DEVELOPMENT OF A MEASUREMENT SYSTEM FOR QUANTIFYING ULTRASOUND FIELD DISTRIBUTION OF DIFFERENT TRANSDUCERS

by

JAYANTH KANDUKURI

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN BIOENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

Summer 2012

To my Sister

ACKNOWLEDGEMENTS

I am eternally thankful to my advisor, Professor Baohong Yuan for his continued support and contribution in the development of my research. His honest nature, expert insight and precise guidance has helped me to be perceptive, diligent and made successful completion of my work possible. I would also like to express my gratitude to my thesis committee members: Prof. Kambiz Alavi and Prof. Kytai Nguyen for their patience and support. I would also like to express special gratitude to Prof. Hanli Liu for her encouragement and guidance.

I owe my deepest gratitude to my fellow colleague Yuan Liu who has supported and encouraged me throughout my thesis. I would like to thank Kambiz Alavi for his support in providing IR camera for undertaking few of the HIFU and other related experiments. I would like to especially acknowledge Kambiz Alavi's research team member Mohammadreza Jahangir Moghadam for his exceptional support. I also would like to thank Kytai Nguyen for her support in providing micro-bubbles and thermo-sensitive polymer for investigate application of HIFU induced heat. I also acknowledge Aniket Wadajkar and his team for their support.

I would like to thank my parents Madhusudan and Bharathi, my sister Renukarthika and brother-in-law Sivakanth for their constant encouragement and support without which this work would have not been possible. I would like to convey thanks to all of my UT Arlington friends Kiranmayi, Somdutta, Rahul, Amruta, Bipin, Ankita, Raja, Jagath, Vishal and Warren.

Finally, but really foremost, I thank the Lord, for the strength, intelligence, perseverance and wisdom He has given me to realize my dreams and perform this work.

July 19, 2012.

ABSTRACT

DEVELOPMENT OF A MEASUREMENT SYSTEM FOR QUANTIFYING ULTRASOUND FIELD DISTRIBUTION OF DIFFERENT TRANSDUCERS

JAYANTH KANDUKURI, MS

The University of Texas at Arlington, 2012

Supervising Professor: Baohong Yuan.

Ultrasound has been widely used in many fields, such as nondestructive examination and medical imaging. Ultrasound is mostly characterized by its pressure and, dynamic pressure wave can be considered to be the foundation of ultrasound physics and imaging techniques. it is thus important for understanding ultrasound pressure properties to that of ultrasound intensity. Moreover, there are many applications that require mainly characterizing ultrasound pressure field (instead of intensity field alone). Therefore, reconstruction of an ultrasound pressure field is highly desirable not only for education but also for research. There may be some commercially available systems that can be used for 2- or 3-D reconstruction of an ultrasound pressure field based on a needle-shaped hydrophone such as AIMS system from Onda Inc. However, they are usually very expensive (>tens of thousands US dollars) and unaffordable for many educators, professionals and researchers (even for those researchers whose main research streams are not ultrasound but need to measure a dynamic pressure field for some reasons). Therefore, in this study, we developed a cost efficient and multi-functional ultrasonic system that can be used to reconstruct 2- or 3-D dynamic variation of ultrasound pressure waves based on either a needle-shaped hydrophone or a small diameter wire. In addition, this system can be used to conduct

ultrasound imaging, and study microbubble-enhanced ultrasound imaging. Although the principle of the developed system is straightforward, engineering challenges exist in the development of both hardware and software. Therefore, we would like to share our design, experience and results to the community to make the system affordable and provide multi-functional system for research and education.

Second part of study focuses around understanding the working of High intensity focused ultrasound (HIFU) that is used for delivering high pressure thereby increaseing the localize temperature. Many investigations have been conducted in the past to understand thermal effects of HIFU procedures using many thermal sensitive sensors which are used to characterize the temperature distribution at its focus. The most prominent and reliable was the measurement obtained using fine wire thermocouple. In our study we aim at using very short duration excitation of HIFU and study the thermal distribution by varying combination of both power and duration of each burst using one such fine wire thermocouple. We were also able to verify the focal size and focal volume of the HIFU. Cavitation and viscous heating artifacts that predominantly affect the long duration HIFU investigations have been shown to be absent. Results and discussions have been presented to support our hypothesis.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

LIST OF TABLES

Table Page

CHAPTER 1

Ultrasound

1.1 Motivation

Ultrasound (high frequency sound) has been widely used in many fields, such as nondestructive examination and medical imaging . An ultrasound wave represents a mechanical pressure wave that propagates in a medium. The behavior of an ultrasound wave has been well studied and known to be similar to an optical wave [1]. It can be well focused using an acoustic lens or a curved transducer. Also, it can be reflected and scattered due to acoustic impendence mismatching between different media in which it propagates. Importantly, a short ultrasound pulse can be used to noninvasively image the acoustic mismatch of a medium (especially biological tissues) based on a pulse-echo technique (or a time-of-flight technique), which is the foundation of modern ultrasonography for both medical or nonmedical applications [1]. More interestingly, it has been found that a ultrasound (in MHz range) wave can resonate with a microbubble to generate linear or nonlinear acoustic signal for enhancement of ultrasound imaging, which has been intensively studied during the past decades [1]. Microbubbles are commonly used as an ultrasound imaging contrast agents. They are very small bubbles (diameter in order of microns) with a gas core shelled with either a lipid, protein or polymer layer. Generally, they are smaller than ten microns in diameter and the average size is around 2-3 microns to be able to pass tissue capillaries. They are usually used to enhance tissue ultrasound imaging, such as tissue perfusion imaging, blood flow imaging, molecular imaging and drug delivery [1].

Another interesting application of ultrasound is that it can be used to attain higher degrees of temperature non-invasively which could be used for therapeutic purposes such as tissue ablation and hemostasis. This application gained popularity over past decade and came to be known as High intensity focused ultrasound [2]. Hence, the use of ultrasound in clinical practice is no longer limited to diagnostic imaging or to simple needle guidance in the performance of percutaneous procedures such as amniocentesis or tumor biopsy. High-intensity focused ultrasound (HIFU) is being not only promoted as a noninvasive method to treat certain primary solid tumors and metastatic disease but also to ablate foci of ectopic electrical activity in the heart, and to achieve homeostasis in acute traumatic injuries to the extremities and visceral organs [2]. Therefore, developing a simple, cost efficient and multi-functional system for demonstrating these unique features of ultrasound physics and techniques is significantly useful for students, scientific educators, clinical professionals and even general public who are either interested in ultrasound but with limited ultrasound knowledge or newly entering the field. It is also a valuable tool for scientific researchers who are looking for a tool that can quantify the static or dynamic properties of an ultrasound pressure field but within a limited budget to be able to afford an expensive system.

Several techniques have been widely developed to measure ultrasound intensity distribution in a medium [3-5]. The Schlieren technique is one of them, which is based on the phenomenon of light diffraction by ultrasound waves in a transparent medium [3]. Ultrasound intensity is reconstructed based on the detected optical signals. Therefore, it is much faster than mechanical methods that involve mechanical scanning [4, 5]. A few commercial systems have adopted Schlieren technique, such as OptiSon® ultrasound beam analyzer (from Onda Inc) [6]. Unfortunately, these systems are extremely expensive (>$150K) because of that they are most likely used for ultrasound transducer manufacturers to characterize their products. In addition, what this technique measures is the static or dynamic distribution of ultrasound intensity, instead of the pressure. One may think that the intensity is related to the square of pressure. Therefore, if

one can measure the pressure distribution, the intensity distribution can be calculated. However, it is not true on the other hand, which means one cannot extract the pressure distribution based on the measured intensity distribution because an ultrasound pressure field includes more information (both amplitude and phase) than an ultrasound intensity field (only amplitude information). Therefore, to measure the dynamic propagation of an ultrasound pressure wave, other techniques have to be adopted.

There may be some commercially available systems that can be used for 2- or 3-D reconstruction of an ultrasound pressure field based on a needle-shaped hydrophone (such as AIMS system from Onda Inc.) [7]. However, they are usually very expensive (>tens of thousands US dollars) and unaffordable for many educators, professionals and researchers (even for those researchers whose main research streams are not ultrasound but need to measure a dynamic pressure field for some reasons). Therefore, in this study, we developed a cost efficient and multi-functional ultrasonic system that can be used to reconstruct 2- or 3-D dynamic variation of ultrasound pressure waves based on either a needle-shaped hydrophone or a small diameter wire. In addition, this system can be used to conduct ultrasound imaging, study microbubble-enhanced ultrasound imaging and also able to quantify pressure and temperature profiles of the High intensity focused ultrasound filed. Although the principle of the developed system is straightforward, engineering challenges lie in the development of both hardware and software. Therefore, we would like to share our design, experience and results with the community to make the system affordable and multi-functions for research and education.

1.2 History of Ultrasound

Ultrasound propagates in a given medium as a pressure wave caused by mechanical disturbance whose vibrations are in ultrasonic frequency range. The major attribute of ultrasound is that it does not alter the properties of the medium it travels but changes its own characteristics while in motion which is used by many industrial or medical devices to reconstruct and visualize the interior geometry or structure. A lot of research has been undertaken to understand intricate

characteristics of ultrasound waves and their behavior in various media which made it possible to use ultrasound as a diagnostic tool in clinical medicine.

Ultrasound came into existence when French physicists Pierre and Jacques Curie discovered the piezoelectric effect in 1880 [8]. Later French physicist, Paul Langevin, around the year 1916 attempted to develop piezoelectric materials to generate/receive ultrasonic waves through a material and with the help of another physicist, Constantin Chilowski, developed first ultrasonic submarine detector which employed a new technique that is used in navigation and ranging called SONAR [9, 10]. Later, pioneering works done by Dr. Karl Theodore Dussik in Austria in 1942 on transmission ultrasound investigation of the brain [11] and by Professor Ian Donald and his colleagues in Glasgow in 1958 on pulsed ultrasound Investigation of Abdominal Masses [12] facilitated the development of ultrasound for practical technology and applications, and subsequent decades lead to the wider use of ultrasound in medical practice.

1.3 Mechanism of Sound and its characteristics:

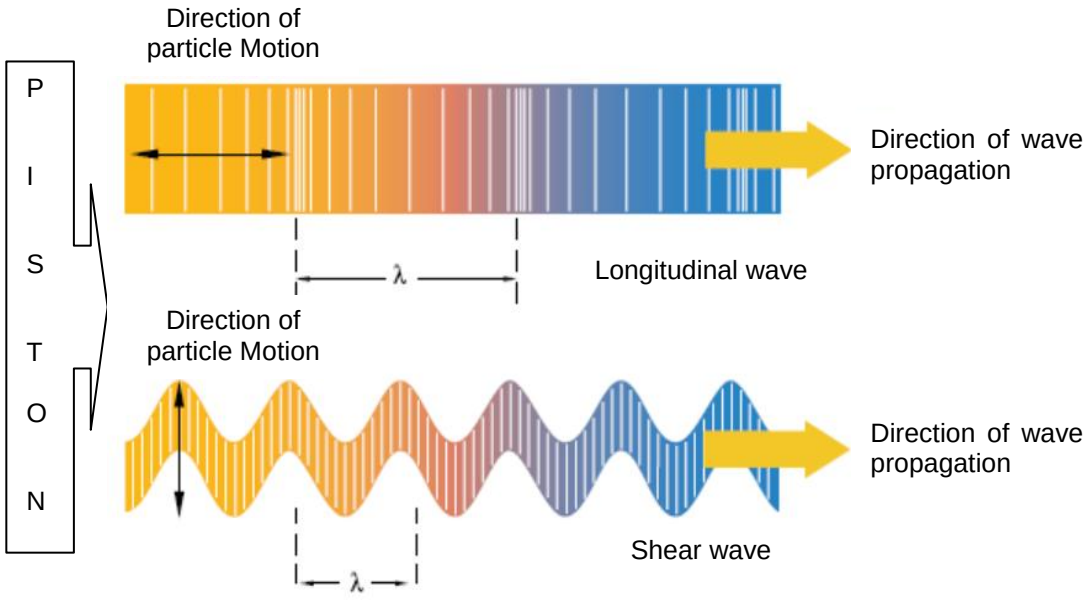

Figure 1.1 Production of an ultrasonic wave [13].

A medium is a collection of molecules that are in a state of continuous random motion and are spaced out more or less uniformly under no external force applied to the medium. External force when applied to the medium such as movement of the piston tends to create an increased pressure due to an increase in the concentration of molecules in front of the piston creating a mechanical disturbance known as compression which migrates away from the source of disturbance through the medium. Upon withdrawal of force a region of reduced pressure is created immediately behind compression zone known as rarefaction. The surrounding molecules tend to occupy this zone in order to restore normal particle density thereby forcing rarefaction zone to migrate away from the source of disturbance accompanying compression. Oscillations of piston thereafter would create pairs of these compressions and rarefaction during which the molecules of the medium vibrate over very short distances in a direction parallel to the longitudinal wave. It is, during this vibration, momentum is transferred among molecules, thereby causing the wave to move through the medium. This is considered as wave disturbance or wave

cycle in the medium termed as longitudinal wave. Frequency of wave cycle would determine the nature of the sound propagation inside the medium [13]. Frequency between 20 to 20,000Hz is audible whereas frequency above 20,000Hz is inaudible to human ear and is known as Ultrasound. A wave cycle can be represented as a graph of local pressure, particle density, in the medium versus distance in the direction of the ultrasound wave propagation as shown in Fig. 1.1. The characteristic of a sound wave cycle can be described by following attributes shown in the table 1.1 below:

Table 1.1 Characteristics of Sound

Wavelength (λ)	The distance covered by one wave cycle.
Frequency (υ)	The number of cycles focused into the medium per second.
Amplitude	The maximum height of the wave cycle
Velocity (c)	The product of the frequency (υ) and the wavelength (λ)

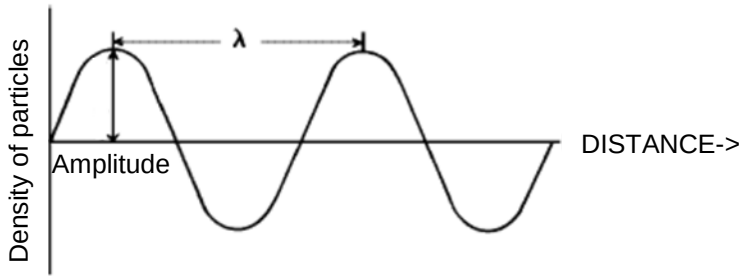

Figure 1.2 Characteristics of an ultrasound wave

When two waves meet, they are said to “interfere” with each other and they are two kinds of interference. When the waves are ‘in phase’, i.e., positive peak meets positive peak and the amplitude add up, thereby undergo constructive interference and on the other hand if the waves are ‘out of phase’, i.e., positive peak meets negative peak thereby undergo destructive

interference. This is considered as an important phenomenon in diagnostic medical ultrasound tools where altering the activation of array of transducers bring about controlled constructive interference at the region of interest (ROI) and the remaining undesired region undergo destructive interference thereby, receiving information only from the desired region and reconstruction of ROI would help to diagnose the underlying abnormality . This technique is known as beam forming. These beams are generally focused onto a small area and as the ultrasound wave passes through a medium it transports energy through the medium. The rate of this energy transport is known as power and the power delivered power per unit area is termed as Intensity (I). Intensity is measured using a logarithmic scale or dB scale given as

$$dB=10\log\frac{I}{I_O}$$

, where I_0 is the reference intensity.

Ultrasound maximum pressure (Pm) can be related to its wave intensity in the medium as [14]:

$$\mathbf{I}=\frac{\mathrm{Pm}^{\mathbf{2}}}{\mathbf{2\rho c}}$$

Where ρ is the density of the medium in grams per cubic centimeter and c is the speed of sound in the medium. Upon substitution the dB equation can be rewritten as:

$$\mathrm{dB}=10.\log\frac{\mathrm{P_m}}{\mathrm{P_{m_o}}}$$

The above equation can give a measure of comparison of two or more waves. An ultrasound transducer converts pressure amplitudes received from the patient (i.e., the reflected ultrasound wave) into voltages. The amplitude of voltages recorded for ultrasound waves is directly proportional to the variations in pressure in the reflected wave. It is normal practice to find the ‘half-power value’, i.e., ratio of 0.5 in power between two waves, which is –3 dB, but should not be confused with the ‘half-amplitude value’, i.e., ratio of 0.5 in amplitude , which is –6 dB. These half power points encompass a range of frequencies termed as the bandwidth of the transducer.

This difference reflects the greater sensitivity of the decibel scale to amplitude when compared with intensity values [13].

Ultrasound is said to get attenuated while travelling through a medium. The attenuation is caused by absorption, scattering and reflection of the beam while travelling. which not only depends on the characteristics of the ultrasound travelling through the medium but also on the medium inherent characteristics, the following table 1.2 shows mutual dependency of both these characteristics:

Table 1.2 Mutual dependency of medium and sound characteristics

If particles size is large compared to wavelength of the sound	• Integrity of ultrasound beam is retained. • Direction of propagation changes. • Part of the sound beam gets reflected. • Rest is transmitted through the particle.
If particle size is comparable or smaller to wavelength of the sound	• Ultrasound energy gets scattered.

Primary mechanism of ultrasound energy dissipation in a medium is by the relaxation processes which involve (a) removal of energy from the ultrasound beam and (b) eventual dissipation of this energy primarily as heat. This displacement requires energy and as the molecules attain maximum displacement from an equilibrium position, their motion stops and their energy is transformed from kinetic energy associated with motion to potential energy associated with position in the compression zone. From this position, the molecules begin to move in the opposite direction, and potential energy is gradually transformed into kinetic energy. The maximum kinetic energy (i.e., the highest molecular velocity) is achieved when the molecules pass through their original equilibrium position, and where the displacement and potential energy are zero. If the kinetic energy of the molecule at this position equals the energy absorbed

originally from the ultrasound beam, then no dissipation of energy occurs, and the medium is an ideal transmitter of ultrasound. Actually, the conversion of kinetic to potential energy (and vice versa) is always accompanied by some dissipation of energy. Therefore, the energy of the ultrasound beam is gradually reduced as it passes through the medium. This reduction is termed relaxation energy loss. The rate at which the beam energy decreases is a reflection of the attenuation properties of the medium. Most of the materials have their own intrinsic attenuation or absorption coefficient. However, complicated structures such as tissue samples often exhibit a rather complex attenuation pattern for different frequencies, which probably reflects the existence of a variety of relaxation frequencies and other molecular energy absorption processes that are poorly understood at present [13]. These complex attenuation patterns concerning various tissues of human body are reflected in the Fig. 1.3 shown below,

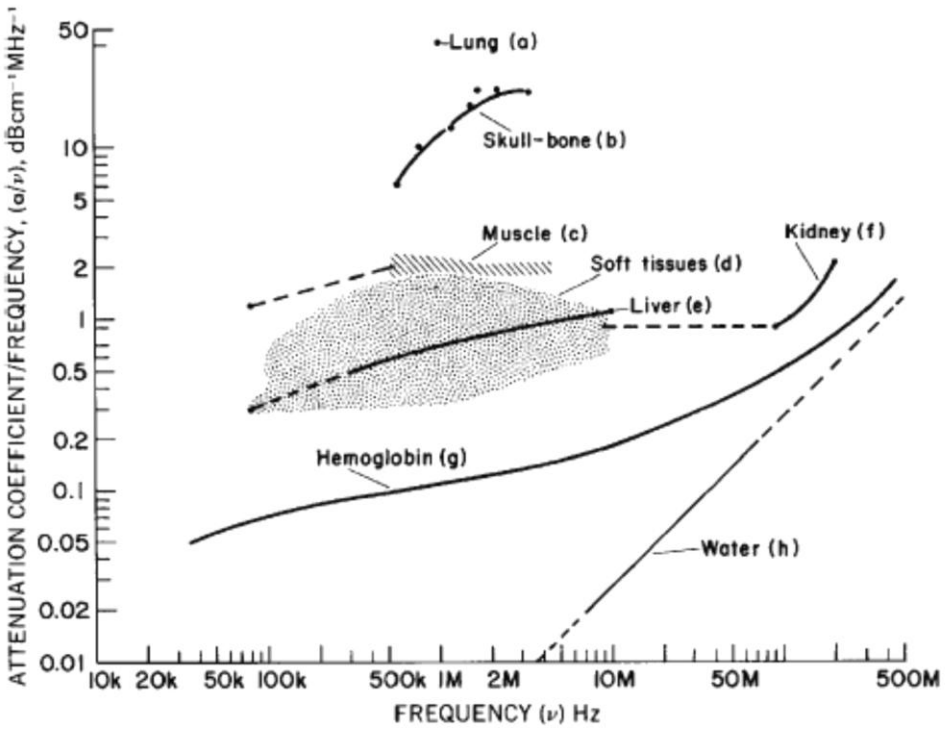

Figure 1.3 Ultrasound attenuation coefficient as a function of frequency for various tissue samples [13].

The most desired phenomenon of ultrasound that has made ultrasound medical devices to be used in diagnostic applications is because of its inherent property to get reflected from interfaces between different tissues in the patient. This is possible due to the difference in acoustic impedance of the media through which the ultrasound propagates; a fraction of the ultrasound energy that gets reflected from this interface is termed as an echo. The acoustic impedance (Z) of a medium is the product of the density (ρ) of the medium and the velocity (c) of ultrasound in the medium. For an ultrasound wave incident perpendicularly upon an interface, the fraction of the incident energy that is reflected, denoted by the reflection coefficient α_R, is given by the equation:

$$\alpha_R = \left(\frac{Z_2 - Z_1}{Z_2 + Z_1}\right)^2$$

Where $Z_1$and Z_2 are the acoustic impedances of the two media. The fraction of the incident energy that is transmitted across an interface is described by the transmission coefficient α_T and is given by equation,

$$\alpha_T = \frac{4Z_1Z_2}{(Z_2 + Z_1)^2}$$

Thus,

$$\alpha_R + \alpha_T = 1$$

Hence, it can be inferred from the above coefficients that larger the acoustic impedance difference greater will be the reflection from that interface. The above discussion is valid only if the ultrasound beam that is directed into the medium is orthogonal to the medium's surface. Sound also follows the principles of light so, likewise most of the energy would be reflected from the first surface if the transducer is placed at an angle other than right angle to the surface of the medium.

Refraction is another phenomenon accompanying reflection and is a principal cause of artifacts in clinical ultrasound images. Assuming that an ultrasound beam is incident at steeper

angle through a medium surrounded by air or medium 1 then the beam emerges from medium 2 and reenters medium 1, thereby resuming its original direction of motion. the presence of medium 2 simply displaces the ultrasound beam laterally for a distance. This displacement depends upon the difference in ultrasound velocity, density in the two media and upon the thickness of medium 2. Suppose a small structure below medium 2 is visualized by reflected ultrasound, the position of the structure would appear to be an extension of the original direction of the ultrasound through medium 1 which adds spatial distortion and resolution loss to ultrasound images [13].

1.4 Ultrasound Transducers

Any device that converts one form of energy to another form of energy is considered as a transducer. As such an ultrasound transducer converts electrical energy into ultrasound pressure energy and the inverse also holds. The functional component of the ultrasound transducer consists of one or more piezoelectric crystals or elements and single element ultrasound transducers (UST), having only single piezoelectric crystal, is used in our present study whereas ultrasound medical devices use multiple element (known as arrays) of piezoelectric crystal for diagnostic and therapeutic applications [13].

The piezoelectric crystals are special type of crystals that exhibit an inherent effect known as piezoelectric effect, where a voltage is developed across the crystal when it experiences an external pressure and vice versa, i.e., external voltage applied across the crystal will deform the crystal thereby generating compression and rarefaction as discussed in earlier section [15, 16]. The magnitude of voltage developed is proportional to the pressure applied and this voltage is recorded as an electrical signal. The efficiency of the transducer is measured as the fraction of the applied energy that is converted to the desired energy mode. A crystal exhibits its greatest response at the resonance frequency which is determined by the thickness of the crystal (the dimension of the crystal along the axis of the ultrasound beam). The most efficient operation is achieved for a crystal with a thickness equal to half the wavelength of the desired ultrasound. Thin crystals yield high resonance frequencies, and vice versa. The faces of the piezoelectric

crystal are coated with a thin conducting film and then electric contacts are applied to establish an electrical contact. The crystal is mounted at one end of a hollow metal or metal-lined plastic cylinder, with the front face of the crystal coated with a protective plastic that provides efficient transfer of sound between the crystal and the body. The plastic coating at the face of the crystal has a thickness of 1/4¸ and is called a quarter-wavelength matching layer. This particular thickness maximizes energy transfer out of the transducer and usually only a single one-quarter wavelength thickness is used for the matching layer. The front face of the crystal is connected through the cylinder to ground potential. The remainder of the crystal is electrically and acoustically insulated from the cylinder. Most ultrasound imaging applications utilize short pulses of ultrasound for which suppression of ultrasound reverberation in the transducer is desirable. Suppression or “damping” of reverberation is accomplished by filling the transducer cylinder with a backing material such as tungsten powder embedded in epoxy resin. Sometimes, rubber is added to the backing to increase the absorption of ultrasound. Often, the rear surface of the backing material is sloped to prevent direct reflection of ultrasound pulses back to the crystal. The Fig.1.4 below illustrates the construction of a typical ultrasound transducer with a flat crystal. The crystal with curved surface is used to focus the ultrasound beam.

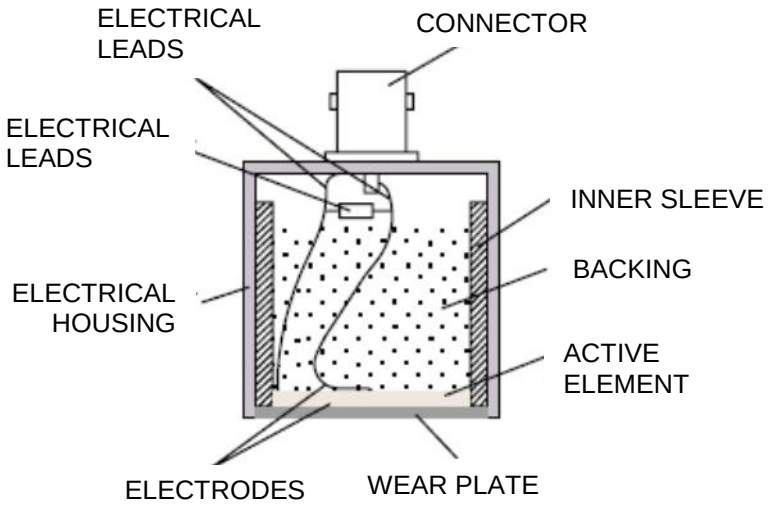

Figure 1.4 Typical ultrasound transducer [17].

Coupling of the transducer to the medium ensure maximum efficiency of energy transfer into the medium and vice versa. If the acoustic impedance of the coupling medium is not too different from that of either the transducer or the medium and if the thickness of the coupling medium is much less than the ultrasound wavelength, then the ultrasound is transmitted into the medium with little energy loss. This is achieved only when the impedance of the coupling medium is intermediate between the impedances of the crystal and the medium. The ideal impedance of the coupling medium is

$$Z_{\text{coupling medium}} = \sqrt{Z_{\text{transducer}} X Z_{\text{medium}}}$$

Two methods are commonly used to generate ultrasound beams. For continuous wave beams, an oscillating voltage is applied with a frequency equal to that desired for the ultrasound beam and for a similar voltage of prescribed duration is used to generate long pulses of ultrasound energy, as shown in the Fig. 1.5 (a) below. For clinical ultrasound imaging, short pulses are used to shock the crystal into mechanical oscillation by a momentary change in the voltage across the crystal. The oscillation is damped quickly by the methods described earlier to furnish ultrasound pulses as short as half a cycle. The duration of a pulse usually is defined as the number of half cycles in the pulse with amplitude greater than one fourth of peak amplitude. The effectiveness of damping is described by the pulse dynamic range, defined as the ratio of the peak amplitude of the pulse divided by the amplitude of ripples following the pulse. A typical ultrasound pulse of short duration is illustrated in the Fig. 1.5 (b).

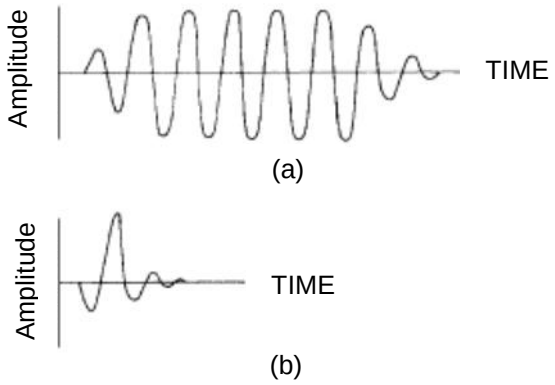

Figure 1.5 (a) Long ultrasound pulse and (b) Short ultrasound pulse

The compression zones as described in the earlier section of an ultrasound wave can be imagined to be lines perpendicular to the motion of the ultrasound wave in the medium. These lines are referred to as wave fronts. For an ultrasound source of large dimensions, the wave fronts are termed as planar wave fronts and the ultrasound wave itself is considered to be a planar or plane wave. On the contrary, ultrasound emerging form a very small dimension or point source exhibits spherical wave fronts. Sources with finite dimension are usually considered for research and diagnostic tools and are considered to have collection of point sources hence, many spherical wavelets radiating from a transducer of reasonable size, with the diameter of the transducer considerably larger than the ultrasound wavelength, can be imagined. These spherical wavelets generate many regions of constructive and destructive interferences in the medium and are progressively less dramatic with increasing distance from the ultrasound source. The region near the source where the interference of wavelets is most apparent is termed the Fresnel (or near) zone. For a disk-shaped transducer of radius r, the length $D_{Fresnel}$ of the Fresnel zone is given by equation,

$$D_{Fresnel} = \frac{r^2}{\lambda}$$

, Where λ is the ultrasound wavelength Within the Fresnel zone, most of the ultrasound energy is confined to a beam width no greater than the transducer diameter. Beyond the Fresnel zone, some of the energy escapes along the periphery of the beam to produce a gradual divergence of the ultrasound beam that is described by the equation,

$$\text{Sin}\,\theta = 0.6\left(\frac{\lambda}{r}\right)$$

, where θ is the Fraunhofer divergence angle in degrees. The region beyond the Fresnel zone is termed the Fraunhofer (or far) zone. The Fig. 1.6 depicts these two regions as shown below,

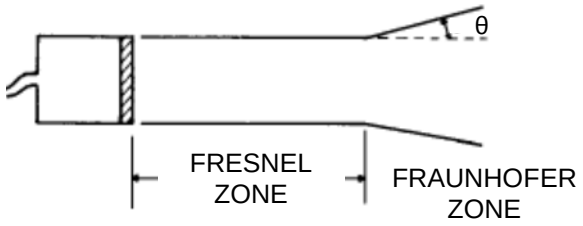

Figure 1.6 Ultrasound field of a unfocused transduccer

Ultrasound transducer rules:

- The length of the near field increases with increasing transducer diameter and frequency and
- Divergence in the far field decreases with increasing transducer diameter and frequency.

Usually beams with little lateral dispersion of energy (i.e., long Fresnel zones) are preferred. Hence, a reasonably high ratio of transducer radius to wavelength (r/λ) is desired. This requirement can be satisfied by using ultrasound of short wavelengths (i.e., high frequencies). However, the absorption of ultrasound energy increases at higher frequencies, and in general frequencies used for clinical imaging are limited to 2 to 20 MHz.

1.4.1 Focused Transducer:

A focused ultrasound transducer generates a beam profile that is narrower (focal zone) at a specific distance from the transducer face compared to its dimension at the face of the transducer [13, 18, 19]. The ultrasound intensity in the focal zone can be considered to be 100 times or more compared with the intensity outside of the focal zone due to which a larger signal will be induced in a transducer from a reflector positioned in the focal zone. The distance between the location for maximum echo in the focal zone and the piezoelectric element responsible for focusing the ultrasound beam would be considered as the focal length of the transducer. The piezoelectric crystal used for focusing applications is shaped like a concave disk. For an ultrasound beam with a circular cross section, focusing characteristics such as pulse-echo response width and relative sensitivity along the beam axis depend on the wavelength of the ultrasound and on the focal length f and radius r (D=2r) of the transducer's focusing element. These variables may be used to distinguish the degree of focusing of transducers by dividing the near field length (r^2/λ) by the focal length (f). The length of the focal zone of a particular ultrasound beam is the distance over which a reasonable focus and pulse-echo response are obtained and can be estimated by the following equation,

$$\text{Focal zone length} = 10\lambda\left(\frac{f}{D}\right)^2$$

1.5 Ultrasound Modes of operation

1.5.1 A-Mode

A-mode presentation of ultrasound images involves observing the amplitude of the echoes emerging from the acoustic mismatch interfaces from a target sample or tissue as a function of time, termed as A-line. Assuming that the speed of sound is known for a given medium the distance from the ultrasound transducers can be measurable. The main aim of A-mode imaging is to reveal the location of echo-producing structures but only in the direction of the ultrasound beam. Some of the applications of A-mode involve localization echo-producing interfaces such as midline structures in the brain (known as echoencephalography) and also used in structures to be

imaged in B-mode. The concept of A-mode is useful in explaining how pixels are obtained from scan lines in B-mode imaging [13].

1.5.2 B-Mode

B-mode presentation of pulse echo images is generated by viewing A-lines, obtained from the location of echo-producing interfaces, as a function of different location, with respect to direction of ultrasound propagation, and is displayed in two dimensions (x and y) usually on a video screen in a Ultrasound medical device. The amplitude of each echo is replaced by the corresponding brightness value at the respective co-ordinate location along XY plane. Thus high-amplitude echoes can be presented as either high brightness or low brightness to provide either “white-on-black” or “black-on-white” presentations. Most images are viewed as white on black, so regions in the patient that are more echogenicity correspond to regions in the image that are brighter (hence B for “brightness” mode).

Advantage of B-mode images is that it can be displayed as either “static” or “real-time” images. In static imaging the image is compiled as the sound beam is scanned across the patient, and the image presented is averaged over the time required to sweep the sound beam. In real-time imaging, the image is also built up, as the sound beam scans across the patient. Moreover, the scanning is performed automatically and quickly, as one image follows another in quick succession. Image frequencies that are typically greater than approximately 24 or 48 per second the motion of moving structures can be viewed in continuous motion. Real-time B-mode images are useful in the display of moving structures such as heart valves. They also permit the technologist to scan through the anatomy to locate structures of interest. Certain applications such as sequential slice imaging of organs are performed better with static imaging [13].

1.5.3 M-Mode

M-mode presentation of ultrasound images is designed specifically to depict moving structures where the position of each echo-producing interface is presented as a function of time. The most frequent application of M-mode scanning is echocardiography, where the motion of

various interfaces in the heart is depicted graphically on a cathode-ray tube (CRT) display or chart recording [13].

1.6 Spatial resolution of the Ultrasound Imaging

Spatial resolution for ultrasound transducer can be classified into axial resolution and lateral resolution. Axial resolution is the ability of the transducer or imaging system to recognize or distinguish two different regions or objects located at slightly different depths from the transducer along the axis of the ultrasound beam propagation. Thus it is given by,

$$\text{Axial resolution} = \frac{\lambda \,.\, (\#\text{ of cycles in one pulse})}{2}$$

Axial resolution thus can be improved by using UST operated at high frequencies but the extent of penetration of the ultrasound beam decreases considerably.

On the other hand, Lateral resolution is the ability of the transducer or imaging system to recognize or distinguish regions or objects that are closely situated along the plane perpendicular to direction of propagation of the ultrasound beam. It is dependent on the beam width of the UST which in turn depends upon the transducer's operating frequency, focusing of the beam and the gain applied to the captured echo signal. Lateral resolution is usually equal to beam width given by,

$$\text{Lateral Resolution} = \frac{\lambda}{\text{NA}}$$

, where NA represent numerical aperture.

For conventional circular probe, the beam width (BW) or beam diameter (BD) is defined by the points at which it has dropped by -6dB from its maximum on both sides of the beam and is given by [17],

$$BD = 1.02\,\frac{F\lambda}{D}$$

where F is the focal length, D is the element diameter of the probe and for a rectangular or flat probe such as a single element of an array, the beam width is given by [6],

$$BD = 0.2568\frac{DF}{N}$$

where N is the near field of the probe and the ratio F/N equals to 1 for a flat transducer.

This study is organized as follows. Chapter 1 introduced the basics of ultrasound and later describes about the principle and working of ultrasound and their respective characteristics. Following chapter 2 illustrates the principal and design of the systems which are used for mapping ultrasound pressure filed and also introduces another ultrasound application experiment for B-mode imaging using single element ultrasound transducer followed by their corresponding data processing. Chapter 3 demonstrates the usage of techniques learned and system designed from earlier chapters to quantify the pressure and temperature profile of a high intensity focused ultrasound (HIFU) with its introduction. Chapter 4 brings together the results from all the various experiments conducted and their respective results are presented with discussion to validate the concepts of ultrasound.

CHAPTER 2

Dynamic pressure wave Imaging and B-mode imaging using a ultrasound transducer

2.1 Purpose of this experiment

The main purpose of this module is to measure the dynamic propagation of ultrasound pressure wave for this purpose a needle-shaped hydrophone or a fine wire with a small diameter can be used [4, 5]. While a needle hydrophone can directly measure the local pressure variation, a fine wire can function as a reflection target to generate an ultrasound echo that indirectly represents the local ultrasound pressure [4, 5]. The reflected ultrasound signal from the wire should be processed by considering the round trip echoes to correctly observe the local pressure variation (see Sections 2 and 3). although these two methods have the capability to reconstruct dynamic ultrasound pressure waves, they are most frequently used to quantify the 2- or 3-dimentional ultrasound intensity distribution in literature [4, 5]. One of possible reasons might be that most time people are interested in intensity than pressure distribution due to their specific applications. Another possible reason might be the requirement of the relatively complicated system synchronization and data processing for reconstructing a 2- or 3-D dynamic pressure variation. However, a dynamic pressure wave is the foundation of ultrasound physics and imaging techniques, that is why it is more important to understand ultrasound properties than intensity. Also, there are many applications that require characterizing ultrasound pressure field (instead of intensity field) for example, when we try to investigate how an interfered ultrasound wave affects the dynamic oscillation of an immobilized microbubble, we would like to know how the ultrasound pressure distributes in the bubble area. Hence reconstruction of an ultrasound pressure field is highly desirable not only for education but also for research.

There may be some commercially available systems that can be used for 2- or 3-D reconstruction of an ultrasound pressure field based on a needle-shaped hydrophone (such as AIMS system from Onda Inc.) [7]. However, they are usually very expensive (>tens of thousands US dollars) and is unaffordable for many educators, professionals and researchers (even for those researchers whose main research streams are not ultrasound but need to measure a dynamic pressure field for some reasons). Therefore, in this study, we developed a cost efficient and multi-functional ultrasonic system that can be used to reconstruct 2- or 3-D dynamic variation of ultrasound pressure waves based on either a needle-shaped hydrophone or pulse-echo from a small diameter wire. In addition, this system can be used to conduct ultrasound imaging, and study microbubble-enhanced ultrasound imaging. Although the principle of the developed system is straightforward, engineering challenges exist in the development of both hardware and software. Therefore, we would like to share our design, experience and results to the community to make the system affordable and provide multi-functions for research and education.

This study is organized as follows. First, the principle of the system is introduced including both hardware and software. Following that, several experiments, data processing and results are demonstrated in including (1) the reconstruction of a dynamic pressure field generated from a focused ultrasound transducer using a needle-shaped hydrophone; (2) the reconstruction of a dynamic pressure field generated from a focused ultrasound transducer using a small diameter wire or needle; (3) the reconstruction of a dynamic interfered pressure field generated from two unfocused ultrasound transducers using a needle-shaped hydrophone; and (4) single element ultrasound imaging and microbubble-enhanced ultrasound imaging. Finally, we conclude this study in chapter 4 with results and discussions.

2.2 Measurement device and its characteristics

2.2.1 Pressure sensing Device

One of the measurement devices that are used for measurement of the pressure is a needle shaped hydrophone (HNP-0200, ONDA Corporation, CA, USA). HNP series are very stable with sensing diameter in order of hundreds of microns. They are also low cost with operating frequency range about 1 to 20 MHz with flatter sensitivity and better directivity with acceptance angle about 110^{O} [20]. Especially HNP-0200 suggests that its sensing diameter is about 200 μm. Mechanical specifications are depicted in following Fig. 2.1,

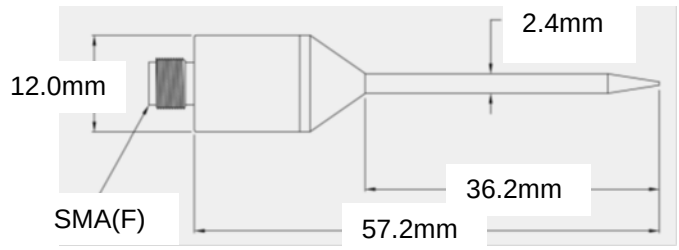

Figure 2.1 Mechanical design of HNP-0200 hydrophone [20].

2.2.2 Focused ultrasound Device

The focused ultrasound sources that are used for the following experiment are single element piezoelectric transducers with resonance frequency of 1MHz (V314, Olympus NDT, Waltham, MA, USA), 2.25MHz (V305, Olympus NDT) and 5MHz (308, Olympus NDT). These US transducers are immersion transducers with nominal element diameter of 0.75” and have 1” focal length. For interference ultrasound pressure mapping, similar type of unfocused 2.25MHz (A306S-SU, Olympus Panametrics) immersion transducer was used [21]. A general diagram depicting focused and unfocused transducer are shown in the Fig. 2.2 below,

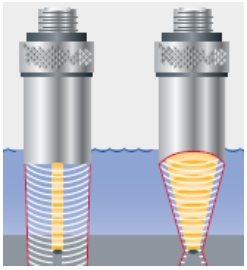

Figure 2.2 Unfocused and focused transducer [21].

2.3 System design

2.3.1 Ultrasonic pressure measurement system I — a needle shaped hydrophone method

The velocity of an ultrasound pulse propagating in water is taken has 1.48 mm/μs which take's tens of microseconds to travel tens of millimeters. This generally can be considered to be very fast to image the pressure dynamic distribution in 2-D or 3-D space. Therefore, our design idea is to reconstruct the dynamic propagation of an ultrasound pulse by measuring multiple identical (assumed) ultrasound pulses at different locations and time points. Eventually, having measured and analyze the relevant data, one can reconstruct the dynamic propagation of an ultrasound pulse in a 2-D or 3-D space thereby significantly reducing the system cost at the expense of data acquisition time.

Fig. 2.3 shows a schematic diagram of the measurement system. Generally, the system consists of three sub-systems: (1) ultrasound source, (2) data acquisition, and (3) mechanical translation. Each sub-system is indicated by a dashed rectangle in Fig.2.3 and the shaded rectangle represents a water tank in which the ultrasound transducer and the hydrophone are submerged. In the ultrasound source section, an ultrasound transducer (UST) was driven by sinusoidal burst signal generated by a function generator (FG, AFG 3252, Tektronix, TX, USA) to produce an ultrasound pulse. In the data acquisition section, the ultrasound pulse that was detected by a needle shaped hydrophone (HNP-0200, Onda Corporation, CA, USA) is acquired and stored for further analysis. The hydrophone converts the ultrasonic pressure signal into an

electronic signal that was further amplified by a broadband amplifier (AH-2010, Onda Corporation, CA, USA). The electronic voltage signal was digitized by a broadband digitizer (USB 5133, National Instrument, TX, USA) and further acquired by a computer via customized GUI software that was programmed using Matlab. When the FG sends an electronic signal to excite the UST, it also sends out a synchronized voltage signal that is used to trigger the digitizer. Therefore, data acquisition was synchronized with the excitation of the ultrasound pulse. In the mechanical translation section, a motorized 3-D translation stage system (x-y-z, VXM motor driven X-Slide assemblies, Velmex, NY, USA) was used, because of its relatively low cost and ease of programming, to translate the hydrophone to different locations. The only disadvantage of this is that it does not provide a feedback signal to indicate the actual location. In order to avoid the motion artifact caused by the mechanical movement (order of milliseconds), an important design requirement is to control the time delay between the data acquisition and the system scanning. However, it cannot be used as a master clock for triggering the entire system. Therefore, an additional DAQ card (PCI-6353, National Instrument, TX, USA) is incorporated to generate digital voltage signal to trigger this scanning system (serving as a slave system). An adequate time interval (order of microseconds) between the two adjacent triggers is incorporated with the help of the DAQ which should allow the hydrophone to arrive at the desired location before data acquisition at each scan point. To measure the wave propagation at the focal zone, the hydrophone is positioned around the focal area by manual translation to the location at which the ultrasound signal is observed to be the maximum. Generally 10 seconds delay before the initiation of a raster scan is adopted in order to allow sufficient time for the hydrophone to be positioned at the desired location, start of the raster scanning. This procedure is controlled by the customized MATLAB GUI software.

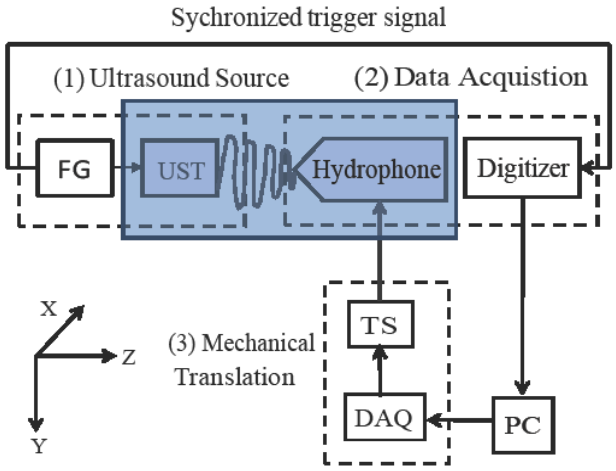

Figure 2.3 Block diagram of the experimental setup for imaging ultrasound propagation using a needle shaped hydrophone. FG: Function generator; UST: Ultrasound transducer; TS: translation stage; DAQ: data acquisition card.

The time sequence of the operation of the entire system is displayed in Fig. 2.4. Initially, the hydrophone is positioned at the focal area of the UST such that the ultrasound pressure reaches maximum at the center of the focal zone. The digitizer starts accepting the trigger signal from the FG, which is controlled by the Matlab software. The driving voltage of the UST and the trigger signal are synchronized (from the same FG) so that an ultrasound pulse, when generated, is acquired during the data acquisition window after the generation of the FG trigger. Thus, an ultrasound pressure wave as a function of time is acquired at a location in the scanning area which we term as scan point. After the completion of the data acquisition, the NI DAQ card sends a digital signal to trigger the translation stage, which is controlled by the software. Then, a relatively long period of waiting is incorporated for the hydrophone to be translated to the next location. After which the digitizer starts waiting for the next trigger signal from the FG and the acquisition cycle describe above repeats. It should be noted that the repetition rate of the trigger from the DAQ card (for the translation stage) can be much longer than that of the trigger from FG which allows to acquire pulse-echo signal multiple times and data averaging of these signal can improve overall signal-to-noise ratio (SNR) at each scan point.

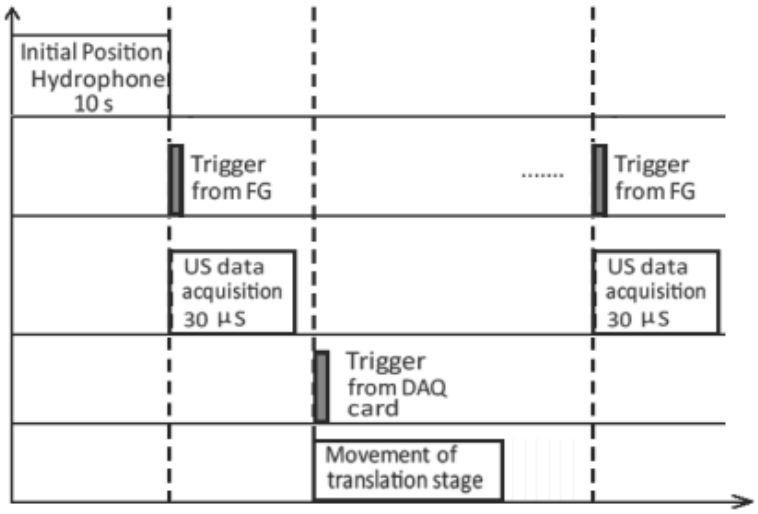

Figure 2.4 Time sequence of the operation for the entire system.

2.3.2 Ultrasonic pressure measurement system II — a fine metal wire based method

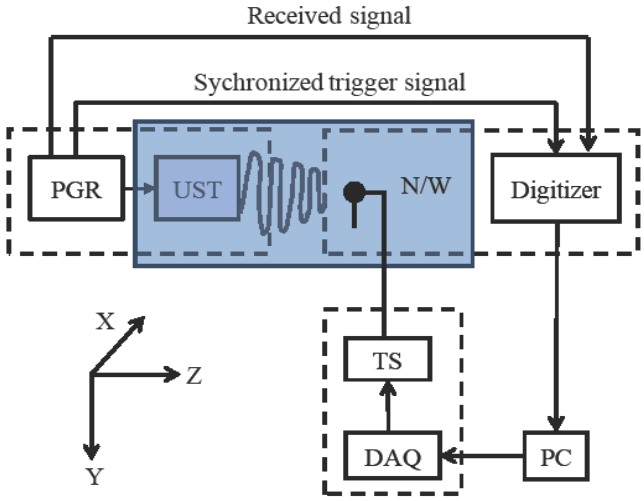

Figure 2.5 Block diagram of the experimental setup to acquire US pressure wave propagation using pulse-echo technique from a wire/needle. PGR: Pulse generator/receiver; UST: Ultrasound transducer; N/W: Needle or metal wire; TS: translation stage; and DAQ: data acquisition card.

The system setup, shown in Fig. 2.5, is similar to the hydrophone setup except that the hydrophone is replaced with a small metal wire with a diameter of ~0.46 mm for 1 MHz or a small metal needle with a diameter of ~0.25 mm for higher frequencies (2.25MHz and 5MHz). A pulse

generator/receiver (5077RP, Olympus NDT, Waltham, Mass, USA) is used to drive the transducer, with a pulse repetition frequency of 200 Hz, energy per pulse of 4 µjoules, a gain of 10 dB and a low pass filter (20MHz). The pulse generator/receiver serves as both an ultrasound source and receiver, and also amplifies the received ultrasonic echoes from the wire or needle. The received signal is stored using a digitizer which is triggered by the synchronized output from the pulse generator/receiver. The sequence of data acquisition, triggering and scanning is similar to the hydrophone system. In contrast to the hydrophone system data processing, the data acquired from this system should be processed carefully because the ultrasonic pulses travel a round trip. An example is illustrated in Fig. 2.6 and 2.7. Fig. 2.5 shows the configuration of the measurement system. At each scanning point we acquire an A-line which represents an ultrasound echo signal along the propagation of the ultrasound. The echoes acquired at any two scanning points along the axial direction (see Fig.2.6) are separated with a distance that is the twice the actual distance because each echo travel a round trip. To remove the effect of the round trip following steps have been implemented 1) each scanning point at the first row is used as a reference for all the other scanning points in their respective columns (see Fig.2.6) 2) the data acquired from all the scan points (at the same column except scan point belonging to the first row) are circularly shifted to the left, considering the ultrasound wave propagation is from right to left (see Fig.2.7(b)). This ensures that the distance between echo signals at each scan points is equal to the actual distance with respect to echo signal at any another scan point in that column. The number of data points to be shifted in each echo signal (A-line) is calculated by the product of the number of the data points acquired per unit distance in water (i.e., the data sampling rate divided by the speed of ultrasound in water) and the axial distance between the 1^{st}/reference scanning point and the scanning point to be shifted. This can be well explained by following example where two echoes signals illustrated in Fig.2.7 (a), are measured at the 1^{st} and 17^{th} scanning points along z (axial) direction and at the 30^{th} scanning point along x (lateral) direction. The distance between the two peaks of the echoes is around 4 mm, which is twice the actual distance of 2 mm. The processed data are shown in Fig. 2.7(b) in which the US data are

circularly shifted toward the left. The distance between the two peaks agrees with the actual distance between the two scanning points (~2 mm) thereby validating the processing technique described earlier. The similar processing is done for each column. The rest of data processing is similar to that in the hydrophone system.

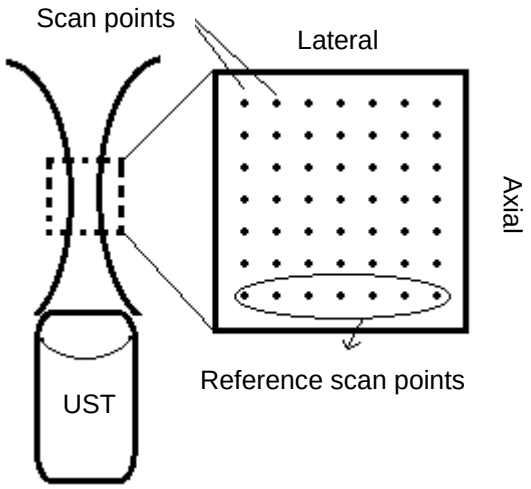

Figure 2.6 Diagram depicting area scanned by hydrophone along with scan points in ZX plane.

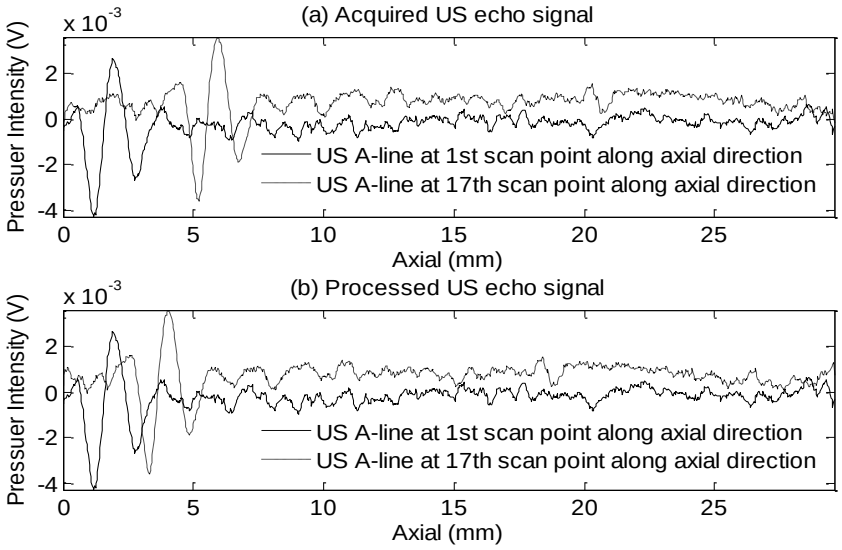

Figure 2.7 (a) Original US echoes acquired from the pulse generator/receiver, (b) Processed echoes by circularly shifting data points to remove the round trip period.

2.3.3 Ultrasonic wave interference using the hydrophone-based system

The interference between two ultrasonic waves is an important phenomenon and to efficiently reconstruct the dynamics of ultrasound interference is one of the possible applications of this study for which the hydrophone-based experimental system discussed earlier can be adapted. In the setup, two synchronized output channels from the FG were used to simultaneously drive two 2.25 MHz unfocused UST (Olympus Panametrics A306S-SU), respectively. These two USTs were arranged perpendicular to each other. The USTs were pulsed at 2.25 MHz with 1 cycle per pulse, with a peak-to-peak voltage of 1.8 V. The hydrophone was positioned at the intersection of the two ultrasonic waves confirmed by manually tuning the translation stages, on which one of the UST was mounted, to the position where maximum signal is observed. The dynamic ultrasound pressure distribution can be reconstructed in the similar fashion, as used in imaging the dynamic ultrasound pressure wave using hydrophone–based system, by scanning the hydrophone in an area of interest which is about 2.5x2.5 mm^2. The results have be presented and discussed in the respective section in chapter 4.

2.3.4 Ultrasound B-mode imaging using a single ultrasound transducer:

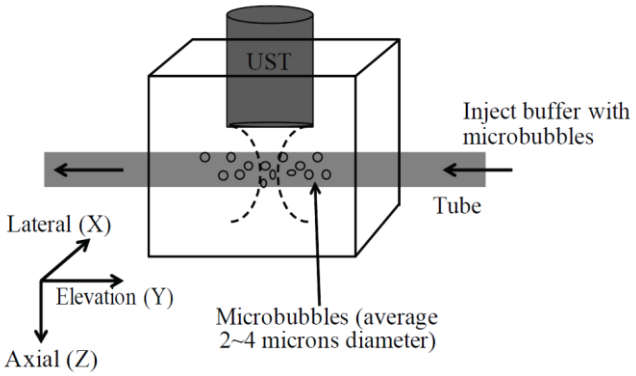

Figure 2.8 Experimental setup for ultrasound imaging of a small tube (outer diameter=2.39mm, inner diameter= 0.79mm) filled with different concentrations of microbubbles.

Another application that can be demonstrated using the single element ultrasound transducer is B-mode ultrasound contrast imaging. By replacing the wire with a plastic tube and scanning with the focused ultrasound transducer (instead of the tube), cross-section of the tube can be reconstructed. Fig. 2.8 shows an experimental setup. Three scenarios were undertaken by circulating air, water and microbubble through the plastic tube that was submerged in a water tank. A 5 MHz focused UST (V308, Olympus Panametrics) was mounted on the support from the motorized translation stage. The tube is positioned at the focus of the 5MHz UST after which scanning was done laterally (along x direction) across the tube and an optimized lateral resolution was achieved. B-mode 2D image of the cross-section of the tube with air, water and microbubbles are acquired and discussion of their comparison is presented in corresponding section of chapter 4.

2.4 Software

The entire system is controlled by a custom software package that is developed based on MATLAB via a GUI (Graphical User Interface). The software consists of three modules: 3D position controlling, data acquisition and data processing. The 3D positioning module controls the motion of the hydrophone (or the metal wire or needle), including the initial position, scan range, step size, scan speed and acceleration. The data acquisition module of the GUI controls the DAQ card and digitizer, which includes sampling rate and time, repetition of scans, delay time between each scanning point, and the data storage. The data processing module controls the reconstruction and display of the ultrasound wave, such as in A-mode and B-mode.

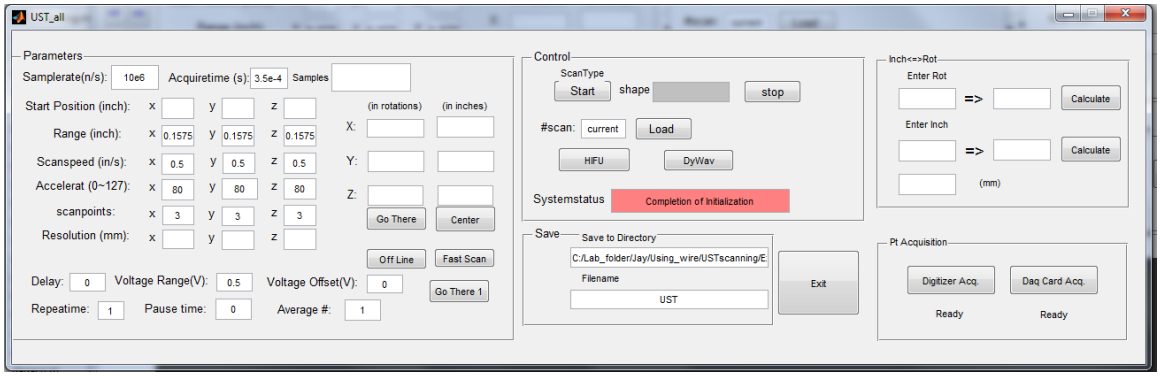

Figure 2.9 Customized MATLAB GUI.

CHAPTER 3

High Intensity focused Ultrasound

3.1 Introduction

High Intensity focused ultrasound is emerging as one of the potential cancer treatment modality in medicine and its major recognition is for its noninvasive method and use of non-ionizing radiation [22]. Theoretical principal revolves around the notion that larger concave geometry of piezoelectric crystals could be driven with higher voltages thereby generate greater than normal pressure waves that are concentrated at region or area much smaller than focal spot area of conventional focused ultrasound transducers (in order or less that few mm). Immense magnitude of pressure and limitation of area of exposure along with absorption characteristics of the medium increases the chance of energy dissipation through generation of corresponding high temperature. The HIFU generated pressure wave which generates proportional heat is highly dependent on the method of excitation of HIFU transducer implying the ability to swiftly increase/achieve moderate (>2 ^{o}C) to high temperature (>50 ^{o}C) over a small region of interest (ROI), which is intrinsically depending upon the frequency of transducer which can be as small as hundreds of microns (ex: 0.16mm lateral 1.5mm depth for 7.5MHz), over considerable less exposure time (<1S). This thermal absorption of HIFU energy leads to lesion formation which becomes the basis for a phenomenon known as thermal ablation [23]. Such attributes are demanding intensive clinical research and regulatory activity for localized thermal ablation of a focused tissue volume with minimal to no damage to the surrounding tissues [24]. The current or potential indications for use include treatment of tumors in the prostate, breast, liver, kidney, and brain, debulking uterine fibroids, cardiac ablation during surgery, and stemming blood flow via acoustic cauterization [25]. ODE (Office of Device Evaluation, component of FDA) has classified

HIFU device as a Class III, in general, when used to treat tissues at considerable large depth (i.e., greater than a few centimeters below an organ surface) but depending upon the other intended usage it sometimes be classified as Class II. Class III is for high risk devices or for devices using new technology in which there is a new intended use or a new type of safety or effectiveness is questioned with mandatory Premarket Approval (PMA) and, human clinical trials normally are conducted as part of the regulatory review [25].

3.2 Goal of this Experiment

The main aim of using HIFU experiment is to generate a controlled temperature ^{o}C increase which can also be considered throughout the rest of the discussions as temperature spike. Keeping this as the primary constraint other characteristics, that can be controlled, needs to be thoroughly examined thereby obtaining and designing an optimum protocol. The characteristics that needs to be scrutinized are,

a. The burst width that governs the duration for which the HIFU needs to be driven.
b. Power delivered at the focus which is directly dependant on voltage parameters of the source driving signal such a Voltage peak to peak (Vpp) thereby the Voltage root mean square (Vrms).

$$P_{avg} = (V_{rms})^2 \times Y$$

$$V_{rms} = V_{pp}/\sqrt{2}$$, where Y is admittance.

c. Duration between burst, controls the interval for which the specimen, in our case a phantom or a tissue, needs to be relaxed (HIFU is not driven) for the temperature to drop down.

Physical characteristics of the HIFU also governs major behavior of the HIFU, such as the area of exposure (AOE) also known as region of interest (ROI), by concave shape of the HIFU transducer along with the diameter of the circular shape of the transducer. However the actual

size of convergence of the pressure waves from the surface of the HIFU's piezo crystal depends primarily, as discussed in the chapter 1, on the frequency at which the HIFU is driven.

The measurement devise or the sensor used for HIFU focal area measurement needs to have small sensing area (in the order of 100 microns) and can repeatedly measure wide frequency range of ultrasound fields with flat frequency response (from 500kHz to 20MHz, when harmonics are included). Therefore a moderate pressure sensing device with no emphasis on high temperature sensing is needed in the HIFU focal area estimation experiment.

To understand the HIFU and its applications, experiments were carried out in three stages:

a. Map out the lateral and depth distribution of the pressure waves to understand the focal area pressure distribution of the HIFU.

b. Calibrate the temperature sensor for temperature measurements.

c. Record the temperature distribution similar to pressure distribution inside a phantom of certain thickness at the focal area overlapped with the temperature sensor (in our case Fine wire thermocouple).

3.3 HIFU Focal area measurement

3.3.1 Purpose of this experiment:

The main aim of this phase of the experiment is to verify the working of HIFU by measuring focal area of the HIFU exposure when driven at 2.5MHz (HIFU's fundamental frequency) and 7.5MHz (HIFU's third harmonic frequency). This is achieved by mapping out the pressure profile from the HIFU transducer either by using pulse echo method, where an echo from the fine wire of diameter much less than the expected focal spot width is used, or recording absolute pressure using a hydrophone. The sensing tip of the hydrophone as mentioned in earlier chapter is about 200μm which can be used for measurement of 2.5MHz focal spot area with precision since 2.5MHz focal width is expected to be around 0.5mm but in case of 7.5MHz, focal area width is

about 0.16mm due to which hydrophone if used would give a near FWHM value since the point spread function of hydrophone is comparable to the 7.5MHz focal width. The result obtained for 7.5MHz therefore is approximate when compared to company's specification. Much reliable option was to measure the temperature profile using a fine wire thermocouple of diameter 25µm. The temperature profile closely resembles the pressure profile for 7.5MHz since its power is focused into much smaller area due to decrease in wavelength but can give a much better estimate to true value. Though 7.5MHz focal area using hydrophone is described in this section a more reliable verification can be seen in temperature measurement section discussed later in this chapter.

3.3.2 System design:

Initial phase of the HIFU experiment is similar to the needle shaped hydrophone method experiment discussed earlier. The HIFU (H-108, Sonic Concepts, WA, USA) is a high-efficient broad bandwidth immersible transducer used to generate an high-intensity focused ultrasound which is driven at its fundamental frequency of 2.5MHz. It operates at free-field conditions and a 50 ohm RF amplifier is recommended to be used in order to generate the desired high intensity with a maximum pulsed power level of 400W. Moreover the structural integrity and working ensure a highly uniform radiating surface. In order to achieve fundamental resonance near 2.5MHz, it is connected to an RF impedance matching circuit via a BNC cable which is protected by a plastic texture surrounding the whole length of connection. The 7.5MHz optional matching network is provided to drive the HIFU at its third harmonic. The HIFU is suspended in place of Olympus NDT transducer which is considered to be optimal for various other experiments that needs to be carried out later. The alignment of the HIFU with respect to the ground is carefully adjusted using small surface bubble leveler with scale thereby ensuring that the ultrasound emitted from the HIFU is perpendicular to the ground facing upwards. The whole arrangement is connected to the 3D motorized translation stage that is controlled using a customized GUI Matlab

application in order to translate the HIFU transducer with highest precision possible (smallest translation of motorized 3D translational stage is about 6.35μm).

For 2.5MHz and 7.5MHz focal area measurement: The needle hydrophone (HNP-0200, Onda Corporation, CA, USA) is carefully suspended exactly above the HIFU such that the hydrophone tip is facing the surface of the HIFU. Care must be taken to drive the HIFU at low Voltage since the sensitivity of the Hydrophone is very high implying that high voltage supply to HIFU will generate corresponding high pressure at the tip of the hydrophone which could permanently damage its surface and would lead to what is known as ringing effect, which is generating multiple damping pulse for single pressure pulse, and may eventually lead to total decommission of the hydrophone. The HIFU is mounted on a motorized translation stage which is used to pin point the maximum voltage output from the hydrophone both in depth (i.e., along the ultrasound propagation direction, XZ plane) as well as along perpendicular plane to the ultrasound propagation direction (XY plane). This is set as the input to customized MATLAB GUI and then a scan area is set accordingly thereby ensuring precise calculation of lateral and depth FWHM pressure distribution. The remaining of the experiment is quite similar to the needle hydrophone experiment conducted for earlier ultrasound transducers and from the event time line from Fig. 2.4 in the earlier chapter.

3.4 HIFU Temperature Measurement

3.4.1 Theory behind HIFU induced Heat Generation:

In theory behind the temperature rise, T, at the sensing part of the thermocouples embedded in the unperfused TMM, that we have employed for the thermal investigation, is better understood using energy-balance equation widely known as Penn's equation stated below in its general form has [26, 27],

$$\rho C \frac{\partial T}{\partial t} = W_b C_b (T_b - T) + Q_m + Q - k\nabla^2 T$$

Where ρ is the density, c the speed of sound in the tissue (in our case TMM), k the thermal conductivity coefficient, W_b is the blood perfusion (kg /m^3. s), C_b is for the specific heat of blood, Q_m is the metabolic heat generation, T is tissue temperature, t is time and Q is heat source.

The exposure time of HIFU for our investigation is very short (about <0.1s) due to which blood perfusion and metabolic heat can be ignored. The heat source can be considered to have two integral terms which as stated below [27],

$$Q = \mu_a I + Q_{vis}$$

Where μ_a =Intensity absorption coefficient, I is the in situ temporal average intensity and Q_{vis} is the rate of conversion of ultrasound energy to heat per unit volume via viscous interactions. It is inferred that the thermal and acoustic tissue properties for a given tissue or TMM greatly influenced by both position and temperature. It can also be admitted that the Penn's equation shows linear dependency with respect to t, it is then possible to consider the temperature rise, T, to be the sum of two different contributions, T_{abs} and T_{vis}, arising from acoustic energy absorption and viscous interaction respectively. Hence by considering thermal properties of medium to be homogenous, Penn's equation can be rewritten as [26],

$$\rho C \frac{\partial T}{\partial t} = [\mu_a I - k\nabla^2 T] + [Q_{vis} - k\nabla^2 T]$$

The right-hand side of the above equation can be considered to have two components which are ultrasound absorption component in the medium, and the component of viscous interactions close to the thermocouple, respectively. At the start of insonation, the viscous heating contribution, Q_{vis}, is expected to be the major term (Fry and Fry, 1954) leading to a rapid temperature increase close to the thermocouple, which in turn leads to an increase in ∇T_{vis} and $k\nabla^2 T_{vis}$ until the sum of the terms in the second bracket of equation (2) approaches zero (at least for points close to the thermocouple). At this time, the viscous temperature contribution no longer varies and any further change in the measured T is due to normal absorption and conduction mechanisms. If the term $k\nabla^2 T_{abs}$ is still very small at this time, the rate of change of temperature is

nearly proportional to $\mu_a I$. This relationship is used for determining the absorption coefficient of tissues when Intensity is known. Alternatively, the cooling curve can be used to determine the absorption coefficient of tissue (Parker 1983b). At the end of insonation, the $\mu_a I$ and Q_{vis} terms go to zero and the curve depends only on the $k\nabla^2 T_{abs}$ and $k\nabla^2 T_{vis}$ terms. Immediately after insonation, the $k\nabla^2 T_{vis}$ will dominate since the thermal gradient is greatest at the thermocouple junction. Thermal diffusion will cause this term to reduce rapidly allowing the $k\nabla^2 T_{abs}$ to dominate the shape of the curve. The rate at which it does so will depend on the beam width of the HIFU field [26].

Dr. Hynynen and Dr. Clarke in order to correct for viscous heating, whether during the heating or cooling phases, assumed that thermal diffusion of the absorptive heating contribution takes place on a much longer timescale than viscous heating diffusion. In order to verify this assumption, Dr. Hugh Morris and his colleges have investigated analytical solution given by Bacon and Shaw to assess the length of time available to make a measurement before thermal conduction ($k\nabla^2 T_{abs}$) has a significant influence on either the temperature or the rate of change of temperature. The conclusion of the investigation suggested that the thermal conduction in typical HIFU fields becomes significant and sometimes critical in less than o.1sec. Thus previous practice of ignoring the initial temperature rise to avoid the false temperature rise due to viscous heating was shown not to be ignored for short duration HIFU exposures [26]. Thus the above equation can be further simplified considering very short duration of HIFU exposure (ignoring the viscous heating influence) and a fixed source of acoustic energy can be a simplified to linear relation given as [27],

$$\rho C \frac{\Delta T(z, r)}{\Delta t} = 2\mu_a I$$

Thus for the very short HIFU exposure time it can be shown that the distribution of temperature and sound intensity has a linear relationship (regardless of the change in α). Therefore we can measure the temperature distribution at the focus to indirectly reflect the sound field distribution [27].

3.4.2 Purpose of our experiment:

The main aim of this phase of the experiment is to measure the temperature profile in the focused area of HIFU driven at 2.5MHz and 7.5MHz using commercially available fine wire thermocouple. The experiment is also aimed to set a threshold of temperature and investigate the combination (by varying) of power, duration, width and interval of each burst that would be feasible. This is proven to be very difficult since a lot of parameters are varying therefore later during the experiment, width of the burst is considered to be maintained at 100mS which has proven to lessen the complexity of measurement.

3.4.3 Calibration the Fine wire thermocouple for HIFU temperature measurements:

Fine wire thermocouple (CHCO-002, Unsheathed fine gage thermocouple, Omega, CT, USA) of diameter 0.002inch is an E-type Chromega-constantan junction thermocouple is used for the HIFU measurement whose sensing part is a bead at which the two materials join. The calibration phase of the thermocouple involved recording the varying voltage across the ends of the thermocouple when the thermocouple sensing bead is kept at corresponding varying temperature, which is controlled using 0.5^{o}C precision miniature bench-top temperature controller (CSi32, Omega, CT, USA). The bench-top temperature controller uses a) Type-T copper-constantan junction thermocouple probe for measuring the absolute temperature which is used in an feedback loop for the controller to control temperature to set value with $\pm 0.5^{o}$C tolerance and b) Heater (VPT-107, Omega, CT, USA) is an immersible 100W stainless steel sheath with PFA coating for maintaining the temperature of the water in a beaker in which the fine wire thermocouple is suspended. In order to measure the temperature at the sensing bead of the Type-E CHCO-002 fine wire thermocouple, a handheld digital thermocouple (HH508, Omega, CT, USA) with its own Type-E thermocouple connector is arranged, with great precision, such that its sensing tip is in very close proximity of the fine wire thermocouple sensing bead. The output of the fine wire thermocouple is connected to the Low noise preamplifier (Model: SR560,

SRS, CA, USA) whose convenient and main features include but are not limited to gain control (adjustable as a product of 1, 2 or 5 and a multiplier (none (i.e. 1), 10, 100, 1000 or 10,000)) and filter control (0.03 Hz to 1MHz) with desirable choice of filter mode (high pass, band pass and low pass). The output of the pre-amplifier is connected to an oscilloscope (DPO 7254, Tektronix, OR, USA) where the mean voltage is observed and noted down for corresponding temperature variations. The protocol for calibration thus involved a) Maintain the temperature of the water beaker, b) Record the temperature at the fine wire thermocouple sensing bead using HH508, c) Record the mean voltage observed in the Tektronix oscilloscope. A plot has been generated using these recording with voltage against the temperature and the slope of the plot would yield the approximate but reliable rate of change of voltage output from fine wire thermocouple with respect to corresponding temperature change.

3.4.4 Tissue mimicking material:

In order to undertake the temperature measurement related to human body, initially it is advisable and a common practice to conduct experiment on materials that have similar thermal and acoustic properties to that of human soft tissue which is commonly referred to has Tissue mimicking material (TMM) or phantom whose main characteristics are reproducibility and reusability. Extensive research with utmost importance to such materials has been conducted over last two decades with sophisticated protocols and method leading to creation of successful samples but this is only possible in controlled laboratories. Such materials, most of which should have most of the above mentioned characteristics, are available commercially due to increase in demand for similar studies and one such material is silicone elastomer (VST50, Factor II, AZ, USA). It is a translucent two component, 10:1 mixing by weight low viscosity addition platinum cure room temperature vulcanizing silicon elastomer. It has proven to exhibit good stability and robustness along with near to practical acoustic and thermal property. Material acoustic parameters for tissue mimicking material obtained from previous research by Prof. Zell and his group is shown in the table 3.1 below [28]:

Table 3.1 Acoustic parameters for water and tissue mimicking material

Water	Velocity of sound (10^3 m/s)	1.482
	Density (10^3Kg m^{-3})	1
	Impedance (10^6Kgm^{-2}s^{-1})	1.48
	Acoustic attenuation coefficient. (dB cm^{-1}) ($\alpha_{water}=0.2f^2$, f(freq. in MHz))	0.002
Silicon Elastomer	Velocity of sound (m/s)	1.03±0.06
	Density (10^3Kg m^{-3})	1.07±0.03
	Impedance (10^6Kgm^{-2}s^{-1})	1.10±0.05
	Acoustic attenuation coefficient. (dB cm^{-1}) ($\alpha_{silicon}=6.06f^{0.49}$, f(freq. in MHz))	14±1.4

3.4.5 Cavitation

3.4.5.1 Introduction and Literature review:

Cavitation, or acoustic cavitation, is gas or vapor-filled cavities caused by the tension from the negative acoustic pressures generated during the rarefaction portion of the HIFU pressure cycle [29]. From the macro scale motion of the acoustic transducer the energy is transferred and focused to the micro scale vapor inside such Microbubbles by almost simultaneous growth and collapse of these Microbubbles [30]. Such behavior generates high pressure and thereby high temperatures inside the bubbles during the vapor phase. High temperature caused by HIFU exposures can also generate similar gas micro-bubbles through non-acoustic, thermal mechanism leading to either vaporization of liquid into vapor or formerly dissolved permanent gas

is exsoluted out of liquid and into gas spaces [30]. High intensity focused ultrasound therefore was widely investigated to be used in non-invasive soft tissue thermal ablation within the body and each of these various perspectives provides useful insight for better understanding of the phenomenon.

According to the summary presented by Di Chen and his group and also from the various studies on the mechanism of HIFU induced cavitation it can be understood that the energy delivered through pressure waves to focal region, where the acoustic pressure has induced bubble formation, will be converted to mechanical energy observed has bubble's oscillations [22]. Depending upon the nature of the acoustic cavitation, it is divided into non-stable or transient (or inertial) and stable (non-inertial). As addressed in the previous investigations during the stable acoustic cavitation the viscous absorption at the bubble boundary causes the rise of heating rate whose underling indication is relatively prolonged and stable oscillation of the bubbles. Consequently during the transient acoustic cavitation, bubbles oscillate randomly reemitting signals of broadband frequencies rich in harmonics of the driving fundamental frequency of the HIFU device. The prolonged and stable oscillations in the former case and the fundamental harmonics rich signal in the latter case both serve conclusively has an excellent indicator that the cavitation phenomenon has been induced during HIFU sonication [22]. So we proposed an experiment that utilizes these indicators to demonstrate the absence or presence of HIFU induced cavitation.

3.4.5.2 Method for cavitation detection:

Active cavitation detection (ACD) is a technique of using an ultrasound transducer (UST) for a pulse-echo sonogram to get an echo signal along the line of propagation of the ultrasound from the UST device [29]. It is reported that in case of HIFU induced cavitation any ultrasound exposure from another ultrasound transducer (probing transducer) at the focal spot of the HIFU sonication would yield either high amplitude echo of fundamental frequency or its harmonics or

both due to the presence of bubbles. The simple but reliable method of using pulse-echo technique for contrast imaging thus can be used to observe the presence of bubbles and their presence, as investigated by previous studies [31, 32], is indicated by hyper-echogenicity along the A-line or B-mode recorded by the probing transducer (UST). The inherent challenges associated in using ACD method include difficulty to differentiate between HIFU induced acoustic cavitation to those caused by boiling bubbles since both of these are seen to create similar hyper-echogenicity in a B-mode scan. But the aim of our experiment is to confirm the absence of these bubble either induced by HIFU acoustic pressure waves or by temperature spike due to HIFU sonication since ACD technique can be considered to be reliable and efficient for micro bubble detection.

3.4.6 Experiment Design:

As discussed above the experimental setup should include two major aspects for investigating the temperature distribution profile inside a TMM material. Primary investigation would be to arrange HIFU transducer to be focused on the fine wire thermocouple inside the TMM and second would be to incorporate cavitation detection system by using the readily available 5MHz single element focused ultrasound transducer (V308, 1 inch focal length (FL)). As basis for any initial research, the assumption made for the following investigations are,

1. The velocity of sound in water is taken to be 1.48mm/μs.
2. Viscous heating does not influence the temperature reading during the insonation of fine wire thermocouple embedded inside TMM.

The experiment design shown in Fig. 3.1 consists of four main section or stages,

a) Source section
b) Translation section
c) Cavitation detection section and
d) Acquisition section.

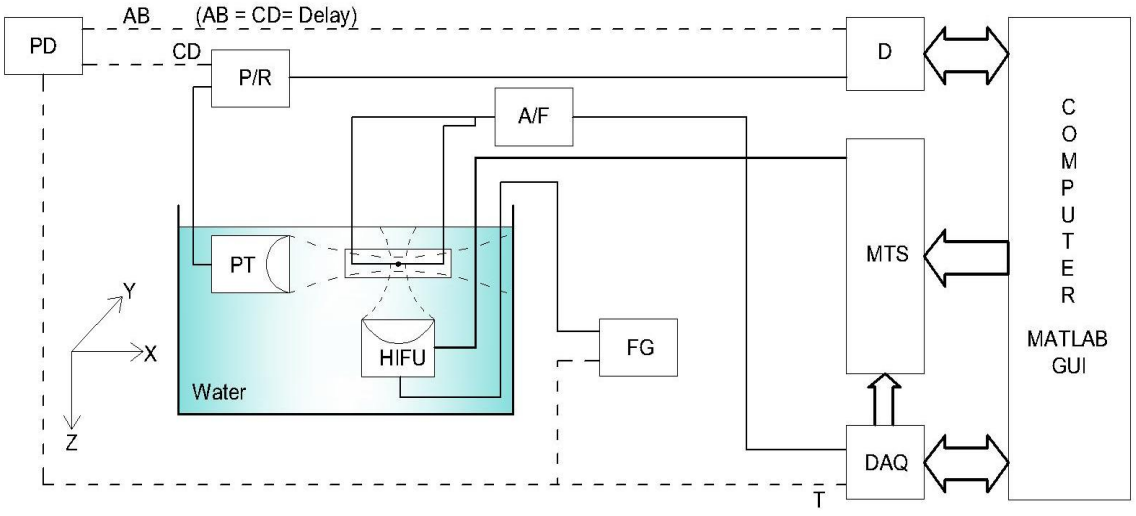

Figure 3.1. HIFU Temperature measurement setup with cavitation detection system. FG: Function generator; PT: Probing Ultrasound transducer; MTS: Motorized translation stage; DAQ: data acquisition card, D: Digitizer, P/R: Pulse generator/receiver, PD: Pulse delay, HIFU: High intensity focused transducer

The source section, consists of dual channel function generator (FG, AFG 3252, Tektronix, TX, USA) used to drive the High Intensity focused ultrasound transducer at the operational frequencies of 2.5MHz (fundamental frequency) and 7.5MHz (third harmonic). The specification of the HIFU's geometry and the capacity of the active element of HIFU transducer cannot be changed. So the only parameter that can be controlled is the power that can be delivered to the HIFU. The threshold power that can be given to HIFU beyond which it can be damaged is 100W (as per Sonic Concepts specifications) when driven in continuous mode. The power delivered to HIFU transducer depends upon the duty cycle and the amplitude of the voltage and is controlled by the function generator (FG) by controlling the number of cycles within a burst and this in turn depends upon the driving frequency of the HIFU transducer. The amplitude given out by the FG is quite less to achieve the desired energy at the focal spot for which Radio frequency (RF) power amplifier (325LA, E&I, NY, USA) is used. The output gain of RF amplifier is about 50dB giving roughly about 300 as gain. The output of power amplifier is given to the HIFU transducer through an impedance matching circuit.

The translation section is similar to that used to translate the motorized linear translators using the DAQ card through customized MATLAB GUI to map out the pressure waves in the chapter 2.

The cavitation detection section is the major modification to the system already in use. This section consists of single element 5MHz (V308, Olympus NDT) ultrasound transducer (UST), delay generator (DG645, SRS, CA, USA), pulse generator/receiver (5077RP, Olympus NDT, Waltham, Mass, USA), function generator (FG) and the broadband digitizer (USB 5133, National Instrument, TX, USA). The delay generator is used in conjunction with pulse generator/receiver (P/R) to record the A-line using pulse-echo method at the sensing part of the thermocouple embedded inside the TMM. The purpose of the delay generator is to delay the trigger to both pulse generator/receiver and the digitizer whose function is to transmit/receive the A-line and store this A-line respectively. This is necessary to record the cavitation effects at three instances while sonication using HIFU burst which are, before the end of HIFU sonication, at the end of HIFU sonication and after the end of HIFU sonication (which is for 100mS 'ON'). The reasonable positioning of 5MHz UST is arranged orthogonal to the HIFU transducer. Before the run of the whole experiment the positioning of the probing transducer (5MHz) with respect to thermocouple embedded inside the TMM is located using the pulse-echo method, described in earlier section, by manually translating the probing transducer along the cross-section of the fine wire thermocouple (TC) until an echo signal from fine wire is observed. Later the 5MHz UST is moved along the length of the TC with specific distance, since the whole 5MHz UST is mounted on 3D manual translation stage. The error in positioning is acceptable to about ±1.6mm along the axial length and ±0.2 along lateral length of 5MHz UST since the focal area of 5MHz UST (called as probing transducer (PT)) is about 3.4mm X 0.4mm respectively. Two instances of measurement were recorded 1) keeping the probing transducer focused at the sensing part of the embedded TC and drive HIFU for sonication to record 2D XY and XZ plane and another instance is to 2) keep the HIFU transducer stationary and focused on the sensing portion and recording multiple

A-line along the lateral length of the TC to generate B-mode image. Either of the two or both can be used to observe echo signal where increase in echo amplitude or harmonics would indicate presence of micro bubbles induced by the HIFU transducer sonication. Also before driving the HIFU, probing transducer was used to examine any presence of bubble inside the TMM in order to ascertain undesired temperature increase from micro-bubbles already present in phantom which might lead to misdiagnosed presence of HIFU induced bubbles inside TMM.

Acquisition section mainly consists of recording of voltage signals across the TC which is fed to Low noise pre-amplifier (AMP, SR560, SRS, CA, USA) whose main feature includes but not limited to gain (set to 1000) and filter (set to 10Hz Low pass filter). The output of the filter is given to DAQ (D) to be recorded. The trigger to start recording the signal from the TC is fed directly from the function generator (FG) to acquire the signal as soon as the HIFU sonication commences. The signal is recorded for about 1.5 seconds but the trigger interval is set to 2 seconds to avoid skipping of successive triggers from FG. The working is quite similar to acquisition section used in the earlier section but during the temperature measurement, signal from embedded fine wire thermocouple (TC) and A-line from the pulse generator/receiver are to be recorded almost simultaneously. The digitizer (D) is used to record the A-line, since it requires high sampling rate, but the voltage reading corresponding to temperature at the sensing part of the TC is quite slow (order of 10ms) which requires much slower sampling rate thus DAQ card is used which has maximum sampling rate about 200 thousand samples per second. Another advantage of using DAQ over digitizer is that, temperature corresponding to voltage reading from TC might have DC shift which corresponds to overall increase in temperature of the region of scan due to which digitizer (D), where there is a need to specify voltage range and voltage offset, might not record with changing DC shift or proper quantization respectively, but in case of DAQ it automatically adjusts itself to corresponding DC base line and quantize accordingly every time it captures the TC signal. Thus DAQ card captures the TC voltage signal with a sampling rate of about 100 thousand samples per sec and digitizer (D) recorded A-line signal from probing

transducer with sampling rate about 100 million samples per sec. Both have been stored individually as a .mat file for each scan point with proper xyz coordinates indicated in the file name.

The recorded results for different set of experiments are generated with appropriate images and are presented with discussion in chapter 4.

CHAPTER 4

Experiments, results and discussions

4.1 Reconstruction of ultrasonic wave propagation using the hydrophone-based system

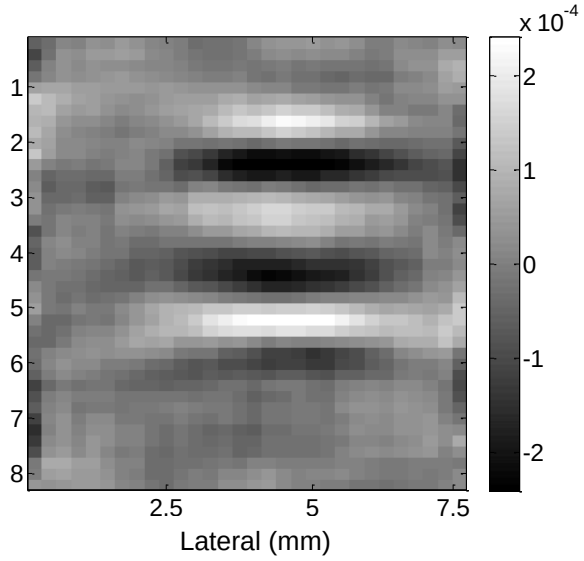

Figure 4.1 A reconstructed 2D (x-z or lateral-and-axial) image of the pressure distribution of an ultrasound pulse at the focal zone generated from a 1 MHz UST and measured using the hydrophone-based system.

Fig. 4.1 shows the reconstructed image of the ultrasonic pressure distribution in XZ plane at the focal area of the UST (V314, Olympus NDT, Center frequency 1MHz) (x: lateral direction and z: axial direction) using the hydrophone-based system. The UST is excited by a 2-cycle sinusoidal burst signal from the FG with a central frequency of 1 MHz and a peak-to-peak voltage of 1 V. Assuming ultrasound speed in water at 20 °C is about 1.48 mm/μs and the wavelength of a 1-MHz ultrasound wave is about 1.48 mm. The image shown in Fig.4.1 represents a snapshot of the ultrasonic wave propagation around the focal zone of the UST. The gray scale represents variation of the acoustic

pressure. White and black represent positive and negative pressure, respectively. The wavelength can be calculated to be about 1.5 mm from the two adjacent positive peaks, which is close to the estimated value of 1.48 mm. Video 1 shows the dynamic propagation of the reconstructed ultrasound pressure wave in the UST focal zone. From these results we can infer that the hydrophone-based system can accurately reconstruct the dynamic ultrasonic pressure distribution. Advantages of using a hydrophone are: (1) it can measure the absolute value of the ultrasonic pressure if the hydrophone is calibrated, (2) compared with the wire-based method the data processing is relatively simple, (3) it does not require a pulse generator/receiver which is usually a little more expensive than a needle hydrophone, and (4) due to the high sensitivity of the hydrophone (in micron Pascal range), it is not necessary to amplify the driving signal with a radio frequency (RF) amplifier that is usually expensive. However the disadvantage is that the lateral spatial resolution of the reconstructed image is limited by the physical diameter of the hydrophone tip (the axial resolution is related to the system dynamic response speed and not limited by the physical size of the hydrophone). In this study, the hydrophone diameter is about 200 microns, which is sufficient for reconstruction of focused pressure waves at low frequency, such as 1-5 MHz, or unfocused ultrasound waves. It is to be duly noted that the measured ultrasound pressure wave represents the convolution between the excitation voltage signal and the impulse response functions of the UST, the hydrophone and its amplifier. Therefore, the bandwidth of the measured signal is determined by the one that has the narrowest bandwidth, which is the UST.

4.2 Reconstruction of ultrasonic wave propagation using the wire-based system

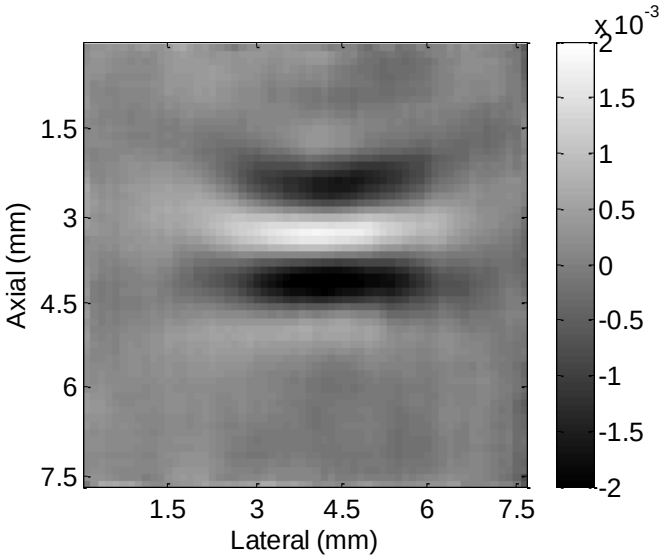

Figure 4.2. A reconstructed 2D (x-z or lateral-and-axial) image of the pressure distribution of an ultrasound pulse at the focal zone generated from a 1 MHz UST and measured using the wire -based system. The diameter of the metal wire is 0.46 mm.

Fig. 4.2 shows the reconstructed image of the ultrasonic pressure distribution in x-z plane at the focal area of the 1 MHz UST (x: lateral direction and z: axial direction) using the wire-based system. The pulse generator/receiver sends a narrow and negative high voltage pulse to excite the UST. Thus, a narrow ultrasonic pulse is generated and propagates in the water. When the pulse reaches the small metal wire, it is partially reflected and detected by the same UST and the pulse generator/receiver. Therefore, the measured signal represents the convolution between the excitation voltage pulse and the impulse response functions of the UST, the wire reflection, the same UST and the receiver amplifier. Similar to the hydrophone system, the signal bandwidth is determined by the UST's bandwidth and the image shown in Fig.4.2 represents a snapshot of the ultrasonic wave propagation around the focal zone of the UST. The gray scale of the image represents the distribution of the ultrasound pressure field in the focal zone in water. The pressure wave approximately has one sinusoidal cycle of a 1-MHz wave (the white and black shows the positive and negative pressure, respectively). This is due to the excitation voltage signal which is a very narrow electronic pulse and the UST has a central frequency of 1-MHz with a limited bandwidth (~50% of the central frequency).

The wavelength observed from the distance between the two negative peaks along the axial direction is about 1.53 mm, which is close to the value measured using the hydrophone system. Video 2 shows the corresponding dynamic propagation of the reconstructed ultrasound pressure wave in the UST focal zone. These results show that the wire-based system can also accurately reconstruct the dynamic ultrasonic pressure distribution. It is to be duly noted that the images in Fig.4.2 and Video 2 have much better SNR than those measured by the hydrophone-based system in Fig. 4.1 and Video 1. This is mainly due to the fact that voltage applied across the UST in the wire-based system is much higher (~ -120 V) than that in the hydrophone-based system (1 V, for avoiding potential damage to the hydrophone).

Similar experiments were carried out using a 2.25 MHz (V305, Olympus NDT) and a 5 MHz (V308, Olympus NDT) UST and a small needle with a diameter of 0.25 mm. The results of the corresponding snapshots are shown in Fig. 4.3 and 4.4, respectively, and the wave dynamic propagations are shown in Video 3 (a) and (b), respectively. It is to be duly noted that since the focal zone is significantly reduced while increasing the ultrasound frequency, the scanning areas are correspondingly reduced from 7.62 ×7.62 mm^2 for 1 MHz UST to 2.54 × 5 mm^2 and 3.81 × 2.54 mm^2 for 2.25 and 5 MHz USTs, respectively. The ultrasound wavelength (along the axial direction) is about 0.61mm and 0.38mm for 2.25 and 5 MHz, respectively, which agrees with the calculated values of 0.66mm and 0.29 mm, respectively, when assuming the ultrasound speed is 1.48 mm/□s in 20 °C water. However large error occurred for 5 MHz UST. This happens because the focal zone becomes smaller when the frequency is increased.

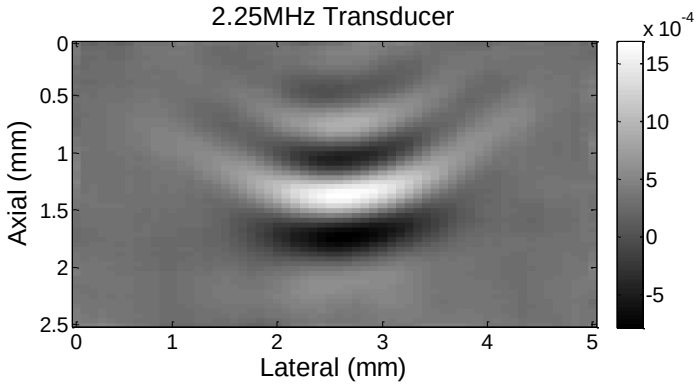

Figure 4.3. A reconstructed 2D (x-z or lateral-and-axial) image of the pressure distribution of an ultrasound pulse at the focal zone generated from a 2.25 MHz UST and measured using a metal wire of diameter 0.25mm.

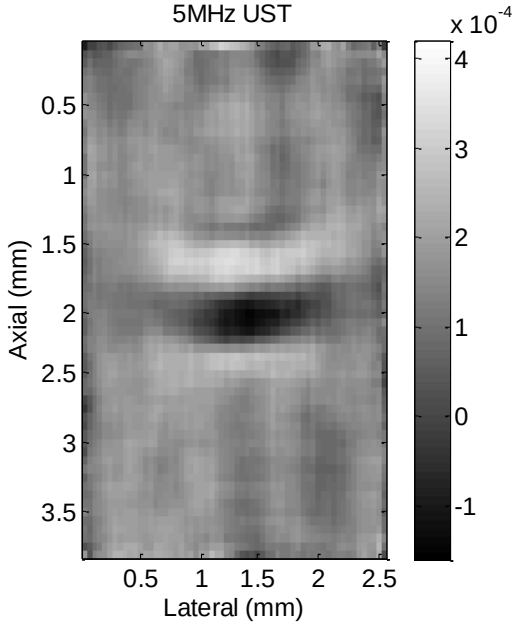

Figure 4.4. A reconstructed 2D (x-z or lateral-and-axial) image of the pressure distribution of an ultrasound pulse at the focal zone generated from a 5 MHz UST and measured using a metal wire of diameter 0.25mm.

Advantages of using the wire-based setup are: (1) no hydrophone is needed, which is valuable because a needle hydrophone is usually vulnerable when exposing to ultrasound pressure long time, (2) a small diameter metal wire is usually available; and (3) the lateral spatial resolution may be improved by reducing the diameter of the wire, which is much easier than reducing the hydrophone diameter. The disadvantage is that (1) a pulse generator/receiver is needed, which is expensive if a

commercial one is adopted, such as the one used in this study (~$3,500). However, a homemade pulse generator/receiver may be much cheaper than a commercial one by reducing those unnecessary features; (2) when processing the data, the round trip traveling of the sound has to be considered; and (3) as the ultrasound pulse is generated and detected by the same UST, the actual ultrasound wave is convolved with the impulse response function of the UST twice, compared to ultrasound wave convolved just once in case of hydrophone system. Because the UST usually has much narrower bandwidth than the hydrophone, the second convolution of the ultrasound pressure with the impulse function of the UST may further significantly distort the actual ultrasound pressure than the convolution of the ultrasound pressure with the impulse response function of the hydrophone. However, this problem can be overcome if one can de-convolve the measured signal from the response function of the UST or hydrophone.

Table 4.1. Summary of Hydrophone based and wire based system used to quantify Ultrasound pressure wave.

System Design	Ultrasound Model	Resolution		Scan area (mm)	Advantages	Limitations
		Lateral (mm)	Axial (mm)			
Hydrophone based	1MHz-V314	1.3	13.69	25 X 7.5	• Measures absolute pressure. • Data processing is simple. • Very sensitive to low pressure waves. • Less expensive than pulse generator/receiver.	• Sensing region is about 200μm. • Demands care in handling. • High pressure might lead to decommission.
	2.25MHz-V305	0.73	6.77	13 X 4		
	5MHz-V308	0.4	3.48	13 X 2.5		

Table 4.1 – continued

Wire based	1MHz-V314	1.53	14.76	20 X 7	• Wire of smaller diameter can be used thereby increase lateral resolution. • Easy to handle. • High pressures can be used thereby increasing SNR.	• Data processing is complicated (in order to include round trip of echo). • High voltages need to be applied to get high pressure waves for better SNR. • Signal recorded is convolved twice with Impulse response of Ultrasound transducer.
	2.25MHz-V305	0.82	8	11 X 4		
	5MHz-V308	0.53	3.88	5 X 2		

4.3 Reconstruction of ultrasonic wave interference using the hydrophone-based system

The interference between two ultrasonic waves is an important phenomenon and efficiently reconstructing the dynamics of ultrasound interference is one of the goals of this study. In the setup, two synchronized output channels from the FG were used to simultaneously drive two 2.25 MHz unfocused UST (Olympus Panametrics A306S-SU), respectively. The two USTs were arranged perpendicular to each other. The USTs were pulsed at 2.25 MHz with 1 cycle per pulse, with a peak-to-peak voltage of 1.8 V. The hydrophone was positioned at the intersection of the two ultrasonic waves confirmed by manually tuning the translation stages on which one of the UST was mounted. Similarly, the dynamic ultrasound pressure distribution can be reconstructed by scanning the hydrophone in an area of interest 2.5x2.5 mm^2. Fig. 4.5(a) and video 4 shows a snapshot and the dynamic propagation of the reconstructed interfered pressure field, respectively. It is quite evident that the two waves have interfered (see Fig. 4.5(a)) and a small negative pressure region (surrounded by a few positive pressure areas) is formed. As the two waves are propagating towards their own

directions (see the two small arrows in Fig. 4.5(a)), the interfered pressure area travels along the diagonal direction (see the large arrow in Fig. 4.5(a)) of the two directions.

In an attempt to investigate the effect of tissue heterogeneity on the interference field, a piece of chicken breast tissue was added in front of one of the ultrasound transducers. Fig. 4.5(b) along with video 5 and Fig. 4.5(c) along with video 6 respectively show the results when a piece of chicken breast tissue with a thickness of 12 and 24 mm is placed in front of one of the transducers. Through visual observation of these Fig.4.5 (a, b and c) and video 4,5 and 6, it can be seen that the chicken breast tissue has very little effects on the patterns of the ultrasound pressure waves and their inference. When the thickness is increased to 24 mm, the ultrasound amplitude is significantly reduced due to the tissue absorption and reflection. However, the patterns of the waves and their interference are less affected. This is understandable because acoustic waves usually are much less scattered by biological soft tissue compared with optical waves.

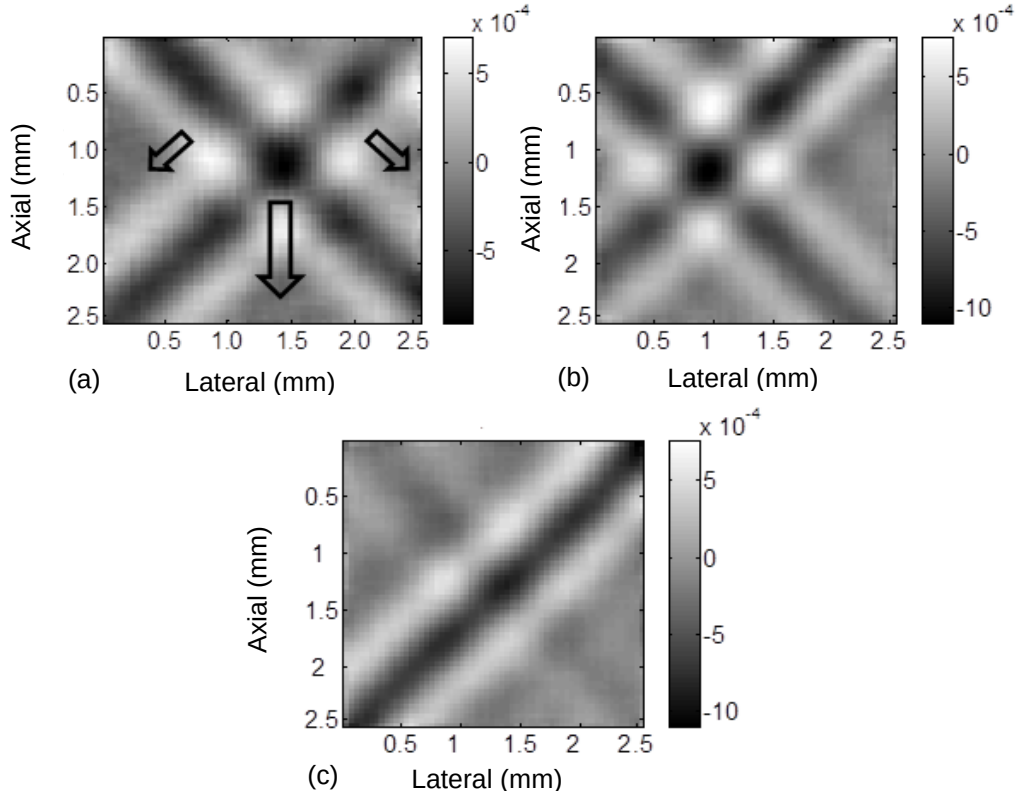

Figure 4.5. Interference of ultrasound pressure waves at 2.25 MHz (a) no chicken tissue, (b) with a piece of chicken breast tissue (12 mm in thickness) in front of the right transducer, and (c) with a piece of chicken breast tissue (24 mm in thickness) in front of right transducer.

4.4 B-mode imaging using single element ultrasound transducer:

Fig.4.6 shows the acquired RF raw data from a water filled tube. Fig. 4.6 (a) represents a RF A-line along the white dotted line indicated in Fig. 4.6 (b). When scanning the transducer laterally multiple A-lines were acquired and their envelopes (as represented by the dotted line over the solid line in Fig 4.6(a)) were displayed in Fig.12 (b) to form a B-mode image and the cross-section of the tube can be visualized. Clearly, four boundaries were resolved. The top and the bottom lines represent the outer boundaries of the tube, and the two lines in the middle represent the inner boundaries of the tube. Due to the attenuation of the ultrasound energy when the pulse propagates from the top to the bottom, reflected peaks from the boundary are usually attenuated exponentially. To compensate this attenuation, the RF raw data are usually compensated by multiplying a gain

factor that is exponentially increasing as a function of the depth. The compensated results are shown in Fig.4.7. From this Fig., the outer and inner diameter of the tube was calculated to be around 0.9 mm and 3mm, respectively, which is close to the actual inner (0.79 mm) and outer (2.39 mm) diameter of the tube. Moreover, it can be seen clearly that the lateral resolution of the image is lower than the axial resolution due to the finite lateral size of the focal zone of the UST.

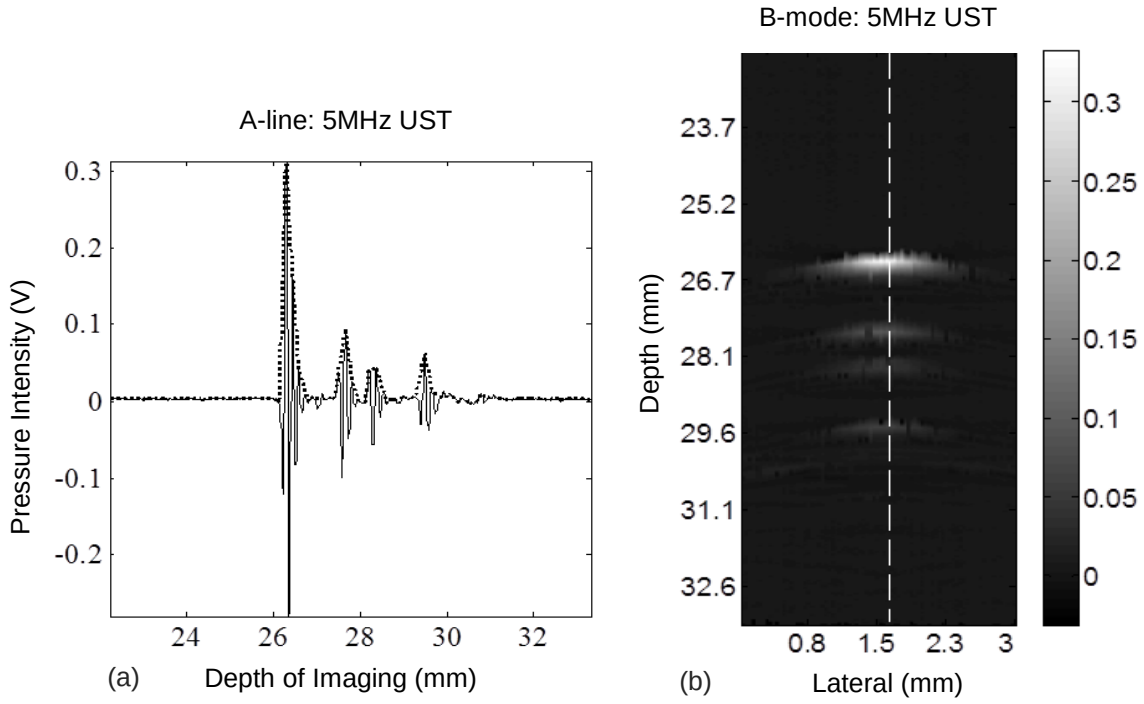

Figure 4.6 (a) An A-line of raw ultrasound signal and its envelop (recorded along the white dotted line shown in (b)) when the tube is filled with water. (b) a B-mode ultrasound image of the cross section of the tube form by displaying multiple envelopes of the A-lines in (a) that were acquired by scanning the ultrasound transducer laterally.

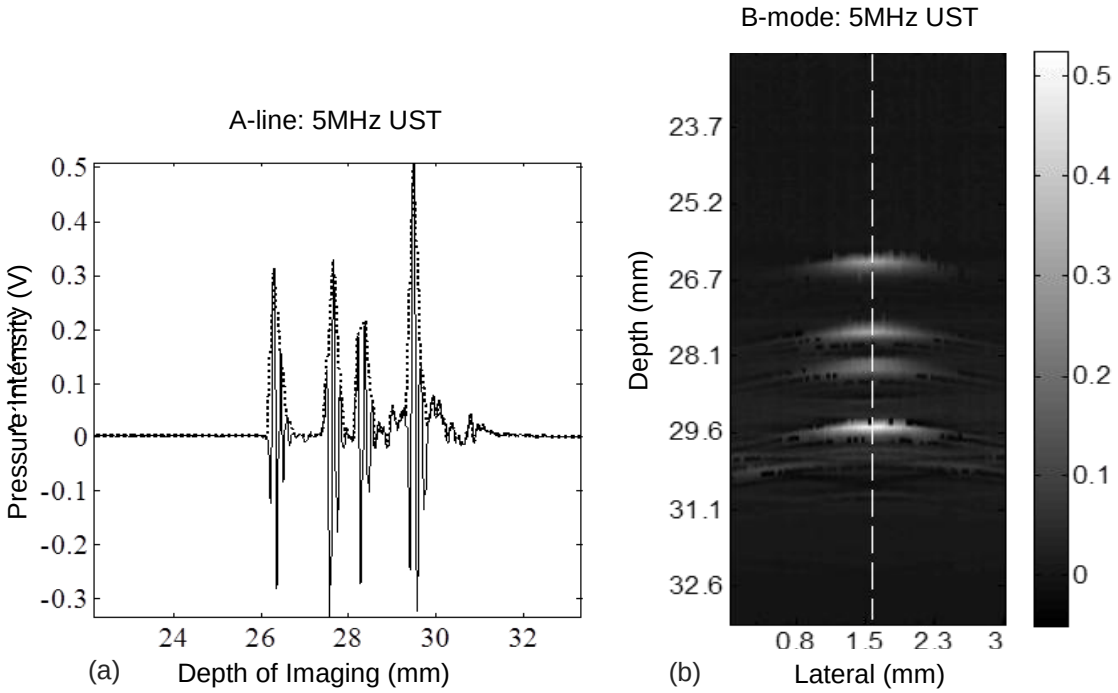

Figure 4.7 The compensated results of Fig.12: (a) A-line and (b) B-mode image.

After draining the water from the tube and leaving air inside, B-mode ultrasound image acquired and is shown in Fig. 4.8. Undoubtedly, the majority ultrasound energy is reflected at the air boundary (see the peak of the A-line in Fig. 4.8(a) as well as the bright line in Fig. 4.8(b)). Moreover an ultrasound shadow is formed below the air layer, which makes the echo from third and fourth boundaries to be very weak or invisible as shown in Fig. 4.8(b). Microbubbles have been considered to be a good ultrasound imaging contrast agents and the following experiment and its respective results are used for validation. Fig. 4.9 shows the results when injecting a high concentration of approximately 9% microbubbles (by mixing 0.02 ml 99.8% microbubble solution to 0.2 ml buffer solution), which is about 1.8 x 10^8 particles/ml). (Targestar-SA, Targeson Inc., mean diameter 2.36±0.03 μm). Image thus obtained is similar to the one acquired using the tube filled with air. This is mainly due to the high concentration of microbubbles reflects large amount of ultrasound energy. While decreasing the microbubble concentration to approximately 2.5% (by mixing 0.02ml 99.8% microbubble solution to 0.6 ml buffer solution), which is about 0.58 x 10^8 particles/ml, the reflected

signal from microbubbles becomes weak (see Fig.4.10). Thus, all the four boundaries tend to be visible again. More importantly, a small signal peak can be observed between the 2nd and 3rd boundaries (see the circle in Fig. 4.9(a)), which is due to the reflection of microbubbles. Correspondingly, compared with Fig.4.10 (b) where almost no microbubbles are used, a small hot spot occurs right below the 2nd boundaries (indicated by dotted circle in Fig. 4.10(b)), which is due to the reflection of microbubbles. Since microbubbles are filled with gas, the density is much lower than the surrounding liquid. Hence we can speculate that most microbubbles are accumulated at the top of the tube.

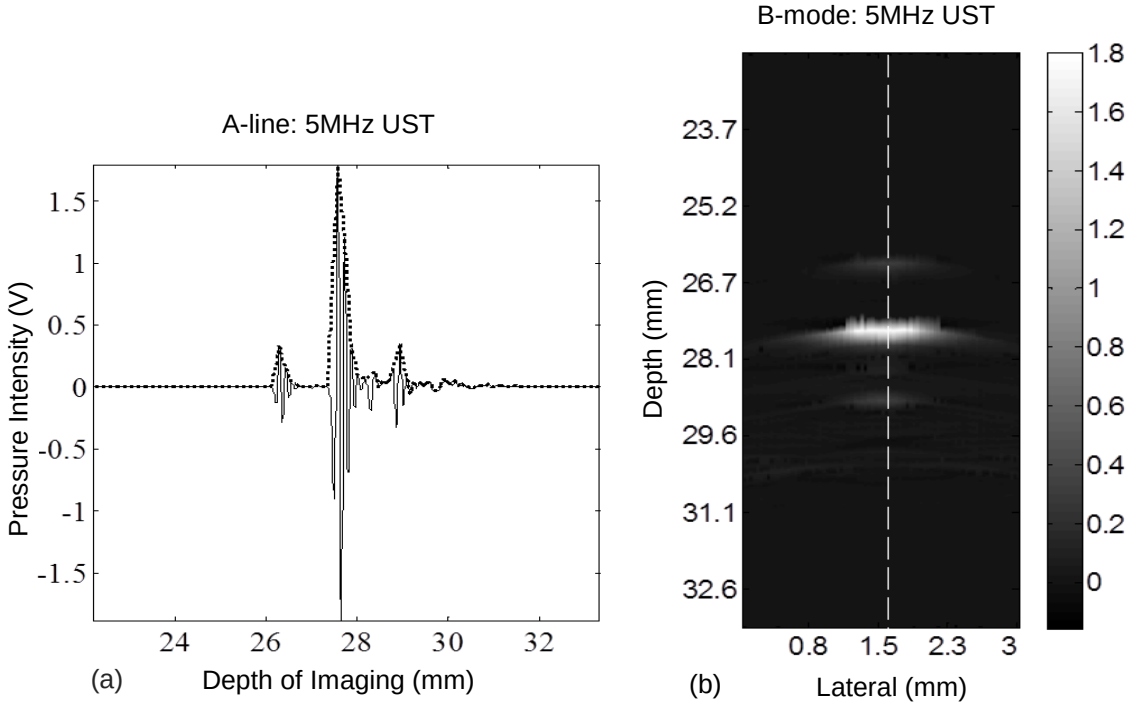

Figure 4.8 (a) An A-line of raw ultrasound signal and it's envelop (recorded along the white dotted line shown in (b)) when the tube is filled with air, and (b) the corresponding B-mode ultrasound image of the cross section of the tube.

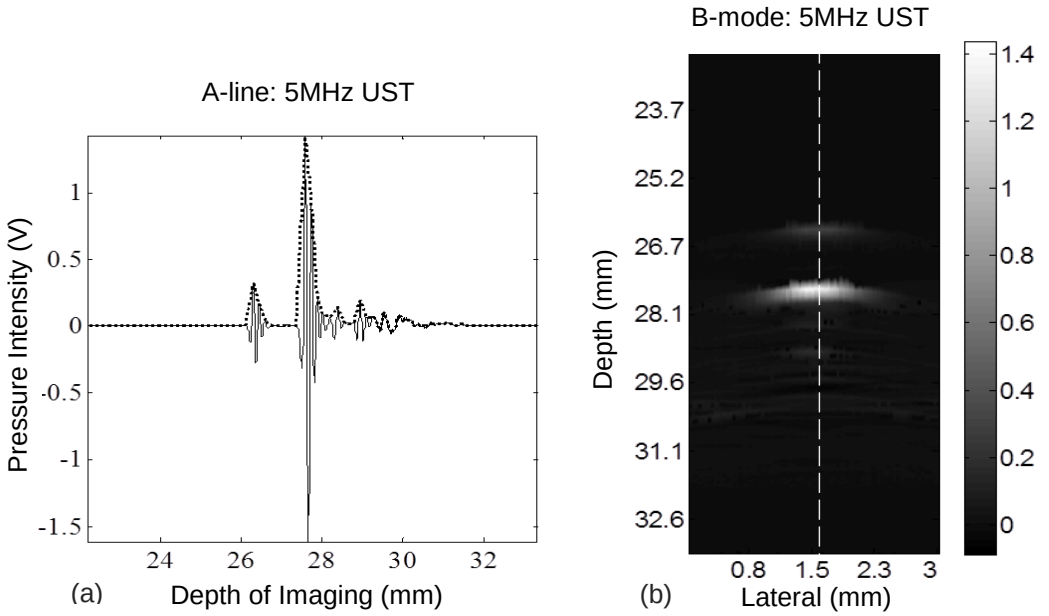

Figure 4.9 (a) An A-line of raw ultrasound signal and its envelop (recorded along the white dotted line shown in (b)) when the tube is filled with a high concentration of microbubble solution (~9%), (b) The corresponding B-mode ultrasound image of the cross section.

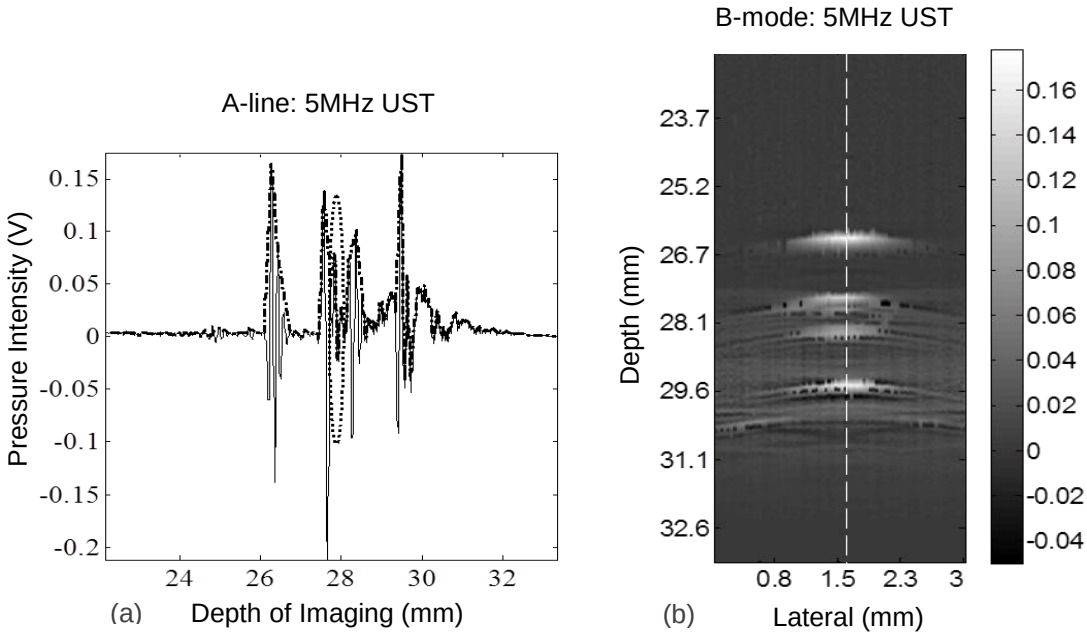

Figure 4.10 (a) An A-line of raw ultrasound signal and its envelop (recorded along the white dotted line shown in (b)) when the tube is filled with a low concentration of microbubble solution (~2.5%), (b) The corresponding B-mode ultrasound image of the cross section.

4.5 HIFU Focal zone measurement:

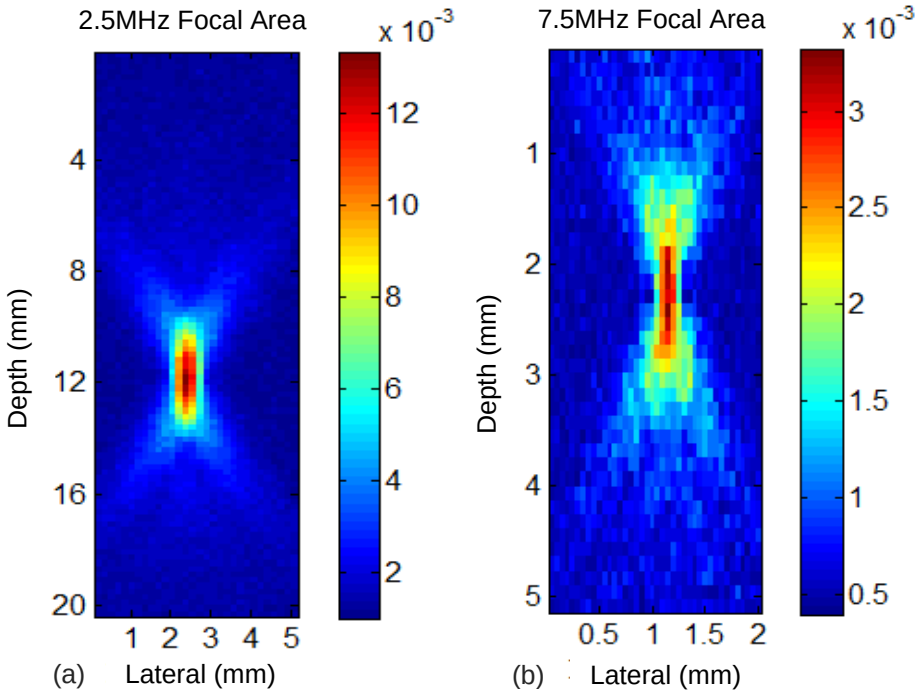

Figure 4.11 Focal zone measurement of HIFU driven at (a) 2.5MHz and (b) 7.5MHz.

Fig. 4.11 depicts the focal zone measurement acquired using hydrophone (HNP-0200) based system employed for measuring HIFU driven at 2.5MHz (Fig 4.11(a)) and 7.5MHz (Fig 4.11(b)). In order to calculate the FWHM of the reconstructed focal zone, the line profile along the lateral and axial direction across the scan point with maximum amplitude is extracted from each of these images and are shown in Fig. 4.12(a) and 4.12(b) for HIFU driven at 2.5MHz and 7.5MHz respectively. The following table 4.2 provides evidence of an agreement between the calculated FWHM and the device specifications provided by the HIFU Company, Sonic Concepts.

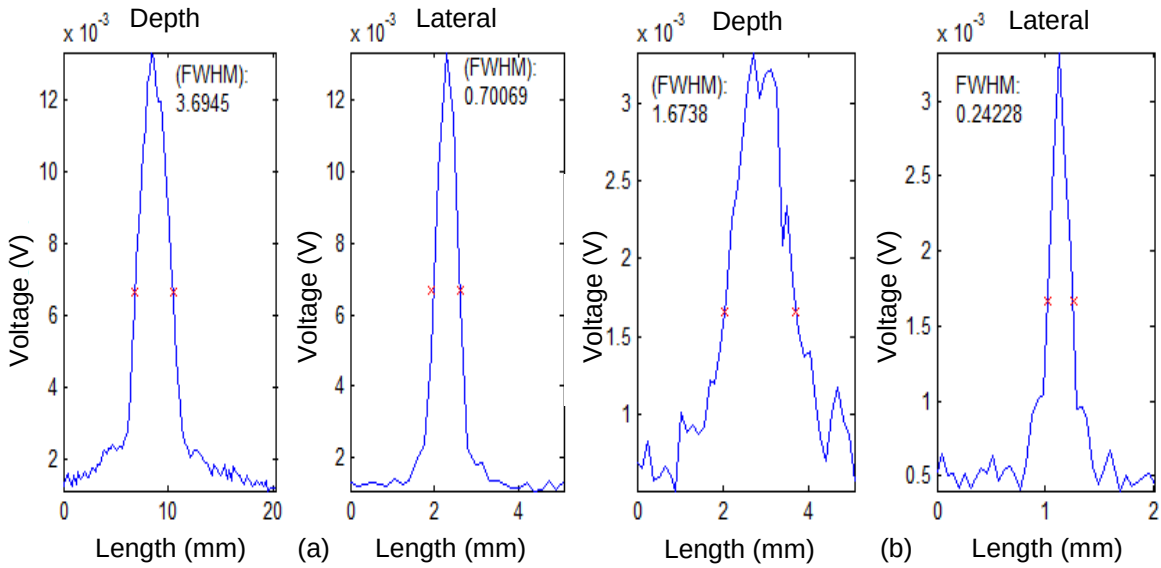

Figure 4.12 Line profile illustration along the Lateral and Axial direction when HIFU was driven at (a) 2.5MHz and (b) 7.5MHz

Table 4.2 Comparison between actual and calculated FWHM for HIFU driven at 2.5MHz and 7.5MHz

	2.5MHz		7.5MHz	
	Lateral (mm)	Axial (mm)	Lateral (mm)	Axial (mm)
FWHM provided	0.5	3.5	0.16	1.5
FWHM calculated	0.7	3.69	0.24	1.67

It is evident that the sensing part of HNP-200, which is about 200μm, is sufficient enough to reconstruct the focal zone distribution along the direction of ultrasound waved propagation. This also provided the visualization of the focal area in which maximum power is delivered using the HIFU inside the TMM thereby providing information of the area where the temperature measurement needs to be performed.

4.6 HIFU Temperature measurement:

4.6.1 Calibration Results:

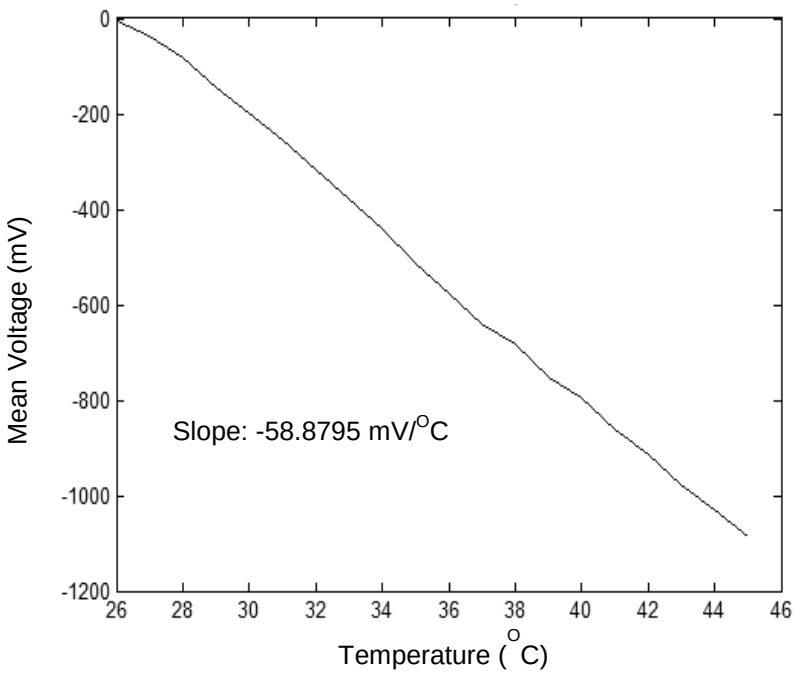

Figure 4.13 Thermocouple (diameter=0.002inch) response (mV) for varying temperature.

The above Fig. 4.13 is a measure of voltage response, recorded using a oscilloscope (DPO 7254, Tektronix, OR, USA), to varying temperature maintained using bench-top temperature controller (CSi32, Omega, CT, USA). The voltage recorded is the mean voltage thereby reducing the error in recording however, it has to be noted that this measurement is not recorded at absolute temperature but at a relative temperature change thus indirectly measuring the relative voltage change which can be considered to be reliable. The experiment design from the thermocouple embedded inside the TMM to the DAQ must not be changed for measuring the heat profile using 2.5MHz and 7.5MHz driven HIFU which implies that the parameters of filter and gain must be kept standard while undertaking other measurement using the same diameter thermocouple. This voltage

change per degree of temperature can be used to convert the absolute voltage recorded across the thermocouple embedded inside the TMM.

4.6.2 HIFU exposed thermocouple response:

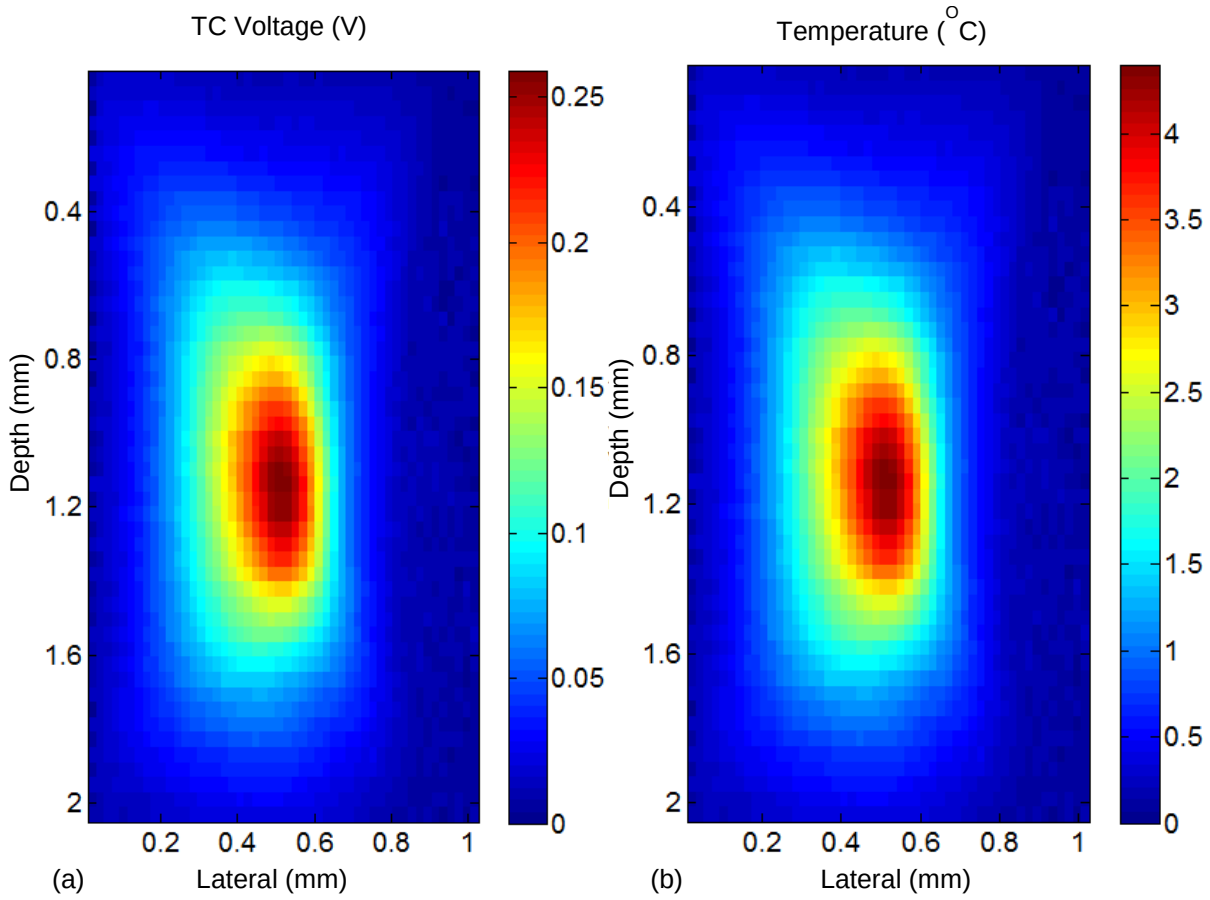

Figure 4.14 (a) Thermocouple voltage signal recorded using DAQ card along XZ plane, (b) Corresponding Temperature converted Heat profile.

The 4.14 and video 7 shows the heat profile distribution along the ultrasound propagation direction (along XZ plane). The thermocouple used is a Type E CHCO-002 with the scan area of about 0.04 X 0.08 inch (XZ) which is about 1 X 2 mm and the step size is about 20 X 40 μm respectively. The HIFU is driven at 7.5MHz, 100mVpp sinusoidal signal in burst mode. The burst width is about 750,000 cycles (100ms 'ON') with trigger interval about 2 seconds. The signal that is

recorded is post processed to find the maximum and minimum between initial data point and the data point before 110ms since the exposure time is about 100ms. The signal is digitally filtered with low pass filter about 10Hz to remove the noise and the line frequency if captured by the exposed leads of the thermocouple. It can be observed from the above figure that the heat distribution is quite smooth; therefore no smoothing algorithm is applied to the signal. Later in order to find the FWHM of the image along the depth and lateral direction, the location of the maximum signal in the image is used and the line profile along the respective directions are extracted, then using interpolation technique the corresponding length between the half maxima is calculated. Its line profile along with FWHM is shown in the Fig 4.15 with their respective raw voltage and temperature. It can be observed that the FWHM is about 0.35mm and 0.85mm along lateral and depth respectively compared to companies specifications about 0.16mm X 1.5mm respectively. Since the results are convolution of the point spread function of system and the diameter of the thermocouple (about 50μm) which are close. The original goal of having 4 degree increase was achieved using these specifications when driven at 7.5MHz frequency.

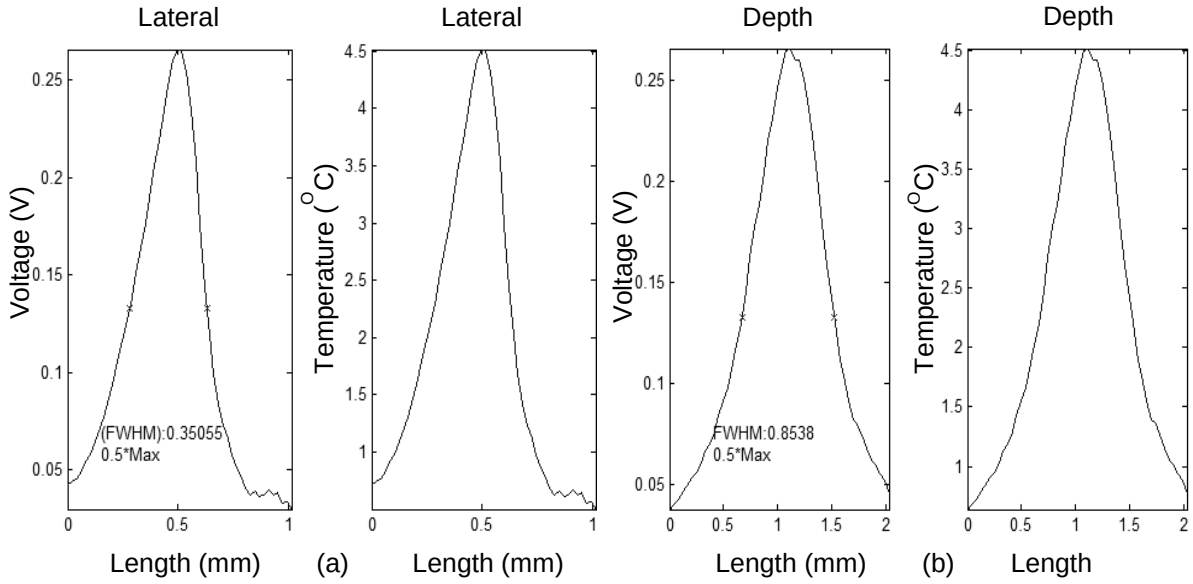

Figure 4.15 (a) Voltage (V) and temperature (^{O}C) along lateral (X) direction with FWHM (b) Voltage (V) and temperature (^{O}C) along depth (Z) direction with FWHM

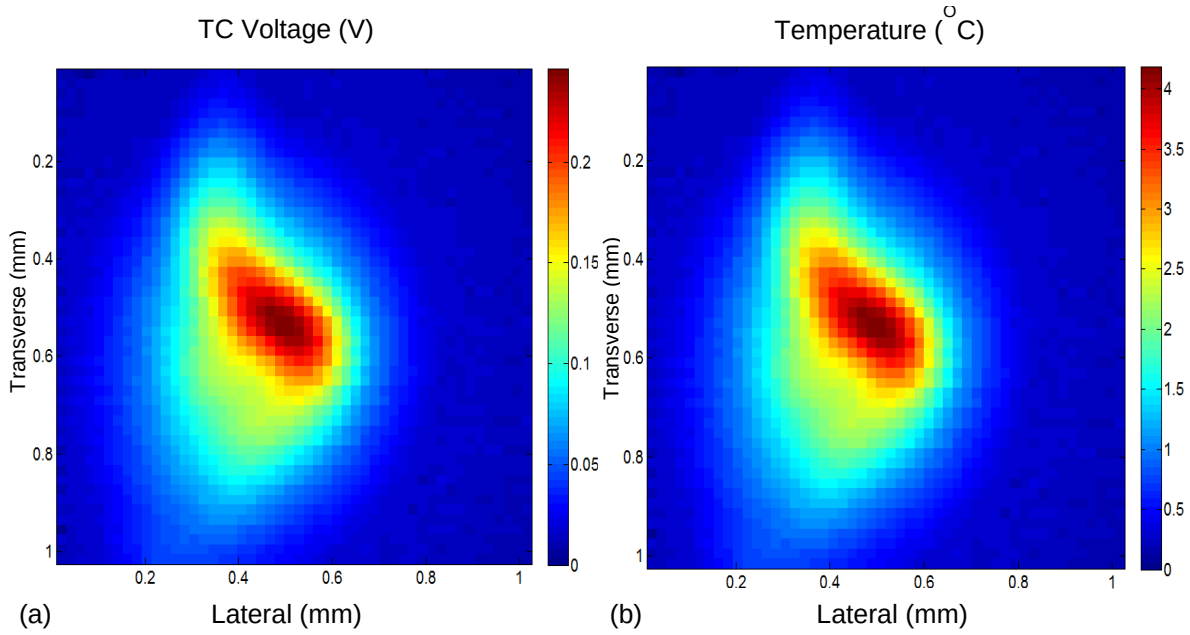

Figure 4.16 (a) Thermocouple voltage signal recorded using DAQ card along XY plane, (b) Corresponding Temperature converted Heat profile.

The Fig.4.16 and video 8 shows the heat profile distribution plane perpendicular to the ultrasound propagation direction (along XY plane). The thermocouple used is a Type E CHCO-002 with the scan area of about 0.04 X 0.04 inch (XZ) which is about 1 X 1 mm and the step size is about 20 X 20 µm respectively. The HIFU is driven at 7.5MHz, 100mVpp sinusoidal signal in burst mode. The burst width is about 750k cycles (100ms 'ON') with trigger interval about 2 seconds. The duration of the signal that is recorded and post processed to find the maximum and minimum between initial data point and the data point before is 110ms, about 10ms more than the exposure duration of 100ms. The signal is digitally filtered with low pass filter about 10Hz to remove the noise and the line frequency if captured by the exposed leads of the thermocouple. It can be observed from the above figure that the heat distribution is quite smooth; therefore no smoothing algorithm is applied to the signal. Later in order to find the FWHM of the image along the transverse and lateral direction, the location of the maximum signal in the image is used and the line profile along the respective directions are extracted, then using interpolation technique the corresponding length between the half maxima is calculated. Its line profile along with FWHM is shown in the Fig.4.17 with its respective raw

voltage and temperature. It can be observed that the FWHM is about 0.34mm and 0.38mm along lateral and transverse respectively compared to companies specifications about 0.16mm X 0.16mm respectively. Since the results are convolution of the point spread function of system and the diameter of the thermocouple (about 50µm)which are close. It should also be noted that the placement of the thermocouple fine wire is along the transverse scan direction which might have contributed to broader line profile compared to line profile along lateral direction. But the original goal of having 4 degree increase has been achieved using these specifications if driven at 7.5MHz frequency.

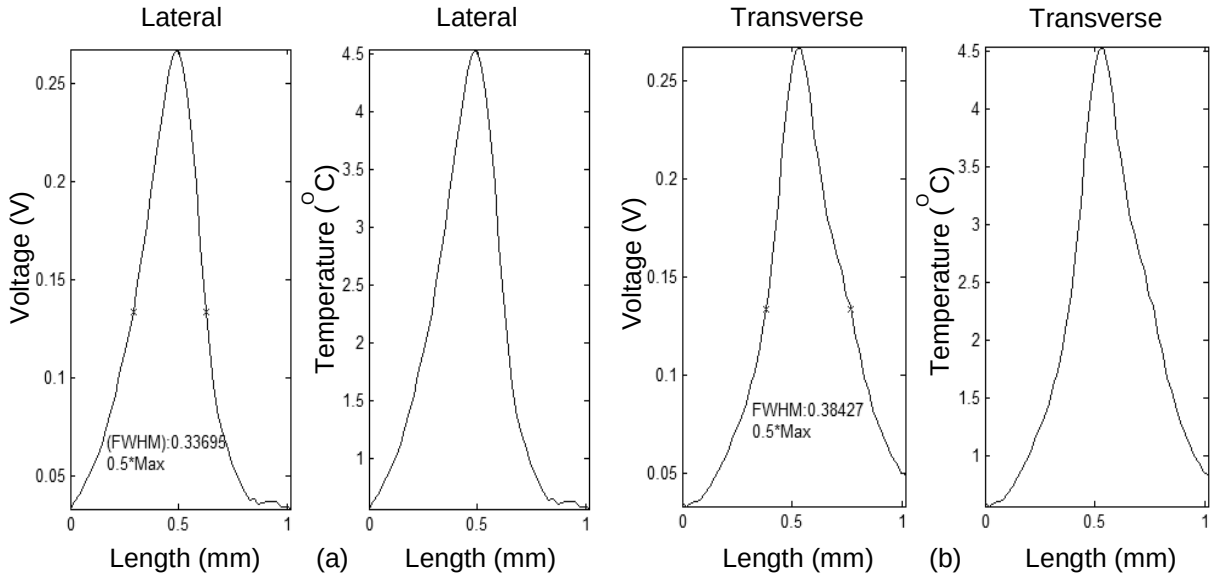

Figure 4.17 (a) Voltage (V) and temperature (^{O}C) along lateral (X) direction with FWHM (b) Voltage (V) and temperature (^{O}C) along transverse (Y) direction with FWHM.

Similar to the experiment conducted for HIFU driven at 7.5MHz the following Fig.4.18 and video 9 illustrates the heat profile distribution of the plane perpendicular to the ultrasound wave propagation direction (along XY plane) keeping the same fine wire thermocouple of diameter 0.002inch when the HIFU was driven at 2.5MHz. The function generator was set at 200mV burst mode, 250k cycles per burst (about 100ms 'ON'), 2 seconds trigger interval between bursts and the output of the RF amplifier is about 100V peak to peak approximately. The pressure calculated was about 6.3 MPa which would explain the immense heat generation observed from the Fig.4.18 (b) for the

corresponding voltage recorded which is shown in Fig.4.18 (a). Similar to previous calculations line profile along both directions are extracted across the scan point where maximum signal is observed from image reconstructed to calculate for FWHM which is shown in Fig.4.19 (a) and (b) representing lateral and transverse directions respectively.

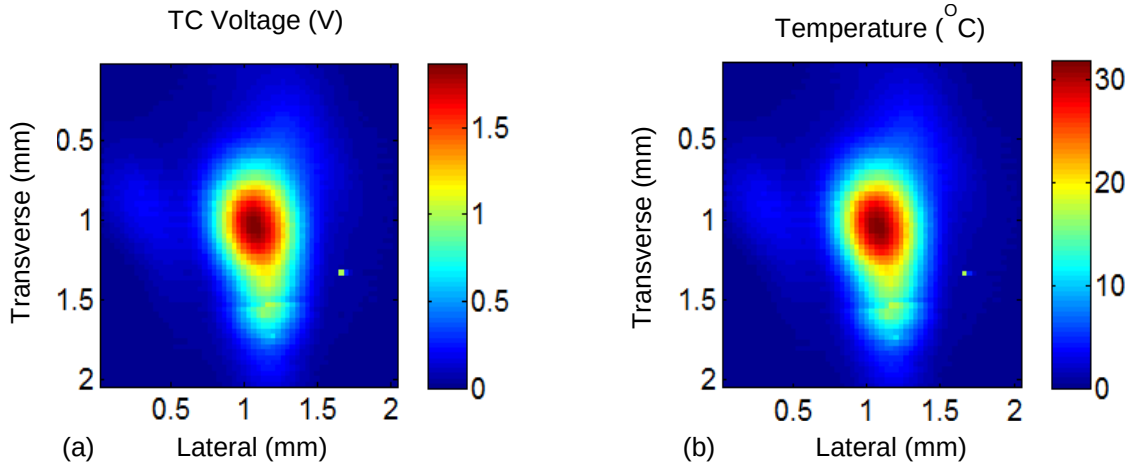

Figure 4.18 (a) Thermocouple voltage signal recorded using DAQ card along XY plane, (b) Corresponding Temperature converted Heat profile.

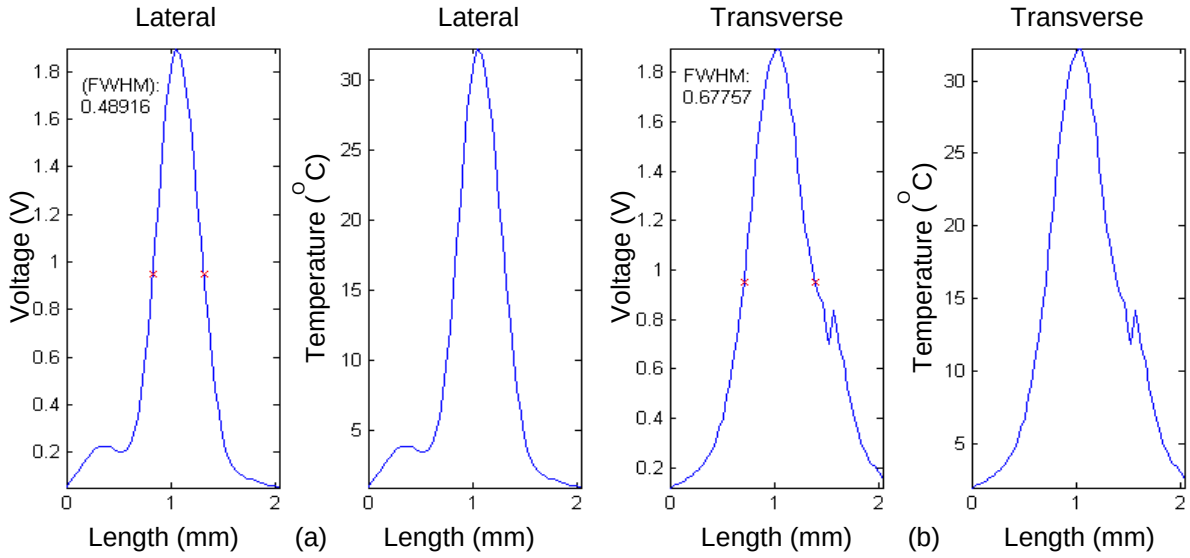

Figure 4.19 (a) Voltage (V) and temperature (OC) along lateral (X) direction with FWHM (b) Voltage (V) and temperature (OC) along Axial (Z) direction with FWHM

Fig.4.20 and video 10 represents the heat distribution profile in XZ plane which is in line with ultrasound wave propagation (along XZ plane) using 2.5MHz to drive the HIFU. Similar parameters are set to drive the HIFU at 2.5MHz. Compared to the experiments discussed earlier, It can be observed that the heat distribution is quite uniform and the FWHM calculated in similar manner as done for previous experiment resembles well with the actual FWHM values as shown in Fig.4.21 (a) and 4.21 (b). Since the image reconstructed showed minor data points shift, which might be due to the moment of the tissue sample caused from the vibration of the motorized translation stage, the image is smoothed using moving average algorithm. The data was then scaled back to its original reconstructed data thereby modifying it in order to maintain the similarity with the non-modified image. From Fig.4.20 it can be observed that majority of temperature increase is concentrated in the focal area and since the phantom more or less resembles the human tissue it is possible to increase the temperature at the desired location non-invasively and also by controlling a combination of excitation voltage, duration of exposure and interval between the exposure. By doing so, we will be able to increase the temperature to desired threshold.

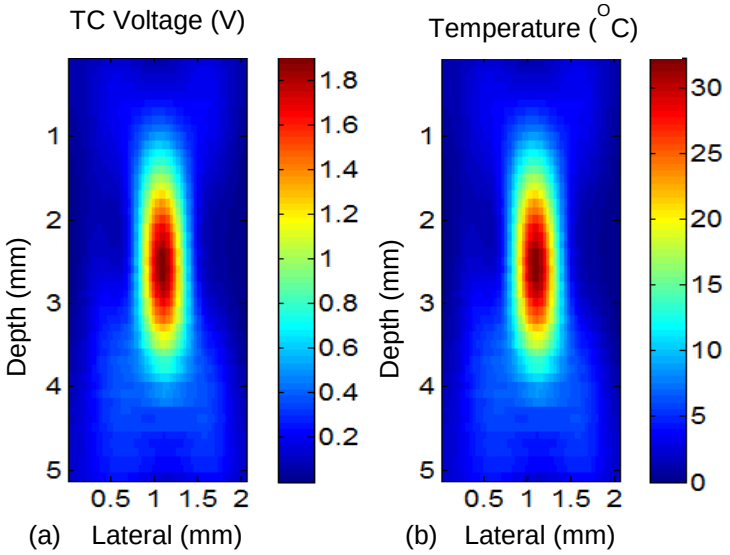

Figure 4.20 (a) Thermocouple voltage signal recorded using DAQ card along XZ plane, (b) Corresponding Temperature converted Heat profile.

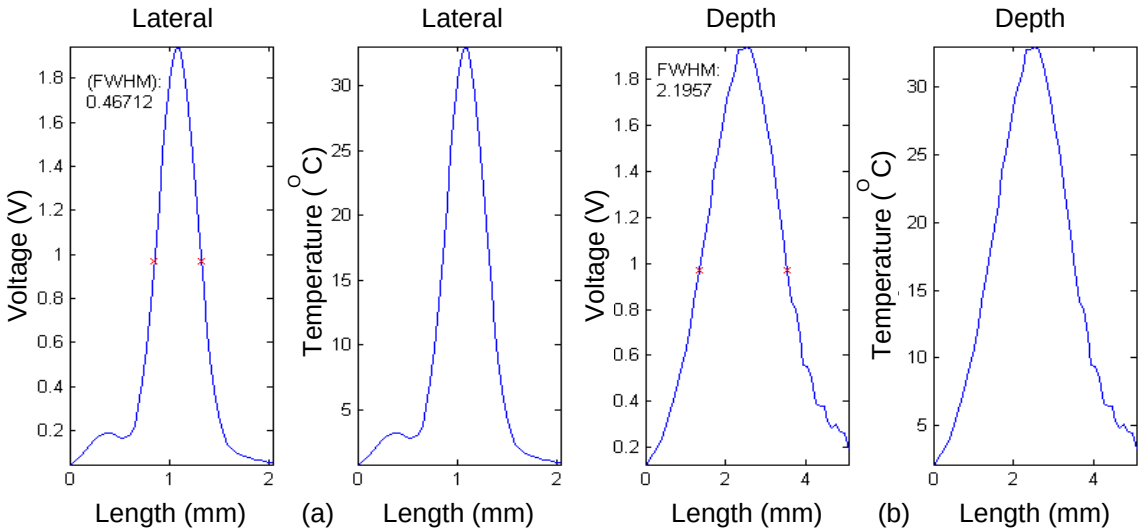

Figure 4.21 (a) Voltage (V) and temperature (OC) along lateral (X) direction with FWHM (b) Voltage (V) and temperature (OC) along Axial (Z) direction with FWHM

4.7 HIFU cavitation detection system:

As described, in chapter 3, the cavitation detection system and the results from B-mode imaging discussed earlier in this chapter, infer that (conclude that) we can use a single element 5MHz transducer whose focus is overlapped with focus of HIFU to detect the HIFU induced cavitation and the design is arranged orthogonal with respect to each other.

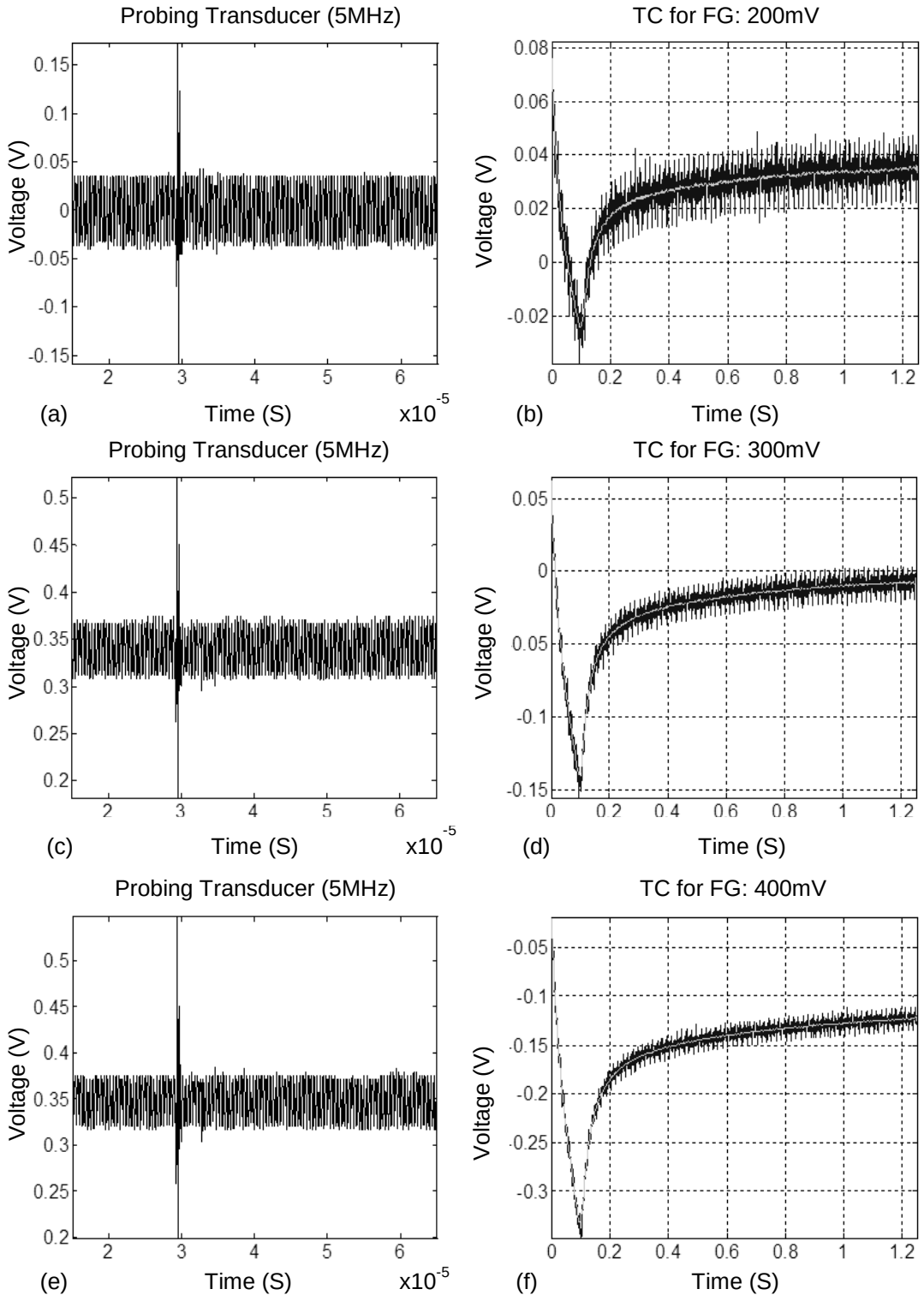

Figure 4.22 Signal recorded by (a, c, e) probing transducer and (b, d, f) across the fine wire thermocouple for increasing voltage from function generator. TC: Fine wire thermocouple.

Using this setup for cavitation detection two experiments were undertaken. Fig. 4.22 (a, b), (c, d) and (e, f) depicts the measurements acquired from probing transducer and the thermocouple respectively. For increasing excitation voltage of 200mVpp, 300mVpp and 400mVpp from the function generator the pressure was indirectly increased from 6.3MPa, 9.5MPz and 12.7 MPa respectively. The delay generator is used to trigger the digitizer to acquire the A-line echo signal from the Pulse generator/receiver after a delay of 100mS thereby acquiring at the end of HIFU exposure. Any cavitation if induced by HIFU sonication would act as a good contrast agent for ultrasound signal and a high amplitude echo signal should be visible in the A-line which has been acquired around the focus of the probing 5MHz transducer. A frequency plot of this A-line would be a better indicator of echo signal; the frequency plot is shown in Fig.4.23. Abnormal high amplitude around 5MHz seen in Fig.4.22 (b, c) does not indicate signal from cavitation because the echo signal from cavitation in general should increase with increasing power and such a behavior is clearly not observed. Hence, it can be concluded that cavitation is not induced by HIFU during this very short 100ms burst exposure.

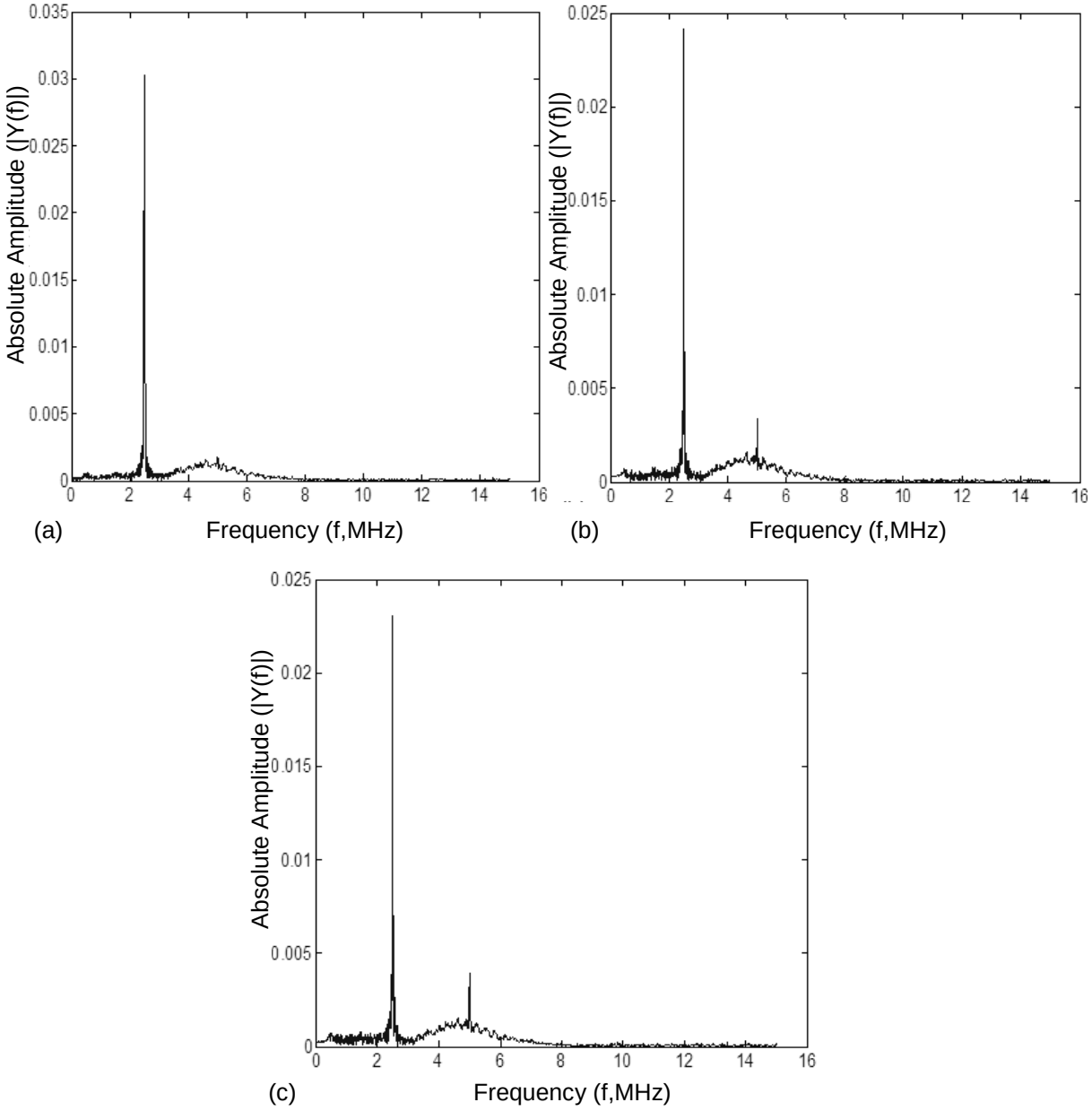

Figure 4.23 Single sided frequency spectrum of signal recorded by probing transducer for HIFU driven at (a) 200mVpp (b) 300mVpp and (c) 400mVpp from function generator.

Using the same setup for cavitation detection, results of the second experiment that was undertaken among is shown in the Fig.4.24 (a, b), (c, d) and (e, f) which illustrates the measurement acquired at three time sequences which are before, during and after the end of HIFU sonication which was about 100ms in duration. The trigger signals given to digitizer were at 90ms, 100ms and 110ms

respectively and this is achieved by using delay generator as discussed in chapter 3. The voltage from function generator was kept at 150mVpp which is about 5 MPa of pressure at the focus of 2.5MHz driven HIFU. It is evident that three A-line during, at the end of; and after the HIFU sonication is captured successfully and the presence of induced cavitation by HIFU is clearly not observed. Additional evidence given by frequency domain of these A-line (shown in Fig. 4.25 (a, b and c)) also validates this conclusion.

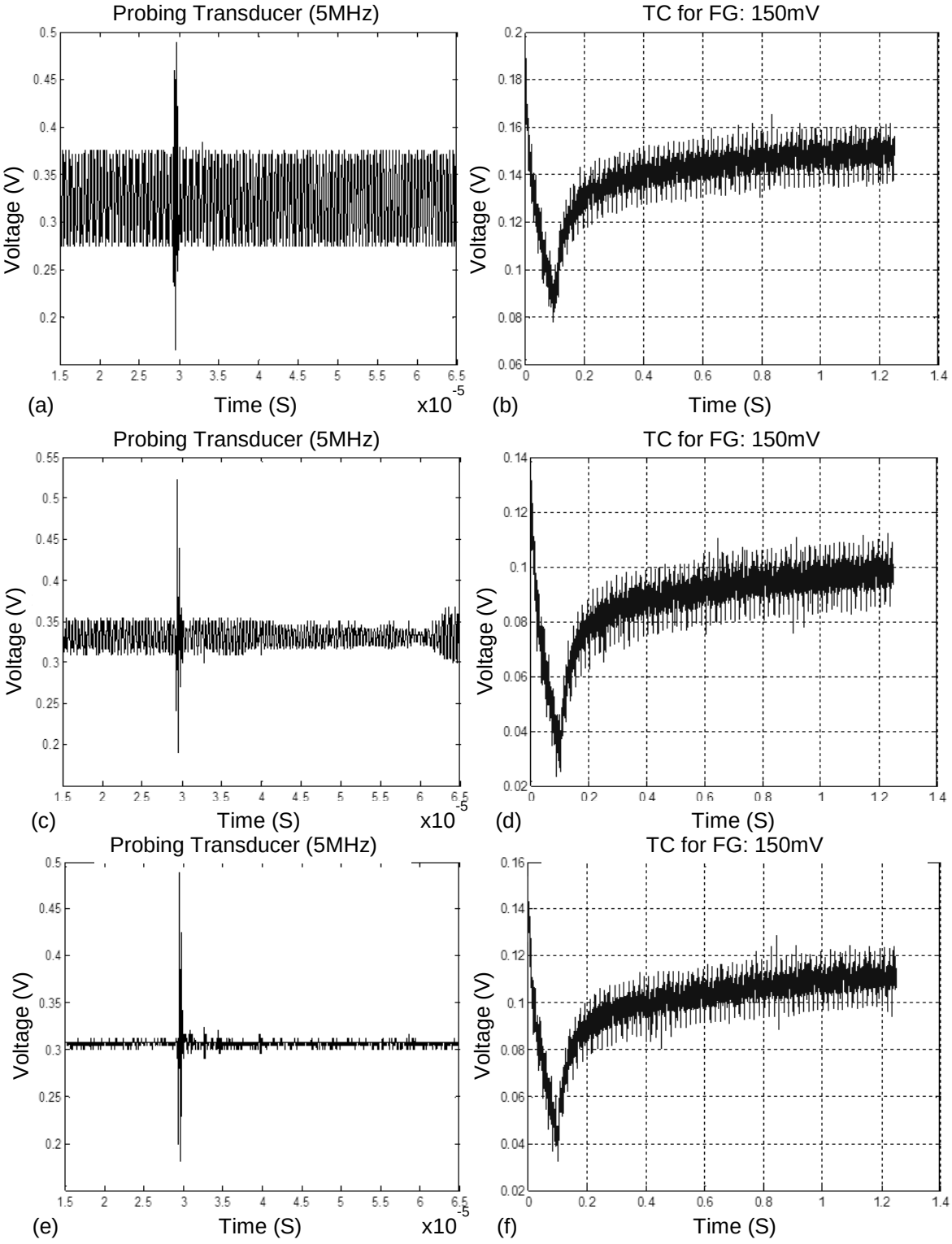

Figure 4.24 Signal recorded by (a, c, e) probing transducer and (b, d, f) across the fine wire thermocouple for increasing voltage from function generator.

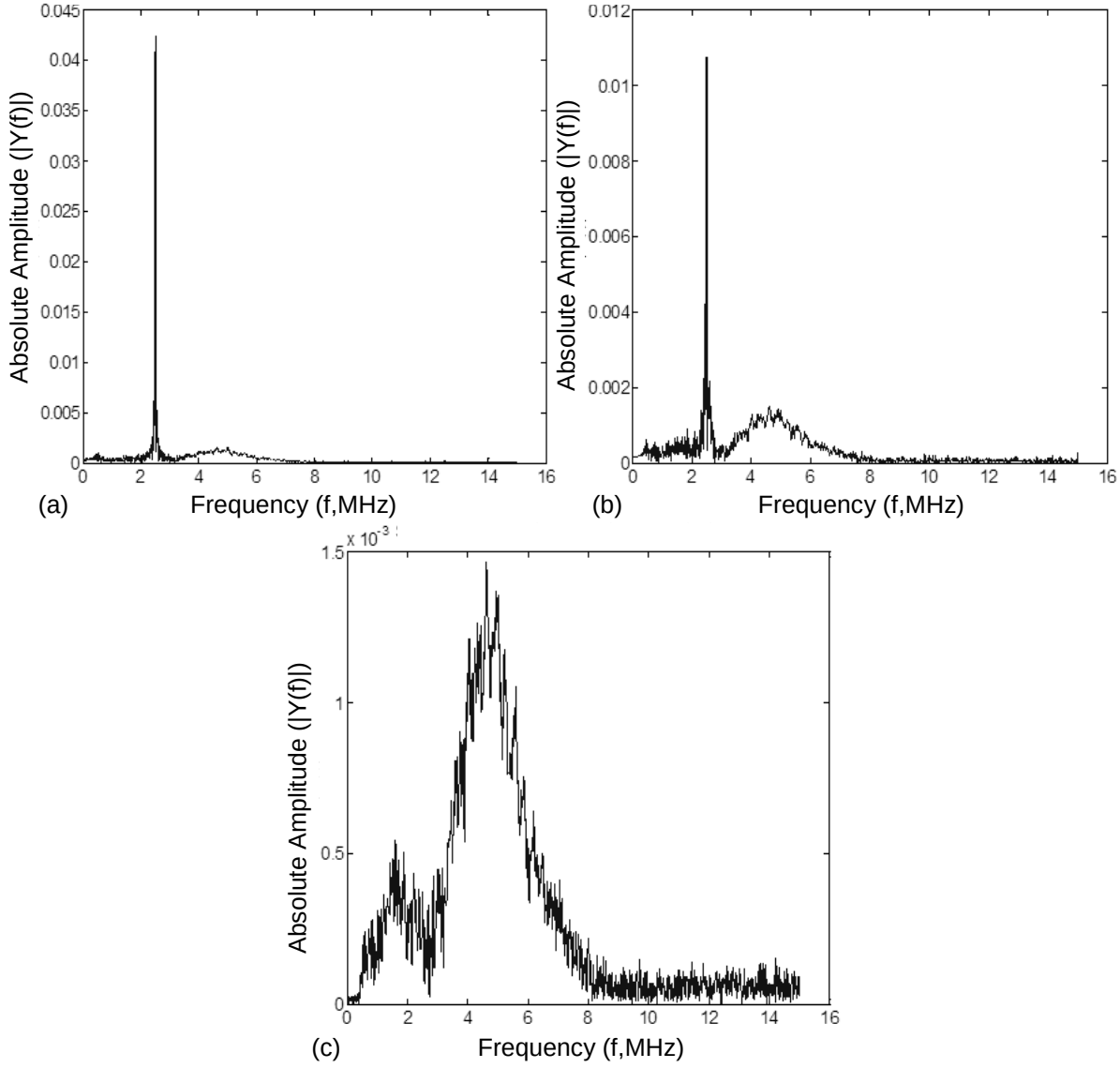

Figure 4.25 Single sided frequency spectrum of signal recorded by probing transducer for HIFU driven at (a) 200mVpp (b) 300mVpp and (c) 400mVpp from function generator.

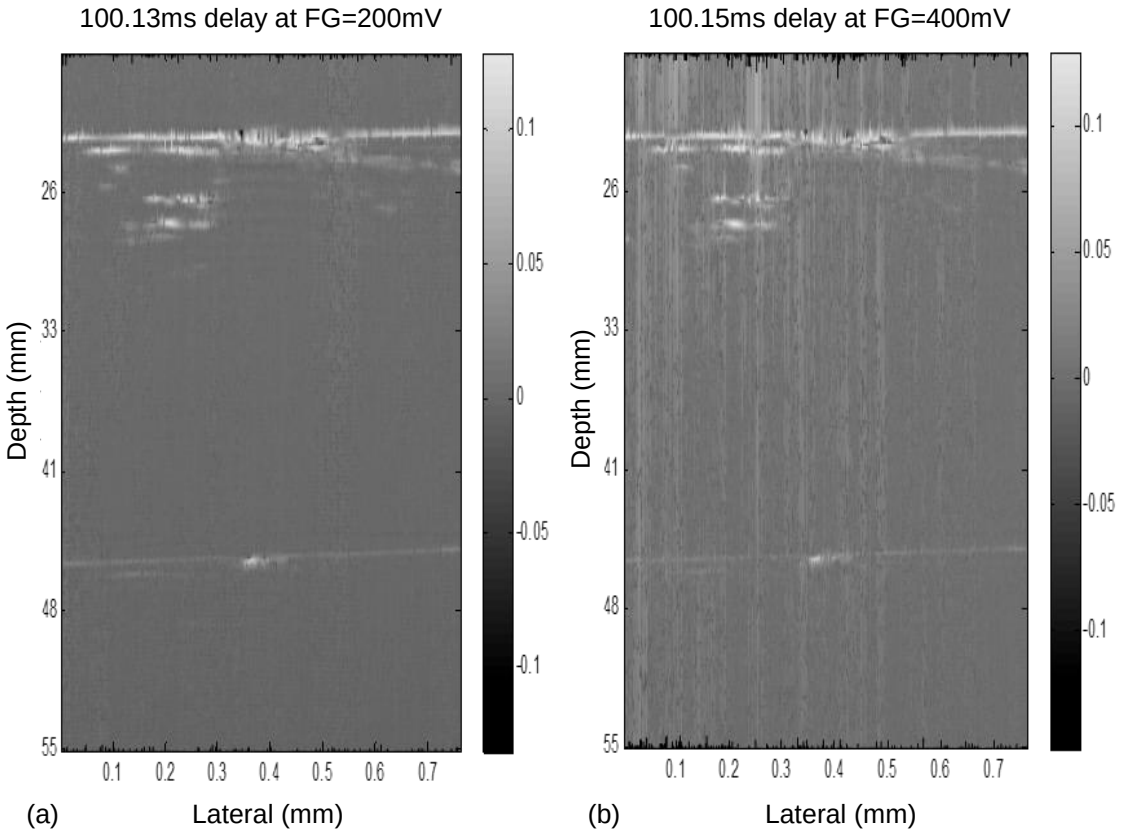

Figure 4.26 B-mode along the fine wire thermocouple embedded in TMM at (a) 200mVpp from FG, about 6.3MPa and (b) 400mVpp from FG, about 12.7 MPa

Fig. 4.26 is a B-mode image of multiple A-line recorded across the fine wire thermocouple embedded inside the TMM during the HIFU sonication. If cavitation was induced B-mode would contain hyperechoic region around the sensing bead of the fine wire thermocouple which is clearly not seen in either of the Fig.4.26 (a) and (b). This also validates the conclusion that the major abnormality of HIFU induced cavitation would not occur while driving the HIFU at very short sonication bursts.

CHAPTER 5

Conclusion and Future work

It has been well illustrated from chapter 2, and its related results in chapter 4, that single element ultrasound transducer with both hydrophone based and wire based system design that we have employed can be used to demonstrate imaging of dynamic nature of Ultrasound pressure wave along with tube based system design for B-mode imaging, thereby verifying that micro-bubbles are the best contrast agent for ultrasound imaging.

Chapter 3 illustrates the combination of techniques demonstrated in chapter 2 thereby modifying the previous system design for the purpose of quantification of temperature profile of a High intensity focused ultrasound transducer in a tissue mimicking material. We were thus able to use the techniques demonstrated to verify presence or absence of abnormality using a single element transducer thereby demonstrating an application based project and the corresponding results in chapter 4 validates these claims.

Overall the study does not impose an improvement over previous studies but proposes a cost efficient, simple and multi-functional system design. This system design when verified with results proves that it can be used to demonstrate the unique feature of ultrasound physics and techniques. Also quantification of HIFU field was successfully demonstrated. Finally the most attractive feature of the overall system design is that it can be adapted for any transducers and also for other ultrasound related studies with minimal modification.

This study can be further extended to quantify the pressure and temperature profile of an HIFU transducer in vitro, and later on can be extended to be used in vivo studies. Before which it would be interesting to use multiple thermocouple which are strategically placed inside tissue mimicking phantom to improve the quantification of HIFU induced temperature profile. In addition, the whole system design proposed can be made to be portable thereby increasing the possible application and making the app to be user friendly. Such problems and many more can be attempted in the future by us and other investigators which will help us explore some other applications that require ultrasound intervention.

APPENDIX A

MATLAB CODES

```
% Created by Jayanth Kandukuri with Yuan Liu assistance under
% supervision of Prof. Baohong Yuan at Optical Bioengineering
 % Laboratory, University of Texas Arlington, Texas, USA.

function varargout = UST_all(varargin)
% UST_ALL MATLAB code for UST_all.fig
%      UST_ALL, by itself, creates a new UST_ALL or raises the existing
%      singleton*.
%
%      H = UST_ALL returns the handle to a new UST_ALL or the handle to
%      the existing singleton*.
%
%      UST_ALL('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in UST_ALL.M with the given input arguments.
%
%      UST_ALL('Property','Value',...) creates a new UST_ALL or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before UST_all_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to UST_all_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help UST_all

% Last Modified by GUIDE v2.5 09-Aug-2012 22:14:54

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @UST_all_OpeningFcn, ...
                   'gui_OutputFcn',  @UST_all_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 - DO NOT EDIT


% --- Executes just before UST_all is made visible.
function UST_all_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 UST_all (see VARARGIN)

% Choose default command line output for UST_all
handles.output = hObject;
%% 1 %%. Initializing the saving folder of the test: the default folder is C;\data\.

%if strcmp(get(handles.savedirectory,'String'),'Enter Experiment Save-to Directory')==1
   answer=questdlg('Please Select a Directory to save Experiment Data. Choose Skip if you
would like a directory specified for you.','System Notification','Browse for
Directory','Skip','Cancel','Browse for Directory');
   if strcmp(answer,'Browse for Directory')
      directory = uigetdir('C:/Lab_folder/Jay/Using_wire/', 'Select Directory to Save Experiment
Data to');
      if directory==0;
         errordlg('No Directory Selected. Please Press Initialize Again.','System Notification');
         return;
      end
      directory(end+1)='\';
      set(handles.savedirectory,'String',directory);
   end
   if strcmp(answer,'Skip')

Edirname=eval(sprintf('''C:/Lab_folder/Jay/Using_wire/USTscanning/Experiment_%s''',datestr(no
w,'mm-dd-yyyy')));
      %Edirname=sprintf('D:/Data/Experiment_%s',datestr(now,'mm-dd-yyyy')); this is the same as
the right above commond
      m=0;
      while eval(sprintf('isdir(''%s'')',Edirname))==1;
         m=m+1;
         if m==1
            eval(sprintf('Edirname=''%s%s''',Edirname,char(96+m)));
         else
            eval(sprintf('Edirname(end)=''%s''',char(96+m)));
         end
      end
      if m==0

eval(sprintf('mkdir(''C:/Lab_folder/Jay/Using_wire/USTscanning'',''Experiment_%s'')',datestr(now,'
mm-dd-yyyy')));

eval(sprintf('set(handles.savedirectory,''String'',''C:/Lab_folder/Jay/Using_wire/USTscanning/Expe
riment_%s/'');',datestr(now,'mm-dd-yyyy')));
         disp('A')
      else

eval(sprintf('mkdir(''C:/Lab_folder/Jay/Using_wire/USTscanning'',''Experiment_%s%s'')',datestr(no
w,'mm-dd-yyyy'),char(96+m)));

eval(sprintf('set(handles.savedirectory,''String'',''C:/Lab_folder/Jay/Using_wire/USTscanning/Expe
riment_%s%s/'');',datestr(now,'mm-dd-yyyy'),char(96+m)));
```

```
        disp('B')
      end
   end
   if strcmp(answer,'Cancel')
      warndlg('''Come on, will ya make a decision already!?'''','System Nofication');
      return
   end
%end

directorycheck=get(handles.savedirectory,'String');

% eval(sprintf('mkdir(''%s'',''normal_scans'')',directorycheck));
% eval(sprintf('mkdir(''%s'',''save_scans'')',directorycheck));
% eval(sprintf('handles.info.directories.normal_scans=''%snormal_scans'';',directorycheck));
eval(sprintf('handles.info.directories.scans=''%s'';',directorycheck));

% eval(sprintf('handles.info.directories.save_scans=''%ssave_scans'';',directorycheck));

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

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

%% 2. NI DAQ Setup (initialization)

%% 2.1 Get the default parameters and set up the parameters for NI-DAQ PCIe-6353 (Dev1).
acquiretime=str2double(get(handles.acquiretime,'String'));
acquirerate=str2double(get(handles.acquirerate,'String'));
samples=acquirerate*acquiretime;

dio=digitalio('nidaq','Dev1');
addline(dio,0,1,'in'); %PFI0/P1.0 as the trigger of both ai/data aquisitoin;
addline(dio,4,2,'out'); %PFI 12/P2.4 as the digital output pin for the trigger;
putvalue(dio.line(2),[0]); %to initiate the value of digital output to be 0;

ai=analoginput('nidaq','Dev1');
duration = 2.5;
ch = addchannel(ai,0:1);
set(ai,'TriggerChannel',ch(1))
set(ai,'TriggerType','HwAnalogChannel')
set(ai,'TriggerCondition','AboveHighLevel')
set(ai,'TriggerConditionValue',3)
set(ai,'samplerate',2e5);
requiredSamples = floor(1e5 * duration);% sample rate multiplied by duration of acquisition
set(ai, 'SamplesPerTrigger', requiredSamples);
% get(ai, 'SamplesPerTrigger')
set(ai, 'TriggerRepeat', 1);
set(ai, 'Timeout',15)
get(ai)
% set(ai,'TriggerType','Hwdigital');
% set(ai,'HwDigitalTriggersource','PFI0'); % this trigger port can be connected to external source;
```

```
ao=analogoutput('nidaq','Dev1');
addchannel(ao,1);
set(ao,'samplerate',1000);
putdata(ao,0);
% set(ao,'TriggerType','Hwdigital');
% set(ao,'HwDigitalTriggersource','PFI0'); % we set the same triggersource for Ao to give out a
trigger signal to PFI0;

%% 2.2 Get the default parameters and set up the parameters for NI-Digitizer 5133 (Dev2).
makemid('niscope');
niScopeObj=icdevice('niscope.mdd','DAQ::Dev2');
connect(niScopeObj);
configuration = get(niScopeObj, 'configuration'); % assign all property names to configuration
NISCOPE_VAL_NORMAL = 0; % configure to the normal (autoset);
invoke(configuration, 'configureacquisition', NISCOPE_VAL_NORMAL);

%set up the vertical range;
VerticalRange = str2double(get(handles.verticalrange,'String'));
disp(['Vertical range is: ', num2str(VerticalRange)])
handles.VerticalRange=VerticalRange;
% vertical range is 8 volts
VerticalOffset = str2double(get(handles.edit48,'String'));
VerticalCoupling = 1;
VerticalAttenuation = 1;
ChannelEnable = 1;

groupObjv = get(niScopeObj, 'Configurationfunctionsvertical');
invoke(groupObjv, 'configurevertical','0',VerticalRange,VerticalOffset,...
    VerticalCoupling,VerticalAttenuation,ChannelEnable)% enabled channel 0


% Configure the trigger functions (analog for now)
groupObjt = get(niScopeObj, 'Configurationfunctionstrigger');
% for analog trigger
% % % NISCOPE_VAL_POSITIVE = 1;
% % % NISCOPE_VAL_NEGATIVE = -1;
% % %
% % % triggerSource = '1'; % channel 1 (analog signal). Specifies the trigger source. Refer to
NISCOPE_ATTR_TRIGGER_SOURCE for defined values.
% % % level = 1; % The voltage threshold for the trigger. Refer to
NISCOPE_ATTR_TRIGGER_LEVEL for more information.
% % % slope = NISCOPE_VAL_POSITIVE; %Specifies whether you want a rising edge or a
falling edge to trigger the digitizer. Refer to NISCOPE_ATTR_TRIGGER_SLOPE for more
information.
% % % triggerCoupling = 1; %Applies coupling and filtering options to the trigger signal. Refer to
NISCOPE_ATTR_TRIGGER_COUPLING for more information.
% % % holdoff  = 0;%The length of time the digitizer waits after detecting a trigger before
enabling NI-SCOPE to detect another trigger. Refer to NISCOPE_ATTR_TRIGGER_HOLDOFF
for more information.
% % % delay  = 0;%3e-5;
% % % invoke(groupObjt,
'configuretriggeredge',triggerSource,level,slope,triggerCoupling,holdoff,delay);
```

```
% for digital trigger
NISCOPE_VAL_POSITIVE = 1;
NISCOPE_VAL_NEGATIVE = 0;

triggerSource = 'PFI1'; % Specifies the trigger source. Refer to
NISCOPE_ATTR_TRIGGER_SOURCE for defined values.
slope = NISCOPE_VAL_POSITIVE; %Specifies whether you want a rising edge or a falling edge
to trigger the digitizer. Refer to NISCOPE_ATTR_TRIGGER_SLOPE for more information.
holdoff  = 0;%The length of time the digitizer waits after detecting a trigger before
%enabling NI-SCOPE to detect another trigger. Refer to NISCOPE_ATTR_TRIGGER_HOLDOFF
for more information.
delay  =0;%50e-6;

invoke(groupObjt, 'configuretriggerdigital',triggerSource,slope,...
    holdoff,delay);



handles.niScopeObj=niScopeObj;
%handles.samples=samples;
% handles.acquirerate=acquirerate;
% handles.acquiretime=acquiretime;
% acquiretime=str2double(get(handles.acquiretime,'String'));
% acquirerate=str2double(get(handles.acquirerate,'String'));

guidata(hObject, handles);


%% updat and organize handles.info structure
%--------------------------------------Updata handles.info structure------
handles.daq.ao=ao;
handles.daq.ai=ai;
handles.daq.dio=dio;
% handles.daq.niscopeobj=niscopeobj;
handles.info.daq.acquiretime=acquiretime;
handles.info.daq.acquirerate=acquirerate;

guidata(hObject, handles);

%% 2.3 initializing the connection of translation stage

s = serial('COM5');
set(s,'BaudRate',9600);
set(s,'InputBufferSize',50000);
fopen(s);
fprintf(s,'E,C,I1M-0,I1M200,IA1M-0,I2M-0,I2M200,IA2M-0,I3M-0,I3M200,IA3M-0,R');
%for PA and US imaging

% fclose(s);
% delete(s);
handles.s=s;
set(handles.systemstatus,'String','Completion of Initialization');
```

```
guidata(hObject,handles);

% --- Outputs from this function are returned to the command line.
function varargout = UST_all_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 on button press in stop.
function stop_Callback(hObject, eventdata, handles)
% hObject    handle to stop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
(cd('C:\Lab_folder\Jay\Using_wire'))
stop(handles.daq.ai);
stop(handles.daq.ao);
handles.jj=0;

s=handles.s;
% s = serial('COM4');
% set(s,'BaudRate',9600);
% set(s,'InputBufferSize',50000);
% fclose(s);
% fopen(s);
fprintf(s,'K,E,C,I3M-0,I3M200,IA3M-0,I1M-0,I1M200,IA1M-0,I2M-0,I2M200,IA2M-0,R');
% fclose(handles.s);
% for PA and US imaging
% fprintf(s,'K,Q,E,C,IA1M0,IA2M0,IA3M0,S1M2000,S2M2000,S3M2000,R,Q');

% fprintf(s,'K,E,C,Q');

% fclose(s);
% delete(s);
% Pause(0.01);
set(handles.systemstatus,'String','DAQ cards successfully stopped');

guidata(hObject,handles);


function edit21_Callback(hObject, eventdata, handles)
% hObject    handle to edit21 (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 edit21 as text
%        str2double(get(hObject,'String')) returns contents of edit21 as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit21_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit21 (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 edit22_Callback(hObject, eventdata, handles)
% hObject    handle to edit22 (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 edit22 as text
%        str2double(get(hObject,'String')) returns contents of edit22 as a double


% --- Executes during object creation, after setting all properties.
function edit22_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit22 (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 edit23_Callback(hObject, eventdata, handles)
% hObject    handle to edit23 (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 edit23 as text
%        str2double(get(hObject,'String')) returns contents of edit23 as a double


% --- Executes during object creation, after setting all properties.
function edit23_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit23 (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


% --- Executes on button press in start.
function start_Callback(hObject, eventdata, handles)
% hObject    handle to start (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% fopen(handles.s)
[hObject,eventdata,handles]=scancalculations(hObject,eventdata,handles);
% while strcmp(get(handles.daq.ai,'Running'),'On')
%     pause(0.01);
% end

filename=get(handles.filename,'string');
scans_dir=dir(handles.info.directories.scans);
pathname=handles.info.directories.scans;
% pathname(end+1)='/';
set(handles.savedirectory,'string',pathname);

if size(strfind(cat(2,scans_dir.name),filename),1)>0 % normal_scans_dir is a structure including
'name', 'data', 'bytes', and 'isidir' fields, which is generated by function of "aaa=dir(directory)"
%             %% CAT(2,A,B) is the same as [A,B]. CAT(1,A,B) is the same as
%             [A;B]. strfind('How are you','are'), ans=5; strfind('How are you','you'), ans=8;
strfind('How are you','ard'), ans=[];
    eval(sprintf('filename="%s%d";',filename,handles.info.incrementnumber+1)); %s: string to
string, %d number to string.
    handles.info.incrementnumber=handles.info.incrementnumber+1;
    guidata(hObject,handles);
else
    handles.info.incrementnumber=0;
    guidata(hObject,handles);
end


s=handles.s;
repeatime=str2double(get(handles.repeatime,'string'));

% when the scanning mode is point-point scanning;
%     set(handles.daq.ai,'SamplesPerTrigger',handles.info.daq.acquirerate);
xsteps=handles.info.daq.xstep;
ysteps=handles.info.daq.ystep;
zsteps=handles.info.daq.zstep;

xac=handles.info.daq.xac;
yac=handles.info.daq.yac;
zac=handles.info.daq.zac;

xspeed=handles.daq.xspeed;
```

```
yspeed=handles.daq.yspeed;
zspeed=handles.daq.zspeed;

if str2double(get(handles.zapos,'string'))>6000 || ((str2double(get(handles.zrange...
      ,'string'))/(2*.00025))+str2double(get(handles.zapos,'string')))>6100
   errordlg(['Z translation is large'])
   error('Z translation is large')
end

if str2double(get(handles.xapos,'string'))>12500 || ((str2double(get(handles.xrange...
      ,'string'))/(2*.00025))+str2double(get(handles.xapos,'string')))>12550
   errordlg(['X translation is large'])
   error('X translation is large')
end

pausetime=handles.info.daq.pausetime;

disp(['Acq: The x y z: ',num2str([handles.info.daq.xstep handles.info.daq.ystep...
   handles.info.daq.zstep]*0.00025),' inches']);
set(handles.xres,'String',num2str(handles.info.daq.xstep * 0.00025 * 25.4));
set(handles.yres,'String',num2str(handles.info.daq.ystep * 0.00025 * 25.4));
set(handles.zres,'String',num2str(handles.info.daq.zstep * 0.00025 * 25.4));

a=[zeros(50,1)' 3*ones(50,1)' zeros(10,1)']; % the low to high voltage for AO0.

niScopeObj=handles.niScopeObj;

acquiretime=str2double(get(handles.acquiretime,'String'));
acquirerate=str2double(get(handles.acquirerate,'String'));

samples=acquirerate*acquiretime;
handles.info.acquiretime=acquiretime;
handles.info.acquirerate=acquirerate;
handles.info.samples=samples;

handles.info.xapos=(str2double(get(handles.xapos,'string')));
handles.info.yapos=(str2double(get(handles.yapos,'string')));
handles.info.zapos=(str2double(get(handles.zapos,'string')));

set(handles.numsamples,'String',num2str(samples));
% disp(['No. of samples: ',num2str(samples)])

handles.samples=samples;

%%%%%%%%

%set up the vertical range;
VerticalRange = str2double(get(handles.verticalrange,'String'));
disp(['Vertical range is: ', num2str(VerticalRange)])
handles.VerticalRange=VerticalRange;% vertical range is 8 volts
VerticalOffset = str2double(get(handles.edit48,'String'));
VerticalCoupling = 1;
VerticalAttenuation = 1;
```

```
ChannelEnable = 1;

groupObjv = get(niScopeObj, 'Configurationfunctionsvertical');
invoke(groupObjv, 'configurevertical','1',VerticalRange,VerticalOffset,...
   VerticalCoupling,VerticalAttenuation,ChannelEnable)% enabled channel 0

%set up the horizontal range;
RefPosition = 0;
NumRecords = 1;
enforceRealtime = 0;
groupObjh = get(niScopeObj, 'Configurationfunctionshorizontal');
pause(5)
invoke(groupObjh, 'configurehorizontaltiming',acquirerate,samples,...
   RefPosition,NumRecords,enforceRealtime);

% Configure the trigger functions
groupObjt = get(niScopeObj, 'Configurationfunctionstrigger');
% for analog trigger
% % % NISCOPE_VAL_POSITIVE = 1;
% % % NISCOPE_VAL_NEGATIVE = -1;
% % %
% % % triggerSource = '1'; % channel 1 (analog signal). Specifies the trigger source. Refer to
NISCOPE_ATTR_TRIGGER_SOURCE for defined values.
% % % level = 1; % The voltage threshold for the trigger. Refer to
NISCOPE_ATTR_TRIGGER_LEVEL for more information.
% % % slope = NISCOPE_VAL_POSITIVE; %Specifies whether you want a rising edge or a
falling edge to trigger the digitizer. Refer to NISCOPE_ATTR_TRIGGER_SLOPE for more
information.
% % % triggerCoupling = 1; %Applies coupling and filtering options to the trigger signal. Refer to
NISCOPE_ATTR_TRIGGER_COUPLING for more information.
% % % holdoff  = 0;%The length of time the digitizer waits after detecting a trigger before
enabling NI-SCOPE to detect another trigger. Refer to NISCOPE_ATTR_TRIGGER_HOLDOFF
for more information.
% % % % delay  = 0%3e-5;
% % % delay = str2double(get(handles.Delay_Trigger,'String'));
% % % invoke(groupObjt,
'configuretriggeredge',triggerSource,level,slope,triggerCoupling,holdoff,delay);

% for digital trigger
NISCOPE_VAL_POSITIVE = 1;
NISCOPE_VAL_NEGATIVE = 0;

triggerSource = 'PFI1'; % Specifies the trigger source. Refer to
NISCOPE_ATTR_TRIGGER_SOURCE for defined values.
slope = NISCOPE_VAL_POSITIVE; %Specifies whether you want a rising edge or a falling edge
to trigger the digitizer. Refer to NISCOPE_ATTR_TRIGGER_SLOPE for more information.
holdoff  = 0;%The length of time the digitizer waits after detecting a trigger before
%enabling NI-SCOPE to detect another trigger. Refer to NISCOPE_ATTR_TRIGGER_HOLDOFF
for more information.
delay = str2double(get(handles.Delay_Trigger,'String'));

invoke(groupObjt, 'configuretriggerdigital',triggerSource,slope,...
   holdoff,delay);
```

```
%send out the command to control the stages for step movement;
fprintf(s,sprintf('''K,F,C,U4,IA2M%s,IA3M%s,IA1M%s,S1M%s,S2M%s,S3M%s,A1M%s,A2M%s,A
3M%s,R''',...
    num2str(handles.daq.yposition),num2str(handles.daq.zposition),...
    num2str(handles.daq.xposition),num2str(xspeed),num2str(yspeed),...
    num2str(zspeed),num2str(xac),num2str(yac),num2str(zac)));
%set up parameters, U4 means User output 1 'low';

pause(5);
% fprintf(s,sprintf('''LM0,U30,I1M%s,L%s,I2M%s,L-
%s,IA1M%s,IA2M%s,I3M%s,L%s,R''',num2str(xsteps),num2str(handles.info.daq.xpoint),num2str(
ysteps),num2str(handles.info.daq.ypoint),...
%
num2str(handles.daq.xposition),num2str(handles.daq.yposition),num2str(zsteps),num2str(handle
s.info.daq.zpoint)));
% set the 3 axis movement of the stages: U30 means wait for a low to high transition on user
input 1;

It=[];I=[];

WaveformArray = zeros(1, samples);
WfmInfo.absoluteInitialX =  0;
WfmInfo.relativeInitialX =  0;
WfmInfo.xIncrement =  1/handles.acquirerate; % Time delta
WfmInfo.actualSamples =  samples;
WfmInfo.gain =  1;
WfmInfo.reserved1 =  0;
WfmInfo.reserved2 =  0;
niScopeObj=handles.niScopeObj;
groupObja = get(niScopeObj, 'Acquisition');
channelList ='1';
time = (0:1:samples-1)*1/acquirerate;

tic
set(handles.systemstatus,'String',sprintf('Scan in Progress!'));
xpoint=handles.info.daq.xpoint;
% disp(['No of samples are: ',num2str(samples)])

disp(['Acquire rate: ', num2str(acquirerate),' Acquire time: ', num2str(acquiretime)])
disp(['Trigger delay: ', num2str(delay),' Voltage range: ', num2str(VerticalRange)])
handles.info.daq.VerticalRange=VerticalRange;
handles.info.daq.Delay=delay;

% clc
filename
pathname

fgpara=[{'FG_mV'} {'Cycles'} {'Interval'} {'Freq_MHz'}]
for i=1:4
```

```
ab(i)=input([char(fgpara(i)),': ']);
eval(sprintf('handles.info.daq.%s=%s;',char(fgpara(i)),num2str(ab(i))));
end

dataout.info=handles.info;

eval(sprintf('save %s%s.mat dataout',pathname,filename));

stime=clock;
avg=str2double(get(handles.loop_avg,'string'));

% % % % % % FOR all the experiment except PA and US imaging

% % disp(['The required samples are: ',num2str(handles.daq.ai.SamplesPerTrigger)])
% % tic
% % clc
% %  handles.daq.xposition
% %  handles.info.daq.xstep
% % if (handles.info.daq.xpoint>3 && (handles.info.daq.ypoint<3 || handles.info.daq.zpoint<3) )
% %     handles.daq.xposition
% %     fprintf(s,sprintf('"I1M%s,R"',num2str(handles.daq.xposition)));
% %     a=handles.daq.xposition;
% %     clc
% %     for i=1:handles.info.daq.xpoint
% %         [onlyx , WfmInfo] =  invoke(groupObja, 'read', channelList, 50,...
% %                 samples, WaveformArray, WfmInfo );
% %         eval(sprintf('save %s%s_x%d_p.mat onlyx time -v6',...
% %                 'C:\Lab_folder\Jay\Test\Test_data\',filename,i));
% %             handles.info.daq.xstep
% %             a=a+handles.info.daq.xstep;
% %             fprintf(s,sprintf('"I1M%s,R"',num2str(a)));
% %             putdata(handles.daq.ao,a);
% %     end
% % end
% % disp('Done')

%% Scan type

% handles.info.scantype=input('Enter the scan type (yx or zx):','s');
try
if handles.info.scantype=='zx'
%% for zx scan

for step1=1:handles.info.daq.ypoint%zpoint

   %% for zx scan
   fprintf(s,sprintf('"LM0,U30,I1M%s,L%s,I3M%s,L-%s,IA1M%s,IA2M%s,I3M%s,L%s,R"',...
      num2str(xsteps),num2str(handles.info.daq.xpoint),...
      num2str(zsteps),num2str(handles.info.daq.zpoint),...
      num2str(handles.daq.xposition),num2str(handles.daq.yposition),...
      num2str(zsteps),num2str(handles.info.daq.zpoint)));

   msgbox(['Z point is: ',num2str(step1)])
```

```
%     pause(5)
if step1>1
   errordlg(['2nd Y plane'])
   error('stopped after 1st plane image')
end
   for step2=1:handles.info.daq.zpoint%ypoint
%        a=input('Enter n to discontinue','s');
%        if a=='n'
%           errordlg(['program stopped'])
%           error('program stopped')
%        end

       for step3=1:handles.info.daq.xpoint

%           pause(str2double(get(handles.pausetime,'string')))
   % TO START DAQ ACQUISITION
         start(handles.daq.ai);% to start acquiring data

   % TO START DIGITIZER ACQUISITION
%           spue=zeros(1, samples);
%           size(spue)
%              for i=1:avg(1,1)
%           [Waveform_Ch0 , WfmInfo] =  invoke(groupObja, 'read', channelList, 50,...
%              samples, WaveformArray, WfmInfo );
%           I=Waveform_Ch0;
%               spue=I+spue;
%               pause(0.5)
%               i
%           end
%           avg(1,1)
%           I=spue/avg(1,1);
%           size(I)
   % ACQUIRING DATA USING DAQ CARD
         [data, daqtime] = getdata(handles.daq.ai,handles.daq.ai.SamplesPerTrigger);
         stop(handles.daq.ai);% to stop acquiring data
         assignin('base','data',data)
%% Changed on 02/23
%           if rem(step2,2)==1
%              I=fliplr(I);
%           end
%% for 7.5MHz H-108 transducer signal conditioning
%           Fs =length(I)/(time(length(time))-time(1));
%           [qc,wc]=cheby1(4,0.5,[((6*1e6)./(Fs/2)) ((9*1e6)./(Fs/2))]);
%           I=filtfilt(qc,wc,I);

         %It=[It {I'}]; % rearrange the data and get the line data
%           disp(step3)
%% For translation while saving in oscilloscope.
%           clc;disp('Save');
%           pause(2)
%           check=[5:5:40];
%           maxcheck= (step2==check);
%           ent='n';
```

```
%            while (max(maxcheck)==1 && ent=='n' && step3==1)
%               ent=input('Do u want to continue (y): ','s');
%            end
%            for i=1:5
%               pause(1)
%               clc
%              disp(['Time:',num2str(5-i)])
             disp('z  x')
%               if rem(step2,2)==1
                disp([num2str(step2),'  ',num2str(step3)]);
%               else
%                   disp([num2str(step2),'  ',num2str(handles.info.daq.xpoint-step3+1)]);
%               end
%            end

%% To trigger the motorized translation stage using DAQ card

          putdata(handles.daq.ao,a);
          start(handles.daq.ao); % the square wave is used to trigger the stage to move;

%% To save the acquired data
%            eval(sprintf('save %s%s_%d%d.mat It -v6',pathname, filename,step2,step1));
%            eval(sprintf('save %s%s_y%d_x%d_z%d_tcdigi.mat I time -v6',...
%               pathname,filename,step2,step1,step3));
          eval(sprintf('save %s%s_%d_%d_%d_tcdaq.mat data daqtime -v6',...
             pathname, filename,step2,step1,step3));
% pause(0.1)
       end
    end
    fprintf(s,sprintf('"F,C,IA1M%s,IA2M%s,I3M%s,R"',num2str(handles.daq.xposition),...
       num2str(handles.daq.yposition),num2str(zsteps)));
    % set the 3 axis movement of the stages: U30 means wait for a low to high transition on user
input 1;
%     stop(handles.daq.ao)
% pause(5)
end
elseif handles.info.scantype=='yx'
%% for yx scan

for step1=1:handles.info.daq.zpoint%ypoint

    %% for yx scan
     fprintf(s,sprintf('"LM0,U30,I1M%s,L%s,I2M%s,L-%s,IA2M%s,IA1M%s,I3M%s,L%s,R"',...
       num2str(xsteps),num2str(handles.info.daq.xpoint),...
       num2str(ysteps),num2str(handles.info.daq.ypoint),...
       num2str(handles.daq.yposition),num2str(handles.daq.xposition),...
       num2str(zsteps),num2str(handles.info.daq.zpoint)));
% for right angled 5MHz probing

%% for xy scan

%      fprintf(s,sprintf('"LM0,U30,I2M%s,L%s,I3M%s,L-%s,IA3M%s,IA2M%s,I1M%s,L%s,R"',...
%         num2str(ysteps),num2str(handles.info.daq.ypoint),...
```

```
%        num2str(zsteps),num2str(handles.info.daq.zpoint),...
%        num2str(handles.daq.zposition),num2str(handles.daq.yposition),...
%        num2str(xsteps),num2str(handles.info.daq.xpoint)));

%   fprintf(s,sprintf('''LM0,U30,I1M%s,L%s,I2M%s,L-%s,R''',num2str(xsteps),...
%   num2str(handles.info.daq.xpoint),num2str(ysteps),num2str(handles.info.daq.ypoint))); % set
the 3 axis movement of the stages: U30 means wait for a low to high transition on user input 1;
%     pause(5)
   msgbox(['Z point is: ',num2str(step1)])
%     pause(5)
%     if step1>1
if step1>1
   errordlg(['2nd Z plane'])
   error('stopped after 1st plane image')
end
% if step1>1
%     errordlg(['2nd Y plane'])
%     error('stopped after 1st plane image')
% end
   for step2=1:handles.info.daq.ypoint%zpoint
%        a=input('Enter n to discontinue','s');
%        if a=='n'
%            errordlg(['program stopped'])
%            error('program stopped')
%        end
      %%% to send the HIFU back to its starting position
%        if step2>1
% % % % %            fprintf(s,sprintf('''LM0,U30,I1M%s,L%s,I2M%s,L-
%s,IA1M%s,IA2M%s,I3M%s,L%s,R''',num2str(xsteps),num2str(handles.info.daq.xpoint),num2str(
ysteps),num2str(handles.info.daq.ypoint),...
% % % % %
num2str(handles.daq.xposition),num2str(handles.daq.yposition),num2str(zsteps),num2str(handle
s.info.daq.zpoint)));
% % % % %            break
%            error('stopped after first line acquisition')
%        end
      for step3=1:handles.info.daq.xpoint
         spue=zeros(1, samples);
         size(spue)
%            pause(str2double(get(handles.pausetime,'string')))
         start(handles.daq.ai);% to start acquiring data
%            for i=1:avg(1,1)
%            [Waveform_Ch0 , WfmInfo] =  invoke(groupObja, 'read', channelList, 50,...
%               samples, WaveformArray, WfmInfo );
%            I=Waveform_Ch0;
%                spue=I+spue;
% %                 pause(0.5)
%                i
%            end
%            avg(1,1)
%            I=spue/avg(1,1);
%            size(I)
%            disp('x  z')
```

```
%            disp([num2str(step2),'  ',num2str(step3)]);

        [data, daqtime] = getdata(handles.daq.ai,handles.daq.ai.SamplesPerTrigger);
        stop(handles.daq.ai);% to stop acquiring data
        assignin('base','data',data)
%% Changed on 02/23
%           if rem(step2,2)==1
%               I=fliplr(I);
%           end
%% for 7.5MHz H-108 transducer signal conditioning
%            Fs =length(I)/(time(length(time))-time(1));
%            [qc,wc]=cheby1(4,0.5,[((6*1e6)./(Fs/2)) ((9*1e6)./(Fs/2))]);
%            I=filtfilt(qc,wc,I);

        %It=[It {I'}]; % rearrange the data and get the line data
%            disp(step3)
%% For translation while saving in oscilloscope.
%            clc;disp('Save');
%            pause(2)
%            check=[5:5:40];
%            maxcheck= (step2==check);
%            ent='n';
%            while (max(maxcheck)==1 && ent=='n' && step3==1)
%               ent=input('Do u want to continue (y): ','s');
%            end
%            for i=1:5
%               pause(1)
%               clc
%              disp(['Time:',num2str(5-i)])
            disp('z  x')
%               if rem(step2,2)==1
               disp([num2str(step2),'  ',num2str(step3)]);
%               else
%                  disp([num2str(step2),'  ',num2str(handles.info.daq.xpoint-step3+1)]);
%               end
%            end

%% To trigger the motorized translation stage using DAQ card

        putdata(handles.daq.ao,a);
        start(handles.daq.ao); % the square wave is used to trigger the stage to move;
%            get(handles.daq.ao)
%            pause(3)
%            stop(handles.daq.ao);
% To save the acquired data
%            eval(sprintf('save %s%s_%d%d.mat It -v6',pathname, filename,step2,step1));
%            eval(sprintf('save %s%s_y%d_x%d_z%d_tcdigi.mat I time -v6',...
%               pathname,filename,step2,step1,step3));
        eval(sprintf('save %s%s_%d_%d_%d_tcdaq.mat data daqtime -v6',...
           pathname, filename,step2,step1,step3));
% pause(0.1)
      end
%        if rem(step2,2)==1
```

```
%            It=fliplr(It);
%        end
%        eval(sprintf('save %s%s_%d_%d.mat It -v6',pathname, filename,step2,step1));
%        It=[];
   end
%    else
%        putdata(handles.daq.ao,a);
%            start(handles.daq.ao);
%    end
   %clock
   %disp(['Scan Time mins: ', num2str(ans(5)),'sec:', num2str(ans(6)) ]);
         %handles.daq.yposition;
    %zsteps;
   fprintf(s,sprintf('"F,C,IA1M%s,IA2M%s,I3M%s,R"',num2str(handles.daq.xposition),...
      num2str(handles.daq.yposition),num2str(zsteps)));
   % set the 3 axis movement of the stages: U30 means wait for a low to high transition on user
input 1;
%    stop(handles.daq.ao)
% pause(5)
end

end
catch ME
   send_text_message('801-746-9712','AT&T','Simulation stopped',ME.message)
end
handles.info.scantime=toc;
% FOR the experiment PA and US imaging
% % % % % % % % % % % % for step1=1:handles.info.daq.zpoint
% % % % % % % %
fprintf(s,sprintf('"LM0,U30,I2M%s,L%s,R"',num2str(ysteps),num2str(handles.info.daq.ypoint)));
% % % % % % % %    % set the 3 axis movement of the stages: U30 means wait for a low to
high transition on user input 1;
% % % % % % % %    handles.info.daq.ypoint
% % % % % % % %    disp('yo')
% % % % % % % % % % % % %    pause(2)
% % % % % % % % %    msgbox(['Z point is: ',num2str(step1)])
% % % % % % % % %    pause(5)
% % % % % % % % %    if step1>1
% % % % % % % %    for step2=1:handles.info.daq.ypoint
% % % % % % % % % % % % %        for step3=1:handles.info.daq.xpoint
% % % % % % % %
% % % % % % % % % % % % %            if rem(step2,2)==1
% % % % % % % % % % % % %            set(handles.systemstatus,'String',sprintf(['Scan in
Progress: ',...
% % % % % % % % % % % % %                num2str([step3 step2 step1]),' done']));
% % % % % % % % % % % % %            else
% % % % % % % % % % % % %                set(handles.systemstatus,'String',sprintf(['Scan in
Progress: ',...
% % % % % % % % % % % % %                num2str([(xpoint-step3+1) step2 step1]),' done']));
% % % % % % % % % % % % %            end
% % % % % % % % %            disp(['samples are: ',num2str(samples)]);
% % % % % % % % beep
% % % % % % % %            spue=zeros(1, samples);
```

```
% % % % % % % %           pause(str2double(get(handles.pausetime,'string')))
% % % % % % % %           for i=1:avg(1,1)
% % % % % % % %           [Waveform_Ch0 , WfmInfo] =  invoke(groupObja, 'read', channelList,
10,...
% % % % % % % %              samples, WaveformArray, WfmInfo );
% % % % % % % %           I=Waveform_Ch0;
% % % % % % % %               spue=I+spue;
% % % % % % % %           end
% % % % % % % %           I=spue/avg(1,1);
% % % % % % % %           It=[It {I'}]; % rearrange the data and get the line data
% % % % % % % % %            disp(step3)
% % % % % % % %           putdata(handles.daq.ao,a);
% % % % % % % %           start(handles.daq.ao); % the square wave is used to trigger the
stage to move;
% % % % % % % % %            stop(handles.daq.ao);
% % % % % % % % %            eval(sprintf('save %s%s_%d%d.mat It -v6',pathname,
filename,step2,step1));
% % % % % % % % % % % %       end
% % % % % % % %         if rem(step2,2)==1
% % % % % % % %            It=fliplr(It);
% % % % % % % %         end
% % % % % % % %         set(handles.systemstatus,'String',sprintf(['Scan in Progress:
',num2str([step2]),' done']));
% % % % % % % %         eval(sprintf('save %s%s_%d.mat It -v6',pathname, filename,step2));
% % % % % % % %         It=[];
% % % % % % % % % % % % %     end
% % % % % % % % %     else
% % % % % % % % %         putdata(handles.daq.ao,a);
% % % % % % % % %            start(handles.daq.ao);
% % % % % % % % %     end
% % % % % % % %     %clock
% % % % % % % %     %disp(['Scan Time mins: ', num2str(ans(5)),'sec:', num2str(ans(6)) ]);
% % % % % % % %           %handles.daq.yposition;
% % % % % % % %      %zsteps;
% % % % % % % % % % % % %
fprintf(s,sprintf('"F,C,IA1M%s,IA2M%s,I3M%s,R"',num2str(handles.daq.xposition),...
% % % % % % % % % % % %         num2str(handles.daq.yposition),num2str(zsteps))); % set
the 3 axis movement of the stages: U30 means wait for a low to high transition on user input 1;
% % % % % % % %
% % % % % % % % %      fprintf(s,sprintf('"F,C,IA2M%s,R"',num2str(handles.daq.yposition))); %
set the 3 axis movement of the stages: U30 means wait for a low to high transition on user input
1;
% % % % % % % %
% % % % % % % %
% % % % % % % % %     stop(handles.daq.ao)
% % % % % % % % % pause(5)
% % % % % % % %     end
% % % % % % % %     pause(2)
% % % % % % % %
fprintf(s,sprintf('"F,C,IA1M%s,IA2M%s,R"',num2str(handles.daq.xposition),...
% % % % % % % %         num2str(handles.daq.yposition)));

etime(clock,stime)
```

```
disp(['Elapsed time is: ',num2str(etime(clock,stime)/60),' mins'])

% dataout.info=handles.info;
% %dataout.info.extra=handles;
% eval(sprintf('save %s%s.mat dataout',pathname,filename));

% fclose(s);
% delete(s);
% disconnect(niScopeObj);
% delete (niScopeObj);
set(handles.systemstatus,'String',sprintf('Scan finished successfully!'));
toc
% plot(I);
handles.info.pathname=pathname;
handles.info.filename=filename;
handles.I=I;
clear It;
clear dataout;
clear I;
guidata(hObject,handles);

%% sub_function of the scancalculation:

function [hObject eventdata handles a b c d]=scancalculations(hObject,eventdata,handles)

clc

if strcmp(handles.info.scantype,'zx')==1
   xsp= (str2double(get(handles.xapos,'string'))*0.00025)-
(str2double(get(handles.xrange,'string'))/2)
   set(handles.xstartposition,'String',num2str(xsp));
   ysp= (str2double(get(handles.yapos,'string'))*0.00025)%-
(str2double(get(handles.yrange,'string'))/2)
   set(handles.ystartposition,'String',num2str(ysp));
   zsp= (str2double(get(handles.zapos,'string'))*0.00025)-
(str2double(get(handles.zrange,'string'))/2)
   set(handles.zstartposition,'String',num2str(zsp));

elseif strcmp(handles.info.scantype,'yx')==1
   xsp= (str2double(get(handles.xapos,'string'))*0.00025)-
(str2double(get(handles.xrange,'string'))/2)
   set(handles.xstartposition,'String',num2str(xsp));
   ysp= (str2double(get(handles.yapos,'string'))*0.00025)-
(str2double(get(handles.yrange,'string'))/2)
   set(handles.ystartposition,'String',num2str(ysp));
   zsp= (str2double(get(handles.zapos,'string'))*0.00025)%-
(str2double(get(handles.zrange,'string'))/2)
   set(handles.zstartposition,'String',num2str(zsp));
end

xposition=str2double(get(handles.xstartposition,'string'))/0.00025;
yposition=str2double(get(handles.ystartposition,'string'))/0.00025;
zposition=str2double(get(handles.zstartposition,'string'))/0.00025;
```

```
xrange_mod=str2double(get(handles.xrange,'string'))/0.00025;
yrange_mod=str2double(get(handles.yrange,'string'))/0.00025;
zrange_mod=str2double(get(handles.zrange,'string'))/0.00025;

xspeed=str2double(get(handles.xscanrate,'String'))/0.00025;
yspeed=str2double(get(handles.yscanrate,'String'))/0.00025;
zspeed=str2double(get(handles.zscanrate,'String'))/0.00025;

if xrange_mod==0
   xpoint=1
   xrange=0
else
   xpoint=str2double(get(handles.xpoint,'string'))
   xrange=(xpoint/(xpoint-1))*xrange_mod
end
if yrange_mod==0
   ypoint=0
   yrange=0
else
   ypoint=str2double(get(handles.ypoint,'string'))
   yrange=(ypoint/(ypoint-1))*yrange_mod
end
if zrange_mod==0
   zpoint=0
   zrange=0
else
   zpoint=str2double(get(handles.zpoint,'string'))
   zrange=(zpoint/(zpoint-1))*zrange_mod
end

xac=str2double(get(handles.xac,'String'));
yac=str2double(get(handles.yac,'String'));
zac=str2double(get(handles.zac,'String'));

xsteps=xrange/xpoint
ysteps=yrange/ypoint
zsteps=zrange/zpoint

% repeatime=str2double(get(handles.repeatime,'String'));
pausetime=str2double(get(handles.pausetime,'String'));
acquiretime=str2double(get(handles.acquiretime,'String'));
acquirerate=str2double(get(handles.acquirerate,'String'));

samples=acquirerate*acquiretime;

if (xspeed>5500) || (yspeed>5500)
   warndlg('"The speed is to much for the system, please check again"','System Nofication');
   return
end
if (xrange>16000) || (yrange>16000)
   warndlg('"The range is over system limit!"','System Nofication');
   return
```

```
end
% acquiretime=pixelx*pixely/xscanrate;
% xx=xrange*2*repmat([(1:pixelx) fliplr((1:pixelx))],[1,pixely/2])/pixelx-xrange;
% a = (-(pixely-1):2:(pixely-1))*yrange/(pixely-1);
% b = repmat(a,pixelx,1);
% yy = load(b,[1,(pixely)*(pixelx)]);

% set(handles.daq.ai,'SampleRate',acquirerate); need to set the samplerate
% to NI-5133 digitizer;
%
% if get(handles.mode,'value')== 1
%     set(handles.acquiretime,'String',repeatime*xrange/xspeed*ypoint);
%     set(handles.daq.ai,'SamplesPerTrigger',acquirerate*xrange/xspeed*ypoint);
%     handles.daq.samplepertrigger=acquirerate*xrange/xspeed*ypoint;
%     handles.info.daq.acquiretime=repeatime*xrange/xspeed*ypoint;
%
% elseif get(handles.mode,'value')== 2
%     set(handles.acquiretime,'String',repeatime*xrange/xspeed*ypoint);
%     set(handles.daq.ai,'SamplesPerTrigger',acquirerate*xrange/xspeed);
%     handles.daq.samplepertrigger=acquirerate*xrange/xspeed;
%     handles.info.daq.acquiretime=repeatime*xrange/xspeed*ypoint;
%
%     get(handles.mode,'value')==3
%
set(handles.acquiretime,'String',repeatime*xrange/xspeed*ypoint*zpoint+pausetime*xpoint*ypoint
*zpoint);
%     set(handles.daq.ai,'SamplesPerTrigger',samples); need to set up the NI-5133 digitizer for
the samples/trigger.
set(handles.shape,'string',sprintf('%d*%d*%d',xpoint,ypoint,zpoint));
handles.daq.samplepertrigger=samples;
handles.info.daq.pausetime=pausetime;
%     handles.info.daq.acquiretime=repeatime*xrange/xspeed*ypoint+pausetime*xpoint*ypoint;
%     handles.info.daq.samples=samples;
% end

% set(handles.daq.ai,'LoggingMode','Memory');

% update info;
handles.info.daq.acquirerate=acquirerate;
handles.info.daq.acquiretime=acquiretime;

handles.info.daq.xposition=xposition*0.00025; handles.daq.xposition=xposition;
handles.info.daq.yposition=yposition*0.00025; handles.daq.yposition=yposition;
handles.info.daq.zposition=zposition*0.00025; handles.daq.zposition=zposition;

handles.info.daq.xrange=xrange*0.00025; handles.daq.xrange=xrange;
handles.info.daq.yrange=yrange*0.00025; handles.daq.yrange=yrange;
handles.info.daq.zrange=zrange*0.00025; handles.daq.zrange=zrange;

handles.info.daq.xrange_mod=xrange_mod*0.00025;
handles.info.daq.yrange_mod=yrange_mod*0.00025;
handles.info.daq.zrange_mod=zrange_mod*0.00025;
```

```
handles.info.daq.xspeed=xspeed*0.00025; handles.daq.xspeed=xspeed;
handles.info.daq.yspeed=yspeed*0.00025; handles.daq.yspeed=yspeed;
handles.info.daq.zspeed=zspeed*0.00025; handles.daq.zspeed=zspeed;

handles.info.daq.xpoint=xpoint;
handles.info.daq.ypoint=ypoint;
handles.info.daq.zpoint=zpoint;

handles.info.daq.xac=xac;
handles.info.daq.yac=yac;
handles.info.daq.zac=zac;

% handles.info.daq.repeatime=repeatime;

handles.info.daq.xstep=xsteps;
handles.info.daq.ystep=ysteps;
handles.info.daq.zstep=zsteps;

guidata(hObject,handles);

function acquirerate_Callback(hObject, eventdata, handles)
% hObject    handle to acquirerate (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 acquirerate as text
%        str2double(get(hObject,'String')) returns contents of acquirerate as a double


% --- Executes during object creation, after setting all properties.
function acquirerate_CreateFcn(hObject, eventdata, handles)
% hObject    handle to acquirerate (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 acquiretime_Callback(hObject, eventdata, handles)
% hObject    handle to acquiretime (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 acquiretime as text
%        str2double(get(hObject,'String')) returns contents of acquiretime as a double


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

```
function acquiretime_CreateFcn(hObject, eventdata, handles)
% hObject    handle to acquiretime (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 repeatime_Callback(hObject, eventdata, handles)
% hObject    handle to repeatime (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 repeatime as text
%        str2double(get(hObject,'String')) returns contents of repeatime as a double


% --- Executes during object creation, after setting all properties.
function repeatime_CreateFcn(hObject, eventdata, handles)
% hObject    handle to repeatime (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 xscanrate_Callback(hObject, eventdata, handles)
% hObject    handle to xscanrate (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 xscanrate as text
%        str2double(get(hObject,'String')) returns contents of xscanrate as a double


% --- Executes during object creation, after setting all properties.
function xscanrate_CreateFcn(hObject, eventdata, handles)
% hObject    handle to xscanrate (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 xpoint_Callback(hObject, eventdata, handles)
% hObject    handle to xpoint (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 xpoint as text
%        str2double(get(hObject,'String')) returns contents of xpoint as a double


% --- Executes during object creation, after setting all properties.
function xpoint_CreateFcn(hObject, eventdata, handles)
% hObject    handle to xpoint (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 yscanrate_Callback(hObject, eventdata, handles)
% hObject    handle to yscanrate (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 yscanrate as text
%        str2double(get(hObject,'String')) returns contents of yscanrate as a double


% --- Executes during object creation, after setting all properties.
function yscanrate_CreateFcn(hObject, eventdata, handles)
% hObject    handle to yscanrate (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 zscanrate_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to zscanrate (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 zscanrate as text
%        str2double(get(hObject,'String')) returns contents of zscanrate as a double


% --- Executes during object creation, after setting all properties.
function zscanrate_CreateFcn(hObject, eventdata, handles)
% hObject    handle to zscanrate (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 xrange_Callback(hObject, eventdata, handles)
% hObject    handle to xrange (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 xrange as text
%        str2double(get(hObject,'String')) returns contents of xrange as a double


% --- Executes during object creation, after setting all properties.
function xrange_CreateFcn(hObject, eventdata, handles)
% hObject    handle to xrange (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 yrange_Callback(hObject, eventdata, handles)
% hObject    handle to yrange (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 yrange as text
%        str2double(get(hObject,'String')) returns contents of yrange as a double
```

```
% --- Executes during object creation, after setting all properties.
function yrange_CreateFcn(hObject, eventdata, handles)
% hObject    handle to yrange (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 zrange_Callback(hObject, eventdata, handles)
% hObject    handle to zrange (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 zrange as text
%        str2double(get(hObject,'String')) returns contents of zrange as a double


% --- Executes during object creation, after setting all properties.
function zrange_CreateFcn(hObject, eventdata, handles)
% hObject    handle to zrange (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 xac_Callback(hObject, eventdata, handles)
% hObject    handle to xac (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 xac as text
%        str2double(get(hObject,'String')) returns contents of xac as a double


% --- Executes during object creation, after setting all properties.
function xac_CreateFcn(hObject, eventdata, handles)
% hObject    handle to xac (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 yac_Callback(hObject, eventdata, handles)
% hObject    handle to yac (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 yac as text
%        str2double(get(hObject,'String')) returns contents of yac as a double


% --- Executes during object creation, after setting all properties.
function yac_CreateFcn(hObject, eventdata, handles)
% hObject    handle to yac (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 zac_Callback(hObject, eventdata, handles)
% hObject    handle to zac (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 zac as text
%        str2double(get(hObject,'String')) returns contents of zac as a double


% --- Executes during object creation, after setting all properties.
function zac_CreateFcn(hObject, eventdata, handles)
% hObject    handle to zac (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 ypoint_Callback(hObject, eventdata, handles)
% hObject    handle to ypoint (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 ypoint as text
%        str2double(get(hObject,'String')) returns contents of ypoint as a double


% --- Executes during object creation, after setting all properties.
function ypoint_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ypoint (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 zpoint_Callback(hObject, eventdata, handles)
% hObject    handle to zpoint (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 zpoint as text
%        str2double(get(hObject,'String')) returns contents of zpoint as a double


% --- Executes during object creation, after setting all properties.
function zpoint_CreateFcn(hObject, eventdata, handles)
% hObject    handle to zpoint (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 xstartposition_Callback(hObject, eventdata, handles)
% hObject    handle to xstartposition (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 xstartposition as text
```

```
%        str2double(get(hObject,'String')) returns contents of xstartposition as a double


% --- Executes during object creation, after setting all properties.
function xstartposition_CreateFcn(hObject, eventdata, handles)
% hObject    handle to xstartposition (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 ystartposition_Callback(hObject, eventdata, handles)
% hObject    handle to ystartposition (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 ystartposition as text
%        str2double(get(hObject,'String')) returns contents of ystartposition as a double


% --- Executes during object creation, after setting all properties.
function ystartposition_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ystartposition (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 zstartposition_Callback(hObject, eventdata, handles)
% hObject    handle to zstartposition (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 zstartposition as text
%        str2double(get(hObject,'String')) returns contents of zstartposition as a double


% --- Executes during object creation, after setting all properties.
function zstartposition_CreateFcn(hObject, eventdata, handles)
% hObject    handle to zstartposition (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 pausetime_Callback(hObject, eventdata, handles)
% hObject    handle to pausetime (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 pausetime as text
%        str2double(get(hObject,'String')) returns contents of pausetime as a double


% --- Executes during object creation, after setting all properties.
function pausetime_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pausetime (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 Delay_Trigger_Callback(hObject, eventdata, handles)
% hObject    handle to Delay_Trigger (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 Delay_Trigger as text
%        str2double(get(hObject,'String')) returns contents of Delay_Trigger as a double


% --- Executes during object creation, after setting all properties.
function Delay_Trigger_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Delay_Trigger (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 systemstatus_Callback(hObject, eventdata, handles)
% hObject    handle to systemstatus (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 systemstatus as text
%        str2double(get(hObject,'String')) returns contents of systemstatus as a double


% --- Executes during object creation, after setting all properties.
function systemstatus_CreateFcn(hObject, eventdata, handles)
% hObject    handle to systemstatus (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 shape_Callback(hObject, eventdata, handles)
% hObject    handle to shape (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 shape as text
%        str2double(get(hObject,'String')) returns contents of shape as a double


% --- Executes during object creation, after setting all properties.
function shape_CreateFcn(hObject, eventdata, handles)
% hObject    handle to shape (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


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

```
s=handles.s;
handles.info.scantype=input('Enter scantype (yx or zx): ','s');
set(handles.scantype,'String',handles.info.scantype);
% s = serial('COM4');
% set(s,'BaudRate',9600);
% set(s,'InputBufferSize',50000);
% fopen(s);
% fprintf(s,'E,C,I1M-0,I1M200,IA1M-0,I2M-0,I2M200,IA2M-0,I3M-0,I3M200,IA3M-0,R');
if strcmp(handles.info.scantype,'zx')==1
   xsp= (str2double(get(handles.xapos,'string'))*0.00025)-
(str2double(get(handles.xrange,'string'))/2)
   set(handles.xstartposition,'String',num2str(xsp));
   ysp= (str2double(get(handles.yapos,'string'))*0.00025)%-
(str2double(get(handles.yrange,'string'))/2)
   set(handles.ystartposition,'String',num2str(ysp));
   zsp= (str2double(get(handles.zapos,'string'))*0.00025)-
(str2double(get(handles.zrange,'string'))/2)
   set(handles.zstartposition,'String',num2str(zsp));

elseif strcmp(handles.info.scantype,'yx')==1
   xsp= (str2double(get(handles.xapos,'string'))*0.00025)-
(str2double(get(handles.xrange,'string'))/2)
   set(handles.xstartposition,'String',num2str(xsp));
   ysp= (str2double(get(handles.yapos,'string'))*0.00025)-
(str2double(get(handles.yrange,'string'))/2)
   set(handles.ystartposition,'String',num2str(ysp));
   zsp= (str2double(get(handles.zapos,'string'))*0.00025)%-
(str2double(get(handles.zrange,'string'))/2)
   set(handles.zstartposition,'String',num2str(zsp));
end

if str2double(get(handles.zapos,'string'))>6000
   errordlg(['Z translation is large'])
   error('Z translation is large')
end

xposition=str2double(get(handles.xstartposition,'string'))/0.00025;
yposition=str2double(get(handles.ystartposition,'string'))/0.00025;
zposition=str2double(get(handles.zstartposition,'string'))/0.00025;

fprintf(s,sprintf('"F,E,C,IA2M%s,IA3M%s,IA1M%s,R"',num2str(yposition),num2str(zposition),...
   num2str(xposition))); %set up parameters, U4 means User output 1 'low';
% fclose(s);
% delete(s);
guidata(hObject,handles);


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

```
% s = serial('COM4');
% set(s,'BaudRate',9600);
% set(s,'InputBufferSize',50000);
% fopen(s);
% fprintf(s,'E,C,I1M-0,I1M200,IA1M-0,I2M-0,I2M200,IA2M-0,I3M-0,I3M200,IA3M-0,R');
xrange=str2double(get(handles.xrange,'string'))/0.00025;
yrange=str2double(get(handles.yrange,'string'))/0.00025;
zrange=str2double(get(handles.zrange,'string'))/0.00025;
% scan the stage roughly and check the range of it.
fprintf(s,sprintf('"F,C,I1M%s,I2M%s,I3M%s,R"',num2str(xrange),num2str(yrange),num2str(zrange
))); %
% fclose(s);
% delete(s);
guidata(hObject,handles);



function verticalrange_Callback(hObject, eventdata, handles)
% hObject    handle to verticalrange (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 verticalrange as text
%        str2double(get(hObject,'String')) returns contents of verticalrange as a double


% --- Executes during object creation, after setting all properties.
function verticalrange_CreateFcn(hObject, eventdata, handles)
% hObject    handle to verticalrange (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 filename_Callback(hObject, eventdata, handles)
% hObject    handle to filename (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 filename as text
%        str2double(get(hObject,'String')) returns contents of filename as a double


% --- Executes during object creation, after setting all properties.
function filename_CreateFcn(hObject, eventdata, handles)
% hObject    handle to filename (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 savedirectory_Callback(hObject, eventdata, handles)
% hObject    handle to savedirectory (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 savedirectory as text
%        str2double(get(hObject,'String')) returns contents of savedirectory as a double


% --- Executes during object creation, after setting all properties.
function savedirectory_CreateFcn(hObject, eventdata, handles)
% hObject    handle to savedirectory (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 edit31_Callback(hObject, eventdata, handles)
% hObject    handle to filename (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 filename as text
%        str2double(get(hObject,'String')) returns contents of filename as a double


% --- Executes during object creation, after setting all properties.
function edit31_CreateFcn(hObject, eventdata, handles)
% hObject    handle to filename (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


% --- Executes when figure1 is resized.
function figure1_ResizeFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in quitbutton.
function quitbutton_Callback(hObject, eventdata, handles)
% hObject    handle to quitbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
fig = handles.figure1;
fprintf(handles.s,sprintf("'F,C,Q"'))
fclose(handles.s);
delete(handles.s);
clear handles.s;
disconnect(handles.niScopeObj);
delete(handles.niScopeObj);
close(fig);

function scannumber_Callback(hObject, eventdata, handles)
% hObject    handle to scannumber (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% handles.scannumber=
% Hints: get(hObject,'String') returns contents of scannumber as text
%        str2double(get(hObject,'String')) returns contents of scannumber as a double


% --- Executes during object creation, after setting all properties.
function scannumber_CreateFcn(hObject, eventdata, handles)
% hObject    handle to scannumber (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




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

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


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


% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes1


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

function axes2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes2



function sampledisp_Callback(hObject, eventdata, handles)
% hObject    handle to sampledisp (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 sampledisp as text
%        str2double(get(hObject,'String')) returns contents of sampledisp as a double


% --- Executes during object creation, after setting all properties.
function sampledisp_CreateFcn(hObject, eventdata, handles)
% hObject    handle to sampledisp (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 numsamples_Callback(hObject, eventdata, handles)
% hObject    handle to numsamples (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 numsamples as text
%        str2double(get(hObject,'String')) returns contents of numsamples as a double


% --- Executes during object creation, after setting all properties.
function numsamples_CreateFcn(hObject, eventdata, handles)
% hObject    handle to numsamples (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 timestamp_Callback(hObject, eventdata, handles)
% hObject    handle to timestamp (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 timestamp as text
%        str2double(get(hObject,'String')) returns contents of timestamp as a double


% --- Executes during object creation, after setting all properties.
function timestamp_CreateFcn(hObject, eventdata, handles)
% hObject    handle to timestamp (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 xres_Callback(hObject, eventdata, handles)
% hObject    handle to xres (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 xres as text
%        str2double(get(hObject,'String')) returns contents of xres as a double


% --- Executes during object creation, after setting all properties.
function xres_CreateFcn(hObject, eventdata, handles)
% hObject    handle to xres (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 yres_Callback(hObject, eventdata, handles)
% hObject    handle to yres (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 yres as text
%        str2double(get(hObject,'String')) returns contents of yres as a double


% --- Executes during object creation, after setting all properties.
function yres_CreateFcn(hObject, eventdata, handles)
% hObject    handle to yres (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 zres_Callback(hObject, eventdata, handles)
% hObject    handle to zres (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 zres as text
```

```
%        str2double(get(hObject,'String')) returns contents of zres as a double


% --- Executes during object creation, after setting all properties.
function zres_CreateFcn(hObject, eventdata, handles)
% hObject    handle to zres (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 loadxres_Callback(hObject, eventdata, handles)
% hObject    handle to loadxres (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 loadxres as text
%        str2double(get(hObject,'String')) returns contents of loadxres as a double


% --- Executes during object creation, after setting all properties.
function loadxres_CreateFcn(hObject, eventdata, handles)
% hObject    handle to loadxres (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 loadyres_Callback(hObject, eventdata, handles)
% hObject    handle to loadyres (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 loadyres as text
%        str2double(get(hObject,'String')) returns contents of loadyres as a double


% --- Executes during object creation, after setting all properties.
function loadyres_CreateFcn(hObject, eventdata, handles)
% hObject    handle to loadyres (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 loadzres_Callback(hObject, eventdata, handles)
% hObject    handle to loadzres (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 loadzres as text
%        str2double(get(hObject,'String')) returns contents of loadzres as a double


% --- Executes during object creation, after setting all properties.
function loadzres_CreateFcn(hObject, eventdata, handles)
% hObject    handle to loadzres (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 xapos_Callback(hObject, eventdata, handles)
% hObject    handle to xapos (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 xapos as text
%        str2double(get(hObject,'String')) returns contents of xapos as a double


% --- Executes during object creation, after setting all properties.
function xapos_CreateFcn(hObject, eventdata, handles)
% hObject    handle to xapos (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 yapos_Callback(hObject, eventdata, handles)
% hObject    handle to yapos (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 yapos as text
%        str2double(get(hObject,'String')) returns contents of yapos as a double


% --- Executes during object creation, after setting all properties.
function yapos_CreateFcn(hObject, eventdata, handles)
% hObject    handle to yapos (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 zapos_Callback(hObject, eventdata, handles)
% hObject    handle to zapos (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 zapos as text
%        str2double(get(hObject,'String')) returns contents of zapos as a double


% --- Executes during object creation, after setting all properties.
function zapos_CreateFcn(hObject, eventdata, handles)
% hObject    handle to zapos (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


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

```
s=handles.s;
% s = serial('COM4');
% set(s,'BaudRate',9600);
% set(s,'InputBufferSize',50000);
% fopen(s);
% fprintf(s,'E,C,I1M-0,I1M200,IA1M-0,I2M-0,I2M200,IA2M-0,I3M-0,I3M200,IA3M-0,R');

xcenter=str2double(get(handles.xapos,'string'));
ycenter=str2double(get(handles.yapos,'string'));
zcenter=str2double(get(handles.zapos,'string'));

if str2double(get(handles.zapos,'string'))>6000
   errordlg(['Z translation is large'])
   error('Z translation is large')
end

set(handles.xcen,'String',num2str(xcenter*0.00025));
set(handles.ycen,'String',num2str(ycenter*0.00025));
set(handles.zcen,'String',num2str(zcenter*0.00025));

fprintf(s,sprintf('"K,F,C,IA2M%s,IA3M%s,IA1M%s,R"',num2str(ycenter),num2str(zcenter),...
   num2str(xcenter))); %set up parameters, U4 means User output 1 'low';
%for PA and US imaging

% fprintf(s,sprintf('"F,C,IA3M%s,R"',num2str(zcenter)));
%
% fprintf(s,sprintf('"F,C,IA2M%s,IA1M%s,R"',num2str(ycenter),num2str(xcenter))); %set up
parameters, U4 means User output 1 'low';

% fclose(s);
% delete(s);
guidata(hObject,handles);



function xcen_Callback(hObject, eventdata, handles)
% hObject    handle to xcen (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 xcen as text
%        str2double(get(hObject,'String')) returns contents of xcen as a double


% --- Executes during object creation, after setting all properties.
function xcen_CreateFcn(hObject, eventdata, handles)
% hObject    handle to xcen (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 ycen_Callback(hObject, eventdata, handles)
% hObject    handle to ycen (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 ycen as text
%        str2double(get(hObject,'String')) returns contents of ycen as a double


% --- Executes during object creation, after setting all properties.
function ycen_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ycen (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 zcen_Callback(hObject, eventdata, handles)
% hObject    handle to zcen (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 zcen as text
%        str2double(get(hObject,'String')) returns contents of zcen as a double


% --- Executes during object creation, after setting all properties.
function zcen_CreateFcn(hObject, eventdata, handles)
% hObject    handle to zcen (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


% --- Executes on button press in Offline.
function Offline_Callback(hObject, eventdata, handles)
% hObject    handle to Offline (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
s=handles.s;
fprintf(s,sprintf('''F,Q'''));
guidata(hObject,handles);


% --- Executes on button press in pushbutton19.
function pushbutton19_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
s=handles.s;
% s = serial('COM4');
% set(s,'BaudRate',9600);
% set(s,'InputBufferSize',50000);
% fopen(s);
% fprintf(s,'E,C,I1M-0,I1M200,IA1M-0,I2M-0,I2M200,IA2M-0,I3M-0,I3M200,IA3M-0,R');
xposition=str2double(get(handles.xstartposition,'string'))/0.00025;
yposition=str2double(get(handles.ystartposition,'string'))/0.00025;
zposition=str2double(get(handles.zstartposition,'string'))/0.00025;

fprintf(s,sprintf('''F,C,IA2M%s,IA1M%s,IA3M%s,R''',num2str(yposition),num2str(xposition),...
   num2str(zposition))); %set up parameters, U4 means User output 1 'low';
% fclose(s);
% delete(s);
guidata(hObject,handles);


function loop_avg_Callback(hObject, eventdata, handles)
% hObject    handle to loop_avg (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 loop_avg as text
%        str2double(get(hObject,'String')) returns contents of loop_avg as a double


% --- Executes during object creation, after setting all properties.
function loop_avg_CreateFcn(hObject, eventdata, handles)
% hObject    handle to loop_avg (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 entrot_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to entrot (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 entrot as text
%        str2double(get(hObject,'String')) returns contents of entrot as a double


% --- Executes during object creation, after setting all properties.
function entrot_CreateFcn(hObject, eventdata, handles)
% hObject    handle to entrot (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 entinch_Callback(hObject, eventdata, handles)
% hObject    handle to entinch (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 entinch as text
%        str2double(get(hObject,'String')) returns contents of entinch as a double


% --- Executes during object creation, after setting all properties.
function entinch_CreateFcn(hObject, eventdata, handles)
% hObject    handle to entinch (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 edit55_Callback(hObject, eventdata, handles)
% hObject    handle to edit55 (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 edit55 as text
%        str2double(get(hObject,'String')) returns contents of edit55 as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit55_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit55 (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 edit56_Callback(hObject, eventdata, handles)
% hObject    handle to edit56 (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 edit56 as text
%        str2double(get(hObject,'String')) returns contents of edit56 as a double


% --- Executes during object creation, after setting all properties.
function edit56_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit56 (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


% --- Executes on button press in pushbutton21.
function pushbutton21_Callback(hObject, eventdata, handles)

entrot=str2double(get(handles.entrot,'string'));

set(handles.edit55,'String',num2str(entrot*0.00025));

guidata(hObject,handles);
% hObject    handle to pushbutton21 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton22.
function pushbutton22_Callback(hObject, eventdata, handles)
```

```
entinch=str2double(get(handles.entinch,'string'));

set(handles.edit56,'String',num2str(entinch/0.00025));
set(handles.convtomm,'String',num2str(entinch*25.4));

guidata(hObject,handles);
% hObject    handle to pushbutton22 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)



function loadxres_Callback(hObject, eventdata, handles)
% hObject    handle to loadxres (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 loadxres as text
%        str2double(get(hObject,'String')) returns contents of loadxres as a double


% --- Executes during object creation, after setting all properties.
function loadxres_CreateFcn(hObject, eventdata, handles)
% hObject    handle to loadxres (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 loadyres_Callback(hObject, eventdata, handles)
% hObject    handle to loadyres (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 loadyres as text
%        str2double(get(hObject,'String')) returns contents of loadyres as a double


% --- Executes during object creation, after setting all properties.
function loadyres_CreateFcn(hObject, eventdata, handles)
% hObject    handle to loadyres (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 loadzres_Callback(hObject, eventdata, handles)
% hObject    handle to loadzres (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 loadzres as text
%        str2double(get(hObject,'String')) returns contents of loadzres as a double


% --- Executes during object creation, after setting all properties.
function loadzres_CreateFcn(hObject, eventdata, handles)
% hObject    handle to loadzres (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


% --- Executes on button press in digitizer.
function digitizer_Callback(hObject, eventdata, handles)
% hObject    handle to digitizer (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%set up the vertical range;
clc
set(handles.text54,'String','Initializing')
acquiretime=str2double(get(handles.acquiretime,'String'));
acquirerate=str2double(get(handles.acquirerate,'String'));
samples=acquiretime*acquirerate;
niScopeObj=handles.niScopeObj;
% handles.info.acquiretime=acquiretime;
% handles.info.acquirerate=acquirerate;
if samples< 4*1024*1024
    set(handles.text54,'String',['Initializing: OK ',num2str(samples)])
end


VerticalRange = str2double(get(handles.verticalrange,'String'));
% disp(['Vertical range is: ', num2str(VerticalRange)])
handles.VerticalRange=VerticalRange;% vertical range is 8 volts
VerticalOffset = str2double(get(handles.edit48,'String'));
VerticalCoupling = 1;
VerticalAttenuation = 1;
ChannelEnable = 1;
```

```
groupObjv = get(niScopeObj, 'Configurationfunctionsvertical');
invoke(groupObjv, 'configurevertical','1',VerticalRange,VerticalOffset,...
   VerticalCoupling,VerticalAttenuation,ChannelEnable)% enabled channel 0

%set up the horizontal range;
RefPosition = 0;
NumRecords = 1;
enforceRealtime = 0;
groupObjh = get(niScopeObj, 'Configurationfunctionshorizontal');
pause(5)
invoke(groupObjh, 'configurehorizontaltiming',acquirerate,samples,...
   RefPosition,NumRecords,enforceRealtime);

% Configure the trigger functions
groupObjt = get(niScopeObj, 'Configurationfunctionstrigger');
% for analog trigger
% % % NISCOPE_VAL_POSITIVE = 1;
% % % NISCOPE_VAL_NEGATIVE = -1;
% % %
% % % triggerSource = '1'; % channel 1 (analog signal). Specifies the trigger source. Refer to
NISCOPE_ATTR_TRIGGER_SOURCE for defined values.
% % % level = 1; % The voltage threshold for the trigger. Refer to
NISCOPE_ATTR_TRIGGER_LEVEL for more information.
% % % slope = NISCOPE_VAL_POSITIVE; %Specifies whether you want a rising edge or a
falling edge to trigger the digitizer. Refer to NISCOPE_ATTR_TRIGGER_SLOPE for more
information.
% % % triggerCoupling = 1; %Applies coupling and filtering options to the trigger signal. Refer to
NISCOPE_ATTR_TRIGGER_COUPLING for more information.
% % % holdoff  = 0;%The length of time the digitizer waits after detecting a trigger before
enabling NI-SCOPE to detect another trigger. Refer to NISCOPE_ATTR_TRIGGER_HOLDOFF
for more information.
% % % % delay  = 0%3e-5;
% % % delay = str2double(get(handles.Delay_Trigger,'String'));
% % % invoke(groupObjt,
'configuretriggeredge',triggerSource,level,slope,triggerCoupling,holdoff,delay);

% for digital trigger
NISCOPE_VAL_POSITIVE = 1;
NISCOPE_VAL_NEGATIVE = 0;

triggerSource = 'PFI1'; % Specifies the trigger source. Refer to
NISCOPE_ATTR_TRIGGER_SOURCE for defined values.
slope = NISCOPE_VAL_POSITIVE; %Specifies whether you want a rising edge or a falling edge
to trigger the digitizer. Refer to NISCOPE_ATTR_TRIGGER_SLOPE for more information.
holdoff  = 0;%The length of time the digitizer waits after detecting a trigger before
%enabling NI-SCOPE to detect another trigger. Refer to NISCOPE_ATTR_TRIGGER_HOLDOFF
for more information.
delay = str2double(get(handles.Delay_Trigger,'String'));
set(handles.text54,'String','Acquiring')

invoke(groupObjt, 'configuretriggerdigital',triggerSource,slope,...
   holdoff,delay);
```

```
WaveformArray = zeros(1, samples);
WfmInfo.absoluteInitialX =  0;
WfmInfo.relativeInitialX =  0;
WfmInfo.xIncrement =  1/handles.acquirerate; % Time delta
WfmInfo.actualSamples =  samples;
WfmInfo.gain =  1;
WfmInfo.reserved1 =  0;
WfmInfo.reserved2 =  0;
niScopeObj=handles.niScopeObj;
groupObja = get(niScopeObj, 'Acquisition');
channelList ='1';
time = (0:1:samples-1)*1/acquirerate;

disp(['Acquire rate: ', num2str(acquirerate),' Acquire time: ', num2str(acquiretime)])
disp(['Trigger delay: ', num2str(delay),' Voltage range: ', num2str(VerticalRange)])
[Waveform_Ch , WfmInfo] =  invoke(groupObja, 'read', channelList, 50,...
            samples, WaveformArray, WfmInfo );
h=figure;plot(time,Waveform_Ch)
xlabel('Time (S)')
ylabel('Voltage (V)')
title(['Digitizer Acquired Signal'])
gtext({['V.R: ',num2str(VerticalRange),'(V), V.O: ',num2str(VerticalOffset),'(V)'],...
   ['X: ',num2str(get(handles.xapos,'string')),...
   ' Y: ',num2str(get(handles.yapos,'string')),...
   ' Z: ',num2str(get(handles.zapos,'string'))]})
% gtext(['V.R: ',num2str(VerticalRange),'(V), V.O: ',num2str(VerticalOffset),'(V)'])
% gtext(['X: ',num2str(get(handles.xapos,'string')),...
%     ' Y: ',num2str(get(handles.yapos,'string')),'\n',...
%     ' Z: ',num2str(get(handles.zapos,'string'))])
digisave=questdlg('Select yes to save','Pt Recording');
%
% handles.info.acquiretime=acquiretime;
% handles.info.acquirerate=acquirerate;
% handles.info.samples=samples;
%
% handles.info.xapos=(str2double(get(handles.xapos,'string')));
% handles.info.yapos=(str2double(get(handles.yapos,'string')));
% handles.info.zapos=(str2double(get(handles.zapos,'string')));
%
% handles.info.daq.VerticalRange=VerticalRange;
% handles.info.daq.Delay=delay;
% % handles.info.pathname=pathname;
% % handles.info.filename=filename;
% [a b c d e f g]=scancalculations;
% dataout.info=c;


if strcmp(digisave,'Yes')
   digidir=uigetdir('C:\Lab_folder\Jay\');
   digifile=input('Enter file name: ','s');
   digidirf=[digidir,'\',digifile,'.mat'];
   save(digidirf, 'Waveform_Ch', 'time');
```

```
   saveas(h,[digidir,'\',digifile,'.jpg'])
end
   set(handles.text54,'String','Done->Ready')

guidata(hObject,handles);

% --- Executes on button press in daqacq.
function daqacq_Callback(hObject, eventdata, handles)
set(handles.text55,'String','Initializing')
acquiretime=str2double(get(handles.acquiretime,'String'));
acquirerate=str2double(get(handles.acquirerate,'String'));
samples=acquiretime*acquirerate;
set(handles.daq.ai,'TriggerConditionValue',1);
set(handles.daq.ai,'samplerate',acquirerate);
set(handles.daq.ai, 'SamplesPerTrigger', samples);
set(handles.daq.ai, 'TriggerRepeat', 1);
set(handles.daq.ai, 'Timeout',15)
clc
set(handles.text55,'String','Acquiring')
% get(handles.daq.ai)

        start(handles.daq.ai);% to start acquiring data
        pause(acquiretime);
        [data, daqtime] = getdata(handles.daq.ai,samples);
        stop(handles.daq.ai);% to stop acquiring data
%           timst=find(daqtime<1.0001 & daqtime>0.999);
%           ab1=mean(data((timst(1)-2000):timst(1)));
%           ab2=mean(data(timst(length(timst)):(timst(length(timst))+2000)));
%           data(timst)=linspace(ab1,ab2,length(timst));
%           comp=max(data(timst,2))-min(data(timst,2));
%           compfrom=find(min(data(timst,2))==data(timst,2));
%           data((compfrom+timst(1)-4):length(data),2)=data((compfrom+timst(1)-
4):length(data),2)+comp;
        assignin('base','daqout',[daqtime data])
        h=figure;plot(daqtime,data(:,2))
xlabel('Time (S)')
ylabel('Voltage (V)')
title(['DAQ Acquired Signal'])
daqsave=questdlg('Select yes to save','Pt Recording');

if strcmp(daqsave,'Yes')
   daqdir=uigetdir('C:\Lab_folder\Jay\');
   daqfile=input('Enter file name: ','s');
   daqdirf=[daqdir,'\',daqfile,'.mat'];
   save(daqdirf, 'Waveform_Ch', 'time');
   saveas(h,[daqdir,'\',daqfile,'.jpg'])
end
   set(handles.text55,'String','Done->Ready')

guidata(hObject,handles);


% hObject    handle to daqacq (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


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



function convtomm_Callback(hObject, eventdata, handles)
% hObject    handle to convtomm (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 convtomm as text
%        str2double(get(hObject,'String')) returns contents of convtomm as a double


% --- Executes during object creation, after setting all properties.
function convtomm_CreateFcn(hObject, eventdata, handles)
% hObject    handle to convtomm (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 edit48_Callback(hObject, eventdata, handles)
% hObject    handle to edit48 (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 edit48 as text
%        str2double(get(hObject,'String')) returns contents of edit48 as a double


% --- Executes during object creation, after setting all properties.
function edit48_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit48 (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


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


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


% --- Executes on button press in HIFU.
function HIFU_Callback(hObject, eventdata, handles)
% hObject    handle to HIFU (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%% To call the HIFU program
evalin('base','hifufocalplane')



% --- Executes on button press in DyWav.
function DyWav_Callback(hObject, eventdata, handles)
% hObject    handle to DyWav (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%% to call the program to view different Ultrasound wave dynamic propagation
a=uigetdir;
a1=uigetfile(a);
run([a '\' a1]);


% The following MATLAB codes should be stored separately in the same
% folder as the main program.

% program for 1MHz wire

%% select the pathname of the datafile
load C:\Lab_folder\Jay\Using_wire\USTscanning\Experiment_10-04-2011c\UST.mat;
pathname='C:\Lab_folder\Jay\Using_wire\USTscanning\Experiment_10-04-2011c\UST';

%% Processing and Analysis
for zpoint=1;
    n=1;

    w1=(dataout.info.daq.xrange_mod/dataout.info.daq.xpoint)...
```

```
        *(dataout.info.daq.acquirerate/(1.481e6))*25.4;
    w=round(w1);
    time = (1:(dataout.info.daq.acquiretime*dataout.info.daq.acquirerate))...
        *1/dataout.info.daq.acquirerate;
%     time_mod=cosd(40)*time;
%     time1=time;
    time=time+dataout.info.daq.Delay;


    for i=1:dataout.info.daq.ypoint
    eval(sprintf('load %s_%d_%d.mat',pathname,i,zpoint));
%     It=cell2mat(It);
    final_max(i,:)=max(It);
    final_min(i,:)=min(It);
    [r e]=size(It);
    for k=1:e
        It(:,k)=It(:,k)-mean(It,2);
    end
    for k=1:e
    It_mod(:,k)=circshift(It(:,k),-((k-1)*w));
    end
    us_image(:,:,i)=moving_average(It,5,1);
    us_image_mod(:,:,i)=It_mod;%moving_average(It_mod,5,1);

    end

% % % % to create a movie file of the sound propagation

%% For Movie
% num_frames_per_second = 10;clc
% pathname3 = uigetdir('C:\Lab_folder\Jay\','Pick folder to store the movie file: ');
% moviename=input('Enter the name of the movie file to be created: ','s');
% moviename1=[pathname3 '\' moviename '_mod.avi'];
% moviename=[pathname3 '\' moviename '.avi'];
% aviobj = avifile ( moviename, 'fps', num_frames_per_second );
%%

disp(['The Axial length is: ',num2str(dataout.info.daq.xrange_mod*25.4)]);
ay=input('Increment of axial axis: ');
disp(['The Lateral length is: ',num2str(dataout.info.daq.yrange_mod*25.4)]);
ax=input('Increment of lateral axis: ');

    for i=1:3:290
        d=squeeze(us_image_mod(i,:,:));
        de=d;
        de=moving_average(de,3,2);
        de=moving_average(de,3,1);
        d=de;
        h=figure(5);%set(h,'Position', [300 300 600 500])
        imagesc(d-mean(mean(d)),[-2.2e-3 2e-3])%,[-2e-3 6e-3]);
        colormap(gray); axis image;colorbar
title(['Time: ',num2str(time(i))]);
disp(['Frame is: ',num2str(i)])
```

```
set(gca,'FontSize',16)
title([{'1MHz UST (Wire)'}; {strcat('Time: ',num2str((time(i)*1e6)-20),'\mu','S')}])

ylabel('Axial (mm)')

by=ay/((dataout.info.daq.xrange_mod/dataout.info.daq.xpoint)*25.4);
cy=[1:floor(dataout.info.daq.xpoint/by)]*ay;
    set(gca, 'YTick',cy*by/ay);
    set(gca, 'YTicklabel',cy);%round(g1*10)/10

xlabel('Lateral (mm)')

bx=ax/((dataout.info.daq.yrange_mod/dataout.info.daq.ypoint)*25.4);
cx=[1:floor(dataout.info.daq.xpoint/bx)]*ax;
    set(gca, 'XTick',cx*bx/ax);
    set(gca, 'XTicklabel',cx);

%%
%    d3=getframe(gcf);
%    aviobj = addframe ( aviobj, d3 );
%%
   end
 %%
%   aviobj = close ( aviobj );
 %%
end

figure(1);set(gcf,'Position',[200 200 800 500])
subplot(212);set(gca,'FontSize',14)
plot((time- time(1)) * 1.481e6,us_image_mod(:,1,30),'k');hold on
plot((time- time(1)) * 1.481e6,us_image_mod(:,17,30),'-.k')
hold off;axis tight
title(['(b) Processed US echo signal'])
xlabel(['Axial (mm)'])
ylabel(['Pressuer Intensity (V)'])
legend([{'US A-line at 1st scan point along axial direction'}...
   {'US A-line at 17th scan point along axial direction'}],'Location','Best')
legend(gca,'boxoff')
subplot(211);set(gca,'FontSize',14)
plot((time- time(1)) * 1.481e6,us_image(:,1,30),'k');hold on
plot((time- time(1)) * 1.481e6,us_image(:,17,30),'-.k')
hold off; axis tight
title(['(a) Acquired US echo signal'])
xlabel(['Axial (mm)'])
ylabel(['Pressuer Intensity (V)'])
legend([{'US A-line at 1st scan point along axial direction'}...
   {'US A-line at 17th scan point along axial direction'}],'Location','Best')
legend(gca,'boxoff')


% program for 2.25MHz wire
clear all
clc
```

```
close all

load C:\Lab_folder\Jay\Using_wire\USTscanning\Experiment_09-30-2011\UST.mat;
pathname='C:\Lab_folder\Jay\Using_wire\USTscanning\Experiment_09-30-2011\UST';


for zpoint=2;
   n=1;

   w1=(dataout.info.daq.xrange_mod/dataout.info.daq.xpoint)...
      *(dataout.info.daq.acquirerate/(1.481e6))*25.4;
   w=round(w1);
   time = (1:(dataout.info.daq.acquiretime*dataout.info.daq.acquirerate))...
      *1/dataout.info.daq.acquirerate;
   time_mod=cosd(40)*time;
   time=time+30e-6;


   for i=1:dataout.info.daq.ypoint
   eval(sprintf('load %s_%d_%d.mat',pathname,i,zpoint));
%     It=cell2mat(It);
   final_max(i,:)=max(It);
   final_min(i,:)=min(It);
   [r e]=size(It);
   for k=1:e
      It(:,k)=It(:,k)-mean(It,2);
   end
   for k=1:e
   It_mod(:,k)=circshift(It(:,k),-((k-1)*w));
   end
   us_image_mod(:,:,i)=moving_average(It_mod,5,1);

   end

% % % % to create a movie file of the sound propagation

%% For Movie
num_frames_per_second = 10;clc
pathname3 = uigetdir('C:\Lab_folder\Jay\','Pick folder to store the movie file: ');
moviename=input('Enter the name of the movie file to be created: ','s');
moviename1=[pathname3 '\' moviename '_mod.avi'];
moviename=[pathname3 '\' moviename '.avi'];
aviobj = avifile ( moviename, 'fps', num_frames_per_second );
%%

disp(['The Axial length is: ',num2str(dataout.info.daq.xrange_mod*25.4)]);
ay=input('Increment of axial axis: ');
disp(['The Lateral length is: ',num2str(dataout.info.daq.yrange_mod*25.4)]);
ax=input('Increment of lateral axis: ');

   for i=400:550
      d=squeeze(us_image_mod(i,:,:));
      de=d;
```

```
        de=moving_average(de,3,2);
        de=moving_average(de,3,1);
        d=de;
        h=figure(5);%set(h,'Position', [300 300 600 500])
        imagesc(d,[-0.8e-3 1.6e-3])%,[-2e-3 6e-3]);
        colormap(gray); axis image;colorbar
title(['Time: ',num2str(time(i))]);
disp(['Frame is: ',num2str(i)])
set(gca,'FontSize',16)
title([{'2.25MHz UST (Wire)'}; {strcat('Time: ',num2str((time(i)*1e6)-20),'\mu','S')}])

ylabel('Axial (mm)')

by=ay/((dataout.info.daq.xrange_mod/dataout.info.daq.xpoint)*25.4);
cy=[1:floor(dataout.info.daq.xpoint/by)]*ay;
    set(gca, 'YTick',cy*by/ay);
    set(gca, 'YTicklabel',cy);%round(g1*10)/10

xlabel('Lateral (mm)')

bx=ax/((dataout.info.daq.yrange_mod/dataout.info.daq.ypoint)*25.4);
cx=[1:floor(dataout.info.daq.xpoint/bx)]*ax;
    set(gca, 'XTick',cx*bx/ax);
    set(gca, 'XTicklabel',cx);

%%
   d3=getframe(gcf);
   aviobj = addframe ( aviobj, d3 );
%%
   end
 %%
 aviobj = close ( aviobj );
 %%
end

% program for 5MHz wire
clear all
clc
close all

load C:\Lab_folder\Jay\Using_wire\USTscanning\Experiment_09-30-2011a\UST.mat;
pathname='C:\Lab_folder\Jay\Using_wire\USTscanning\Experiment_09-30-2011a\UST';


for zpoint=2;
   n=1;

   w1=(dataout.info.daq.xrange_mod/dataout.info.daq.xpoint)...
      *(dataout.info.daq.acquirerate/(1.481e6))*25.4;
   w=round(w1);
   time = (1:(dataout.info.daq.acquiretime*dataout.info.daq.acquirerate))...
      *1/dataout.info.daq.acquirerate;
%     time_mod=cosd(40)*time;
```

```
   time=time+30e-6;


   for i=1:dataout.info.daq.ypoint
   eval(sprintf('load %s_%d_%d.mat',pathname,i,zpoint));
%     It=cell2mat(It);
   final_max(i,:)=max(It);
   final_min(i,:)=min(It);
   [r e]=size(It);
%     for k=1:e
%         It(:,k)=It(:,k)-mean(It,2);
%     end
   for k=1:e
   It_mod(:,k)=circshift(It(:,k),-((k-1)*w));
   end
   us_image_mod(:,:,i)=moving_average(It_mod,5,1);

   end

% % % % to create a movie file of the sound propagation

%% For Movie
num_frames_per_second = 10;clc
pathname3 = uigetdir('C:\Lab_folder\Jay\','Pick folder to store the movie file: ');
moviename=input('Enter the name of the movie file to be created: ','s');
moviename1=[pathname3 '\' moviename '_mod.avi'];
moviename=[pathname3 '\' moviename '.avi'];
aviobj = avifile ( moviename, 'fps', num_frames_per_second );
%%

disp(['The Axial length is: ',num2str(dataout.info.daq.xrange_mod*25.4)]);
ay=input('Increment of axial axis: ');
disp(['The Lateral length is: ',num2str(dataout.info.daq.yrange_mod*25.4)]);
ax=input('Increment of lateral axis: ');

   for i=90:270
       d=squeeze(us_image_mod(i,:,:));
       de=d;
       de=moving_average(de,3,2);
       de=moving_average(de,3,1);
       d=de;
       h=figure(5);%set(h,'Position', [300 300 600 500])

       by=ay/((dataout.info.daq.xrange_mod/dataout.info.daq.xpoint)*25.4);
cy=[1:floor(dataout.info.daq.xpoint/by)]*ay;

bx=ax/((dataout.info.daq.yrange_mod/dataout.info.daq.ypoint)*25.4);
cx=[1:floor(dataout.info.daq.xpoint/bx)]*ax;

%axis image;
    set(h,'Position',[250 250 500 ((cy(length(cy))/cx(length(cx)))-0.2)*500])

       imagesc(d,[-1.8e-4 4.2e-4])%,[-2e-3 6e-3]);
```

```
      colormap(gray); colorbar;

disp(['Frame is: ',num2str(i)])
set(gca,'FontSize',16)
title([{'5MHz UST (Wire)'}; {strcat('Time: ',num2str((time(i)*1e6)-20),'\mu','S')}])

ylabel('Axial (mm)')

    set(gca, 'YTick',cy*by/ay);
    set(gca, 'YTicklabel',cy);%round(g1*10)/10

xlabel('Lateral (mm)')

    set(gca, 'XTick',cx*bx/ax);
    set(gca, 'XTicklabel',cx);



%%
   d3=getframe(gcf);
   aviobj = addframe ( aviobj, d3 );
%%
   end
  %%
  aviobj = close ( aviobj );
  %%
end


% program for 1MHz hydrophone
clear all
clc
close all

load C:\Lab_folder\Jay\Using_wire\USTscanning\Experiment_10-13-2011d\UST.mat;
pathname='C:\Lab_folder\Jay\Using_wire\USTscanning\Experiment_10-13-2011d\UST';


for zpoint=1;
   n=1;

   w1=(dataout.info.daq.xrange_mod/dataout.info.daq.xpoint)...
      *(dataout.info.daq.acquirerate/(1.481e6))*25.4;
   w=round(w1);
   time = (1:(dataout.info.daq.acquiretime*dataout.info.daq.acquirerate))...
      *1/dataout.info.daq.acquirerate;
   time_mod=cosd(40)*time;
   time=time+dataout.info.daq.Delay;


   for i=1:dataout.info.daq.ypoint
   eval(sprintf('load %s_%d_%d.mat',pathname,i,zpoint));
   It=cell2mat(It);
```

```
   final_max(i,:)=max(It);
   final_min(i,:)=min(It);
   [r e]=size(It);
   for gh=1:e
      It(:,gh)=It(:,gh)-mean(It(:,gh));
   end
%     for k=1:e
%        It(:,k)=It(:,k)-mean(It,2);
%     end
%     for k=1:e
%     It_mod(:,k)=circshift(It(:,k),-((k-1)*w));
%     end
   us_image(:,:,i)=moving_average(It,3,1);
%     us_image_mod(:,:,i)=moving_average(It_mod,5,1);
%     us_image(:,:,i)=moving_average(It,3,1);

   end

% % % % to create a movie file of the sound propagation

% num_frames_per_second = 10;clc
% pathname3 = uigetdir('C:\Lab_folder\Jay\','Pick folder to store the movie file: ');
% moviename=input('Enter the name of the movie file to be created: ','s');
% moviename1=[pathname3 '\' moviename '_mod.avi'];
% moviename=[pathname3 '\' moviename '.avi'];
% aviobj = avifile ( moviename, 'fps', num_frames_per_second );

disp(['The Axial length is: ',num2str(dataout.info.daq.xrange_mod*25.4 *.41)]);
ay=input('Increment of axial axis: ');
disp(['The Lateral length is: ',num2str(dataout.info.daq.yrange_mod*25.4)]);
ax=input('Increment of lateral axis: ');

by=ay/((dataout.info.daq.xrange_mod/dataout.info.daq.xpoint)*25.4);
cy=[1:floor(41/by)]*ay;

bx=ax/((dataout.info.daq.yrange_mod/dataout.info.daq.ypoint)*25.4);
cx=[1:floor(dataout.info.daq.xpoint/bx)]*ax;

   for i=1600:10:2400
%        i=2100;
      d=squeeze(us_image(i,:,:));
      de=d;
      de=moving_average(de,3,1);
      de=moving_average(de,3,2);
      d=de;
      h=figure(5);%set(h,'Position', [300 300 600 500])
      imagesc(d(40:80,:)-mean(mean(d(40:80,:))),[-2.2e-4 2.2e-4])%,[-2e-3 6e-3]);
      colormap(gray); axis image;colorbar
title(['Time: ',num2str(time(i))]);
disp(['Frame is: ',num2str(i)])
set(gca,'FontSize',16)
title([{'1MHz UST (Hydrophone)'}; {strcat('Time: ',num2str(time(i)*1e6),'\mu','S')}])
```

```
ylabel('Axial (mm)')

    set(gca, 'YTick',cy*by/ay);
    set(gca, 'YTicklabel',cy);%round(g1*10)/10

xlabel('Lateral (mm)')

    set(gca, 'XTick',cx*bx/ax);
    set(gca, 'XTicklabel',cx);


%    d3=getframe(gcf);
%    aviobj = addframe ( aviobj, d3 );

   end

   select=input('enter the frame to cal wavelength: ');
   d=squeeze(us_image(select,:,:));
      de=d;
      de=moving_average(de,3,2);
      de=moving_average(de,3,1);
      de=de(30:70,:)-mean(mean(de(30:70,:)));
      [a1 b1]=find(max(max(de))==de);
%        c=cy*by/ay;
      aline=de(:,b1);plot(aline);axis tight
      [a2 b2]=find(max(aline(1:(a1-5)))==aline);
      disp(['The wavelength is: ',num2str((((a1-a2)*0.8)/100)*25.4)])

%    aviobj = close ( aviobj );
end

% program for 2.25MHz interference
clear all
clc
close all
%% previous
% load C:\Lab_folder\Jay\Using_wire\USTscanning\Experiment_10-08-2011\UST.mat;
% pathname='C:\Lab_folder\Jay\Using_wire\USTscanning\Experiment_10-08-2011\UST';


%% changed
load C:\Lab_folder\Jay\Using_wire\USTscanning\Experiment_10-30-2011a\UST1.mat;
pathname='C:\Lab_folder\Jay\Using_wire\USTscanning\Experiment_10-30-2011a\UST1';

%%
for zpoint=1;
   n=1;

   w1=(dataout.info.daq.xrange_mod/dataout.info.daq.xpoint)...
      *(dataout.info.daq.acquirerate/(1.481e6))*25.4;
   w=round(w1);
   time = (1:(dataout.info.daq.acquiretime*dataout.info.daq.acquirerate))...
      *1/dataout.info.daq.acquirerate;
```

```
   time_mod=cosd(40)*time;
   time=time+dataout.info.daq.Delay;


   for i=1:dataout.info.daq.ypoint
   eval(sprintf('load %s_%d_%d.mat',pathname,i,zpoint));
   It=cell2mat(It);
   final_max(i,:)=max(It);
   final_min(i,:)=min(It);
   [r e]=size(It);
%     for k=1:e
%         It(:,k)=It(:,k)-mean(It,2);
%     end
%     for k=1:e
%     It_mod(:,k)=circshift(It(:,k),-((k-1)*w));
%     end
   us_image(:,:,i)=moving_average(It,5,1);

   end

% % % % to create a movie file of the sound propagation

%% For Movie
% num_frames_per_second = 10;clc
% pathname3 = uigetdir('C:\Lab_folder\Jay\','Pick folder to store the movie file: ');
% moviename=input('Enter the name of the movie file to be created: ','s');
% moviename1=[pathname3 '\' moviename '_mod.avi'];
% moviename=[pathname3 '\' moviename '.avi'];
% aviobj = avifile ( moviename, 'fps', num_frames_per_second );
%%

disp(['The Axial length is: ',num2str(dataout.info.daq.xrange_mod*25.4)]);
ay=input('Increment of axial axis: ');
disp(['The Lateral length is: ',num2str(dataout.info.daq.yrange_mod*25.4)]);
ax=input('Increment of lateral axis: ');

   for i=750:2:980 % 690:830
      d=squeeze(us_image(i,:,:));
      de=d;
      de=moving_average(de,3,2);
      de=moving_average(de,3,1);
      d=de;
      h=figure(5);%set(h,'Position', [300 300 600 500])
      imagesc(d,[-8.4e-4 6.4e-4])%,[-9e-4 7e-4]);
      colormap(gray); axis image;colorbar

disp(['Frame is: ',num2str(i)])
set(gca,'FontSize',16)
title([{'2.25MHz UST Interference'}; {strcat('Time: ',num2str((time(i)*sqrt(2)*1e6)-3),'\mu','S')}])

ylabel('Axial (mm)')
by=ay/((dataout.info.daq.xrange_mod/dataout.info.daq.xpoint)*25.4);
cy=(1:floor(dataout.info.daq.xpoint/by))*ay;
```

```
    set(gca, 'YTick',cy*by/ay);
    set(gca, 'YTicklabel',cy);%round(g1*10)/10

xlabel('Lateral (mm)')
bx=ax/((dataout.info.daq.yrange_mod/dataout.info.daq.ypoint)*25.4);
cx=(1:floor(dataout.info.daq.xpoint/bx))*ax;
    set(gca, 'XTick',cx*bx/ax);
    set(gca, 'XTicklabel',cx);

%%
%    d3=getframe(gcf);
%    aviobj = addframe ( aviobj, d3 );
%%
   end
  %%
%   aviobj = close ( aviobj );
  %%
end


% program for 2.25MHz interference with 12mm thick chicken tissue
clear all
clc
close all

load C:\Lab_folder\Jay\Using_wire\USTscanning\Experiment_11-03-2011b\UST2.mat;
pathname='C:\Lab_folder\Jay\Using_wire\USTscanning\Experiment_11-03-2011b\UST2';


for zpoint=1;
   n=1;

%    w1=(dataout.info.daq.xrange_mod/dataout.info.daq.xpoint)...
%        *(dataout.info.daq.acquirerate/(1.481e6))*25.4;
%    w=round(w1);
   time = (1:(dataout.info.daq.acquiretime*dataout.info.daq.acquirerate))...
       *1/dataout.info.daq.acquirerate;
%    time_mod=cosd(40)*time;
   time=time+dataout.info.daq.Delay;


   for i=1:dataout.info.daq.ypoint
   eval(sprintf('load %s_%d_%d.mat',pathname,i,zpoint));
   It=cell2mat(It);
   final_max(i,:)=max(It);
   final_min(i,:)=min(It);
   [r e]=size(It);
   us_image(:,:,i)=moving_average(It,5,1);

   end

% % % % to create a movie file of the sound propagation
```

```
%% For Movie
num_frames_per_second = 10;clc
pathname3 = uigetdir('C:\Lab_folder\Jay\','Pick folder to store the movie file: ');
moviename=input('Enter the name of the movie file to be created: ','s');
moviename1=[pathname3 '\' moviename '_mod.avi'];
moviename=[pathname3 '\' moviename '.avi'];
aviobj = avifile ( moviename, 'fps', num_frames_per_second );
%%

disp(['The Axial length is: ',num2str(dataout.info.daq.xrange_mod*25.4)]);
ay=input('Increment of axial axis: ');
disp(['The Lateral length is: ',num2str(dataout.info.daq.yrange_mod*25.4)]);
ax=input('Increment of lateral axis: ');

   for i=720:850
      d=squeeze(us_image(i,:,:));
      de=d;
      de=moving_average(de,3,2);
      de=moving_average(de,3,1);
      d=de;
      h=figure(5);%set(h,'Position', [300 300 600 500])
      imagesc(d,[-10e-4 8e-4])%,[-2e-3 6e-3]);
      colormap(gray); axis image;colorbar

disp(['Frame is: ',num2str(i)])
set(gca,'FontSize',16)
title([{'2.25MHz UST Interference (12mm Chicken Tissue)'};...
   {strcat('Time: ',num2str((time(i)*sqrt(2)*1e6)-9),'\mu','S')}])

ylabel('Axial (mm)')
by=ay/((dataout.info.daq.xrange_mod/dataout.info.daq.xpoint)*25.4);
cy=(1:floor(dataout.info.daq.xpoint/by))*ay;
    set(gca, 'YTick',cy*by/ay);
    set(gca, 'YTicklabel',cy);
xlabel('Lateral (mm)')
bx=ax/((dataout.info.daq.yrange_mod/dataout.info.daq.ypoint)*25.4);
cx=(1:floor(dataout.info.daq.xpoint/bx))*ax;
    set(gca, 'XTick',cx*bx/ax);
    set(gca, 'XTicklabel',cx);

%%
   d3=getframe(gcf);
   aviobj = addframe ( aviobj, d3 );
%%
   end
 %%
 aviobj = close ( aviobj );
 %%
end


% program for 2.25MHz interference with 24mm thick chicken tissue
clear all
```

```
clc
close all

load C:\Lab_folder\Jay\Using_wire\USTscanning\Experiment_11-03-2011b\UST3.mat;
pathname='C:\Lab_folder\Jay\Using_wire\USTscanning\Experiment_11-03-2011b\UST3';


for zpoint=1;
   n=1;

%     w1=(dataout.info.daq.xrange_mod/dataout.info.daq.xpoint)...
%        *(dataout.info.daq.acquirerate/(1.481e6))*25.4;
%     w=round(w1);
   time = (1:(dataout.info.daq.acquiretime*dataout.info.daq.acquirerate))...
      *1/dataout.info.daq.acquirerate;
%     time_mod=cosd(40)*time;
   time=time+dataout.info.daq.Delay;


   for i=1:dataout.info.daq.ypoint
   eval(sprintf('load %s_%d_%d.mat',pathname,i,zpoint));
   It=cell2mat(It);
   final_max(i,:)=max(It);
   final_min(i,:)=min(It);
   [r e]=size(It);
   us_image(:,:,i)=moving_average(It,5,1);

   end

% % % % to create a movie file of the sound propagation

%% For Movie
num_frames_per_second = 10;clc
pathname3 = uigetdir('C:\Lab_folder\Jay\','Pick folder to store the movie file: ');
moviename=input('Enter the name of the movie file to be created: ','s');
moviename1=[pathname3 '\' moviename '_mod.avi'];
moviename=[pathname3 '\' moviename '.avi'];
aviobj = avifile ( moviename, 'fps', num_frames_per_second );
%%

disp(['The Axial length is: ',num2str(dataout.info.daq.xrange_mod*25.4)]);
ay=input('Increment of axial axis: ');
disp(['The Lateral length is: ',num2str(dataout.info.daq.yrange_mod*25.4)]);
ax=input('Increment of lateral axis: ');

   for i=720:850
      d=squeeze(us_image(i,:,:));
      de=d;
      de=moving_average(de,3,2);
      de=moving_average(de,3,1);
      d=de;
      h=figure(5);%set(h,'Position', [300 300 600 500])
      imagesc(d,[-10e-4 8e-4])%,[-2e-3 6e-3]);
```

```
      colormap(gray); axis image;colorbar

disp(['Frame is: ',num2str(i)])
set(gca,'FontSize',16)
title([{'2.25MHz UST Interference (24mm Chicken Tissue)'};...
   {strcat('Time: ',num2str((time(i)*sqrt(2)*1e6)-11),'\mu','S')}])

ylabel('Axial (mm)')
by=ay/((dataout.info.daq.xrange_mod/dataout.info.daq.xpoint)*25.4);
cy=(1:floor(dataout.info.daq.xpoint/by))*ay;
    set(gca, 'YTick',cy*by/ay);
    set(gca, 'YTicklabel',cy);%round(g1*10)/10

xlabel('Lateral (mm)')
bx=ax/((dataout.info.daq.yrange_mod/dataout.info.daq.ypoint)*25.4);
cx=(1:floor(dataout.info.daq.xpoint/bx))*ax;
    set(gca, 'XTick',cx*bx/ax);
    set(gca, 'XTicklabel',cx);

% %%
   d3=getframe(gcf);
   aviobj = addframe ( aviobj, d3 );
%%
   end
 %%
 aviobj = close ( aviobj );
 %%
end



% For the HIFU focal measurement

clear all
% close all
a=uigetdir;
a1=input('enter the file name: ','s');
a=[a '\UST' a1];
load(a)
time=linspace(0,dataout.info.daq.acquiretime,dataout.info.daq.acquirerate...
   *dataout.info.daq.acquiretime);

%% Calibration of thermocouple

% for new data 0.001inch TC calibration
temp1=[25:45];
temp2=[26:45];
osc1=-1.*[40 89.4 147 200 263 313 361 421 493 546 609 668 718 786 839 908 954 1030 1093
1156 1211]
osc2=-1.*[4 38 81 146 199 255 317 378 439 513 577 639 684 753 796 860 913 976 1030 1087]
v(1,:)=polyfit(temp1,osc1,1);
v(2,:)=polyfit(temp2,osc2,1);
disp('Slope is: mV/degree C')
```

```
disp(mean(v(:,1)));

% [m,n]=polyfit(x,y,1);
slope=mean(v(:,1));%m(1)*1e-3;%*1e3
figure(100);
set(gcf,'Position',[500 100 470 300])
plot(temp2,osc2);
gtext(['Slope is: ',num2str(slope),' mV/ \circC']);
ylabel('Mean voltage (mV)')
xlabel('Temperature (\circC)')
title('Recorded for 200mV/div; 1Mohm')

%%% for delay 0 sec and acq 10 sec

finalmin=1000;
finalmax=-1000;

dataout.info.daq

if isfield(dataout.info,'scantype')==1
   plane=(dataout.info.scantype);
else
plane=input('Enter either yx or zx: ','s');
end

if strcmp(plane,'yx')==1
   e.point=dataout.info.daq.ypoint;
   e.range_mod=dataout.info.daq.yrange_mod;
   ylab='Transverse';
else
   e.point=dataout.info.daq.zpoint;
   e.range_mod=dataout.info.daq.zrange_mod;
   ylab='Depth';
end

div=2;

%% for line measurement for HIFU heat distribution just xpoint
for j=1:e.point
for i=1:dataout.info.daq.xpoint
%     eval(sprintf('load %s_%d_1_%d_tc.mat',a,j,i));
%     load([a,'_y',num2str(j),'_x1_z',num2str(i),'_p.mat']);
%     load([a,'_',num2str(j),'_1_',num2str(i),'_tc.mat']);
   load([a,'_',num2str(j),'_1_',num2str(i),'_tcdaq.mat']);
% daqtime=time;
   % Fs=1/daqtime(2);
% Wp=[((50)./(Fs/2))];%((lowlimit)./(Fs/2)) ,
% [q,w]=cheby1(4,0.5,Wp,'low');%*1e6
   I=data(:,2)';time=daqtime;

   % for ust5&6.mat for 7.5MHz
%        I=I(2000:length(I));
%        time=time(1:(length(time)-2000+1));
```

```
%     b=cell2mat(I);% It replaced by I
    [sm_sig]=moving_average(I,1000,2);% I
    for k=1%:size(I,2)
%         b=cell2mat(I(1,k));
%         sm_sig=moving_average(I,1000,2);
%         sm_sig=I;%filtfilt(q,w,I);
%         sm_sig1=(abs(sm_sig-max(sm_sig)));
        sm_sig2=(abs(sm_sig-max(sm_sig)));
%         sm_sig3=[sm_sig2(1:1000) sm_sig2(4000:length(sm_sig2))];
%         vid(:,i)=sm_sig(1,4*dataout.info.daq.acquirerate:6*dataout.info.daq.acquirerate);%0.65
        %vidc(:,i)=sm_sig1(1,4*dataout.info.daq.acquirerate:6*dataout.info.daq.acquirerate);
        %vid1(:,i)=I(1,4.8*dataout.info.daq.acquirerate:7*dataout.info.daq.acquirerate);%0.65
        vid2(:,i)=sm_sig2(1,1:div:round(0.9*(1/daqtime(2))));%dataout.info.daq.acquirerate));%0.65,
round(0*dataout.info.daq.acquirerate)
        [first(j,i), d1] =max(sm_sig(1,1:(1/(time(2))
*0.11)));%round(0.25*(1/daqtime(2))));%(1*dataout.info.daq.acquirerate:2.2*dataout.info.daq.acq
uirerate));
        [second(j,i), d2]=min(sm_sig(1,1:(1/(time(2)) *
0.11)));%round(0.25*(1/daqtime(2))));%(1:5*dataout.info.daq.acquirerate));
        %((1/(time(2)) * 0.001))
        if finalmax<max(sm_sig2)
            finalmax=max(sm_sig2);
        end
%         if finalmin>second(j,i)
%             finalmin=second(j,i)
%         end
        meansig=max(sm_sig2);%1:4.5*dataout.info.daq.acquirerate
        minsig=min(sm_sig2);
        final(j,i)=first(j,i)-second(j,i);
        finalsig(j,i)=meansig-minsig;
        h1=figure(20);subplot(121)
        plot(time,I);hold on%(:,k)
        plot(time,sm_sig,'c');%(:,k)
        plot(time(d1+round((1/(time(2)) * 0))),first(j,i),'xr')
        plot(time(d2+round((1/(time(2)) * 0))),second(j,i),'xr')
hold off
xlabel(['Time: ',num2str(time(d2+round((1/(time(2)) * 0))))])
% set(h1,'Position',[1000 200 800 800])
% h2=figure(20)
subplot(122)
plot(time,sm_sig2);
% set(h2,'Position',[100 200 800 800])
    end
disp(['y is: ',num2str(j),'; x is: ',num2str(i)])
end
% fvid(j)={vid};clear vid
% fvid1(j)={vid1};clear vid1
% fvidc(j)={vidc};clear vidc
fvid2(j)={vid2};clear vidc
final1(j,:)=final(j,:);
% for j=1:51
if rem(j,2)==1
```

```
        final(j,:)=fliplr(final(j,:));
        finalsig(j,:)=fliplr(finalsig(j,:));
end
% end

% if rem(j,2)==0
%           final1(j,:)=fliplr(final(j,:));
% end

end
% for i=1:10
%     if rem(1,2)==1
%     re(i,:)=fliplr(final(i,:));
%     end
% end
% (dataout.info.daq.yposition+e.range_mod/2)/0.00025
% save 'C:\Lab_folder\Jay\HIFU\hifu\heat_size\15' final dataout;

%%% for line measurement for HIFU heat distribution just xpoint wit delay 5
%%% and acquisition 10 sec
% % % % for j=1%:e.point
% % % % for i=1:dataout.info.daq.xpoint
% % % %     eval(sprintf('load %s_%d_1_%d.mat',a,j,i));
% % % % %     b=cell2mat(It);
% % % % %     [sm_sig]=moving_average(b,1000,1);
% % % %     for k=1%:size(I,2)
% % % % %         b=cell2mat(I(1,k));
% % % %         [sm_sig]=moving_average(I,1000,2);
% % % %         [first(j,i),
d1]=max(sm_sig(1*dataout.info.daq.acquirerate:2.2*dataout.info.daq.acquirerate));
% % % %         [second(j,i),
d2]=min(sm_sig(2.2*dataout.info.daq.acquirerate:4.8*dataout.info.daq.acquirerate));
% % % %         final(j,i)=first(j,i)-second(j,i);
% % % %         figure(k)
% % % %         plot(time,I);hold on%(:,k)
% % % %         plot(time,sm_sig,'c');%(:,k)
% % % %         plot(time(d1+1*dataout.info.daq.acquirerate),first(j,i),'xr')
% % % %         plot(time(round(d2+2.2*dataout.info.daq.acquirerate)),second(j,i),'xr')
% % % % hold off
% % % %     end
% % % % disp(['y is: ',num2str(j),'; x is: ',num2str(i)])
% % % % end
% % % % final1(j,:)=final(j,:);
% % % % if rem(j,2)==1
% % % %           final(j,:)=fliplr(final(j,:));
% % % % end
% % % %
% % % % if rem(j,2)==0
% % % %           final1(j,:)=fliplr(final(j,:));
% % % % end
% % % %
% % % % end
```

```
 disp(['The Lateral length is: ',num2str(dataout.info.daq.xrange_mod*25.4)]);
    ax=input('Increment of Lateral axis: ');
    disp(['The ',ylab,' length is: ',num2str(e.range_mod*25.4)]);
    ay=input('Increment of Axial axis: ');

    by=ay/((e.range_mod/e.point)*25.4);
    cy=(1:floor(e.point/by))*ay;

    bx=ax/((dataout.info.daq.xrange_mod/dataout.info.daq.xpoint)*25.4);
    cx=(1:floor(dataout.info.daq.xpoint/bx))*ax;

%     figure;plot((final))
% finalsig=finalchg;
% finalchg=finalsig;
% finalsig=finalchg;
% finalsig=finalsig(:,1:41);
finalnew=finalsig;
finalnew2=final1;

%% To shift the image
for j=1: size(finalsig,1)
if rem(j,2)==1
          finalnew(j,:)=circshift(finalsig(j,:),[0 1]);
          finalnew2(j,:)=circshift(final1(j,:),[0 -1]);
end
end
maxcm=max(max(finalsig))
finalnew1=moving_average(finalsig,2,1);
% figure;imagesc(finalnew1);
maxcm1=max(max(finalnew1));
finalnew1=finalnew1*(maxcm/maxcm1);
% finalnew1(find(finalnew1==min(min(finalnew1))))=min(min(finalsig));

namecall=[{'finalnew'} {'finalsig'} {'finalnew1'}];
nam=[{'Shifted Image'} {'Raw Image'} {'Smoothed Image'}];

for i=1:length(namecall)
    eval(sprintf('call=%s;',char(namecall(i))));
h=figure;
subplot(121);
    imagesc((call-min(min(call))));colorbar;%axis image
%     set(gcf,'Position',[200 200 900*(dataout.info.daq.xrange_mod*25.4/(e.range_mod*25.4))
650*(1)])
%     set(gcf,'Position',[200 -200 720 400])
    set(gca, 'XTick',cx*bx/ax);
    set(gca, 'XTicklabel',cx);
    set(gca, 'YTick',cy*by/ay);
    set(gca, 'YTicklabel',cy);%round(g1*10)/10
    xlabel('Lateral (mm)','FontSize',14,'FontName','Times New Roman')
    ylabel([ylab,' (mm)'],'FontSize',14,'FontName','Times New Roman')
    title(nam(i),'FontSize',14,'FontName','Times New Roman')
    set(gca,'FontSize',14);
subplot(122);% figure
```

```
   imagesc((call-min(min(call)))/(-slope*1e-3));colorbar;%axis image
%     set(gcf,'Position',[500 200 900*(dataout.info.daq.xrange_mod*25.4/(e.range_mod*25.4))
650*(1)])
%     set(gcf,'Position',[200 -200 720 400])
   set(gca, 'XTick',cx*bx/ax);
   set(gca, 'XTicklabel',cx);
   set(gca, 'YTick',cy*by/ay);
   set(gca, 'YTicklabel',cy);%round(g1*10)/10
   xlabel('Lateral (mm)','FontSize',14,'FontName','Times New Roman')
   ylabel([ylab,' (mm)'],'FontSize',14,'FontName','Times New Roman')
   title(['Temperature (\circC)'],'FontSize',14,'FontName','Times New Roman')
   set(gca,'FontSize',14);
%     set(gcf,'Position',[200 200 750 400])
% Create textbox
annotation(h,'textbox',...
   [0.107790378006871 0 0.0392061855670103 0.0725],'String',{'(a)'},...
   'FontSize',16,...
   'FontName','Agency FB',...
   'FitBoxToText','off',...
   'LineStyle','none');

% Create textbox
annotation(h,'textbox',...
   [0.550457044673533 0.0025 0.0392061855670103 0.0725],'String',{'(b)'},...
   'FontSize',16,...
   'FontName','Agency FB',...
   'FitBoxToText','off',...
   'LineStyle','none');
end
   final_max1=finalnew1;
%     final_max1=finalnew;
%     glatlow=input('enter glat lower value: ');
%     glathigh=input('enter glat higher value: ');
%     glat=final-min(final);%glat=glat(1,glatlow:glathigh-1);
%     glat=((glat-min(glat))/max(glat-min(glat))); Normalize

      max_signal=zeros(size(final_max1));
      [o,p]=ind2sub(size(final_max1),find(final_max1==max(max(final_max1))));
      max_signal(:,p)=final_max1(:,p);
      max_signal(o,:)=final_max1(o,:);
      h=figure;mesh(max_signal);
      fwhm_axial=linspace(0,(e.range_mod...
         *25.4),size(finalsig,1));
      fwhm_lateral=linspace(0,(dataout.info.daq.xrange_mod...
         *25.4),dataout.info.daq.xpoint);%glathigh-glatlow

%% start of lateral FWHM calculation
      t=0.5;%input('Enter the fraction for Lateral FWHM: ');
      t1=num2str(t,6);

      glat=final_max1(o,:);%glat=glat-min(glat);
      h=figure;subplot(141);plot(fwhm_lateral,glat);axis tight
      set(gcf,'Position',[500 100 970 400])
```

```
        title('Lateral','FontSize',12,'FontName','Times New Roman');
        xlabel('Length (mm)','FontSize',12,'FontName','Times New Roman')
        ylabel('Voltage (V)','FontSize',12,'FontName','Times New Roman')

%% using FWHM method
%     [a1,b1]=max(glat)
%     Lmm1=max(find(glat(1:b1)<=((t)*a1)))
%     L1=glat(max(find(glat(1:b1)<=((t)*a1))))
%     Lmm2=min(find(glat((b1+1):length(glat))<((t)*a1)))
%     L2=glat(max(find(glat(b1+1:length(glat))<((t)*a1))))
%     a3=b1+min(find(glat(b1:length(glat))<((t)*a1)));
%     hold on
%     plot(fwhm_lateral(b1+Lmm2),max(glat)*(t),'rx');%max(glat)*(t)%fwhm_lateral(a2),
%     plot(fwhm_lateral(Lmm1),max(glat)*(t),'rx');
%     hold off
%
%     disp(['Lateral '])
%     disp(['Lateral: ',num2str(fwhm_lateral(Lmm2+b1)-fwhm_lateral(Lmm1))])
%     gtext({['(FWHM): ',num2str(fwhm_lateral(Lmm2+b1)-
fwhm_lateral(Lmm1))],[num2str(t),'*Max']})
%     subplot(122);plot(fwhm_lateral,glat/(-slope*1e-3));%(1:(size(fwhm_axial,2)-4))
%         title('Lateral');axis tight
%         xlabel('Length (mm)')
%         ylabel('Temperature (oC)')
%% using interpolation
xi(1,1) = interp1(glat(1:(find(max(glat)==glat))),fwhm_lateral(1:(find(max(glat)==glat)))...
    ,max(glat)*(t));
xi(1,2) =
interp1(glat((find(max(glat)==glat)):length(glat)),fwhm_lateral((find(max(glat)==glat)):length(glat))..
.
    ,max(glat)*(t));
hold on
plot(xi(1,1),max(glat)*(t),'rx');%max(glat)*(t)%fwhm_lateral(a2),
plot(xi(1,2),max(glat)*(t),'rx');
hold off
% saveas(h,[dirnam,'/',num2str(m),'.fig']);m=m+1;
clc
disp(['Lateral '])
disp(['    ' t1]);
disp(['Lateral: ',num2str(xi(1,2)-xi(1,1))])
% gtext({['(FWHM):',num2str(xi(1,2)-xi(1,1))],[num2str(t),'*Max']})
gtext({['(FWHM):'],[num2str(xi(1,2)-xi(1,1))]})
subplot(142);plot(fwhm_lateral,glat/(-slope*1e-3));%(1:(size(fwhm_axial,2)-4))
        title('Lateral','FontSize',12,'FontName','Times New Roman');axis tight
        xlabel('Length (mm)','FontSize',12,'FontName','Times New Roman')
        ylabel('Temperature (oC)','FontSize',12,'FontName','Times New Roman')


%% start of Axial FWHM calculation
t=1/2;%input('Enter the fraction for Axial FWHM: ');
 t2=num2str(t,6);
gax=final_max1(:,p);%gax=gax-min(gax);%1:(size(final_max1,2)-4)
%         h=figure;
```

```
        subplot(143);plot(fwhm_axial,gax);%(1:(size(fwhm_axial,2)-4))
        title(ylab,'FontSize',12,'FontName','Times New Roman');axis tight
        xlabel('Length (mm)','FontSize',12,'FontName','Times New Roman')
        ylabel('Voltage (V)','FontSize',12,'FontName','Times New Roman')


%% using interpolation
xi(2,1) = interp1(gax(1:(find(max(gax)==gax))),...
    fwhm_axial(1:(find(max(gax)==gax))),max(gax)*(t));
xi(2,2) = interp1(gax((find(max(gax)==gax)):length(gax)),...
    fwhm_axial((find(max(gax)==gax)):length(gax)),max(gax)*(t));
hold on
plot(xi(2,1),max(gax)*(t),'rx');
plot(xi(2,2),max(gax)*(t),'rx');
hold off
clc
disp([ylab,num2str(xi(2,2)-xi(2,1))])

% gtext({['FWHM:',num2str(xi(2,2)-xi(2,1))],[num2str(t),'*Max']})
gtext({['FWHM:'],[num2str(xi(2,2)-xi(2,1))]})
subplot(144);plot(fwhm_axial,gax/(-slope*1e-3));%(1:(size(fwhm_axial,2)-4))
        title(ylab,'FontSize',12,'FontName','Times New Roman');axis tight
        xlabel('Length (mm)','FontSize',12,'FontName','Times New Roman')
        ylabel('Temperature (oC)','FontSize',12,'FontName','Times New Roman')


%         % Create subplot
% subplot1 = subplot(1,4,1,'Parent',figure1);
% % Uncomment the following line to preserve the X-limits of the axes
% % xlim(subplot1,[0 1.016]);
% % Uncomment the following line to preserve the Y-limits of the axes
% % ylim(subplot1,[0.0340208531901851 0.289968537280505]);
% box(subplot1,'on');
% hold(subplot1,'all');
% Create textbox
annotation(h,'textbox',...
    [0.305123711340206 0.015 0.0392061855670103 0.0725],'String',{'(a)'},...
    'FontSize',16,...
    'FontName','Agency FB',...
    'FitBoxToText','off',...
    'LineStyle','none');

% Create textbox
annotation(h,'textbox',...
    [0.717907216494847 0.0125 0.0392061855670103 0.0725],'String',{'(b)'},...
    'FontSize',16,...
    'FontName','Agency FB',...
    'FitBoxToText','off',...
    'LineStyle','none');


%% using FWHM method
% [a2,b2]=max(gax);
```

```
%     Dmm1=max(find(gax(1:b2)<=(1*a2/2)))
%     D1=gax(max(find(gax(1:b2)<=(1*a2/2))))
%     Dmm2=min(find(gax((b2+1):length(gax))<(1*a2/2)))
%     D2=gax(max(find(gax(b2+1:length(gax))<(1*a2/2))))
%     a4=b2+min(find(gax(b2:length(gax))<(1*a2/2)));
%     hold on
%     plot(fwhm_axial(b2+Dmm2),max(gax)*(t),'rx');%max(glat)*(t)%fwhm_lateral(a2),
%     plot(fwhm_axial(Dmm1),max(gax)*(t),'rx');
%     hold off
%  disp(['Depth '])
%     disp([ylab,': ',num2str(fwhm_axial(Dmm2+b2)-fwhm_axial(Dmm1))])
%     gtext({['(FWHM): ',num2str(fwhm_axial(Dmm2+b2)-fwhm_axial(Dmm1))],...
%         [num2str(t),'*Max']})
%     subplot(122);plot(fwhm_axial,gax/(-slope*1e-3));%(1:(size(fwhm_axial,2)-4))
%         title(ylab);axis tight
%         xlabel('Length (mm)')
%         ylabel('Temperature (oC)')

%% FWHM of the smoothed singal
% glat=moving_average(glat,2,2);
% %     glat=glat-min(glat);
%
%        fwhm_lateral=linspace(0,(dataout.info.daq.xrange_mod...
%            *25.4),dataout.info.daq.xpoint);
%
%        t=1/2;%input('Enter the fraction for Lateral FWHM: ');
%        t1=num2str(t,6);
%
%        h=figure;plot(fwhm_lateral,glat,'r')
%        xlabel('Lateral across (mm)')
%        ylabel('Relative Temperature change (Normalized)')
%
% xi(1,1) = interp1(glat(1:(find(max(glat)==glat))),fwhm_lateral(1:(find(max(glat)==glat)))...
%     ,max(glat)*(t));
% xi(1,2) =
interp1(glat((find(max(glat)==glat)):length(glat)),fwhm_lateral((find(max(glat)==glat)):length(glat))..
.
%     ,max(glat)*(t));
% hold on
% plot(fwhm_lateral,glat1);axis tight
% plot(xi(1,1),max(glat)*(t),'kx');%max(glat)*(t)%fwhm_lateral(a2),
% plot(xi(1,2),max(glat)*(t),'kx');
% hold off
% clc
% disp(['Lateral Axial'])
% disp(['   ' t1]);
% disp(['Lateral: ',num2str(xi(1,2)-xi(1,1))])
%
% gtext(['Lateral',' (FWHM): ',num2str(xi(1,2)-xi(1,1))])
% w=['Smoothed';'Original'];
% legend(w,'Location','Best')
%
%
```

```
% xlabel('Lateral across (mm)')
%         ylabel('Voltage spike (temperature change, V)')

%%% to heat the hear dstribution using individual 1D line scans

% for i=1:15
%     a=['C:\Lab_folder\Jay\HIFU\hifu\heat_size\' num2str(i)];
%     load(a)
% %     if i<13
%     er(i,:)=final;
% %     else
% %         er(i,:)=2*final;
% %     end
% end
% % a=['C:\Lab_folder\Jay\HIFU\hifu\heat_size\' num2str(13)];
% % load(a)
% % er(13:22,:)=final;
% figure;imagesc(er)

%% for the video
%
% num_frames_per_second = 10;clc
% pathname3 = uigetdir('C:\Lab_folder\Jay\','Pick folder to store the movie file: ');
% moviename=input('Enter the name of the movie file to be created: ','s');
% moviename1=[pathname3 '\' moviename '_mod.avi'];
% moviename=[pathname3 '\' moviename '.avi'];
% aviobj = avifile ( moviename, 'fps', num_frames_per_second );


imgy=input((['Enter ROI for avg ',ylab,num2str(xi(2,2)-xi(2,1)),' :']))
imgx=input((['Enter ROI for avg lateral (mm)',num2str(xi(1,2)-xi(1,1)),' :']))
np2mx=round((imgx/((2*dataout.info.daq.xrange_mod*25.4)))*dataout.info.daq.xpoint)
np2my=round((imgy/((2*e.range_mod*25.4)))*e.point)

ay1=ay/2;
ax1=ax/4;

timemod=linspace(0,0.42,51);


curvop=[];curvop1=[];g=1;
cell2mat(fvid2(o));
[m,n]=size(ans);
for
i=1:500:((0.51*(1/daqtime(2)))/div)%(0.002*dataout.info.daq.acquirerate):(1*dataout.info.daq.acqu
irerate)
for j=1:e.point
   ag=cell2mat(fvid2(j));
   if rem(j,2)==1
      ag=fliplr(ag);
      ag=circshift(ag,[0 1]);
   end
   imgfinal(j,:)=ag(i,:);
```

```
end

%     timen(g)=timemod(g);
    timen(g)=time(i)*div;

% newimg=moving_average(imgfinal,4,1);
h=figure(50);%subplot(321);
set(h,'Position',[200 10 1500 1000])
subplot(221);axis image
imagesc(imgfinal(:,4:size(imgfinal,2))/(-slope*1e-3),[0 max(max(finalsig/(-slope*1e-3)))]);colorbar
% get(h,'Position')

set(gca,'FontSize',12)
 set(gca, 'XTick',cx*bx/ax);
    set(gca, 'XTicklabel',cx);
    set(gca, 'YTick',cy*by/ay);
    set(gca, 'YTicklabel',cy);%round(g1*10)/10
    xlabel('Lateral (mm)')
    ylabel([ylab,' (mm)'])
% title(['Abs Time: ',num2str(i/(dataout.info.daq.acquirerate))])
title(['(a) Abs Time:
',num2str(timen(g)),'Sec']);%i+4*dataout.info.daq.acquirerate%num2str(((i)/(dataout.info.daq.acqu
irerate)))
subplot(222);
set(h,'Position',[200 10 1500 1000])
imagesc(imgfinal((o-np2mx):(o+np2mx),(p-np2my):(p+np2my))/(-slope*1e-3),[0
max(max(finalsig/(-slope*1e-3)))]);colorbar
% get(h,'Position')
axis image

set(gca,'FontSize',12)
 set(gca, 'XTick',(0:(ax1/(imgx/(2*np2my +1))):(2*np2my +1)));
    set(gca, 'XTicklabel',(0:ax1:imgx));
    set(gca, 'YTick',(0:(ay1/(imgy/(2*np2mx +1))):(2*np2mx +1)));
    set(gca, 'YTicklabel',(0:ay1:imgy));%round(g1*10)/10
    xlabel(['Lateral (mm)'])
    ylabel([ylab,' (mm)'])
% title(['Abs Time: ',num2str(i/(dataout.info.daq.acquirerate))])
%title(['(b) Abs Time: ',num2str(((i)/(dataout.info.daq.acquirerate))),'Sec']);
title(['Only focused Area'])

% h4=figure(51);
figure(50);
subplot(223);
a12=cell2mat(fvid2(o));%a12=circshift(a12,[0 -1]);
if rem(o,2)==1
    curvop(g)=a12(i,e.point-p+1);
else
    curvop(g)=a12(i,p);
end

plot(timen(1:g),curvop(1:g)/(-slope*1e-3),'r');grid on
set(gca,'FontSize',12)
```

```
axis([0 0.5 0 max(max((finalsig-min(min(finalsig)))/(-slope*1e-3)))+max(max((finalsig-
min(min(finalsig)))/(-slope*1e-3)))*0.3])
xlabel('Time (Sec)')
ylabel('Temperature (oC)')
title('Temperature vs Time')
% set(h4,'Position',[680+560+10 558   560   420])

subplot(224);
% plot(timen(1:g),curvop(1:g),'r');grid on
% set(gca,'FontSize',12)
% axis([0 0.5 0 max(max((finalsig-min(min(finalsig)))))+max(max((finalsig-
min(min(finalsig)))))*0.3])
% xlabel('Time (Sec)')
% ylabel('Voltage (V)')
% title('Point Voltage (~Temperature) vs Time')
% Average of the whole image
a12=cell2mat(fvid2(o));
curvop1(g)=mean(mean(imgfinal((o-np2mx):(o+np2mx),(p-np2my):(p+np2my))));%(15:32,12:30)
plot(timen(1:g),curvop1(1:g),'r');grid on
set(gca,'FontSize',12)
axis([0 0.5 0 max(max((finalsig-min(min(finalsig)))/(-slope*1e-
3)))*(1/(np2mx*np2my*40))+max(max((finalsig-min(min(finalsig)))/(-slope*1e-3)))*0.05])
% axis([0 0.25 0 0.15])
xlabel('Time (Sec)')
ylabel('Temperature (oC)')
title('Average of the Temperature in Focused Area (b) vs Time')

%     d3=getframe(gcf);
%     aviobj = addframe ( aviobj, d3 );
g=g+1;
end
%   aviobj = close ( aviobj );
```

References

[1] T. L. Szabo, *Diagnostic ultrasound imaging : inside out*. Amsterdam ; Boston: Elsevier Academic Press, 2004.

[2] T. J. Dubinsky, C. Cuevas, M. K. Dighe, O. Kolokythas, and J. H. Hwang, "High-intensity focused ultrasound: current potential and oncologic applications," *AJR Am J Roentgenol,* vol. 190, pp. 191-9, Jan 2008.

[3] T. Charlebois, and Pelton, R., "Quantitative 2D and 3D Schlieren Imaging for acoustic power and intensity measurements," *Medical Electronics,* pp. 789-792.

[4] K. Raum and W. D. O'Brien, *Pulse-echo field distribution measurement technique for high-frequency ultrasound sources*, 1997.

[5] B. Huang and K. K. Shung, "Characterization of very high frequency transducers with wire target and hydrophone," in *AMUM 2004: Advanced Metrology for Ultrasound in Medicine 2004*. vol. 1, A. Shaw, Ed., ed, 2004, pp. 161-166.

[6] ONDA, "OptiSon® Ultrasound Beam Analyzer," ed.

[7] ONDA, "Acoustic Intensity Measurement System," ed.

[8] K. F. Graff, "POWER ULTRASONICS - HISTORICAL ASPECTS," *Ieee Transactions on Sonics and Ultrasonics,* vol. 25, pp. 231-231, 1978.

[9] W. R. Hendee, "Cross sectional medical imaging: a history," *Radiographics,* vol. 9, pp. 1155-80, Nov 1989.

[10] American Association of Physicists in Medicine. Summer School (1980 : University of Wisconsin--La Crosse), G. D. Fullerton, J. A. Zagzebski, and American Association of Physicists in Medicine., *Medical physics of CT and ultrasound : tissue imaging and characterization*. New York, N.Y.: Published for the American Association of Physicists in Medicine by the American Institute of Physics, 1980.

[11] K. T. Dussik, "The ultrasonic field as a medical tool," *Am J Phys Med,* vol. 33, pp. 5-20, Feb 1954.

[12] I. Donald, J. Macvicar, and T. G. Brown, "Investigation of abdominal masses by pulsed ultrasound," *Lancet,* vol. 1, pp. 1188-95, Jun 7 1958.

[13] W. R. Hendee and E. R. Ritenour, *Medical imaging physics*, 4th ed. New York: Wiley-Liss, 2002.

[14] C. R. Hill, J. C. Bamber, and G. t. Haar, *Physical principles of medical ultrasonics*, 2nd ed. Chichester ; Hoboken, N.J.: John Wiley & Sons, 2004.

[15] W. N. McDicken, *Diagnostic ultrasonics : principles and use of instruments*. New York: Wiley, 1976.

[16] P. N. T. Wells, *Biomedical ultrasonics*. London ; New York: Academic Press, 1977.

[17] Olympus, "Panametrics transducers," ed: Envirocoustics.

[18] G. Kossoff, "Improved techniques in ultrasonic cross sectional echography," *Ultrasonics,* vol. 10, pp. 221-7, Sep 1972.

[19] J. C. Spanner, J. W. McElroy, and American Society for Testing and Materials. Committee E-7 on Nondestructive Testing., *Monitoring structural integrity by acoustic emission : a symposium presented at Ft. Lauderdale, Fla., 17-18 Jan. 1974*. Philadelphia: The Society, 1975.

[20] ONDA, "HNP Hydrophone," ed.

[21] Olympus_Panametrics_NDT, "Ultrasonic immersion transducers," ed.

[22] D. Chen, T. Fan, D. Zhang, and J. Wu, "A feasibility study of temperature rise measurement in a tissue phantom as an alternative way for characterization of the therapeutic high intensity focused ultrasonic field," *Ultrasonics,* vol. 49, pp. 733-42, Dec 2009.

[23] K. l. Lee and M. J. Choi, "Prediction and Measurement of the Size of Thermal Lesion Induced by High Intensity Focused Ultrasound in a Tissue-Mimicking Phantom," *Japanese Journal of Applied Physics,* vol. 48, p. 027003, 2009.

[24] G. T. Haar and C. Coussios, "High intensity focused ultrasound: physical principles and devices," *Int J Hyperthermia,* vol. 23, pp. 89-104, Mar 2007.

[25] G. R. Harris, "FDA regulation of clinical high intensity focused ultrasound (HIFU) devices," *Conf Proc IEEE Eng Med Biol Soc,* vol. 2009, pp. 145-8, 2009.

[26] H. Morris, I. Rivens, A. Shaw, and G. T. Haar, "Investigation of the viscous heating artefact arising from the use of thermocouples in a focused ultrasound field," *Phys Med Biol,* vol. 53, pp. 4759-76, Sep 7 2008.

[27] L. Faqi, C. Jie, Z. Deping, W. Qi, and Z. Tao, "Measuring Temperature Rise in Phantom to Determine High Power High-Intensity Focused Ultrasound Sound Field," in *Bioinformatics and Biomedical Engineering (iCBBE), 2010 4th International Conference on*, 2010, pp. 1-4.

[28] K. Zell, J. I. Sperl, M. W. Vogel, R. Niessner, and C. Haisch, "Acoustical properties of selected tissue phantom materials for ultrasound imaging," *Phys Med Biol,* vol. 52, pp. N475-84, Oct 21 2007.

[29] J. McLaughlan, I. Rivens, T. Leighton, and G. Ter Haar, "A study of bubble activity generated in ex vivo tissue by high intensity focused ultrasound," *Ultrasound Med Biol,* vol. 36, pp. 1327-44, Aug 2010.

[30] C. L. Gong and D. P. Hart, "Ultrasound induced cavitation and sonochemical yields," *Journal of the Acoustical Society of America,* vol. 104, pp. 2675-2682, Nov 1998.

[31] S. D. Nandlall, E. Jackson, and C. C. Coussios, "Real-time passive acoustic monitoring of HIFU-induced tissue damage," *Ultrasound Med Biol,* vol. 37, pp. 922-34, Jun 2011.

[32] R. L. King, Y. Liu, S. Maruvada, B. A. Herman, K. A. Wear, and G. R. Harris, "Development and characterization of a tissue-mimicking material for high-intensity focused ultrasound," *IEEE Trans Ultrason Ferroelectr Freq Control,* vol. 58, pp. 1397-405, Jul 2011.

BIOGRAPHICAL INFORMATION

Jayanth Kandukuri is pursuing his Master of Science in Biomedical Engineering at University of Texas at Arlington. He obtained his Bachelors of Engineering in Biomedical Engineering from Padmasri Dr. B.V Raju Institute of Technology, Medak, affiliated to Jawaharlal Nehru Technological University, Hyderabad, India after which obtained Master of Engineering in Electrical Engineering from University of Queensland, Australia. He has maintained a good overall academic record in his time in University of Texas at Arlington as a graduate student. His area of interest is Optical Imaging, Medical Imaging and Biomedical devices. After graduating from University of Texas at Arlington he plans to continue his learning process by pursuing his Doctoral studies here in the United States. Apart from academics his other interests are table tennis, fitness, experiencing different cuisines and travelling to different places and understanding and appreciating their culture.