not exactly the same. The signal shape differences are due to diffraction, spreading loss and water absorption effects at the two path lengths.

The water sound speed $c = \dfrac{4dD}{\left(t_{W22} - t_{W12} + t_{Cor(2-1)2}\right)}$, where dD is the differential distance between path length 2 and path length 1 as measured by the laser interferometer.

For use as a primary calibration standard the apparatus would require the calibration standard and sensor under test to measure the same liquid at the same conditions, i.e. simultaneously in the same water. To allow calibration points at high sound speed values, >1530 m s^{-1}, it will be necessary to vary the temperature, pressure and/or liquid within the calibration standard. The proposed measurement system is enclosed in a temperature controlled pressure vessel. Since the pressure vessel will be exposed to salt water for prolonged periods, stainless steel should not be used since it suffers from crevice corrosion in salt water. The pressure vessel is therefore designed using titanium.

To reduce pressure and thermal expansion errors the laser must be reflected off a mirror coplanar with the reflector plate, as Del Grosso and Mader (1972) discovered.

The proposed plumbing and mechanicals system concept for the sound speed standard is shown in figure 5.4.

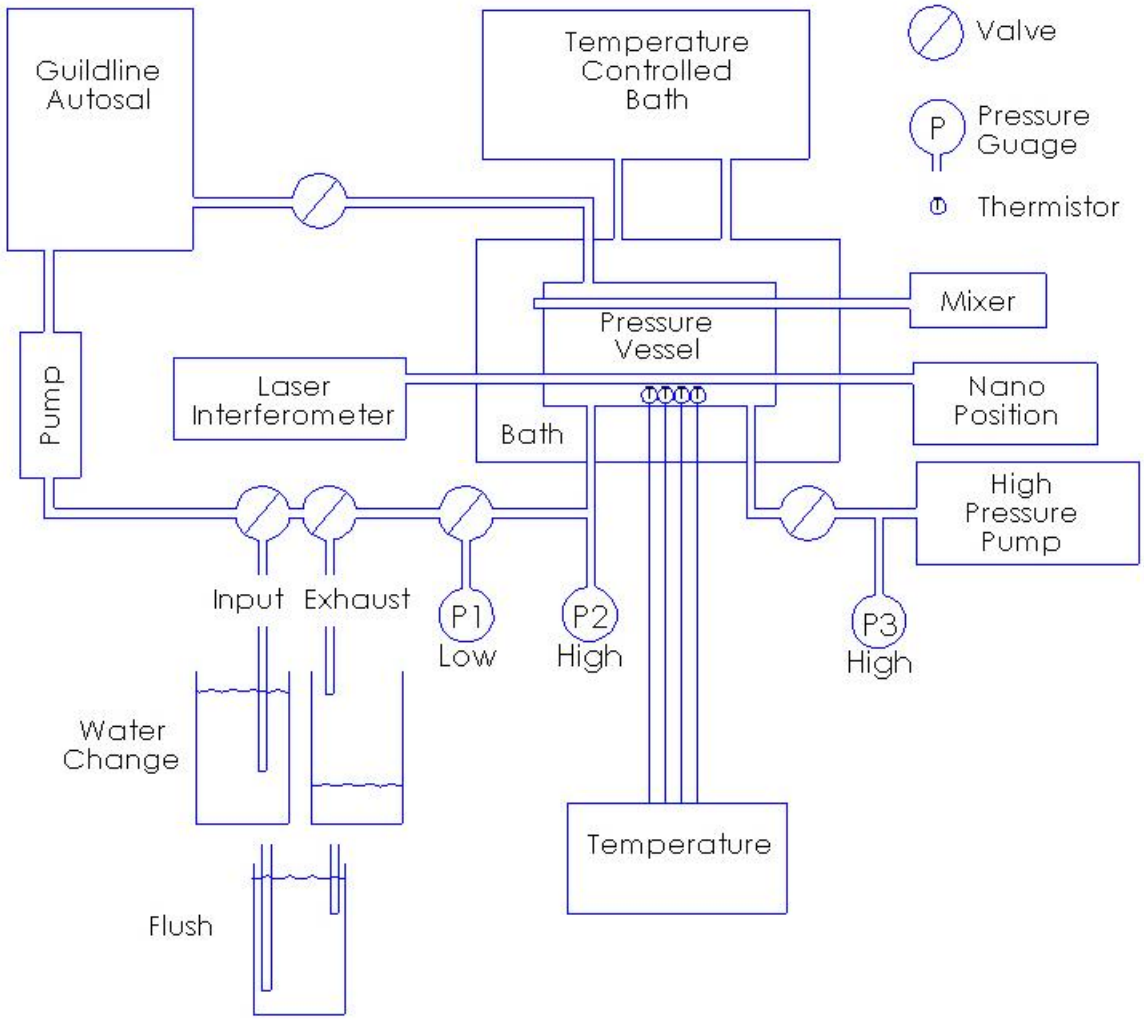
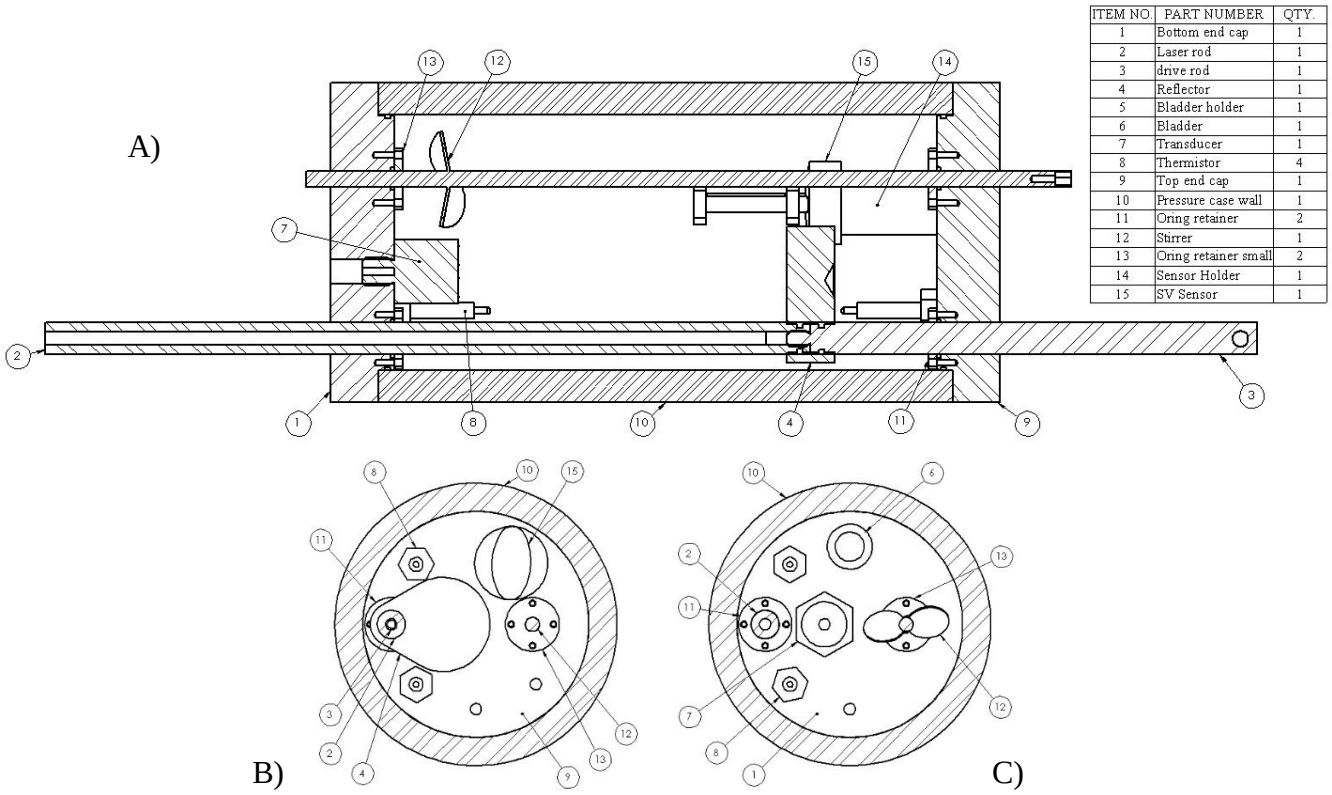


Figure 5.4 *System concept for the sound speed standard.*

A nano-positioning actuator is used to slide the reflector plate along the acoustic axis within the pressure chamber. As with the Del Grosso and Mader (1972) apparatus the reflector plate drive rod passes through both ends of the pressure case so that the actuator force requirement is

limited to overcoming the piston seal o-ring friction on the shaft. A commercial laser interferometer is proposed to automatically measure the displacement. A mixer is used inside the pressure chamber to keep the sample from stratifying during the measurements. Inlet and outlet ports on the pressure chamber allow the sample fluid to be drained and replenished as required. The ports also allow the sample to be circulated through a Guildline Autosal for accurate conductivity measurements of the sample fluid. After the sample has been circulated and measured the Autosal is isolated and pressure can be applied using the deadweight tester (piston). The pressure chamber is brought to temperature using a water jacket plumbed to a temperature controlled bath. Two digital pressure sensors are used to improve pressure accuracy, a low pressure sensor 0 to 40 bar, and a high pressure sensor 0 to 1000 bar. A third analog pressure sensor is used at the deadweight tester for safety during pressure changes. Four temperature sensors are mounted inside the sample chamber to ensure homogeneity and stability of the sample prior to measuring.

The initial mechanical design of the test chamber, shown in figure 5.5, utilized SolidWorks as the design tool. Parts, equipment and services such as laser interferometers, time standards, vibration isolation tables, nano-position drivers, gigahertz digitizers and the environmentally controlled room were sourced to ensure all specifications could be achieved.



ITEM NO.	PART NUMBER	QTY.
1	Bottom end cap	1
2	Laser rod	1
3	drive rod	1
4	Reflector	1
5	Bladder holder	1
6	Bladder	1
7	Transducer	1
8	Thermistor	4
9	Top end cap	1
10	Pressure case wall	1
11	Oring retainer	2
12	Stirrer	1
13	Oring retainer small	2
14	Sensor Holder	1
15	SV Sensor	1

Figure 5.5 *Mechanical design of the titanium pressure chamber.*
A) Cross sectional view of the pressure chamber along the long axis. B) Cross sectional view from the centre looking towards the top end cap. C) Cross sectional view from the centre looking towards the bottom end cap. The plastic sleeve for the temperature controlled water jacket is not shown.

5.3 Sound speed standard- Summary

Lack of a sound speed standard for calibrating sound speed sensors is the primary limitation for using sound speed sensors for TEOS-10 applications. For financial reasons the sound speed standard research carried out by the author in 2008 was stopped before any major advances could be achieved. However, the theory and preliminary design work indicate that a sound speed accuracy of 0.005 m s^{-1} may be achievable over the full oceanographic sound speed range. The technology advances and associated cost savings described in section 5.2 may now make the sound speed standard cost effective.

Chapter 6

Summary and Discussion

The lack of an *in situ* sea water sensor to measure salinity or density independent of the seawater Reference Composition assumption has been a known, yet unresolved, problem for the ocean science community since the introduction of the practical salinity scale (Lewis and Perkin, 1978). The new TEOS-10 equation provides an opportunity to improve the accuracy of *in situ* measurements of salinity and density by removing the assumption of constant salt ratios in sea water. The threshold for switching to a new sensor technique is 0.033 kg m^{-3} for density, or equivalently 0.025 g kg^{-1} for absolute salinity. This thesis explores several possible methods to improve *in situ* measurements for salinity and density. The challenges faced in this thesis reflect the difficulties of the *in situ* measurement problem for the oceanographic community.

The first research theme, phased conductivity, was an attempt to verify the Reference Composition of sea water. This was considered to be an interim but important solution. If the Reference Composition could be verified *in situ*, then water samples would not need to be collected with the CTD measurements thus saving time and funds. Gurriana et al. (2005) reported conductivity phase shifts with various types of saline solutions. This thesis proved there was no phase shift attributable to the bulk properties of saline solutions. Gurriana et al.'s phase shifts were likely due to boundary layers on the electrodes of the conductivity sensor which would render the technique unsuitable for *in situ* sea water use.

The majority of the effort in this thesis was the theoretical development, prototyping and testing of an *in situ* density sensor based on acoustic reflection coefficients. The theory showed great promise but the prototypes demonstrated that the critical measurement interface between

the water sample and the reference material was overly susceptible to microscopic contamination, micro bubbles, in particular. While the technique could be improved, the sensitivity to boundary contaminants does not appear to allow the method to be implementable *in situ*. The research incidentally showed that even a simple sound speed sensor could measure sound speed very accurately with respect to TEOS-10 over a wide range of salinities, when the sensor was constructed with the design criteria developed for the *in situ* density sensor. The single calibration point sensor yielded a precision of 0.004 m s^{-1} and an overall accuracy of 0.053 m s^{-1}. The sound speed derived density was also very accurate at 0.046 kg m^{-3} over a wide range of saline waters. The overall results exceeded the performance of the commercial sound speed sensors tested in chapter 4, including those sensors developed previously by the author for commercial use.

Tests performed on commercial sound speed sensors identified several sources of measurement error, including: water absorption in the carbon fiber rods, diffraction, overly narrow calibration range, insufficient accuracy of the calibration reference, time interpolation, and salinity spiking. Mitigation strategies for each of the error sources are presented in chapter 4. If these strategies are successfully implemented in commercial sensors, then sound speed sensors could exceed the threshold sound speed accuracy of 0.029 m s^{-1} needed to implement TEOS-10 *in situ* measurements.

Testing of sound speed sensors also supported Von Rohden et al.'s (2015) reporting of possible errors in the TEOS-10 sound speed coefficients. The TEOS-10 sound speed rolls off high above 1550 m s^{-1} compared to both commercial sensors and Del Grosso's (1974) NRL II equation.

Development of a new oceanographic sound speed standard would verify or correct the TEOS-10 sound speed coefficients, provide a full range and accurate calibration reference for the *in situ* sound speed sensors, and verify the data of Del Grosso and Mader (1972). A design concept has been outlined for such a device in chapter 5.

In summary the key results of the research, presented in order of importance to the oceanographic community, are:

1. the TEOS-10 sound speed above 1530 m s^{-1} diverges for the instrument test results of this dissertation, the test results of Von Rohden et al. (2015) and Del Grosso's NRL II equation (1974); indicating there is a potential error in the TEOS-10 sound speed coefficients,
2. a sound speed standard is required to either verify, or correct, the TEOS-10 coefficients and to provide an accurate calibration reference for *in situ* sound speed sensors,
3. high accuracy sound speed sensors are close to providing an *in situ* TEOS-10 measurement solution and there are specific technical issues which can be addressed to make the method practical,
4. temperature sensor response time is the limiting factor for water column profiles for both CTDs and sound speed profilers,
5. oversampling, combined with a sliding window modification of the conductivity or sound speed sensor data to match the temperature sensor response time could resolve the salinity spiking problem,

6. sensing methods relying on a boundary layer have higher measurement uncertainties than bulk measurement sensing methods, and
7. conductivity phase shifts do not occur in seawater.

Although addressing the TEOS-10 requirements was found to be a challenging problem and the approaches considered here were either not successful (phased conductivity measurements and the reflection-coefficient density sensor) or promising but unproven (sound speed measurements), the oceanographic justification for TEOS-10 is strong, and efforts in the oceanographic research community should continue. The work carried out in this thesis should represent a valuable contribution to this overall effort.

Bibliography

Balanis, C.A., (2005) Antenna *Theory: Analysis and Design*, 3rd ed. Wiley Interscience

Bilaniuk, N., and Wong G.S.K. (1993) *Speed of sound in pure water as a function of temperature*, J. Acoust. Soc. Am. 93(3) pp 1609-1612.

Bilaniuk, N., and Wong G.S.K. (1996) *Erratum: Speed of sound in pure water as a function of temperature*. J. Acoust. Soc. Am. 99(5) p 3257.

Bjørndal, E., Frøysa, K.E., and Engeseth, S.A. (2008) *A novel approach to acoustic liquid density measurements using a buffer rod based measuring cell*, IEEE Trans. Ultrason., Ferroelect., Freq. Contr. 55(8) pp. 1794–1808.

Caruthers, J.W., (2011) *On Rayleigh and Mie scattering*, Proceedings of Meetings on Acoustics, Vol. 14, 070001

Chen, C-T., Millero, F.J. (1977) *Speed of sound in seawater at high pressures*, J. Acoust. Soc. Am. 62 (5) pp 1129-1135.

Dakin, D.T., (1999) *Applied Microsystem's Equation for Salinity Calculation from SV,T&P*, Application notes, AML Oceanographic.

Del Grosso, V. A. (1974) *New equation for the speed of sound in natural waters (with comparisons to other equations)*. J. Acoust. Soc. Am. 56 (4) pp 1084-1091.

Del Grosso, V. A., and Mader, C. W. (1972) *Speed of sound in sea water samples*. J. Acoust. Soc. Am. 52 (3) pp 961-974.

DiFoggio, R. (2006) *Method and apparatus for an acoustic pulse decay density determination*, US Patent No. 7,024,917.

Dushaw B.D., Peter F., Worchester, Cornuelle B.D., Howe B.M. (1993), *On equations for the speed of sound in seawater*, J. Acoust. Soc. Am. 93 (1), pp 255-275.

Eaton, G., D.T. Dakin (1996) *Miniature time of flight sound velocimeter offers increased accuracy over Sing-around technology and CTD instrumentation.* Oceanology International proceedings 1996.

Fofonoff, N.P., and Millard, R.C. (1983) *Algorithms for computation of fundamental properties of seawater*. UNESCO technical papers in marine science 44, 53pp.

Francois, R.E., and Garrison, G.R. (1982) *Sound absorption based on ocean measurements. Part II: Boric acid contribution and equation for total absorption*. J. Acoust. Soc. Am. 72, 1879

Gurriana L.M., Dias Pereira1 J.M., Ramos H.G. (2005) *Development and Characterization of a pH and Conductivity Measurement System for Water Quality Assessment*, 5th conference on Telecomunications (Conftele'05), Tomar

Haberman, M.R., Guild, M.D. (2016) *Acoustic metamaterials*, Physics Today June 2016 edition, pp 42-48.

Hamon, (1955) *A temperature-salinity-depth recorder*, Journal du Conseil International pour l'Exploration de la Mer, letter to the editor, January 1955.

Harvey, A. H., Kaplan, S. G., Burnett, J. H. (2005) *Effect of Dissolved Air on the Density and Refractive Index of Water*, International Journal of Thermophysics, Vol. 26, No. 5.

Fresnel, A., (1816) Ann Chim et Phys, (2), 1, Oeuvres, Vol. 1, 89, 129

IOC, SCOR and IAPSO (2010) The international thermodynamic equation of seawater – 2010: Calculation and use of thermodynamic properties. Intergovernmental Oceanographic Commission, Manuals and Guides No. 56, UNESCO (English), 196 pp.

Lewis and Perkin, (1978) *The practical salinity scale 1978: conversion of existing data*, Deep-Sea Research, Vol. 28A, No. 4, pp 307-328, 1981.

Liebermann, L.N., (1949) *Sound propagation in chemically active media*, Phys. Rev., vol 76, p 1520.

Liptak, B.G. (2005) *Instrument Engineers' Handbook, Process Control and Optimization*, CRC Press, Fourth Edition, vol 2, pp 99-101.

Mair, H.D., Bresse, L.F., Hutchins, D.A., (1987) A study of diffraction effects of planar transducers using a numerical expression for edge waves, IEEE Ultrasonics symposium, Denver Colorado, USA, pp 771-774.

McDougall, T.J., Feistel, R., Millero, F.J., Jackett, D.R., Wright, D.G., King, B.A., Marion, G.M., Chen, C-T. A., Spitzer, P., (2009) *Calculation of the Thermodynamic Properties of Seawater*, Global Ship-based Repeat Hydrography Manual, IOCCP Report No. 14, ICPO Publication, series no. 134.

Millero, F.J., Feistel, R., Wright, D.G., McDougall, T.J., (2008) *The composition of Standard Seawater and the definition of the Reference-Composition Salinity Scale, Deep-Sea Res. I*, Vol. 55, pp 50-72.

Millero, F.J., Li, X. (1994) Comments on "*Speed of sound in seawater at high pressures"*, J. Acoust. Soc. Am. 95 (5) pp 2757-2759.

Papadakis, E. P., Fowler, K. A., and Lynnworth, L. C., *Ultrasonic attenuation by spectrum analysis of pulses in buffer rods: Method and diffraction corrections*, J. Acoust. Soc. Amer., vol. 53, no. 5, pp. 1336–1343, 1973.

Short, R.T., Fries, D.P., Toler, S.K., Lembke, C.E., and Byrne, R.H. (1999). *Development of an underwater mass-spectrometry system for in situ chemical analysis*. Meas Sci Technol 10: 1195-1201.

Short, R.T., and others, (2001). *Underwater mass spectrometers for in situ chemical analysis of the hydrosphere*. J Am Soc Mass Spectr 12: 676-682.

Stogryn, A. (1971) *Equations for Calculating the Dielectric Constant of Saline Water*, IEEE Transactions on Microwave Theory and Techniques", Vol. MTT-19, pp 733-736.

Urick, R. J., (1983) *Principles of underwater sound*, McGraw-Hill, third edition, pp249-251.

Von Rohden, C., Fehres, S., Rudtsch, S., (2015) *Capability of pure water calibrated time-of-flight sensors for the determination of speed of sound in seawater*, J. Acoust. Soc. Am. 138 (2) pp. 651-662.

Wu, C., Guan, B., Lu, C., Tam, H., (2011) *Salinity sensor based on polyimide-coated photonic crystal fiber*, Optics Society of America, Vol. 19, No. 21, 20003-20008.

Wu, J., (1994) *Bubbles in the near-Surface Ocean: their various structures*, Journal of physical oceanography, Vol. 24 pp. 1955-1965

Yamada, S., and Sato, H., (1962) *Some physical properties of glassy carbon*. Nature No. 4812, 261-262.

Appendix 1 Glossary

This glossary provides a description of terms commonly used in the disciplines of underwater acoustics and ocean sensors. The terms are provided in alphabetical order.

Acoustic impedance: The acoustic impedance of a material at an interface is the characteristic acoustic impedance of the material multiplied by the cosine of the incident angle of an acoustic wave propagating in the material. The acoustic impedance is the magnitude of impedance acting normal to the boundary.

Characteristic acoustic impedance: The characteristic acoustic impedance of a material is the density of the material multiplied by the sound speed of the material and is independent of the acoustic wave propagation angle with respect to the material. The characteristic acoustic impedance is equal to the acoustic impedance at a material interface when the acoustic wave propagation is normal to the interface.

Conductivity: The electrical conductivity of seawater. There are two types of sensors used to measure conductivity, conductive cells and inductive cells. Conductive cells have electrodes in contact with the sea water and measure the conductivity within a constrained and fixed volume of the seawater. The electrodes and cell walls within conductive cells are extremely susceptible to fouling. Inductive cells use a magnetic field to induce a current through a partially, or fully, constrained loop of seawater and use a coil to measure the current flow which is proportional to the electrical resistance of the

seawater loop. Inductive cells usually have external electric fields which can affect sensor accuracy if an object enters or grows within the field.

CTD: A CTD is a conductivity, temperature and depth instrument. Used to measure conductivity, temperature and pressure. These parameters can then be used to calculate depth, salinity and density.

EOS-80: 1980 equations of state of seawater

FPGA: Field programmable gate array

Far field: The sound field of a transducer is divided into two zones, the near field and the far field. The far field is the region distant from the transducer where the sound pressure decays with range according to the inverse square law.

Grazing angle: The angle between the acoustic wave propagation direction in a material and the boundary of the material.

GS/s: Giga samples per second. Used in the acoustics instrumentation industry to distinquish between digitizing rate (S/s) and acoustic frequency (Hz).

Incident angle: The angle between the acoustic wave propagation direction in a material and the vector normal to the material boundary.

Material: The solid, liquid or gas through which an acoustic wave is propagating.

Medium: The solid, liquid or gas through which an acoustic wave is propagating.

Near field: The sound field of a transducer is divided into two zones, the near field and the far field. The near field is the region close to the transducer where the sound pressure

goes through a series of maxima and minima, and it ends at the last on-axis maximum at distance from the face. (from Olympus NDT website)

Pressure reflection coefficient: The acoustic pressure reflection coefficient is the ratio of the amount of acoustic pressure reflected from an interface such as the seawater sea floor boundary to the incident acoustic pressure. The pressure reflection coefficient is dependent on the acoustic impedance mismatch between the two media and is therefore dependent on the incident angle of the acoustic energy to the interface. The pressure reflection coefficient is sometimes confused with the reflection coefficient which is the ratio of reflected acoustic energy to the incident acoustic energy. The pressure reflection coefficient is the square root of the reflection coefficient.

Ra: Mechanical engineering specification for surface finish based on the average roughness.

Reference Composition: The exact mole fractions of specific salts, as defined by Millero et al. (2008), which is representative of standard sea water. The salt ratios are listed in table 1.1.

Reflection coefficient:The acoustic reflection coefficient is the ratio of the amount of acoustic energy reflected from an interface such as the seawater sea floor boundary to the incident acoustic energy. The reflection coefficient is dependent on the acoustic impedance mismatch between the two media and is therefore dependent on the incident angle of the acoustic energy to the interface. The reflection coefficient is sometimes confused with the pressure reflection coefficient which is the ratio of reflected acoustic pressure to the

incident acoustic pressure. The reflection coefficient is the square of the pressure reflection coefficient.

Response time: A sensor's response time is the time required for a sensor to react to a step change in the sample medium. For a linear time invariant system the time varying voltage response is $V_t = V_0 e^{-t/\tau} + V_\infty(1 - e^{-t/\tau})$, where V_0 is the initial reading at time $t = 0$ and V_∞ is the reading once the system has regained a steady state. The time is usually given as the time to achieve one time constant (τ) for the change. For a step change, one time constant is equivalent to the time required to achieve $1 - 1/e \approx 63.2\%$ of the final reading.

RSS: RSS is an acronym for root sum of squares. This is a common method for sensor manufacturers to approximate the total error from several sources. Each contributing error source has its error value squared, the squared values are summed, and the square root of the sum is given as the combined error.

Salinity spiking: An error in salinity estimation caused mainly by the mismatch in conductivity and temperature sensor response times. During instrument profiling of the water column the difference in sensor response times cause the sensors to measure each parameter at slightly different depths even though the sensors are synchronously sampled. Since the sensors are not measuring the same water at the same time this results in a salinity error when the measured parameters are combined. The error is larger with greater mismatches in sensor response times, faster profiling speeds and in areas of fast changing parameters (such as thermoclines).

Sound speed: Sound speed is the acoustic pressure wave propagation rate. It is often called sound velocity however this is incorrect since it is a directionless quantity. The sound

speed varies significantly in seawater and is dependent on depth, salinity and temperature. The sound speed can vary from 1402 m s^{-1} at a cold river mouth to 1630 m s^{-1} in deep seawater. Sound speed profiles are critical for sound propagation modelling, sonar range estimates and bearing estimates. Sound speed can also be used with temperature and pressure measurements to calculate the absolute salinity.

S/s: Samples per second. Used in the acoustics instrumentation industry to distinquish between digitizing rate (S/s) and acoustic frequency (Hz).

SSP: SSP is an acronym for sound speed profiler. It is an instrument similar to a CTD but uses a sound speed sensor instead of a conductivity sensor. The instrument is used to profile the sound speed and temperature at many depths through the water column. The instrument can be used to calculate salinity and density profiles as well.

TEOS-10: 2010 thermodynamic equation of state of seawater.

Appendix 2 SensorModel

In an effort to save money and time testing various prototype configurations, an acoustic sensor model, SensorModel, was developed by the author in MatLab to assess the material and dimensional choices. SensorModel is a one-dimensional acoustic propagation model allowing up to 26 material layers. This appendix contains the MatLab code used in the model. The primary script is SensorModel.m. SensorModel calls the following functions; SenPlots.m, CalcSensor.m, and CalcSenMod.m.

A2.1. SensorModel.m

SensorModel controls the user interface allowing the user to enter the transducer materials in up to 26 layers. The user can select from a large list of materials with associated material properties such as density, sound speed and attenuation, or the user can manually enter the material and properties. Figure 3.7, in the thesis, shows the SensorModel interface set up for the prototype *In situ* Seawater Density Sensor design.

The model allows the user to inject an acoustic signal into the transmitting piezoelectric material. This signal is propagated out in both directions from the transmitting element, with reflected and transmitted waves calculated at each boundary layer. The minimum propagating acoustic intensity can be selected by the user. A high intensity setting relative to the originating signal shortens the computation time but reduces the accuracy of the sensor output waveform.

The user can save the parameters and results for three sensor configurations. The saved sensor configurations can save the user significant time in initial setup time for sensor models.

The code for SensorModel.m is shown below.

```
function varargout = SensorModel(varargin)
% SENSORMODEL M-file for SensorModel.fig
%      SENSORMODEL, by itself, creates a new SENSORMODEL or raises the existing
%      singleton*.
%
%      H = SENSORMODEL returns the handle to a new SENSORMODEL or the handle to
%      the existing singleton*.
%
%      SENSORMODEL('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in SENSORMODEL.M with the given input arguments.
%
%      SENSORMODEL('Property','Value',...) creates a new SENSORMODEL or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before SensorModel_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to SensorModel_OpeningFcn via varargin.
%

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @SensorModel_OpeningFcn, ...
                   'gui_OutputFcn',  @SensorModel_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code


% --- Executes just before SensorModel is made visible.
function SensorModel_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to SensorModel (see VARARGIN)

% Choose default command line output for SensorModel
handles.output = hObject;
h=handles;
save h.mat h    %save handles for later

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes SensorModel wait for user response (see UIRESUME)
% uiwait(handles.figure1);

%load matrices of starting parameters
hplot1=getfield(h,'plot1');
```

```
hplot2=getfield(h,'plot2');
hplot3=getfield(h,'plot3');
hax2=getfield(h,'ax2');
hax3=getfield(h,'ax3');
format short

%load last data
load Sensor1.mat    %load saved sensor 1 name
set(h.edit1,'string',memo); %load into UITABLE
load Sensor2.mat    %load saved sensor 2 name
set(h.edit2,'string',memo); %load into UITABLE
load Sensor3.mat    %load saved sensor 3 name
set(h.edit3,'string',memo); %load into UITABLE
%load last values from last session.
Sensor=importdata('Sensor.mat');
set(h.Sensor,'Data',Sensor); %load into UITABLE
Limits=importdata('Limits.mat');
set(h.Limits,'Data',Limits); %load into UITABLE
load Excitation.mat;
Freq=Excitation(1,1);
set(h.Excitation,'Data',Excitation); %load into UITABLE
PlotLim=importdata('PlotLim.mat');
set(h.PlotLim,'Data',PlotLim); %load into UITABLE
tmax=Limits(1,1);       %max time
TxWaveform=importdata('TxWaveform.mat');
TxEchoes=importdata('TxEchoes.mat');
RxWaveform=importdata('RxWaveform.mat');
RxEchoes=importdata('RxEchoes.mat');

TxBndry=Excitation(5,1);
RxBndry=Excitation(6,1);
%update materials list
Materials=importdata('Materials.mat'); %load newest material properties matrix
%Materials=[material,thickness,density,sound speed,impedance,absorption]
matlist=char(Materials(:,1));
set(h.Layer1,'string',strvcat('Layer 1', matlist)); %load into drop list
set(h.Layer2,'string',strvcat('Layer 2', matlist)); %load into drop list
set(h.Layer3,'string',strvcat('Layer 3', matlist)); %load into drop list
set(h.Layer4,'string',strvcat('Layer 4', matlist)); %load into drop list
set(h.Layer5,'string',strvcat('Layer 5', matlist)); %load into drop list
set(h.Layer6,'string',strvcat('Layer 6', matlist)); %#ok<*VCAT> %load into drop list
set(h.Layer7,'string',strvcat('Layer 7', matlist)); %load into drop list
set(h.Layer8,'string',strvcat('Layer 8', matlist)); %load into drop list
set(h.Layer9,'string',strvcat('Layer 9', matlist)); %load into drop list
set(h.Layer10,'string',strvcat('Layer 10', matlist)); %load into drop list
set(h.Layer11,'string',strvcat('Layer 11', matlist)); %load into drop list
set(h.Layer12,'string',strvcat('Layer 12', matlist)); %load into drop list
set(h.Layer13,'string',strvcat('Layer 13', matlist)); %load into drop list
set(h.Layer14,'string',strvcat('Layer 14', matlist)); %load into drop list
set(h.Layer15,'string',strvcat('Layer 15', matlist)); %load into drop list
set(h.Layer16,'string',strvcat('Layer 16', matlist)); %load into drop list
set(h.Layer17,'string',strvcat('Layer 17', matlist)); %load into drop list
set(h.Layer18,'string',strvcat('Layer 18', matlist)); %load into drop list
set(h.Layer19,'string',strvcat('Layer 19', matlist)); %load into drop list
set(h.Layer20,'string',strvcat('Layer 20', matlist)); %load into drop list
set(h.Layer21,'string',strvcat('Layer 21', matlist)); %load into drop list
set(h.Layer22,'string',strvcat('Layer 22', matlist)); %load into drop list
set(h.Layer23,'string',strvcat('Layer 23', matlist)); %load into drop list
set(h.Layer24,'string',strvcat('Layer 24', matlist)); %load into drop list
set(h.Layer25,'string',strvcat('Layer 25', matlist)); %load into drop list
set(h.Layer26,'string',strvcat('Layer 26', matlist)); %load into drop list
```

```
% call the calculate sensor function
[Sensor,coefdn,coefup]=CalcSensor(Sensor,TxBndry,h);

%Plots for GUI
thi=tmax;                %max plot time
tlow=0;                  %min plot time
elow=-inf;
ehi=inf;
w2low=-inf;
w2hi=inf;
w3low=-inf;
w3hi=inf;
%Set up Tx plots
set(hplot2,...
            'XAxisLocation','top',...
            'YAxisLocation','right',...
            'Color','w',...
            'XColor','k','YColor','k',...
            'XMinorTick','on',...
            'YMinorTick','on');
axes(hplot2)
hold off
xlabel('Frequency (Hz)');
ylabel('FFT Magnitude');
title('Output pressure , sum of all echoes   and   FFT amplitude spectum');
gcf;
axes(hax2)
set(hax2,...
            'Position',get(hplot2,'Position'),...
            'XAxisLocation','bottom',...
            'YAxisLocation','left',...
            'Color','none',...
            'XColor','r','YColor','r');
hold off
ylabel('Tx pressure (Pa)');
xlabel('Time (s)');
set(gca,'XMinorTick','on','YMinorTick','on')
grid on

%Set up Rx plots
set(hplot3,...
            'XAxisLocation','top',...
            'YAxisLocation','right',...
            'Color','w',...
            'XColor','k','YColor','k',...
            'XMinorTick','on',...
            'YMinorTick','on');
axes(hplot3)
hold off
xlabel('Frequency (Hz)');
ylabel('FFT Magnitude');
gcf;
axes(hax3)
set(hax3,...
            'Position',get(hplot3,'Position'),...
            'XAxisLocation','bottom',...
            'YAxisLocation','left',...
            'Color','none',...
            'XColor','b','YColor','b');
hold off
ylabel('Rx pressure (Pa)');
xlabel('Time (s)');
```

```
set(gca,'XMinorTick','on','YMinorTick','on')
grid on

%generate separate Rx figure for real world comparisons
figure('Name','Receive Waveform','NumberTitle','off');
cla(1)
hold off
line(RxWaveform(:,1),RxWaveform(:,2),'Color','b');
axis([tlow,thi,w3low,w3hi])
ylabel('Rx pressure (Pa)');
xlabel('Time (s)');
set(gca,'XMinorTick','on','YMinorTick','on')
grid on

SenPlots(hplot1,hplot2,hax2,hplot3,hax3,TxEchoes,RxEchoes,TxWaveform,...
    RxWaveform,tlow,thi,elow,ehi,w2low,w2hi,w3low,w3hi,Freq)

% --- Outputs from this function are returned to the command line.
function varargout = SensorModel_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes when entered data in editable cell(s) in Sensor.
function Sensor_CellEditCallback(hObject, eventdata, handles) %#ok<*DEFNU>
% hObject    handle to Sensor (see GCBO)
% eventdata  structure with the following fields (see UITABLE)
%   Indices: row and column indices of the cell(s) edited
%   PreviousData: previous data for the cell(s) edited
%   EditData: string(s) entered by the user
%   NewData: EditData or its converted form set on the Data property. Empty if Data
was not changed
%   Error: error string when failed to convert EditData to appropriate value for Data
% handles    structure with handles and user data (see GUIDATA)
% Load variables from table
handles.output = hObject;
h=handles;

% Update handles structure
guidata(hObject, handles);

%change Calculate button to yellow to indicate plots do not match numbers
set(h.Calculate,'BackgroundColor','y'); %
set(h.Calculate,'string','Requires Calc'); %
drawnow

load Excitation.mat
TxBndry=Excitation(5,1);

Sensor=get(h.Sensor,'Data'); %get data from last window settings

% call the calculate sensor function
CalcSensor(Sensor,TxBndry,h);



% --- Executes when entered data in editable cell(s) in Excitation.
```

```
function Excitation_CellEditCallback(hObject, eventdata, handles)
% hObject    handle to Excitation (see GCBO)
% eventdata  structure with the following fields (see UITABLE)
%   Indices: row and column indices of the cell(s) edited
%   PreviousData: previous data for the cell(s) edited
%   EditData: string(s) entered by the user
%   NewData: EditData or its converted form set on the Data property. Empty if Data
was not changed
%   Error: error string when failed to convert EditData to appropriate value for Data
% handles    structure with handles and user data (see GUIDATA)
% Load variables from table
handles.output = hObject;
h=handles;

% Update handles structure
guidata(hObject, handles);

%change Calculate button to yellow to indicate plots do not match numbers
set(h.Calculate,'BackgroundColor','y'); %
set(h.Calculate,'string','Requires Calc'); %
drawnow
%Excitation=get(h.Excitation,'Data'); %get data from last window settings


% --- Executes when entered data in editable cell(s) in Limits.
function Limits_CellEditCallback(hObject, eventdata, handles)
% hObject    handle to Limits (see GCBO)
% eventdata  structure with the following fields (see UITABLE)
%   Indices: row and column indices of the cell(s) edited
%   PreviousData: previous data for the cell(s) edited
%   EditData: string(s) entered by the user
%   NewData: EditData or its converted form set on the Data property. Empty if Data
was not changed
%   Error: error string when failed to convert EditData to appropriate value for Data
% handles    structure with handles and user data (see GUIDATA)
% Load variables from table
handles.output = hObject;
h=handles;

% Update handles structure
guidata(hObject, handles);

%change Calculate button to yellow to indicate plots do not match numbers
set(h.Calculate,'BackgroundColor','y'); %
set(h.Calculate,'string','Requires Calc'); %
drawnow
%load matrices of starting parameters
%Limits=get(h.Limits,'Data'); %get data from last window settings




% --- Executes on button press in Calculate.
function Calculate_Callback(hObject, eventdata, handles)
% hObject    handle to Calculate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Choose default command line output for SensorModel
handles.output = hObject;
h=handles;
save h.mat h    %save handles for later
```

```
% Update handles structure
guidata(hObject, handles);

%change Calculate button to red while processing
set(h.Calculate,'BackgroundColor','r'); %
set(h.Calculate,'string','Processing'); %

%load matrices of parameters
hplot1=getfield(h,'plot1');
hplot2=getfield(h,'plot2');
hplot3=getfield(h,'plot3');
hax2=getfield(h,'ax2');
hax3=getfield(h,'ax3');
Sensor=get(h.Sensor,'Data'); %get data from last window settings
Excitation=get(h.Excitation,'Data'); %get data from last window settings
Limits=get(h.Limits,'Data'); %get data from last window settings
PlotLim=get(h.PlotLim,'Data'); %get data from last window settings
Freq=Excitation(1,1);
TxBndry=Excitation(5,1);
RxBndry=Excitation(6,1);

% call the calculate sensor function
[Sensor,coefdn,coefup]=CalcSensor(Sensor,TxBndry,h);

% call the calculation function
[Sensor,TxWaveform,RxWaveform,TxEchoes,RxEchoes]=...
    CalcSenMod(Sensor,Limits,Excitation,coefdn,coefup,...
    TxBndry,RxBndry,h);

%save data for next time
save Limits.mat Limits  %save Limits
save PlotLim.mat PlotLim  %save Limits
save Excitation.mat Excitation   %save Excitation
save Sensor.mat Sensor  %save Sensor
set(h.Sensor,'Data',Sensor); %load into UITABLE

%change Calculate button to green at end of processing
set(h.Calculate,'BackgroundColor','g'); %

%change Calculate button to green at end of processing
set(h.Calculate,'BackgroundColor','g'); %
set(h.Calculate,'string','Calculate'); %

PlotLim=get(h.PlotLim,'Data'); %get data from last window settings

%plot variables
tlow=PlotLim(1,1);      %low time
thi=PlotLim(1,2);       %high time
w2low=PlotLim(2,1);      %Tx waveform low press limit
w2hi=PlotLim(2,2);
w3low=PlotLim(3,1);      %Rx waveform low press limit
w3hi=PlotLim(3,2);
elow=PlotLim(4,1);      %echo low Intensity limit
ehi=PlotLim(4,2);

% plot
SenPlots(hplot1,hplot2,hax2,hplot3,hax3,TxEchoes,RxEchoes,TxWaveform,RxWaveform,...
    tlow,thi,elow,ehi,w2low,w2hi,w3low,w3hi,Freq)

% --- Executes when entered data in editable cell(s) in PlotLim.
function PlotLim_CellEditCallback(hObject, eventdata, handles)
% hObject    handle to PlotLim (see GCBO)
```

```
% eventdata  structure with the following fields (see UITABLE)
%   Indices: row and column indices of the cell(s) edited
%   PreviousData: previous data for the cell(s) edited
%   EditData: string(s) entered by the user
%   NewData: EditData or its converted form set on the Data property. Empty if Data
was not changed
%   Error: error string when failed to convert EditData to appropriate value for Data
% handles    structure with handles and user data (see GUIDATA)
handles.output = hObject;
h=handles;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes SensorModel wait for user response (see UIRESUME)
% uiwait(handles.figure1);

%load matrices of parameters
hplot1=getfield(h,'plot1');
hplot2=getfield(h,'plot2');
hplot3=getfield(h,'plot3');
hax2=getfield(h,'ax2');
hax3=getfield(h,'ax3');
load Excitation.mat;
Freq=Excitation(1,1);
TxWaveform=importdata('TxWaveform.mat');
TxEchoes=importdata('TxEchoes.mat');
RxWaveform=importdata('RxWaveform.mat');
RxEchoes=importdata('RxEchoes.mat');
PlotLim=get(h.PlotLim,'Data'); %get data from last window settings

%plot variables
tlow=PlotLim(1,1);      %low time
thi=PlotLim(1,2);       %high time
w2low=PlotLim(2,1);      %Tx waveform low press limit
w2hi=PlotLim(2,2);
w3low=PlotLim(3,1);      %Rx waveform low press limit
w3hi=PlotLim(3,2);
elow=PlotLim(4,1);      %echo low Intensity limit
ehi=PlotLim(4,2);

% plot
SenPlots(hplot1,hplot2,hax2,hplot3,hax3,TxEchoes,RxEchoes,TxWaveform,RxWaveform,...
    tlow,thi,elow,ehi,w2low,w2hi,w3low,w3hi,Freq)



% --- Executes on button press in RSen1.
function RSen1_Callback(hObject, eventdata, handles)
% hObject    handle to RSen1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.output = hObject;
h=handles;
guidata(hObject, handles);

%change Calculate button to red while processing
set(h.Calculate,'BackgroundColor','r'); %
set(h.Calculate,'string','Processing'); %
drawnow

format short
```

```
hplot1=getfield(h,'plot1');
hplot2=getfield(h,'plot2');
hplot3=getfield(h,'plot3');
hax2=getfield(h,'ax2');
hax3=getfield(h,'ax3');

load Sensor1.mat
%save Sensor1.mat Sensor Limits PlotLim Excitation memo ...
    %RxEchoes RxWaveform TxEchoes TxWaveform ;
set(h.Sensor,'Data',Sensor); %load into UITABLE
set(h.Limits,'Data',Limits); %load into UITABLE
set(h.Excitation,'Data',Excitation); %load into UITABLE
set(h.PlotLim,'Data',PlotLim); %load into UITABLE
set(h.edit1,'string',memo); %load into UITABLE
Freq=Excitation(1,1);
TxBndry=Excitation(5,1);
RxBndry=Excitation(6,1);


% call the calculate sensor function
%[Sensor,coefdn,coefup]=CalcSensor(Sensor,TxBndry,h);

% call the calculation function
%[Sensor,TxWaveform,RxWaveform,TxEchoes,RxEchoes]=...
%    CalcSenMod(Sensor,Limits,Excitation,coefdn,coefup,...
%    TxBndry,RxBndry,h);


set(h.Calculate,'string','Calculate'); %

%plot variables
tlow=PlotLim(1,1);      %low time
thi=PlotLim(1,2);       %high time
w2low=PlotLim(2,1);      %Tx waveform low press limit
w2hi=PlotLim(2,2);
w3low=PlotLim(2,1);      %Rx waveform low press limit
w3hi=PlotLim(2,2);
elow=PlotLim(3,1);      %echo low Intensity limit
ehi=PlotLim(3,2);

% plot
SenPlots(hplot1,hplot2,hax2,hplot3,hax3,TxEchoes,RxEchoes,TxWaveform,RxWaveform,...
    tlow,thi,elow,ehi,w2low,w2hi,w3low,w3hi,Freq)

%change Calculate button to green at end of processing
set(h.Calculate,'BackgroundColor','g'); %
set(h.Calculate,'string','Calculate'); %




% --- Executes on button press in RSen2.
function RSen2_Callback(hObject, eventdata, handles)
% hObject    handle to RSen2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.output = hObject;
h=handles;
guidata(hObject, handles);

%change Calculate button to red while processing
```

```
set(h.Calculate,'BackgroundColor','r'); %
set(h.Calculate,'string','Processing'); %
drawnow

format short
hplot1=getfield(h,'plot1');
hplot2=getfield(h,'plot2');
hplot3=getfield(h,'plot3');
hax2=getfield(h,'ax2');
hax3=getfield(h,'ax3');

load Sensor2.mat
%save Sensor2.mat Sensor Limits PlotLim Excitation memo ...
    %RxEchoes RxWaveform TxEchoes TxWaveform ;
set(h.Sensor,'Data',Sensor); %load into UITABLE
set(h.Limits,'Data',Limits); %load into UITABLE
set(h.Excitation,'Data',Excitation); %load into UITABLE
set(h.PlotLim,'Data',PlotLim); %load into UITABLE
set(h.edit2,'string',memo); %load into UITABLE
Freq=Excitation(1,1);
TxBndry=Excitation(5,1);
RxBndry=Excitation(6,1);


% call the calculate sensor function
%[Sensor,coefdn,coefup]=CalcSensor(Sensor,TxBndry,h);

% call the calculation function
%[Sensor,TxWaveform,RxWaveform,TxEchoes,RxEchoes]=...
%    CalcSenMod(Sensor,Limits,Excitation,coefdn,coefup,...
%    TxBndry,RxBndry,h);


set(h.Calculate,'string','Calculate'); %

%plot variables
tlow=PlotLim(1,1);      %low time
thi=PlotLim(1,2);       %high time
w2low=PlotLim(2,1);      %Tx waveform low press limit
w2hi=PlotLim(2,2);
w3low=PlotLim(2,1);      %Rx waveform low press limit
w3hi=PlotLim(2,2);
elow=PlotLim(3,1);      %echo low Intensity limit
ehi=PlotLim(3,2);

% plot
SenPlots(hplot1,hplot2,hax2,hplot3,hax3,TxEchoes,RxEchoes,TxWaveform,RxWaveform,...
    tlow,thi,elow,ehi,w2low,w2hi,w3low,w3hi,Freq)

%change Calculate button to green at end of processing
set(h.Calculate,'BackgroundColor','g'); %
set(h.Calculate,'string','Calculate'); %




% --- Executes on button press in RSen3.
function RSen3_Callback(hObject, eventdata, handles)
% hObject    handle to RSen3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.output = hObject;
```

```
h=handles;
guidata(hObject, handles);

%change Calculate button to red while processing
set(h.Calculate,'BackgroundColor','r'); %
set(h.Calculate,'string','Processing'); %
drawnow

format short
hplot1=getfield(h,'plot1');
hplot2=getfield(h,'plot2');
hplot3=getfield(h,'plot3');
hax2=getfield(h,'ax2');
hax3=getfield(h,'ax3');

load Sensor3.mat
%save Sensor3.mat Sensor Limits PlotLim Excitation memo ...
    %RxEchoes RxWaveform TxEchoes TxWaveform ;
set(h.Sensor,'Data',Sensor); %load into UITABLE
set(h.Limits,'Data',Limits); %load into UITABLE
set(h.Excitation,'Data',Excitation); %load into UITABLE
set(h.PlotLim,'Data',PlotLim); %load into UITABLE
set(h.edit3,'string',memo); %load into UITABLE
Freq=Excitation(1,1);
TxBndry=Excitation(5,1);
RxBndry=Excitation(6,1);


% call the calculate sensor function
%[Sensor,coefdn,coefup]=CalcSensor(Sensor,TxBndry,h);

% call the calculation function
%[Sensor,TxWaveform,RxWaveform,TxEchoes,RxEchoes]=...
%    CalcSenMod(Sensor,Limits,Excitation, coefdn,coefup,...
%    TxBndry,RxBndry,h);


set(h.Calculate,'string','Calculate'); %

%plot variables
tlow=PlotLim(1,1);      %low time
thi=PlotLim(1,2);       %high time
w2low=PlotLim(2,1);      %Tx waveform low press limit
w2hi=PlotLim(2,2);
w3low=PlotLim(2,1);      %Rx waveform low press limit
w3hi=PlotLim(2,2);
elow=PlotLim(3,1);      %echo low Intensity limit
ehi=PlotLim(3,2);

% plot
SenPlots(hplot1,hplot2,hax2,hplot3,hax3,TxEchoes,RxEchoes,TxWaveform,RxWaveform,...
    tlow,thi,elow,ehi,w2low,w2hi,w3low,w3hi,Freq)

%change Calculate button to green at end of processing
set(h.Calculate,'BackgroundColor','g'); %
set(h.Calculate,'string','Calculate'); %



% --- Executes on button press in SSen1.
function SSen1_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to SSen1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.output = hObject;
h=handles;
guidata(hObject, handles);

%load matrices of parameters
Sensor=get(h.Sensor,'Data'); %get data from last window settings
Limits=get(h.Limits,'Data'); %get data from last window settings
PlotLim=get(h.PlotLim,'Data'); %get data from last window settings
Excitation=get(h.Excitation,'Data'); %get data from last window settings
memo=get(h.edit1,'string');
load RxEchoes.mat;
load RxWaveform.mat;
load TxEchoes.mat;
load TxWaveform.mat;

%save Sensor 1 Sensor table for next time
save Sensor1.mat Sensor Limits PlotLim Excitation memo ...
    RxEchoes RxWaveform TxEchoes TxWaveform ;



% --- Executes on button press in SSen2.
function SSen2_Callback(hObject, eventdata, handles)
% hObject    handle to SSen2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.output = hObject;
h=handles;
guidata(hObject, handles);

%load matrices of parameters
Sensor=get(h.Sensor,'Data'); %get data from last window settings
Limits=get(h.Limits,'Data'); %get data from last window settings
PlotLim=get(h.PlotLim,'Data'); %get data from last window settings
Excitation=get(h.Excitation,'Data'); %get data from last window settings
memo=get(h.edit2,'string');
load RxEchoes.mat;
load RxWaveform.mat;
load TxEchoes.mat;
load TxWaveform.mat;

%save Sensor 2 Sensor table for next time
save Sensor2.mat Sensor Limits PlotLim Excitation memo ...
    RxEchoes RxWaveform TxEchoes TxWaveform ;



% --- Executes on button press in SSen3.
function SSen3_Callback(hObject, eventdata, handles)
% hObject    handle to SSen3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.output = hObject;
h=handles;
guidata(hObject, handles);

%load matrices of parameters
Sensor=get(h.Sensor,'Data'); %get data from last window settings
```

```
Limits=get(h.Limits,'Data'); %get data from last window settings
PlotLim=get(h.PlotLim,'Data'); %get data from last window settings
Excitation=get(h.Excitation,'Data'); %get data from last window settings
memo=get(h.edit3,'string');
load RxEchoes.mat;
load RxWaveform.mat;
load TxEchoes.mat;
load TxWaveform.mat;

%save Sensor 3 Sensor table for next time
save Sensor3.mat Sensor Limits PlotLim Excitation memo ...
    RxEchoes RxWaveform TxEchoes TxWaveform ;




% --- Executes during object creation, after setting all properties.
function Layer1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Layer1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in Layer1.
function Layer1_Callback(hObject, eventdata, handles)
% hObject    handle to Layer1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns Layer1 contents as cell
array
%        contents{get(hObject,'Value')} returns selected item from Layer1
mat={get(hObject,'Value')}; %returns selected item
load Materials.mat
load Sensor.mat
load Excitation.mat
TxBndry=Excitation(5,1);
%Load basic material properties but do not change the thickness
Sensor(1,1)=Materials(mat{1,1}-1,1);
Sensor(1,3:4)=Materials(mat{1,1}-1,3:4);
Sensor(1,6)=Materials(mat{1,1}-1,6);

load h.mat
% call the calculate sensor function
CalcSensor(Sensor,TxBndry,h);

%change Calculate button to yellow to indicate plots do not match numbers
set(h.Calculate,'BackgroundColor','y'); %
set(h.Calculate,'string','Requires Calc'); %
drawnow



% --- Executes during object creation, after setting all properties.
```

```
function Layer2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Layer2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in Layer2.
function Layer2_Callback(hObject, eventdata, handles)
% hObject    handle to Layer2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns Layer2 contents as cell
array
%        contents{get(hObject,'Value')} returns selected item from Layer2
mat={get(hObject,'Value')}; %returns selected item
load Materials.mat
load Sensor.mat
load Excitation.mat
TxBndry=Excitation(5,1);
%Load basic material properties but do not change the thickness
Sensor(2,1)=Materials(mat{1,1}-1,1);
Sensor(2,3:4)=Materials(mat{1,1}-1,3:4);
Sensor(2,6)=Materials(mat{1,1}-1,6);

load h.mat
% call the calculate sensor function
CalcSensor(Sensor,TxBndry,h);

%change Calculate button to yellow to indicate plots do not match numbers
set(h.Calculate,'BackgroundColor','y'); %
set(h.Calculate,'string','Requires Calc'); %
drawnow




% --- Executes during object creation, after setting all properties.
function Layer3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Layer3 (see GCBO)
% handles    empty - handles not created until after all CreateFcns called
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in Layer3.
function Layer3_Callback(hObject, eventdata, handles)
% hObject    handle to Layer3 (see GCBO)
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns full contents as cell array
%        contents{get(hObject,'Value')} returns selected item
mat={get(hObject,'Value')}; %returns selected item
load Materials.mat
load Sensor.mat
load Excitation.mat
```

```
TxBndry=Excitation(5,1);
%Load basic material properties but do not change the thickness
Sensor(3,1)=Materials(mat{1,1}-1,1);
Sensor(3,3:4)=Materials(mat{1,1}-1,3:4);
Sensor(3,6)=Materials(mat{1,1}-1,6);

load h.mat
% call the calculate sensor function
CalcSensor(Sensor,TxBndry,h);

%change Calculate button to yellow to indicate plots do not match numbers
set(h.Calculate,'BackgroundColor','y'); %
set(h.Calculate,'string','Requires Calc'); %
drawnow



% --- Executes during object creation, after setting all properties.
function Layer4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Layer4 (see GCBO)
% handles    empty - handles not created until after all CreateFcns called
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in Layer4.
function Layer4_Callback(hObject, eventdata, handles)
% hObject    handle to Layer4 (see GCBO)
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns full contents as cell array
%        contents{get(hObject,'Value')} returns selected item
mat={get(hObject,'Value')}; %returns selected item
load Materials.mat
load Sensor.mat
load Excitation.mat
TxBndry=Excitation(5,1);
%Load basic material properties but do not change the thickness
Sensor(4,1)=Materials(mat{1,1}-1,1);
Sensor(4,3:4)=Materials(mat{1,1}-1,3:4);
Sensor(4,6)=Materials(mat{1,1}-1,6);

load h.mat
% call the calculate sensor function
CalcSensor(Sensor,TxBndry,h);

%change Calculate button to yellow to indicate plots do not match numbers
set(h.Calculate,'BackgroundColor','y'); %
set(h.Calculate,'string','Requires Calc'); %
drawnow



% --- Executes during object creation, after setting all properties.
function Layer5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Layer5 (see GCBO)
% handles    empty - handles not created until after all CreateFcns called
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end

% --- Executes on selection change in Layer5.
function Layer5_Callback(hObject, eventdata, handles)
% hObject    handle to Layer5 (see GCBO)
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns full contents as cell array
%        contents{get(hObject,'Value')} returns selected item
mat={get(hObject,'Value')}; %returns selected item
load Materials.mat
load Sensor.mat
load Excitation.mat
TxBndry=Excitation(5,1);
%Load basic material properties but do not change the thickness
Sensor(5,1)=Materials(mat{1,1}-1,1);
Sensor(5,3:4)=Materials(mat{1,1}-1,3:4);
Sensor(5,6)=Materials(mat{1,1}-1,6);

load h.mat
% call the calculate sensor function
CalcSensor(Sensor,TxBndry,h);

%change Calculate button to yellow to indicate plots do not match numbers
set(h.Calculate,'BackgroundColor','y'); %
set(h.Calculate,'string','Requires Calc'); %
drawnow



% --- Executes during object creation, after setting all properties.
function Layer6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Layer6 (see GCBO)
% handles    empty - handles not created until after all CreateFcns called
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in Layer6.
function Layer6_Callback(hObject, eventdata, handles)
% hObject    handle to Layer6 (see GCBO)
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns full contents as cell array
%        contents{get(hObject,'Value')} returns selected item
mat={get(hObject,'Value')}; %returns selected item
load Materials.mat
load Sensor.mat
load Excitation.mat
TxBndry=Excitation(5,1);
%Load basic material properties but do not change the thickness
Sensor(6,1)=Materials(mat{1,1}-1,1);
Sensor(6,3:4)=Materials(mat{1,1}-1,3:4);
Sensor(6,6)=Materials(mat{1,1}-1,6);

load h.mat
% call the calculate sensor function
CalcSensor(Sensor,TxBndry,h);

%change Calculate button to yellow to indicate plots do not match numbers
set(h.Calculate,'BackgroundColor','y'); %
set(h.Calculate,'string','Requires Calc'); %
```

```
drawnow



% --- Executes during object creation, after setting all properties.
function Layer7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Layer7 (see GCBO)
% handles    empty - handles not created until after all CreateFcns called
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in Layer7.
function Layer7_Callback(hObject, eventdata, handles)
% hObject    handle to Layer7 (see GCBO)
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns full contents as cell array
%        contents{get(hObject,'Value')} returns selected item
mat={get(hObject,'Value')}; %returns selected item
load Materials.mat
load Sensor.mat
load Excitation.mat
TxBndry=Excitation(5,1);
%Load basic material properties but do not change the thickness
Sensor(7,1)=Materials(mat{1,1}-1,1);
Sensor(7,3:4)=Materials(mat{1,1}-1,3:4);
Sensor(7,6)=Materials(mat{1,1}-1,6);

load h.mat
% call the calculate sensor function
CalcSensor(Sensor,TxBndry,h);

%change Calculate button to yellow to indicate plots do not match numbers
set(h.Calculate,'BackgroundColor','y'); %
set(h.Calculate,'string','Requires Calc'); %
drawnow



% --- Executes during object creation, after setting all properties.
function Layer8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Layer8 (see GCBO)
% handles    empty - handles not created until after all CreateFcns called
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in Layer8.
function Layer8_Callback(hObject, eventdata, handles)
% hObject    handle to Layer8 (see GCBO)
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns full contents as cell array
%        contents{get(hObject,'Value')} returns selected item
mat={get(hObject,'Value')}; %returns selected item
load Materials.mat
load Sensor.mat
load Excitation.mat
```

```
TxBndry=Excitation(5,1);
%Load basic material properties but do not change the thickness
Sensor(8,1)=Materials(mat{1,1}-1,1);
Sensor(8,3:4)=Materials(mat{1,1}-1,3:4);
Sensor(8,6)=Materials(mat{1,1}-1,6);

load h.mat
% call the calculate sensor function
CalcSensor(Sensor,TxBndry,h);

%change Calculate button to yellow to indicate plots do not match numbers
set(h.Calculate,'BackgroundColor','y'); %
set(h.Calculate,'string','Requires Calc'); %
drawnow



% --- Executes during object creation, after setting all properties.
function Layer9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Layer9 (see GCBO)
% handles    empty - handles not created until after all CreateFcns called
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in Layer9.
function Layer9_Callback(hObject, eventdata, handles)
% hObject    handle to Layer9 (see GCBO)
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns full contents as cell array
%        contents{get(hObject,'Value')} returns selected item
mat={get(hObject,'Value')}; %returns selected item
load Materials.mat
load Sensor.mat
load Excitation.mat
TxBndry=Excitation(5,1);
%Load basic material properties but do not change the thickness
Sensor(9,1)=Materials(mat{1,1}-1,1);
Sensor(9,3:4)=Materials(mat{1,1}-1,3:4);
Sensor(9,6)=Materials(mat{1,1}-1,6);

load h.mat
% call the calculate sensor function
CalcSensor(Sensor,TxBndry,h);

%change Calculate button to yellow to indicate plots do not match numbers
set(h.Calculate,'BackgroundColor','y'); %
set(h.Calculate,'string','Requires Calc'); %
drawnow



% --- Executes during object creation, after setting all properties.
function Layer10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Layer10 (see GCBO)
% handles    empty - handles not created until after all CreateFcns called
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end

% --- Executes on selection change in Layer10.
function Layer10_Callback(hObject, eventdata, handles)
% hObject    handle to Layer10 (see GCBO)
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns full contents as cell array
%        contents{get(hObject,'Value')} returns selected item
mat={get(hObject,'Value')}; %returns selected item
load Materials.mat
load Sensor.mat
load Excitation.mat
TxBndry=Excitation(5,1);
%Load basic material properties but do not change the thickness
Sensor(10,1)=Materials(mat{1,1}-1,1);
Sensor(10,3:4)=Materials(mat{1,1}-1,3:4);
Sensor(10,6)=Materials(mat{1,1}-1,6);

load h.mat
% call the calculate sensor function
CalcSensor(Sensor,TxBndry,h);

%change Calculate button to yellow to indicate plots do not match numbers
set(h.Calculate,'BackgroundColor','y'); %
set(h.Calculate,'string','Requires Calc'); %
drawnow



% --- Executes during object creation, after setting all properties.
function Layer11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Layer11 (see GCBO)
% handles    empty - handles not created until after all CreateFcns called
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in Layer11.
function Layer11_Callback(hObject, eventdata, handles)
% hObject    handle to Layer11 (see GCBO)
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns full contents as cell array
%        contents{get(hObject,'Value')} returns selected item
mat={get(hObject,'Value')}; %returns selected item
load Materials.mat
load Sensor.mat
load Excitation.mat
TxBndry=Excitation(5,1);
%Load basic material properties but do not change the thickness
Sensor(11,1)=Materials(mat{1,1}-1,1);
Sensor(11,3:4)=Materials(mat{1,1}-1,3:4);
Sensor(11,6)=Materials(mat{1,1}-1,6);

load h.mat
% call the calculate sensor function
CalcSensor(Sensor,TxBndry,h);

%change Calculate button to yellow to indicate plots do not match numbers
set(h.Calculate,'BackgroundColor','y'); %
set(h.Calculate,'string','Requires Calc'); %
```

```
drawnow



% --- Executes during object creation, after setting all properties.
function Layer12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Layer12 (see GCBO)
% handles    empty - handles not created until after all CreateFcns called
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in Layer12.
function Layer12_Callback(hObject, eventdata, handles)
% hObject    handle to Layer12 (see GCBO)
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns full contents as cell array
%        contents{get(hObject,'Value')} returns selected item
mat={get(hObject,'Value')}; %returns selected item
load Materials.mat
load Sensor.mat
load Excitation.mat
TxBndry=Excitation(5,1);
%Load basic material properties but do not change the thickness
Sensor(12,1)=Materials(mat{1,1}-1,1);
Sensor(12,3:4)=Materials(mat{1,1}-1,3:4);
Sensor(12,6)=Materials(mat{1,1}-1,6);

load h.mat
% call the calculate sensor function
CalcSensor(Sensor,TxBndry,h);

%change Calculate button to yellow to indicate plots do not match numbers
set(h.Calculate,'BackgroundColor','y'); %
set(h.Calculate,'string','Requires Calc'); %
drawnow



% --- Executes during object creation, after setting all properties.
function Layer13_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Layer13 (see GCBO)
% handles    empty - handles not created until after all CreateFcns called
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in Layer13.
function Layer13_Callback(hObject, eventdata, handles)
% hObject    handle to Layer13 (see GCBO)
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns full contents as cell array
%        contents{get(hObject,'Value')} returns selected item
mat={get(hObject,'Value')}; %returns selected item
load Materials.mat
load Sensor.mat
load Excitation.mat
TxBndry=Excitation(5,1);
```

```
%Load basic material properties but do not change the thickness
Sensor(13,1)=Materials(mat{1,1}-1,1);
Sensor(13,3:4)=Materials(mat{1,1}-1,3:4);
Sensor(13,6)=Materials(mat{1,1}-1,6);

load h.mat
% call the calculate sensor function
CalcSensor(Sensor,TxBndry,h);

%change Calculate button to yellow to indicate plots do not match numbers
set(h.Calculate,'BackgroundColor','y'); %
set(h.Calculate,'string','Requires Calc'); %
drawnow



% --- Executes during object creation, after setting all properties.
function Layer14_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Layer14 (see GCBO)
% handles    empty - handles not created until after all CreateFcns called
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in Layer14.
function Layer14_Callback(hObject, eventdata, handles)
% hObject    handle to Layer14 (see GCBO)
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns full contents as cell array
%        contents{get(hObject,'Value')} returns selected item
mat={get(hObject,'Value')}; %returns selected item
load Materials.mat
load Sensor.mat
load Excitation.mat
TxBndry=Excitation(5,1);
%Load basic material properties but do not change the thickness
Sensor(14,1)=Materials(mat{1,1}-1,1);
Sensor(14,3:4)=Materials(mat{1,1}-1,3:4);
Sensor(14,6)=Materials(mat{1,1}-1,6);

load h.mat
% call the calculate sensor function
CalcSensor(Sensor,TxBndry,h);

%change Calculate button to yellow to indicate plots do not match numbers
set(h.Calculate,'BackgroundColor','y'); %
set(h.Calculate,'string','Requires Calc'); %
drawnow



% --- Executes during object creation, after setting all properties.
function Layer15_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Layer15 (see GCBO)
% handles    empty - handles not created until after all CreateFcns called
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on selection change in Layer15.
function Layer15_Callback(hObject, eventdata, handles)
% hObject    handle to Layer15 (see GCBO)
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns full contents as cell array
%        contents{get(hObject,'Value')} returns selected item
mat={get(hObject,'Value')}; %returns selected item
load Materials.mat
load Sensor.mat
load Excitation.mat
TxBndry=Excitation(5,1);
%Load basic material properties but do not change the thickness
Sensor(15,1)=Materials(mat{1,1}-1,1);
Sensor(15,3:4)=Materials(mat{1,1}-1,3:4);
Sensor(15,6)=Materials(mat{1,1}-1,6);

load h.mat
% call the calculate sensor function
CalcSensor(Sensor,TxBndry,h);

%change Calculate button to yellow to indicate plots do not match numbers
set(h.Calculate,'BackgroundColor','y'); %
set(h.Calculate,'string','Requires Calc'); %
drawnow



% --- Executes during object creation, after setting all properties.
function Layer16_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Layer16 (see GCBO)
% handles    empty - handles not created until after all CreateFcns called
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in Layer16.
function Layer16_Callback(hObject, eventdata, handles)
% hObject    handle to Layer16 (see GCBO)
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns full contents as cell array
%        contents{get(hObject,'Value')} returns selected item
mat={get(hObject,'Value')}; %returns selected item
load Materials.mat
load Sensor.mat
load Excitation.mat
TxBndry=Excitation(5,1);
%Load basic material properties but do not change the thickness
Sensor(16,1)=Materials(mat{1,1}-1,1);
Sensor(16,3:4)=Materials(mat{1,1}-1,3:4);
Sensor(16,6)=Materials(mat{1,1}-1,6);

load h.mat
% call the calculate sensor function
CalcSensor(Sensor,TxBndry,h);

%change Calculate button to yellow to indicate plots do not match numbers
set(h.Calculate,'BackgroundColor','y'); %
set(h.Calculate,'string','Requires Calc'); %
drawnow
```

```
% --- Executes during object creation, after setting all properties.
function Layer17_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Layer17 (see GCBO)
% handles    empty - handles not created until after all CreateFcns called
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in Layer17.
function Layer17_Callback(hObject, eventdata, handles)
% hObject    handle to Layer17 (see GCBO)
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns full contents as cell array
%        contents{get(hObject,'Value')} returns selected item
mat={get(hObject,'Value')}; %returns selected item
load Materials.mat
load Sensor.mat
load Excitation.mat
TxBndry=Excitation(5,1);
%Load basic material properties but do not change the thickness
Sensor(17,1)=Materials(mat{1,1}-1,1);
Sensor(17,3:4)=Materials(mat{1,1}-1,3:4);
Sensor(17,6)=Materials(mat{1,1}-1,6);

load h.mat
% call the calculate sensor function
CalcSensor(Sensor,TxBndry,h);

%change Calculate button to yellow to indicate plots do not match numbers
set(h.Calculate,'BackgroundColor','y'); %
set(h.Calculate,'string','Requires Calc'); %
drawnow



% --- Executes during object creation, after setting all properties.
function Layer18_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Layer18 (see GCBO)
% handles    empty - handles not created until after all CreateFcns called
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in Layer18.
function Layer18_Callback(hObject, eventdata, handles)
% hObject    handle to Layer18 (see GCBO)
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns full contents as cell array
%        contents{get(hObject,'Value')} returns selected item
mat={get(hObject,'Value')}; %returns selected item
load Materials.mat
load Sensor.mat
load Excitation.mat
TxBndry=Excitation(5,1);
%Load basic material properties but do not change the thickness
```

```
Sensor(18,1)=Materials(mat{1,1}-1,1);
Sensor(18,3:4)=Materials(mat{1,1}-1,3:4);
Sensor(18,6)=Materials(mat{1,1}-1,6);

load h.mat
% call the calculate sensor function
CalcSensor(Sensor,TxBndry,h);

%change Calculate button to yellow to indicate plots do not match numbers
set(h.Calculate,'BackgroundColor','y'); %
set(h.Calculate,'string','Requires Calc'); %
drawnow



% --- Executes during object creation, after setting all properties.
function Layer19_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Layer19 (see GCBO)
% handles    empty - handles not created until after all CreateFcns called
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in Layer19.
function Layer19_Callback(hObject, eventdata, handles)
% hObject    handle to Layer19 (see GCBO)
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns full contents as cell array
%        contents{get(hObject,'Value')} returns selected item
mat={get(hObject,'Value')}; %returns selected item
load Materials.mat
load Sensor.mat
load Excitation.mat
TxBndry=Excitation(5,1);
%Load basic material properties but do not change the thickness
Sensor(19,1)=Materials(mat{1,1}-1,1);
Sensor(19,3:4)=Materials(mat{1,1}-1,3:4);
Sensor(19,6)=Materials(mat{1,1}-1,6);

load h.mat
% call the calculate sensor function
CalcSensor(Sensor,TxBndry,h);

%change Calculate button to yellow to indicate plots do not match numbers
set(h.Calculate,'BackgroundColor','y'); %
set(h.Calculate,'string','Requires Calc'); %
drawnow



% --- Executes during object creation, after setting all properties.
function Layer20_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Layer20 (see GCBO)
% handles    empty - handles not created until after all CreateFcns called
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on selection change in Layer20.
function Layer20_Callback(hObject, eventdata, handles)
% hObject    handle to Layer20 (see GCBO)
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns full contents as cell array
%        contents{get(hObject,'Value')} returns selected item
mat={get(hObject,'Value')}; %returns selected item
load Materials.mat
load Sensor.mat
load Excitation.mat
TxBndry=Excitation(5,1);
%Load basic material properties but do not change the thickness
Sensor(20,1)=Materials(mat{1,1}-1,1);
Sensor(20,3:4)=Materials(mat{1,1}-1,3:4);
Sensor(20,6)=Materials(mat{1,1}-1,6);

load h.mat
% call the calculate sensor function
CalcSensor(Sensor,TxBndry,h);

%change Calculate button to yellow to indicate plots do not match numbers
set(h.Calculate,'BackgroundColor','y'); %
set(h.Calculate,'string','Requires Calc'); %
drawnow



% --- Executes during object creation, after setting all properties.
function Layer21_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Layer21 (see GCBO)
% handles    empty - handles not created until after all CreateFcns called
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in Layer21.
function Layer21_Callback(hObject, eventdata, handles)
% hObject    handle to Layer21 (see GCBO)
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns full contents as cell array
%        contents{get(hObject,'Value')} returns selected item
mat={get(hObject,'Value')}; %returns selected item
load Materials.mat
load Sensor.mat
load Excitation.mat
TxBndry=Excitation(5,1);
%Load basic material properties but do not change the thickness
Sensor(21,1)=Materials(mat{1,1}-1,1);
Sensor(21,3:4)=Materials(mat{1,1}-1,3:4);
Sensor(21,6)=Materials(mat{1,1}-1,6);

load h.mat
% call the calculate sensor function
CalcSensor(Sensor,TxBndry,h);

%change Calculate button to yellow to indicate plots do not match numbers
set(h.Calculate,'BackgroundColor','y'); %
set(h.Calculate,'string','Requires Calc'); %
drawnow
```

```
% --- Executes during object creation, after setting all properties.
function Layer22_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Layer22 (see GCBO)
% handles    empty - handles not created until after all CreateFcns called
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in Layer22.
function Layer22_Callback(hObject, eventdata, handles)
% hObject    handle to Layer22 (see GCBO)
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns full contents as cell array
%        contents{get(hObject,'Value')} returns selected item
mat={get(hObject,'Value')}; %returns selected item
load Materials.mat
load Sensor.mat
load Excitation.mat
TxBndry=Excitation(5,1);
%Load basic material properties but do not change the thickness
Sensor(22,1)=Materials(mat{1,1}-1,1);
Sensor(22,3:4)=Materials(mat{1,1}-1,3:4);
Sensor(22,6)=Materials(mat{1,1}-1,6);

load h.mat
% call the calculate sensor function
CalcSensor(Sensor,TxBndry,h);

%change Calculate button to yellow to indicate plots do not match numbers
set(h.Calculate,'BackgroundColor','y'); %
set(h.Calculate,'string','Requires Calc'); %
drawnow



% --- Executes during object creation, after setting all properties.
function Layer23_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Layer23 (see GCBO)
% handles    empty - handles not created until after all CreateFcns called
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in Layer23.
function Layer23_Callback(hObject, eventdata, handles)
% hObject    handle to Layer23 (see GCBO)
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns full contents as cell array
%        contents{get(hObject,'Value')} returns selected item
mat={get(hObject,'Value')}; %returns selected item
load Materials.mat
load Sensor.mat
load Excitation.mat
TxBndry=Excitation(5,1);
%Load basic material properties but do not change the thickness
Sensor(23,1)=Materials(mat{1,1}-1,1);
```

```
Sensor(23,3:4)=Materials(mat{1,1}-1,3:4);
Sensor(23,6)=Materials(mat{1,1}-1,6);

load h.mat
% call the calculate sensor function
CalcSensor(Sensor,TxBndry,h);

%change Calculate button to yellow to indicate plots do not match numbers
set(h.Calculate,'BackgroundColor','y'); %
set(h.Calculate,'string','Requires Calc'); %
drawnow



% --- Executes during object creation, after setting all properties.
function Layer24_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Layer24 (see GCBO)
% handles    empty - handles not created until after all CreateFcns called
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in Layer24.
function Layer24_Callback(hObject, eventdata, handles)
% hObject    handle to Layer24 (see GCBO)
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns full contents as cell array
%        contents{get(hObject,'Value')} returns selected item
mat={get(hObject,'Value')}; %returns selected item
load Materials.mat
load Sensor.mat
load Excitation.mat
TxBndry=Excitation(5,1);
%Load basic material properties but do not change the thickness
Sensor(24,1)=Materials(mat{1,1}-1,1);
Sensor(24,3:4)=Materials(mat{1,1}-1,3:4);
Sensor(24,6)=Materials(mat{1,1}-1,6);

load h.mat
% call the calculate sensor function
CalcSensor(Sensor,TxBndry,h);

%change Calculate button to yellow to indicate plots do not match numbers
set(h.Calculate,'BackgroundColor','y'); %
set(h.Calculate,'string','Requires Calc'); %
drawnow



% --- Executes during object creation, after setting all properties.
function Layer25_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Layer25 (see GCBO)
% handles    empty - handles not created until after all CreateFcns called
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in Layer25.
```

```
function Layer25_Callback(hObject, eventdata, handles)
% hObject    handle to Layer25 (see GCBO)
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns full contents as cell array
%        contents{get(hObject,'Value')} returns selected item
mat={get(hObject,'Value')}; %returns selected item
load Materials.mat
load Sensor.mat
load Excitation.mat
TxBndry=Excitation(5,1);
%Load basic material properties but do not change the thickness
Sensor(25,1)=Materials(mat{1,1}-1,1);
Sensor(25,3:4)=Materials(mat{1,1}-1,3:4);
Sensor(25,6)=Materials(mat{1,1}-1,6);

load h.mat
% call the calculate sensor function
CalcSensor(Sensor,TxBndry,h);

%change Calculate button to yellow to indicate plots do not match numbers
set(h.Calculate,'BackgroundColor','y'); %
set(h.Calculate,'string','Requires Calc'); %
drawnow



% --- Executes during object creation, after setting all properties.
function Layer26_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Layer26 (see GCBO)
% handles    empty - handles not created until after all CreateFcns called
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in Layer26.
function Layer26_Callback(hObject, eventdata, handles)
% hObject    handle to Layer26 (see GCBO)
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns full contents as cell array
%        contents{get(hObject,'Value')} returns selected item
mat={get(hObject,'Value')}; %returns selected item
load Materials.mat
load Sensor.mat
load Excitation.mat
TxBndry=Excitation(5,1);
%Load basic material properties but do not change the thickness
Sensor(26,1)=Materials(mat{1,1}-1,1);
Sensor(26,3:4)=Materials(mat{1,1}-1,3:4);
Sensor(26,6)=Materials(mat{1,1}-1,6);

load h.mat
% call the calculate sensor function
CalcSensor(Sensor,TxBndry,h);

%change Calculate button to yellow to indicate plots do not match numbers
set(h.Calculate,'BackgroundColor','y'); %
set(h.Calculate,'string','Requires Calc'); %
drawnow
```

```
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

A2.2. SenPlots.m

SenPlots.m is called by SensorModel.m to plot the transmitter waveform, the receiver waveform and the intensity times phase plot on the user interface.

The code for SenPlots.m is shown below.

```
function SenPlots(hplot1,hplot2,hax2,hplot3,hax3,TxEchoes,RxEchoes,TxWaveform,...
    RxWaveform,tlow,thi,elow,ehi,w2low,w2hi,w3low,w3hi,Freq)

%_____________________________________________________________________
%Update Sensor Plots
%---------------------------------------------------------------------
%Called by SensorModel.m function
%Tom Dakin
%
%This plots the Tx, Rx, and intensity plots.
%---------------------------------------------------------------------
%Plots for GUI

if tlow == -inf; tlow=0; end
if thi == inf; thi=max(TxWaveform(:,1)); end
if w2low == -inf; w2low=floor(min(TxWaveform(:,2))); end
if w2hi == inf; w2hi=ceil(max(TxWaveform(:,2))); end
if w3low == -inf; w3low=floor(min(RxWaveform(:,2))); end
if w3hi == inf; w3hi=ceil(max(RxWaveform(:,2))); end


%Scatter plot of Tx and Rx echo return times and strengths, negative if
%phase is 180 degrees
set(gcf,'color','w')
axes(hplot1) %call axes for plot1
scatter(TxEchoes(:,1),TxEchoes(:,2).*TxEchoes(:,3),'r') %Plot Tx echo scatter
axis([tlow,thi,elow,ehi])
grid on
set(gca,'XMinorTick','on','YMinorTick','on')
ylabel('I*phase (W/m^2)')
xlabel('Echo return time (s)')
title('Echo rms intensity times phase versus return time')
hold on
scatter(RxEchoes(:,1),RxEchoes(:,2).*RxEchoes(:,3),'b') %Plot Rx echo scatter
hold off


%Tx FFT, plot dotted line on same plot as Tx waveform
%calculate Tx fft
```

```
res=TxWaveform(2,1);    %time resolution = 1/Fs
TxLow=floor(tlow/res+1);    %start row of displayed waveform
TxHi=floor(thi/res+1);    %stop row of displayed waveform
%check to ensure that TxHi is not outside limit of waveform
if TxHi>size(TxWaveform(:,1),1);TxHi=size(TxWaveform(:,1),1); end
TxNumPoints=TxHi-TxLow+1;     %number of data points
NFFT = 2^nextpow2(TxNumPoints); %Next power of 2 from length of displayed waveform
if NFFT <=2^24      %if there are too many data points don't run the fft
    FFT = fft(TxWaveform(TxLow:TxHi,2),NFFT)/TxNumPoints;    %calculate fft
    Frequencies = 1/res/2*linspace(0,1,NFFT/2+1)';%calculate the fft frequency bins
    %Collate Tx fft data
    %col 1 = time
    %col 2 = pressure
    %col 3 = fft freq
    %col 4 = fft magnitude
    TxData=TxWaveform(TxLow:TxHi,1:2); %Time and pressure data within window
    TxData(1:NFFT/2+1,3)=Frequencies;
    TxData(1:NFFT/2+1,4)=FFT(1:NFFT/2+1,1);
    save TxData.mat TxData           % save Tx data for post processing
    %call axes for plot2 and plot fft in behind
    axes(hplot2)
    cla(hplot2)
    line(TxData(:,3),2*abs(TxData(:,4)),'Color','k','Parent',hplot2);
    %set up grid and tick marks
    xlim([0 10^(ceil(log10(Freq)))]) %sets the axis limits in the current axes to the
specified values.
    xlimits = get(hplot2,'XLim');
    ylimits = get(hplot2,'YLim');
    xinc = (xlimits(2)-xlimits(1))/20;
    yinc = (ylimits(2)-ylimits(1))/4;
    set(hplot2,'XTick',[xlimits(1):xinc:xlimits(2)],...
        'YTick',[ylimits(1):yinc:ylimits(2)])

end

%Plot Tx waveform pressures vs time
axes(hax2);
cla(hax2)
line(TxWaveform(:,1),TxWaveform(:,2),'Color','r','Parent',hax2);
axis([tlow,thi,w2low,w2hi])

%set up grid and tick marks
xlimits = get(hax2,'XLim');
%ylimits = get(hax2,'YLim');
xinc = (xlimits(2)-xlimits(1))/20;
yinc = (w2hi-w2low)/4;
set(hax2,'XTick',[xlimits(1):xinc:xlimits(2)],...
        'YTick',[w2low:yinc:w2hi]);


%Rx FFT, plot dotted line on same plot as Rx waveform
%calculate Rx fft
RxLow=floor(tlow/res+1);    %start row of displayed waveform
RxHi=floor(thi/res+1);    %stop row of displayed waveform
%check to ensure that RxHi is not outside limit of waveform
if RxHi>size(RxWaveform(:,1),1);RxHi=size(RxWaveform(:,1),1); end
RxNumPoints=RxHi-RxLow+1;     %number of data points
NFFT = 2^nextpow2(RxNumPoints); %Next power of 2 from length of displayed waveform
if NFFT <=2^24      %if there are too many data points don't run the fft
    FFT = fft(RxWaveform(RxLow:RxHi,2),NFFT)/RxNumPoints;    %calculate fft
    Frequencies = 1/res/2*linspace(0,1,NFFT/2+1)';%calculate the fft frequency bins
    %Collate Tx fft data
```

```
    %col 1 = time
    %col 2 = pressure
    %col 3 = fft freq
    %col 4 = fft magnitude
    RxData=RxWaveform(RxLow:RxHi,1:2); %Time and pressure data within window
    RxData(1:NFFT/2+1,3)=Frequencies;
    RxData(1:NFFT/2+1,4)=FFT(1:NFFT/2+1,1);
    save RxData.mat RxData           % save Tx data for post processing
    %call axes for plot2 and plot fft in behind
    axes(hplot3)
    cla(hplot3)
    line(RxData(:,3),2*abs(RxData(:,4)),'Color','k','Parent',hplot3);
    %set up grid and tick marks
    xlim([0 10^(ceil(log10(Freq)))]) %sets the axis limits in the current axes to the
specified values.
    xlimits = get(hplot3,'XLim');
    ylimits = get(hplot3,'YLim');
    xinc = (xlimits(2)-xlimits(1))/20;
    yinc = (ylimits(2)-ylimits(1))/4;
    set(hplot3,'XTick',[xlimits(1):xinc:xlimits(2)],...
        'YTick',[ylimits(1):yinc:ylimits(2)])

end

%Plot Rx waveform pressures vs time
axes(hax3);
cla(hax3)
line(RxWaveform(:,1),RxWaveform(:,2),'Color','b','Parent',hax3);
axis([tlow,thi,w3low,w3hi])

%set up grid and tick marks
xlimits = get(hax3,'XLim');
%ylimits = get(hax3,'YLim');
%maxY=ceil(max(abs(ylimits)));
xinc = (xlimits(2)-xlimits(1))/20;
yinc = (w3hi-w3low)/4;
set(hax3,'XTick',[xlimits(1):xinc:xlimits(2)],...
        'YTick',[w3low:yinc:w3hi]);

%generate separate Rx figure for real world comparisons
figure (1)
cla(1)
hold off
line(RxWaveform(:,1),RxWaveform(:,2),'Color','b');
axis([tlow,thi,w3low,w3hi])
ylabel('Rx pressure (Pa)');
xlabel('Time (s)');
set(gca,'XMinorTick','on','YMinorTick','on')
grid on
```

A2.3. CalcSensor.m

CalcSensor.m computes the boundary coefficients for the user specified sensor configuration. The coefficients include the acoustic propagation times for each layer and the transmit and

reflection coefficients for each layer boundary. This script is run each time the user changes a parameter in the sensor configuration.

The code for CalcSensor.m is shown below.

```
function [Sensor,coefdn,coefup]=CalcSensor(Sensor,TxBndry,h)

%_____________________________________________________________________________
%Calculate Sensor Parameters
%-----------------------------------------------------------------------------
%Called by SensorModel.m function
%Tom Dakin
%
%This function calculates the sensor parameters based on the settings in
%the GUI for the SensorModel.m function.
%Sensor=[layer material, thickness, density, sound speed, impedance,
%absorption, one way propagation time, time to outboard boundary of layer,
%downward reflection coef, upward reflection coef]
%Sensor=[mat,d,rho,c,z,a,t,1stEcho,coefdn,coefup]
%-----------------------------------------------------------------------------

% Calculate layer parameters
for n=1:26
%acoustic impedance in MRayles
Sensor{n,5}=cell2mat(Sensor(n,3)).*cell2mat(Sensor(n,4))/1e6;
%travel time in s, one way
Sensor{n,7}=cell2mat(Sensor(n,2))./cell2mat(Sensor(n,4));
end
t=cell2mat(Sensor(:,7));    %layer propagation times

%1st transit time to outboard side of layer
%Start with piezo layers
Sensor{TxBndry,8}=cell2mat(Sensor(TxBndry,7));
Sensor{TxBndry+1,8}=cell2mat(Sensor(TxBndry+1,7));

for m=1:TxBndry-1
    Sensor{TxBndry-m,8}=cell2mat(Sensor(TxBndry+1-m,8))+t(TxBndry-m,1);
end
for m=TxBndry+2:26
    Sensor{m,8}=cell2mat(Sensor(m-1,8))+t(m,1);
end

% Compute Intensity reflection and transmission coefficients
z=cell2mat(Sensor(:,5))*1e6;    %acoustic impedance in Rayles

%Coefficient matrix format
%columns are [Reflect Coeff, phase, Transm Coeff]

coefdn=zeros(26,3); %downward propagating
coefup=zeros(26,3); %upward propagating

%Downward propagation
for n=1:25
    coefdn(n,:)=[((z(n+1)-z(n))/(z(n+1)+z(n)))^2 ...
                            1 ...
                            4*z(n+1)*z(n)/(z(n+1)+z(n))^2];
    %if signal phase reverses on reflection change phase sign
    if z(n+1)-z(n) < 0; coefdn(n,2)=-1; end;
```

```
end

%Upward propagation
coefup(2:26,:)=coefdn(1:25,:);
%reverse the reflection phases
coefup(:,2)=-coefup(:,2);

%Update Sensor cell array
Sensor(:,9)=num2cell(coefdn(:,1));
Sensor(:,10)=num2cell(coefup(:,1));

save Sensor.mat Sensor
set(h.Sensor,'Data',Sensor); %load into UITABLE
```

A2.4. CalcSenMod.m

CalcSenMod.m is the iterative calculation engine for SensorModel. It is only run when the user selects the 'Calculate' button in the user interface. The code starts with two propagating waves each with half of the starting acoustic intensity. The two starting waves propagate in opposite directions from the centre of the transmitting piezoelectric element. The code steps each propagating wave out to the next layer interface and computes the time, intensity, direction and phase for each reflected and transmitted wave. The incident wave is removed from the propagating wave list and the two new waves are added to the propagating wave list. If a propagating wave has less than the user selected minimum intensity it is removed from the propagating wave list. The process iterates until all propagating waves are removed from the list.

The code for CalcSenMod.m is shown below.

```
function [Sensor,TxWaveform,RxWaveform,TxEchoes,RxEchoes]=...
    CalcSenMod(Sensor,Limits,Excitation,coefdn,coefup,...
    TxBndry,RxBndry,h)
%_____________________________________________________________________
%Calculate Sensor Model
%---------------------------------------------------------------------
%Called by SensorModel.m function
%Tom Dakin
%
%This function calculates the echo waveform based on the setting in the GUI
%for the SensorModel.m function.
%TxBndry & RxBndry are the upper piezo layers
%---------------------------------------------------------------------
```

```
%Gaussian envelope equation uses mean=duration/M, stdev=duration/S
M=2.5;
S=7;

%load variables
tmax=Limits(1,1);    %max time
minAmpl=Limits(2,1); %minimum intensity threshold
maxiter=Limits(3,1); %maximum iterations, 30000 is good
Freq=Excitation(1,1);               %acoustic freq in Hz
NumCycles=Excitation(2,1);          %number of cycles in tone burst
Intensity=Excitation(3,1);          %pressure amplitude, relative
Fs=Excitation(4,1);                 %A/D sampling freq
Q=Excitation(7,1);                  %Q in cycles
d=cell2mat(Sensor(:,2));            %thicknesses
den=cell2mat(Sensor(:,3));          %densities in kg/m^3
c=cell2mat(Sensor(:,4));            %sound speed in m/s
z=cell2mat(Sensor(:,5))*1e6;        %acoustic impedance in Rayles
a_dB=cell2mat(Sensor(:,6));         %acoustic absorption in dB/cm
t=cell2mat(Sensor(:,7));            %travel time in s

%dB loss in propagating through layer
dB=a_dB.*d*100;  %a_dB/cm *thickness in m * 100 cm/m
%calculated power attenuation coefficient for propagation through each layer
PaC=10.^(-dB/10);

% Compute paths, signal leading edge time, intensity, phase
%Two matrices are required.
    %for storing propagating waves.
        %Waves=(time to next boundary, layer, direction, intensity,
        %       phase)
    %for storing detected echoes at the Tx boundary
        %TxEchoes=(leading edge time, intensity, phase)
    %for storing detected echoes at the Rx boundary
        %RxEchoes=(leading edge time, intensity, phase)

%preload with forward and backward propagating signals
Waves=zeros(2,5);
Waves(1,:)=[t(TxBndry) TxBndry -1 Intensity/2 1];   %upward transmit pulse
Waves(2,:)=[t(TxBndry) TxBndry+1 1 Intensity/2 1];  %downward transmit pulse

TxEchoes=zeros(20,3);
RxEchoes=zeros(20,3);

%Compute signal propagation
%initialize counters
WaveRows=2;
TxEchoRows=2;
RxEchoRows=0;
iteration=0;

%load transmit pulse, half energy in each direction
TxEchoes(1:2,:)=[0, Intensity/2, 1; ...
                 0, Intensity/2, 1];
if TxBndry==RxBndry
    RxEchoRows=2;
    RxEchoes(1:2,:)=[0, Intensity/2, 1; ...
                     0, Intensity/2, 1];
end
Calculation=[2;0;1];   %preload calculation with
                        %2 iterations
                        %0 time
```

```
                        %1 iteration

while iteration < maxiter && WaveRows > 0

    %skip min time search to speed up processing
    %just process row 1
    B=1;
    %if time is outside of limit then delete the row
    if Waves(B,1) > tmax;
        Waves(B,:)=[];
        WaveRows=WaveRows-1;
    else
        %Waves=(time to next boundary, layer, direction, ...
                %intensity at next boundary, phase)
        %TxEchoes=(leading edge time, intensity, phase)
        %RxEchoes=(leading edge time, intensity, phase)
        %coefxx=(Reflect Coeff, phase, Transm Coeff)

        %determine if the boundary is the Tx piezo centre.
        %If so, an echo output is required.
        if (Waves(B,2) == TxBndry && Waves(B,3) == 1) ||...
                (Waves(B,2) == TxBndry+1 && Waves(B,3) == -1)  %echo output required
            TxEchoRows = TxEchoRows+1;  %increment the number of echoes detected
            %add the echo specifications to the TxEchoes matrix
            TxEchoes(TxEchoRows,:)= ...
                [Waves(B,1), ... %time
                Waves(B,4), ...  %intensity
                Waves(B,5)]; ...  %phase
        end

        %determine if the boundary is the Rx piezo centre.
        %If so, an echo output is required.
        if (Waves(B,2) == RxBndry && Waves(B,3) == 1) ||...
                (Waves(B,2) == RxBndry+1 && Waves(B,3) == -1)  %echo output required
            RxEchoRows = RxEchoRows+1;  %increment the number of echoes detected
            %add the echo specifications to the RxEchoes matrix
            RxEchoes(RxEchoRows,:)= ...
                [Waves(B,1), ... %time
                Waves(B,4), ...  %intensity
                Waves(B,5)]; ...  %phase
        end

        %compute the reflected and transmitted waves at the boundary
        %address downward propagating waves first
        if Waves(B,3) == 1  %for downward propagating wave
            %compute reflection
            %if reflected wave intensity is strong calculate reflection
            %I.e. boundary reflection coef times the wave intensity is greater
            %than the min amplitude threshold.
            %Also, to reduce calculaton time, if the next boundary for the
            %reflected wave is greater than the max time limit then ignore the
            %ray
            if coefdn(Waves(B,2),1)*Waves(B,4) >= minAmpl &&...
                    Waves(B,1) + t(Waves(B,2)) < tmax
                %add the reflected wave to the waves list
                WaveRows=WaveRows+1;
                Waves(WaveRows,:)=[...
                    Waves(B,1) + t(Waves(B,2)); ...         %time to next boundary
                    Waves(B,2); ...                         %layer
                    Waves(B,3)*-1; ...                      %direction
                    %intensity at next boundary
                    coefdn(Waves(B,2),1)*Waves(B,4)*PaC(Waves(B,2)); ...
```

```
            coefdn(Waves(B,2),2)*Waves(B,5)]; ...    %phase
    end

    %compute transmission
    %if the downward transmitted wave is entering the bottom most layer
    %then ignore the transmitted wave.  I.e. assume the bottom layer is
    %infinite.  Also check to determine if the transmitted wave
    %intensity is above the intensity threshold.
    %Also, to reduce calculaton time, if the next boundary for the
    %transmitted wave is greater than the max time limit then ignore
    %the ray
    if Waves(B,2)<25 && coefdn(Waves(B,2),3)*Waves(B,4) >= minAmpl && ...
            Waves(B,1) + t(Waves(B,2)+Waves(B,3)) < tmax
        %add the transmitted wave to the waves list
        WaveRows=WaveRows+1;
        Waves(WaveRows,:)=[...
            Waves(B,1) + t(Waves(B,2)+Waves(B,3)); ...  %time to boundary
            Waves(B,2)+1; ...                           %layer
            Waves(B,3); ...                             %direction
            %intensity at next boundary
            coefdn(Waves(B,2),3)*Waves(B,4)*PaC(Waves(B,2)); ...
            Waves(B,5)]; ...                            %phase
    end
    %when the reflected and transmitted waves have been added delete the
    %incident wave from the waves list.
    Waves(B,:)=[];
    WaveRows=WaveRows-1;
    iteration=iteration + 1;

%Now address the upward propagating waves
else    %For upward propagating wave

    %Waves=(time to next boundary, layer, direction, ...
        %intensity at next boundary, phase)
    %Echoes=(leading edge time, intensity, phase)
    %coefxx=(Reflect Coeff, phase, Transm Coeff)

    %compute reflection
    %if reflected wave intensity is strong calculate reflection
    %I.e. boundary reflection coef times the wave intensity is greater
    %than the min amplitude threshold.
    %Also, to reduce calculaton time, if the next boundary for the
    %reflected wave is greater than the max time limit then ignore the
    %ray
    if coefup(Waves(B,2),1)*Waves(B,4) >= minAmpl && ...
            Waves(B,1) + t(Waves(B,2)) < tmax
        %add the reflected wave to the waves list
        WaveRows=WaveRows+1;
        Waves(WaveRows,:)=[...
            Waves(B,1) + t(Waves(B,2)); ...         %time to next boundary
            Waves(B,2); ...                         %layer
            Waves(B,3)*-1; ...                      %direction
            %intensity at next boundary
            coefup(Waves(B,2),1)*Waves(B,4)*PaC(Waves(B,2)); ...
            coefup(Waves(B,2),2)*Waves(B,5)]; ...   %phase
    end

    %compute transmission
    %if the upward transmitted wave is entering the top most layer
    %then ignore the transmitted wave.  I.e. assume the top layer is
    %infinite.  Also check to determine if the transmitted wave
    %intensity is above the intensity threshold.
```

```
            %Also, to reduce calculaton time, if the next boundary for the
            %transmitted wave is greater than the max time limit then ignore
            %the ray
            if Waves(B,2)>2 && coefup(Waves(B,2),3)*Waves(B,4) >= minAmpl && ...
                    Waves(B,1) + t(Waves(B,2)+Waves(B,3)) < tmax
                %add the transmitted wave to the waves list
                WaveRows=WaveRows+1;
                Waves(WaveRows,:)=[...
                    Waves(B,1) + t(Waves(B,2)+Waves(B,3)); ...  %time to boundary
                    Waves(B,2)+Waves(B,3); ...                  %layer
                    Waves(B,3); ...                             %direction
                    %intensity at next boundary
                    coefup(Waves(B,2),3)*Waves(B,4)*PaC(Waves(B,2)); ...
                    Waves(B,5)]; ...                            %phase
            end
            %when the reflected and transmitted waves have been added delete the
            %incident wave from the waves list.
            Waves(B,:)=[];   %delete the incident wave
            WaveRows=WaveRows-1;
            iteration=iteration + 1;
        end

        %display the time and number of wave iterations used
        if mod(iteration,1000)==0;
            Calculation=[iteration;WaveRows;max(Waves(:,4))];
            set(h.Calc,'Data',Calculation); %load into UITABLE
            drawnow;
        end
    end
end

%display the wave iterations used, waves remaining, and min time of waves
%remaining
Calculation=[iteration;WaveRows;max(Waves(:,4))];
set(h.Calc,'Data',Calculation); %load into UITABLE

if WaveRows==0;Calculation=[iteration;0;0];end    %if no rays are propagating then
                                                  %max time is reached.

% time
% set waveform time resolution
res=1/Fs;
PtsPerCycle=Fs/Freq;
% time axis
p=floor(tmax/res);  %number of time samples (round down)


% compute envelope of signal
duration=floor((NumCycles+3*Q)*PtsPerCycle); %number of samples in wave burst
EchoSample=1:duration;
envelope=(1-exp(-EchoSample/(Q/10*PtsPerCycle)));%envelope rise time
envelope2=exp(-(1:3*Q*PtsPerCycle)/(Q/10*PtsPerCycle));% envelope decay
envelope(NumCycles*PtsPerCycle+1:(NumCycles+3*Q)*PtsPerCycle)=envelope2;%full envelope


% compute the Tx waveform
TxWaveform=zeros(p+length(envelope),2);
                            %time in colum 1 pressure in column 2
                            %add an extra tone duration to the matrix to
                            %prevent over runs during the calculations
TxWaveform(1:p,1)=0:res:(p-1)*res;  %time axis
```

```
% Compute detected Tx waveform at the Tx boundary
for n=1:TxEchoRows
    %wave duration
    duration=floor((NumCycles+3*Q)*PtsPerCycle); %number of samples in wave burst
    start=ceil(TxEchoes(n,1)/res)+1;  %start time in terms of sample count.
                                      %Round up. Since no echo is present if
                                      %rounded down the first sample the echo
                                      %affects is the next one.  Add one
                                      %sample since the matrix starts at 1
                                      %not 0.
    shift=rem(TxEchoes(n,1),res);
                                  %this is the starting phase shift of the
                                  %echo at the first discrete sample point
                                  %of the waveform matrix (time)
    if shift~=0                       %if the remainder is not an integer
        duration=duration-1;          %then the last sample in the echo burst
    end                               %will be zero. So chop off last sample
                                      %for this echo
    shift=shift/PtsPerCycle*2*pi; %Convert the phase shift to radians

    for m=0:duration-1              %add echo contribution to each point in the
                                  %waveform burst.
                                  %wavform = waveform + echo
                                  % note that the echoes were computed as
                                  %intensities (rms power) The waveform
                                  %pressure output requires taking the  of
                                  %squareroot the (2*intensity*impedance)
                                  %times the phase.
        TxWaveform(start+m,2)=TxWaveform(start+m,2)+...
            TxEchoes(n,3)*sqrt(TxEchoes(n,2)*2*z(TxBndry))*... %convert to pressure
            sin(2*pi/PtsPerCycle*m+shift)*...   %sine wave
            envelope(m+1); % envelope modulation
    end

end

TxWaveform=TxWaveform(1:p,:);   %after the calculations truncate the waveform
                              %to tmax
save TxEchoes.mat TxEchoes  %save Echoes
    %Echoes=(leading edge time, intensity, phase, path)
    % this allows the user to view the paths of the various echoes

save TxWaveform.mat TxWaveform  %save TxWaveform
    %time in colum 1 pressure in column 2



%compute the Rx waveform
RxWaveform=zeros(p+length(envelope),2);;
                              %time in colum 1 pressure in column 2
                              %add an extra tone duration to the matrix to
                              %prevent over runs during the calculations
RxWaveform(1:p,1)=0:res:(p-1)*res;  %time axis

% Compute detected Rx waveform at the Rx boundary
for n=1:RxEchoRows
    %wave duration
    duration=floor((NumCycles+3*Q)*PtsPerCycle); %number of samples in wave burst
    start=ceil(RxEchoes(n,1)/res)+1;  %start time in terms of sample count.
                                      %Round up. Since no echo is present if
                                      %rounded down the first sample the echo
                                      %affects is the next one.  Add one
```

```
                                    %sample since the matrix starts at 1
                                    %not 0.
    shift=rem(RxEchoes(n,1),res);
                                %this is the starting phase shift of the
                                %echo at the first discrete sample point
                                %of the waveform matrix (time)
    if shift~=0                     %if the remainder is not an integer
        duration=duration-1;        %then the last sample in the echo burst
    end                             %will be zero. So chop off last sample
                                    %for this echo
    shift=shift/PtsPerCycle*2*pi; %Convert the phase shift to radians

    for m=0:duration-1            %add echo contribution to each point in the
                                %waveform burst.
                                %wavform = waveform + echo
                                % note that the echoes were computed as
                                %intensities (rms power) The waveform
                                %pressure output requires taking the  of
                                %squareroot the (2*intensity*impedance)
                                %times the phase.

        %RxEchoes=(leading edge time, intensity, phase)

        RxWaveform(start+m,2)=RxWaveform(start+m,2)+...
            RxEchoes(n,3)*sqrt(RxEchoes(n,2)*2*z(RxBndry))*...  %convert to pressure
            sin(2*pi/PtsPerCycle*m+shift)*...                   %sine wave
            envelope(m+1); % envelope modulation
            %Gaussian equation uses mean=duration/M, stdev=duration/S
            %for choice between square envelope and Gaussian envelope raise
            %the Gaussian equation to the power of Envelope-1.  I.e.
            %Gaussian^0=1  Gaussian^1=Gaussian

    end

end

RxWaveform=RxWaveform(1:p,:);   %after the calculations truncate the waveform
                            %to tmax
save RxEchoes.mat RxEchoes  %save Echoes
    %RxEchoes=(leading edge time, intensity, phase, path)
    % this allows the user to view the paths of the various echoes

save RxWaveform.mat RxWaveform  %save RxWaveform
    %time in colum 1 pressure in column 2
```

Appendix 3 NearField

NearField was developed by the author in MatLab to assess the diffraction effects of a transmitted acoustic wave in a sensor. NearField is based on Huygens-Fresnel principle (Fresnel, 1816) principle, to simulate the waveform as it approaches any given distance from the transducer. The model is a combined three-dimensional and two-dimensional model. The transmitter is modelled as a planar surface divided into small areas, with each surface radiating in three dimensions. The receiver is only modelled as a line array, to reduce computation time, which is divided into small segments. Each receiver segment sums the energy received at its location from all of the transmitter segments. This is used to calculate the acoustic pressures on a plane normal to the two transducers and parallel to the length axis of the transducers. The model can be run for any plane in the transducer width axis. One version of NearField.m allows the user to inject a sinusoidal wave and another version allows the user to inject an arbitrary wave such as the transmitted wave from SensorModel.

This appendix contains the MatLab code used in the sinusoidal NearField.m model. The code for NearField.m is shown below.

```
function NearField
%Compute in time domain
format compact
tic
%function to plot nearfield pressure waves for ISDS
%Calculate everything in time domain


%Define transmit burst
%Distance from transducer to leading edge of wave
%Define dimensions of transmit face
%Define dimensions of cross section plane to assess
%Resolution
Freq = 4;          % MHz
Cycles = 3;          % cycles transmitted
c = 1402;            % sound speed m/s
RequestedDist=230*2;    % mm
TxWidth = 0.05;      % mm
TxLength = 5;       % mm
```

```
RxWidth = 0.05;       % mm
RxLength = 5;        % mm

%Save to file
SaveFile = [num2str(Freq) 'MHz' ...
            num2str(Cycles) 'Cyc' ...
            num2str(RequestedDist) 'mm' ...
            num2str(TxLength) 'mmX' ...
            num2str(RxLength) 'mm' ...
            num2str(c) 'ss.mat'];

Freq = Freq*1000000;
RequestedDist=RequestedDist/1000; % m
TxWidth = TxWidth/1000;           % m
TxLength = TxLength/1000;         % m
RxWidth = RxWidth/1000;           % m
RxLength = RxLength/1000;         % m
CycleRes = 20;                  %divide each cycle into _ parts
Tresolution = 1/Freq/CycleRes;  %time resolution
Dresolution = c*Tresolution;    %distance resolution

%Recompute max distance to coincide with bin resolution
StopDistBin=floor(RequestedDist/Dresolution);   %distance in m
MaxDist=StopDistBin*Dresolution;

%time
t=MaxDist/c; %time in seconds to leading edge
BurstTime = Cycles/Freq; %burst duration in sec
tmin = t - BurstTime;   %time to trailing edge in sec

%RX width
RxWidthBinSize = ceil(RxWidth/Dresolution)+1;      %number of width bins
if (RxWidthBinSize/2 == ceil(RxWidthBinSize/2)); RxWidthBinSize = RxWidthBinSize+1;
end; %must be odd number
CtrBinRxWidth = ceil(RxWidthBinSize/2);  %centre bin for receive width
RxWidthDistBins = -(CtrBinRxWidth-1):1:(CtrBinRxWidth-1);
RxWidth = 2*(CtrBinRxWidth-1)*Dresolution; %Update Rx width to coincide with bin sizes

%RX length
RxLengthBinSize = ceil(RxLength/Dresolution)+1;    %number of length bins
if (RxLengthBinSize/2 == ceil(RxLengthBinSize/2));RxLengthBinSize = RxLengthBinSize+1;
end; %must be odd number
CtrBinRxLength = ceil(RxLengthBinSize/2);  %centre bin for receive length
RxLengthDistBins = -(CtrBinRxLength-1):1:(CtrBinRxLength-1);
RxLength = 2*(CtrBinRxLength-1)*Dresolution; %Update Rx width to coincide with bin
sizes

%TX width
TxWidthBinSize = ceil(TxWidth/Dresolution)+1;      %number of width bins
if (TxWidthBinSize/2 == ceil(TxWidthBinSize/2)); TxWidthBinSize = TxWidthBinSize+1;
end; %must be odd number
CtrBinTxWidth = ceil(TxWidthBinSize/2);  %centre bin for Tx width
TxWidthDistBins = -(CtrBinTxWidth-1):1:(CtrBinTxWidth-1);
TxWidth = 2*(CtrBinTxWidth-1)*Dresolution; %Update Tx width to coincide with bin sizes

%TX length
TxLengthBinSize = ceil(TxLength/Dresolution)+1;    %number of length bins
if (TxLengthBinSize/2 == ceil(TxLengthBinSize/2));TxLengthBinSize = TxLengthBinSize+1;
end; %must be odd number
CtrBinTxLength = ceil(TxLengthBinSize/2);  %centre bin for Tx length
TxLengthDistBins = -(CtrBinTxLength-1):1:(CtrBinTxLength-1);
```

```
TxLength = 2*(CtrBinTxLength-1)*Dresolution; %Update Tx length to coincide with bin
sizes

%Define Propagation lengths to assess
SRMax = c*(t); %slant range leading edge in m
SRMin = c*(tmin); %slant range trailing edge in m
SRBinsMax = SRMax/Dresolution; %Slant range leading edge in bins
SRBinsMin = SRMin/Dresolution; %Slant range trailing edge in bins

%calculate minimum distance from transducer for SRMin at widest Rx element
Wmax = (RxWidth+TxWidth)/2; %maximum width travel in m
Lmax = (RxLength+TxLength)/2; %maximum length travel in m
WmaxBins = ((CtrBinRxWidth-1)+(CtrBinTxWidth-1)); %maximum width travel in bins
LmaxBins = ((CtrBinRxLength-1)+(CtrBinTxLength-1)); %maximum length travel in bins
if SRBinsMin^2-WmaxBins^2-LmaxBins^2 <= 0 % bins minimum distance from transducer
    StartDistBin = 0; %0 bins from transducer
else
    StartDistBin = floor(sqrt(SRBinsMin^2-WmaxBins^2-LmaxBins^2));      %minimum bins
from transducer
end
MinDist = StartDistBin*Dresolution; %minimum distance in m

%distance range
DistBinSize = StopDistBin - StartDistBin+1;      %number of distance bins
Dist = MinDist:Dresolution:MaxDist; %Distance bins in m


%Tx elements
TxElements = TxLengthBinSize * TxWidthBinSize;

%calculate the total # of elements
Elements = DistBinSize * CtrBinRxLength;  %Number of elements to process
CalcLoops = Elements * TxWidthBinSize * TxLengthBinSize;  %number of calculation loops
disp(['RxWidth=' num2str(RxWidth) '  bins=' num2str(RxWidthBinSize)])
disp(['RxLength=' num2str(RxLength) '  bins=' num2str(RxLengthBinSize)])
disp(['TxWidth=' num2str(TxWidth) '  bins=' num2str(TxWidthBinSize)])
disp(['TxLength=' num2str(TxLength) '  bins=' num2str(TxLengthBinSize)])
disp(['Number of Tx elements = ' num2str(TxElements)])
disp(['Start Dist=' num2str(MinDist) '  Stop Dist=' num2str(MaxDist) '  bins='
num2str(DistBinSize)])
disp(['Number of elements = ' num2str(Elements)])
disp(['Number of calculation loops = ' num2str(CalcLoops)])

Pressure = zeros(CtrBinRxLength,DistBinSize); %dist,length

%calculate pressure for each bin
RxWidthBin = 0; % Choose Rx width row to calculate and plot pressure along
for DBin = DistBinSize:-1:1;
%for DBin = DistBinSize-1;    %Debug line
    for RxLengthBin = 0:CtrBinRxLength-1;   %only do 1/2 of the plane, the other half
is a mirror
%    for RxLengthBin = 0;   %Debug line
                        %1 is ctrline, CtrBinRxLength is Outside
        Press = 0;  %Reset Pressure accumulator to 0

        %compute sum of pressures for receive bin
        for TxLengthBin = -(CtrBinTxLength-1):(CtrBinTxLength-1); %integrate over
TxLength
%        for TxLengthBin = -(100):20:(100); %Debug line
           for TxWidthBin = -(CtrBinTxWidth-1):(CtrBinTxWidth-1); %integrate over
TxWidth
%            for TxWidthBin = -(5):(5); %Debug line
```

```
                VerticalDistBins = (DBin+StartDistBin-1);
                WidthDistBins = (RxWidthBin-TxWidthBin);
                LengthDistBins = (RxLengthBin-TxLengthBin);
                SlantRangeBins =
sqrt(VerticalDistBins^2+WidthDistBins^2+LengthDistBins^2);   %distance Tx element to
Rx element
                TimeSlantRange= SlantRangeBins*Tresolution;
                if SlantRangeBins > SRBinsMax
                    P=0;
                elseif SlantRangeBins < SRBinsMin
                    P=0;
                else
                    CosPhi = VerticalDistBins/SlantRangeBins;
                    P= CosPhi*sin(2*pi*Freq*(t-TimeSlantRange));
                    Press= P + Press;
                end
            end
        end
        Pressure(RxLengthBin+1,DBin) = Press;    % pressure at spacial bin
    end
    PercentComplete = (DistBinSize-DBin+1)/(DistBinSize);
    disp([num2str(round(toc)) ' seconds  ' ...
          num2str(round(1000*PercentComplete/10)) ' % complete  ' ...
          num2str((round(toc*10/PercentComplete*(1-PercentComplete)/3600))/10) ' of '
...
          num2str((round(toc*10/PercentComplete/3600))/10) ' hours remaining'])
end
WidePressure(1:CtrBinRxLength,:) = flipud(Pressure);
WidePressure(CtrBinRxLength:2*CtrBinRxLength-1,:) = Pressure;
WidePressure = WidePressure/TxElements; %Normalize to number of Tx elements
save(SaveFile);

figure1 = figure(1);
figure1.Color='w';
clf(figure1);
colormap('jet');
axes1 = axes('Parent',figure1);
view(axes1,[-90 90]); %for plan view
grid(axes1,'on');
hold(axes1,'all');
% Create surf
surf(Dist,-Dresolution*(CtrBinRxLength-1):Dresolution:Dresolution*(CtrBinRxLength-
1),WidePressure,'EdgeColor','none');
ylabel('Length [m]');
xlabel('Distance [m]');
zlabel('Amplitude');
colorbar('peer',axes1);
hold off

figure2 = figure(2);
figure2.Color='w';
clf(figure2);
colormap('jet');
axes2 = axes('Parent',figure2);
%view(axes2,[-15 2]); %for 3D
view(axes2,[0 0]); %for 3Dgrid(axes2,'on');
hold(axes2,'all');
% Create surf
surf(Dist,-Dresolution*(CtrBinRxLength-1):Dresolution:Dresolution*(CtrBinRxLength-
1),WidePressure,'EdgeColor','none');
grid(axes2,'on');
ylabel('Length [m]');
```

```
xlabel('Distance [m]');
zlabel('Amplitude');
colorbar('peer',axes2);
hold off
```