INVESTIGATION OF ACOUSTIC CROSSTALK EFFECTS IN CMUT ARRAYS

A Thesis
Presented to
The Academic Faculty

by

Michael Hochman

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Mechanical Engineering

Georgia Institute of Technology
DECEMBER 2011

INVESTIGATION OF ACOUSTIC CROSSTALK EFFECTS IN CMUT ARRAYS

Approved by:

Dr. Levent Degertekin, Advisor
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Karim Sabra
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Suresh Sitaraman
School of Mechanical Engineering
Georgia Institute of Technology

Date Approved: August 15th, 2011

ACKNOWLEDGEMENTS

I would first like to express my appreciation to Dr. F. Levent Degertekin for his invaluable guidance during my research. His technical expertise and unyielding faith in my abilities provided me with the knowledge and confidence to successfully complete this thesis. I would also like to thank Dr. Karim Sabra and Dr. Suresh Sitaraman for offering their time and experience to serve as my thesis committee members.

I would also like to thank my fellow colleagues in the MiST research group. In particular, I would like to acknowledge Jaime Zahorian for mentoring me in cleanroom fabrication and helping me with all aspects of my research, Sarp Satir for his assistance with experimental setups and advice concerning modeling techniques, Toby Xu for helping with cleanroom fabrication and experiments, and Gokce Gurun for designing the ICs used in experiments and providing technical feedback on their operation. I have had a fun and memorable experience being a part of the MiST research group. I consider myself close friends with all of them and will sincerely miss them when I am gone.

Gary Spinner and the MiRC cleanroom staff also deserve my gratitude for providing a safe cleanroom environment and ensuring that all fabrication equipment was well-maintained. My experimental devices could not have been fabricated without them. I would also like to thank the National Institute of Health (NIH) for providing financial support of my research.

Finally, I would like to thank my fiancé, Nina. I would not be where I am today without you. Thank you for your unwavering patience and support even in my most stressful times. You make every day fun and exciting and I am so lucky to have you in my life. I dedicate this thesis to you.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

SUMMARY

Capacitive Micromachined Ultrasonic Transducers (CMUTs) have demonstrated significant potential to advance the state of medical ultrasound imaging beyond the capabilities of the currently employed piezoelectric technology. Because they rely on well-established micro-fabrication techniques, they can achieve complex geometries, densely populated arrays, and tight integration with electronics, all of which are required for advanced intravascular ultrasound (IVUS) applications such as high-frequency or forward-looking catheters. Moreover, they also offer higher bandwidth than their piezoelectric counterparts. Before CMUTs can be effectively used, they must be fully characterized and optimized through experimentation and modeling. Unfortunately, immersed transducer arrays are inherently difficult to simulate due to a phenomenon known as acoustic crosstalk, which refers to the fact that every membrane in an array affects the dynamic behavior of every other membrane in an array as their respective pressure fields interact with one another. In essence, it implies that modeling a single CMUT membrane is not sufficient; the entire array must be modeled for complete accuracy.

Finite element models (FEMs) are the most accurate technique for simulating CMUT behavior, but they can become extremely large considering that most CMUT arrays contain hundreds of membranes. This thesis focuses on the development and application of a more efficient model for transducer arrays first introduced by Meynier et al. [1], which provides accuracy comparable to FEM, but with greatly decreased computation time. It models the stiffness of each membrane using a finite difference approximation of thin plate equations. This stiffness is incorporated into a force balance which accounts for effects from the electrostatic actuation, pressure forces from the fluid environment, mass and damping from the membrane, etc. For fluid coupling effects, a Boundary Element Matrix (BEM) is employed that is based on the Green's function for a

baffled point source in a semi-infinite fluid. The BEM utilizes the nodal mesh created for the finite difference method, and relates the dynamic displacement of each node to the pressure at every node in the array. Use of the thin plate equations and the BEM implies that the entire CMUT array can be reduced to a 2D nodal mesh, allowing for a drastic improvement in computation time compared with FEM.

After the model was developed, it was then validated through comparison with FEM. From these tests, it demonstrated a capability to accurately predict collapse voltage, center frequency, bandwidth, and pressure magnitudes to within 5% difference of FEM simulations. Further validation with experimental results revealed a close correlation with predicted impedance/admittance plots, radiation patterns, frequency responses, and noise current spectrums. More specifically, it accurately predicted how acoustic crosstalk would create sharp peaks and notches in the frequency responses, and enhance side lobes and nulls in the angular radiation pattern.

Preliminary design studies with the model were also performed. They revealed that membranes with larger lateral dimensions effectively increased the bandwidth of isolated membranes. They also demonstrated potential for various crosstalk reduction techniques in array design such as disrupting array periodicity, optimizing inter-membrane pitch, and adjusting the number of membranes per element. It is expected that the model developed in this thesis will serve as a useful tool for future iterations of CMUT array optimizations.

CHAPTER 1

INTRODUCTION AND BACKGROUND

The research described in this paper revolves around the development of a new model for simulating the behavior of CMUT arrays for the particular application of intravascular ultrasound. However, before describing this new model, it is important to provide some background detailing the state of intravascular ultrasound technology, and its associated limitations. This will be the basis of Chapter 1, followed by a thorough explanation of CMUTs in Chapter 2, highlighting their potential for advancing IVUS technology beyond what is currently available commercially. These first two chapters will serve to provide a foundation emphasizing the need for CMUT development, which can only be attained with the aid of accurate, reliable models. These models, including the one developed for this paper, will allow for a greater understanding of the physical principles at play, as well as providing an invaluable tool for optimization and design.

1.1. Ultrasonic Imaging

Ultrasonic imaging is a technique used to map out a surrounding environment by transmitting sound waves and recording the signals that are reflected from various interfaces in the vicinity. These reflected signals contain all the necessary information to construct a map of nearby entities, along with specific details regarding their relative mechanical impedance properties. The time delay between transmitted and reflected signals, assuming the speed of sound in the environment is known, can be utilized to estimate distances. Furthermore, strong reflectors (e.g. bone) will exhibit much larger receive amplitudes than weak reflectors (e.g. tissue). An additional, important property of ultrasound is that it has the capability to penetrate the surface of weak reflectors. This

allows ultrasound to extract information about sub-surface features, which are detected from reflections occurring at interfaces within a material.

1.2. Intracardiac and Intravascular Ultrasound (IVUS)

Ultrasonic imaging is used in everything from fluid sensing to nondestructive testing (NDT) for defects in materials. However, this paper will focus on its particular application in the medical field. More specifically, the focus will be constrained to modeling and designing devices ideal for intracardiac echocardiography (ICE) and intravascular ultrasound (IVUS). Coronary heart disease (CHT) was the leading cause of mortality in the United States in 2009, resulting in the deaths of nearly 600,000 people [2]. To diagnose and treat patients suffering from this disease, IVUS and ICE both utilize minimally-invasive catheters with built-in imaging arrays that can image the inside of blood vessels and heart tissues. Differences in sound reflection coefficients between vessel walls, blood, lipids (vessel occlusions), and artificial structures such as stints or balloons allow for detailed and useful images to be created. These catheters allow physicians to look for evidence of plaque build-up or vessel wall weakening (aneurysm) in coronary arteries. If an artery is fully occluded, they also can also be used to revascularize the blocked vessel by poking through it with a guide wire that runs through the center of the catheter. Stents and balloons can also be positioned and imaged using ultrasonic catheters.

Figure 1 displays an image taken from the interior of a blood vessel, using an ultrasonic catheter. This particular vessel is healthy, with minimal lipid build-up on the inner lining (intimal) of the vessel. The red circle in the photo on the right denotes the location of the catheter. Figure 2 shows an image of a partially occluded blood vessel. From this ultrasound image, the shrinkage of the lumen due to intimal plaque accumulation is clearly visible. Even more noteworthy, the ultrasonic catheter even allows for calcium deposits to be distinctly identified within the plaque layer due to the

reflection coefficient differences between the two materials. Since calcium is a strong reflector, it prevents further penetration of the sound waves and is thus responsible for the acoustic shadowing. It is clear from both of these images that ultrasound allows for detailed images that can provide crucial information to physicians regarding the status of a patient suffering from CHT.

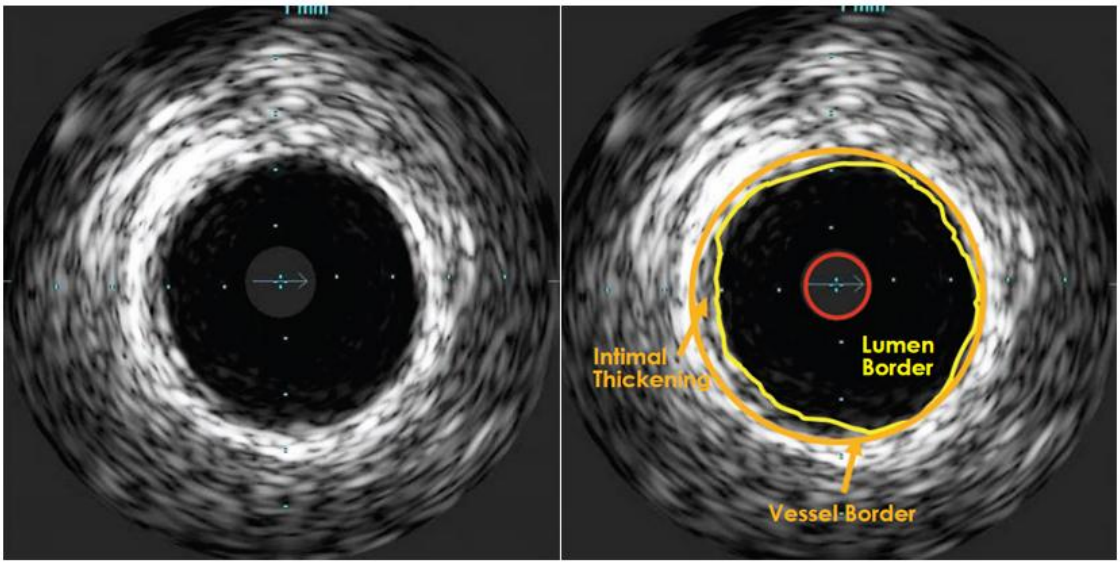


Figure 1. Image of a healthy blood vessel, taken with an Eagle Eye® ultrasonic catheter from Volcano Corporation [3]

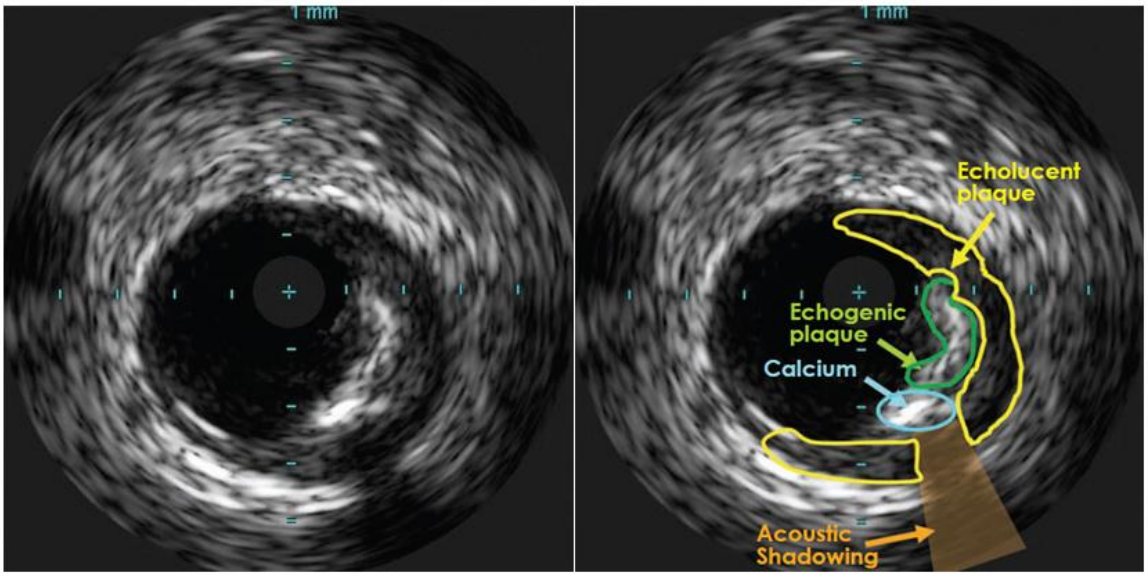


Figure 2. Image of a partially occluded blood vessel, taken with an Eagle Eye® ultrasonic catheter from Volcano Corporation [3]

Resolution of ultrasound images is dependent on the operating frequency. Smaller wavelengths allow for better lateral resolution. Unfortunately, higher frequencies experience much more acoustic attenuation, which in turn limits axial penetration depth. Thus, a 10 MHz catheter may have a penetration depth of 1 cm or higher, with a lateral resolution of approximately 150 um. On the other hand, laboratory tests at frequencies up to 100 MHz have demonstrated that although resolution improves to nearly 15um, penetration depths are limited to only a few millimeters [4].

1.3. Commercially Available Technology for Catheter-Based IVUS Imaging

Currently, all commercially available IVUS and ICE catheters are based on piezoelectric transducer arrays. These imaging arrays employ specialized materials that have the unique characteristic of experiencing strain in response to an applied electric potential, and vice versa. Thus, an applied strain (e.g. incoming pressure wave) can be detected by the material as an oscillation in the potential across it. Alternatively, by applying a sinusoidal voltage, these materials can be forced to oscillate at ultrasonic

frequencies, generating large pressure amplitudes into the surrounding environment. In this manner, piezoelectric array elements can act as both transmitters and receivers in IVUS catheters.

There are essentially two types of piezoelectric imaging catheters. The first type relies on array phasing techniques (Figure 3). The device typically consists of many side-looking elements which are actuated one at a time. After each firing of a single transmit element, all remaining elements act as receivers and record the signal response. This process continues until all elements have been operated as transmitters. Then, since the relative locations of every element are known, phased array imaging algorithms can be used to determine locations of objects in the environment based on the time delay between transmitted and received signals. If the elements are placed along the entire circumference of the catheter, a 360 degree view can be obtained. Otherwise, the view will be limited and the catheter will have to be rotated manually to see all sides of the vessel. Phased array catheters have the advantage of requiring no moving parts for imaging (an advantage over rotating array catheters, described next). However, phasing array elements requires that the array be a certain distance away from the region to be imaged. The array must then be "steered" to bring the cross-section of interest into focus. This can be difficult as it sometimes requires precise positioning of the catheter tip [5].

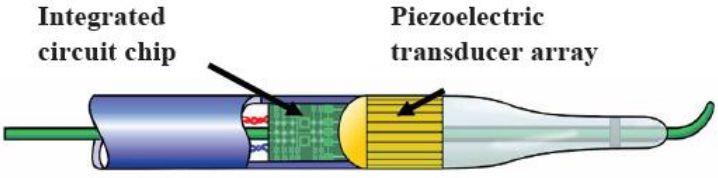


Figure 3. Illustration of a phased array ultrasonic imaging catheter [6]

The second type of piezoelectric catheter utilizes a single, large element (as opposed to an array) which acts as both a transmitter and receiver. This element is either

used directly as a side-facing imager (Figure 4), or is pointed forward at an angled mirror which then directs the acoustic wave radially outward. At locations very near the face of an element, there is a so-called "dead zone" which corresponds to the region in which received signals are corrupted by the large transmit signal that saturates the amplifier [6]. The mirror design reduces the size of the dead zone adjacent to the catheter by increasing the path length of the acoustic wave before it leaves the catheter's outer wall. To achieve a 360 degree image of the vessel, this side-facing element or mirror is rotated mechanically at a sufficiently high speed. This type of catheter in general has a smaller dead zone than phased arrays since, as illustrated in Figure 3, the phased array elements are placed at the outer circumference of the catheter [6]. Rotating element catheters thus are slightly easier to position because they do not need to be placed as far away from the region being imaged. However, rotating catheters often suffer from Non-Uniform Rotation Distortion (NURD) which, as the name implies, occurs from irregular rotation of the catheter and causes image distortion. As shown in Figure 5, this can degrade the accuracy of the image, and severely limit the user's ability to make any type of conclusive diagnoses. This type of distortion does not occur with phased arrays. It should also be noted that rotating arrays must be located at the center axis of the catheter to avoid image distortion. Thus, incorporation of guide wires requires special consideration and they are often fed through the outer section of the catheter, creating a small blind spot in the field of view. Once again, phased arrays would not have this blind spot as the array elements are all located along the outer circumference of the catheter, allowing the guide wire to be positioned along the center axis.

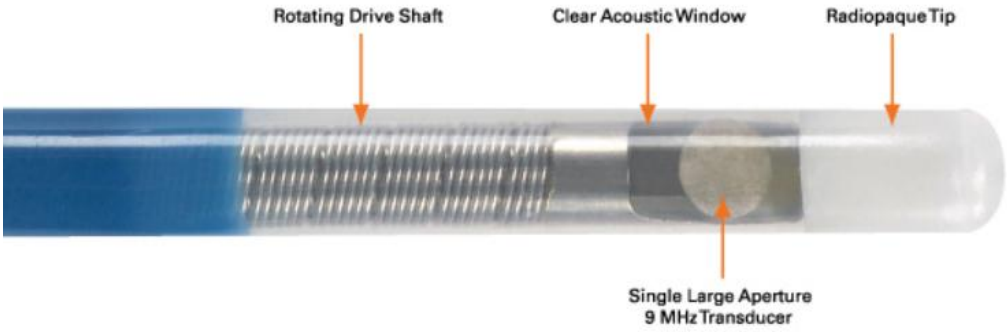


Figure 4. Boston Scientific rotating array ultrasonic imaging catheter for ICE [5]

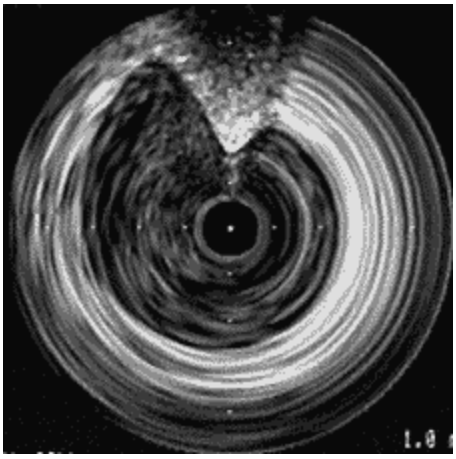


Figure 5. An example of NURD image distortion from a rotating element IVUS catheter [7]

Piezoelectric catheters are in widespread use in the medical field. However, they possess several inherent limitations. Primarily, piezoelectric materials have a high impedance (~30 MRayls) relative to water (~1.5 MRayls), and thus matching layers are required to achieve better bandwidth, at the expense of efficiency [8, 9]. These matching layers typically utilize quarter wavelength thicknesses, making layers for high frequency devices much more difficult to fabricate. Finding materials with ideal mechanical and acoustic properties is also a common problem, and research is still ongoing in this area. Furthermore, piezoelectric ceramics are most commonly diced out of bulk material to minimize defects. However, these materials are brittle and difficult to dice, especially

when small pitches or complex geometries are desired. Dicing blade limitations constrain array geometries to mostly straight-line cuts and features on the order of 100 um or larger [10], hindering the development of high frequency IVUS.

1.4. Forward-Looking IVUS and ICE

Due to the inherent challenges associated with piezoelectric transducer design, all ultrasound catheters currently on the market are side-looking. That is, no IVUS or ICE catheter exists commercially with the ability to see in the forward direction. It is easy to imagine how much difficulty this adds to a procedure such as revascularization of an occluded artery, when the operator must maneuver a guide wire through the blockage. In most cases, the only view that a surgeon has of the blockage is from an external angiogram [11].

The main difficulties associated with forward-looking intravascular ultrasound (FL-IVUS) catheters stem from dimensional constraints. Specifically, catheters need to be as small as 1 mm in diameter in order to navigate the narrowest of blood vessels. An ultrasound transducer would need to fit on the very tip of this catheter in order to see in the forward direction, and thus could be no larger than the catheter diameter. To further complicate the matter, a hole in the center of the catheter should ideally be left open to allow for a guide wire to pass through. In other words, an FL-IVUS transducer is constrained to a donut-shaped ring with a diameter on the order of a few millimeters or less. Finally, the most preferable FL-IVUS catheter would utilize a 2D array to allow for a volumetric image reconstruction of the occlusion in front of the array. The limited available area implies that the array element size must be very small in order to fit a sufficient number of elements for image reconstruction. These small, densely populated arrays are very difficult and tedious to fabricate with modern piezoelectric fabrication techniques. Also, small elements imply a high sensitivity to parasitic capacitances. Thus, the receive elements should be located very close to the amplifier electronics for

optimal Signal-to-Noise Ratio (SNR); ideally they would be fabricated directly on the amplifier circuitry chip to minimize connection lines. Piezoelectric transducer fabrication prevents such on-chip fabrication, so SNR capabilities are limited.

Research for using piezoelectric transducers for FL- IVUS is ongoing, but results thus far have been limited. Wang et al. created a forward looking array with 64 elements and a diameter of 1.2 mm [12]. However, due to fabrication complexity, the devices were not fully functional and were ill-suited for batch fabrication. Light et al. also fabricated piezoelectric, FL-IVUS arrays with improved reliability [13]. However, they have only demonstrated functionality on devices no smaller than 11 French (3.7 mm) and operating frequencies no higher than 5 MHz. These large dimensions and low frequencies are not suitable for many IVUS applications.

An alternative FL-IVUS approach has been proposed by Volcano Corporation that would utilize a piezoelectric rotating element, angled at 45 degrees, and would give the user a conical field of view in front of the catheter [14], as shown in Figure 6. However, this catheter is still in the development phase and has yet to be implemented commercially. Furthermore, this device will still possess the previously mentioned, inherent drawbacks of piezoelectric technology (e.g. fabrication complexity, limited bandwidth, etc.) and rotating element catheters (e.g. NURD). This is not to mention the fact that in bypassing the 2D array approach, the catheter will not provide a volumetric image but rather a conical shaped cross-section.

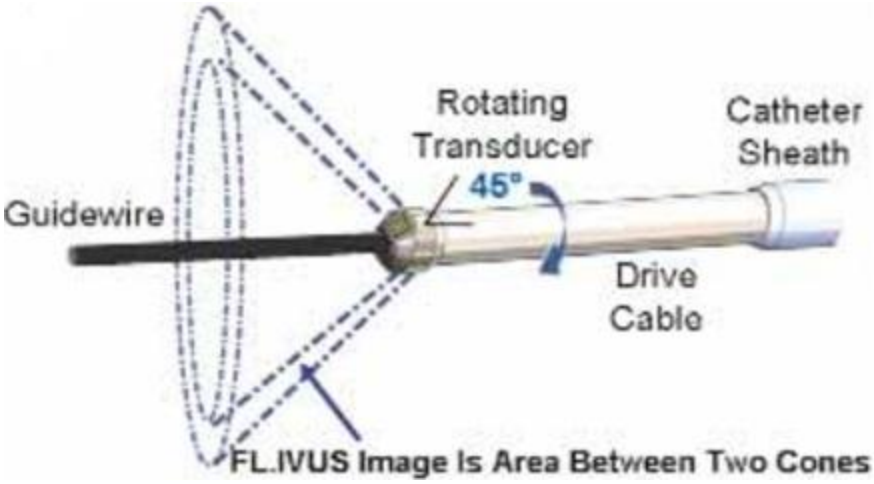


Figure 6. Illustration of the proposed Volcano FL-IVUS catheter [14]

CHAPTER 2

CMUT TECHNOLOGY

As discussed in the previous chapter, piezoelectric IVUS catheters have a number of limitations. Capacitive Micromachined Ultrasonic Transducers (CMUTs) have potential to greatly improve upon these designs. The following chapter will describe the general operating principles for CMUTs, emphasizing their significant potential for advancing current IVUS technology. It will also detail different CMUT modeling techniques and conclude by highlighting the need for an efficient model of CMUT array operation.

2.1. CMUT Basics

CMUTs are flexible membranes that are capacitively actuated at ultrasonic frequencies. Each membrane, as shown in Figure 7, contains an embedded electrode, and each membrane is suspended over a second, fixed electrode. The two electrodes are separated from each other by a vacuum gap and two layers of dielectric isolation, forming a capacitor. When a potential is placed across the terminals of the capacitor, the membrane will deflect downward. When a sinusoidal voltage is applied, the membrane will oscillate. Alternatively, if an electrical bias is applied to the electrodes, an incoming pressure wave would cause the membrane to oscillate and could be detected as a current. Thus, similarly to piezoelectric transducers, CMUTs can be used for both transmit and receive functions in ultrasonic applications. More specific details regarding CMUT operating principles will be given in Chapter 4.

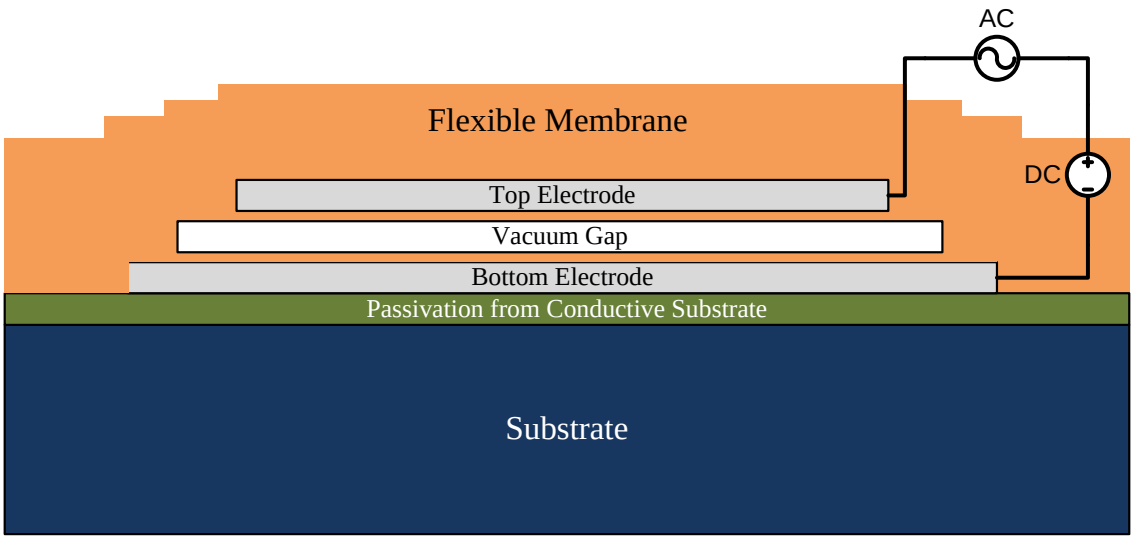


Figure 7. Illustration of a CMUT (dimensions are not to scale for better visualization)

2.2. CMUT Arrays in Intravascular and Intracardiac imaging applications

2.2.1. CMUTs vs. Piezoelectric Technology

CMUTs have a few distinct advantages over piezoelectric ceramics. Primarily, the CMUT membranes have much lower impedances as compared to that of water. This eliminates the need for quarter-wavelength matching layers, simplifying fabrication and improving bandwidth. Furthermore, CMUTs utilize well-established microfabrication processes which have been developed by the microprocessor industry. These manufacturing techniques can create extremely small vacuum gaps between the CMUT electrodes, allowing for low voltage operation. Micromachining also allows for batch fabrication of many devices at once, decreasing production costs. Furthermore, micromachining can create extremely small and complex geometries with minimal difficulty, a stark contrast from piezoelectric arrays which must be manually diced out. This particular characteristic makes CMUTs an ideal choice for FL-IVUS applications, in which small, dense arrays are needed. Additionally, since these fabrication techniques are derived directly from the microprocessor industry, tight integration of CMUTs and

control circuitry is possible. Once again, for FL-IVUS, this is a significant advantage because the required small element sizes demand minimal parasitic capacitances.

It should be noted that although CMUTs exhibit superior performance in terms of bandwidth and fabrication capabilities, the SNR of a standard CMUT array has been shown to be worse than their piezoelectric counterparts by up to 10 dB during in vivo comparison tests [15]. However, in recent years, a number of alternative designs and operating modalities have been proposed and validated that can bring pressure outputs of CMUTs to more competitive levels [16, 17].

2.2.2. Forward Looking Intravascular and Intracardiac Dual Ring Arrays

For forward-looking ultrasound applications, most CMUT designs involve a single annulus of many elements at the outer diameter of the catheter, as in [18]. The annulus design leaves room in the center of the catheter for a guide wire. Each element in the array acts as both a transmitter (Tx) and receiver (Rx). One element is pulsed at a time, and then the echoes are recorded by all elements. This process is repeated until all elements have acted as a transmitter. Delay and sum algorithms are then used to process these signals into a 3D volumetric image of the surrounding environment.

The Degertekin group at Georgia Institute of Technology has a patented dual ring design which employs separate rings for transmit and receive, as shown in Figure 8. This allows for separate optimization of Tx/Rx biases. This also eliminates the need for diode protection circuitry needed to prevent saturation of the Rx amplifiers after the large transmit pulse. Each element consists of 4 equally sized membranes that are actuated simultaneously. These devices have been successfully fabricated in sizes ranging from 0.8 mm to 2.0 mm in diameter, containing 48 - 64 elements in each ring. Imaging is carried out in the same way as described above, except the inner ring elements only act as receivers, and the outer ring elements only act as transmitters. This particular design also

minimizes parasitic capacitances by employing a CMUT on CMOS fabrication technique, described more thoroughly in Section 4.3.

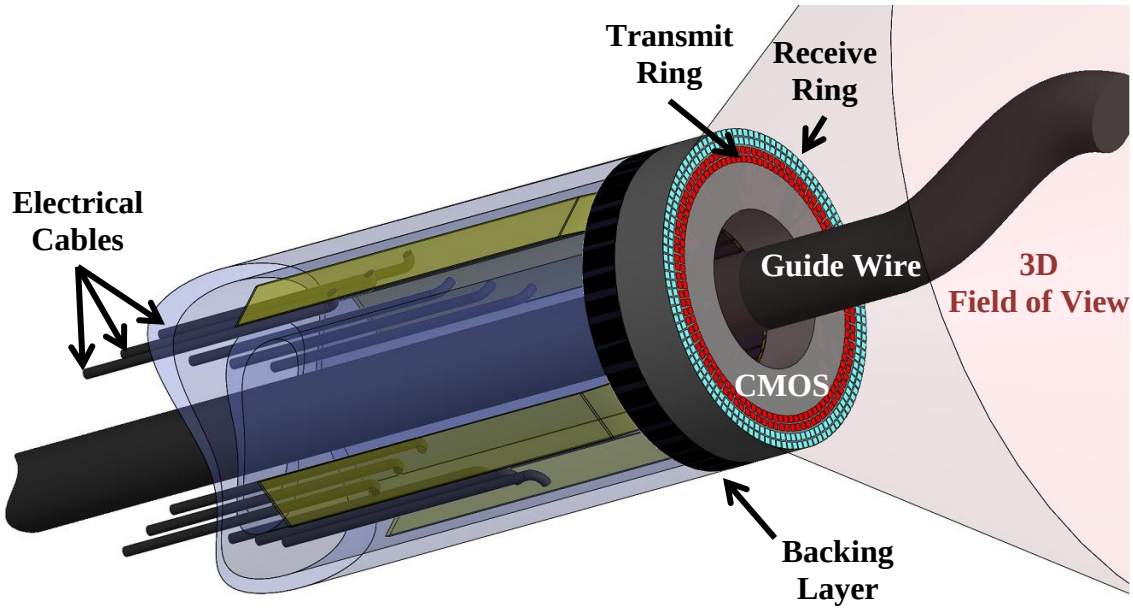


Figure 8. Model of FL-IVUS catheter utilizing a CMUT Dual-Ring Array

2.3. CMUT Operating Principles

2.3.1. Static Operation

In order to better understand CMUT operation, it is useful to implement simplified analytical models. The most basic representation of the system is that of a spring-mass-dashpot parallel plate actuator, as illustrated in Figure 9. In this analysis, the deformed shape of the membrane as it deflects is neglected, and instead the plates are assumed to remain perfectly parallel to each other with increased bias. The parallel plate model also neglects fringing effects in the electric field. These assumptions will result in some inaccuracies, but they are acceptable for the purposes of understanding the physical behavior of a CMUT.

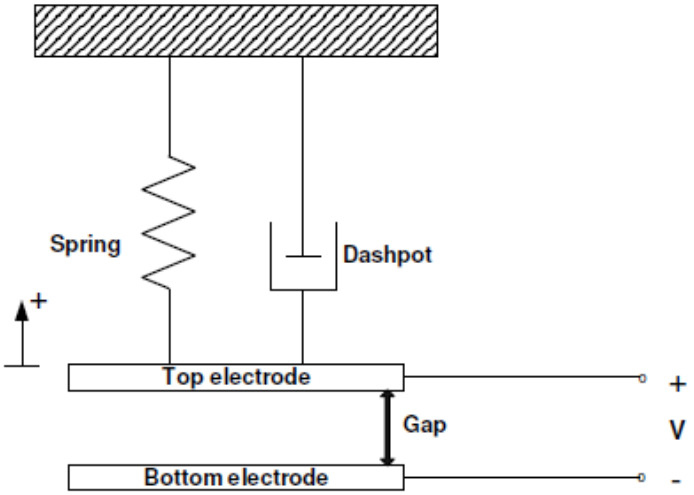


Figure 9. Illustration of a spring-mass-dashpot parallel plate actuator model [17]

Considering only the static case, we can neglect the dashpot and mass terms as they will only affect dynamic operation. Thus, the system can be reduced to a simple force balance equation containing two terms:

$$F_{es} = F_s \tag{1}$$

F_{es} is the electrostatic force given by

$$F_{es} = -\frac{\varepsilon_0 A V_0^2}{2\left(h_{gap} + z\right)^2} \tag{2}$$

Where ε_0 is the permittivity of free space, A is the electrode area, V_0 is the DC bias applied to the electrodes, h_{gap} is the initial height of the vacuum gap, and z is the displacement of the membrane from its initial position. The negative sign indicates that an applied bias generates a downward force. The mechanical spring force is upwards and positive, and can be approximated by Hooke's Law,

$$F_s = kz \tag{3}$$

Where k is the effective spring constant of the membrane.

It is evident that as bias increases, the electrostatic force increases exponentially while the mechanical restoring force increases linearly. Furthermore, for relatively low

voltages, the system will have two equilibrium points at which the electrostatic force equals the mechanical force, as shown in Figure 10 as point A and B. This figure displays the two force curves on a single plot as a function of the normalized gap between the electrodes (a normalized gap of 1 implies that the electrodes are in contact with each other). Although there are 2 equilibrium points, only A is inherently stable. At B any small displacement in either direction would cause one force to become dominant and the membrane would be shifted away from equilibrium. At A, however, any displacement in either direction would result in a restoring force pushing the system back towards equilibrium. Thus, at low voltages, there is only one displacement which will result in a stable equilibrium.

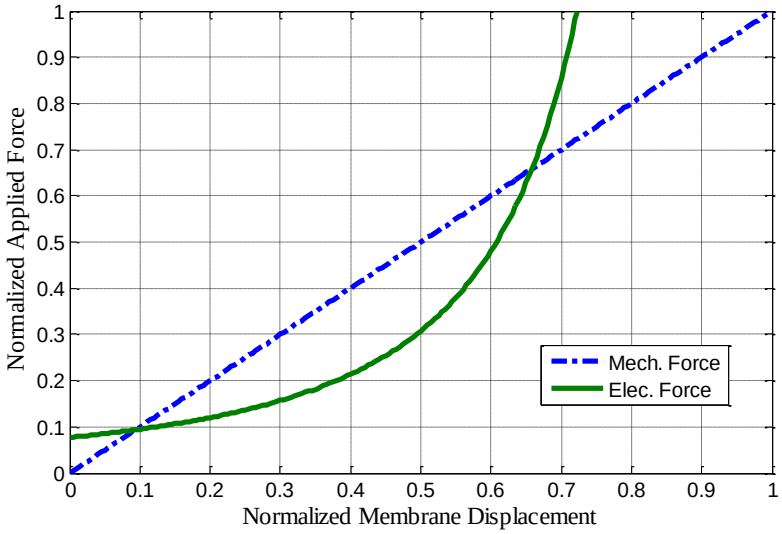


Figure 10. Plot of forces acting on a parallel plate capacitor as a function of normalized membrane displacement for a relatively low operating voltage

As voltage is increased, the electrostatic curve will be shifted upward while the mechanical force will remain unchanged. This implies that above a certain voltage, there will be no point at which mechanical stiffness is greater than or equal to the electrostatic force. This cut-off is referred to as the collapse voltage of a CMUT. The left plot of Figure 11 demonstrates how the number of equilibrium points at collapse is reduced to

one, and the right plot shows the absence of any equilibrium point beyond collapse. Note that the displacement at collapse for parallel plates is exactly 1/3 of the original gap:

$$z_{collapse} = \frac{h_{gap}}{3} \tag{4}$$

As the name implies, when this threshold voltage is exceeded, the CMUT will "collapse" and the two electrodes will come in contact with each other. Combining (2) and (4) yields an approximation for the collapse voltage of a parallel plate actuator:

$$V_{collapse} = \sqrt{\frac{8kh_{gap}^3}{27\varepsilon_0 A}} \tag{5}$$

The collapse voltage is a well-known characteristic of CMUTs, and effectively limits the displacement range of the membrane. This implies that for standard CMUT operation, the maximum transmitted pressure output is limited by the gap size. The receive capabilities are also limited because the sensitivity of the transducer is directly proportional to the gap between the electrodes. Clearly this introduces a design trade-off; a larger initial gap would allow for higher pressure output, but would decrease sensitivity in receive.

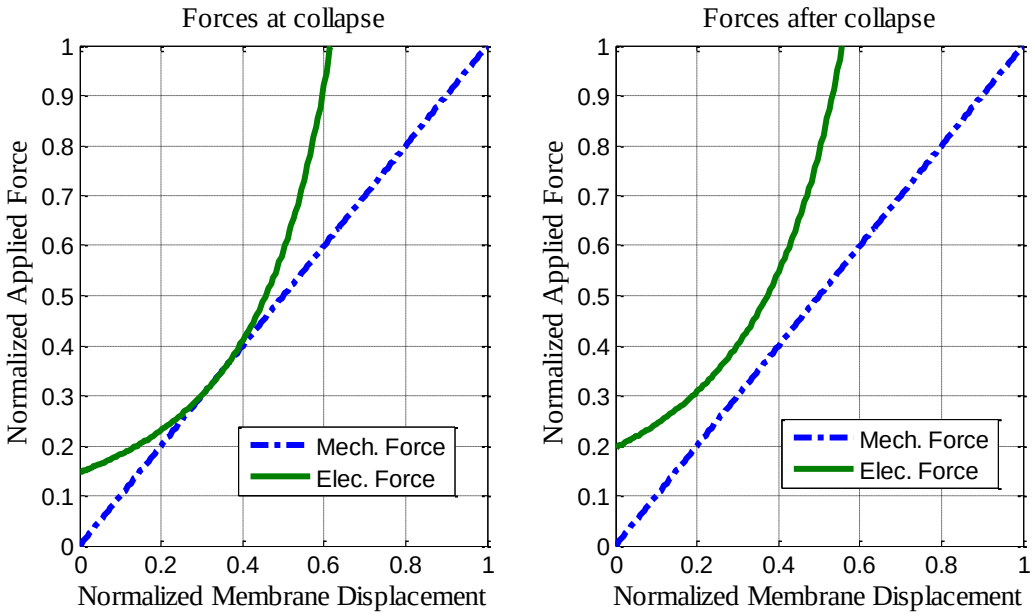


Figure 11. Plot of forces acting on a parallel plate capacitor as a function of normalized membrane displacement at collapse (left) and beyond collapse (right)

As mentioned earlier, CMUTs typically are not just separated by a vacuum gap, but also contain some dielectric isolation layers as well. These are necessary to prevent the electrodes from shorting out if collapse occurs, and are also needed for proper release of the membrane during fabrication, which will be detailed later. Thus, to be more accurate, the electrostatic force can be updated to include these layers by replacing h_{gap} in (2) with an "effective gap:"

$$h_{eff} = h_{gap} + \frac{h_{isol}}{\varepsilon_r} \tag{6}$$

Where h_{isol} is the total thickness of dielectric isolation (sum of the two layers), and ε_r is the relative permittivity of the dielectric material. The effective gap is derived by evaluating the gap and isolation layers as series capacitances. Note that as the membrane deflects under loading, the isolation layer thicknesses will remain unchanged, while the vacuum gap will shrink proportionally to the displacement.

The static capacitance for the parallel plate CMUT can be defined as:

$$C_0 = \frac{A\varepsilon_0}{\left(h_{eff} + z_0\right)} \tag{7}$$

2.3.2. Dynamic Operation

In dynamic operation, the displacement and voltage terms in the above equations must now be updated to include both static DC terms and AC terms. Furthermore, we must now account for inertial and damping effects (mass and dashpot). The new force balance equation (in vacuum) becomes:

$$F_{es} = F_s + F_m + F_\zeta \tag{8}$$

Where the electrostatic force is now:

$$F_{es} = -\frac{\varepsilon_0 A\left(V_0 + \tilde{\mathrm{V}}\right)^2}{2\left(h_{eff} + (z_0 + \tilde{z})\right)^2} \tag{9}$$

The spring force is given by:

$$F_s = k(z_0 + \tilde{z}) \tag{10}$$

The inertial and damping forces are, respectively:

$$F_m = \rho h_{mem} A \frac{\partial^2 \tilde{z}}{\partial t^2} \tag{11}$$

$$F_\zeta = \xi \frac{\partial \tilde{z}}{\partial t} \tag{12}$$

Where ρ and ξ are the density and viscous damping coefficient of the membrane material, and h_{mem} is the thickness of the membrane. The A in (11) would normally refer to the membrane surface area, which may differ from the electrode area. However, for the parallel plate model this difference does not exist. Note that a tilde is being used to differentiate complex-valued dynamic terms from their static counterparts.

For transmit operation, a sinusoidal signal is applied to the electrodes. However, if we expand the electrostatic force in (9) to yield:

$$F_{es} = -\frac{\varepsilon_0 A(V_0^2 + 2V_0\tilde{V} + \tilde{V}^2)}{2\left(h_{eff} + (z_0 + \tilde{z})\right)^2} \tag{13}$$

Or alternatively rewriting this equation by replacing the dynamic terms with

$$\tilde{V} = v_{AC}\cos(\omega t) \tag{14}$$

$$\tilde{z} = z_{AC}\cos(\omega t) \tag{15}$$

We get:

$$F_{es} = -\frac{\varepsilon_0 A}{2\left(h_{eff} + (z_0 + \tilde{z})\right)^2}(V_0^2 + \frac{v_{AC}}{2} + 2V_0 v_{AC}\cos(\omega t) + v_{AC}\cos(2\omega t) \tag{16}$$

It becomes apparent that without a static bias, the membrane will oscillate with twice the frequency of the driving voltage. This is essentially due to the fact that electrostatic force is unipolar (always attractive). Thus, for transmit, a bias is typically applied to the electrodes first, and then a small AC signal (relative to DC) is superimposed on it such that the harmonic term becomes negligible. This DC bias also helps to pull the two electrodes closer together so that the AC voltage exerts a larger force than it would have without the DC bias, allowing for lower voltage operation.

For receive operation, a bias is also required but for different reasons. Primarily, the electrodes must have charge on them for an incoming pressure wave to cause any sort of detectable current. Furthermore, the bias brings the electrodes closer together and effectively increases the sensitivity of the CMUT. Typically, CMUTs in receive are biased very close to collapse to maximize sensitivity.

2.3.3. Coupling coefficient

The coupling coefficient is an important measure of the efficiency of CMUT transducers, and is defined as the unit-less ratio of the mechanical energy delivered to the load (i.e. radiation impedance) over the total energy stored by the transducer:

$$k_T^2 = \frac{E_{mech}}{E_{mech} + E_{elec}} \tag{17}$$

The coupling coefficient is a very useful parameter to characterize because piezoelectric transducers also use this value as a measure of efficiency [9]. This allows for direct comparison in terms of performance between the two different technologies.

For a parallel plate actuator, the coupling coefficient can be derived through analysis of the equivalent circuit (described later), and is given by [19] as:

$$k_T^2 = \frac{Q_0^2}{\varepsilon_0 A k (h_{eff} + z_0)} \tag{18}$$

And since charge can be defined by:

$$Q_0 = C_0 V_0 = \frac{\varepsilon_0 A V_0}{h_{eff} + z_0} \tag{19}$$

We can now rewrite (24) as:

$$k_T^2 = \frac{\varepsilon_0 A V_0^2}{k\left(h_{eff} + z_0\right)^3} \tag{20}$$

Now, if we rearrange (1), we can obtain:

$$V_0 = \sqrt{\frac{-2kz_0\left(h_{eff} + z_0\right)^2}{\varepsilon_0 A}} \tag{21}$$

Finally, after combining (20) with (23), we can obtain a very useful expression for the coupling coefficient:

$$k_T^2 = \frac{-2z_0}{h_{eff} + z_0} \tag{22}$$

If we plot the coupling coefficient as a function of applied bias normalized to collapse, as shown in Figure 12, a few key trends emerge. Primarily, the coupling coefficient will always be zero when no bias is applied, and will approach unity as the voltage is increased to collapse. Furthermore, the highly non-linear behavior of the coupling coefficient implies that for biases up to 90% of the collapse voltage, the coupling coefficient never goes beyond 0.45. However, as biases are applied above 90% towards

collapse, the coefficient can more than double as it approaches 1.0. This once again illustrates the importance of operating CMUTs as close to collapse as possible to achieve maximum efficiency.

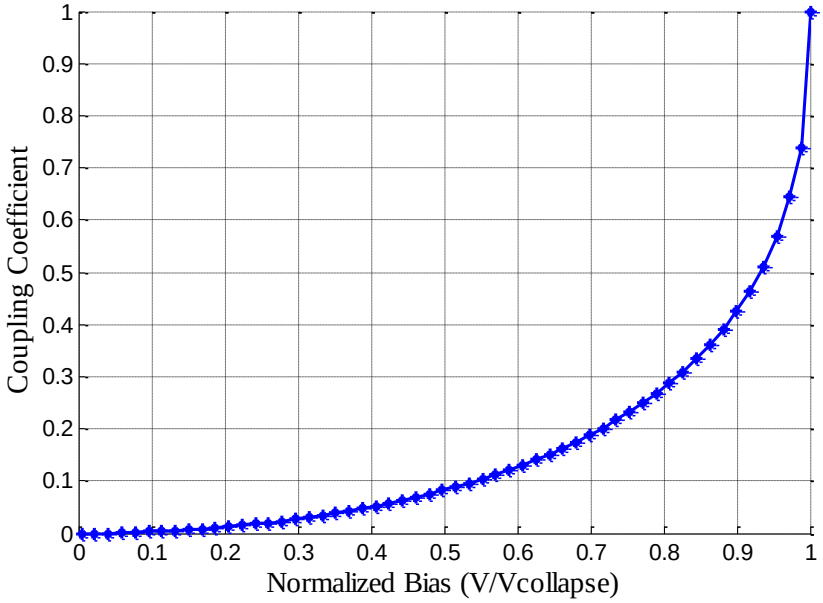


Figure 12. Coupling coefficient of a parallel plate actuator as a function of applied bias

Since experimentally measuring energy or displacement is rather difficult, a more practical definition can be given in terms of capacitance values [20, 21]:

$$k_T^2 = 1 - \frac{C^s}{C^T} \tag{23}$$

C^S is the static capacitance and is equivalent to C_0 defined in (7):

$$C^S = \frac{A\varepsilon_0}{(h_{eff} + z_0)} = \frac{Q}{V_0}\Big| V_0, z_0 \tag{24}$$

Where Q is the total charge stored in the capacitor, which is a function of applied bias and membrane displacement. CT is the free capacitance and is defined as the slope of the charge-voltage curve:

$$C^T = \frac{dQ}{dV}\Big| V_0, z_0 \tag{25}$$

This can be measured by increasing the bias on the membrane slightly from V0 and determining the rate of change of the charge.

2.4. Common Modeling Techniques

2.4.1. Equivalent Electro-Acoustic Circuit

The spring-mass-damper system is sufficient for a very rudimentary understanding of CMUT operation. However, a more accurate representation can be found in Mason's equivalent electro-acoustic circuit [22] for a CMUT in immersion shown in Figure 13. This model assumes a linear, small-signal analysis and parallel-plate actuation. Thus, the deformed shape of the membrane under loading is neglected as with the spring-mass-damper case. The small-signal analysis assumes that the dynamic displacement and voltage terms are much smaller than their static counterparts. In other words, $V_0 >> V_{AC}$ such that (16) becomes:

$$F_{es,AC} \cong -\frac{\varepsilon_0 A}{\left(h_{eff} + (z_0 + \tilde{z})\right)^2} V_0 v_{AC} \cos(\omega t) \tag{26}$$

This is a relatively accurate approximation for receive operation, because the incoming pressure waves typically generate very small deflections and charge variations relative to the static biased state. The approximation is slightly less accurate for transmit, as the driving signals can be relatively high to generate the required pressure amplitudes.

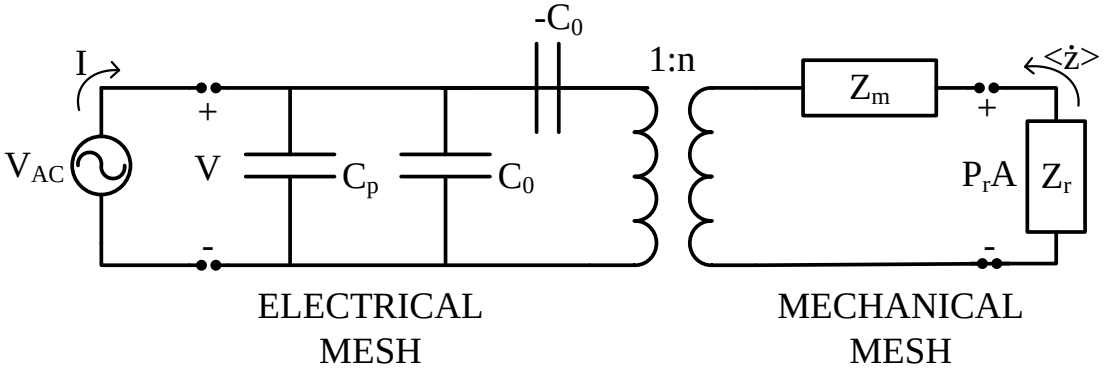


Figure 13. Mason's Equivalent Electro-Acoustic Circuit for a Parallel Plate Transducer

The circuit on the left side of Figure 13 is the electrical mesh, and contains all of the electrical properties of the CMUT represented as lumped circuit elements. The $-C_0$ term is included to account for a phenomenon known as spring softening. Spring softening occurs as a result of the electromechanical interaction and is seen as a softening in the small-signal spring constant of the membrane, and is proportional to the negative of the static capacitance [23]. In other words, as the CMUT is pulled closer to collapse, a given applied electrostatic force will result in a larger displacement, which can be interpreted mathematically as a softening in the mechanical stiffness. C_p accounts for any parasitic capacitances in the system that originate from connection lines or otherwise. The circuit on the right side represents the mechanical mesh, and accounts for all of the mechanical properties of the membrane. Note that in the mechanical mesh, the voltage and current analogs are force and velocity. Z_{mem} encompasses the mass-spring-damper system described earlier. However, in circuit form, it is represented as a series inductor-capacitor-resistor system. Z_{fluid} is for the radiation impedance of the membrane, and is a measure of how much pressure is generated in the surrounding fluid for a given average velocity across the membrane:

$$Z_{fluid} = \frac{P_r A}{\langle \dot{z} \rangle} \tag{27}$$

Where P is the pressure generated in the fluid by an average $\langle \dot{z} \rangle$ of the membrane velocity distribution. This parameter will be discussed at the end of this chapter.

To be clear, there are actually two separate equivalent circuits: One for transmit and one for receive. The circuit shown in Figure 13 is for a transmit circuit, because it contains a driving AC voltage in the electrical mesh, and no forced pressure loading across the Z_r term. A receiver circuit would contain amplifier circuitry in place of the AC driving signal in the electrical mesh, and would have a pressure force acting on the radiation impedance in the mechanical mesh.

2.4.2. Transformer Ratio

The parameter "n" is the transformer ratio and is used to convert components in the electrical mesh over to the mechanical mesh, and vice versa. It is defined as the ratio of the force across the mechanical mesh to the voltage across the electrical mesh. Rearranging (26):

$$n = \frac{F_{es,AC}}{V_{AC}} = -\frac{\varepsilon_0 A}{\left(h_{eff} + (z_0 + \tilde{z})\right)^2} V_0 \tag{28}$$

From this expression, it is clear that in order to maximize the transformer ratio, and hence the efficiency of the CMUT, the applied bias should be maximized which will in turn maximize displacement, further increasing the transformer ratio. It should also be pointed out that CMUT displacement is inherently limited by the collapse phenomenon. Thus, the maximum transformer ratio will occur when the initial gap is minimized, which will enable the smallest, pre-collapse gap height. This creates the design tradeoff mentioned earlier; a minimized gap will create a more efficient transducer, but high pressure outputs can only be achieved by large displacements.

2.4.3. Radiation impedance

The radiation impedance of a CMUT defines how the membrane vibration couples to the surrounding fluid, and vice versa. In vacuum, this parameter will be zero, as there will be no fluid through which a pressure wave can propagate. In air, this parameter will exist, but will be very small because ultrasonic pressure waves do not propagate efficiently in such low density mediums. In water and similar fluids (e.g. blood), the radiation impedance has a significant effect on the frequency response and must be considered carefully.

The simplest approximation for radiation impedance of a CMUT is given by the analytical model for a baffled piston radiator. This model assumes that the entire

membrane surface vibrates with a uniform velocity, as with a parallel plate actuator. As given in [24], for a circular piston of radius a:

$$Z_r = \rho_{fl} c_{fl} A[R_1(2k_{fl}a) + jX_1(2k_{fl}a)] \tag{29}$$

Where ρ_{fl} and c_{fl} are the density and speed of sound of the fluid medium, and k_{fl} is the wave number of the fluid, defined as:

$$k_{fl} = \frac{\omega}{c_{fl}} = \frac{2\pi f}{c_{fl}} = \frac{2\pi}{\lambda_{fl}} \tag{30}$$

Where f is the frequency of vibration of the membrane, and λ_{fl} is the corresponding wavelength in the fluid medium. R_1 and X_1 are given in terms of first order Bessel (J_1) and Struve (H_1) functions, respectively:

$$R_1(x) = 1 - \frac{2J_1(x)}{x} = \frac{x^2}{8} - \frac{x^4}{192} + \frac{x^6}{9216} \ldots \tag{31}$$

$$X_1(x) = \frac{2H_1(x)}{x} = \frac{4}{\pi}\left(\frac{x}{3} - \frac{x^3}{45} + \frac{x^5}{1575} \ldots\right) \tag{32}$$

Plotting the real and imaginary components of Z_r as functions of frequencies yields the plot shown in Figure 14. The impedance magnitudes have been normalized to the area of the arbitrarily sized piston, as well as to the quantity $\rho_{fl}c_{fl}$. This quantity is well-known as the radiation impedance of a plane wave propagating in free space. We can observe that as frequency increases, both terms increase up to a certain threshold. The real part of the normalized radiation impedance then begins to approach 1 and the imaginary part decays to 0. This implies that for high frequencies (small wavelengths) relative to the piston diameter, the radiation impedance will be mostly real, and will be equal to $A\rho_{fl}c_{fl}$.

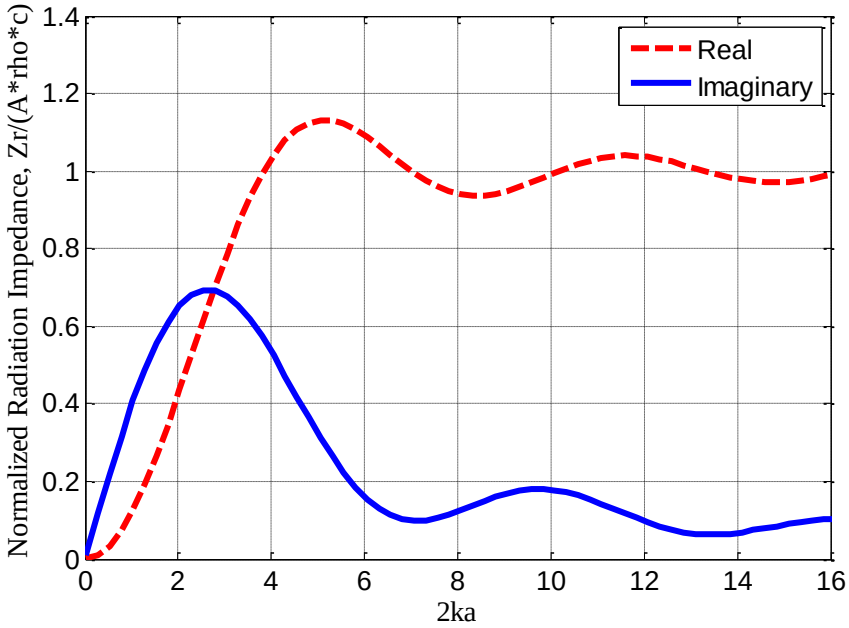


Figure 14. Radiation impedance of a baffled circular piston

For low frequencies, where $ak_{fl} << 1$, this expression can be simplified by only keeping the first terms of the power series expansions:

$$Z_r = R_r + jX_r \cong \frac{1}{2}\rho_{fl}c_{fl}A\left(k_{fl}a\right)^2 + j\left(\frac{8}{3\pi}\right)\rho_0 c_{fl}Ak_{fl}a \tag{33}$$

If we imagine Z_r as an equivalent circuit component, it could be incorporated as a series RL impedance at low frequencies. The resistance term, R_r, represents permanent energy loss as it is transferred from the CMUT system to the fluid environment. The inductor term, X_r, essentially adds an extra mass term to the mechanical mesh of the CMUT in immersion, and is referred to as mass-loading.

In reality, considering only the first mode of vibration, the membrane will flex and have maximum deflection at the center of the membrane, and zero deflection at its clamped edges. This discrepancy is even more pronounced at higher frequencies when higher order mode shapes become significant. Thus, for reasonable accuracy, the piston model is not sufficient for modeling CMUT radiation [1]. Furthermore, it has been shown that the standard equivalent circuit model is not valid for immersion applications

[25]. This is due to the fact that this simplified model assumes that the total membrane impedance in immersion can be computed by summing the membrane impedance in vacuum, Z_m, with the radiation impedance, Z_r, of the CMUT. However, these two parameters are not mutually independent of each other, and must be computed simultaneously to obtain the correct frequency response. This can be done straightforwardly using Finite Element Modeling (FEM) software.

2.4.4. Finite Element Modeling (FEM)

Finite Element Modeling (FEM) is a commonly used technique for simulating CMUT behavior. Of all previously mentioned modeling methods, FEM is the most accurate for a number of reasons. It has the ability to incorporate the exact shape and size of the membrane and its electrodes. It can account for the flexural behavior of the membrane as it oscillates, which eliminates the parallel-plate simplification used up to this point in the discussion. It can also be used for computing Z_m and Z_r simultaneously [25] which, as mentioned earlier, is the only correct way for computing the total mechanical impedance in immersion.

2.4.4.1. Single CMUT Membrane in Immersion

The simplest FE model for a CMUT is that of a single, baffled membrane vibrating in a semi-infinite fluid. In place of electrostatic forcing, a generic pressure force is applied to the membrane. For low biases far from collapse, this is a relatively accurate simplification because spring softening will not have a noticeable effect on the frequency response. Electrostatic forces can be added fairly straightforwardly, but this can often complicate the model and increase computation time.

2D plane-strain structural elements are very accurate for modeling the static deflection of long, rectangular CMUTs under electrostatic loading. However, for frequency response analyses, the 2D models are insufficient because they completely

neglect any mode shapes that occur in the long dimension of the membrane. These mode shapes will have an observable effect on the overall behavior of the CMUT, and should be accounted for. Furthermore, 2D models are even less accurate for membranes with low length-to-width aspect ratios, because the 2D model will neglect the added stiffness effects of the 3rd dimension. Thus, for frequency response analyses of square and rectangular membranes, 3D models are the most desirable.

Unfortunately, 3D models are also the most computationally expensive simulations. This is especially true considering the fluid space must be large enough to allow sufficient propagation of the lower frequency waves, and the mesh size of the fluid elements must be small enough to resolve the higher frequency wavelengths. Validated with convergence testing, the common rule of thumb is to ensure the highest frequency contains at least 10-12 nodes per wavelength, and that the fluid radius is at least 1/5th the size of the largest wavelength in a given frequency sweep simulation. Thus, care must be taken to minimize the complexity of the model without sacrificing accuracy.

A sample FE model of a single CMUT in immersion was constructed using Comsol 3.5a FEM software, and is shown in Figure 15. Note that quarter symmetry was utilized. The outer radius of the fluid space contains an absorbing boundary condition to prevent pressure waves from reflecting off its surface, which would otherwise cause unrealistic standing waves in the fluid. This boundary condition allows for the "infinite" fluid space to be modeled using a finite number of elements. The other 3 edges of the fluid space, excluding the portion in contact with the membrane, are set as hard wall boundaries. On the bottom surface of the fluid, this creates the baffle condition. On the two vertical faces of the fluid, this creates a symmetry boundary condition. On the surfaces in which the membrane and fluid are in contact, a fluid-structure interface boundary condition is applied. This effectively couples the structural elements of the membrane to the fluid elements, allowing for 2-way transference of energy between the

two domains. A harmonic pressure loading is applied to the electrode, and the membrane is fixed at the appropriate locations.

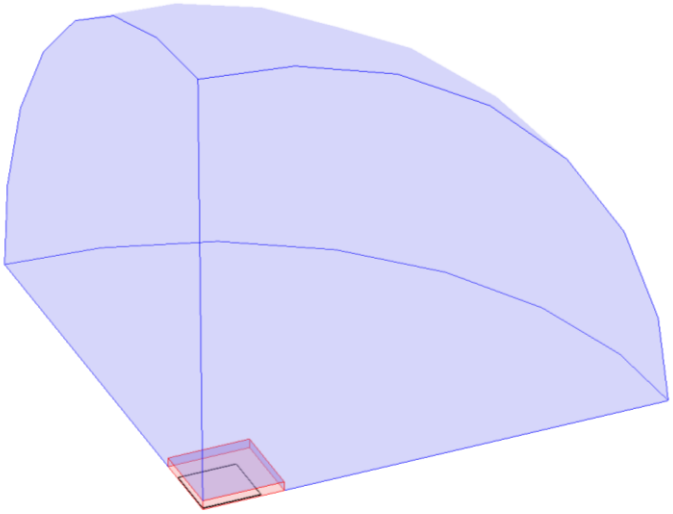


Figure 15. FEM model geometry for a single, baffled CMUT membrane in a semi-infinite fluid

A sample frequency response for a 35um x 35um x 1um membrane vibrating in water is shown in Figure 16. For comparison purposes, a plot for the frequency response of a membrane vibrating in vacuum (no fluid loading) was included by simply removing the fluid space from the FE model and resolving the model. To create the graphs, the surface velocity over the membrane was averaged at each frequency and converted to decibels.

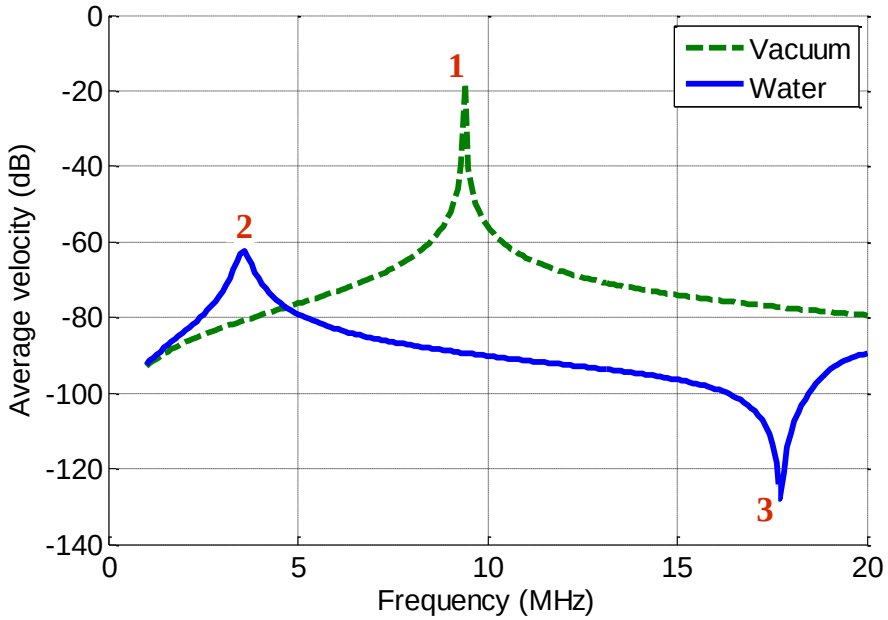


Figure 16. Frequency response from a finite element model of a 35x35x1um single CMUT membrane vibrating in vacuum and water.

As expected, due to the absence of damping from the fluid or membrane, the main resonance in vacuum at peak 1 has a very high Q. The immersed membrane case experiences the fluid-loading effect of water, causing a shift to the lower frequency at peak 2. The energy transferred to the fluid also has a damping effect on the membrane vibration, creating a much lower Q, lower amplitude peak. To be clear, the actual magnitude of the resonance in vacuum is meaningless. This is because without any sort of damping, at true resonance the membrane oscillations will continue increasing to infinite magnitude. The finite magnitude observed in the graph is only due to the fact that the frequency sweep has a finite number of steps, and tends to only find points near resonance. For it to be more realistic, viscoelastic damping could easily be added to the structural elements. In water, however, the magnitude of the resonance is meaningful even without adding viscoelastic damping. This is because the damping effect from water is so large that losses in the membrane material can essentially be neglected. The final characteristic to note in Figure 16 is the dip that occurs at location 3 for the water

curve. This minimum occurs at the 1st mechanical anti-resonance of the membrane, during which the mechanical impedance tends towards infinity. The anti-resonance will also occur in vacuum, but does not occur within the given frequency range. The fluid-loading effect of the water was enough to shift the anti-resonance to a sufficiently low frequency.

Two important parameters of a CMUT are its center frequency and bandwidth, and so they will be explicitly defined here for clarity purposes. Both of these significantly affect the imaging capabilities of the transducer, and these effects will be discussed in more detail at the end of this chapter. The center frequency is usually defined as being halfway between the bounding lower and upper frequencies that occur at 3dB below the spectral maximum:

$$f_c = \frac{f_l + f_h}{2} \tag{34}$$

Note that the center frequency will not always equal the true maximum of the spectral graph. 6dB bandwidth values are also commonly used for pulse-echo measurements in which the same transducer both transmitted and received the same signal, to account for the signal being filtered by the CMUT frequency response two times. Different applications may call for other dB limits as well. The bandwidth of a CMUT is typically defined as the fraction of the distance between the upper and lower frequencies defined earlier, divided by the center frequency:

$$BW = \frac{f_h - f_l}{f_c} * 100\% \tag{35}$$

2.4.4.2. CMUT Array Behavior in Immersion

CMUTs are never operated as single membranes for ultrasound imaging. Instead, they are typically configured into large arrays containing many CMUT membranes. For immersion applications, such as FL-IVUS, the presence of neighboring membranes must

be accounted for when modeling CMUTs. This is due to an effect known as acoustic crosstalk. In essence, when multiple membranes are positioned close to each other and actuated within a fluid environment, the resulting pressure waves that propagate from each membrane have a significant forcing effect on the neighboring membranes. This forcing effect changes the behavior of every CMUT membrane in the array, and thus has a substantial impact on the overall frequency response of the transducer.

To observe this effect, 2 separate finite element models were created: One containing 2 membranes, and one of a 2x2 array of 4 membranes with 5 um spacing in each direction. Similar to the single membrane case, the membranes are baffled and actuated in a lossless, semi-infinite fluid medium. Quarter symmetry boundary conditions were utilized once again to decrease computation time. All membranes were actuated in phase with each other using equal amplitude pressures. A plot of their corresponding frequency responses along with the single membrane case can be found in Figure 17. The curves in the multi-membrane simulation were calculated using the average velocity over all active membranes.

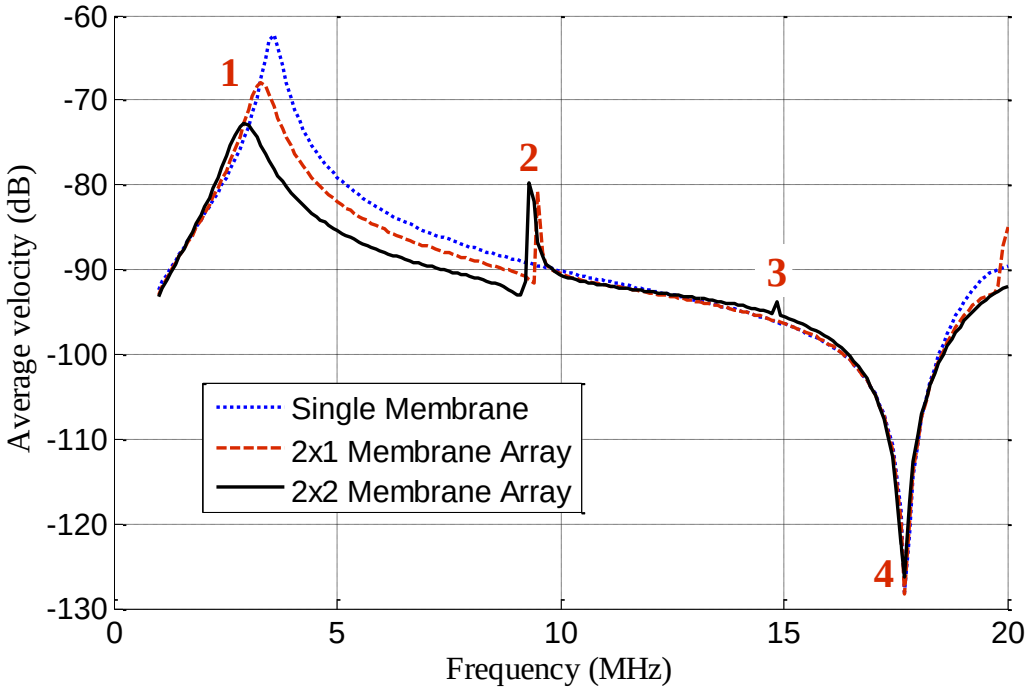


Figure 17. Frequency response of FEM simulation for single membrane, 2x1 array of membranes, and 2x2 array of membranes using 5um spacing

Figure 17 clearly demonstrates the dramatic effect that acoustic crosstalk has on CMUT behavior. The main peak at location 1 shifts downward in frequency as more membranes are added. Depending on the array parameters, this peak could have also shifted to higher frequencies. Peak 1 also becomes higher in bandwidth and lower in overall magnitude with the addition of membranes. It should be noted that although *average* velocity decreases, the *total* pressure output in array operation will be much larger. The dip at location 4 remains unchanged for all 3 simulations because it is a solely a function of the fluid-loaded mechanical properties of the membrane, not the array geometry. At peak 2 and 4, a different phenomenon is occurring. These two sharp features exist because acoustic crosstalk is creating a periodic forcing on neighboring membranes in such a way that the radiation of one membrane is being amplified by the pressure waves being radiated by neighboring membranes. They are extremely narrowband because they correspond to precise frequencies at which the geometry of the

array (e.g. pitch between membranes) is equal to some significant multiple of the wavelength of the propagating frequency. For other array configurations, the opposite will occur, and a sharp dip will exist due to a cancelling effect between CMUTs. These crosstalk effects can be attributed to multiple types of phenomena such as Stoneley-Scholte interface waves, leaky Rayleigh surface waves, Lamb waves, and fluid-coupled excitation of anti-symmetric membrane modes [26], some of which were not accounted for in the FEM used to generate the above plots.

2.5. Effects of Acoustic Crosstalk in CMUTs on Imaging

The frequency response of a CMUT array is an important parameter to consider when evaluating its effectiveness as an imaging transducer. In general, a high bandwidth transducer is preferred. This is because, according to Fourier Transform theory, a higher bandwidth transducer equates to a shorter temporal response, which is necessary to achieve maximum axial resolution during imaging [27]. Furthermore, the operational bandwidth and its band edges should be free of sharp transitions or corners. These features would translate to an increased amount of "ringing" after the main signal, which can cause a single target to be artificially elongated or even to appear as a series of targets in a reconstructed image [28]. Ringing will also increase the size of the dead zone directly in front of the transducer array [29]. Clearly, this implies that the sharp features due to acoustic crosstalk in the figures above need to be carefully considered (and minimized) during the CMUT design process. A high bandwidth is also desirable because it extends the useable frequency range of the transducer, allowing for a range of applications that could benefit from a single transducer capable of imaging at multiple frequencies (e.g. low frequency for deep penetration depths, high frequency for good lateral resolution).

The center frequency is another important characteristic of the frequency response which significantly affects transducer performance. Specifically, center frequency

directly relates to the lateral resolution of a CMUT imaging array. A common approximation for lateral resolution can be estimated from [30]:

$$Lateral\ Resolution = \lambda * \text{F-number} \tag{36}$$

Where the F-number is defined as the ratio of the focal distance to the diameter of the transducer array, and λ is the wavelength of the corresponding center frequency. Thus, a designer needs to be able to predict the center frequency of a transducer in order to ensure that it provides sufficient resolution. Once again, this implies that any model used to design a CMUT transducer must account for acoustic crosstalk because it significantly affects the center frequency.

One other consideration of crosstalk is its effect on the radiation pattern of an array. At certain frequencies, fluid coupling disrupts the uniform displacement of the membranes, causing different membranes to be vibrating with different amplitudes and phases depending on their location in the array. Some frequencies also cause mode shapes other than the preferred 1st mode to occur. This non-ideal behavior can alter the radiation pattern of the individual elements in the array, and decrease the transducer's ability to accurately resolve features in the surrounding environment. For instance, consider an example in which fluid-coupling causes membranes in the center of the array to actuate with larger magnitudes than membranes on the outside of the array given the same input signal. This would cause all signals sent and received from the center of the array to be artificially weighted higher than the other elements in the array during imaging. For another example, consider that crosstalk even causes "dead" elements (elements that have not been actuated with an AC signal) to vibrate. However, image reconstruction algorithms assume that only the active element is contributing to the transmitted pressure wave. This false assumption can lead to inaccuracies.

It is clear that crosstalk has a significant impact on the frequency response of an array. It is also clear that this frequency response is a direct indicator of the transducer's

imaging capabilities. Thus, in order to accurately design and optimize CMUT arrays for imaging applications, fluid crosstalk must be considered.

CHAPTER 3

MODELING CMUT ARRAYS IN IMMERSION

The previous chapter demonstrated how CMUTs show great promise for advancing the state of current IVUS technology. In order to optimize these transducers for particular applications, comprehensive models need to be created that can accurately predict their behavior in a variety of environments. It is essential that these models account for acoustic crosstalk if the CMUTs are being operated in immersion, such as in FL-IVUS applications. The following will discuss the drawbacks of common CMUT simulation techniques, and will then present and thoroughly describe an alternative modeling strategy for efficient, accurate prediction of CMUT array behavior.

3.1. Drawbacks of Common Modeling Techniques

3.1.1. Equivalent Electro-Acoustic Circuit

The equivalent circuit is a very useful and efficient model for understanding the dynamics of a single CMUT membrane. However, as discussed earlier, the standard approach of independently calculating mechanical and radiation impedance terms is not accurate. This implies that an alternative method must be used to solve for both parameters simultaneously, such as FEM. This total impedance could then be substituted into the equivalent circuit as a lumped-element. Unfortunately, this implies a lack of adaptability of the equivalent circuit model as it would require a new FE (or an alternative) model whenever the properties of the membrane or its environment are changed.

Regardless of its usefulness concerning a single membrane, the standard equivalent circuit is ill-suited for calculation of true array behavior. It is possible to duplicate the components of the electrical and mechanical mesh and include these copies

in a larger system-scale equivalent circuit. However, although this modification could account for increases in sensitivity or overall pressure output due to the extra membranes, it completely neglects the mutual fluid-coupling effects that each membrane has on its neighbor. Different membranes may have different impedances based on their location in the array, the number of membranes present, and the array configuration, so simply copying the same impedance multiple times is not sufficient. Once again, some other method of simultaneously solving for the unique impedance terms of each membrane must be utilized. These new values could then be incorporated as lumped elements into the circuit, but once again this demonstrates the equivalent circuit's lack of adaptability to changes in the transducer operating conditions.

3.1.2. Finite Element Model

3.1.2.1. Modeling Full CMUT Array

Finite Element Modeling is essentially the gold standard for accuracy in modeling CMUTs. It is theoretically capable of accounting for all physical phenomena that occur within a given CMUT system including the true deformed shape of the membrane and electrodes (i.e. no parallel plate assumption), non-linear electrostatic effects, and even acoustic crosstalk. Furthermore, any membrane shape or array geometry could potentially be modeled to complete precision.

Unfortunately, with this flexibility and comprehensiveness comes increased computation time. As discussed in Chapter 2, dynamic FE models must be 3D to incorporate the stiffness and mode shape contributions of both lateral dimensions of the membrane. Mesh elements of the membrane must be small enough to accurately predict mechanical performance. In order to resolve the finer features in the geometry such as the very thin electrodes or gap, the mesh elements must be even smaller in those locations. 3D structural meshes in FEM typically contain 3 degrees of freedom (DoFs)

corresponding to displacements along the 3 coordinate axes. Thus, a model only characterizing the structural mechanics of the membrane can easily become relatively large in size. Often, the membrane shape is simplified to neglect the small effects that fine geometry features will have on the overall behavior.

For the model to be useful, however, structural mechanics alone is not sufficient. The spring-softening effects of the static bias must be incorporated, which means the nodes between the electrodes will now have additional voltage degrees of freedom. Additionally, including the mesh for the semi-infinite fluid space adds significantly to the size of the total stiffness matrix. This is because the fluid space radius must be large enough to allow propagation of at least $1/5^{th}$ of the largest wavelength, and its elemental mesh should be small enough to include at least 10-12 nodes per wavelength of the highest frequency, as determined through convergence tests.

Considering all of this information, it is easy to see how a finite element model can quickly become unfeasibly large when extended to account for array effects. Each extra membrane means an additional structural mesh of 3 DoFs per node plus an extended fluid space. Of course, symmetry in an array can be exploited, but in most cases the array is large enough that even this simplification is not sufficient. Consider, for example, the FL-IVUS Dual Ring Array which contains as many as 480 membranes. A frequency sweep could also be split into smaller intervals, so that the fluid mesh would not need to simultaneously have a large diameter and finely meshed elements, but for arrays involving hundreds of membranes this strategy will likely see only limited improvements. Clearly, even with symmetry, careful meshing, and optimized solver parameters, an FE model of a full CMUT array would take an unreasonable amount of time to solve. Of course, a partial model could be constructed that only includes a single membrane and a few of its neighbors, but this simplification could disregard some important effects from membranes located further away. Furthermore, even a partial

model such as this could potentially be too large to solve with a typical laboratory computer.

3.1.2.2. Waveguide Approach

A clever trick that can be applied to finite element simulations of CMUT arrays is the so-called "waveguide" approach [31]. This technique assumes that the array can be modeled using a single membrane vibrating into a long waveguide, who's walls are located on each side of the membrane. The end of the waveguide should include an absorbing boundary condition to prevent unrealistic standing waves from occurring. Considering the Method of Images [24], the rigid walls act as symmetry boundary conditions. In essence, it is as if each membrane is located in the center of a sea of infinite membranes all vibrating in the fluid with identical phase and magnitude. This approach accounts for some of the crosstalk effects that neighboring elements will have on a given membrane. However, it is not completely realistic because all elements of an array do not vibrate in phase with equal magnitude. Crosstalk can cause some membranes to be out of phase with each other. It can excite anti-symmetric modes in membranes. It can also result in location-dependent velocity magnitudes among elements in an array. This is not to mention the effect that dead elements will have on array behavior, even though they are not being electrostatically actuated. The waveguide approach, although somewhat accurate, does not account for any of these effects.

3.2. A Finite Difference, Boundary-Element Approach to Modeling CMUT Arrays

As an alternative solution to simulating CMUT arrays, Meynier et al. [1] proposed a compromise between the accuracy of a full electromechanical FEM and the simplicity and efficiency of an equivalent circuit. Their approach was based on the work of Certon et al. [32], in which the true deflection profile of a CMUT in air was modeled using a finite difference (FD) approximation of the 4^{th} order thin plate equations as given by

Timoshenko [33]. Starting with this finite difference approach, they then extended the calculation to account for fluid coupling by utilizing a Boundary Element Matrix (BEM) based on the Green's function for a baffled radiator in a semi-infinite fluid. These analytical techniques effectively reduced the entire CMUT array simulation to a 2D mesh, as both the BEM and thin plate equations only required a mesh on the surface of the membranes in the array. The rest of this chapter will focus on describing the development of this method in detail, and will conclude with a summary of its capabilities and limitations. The remaining chapters will focus on evaluating the accuracy of the simulation and implementing it for CMUT array design optimization. For reference regarding the upcoming sections, a figure denoting the geometrical parameters of a modeled CMUT membrane is shown in Figure 18.

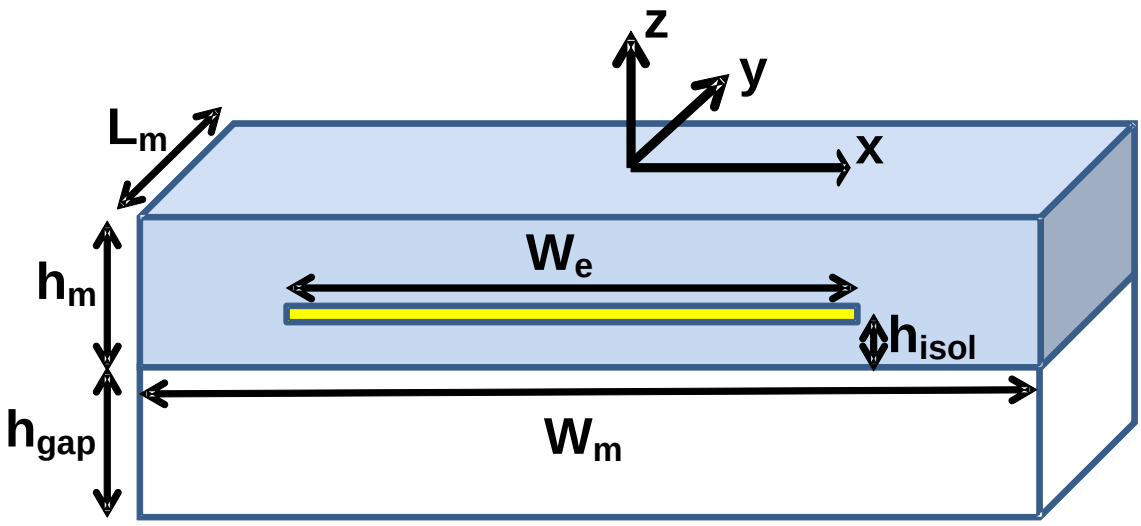


Figure 18. Geometry of a CMUT membrane

3.3. Force Balance Equation

The key characteristic of the technique proposed by Meynier et al. [1] is that it meshes the surface of each CMUT membrane in an array, and then solves a system of

force balance equations at each node using simple matrix operations. For a given location on the unmeshed membrane, the force balance can be written as:

$$P_{es}(x,y,t) = P_s(x,y,t) + P_m(x,y,t) + P_\zeta(x,y,t) + P_r(x,y,t) + P_a(x,y) \quad (37)$$

Where P_{es} is the dynamic electrostatic pressure exerted by the electrodes, P_s is the mechanical return stresses exerted by the membrane, P_m and P_ζ account for mass and damping in the membrane, P_a is the static atmospheric pressure acting on the membrane, and P_r is the local pressure in the fluid acting on the membrane. Note that all of the terms in equation (37) are normalized by the local area and given in units of Force/Area. Each term is described in detail in the following sections.

3.3.1. Mechanical Stiffness

3.3.1.1. Thin Plate Equations

Assuming the lateral dimensions of the CMUT membrane are much larger than its thickness, the stiffness, K, of the membrane can be accurately approximated by the thin plate equations developed by Timoshenko [33]. It is given as the flexural plate operator:

$$P_s(x,y,t) = K * u(x,y,t) = \frac{\partial^2 M_x}{\partial x^2} + \frac{\partial^2 M_y}{\partial y^2} + 2\frac{\partial^2 M_{xy}}{\partial x \partial y} \quad (38)$$

Where u is the time and location dependent displacement of the membrane in the z-direction. Note that the operator K multiplied with displacement will yield the return stresses in an elemental volume of the plate, given in units of Force/Area. M_x, M_y, and M_{xy} are the bending moments on an elemental volume of the plate, as illustrated in Figure 19. They are defined as:

$$M_x = D\left(\frac{d^2u}{dx^2} + \nu\frac{d^2u}{dy^2}\right)$$

$$M_y = D\left(\frac{d^2u}{dy^2} + \nu\frac{d^2u}{dx^2}\right) \tag{39}$$

$$M_{xy} = D(1-\nu)\frac{d^2u}{dxdy}$$

Where ν is the Poisson coefficient of the plate, and D is the flexural rigidity given as:

$$D = \frac{Eh_{mem}^3}{12(1-\nu^2)} \tag{40}$$

E is the Young's modulus of the membrane material, and h_{mem} is the total thickness of the membrane. Note that if the membrane material is layered as it is with the deposition of parylene, D and ν can be approximated by homogenized equivalent values, assuming the membrane deflection is small relative to the membrane thickness and that the additional layer covers the entire surface of the membrane [32, 34]. For layers that only partially cover the membrane, a more complex analysis using stepwise laminated plate theory may be necessary [34].

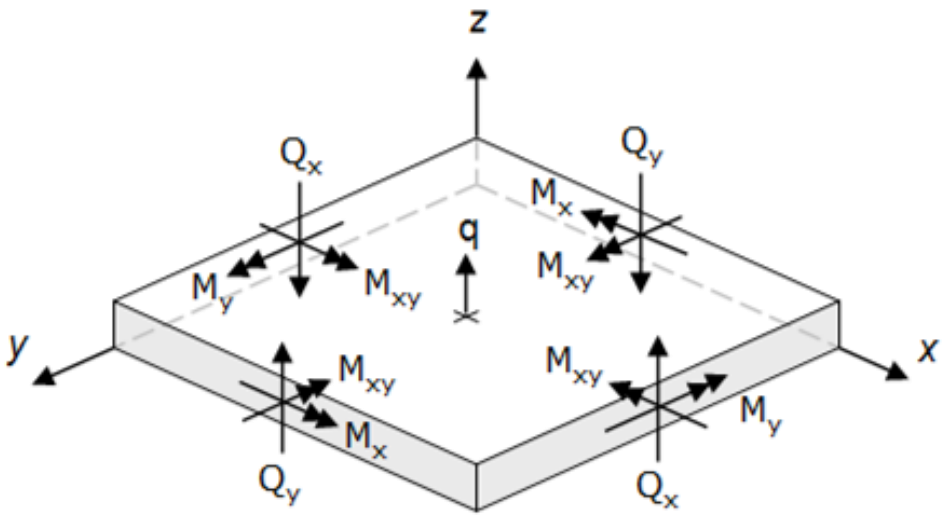


Figure 19. Illustration of the bending moments (M) and shear forces (Q) acting on an elemental volume of a thin plate given an applied stress, q .

Clearly, obtaining a direct analytical solution of the 4th order plate equation is not feasible. Instead, the equation can be solved numerically by sampling the membrane with a nodal 2D mesh, and utilizing finite difference approximations to estimate the higher order derivatives.

3.3.1.2. Finite Difference Method Overview

The finite difference method (FDM) estimates derivatives of a function at a specific node by utilizing known values of the function nearby to that node. It derives its formulas using Taylor series expansions. For example, the value of a function of one variable f(x) at arbitrary location ($x_0 + \Delta x$), can be approximated with a Taylor series as:

$$f(x_0 + \Delta x) = f(x_0) + \frac{f_x(x_0)}{1!}\Delta x + \frac{f_{xx}(x_0)}{2!}(\Delta x)^2 + \cdots + \frac{f_n(x_0)}{n!}(\Delta x)^n \qquad (41)$$

If we rearrange this equation to solve for the first derivative, $f_x(x)$, we get:

$$f_x(x_0) = \frac{f(x_0 + \Delta x) - f_x(x_0)}{\Delta x} + \frac{R(x)}{\Delta x} \qquad (42)$$

If the remainder term is sufficiently small the derivative can be accurately approximated as:

$$f_x(x_0) \approx \frac{f(x_0 + \Delta x) - f_x(x_0)}{\Delta x} \tag{43}$$

This is known as a forward difference approximation for the first derivative of f, because it approximates the derivative using the value of a point slightly forward of itself. A more accurate approximation is obtained using a central difference approximation, which uses information from 2 extra points, one forward and backward from the location of interest. Note that a finer mesh (smaller Δx) will always result in a more accurate approximation because it minimizes the size of the higher order remainder term. More accuracy (and higher order derivatives) can be obtained by keeping more terms of the Taylor series expansions, and by incorporating values of more surrounding nodes.

Central difference approximations for 4th order derivatives require at least 2 points in front and behind the node of interest. Higher order derivatives of two variable functions (e.g. u(x,y)) are even more complex as they require values of nodes in two dimensions. A list of finite difference approximations up to the second order can be found in [35]. A list of all higher order finite difference approximations used for this model can be found in the Appendices. Note how some of the higher order derivatives require as many as 15 different points surrounding the node of interest.

3.3.1.3. Finite Difference Approximation of the Thin Plate Equations

Using the above information for the problem at hand, we can create a nodal mesh over the surface of the membrane, as shown in Figure 20. Then, after expanding the expression for the flexural plate operator in Equation (38), each of its individual derivative components can be replaced with the associated FD approximations listed in the Appendix. The expanded version of (38) can be found in the Appendix as well.

Thus, each node will have a unique expression for the flexural plate operator, K, which is simply a linear function of the nodal displacement values at nearby nodes and the distance between those nodes (mesh size). This implies that there is now a system of N unique force balance equations, where N is the number of nodes in the mesh. We can therefore rewrite the mechanical stiffness term of the force balance equation in matrix form as:

$$P_s(x,y,t) = \left[K^{ij}\right]\{\mathrm{u}^{\mathrm{i}}\}, \qquad i,j = 1,2 \ldots N$$

$$\left[K^{ij}\right] = \begin{bmatrix} K^{11} & K^{12} & \cdots & K^{1N} \\ K^{21} & K^{22} & \cdots & K^{2N} \\ \vdots & \vdots & \ddots & \vdots \\ K^{N1} & K^{N2} & \cdots & K^{NN} \end{bmatrix} \tag{44}$$

Where $[K^{ij}]$ is an NxN stiffness matrix containing the coefficients of each nodal flexural plate operator expression. Note that [K] will be extremely sparse since each equation is only dependent on the displacement of a node and no more than 15 of its neighboring nodes. $\{u^i\}$ is a column vector containing the static $\{u_0^i\}$ and dynamic $\{\tilde{u}^i\}$ displacement of each node in the z-direction at a given time:

$$\{u^i\} = \{u_0^i\} + \{\tilde{u}^i\}e^{j\omega t} = \begin{Bmatrix} u^1 \\ u^2 \\ \vdots \\ u^N \end{Bmatrix} \tag{45}$$

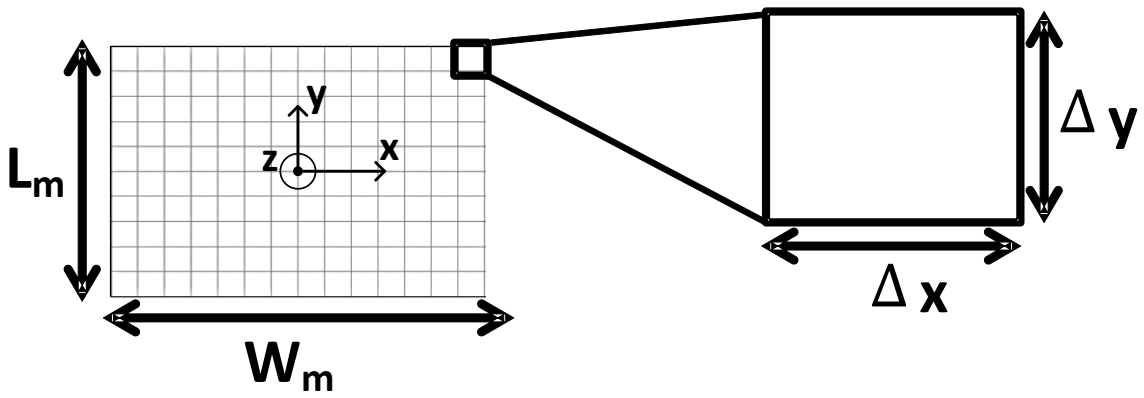


Figure 20. Illustration of the membrane nodal mesh

3.3.1.4. Boundary Conditions of the Flexural Plate Operator

For the mechanical stiffness matrix to be fully defined, boundary conditions must be applied to the edges of the membrane. In this model, only the surface of the membrane has been meshed, but the discretized form of the flexural plate operator requires nodal values outside of this mesh. For example, consider the bottom left node of the mesh in Figure 20 as $x_{i,j}$, where "i" and "j" are indices denoting the node's position in relation to its neighbors, as demonstrated in Figure 21. The finite difference approximations need information regarding the nodal displacements at indices $-2 \leq i,j \leq 2$, but the vector [u] doesn't contain values for any nodes $i,j < 0$. This is where boundary conditions become important. If we assume that the outer edges of the membrane are clamped, then the boundary conditions at the edges of the membrane will be:

$$
\begin{aligned}
u\left(x, \left(y \le \frac{-L_m}{2}\right)\right) &= 0 \\
u\left(\left(x \le \frac{-W_m}{2}\right), y\right) &= 0 \\
u\left(x, \left(y \ge \frac{L_m}{2}\right)\right) &= 0 \\
u\left(\left(x \ge \frac{W_m}{2}\right), y\right) &= 0
\end{aligned}
\tag{46}
$$

Thus, from these boundary conditions, the displacement values outside of the mesh that are required by the finite difference formulation can be set to 0. Because of this, the redundant rows and columns in the stiffness matrix must be removed to account for the reduced displacement vector size, such that:

$$P_s(x, y, t) = \left[K^{ij}\right]\{\mathrm{u}^{\mathrm{i}}\}, \qquad i, j = 1,2, \ldots N^* \tag{47}$$

Where N^* denotes the total number of nodes being solved for, which has removed the clamped nodes on the edge of the membrane.

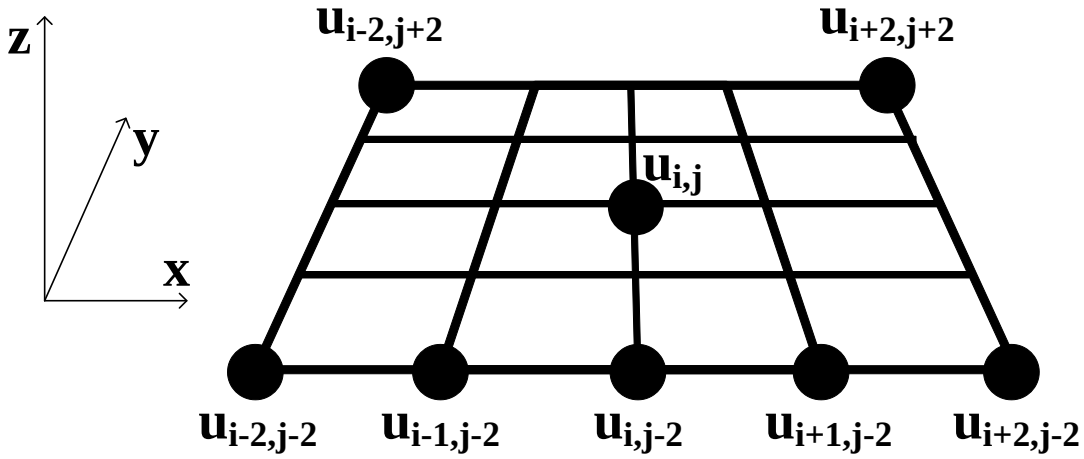


Figure 21. Index notation for finite difference approximations around a specific node, where u is the vertical displacement of a node, and i and j denote the relative location of neighboring nodes

The analysis can be simplified further if quarter symmetry of the membrane is utilized. This would imply that a nodal displacement required by the FD approximations that is across the symmetry boundaries and outside the meshed region, can be assumed to be equal to its "mirror image" node, corresponding to where it would be located were it to be reflected across the symmetry boundary. It would also imply the system of equations to solve would be reduced by $1/4^{th}$, decreasing computation time. However, as will be discussed later, symmetry boundary conditions can only be utilized for single membrane simulations, because array calculations cannot assume symmetrical loading on each membrane. Thus, the usefulness of this simplification is limited.

3.3.2. Mass

The mass term in the original force balance equation can be written in terms of Newton's second law as:

$$P_m(x,y,t) = \rho(x,y) * \frac{d^2u(x,y,t)}{dt^2} \tag{48}$$

Where ρ is the local surface density (mass/area) of the membrane. However, since the membrane surface has now been discretized to accommodate the finite difference approach, it is necessary that all remaining terms in the original force balance equation be converted to matrix form. Thus, utilizing (45), the mass term will now become:

$$P_m\left(\{x^i\},\{y^i\},t\right) = -\omega^2[M]\{\tilde{u}^i\}e^{j\omega t} \tag{49}$$

Note how the static portion of {u} has been eliminated since its time derivative is zero. [M] is a diagonal NxN matrix consisting of the local surface density at each node. For a uniform membrane consisting of a single homogeneous material,

$$[M^{ii}] = \rho_{mem} h_{mem} [I]_N, \quad i = 1,2 \dots N^*$$

$$[M^{ii}] = \begin{bmatrix} M^{11} & 0 & \cdots & 0 \\ 0 & M^{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & M^{NN} \end{bmatrix} \tag{50}$$

Where ρ_{mem} is the density of the membrane material and $[\text{I}]_{\text{N}}$ is an identity matrix of size NxN. [M] is diagonal because, unlike the mechanical stiffness, P_m at a specific node is only dependent on the mass at that node, and is not dependent on neighboring nodes. For a non-uniform membrane, the matrix would simply need to be updated to account for certain nodes having a different local density.

3.3.3. Damping

The damping term can be given by:

$$P_\zeta(x, y, t) = \zeta \frac{\partial u(x, y, t)}{\partial t} \tag{51}$$

Where ζ is a damping coefficient for the membrane. This can be rewritten in matrix form as:

$$P_\zeta(\{x\}, \{y\}, t) = j\omega\zeta [I]_{N^*} \{\tilde{u}^i\} e^{j\omega t} = j\omega [\zeta^{ii}] \{\tilde{u}^i\} e^{j\omega t}, \quad i = 1,2 \dots N^*$$

$$[\zeta^{ii}] = \begin{bmatrix} \zeta^{11} & 0 & \cdots & 0 \\ 0 & \zeta^{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \zeta^{N^*N^*} \end{bmatrix}, \quad \zeta^{11} = \zeta^{22} \dots = \zeta^{N^*N^*} \tag{52}$$

Once again, the static portion of the displacement has disappeared due to the time derivative. Note that this expression assumes a single layer membrane, and defines damping using a diagonal matrix in which all non-zero entries are identical in magnitude.

3.3.4. Atmospheric Pressure

P_a is a static pressure applied on the membrane surface by the environment (atmospheric + hydrostatic pressure). For most purposes, it can be defined as:

$$\{P_a^i\} = (\rho_{fl} g H + P_{atm}) \begin{Bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{Bmatrix}$$

$$\{P_a^i\} = \begin{Bmatrix} P_a^1 \\ P_a^2 \\ \vdots \\ P_a^{N^*} \end{Bmatrix}, \qquad P_a^1 = P_a^2 \ldots = P_a^{N^*} = \rho_{fl} g H + P_{atm} \tag{53}$$

Where P_{atm} is the atmospheric pressure, g is the acceleration due to gravity, ρ_{fl} is the density of the environmental fluid (if in immersion), and H is the height of fluid above the membrane. In general, its effect on the membrane is minimal and it can usually be neglected without sacrificing accuracy.

3.3.5. Electrostatic Pressure

The electrostatic pressure on the electrodes was defined in chapter 2:

$$P_{es}(x,y,t) = -\frac{\varepsilon_0 V(t)^2 \Psi(\mathrm{x,y})}{2\left(h_{eff} + u(x,y,t)\right)^2}, where$$

$$V(t) = V_0 + \tilde{V} e^{j\omega t} \tag{54}$$

$$\Psi(\mathrm{x,y}) = \begin{cases} 1 & -\frac{W_e}{2} \le x \le \frac{W_e}{2} \ and \ -\frac{L_e}{2} \le y \le \frac{L_e}{2} \\ 0 & else \end{cases}$$

ε_0 is the permittivity of free space and h_{eff} is the effective distance between the electrodes due to the series capacitances from the vacuum gap and the isolation layers:

$$h_{eff} = h_{gap} + \frac{h_{isol}}{\varepsilon_r} \tag{55}$$

and ε_r is the relative permittivity of the isolation layers. Note that an extra parameter, Ψ, was included to ensure that the electrostatic force only acts on the electrode area.

Assuming that the static portion of the displacement and voltage is much larger than the dynamic portion, the electrostatic pressure can be decomposed into a 1st order Taylor series expansion:

$$P_{es}(x,y,t) \approx P_{es}(u_0,V_0) + \frac{2P_{es}(u_0,V_0)}{V_0}\tilde{V}e^{j\omega t} + \left.\frac{\partial P_{es}}{\partial u}\right|_{u=u_0,V=V_0}\tilde{u}e^{j\omega t} \quad \text{if} \quad u_0 \gg |\tilde{u}|, \; V_0 \gg |\tilde{V}| \tag{56}$$

3.3.5.1. Static Electrostatic Pressure

The 3 terms of the Taylor series expansion each have a special meaning. The first term is the most obvious, as it corresponds to the static electrostatic pressure being applied by the DC bias.

$$P_{es,0} = -\frac{\varepsilon_0 V_0^2 \Psi(x,y)}{2\left(h_{eff} + u_0(x,y)\right)^2} \tag{57}$$

3.3.5.2. Transformer Ratio

The second term represents the electromechanical transformer ratio, n, multiplied with the AC voltage component:

$$n = \frac{2P_{es,0}}{V_0} \tag{58}$$

In other words, it is the mechanical pressure exerted by AC electrical signal.

3.3.5.3. Spring Softening

The final term represents the well-known spring softening effect. As the CMUT moves closer to collapse, the displacement derivative of the pressure will increase. Consequently, this term will continue to increase and effectively soften the mechanical stiffness of the membrane:

$$K_{ss} = \frac{\partial P_{es}}{\partial u}\Big|_{u=u_0, V=V_0} \approx \frac{(P_{es}(u_0 + \Delta u, V_0) - P_{es,0})}{\Delta u} \tag{59}$$

Once again, these pressures must be put in matrix form to be compatible with the finite difference approach:

$$P_{es}(\{x^i\},\{y^i\},t) = \{P^i_{es,0}\} + \{n^i\}\tilde{V}e^{j\omega t} + [K^{ii}_{ss}]\{\tilde{u}^i\}e^{j\omega t}, \;\; i,j = 1,2,\dots N^*$$

$$\{n^i\} = \begin{Bmatrix} n^1 \\ n^2 \\ \vdots \\ n^{N^*} \end{Bmatrix}$$

$$[K^{ii}_{ss}] = \begin{bmatrix} K^{11}_{ss}(u^1_0, V_0) & 0 & \cdots & 0 \\ 0 & K^{ii}_{ss}(u^2_0, V_0) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & K^{N^*N^*}_{ss}(u^{N^*}_0, V_0) \end{bmatrix} \tag{60}$$

Note that {n} is a column vector but [K_{ss}] is a diagonal matrix in which each of the diagonal elements contain the respective spring softening values at each node.

3.3.6. Radiated Acoustic Pressure

The pressure that a single membrane exerts on the surrounding fluid, and vice versa, will be referred to as the "self-radiation" pressure. This is in contrast to the "mutual-radiation" pressure, described in the next section, which originates from fluid coupling with other membranes in the array. The total dynamic fluid pressure on the acting on the membrane will be the sum of these two pressures:

$$P_r = P_{r,self} + P_{r,mut} \tag{61}$$

For calculating both pressures, a very efficient solution was proposed by Meynier et al. [1], which relies on a Boundary Element Matrix (BEM) based on the Green's function of a baffled radiator in a semi-infinite fluid. According to Pierce [36], the Rayleigh integral derived from the Green's function of a baffled point source is:

$$P_r(r,\omega) = \frac{j\rho_{fl}\omega}{2\pi}\iint \dot{u}(x_s,y_s)\frac{e^{-jrk}}{r}dx_s dy_s$$
$$r = \sqrt{(x_s-x)^2+(y_s-y)^2+(z_s-z)^2} \quad (62)$$

Where P_r is the pressure at a point in space located at (x, y, z), generated by a baffled point source located at (x_s, y_s, z_s). $\dot{u}$ is the dynamic velocity of the point source vibrating at frequency ω. If we integrate over an elementary surface S in the fluid surrounding the source node, and make this surface sufficiently small such that the velocity can be assumed uniform within it, the integral is reduced to:

$$P_r(r,\omega) = \frac{j\rho_{fl}\omega S}{2\pi}\frac{e^{-jrk}}{r}\dot{u}(x_s,y_s) \quad (63)$$

Because a Green's function already accounts for all of the boundary conditions within a given environment, there is no need to mesh the entire fluid space. If the velocity of one node on the CMUT array is known, then the pressure due to that node at any other point in space can be accurately calculated, given the operating frequency, ω, and the distance, r, to the point of interest. Consequently, only the moving surfaces of the CMUT array need to be meshed. This allows for a significant reduction in computation time compared to that of a 3D FE model. Note that if determining the dynamic behavior of the CMUT in vacuum, these self- and mutual- radiation terms will all be zero.

3.3.6.1. Self-Radiation

With this information, it becomes possible to obtain the self-radiation pressure generated in the fluid by a CMUT membrane. Utilizing the mesh that was created for the finite difference approximation, we can assume that each node on the CMUT membrane is now acting as a harmonically oscillating point source. For a sufficiently fine mesh, this is a valid assumption as the variation in velocity surrounding a single node should be

negligible. If we consider the pressure on node "i" due to node "j," as demonstrated in Figure 22, we can rewrite (63) as:

$$P_r(r_{ij},\omega) = \frac{j\rho_{fl}\omega S}{2\pi}\frac{e^{-jr_{ij}k}}{r_{ij}}\,\dot{u}(x_i,y_i) \qquad (64)$$

$$r_{ij} = \left|\vec{r_j} - \vec{r_i}\right|$$

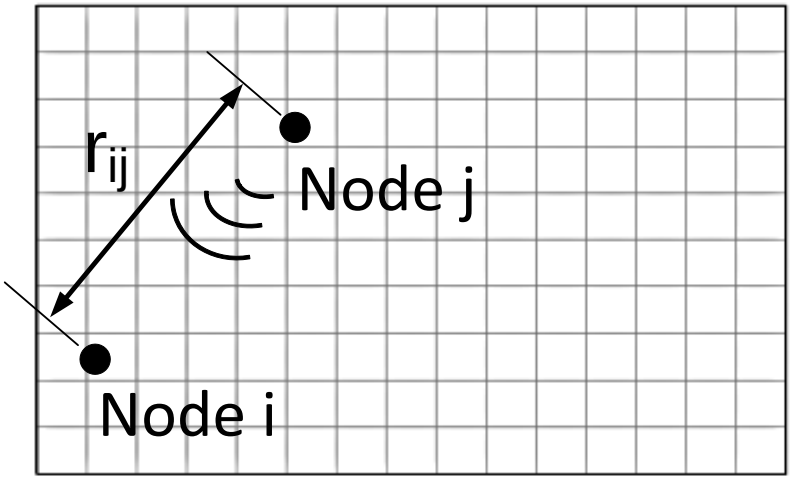


Figure 22. Illustration of nodal indices for self-radiation pressure calculations. Node j is the source and node i is a receiver.

Note that a single node will affect, and will be affected by, every other node in the membrane. The total pressure generated at a particular node will be the sum of the pressures generated by itself and all other nodes in the mesh. In this manner, a system of equations can be constructed that define the pressure at each node. Each equation will be a linear sum of the pressure contributions from every node in the mesh, including itself. This system can be written in matrix form as:

$$P_{r,self} = j\omega\left[Z_{r,self}^{ij}\right]\{\tilde{u}\}e^{j\omega t}, \qquad i,j = 1,2 \ldots N^*$$

$$\left[Z_{r,self}^{ij}\right] = \frac{j\rho_{fl}\omega S}{2\pi}\frac{e^{-jr_{ij}k}}{r_{ij}} = \begin{bmatrix} Z_{r,self}^{11} & Z_{r,self}^{12} & \cdots & Z_{r,self}^{1N^*} \\ Z_{r,self}^{21} & Z_{r,self}^{22} & \cdots & Z_{r,self}^{2N^*} \\ \vdots & \vdots & \ddots & \vdots \\ Z_{r,self}^{N^*1} & Z_{r,self}^{N^*2} & \cdots & Z_{r,self}^{N^*N^*} \end{bmatrix} \tag{65}$$

Where $[Z_{r,self}^{ij}]$ is the Boundary Element Matrix (BEM) describing the self-radiation impedance of the membrane at each node. Unlike the mechanical stiffness matrix, $[Z_{r,self}^{ij}]$ is fully populated due to the fact that every node affects every other node. For the effect that a node has on itself, the distance $r_{ii} = 0$. This would result in an unrealistic infinite impedance. Thus, the node can instead be assumed to act as an infinitesimally small piston, with effective radius, a_{eff}, such that:

$$Z_{r,self}^{ii} = \rho_{fl}c_{fl}\left(\frac{1}{2}\left(ka_{eff}\right)^2 + j\frac{8}{3\pi}\left(ka_{eff}\right)\right) \tag{66}$$

At this point, all parameters affect the dynamic behavior of a single CMUT membrane have been identified and accounted for. The final matrix form of the force balance equation for a single CMUT membrane is:

$$\begin{aligned} &\left\{P_{es,0}^{i}\right\} + \left\{n^i\right\}\tilde{V}e^{j\omega t} \\ &\qquad = \left[K^{ij}\right]\left\{u_0^i\right\} + \left\{P_a^i\right\} \\ &\qquad + \left[\left[K^{ij}\right] - \left[K_{ss}^{ii}\right] - \omega^2\left[M^{ii}\right] + j\omega\left[\zeta^{ii}\right] + j\omega\left[Z_{r,self}^{ij}\right]\right]\left\{\tilde{u}^i\right\}e^{j\omega t} \end{aligned} \tag{67}$$

$i = 1,2, \ldots N^*$

Note how the mechanical stiffness has been split into its static and dynamic components. It is also interesting to observe that the spring softening matrix now clearly decreases the mechanical stiffness matrix with increased bias.

3.3.6.2. Mutual Radiation

The pressure due to mutual radiation effects is calculated in a manner identical to the self-radiation pressure. Only now, we are interested in the pressure on node i of a membrane, p, due to the dynamic displacement of node j, on membrane q (as shown in Figure 23). In this case, we will need to generate 2 impedance matrices for every source-receiver membrane pair, p-q: One for membrane p's effect on q, and one for membrane q's effect on p. The pressure generated by membrane q on p is:

$$P_{r,mut}^{pq} = j\omega\left[Z_{r,mut}^{ij}\right]_{\mathrm{pq}}\left\{\tilde{u}^{i}\right\}_{p}e^{j\omega t}, \quad \begin{matrix} p,q = 1,2..M \\ i,j = 1,2,\ldots N^{*} \end{matrix} \tag{68}$$

Where M is the total number of membranes in the array. The BEM for mutual impedance between two membranes p and q can be written as:

$$\left[Z_{r,mut}^{ij}\right]_{\mathrm{pq}} = \frac{j\rho\omega S}{2\pi}\frac{e^{-j\mathrm{r}_{ij}^{pq}k}}{\mathrm{r}_{ij}^{pq}} = \begin{bmatrix} Z_{r,mut}^{11} & Z_{r,mut}^{12} & \cdots & Z_{r,mut}^{1N^{*}} \\ Z_{r,mut}^{21} & Z_{r,mut}^{22} & \cdots & Z_{r,mut}^{2N^{*}} \\ \vdots & \vdots & \ddots & \vdots \\ Z_{r,mut}^{N^{*}1} & Z_{r,mut}^{N^{*}2} & \cdots & Z_{r,mut}^{N^{*}N^{*}} \end{bmatrix}_{pq} \tag{69}$$

$$\mathrm{r}_{ij}^{pq} = \left|\overrightarrow{\mathrm{r}_{\iota}^{p}} - \overrightarrow{\mathrm{r}_{\jmath}^{q}}\right|$$

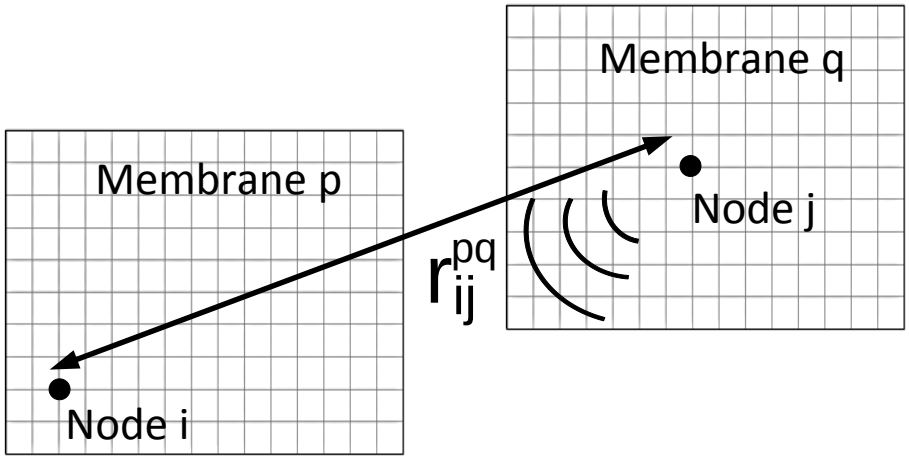


Figure 23. Illustration of nodal indices for mutual radiation pressure calculations. Node j is the source on membrane q, and node i is a receiver on membrane p.

Calculation of the mutual impedance terms requires that we mesh all membranes in the array. Thus, if we are given an array of M identical membranes meshed with N nodes (N^* after removing clamped nodes), then we will need to generate:

$$\#of\ impedance\ matrices = \quad {}^{M}P_2 = \frac{M!}{(M-2)!} \tag{70}$$

different mutual impedance matrices, all of which have the dimensions N^*xN^*. However, the problem size can be reduced significantly by realizing the law of reciprocity between source-receiver pairs. That is:

$$\left[Z_{r,mut}^{ij}\right]_{\mathrm{pq}} = \left[Z_{r,mut}^{ij}\right]_{qp}^{T} \tag{71}$$

Where $[A]^T$ is the array transpose of [A]. Thus, only one impedance matrix needs to be calculated for every membrane pair.

At this point, a change in the scale of the force balance matrix equation is necessary. Now that more membranes are involved, we can no longer use the displacement vector defined in (45) which only accounted for nodes in a single membrane. We must redefine the displacement vector to contain all nodes in the entire meshed array:

$$\left\{u_p^i\right\} = \left\{u_{0,p}^i\right\} + \left\{\tilde{u}_p^i\right\}e^{j\omega t} = \begin{Bmatrix} u_1^1 \\ u_1^2 \\ \vdots \\ u_M^{N^*} \end{Bmatrix} \tag{72}$$

With the expanded displacement vector, we also need to reconstruct all of the N^*xN^* matrices defined earlier. The new system stiffness matrices should increase in size to MN^*xMN^*. Each new matrix will take the generic form:

$$\left[A_{pq}^{ij}\right] = \begin{bmatrix} \left[A^{ij}\right]_{11} & \left[A^{ij}\right]_{12} & \cdots & \left[A^{ij}\right]_{1M} \\ \left[A^{ij}\right]_{21} & \left[A^{ij}\right]_{22} & \cdots & \left[A^{ij}\right]_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ \left[A^{ij}\right]_{M1} & \left[A^{ij}\right]_{M2} & \cdots & \left[A^{ij}\right]_{MM} \end{bmatrix}, \qquad i,j = 1,2,\dots N^*$$

$$where, \tag{73}$$

$$\left[A^{ij}\right]_{pq} = \begin{bmatrix} A^{11} & A^{12} & \cdots & A^{1N^*} \\ A^{21} & A^{22} & \cdots & A^{2N^*} \\ \vdots & \vdots & \ddots & \vdots \\ A^{N^*1} & A^{N^*2} & \cdots & A^{N^*N^*} \end{bmatrix}_{pq}, \quad p,q = 1,2,\dots M$$

Where each inner matrix, $[A^{ij}]_{pq}$, of the large system matrix, $\left[A_{pq}^{ij}\right]$, will contain information specific to the interactions between a single pair of membranes. $[A^{ij}]_{21}$, for example, will represent the effects that the nodes in membrane 1 have on membrane 2, regarding mechanical stiffness, radiation impedance, etc. $[A^{ij}]_{11}$ refers to the effects that nodes internal to membrane 1 have on each other. Note that the only matrices that involve inter-membrane effects come from the mutual impedance terms. All other matrices (mechanical stiffness, self-radiation impedance, electrostatic, etc.) have no relationship to nodes in neighboring membranes. Thus, it becomes clear that in the generic equation above:

$$\left[A^{ij}\right]_{pq} = \left[Z_{r,mut}^{ij}\right]_{\mathrm{pq}}, \qquad i \neq j \tag{74}$$

Similarly, since we know that $\left[A^{ij}\right]_{pp}$ represents the internal effects between nodes of the same membrane, that all other matrices excluding mutual impedance will occupy these positions in the generic equation. In fact, if we assume that every membrane in the array is identical in size, shape, and bias (valid for most CMUT arrays), and meshed identically, then:

$$\left[A^{ij}\right]_{11} = \left[A^{ij}\right]_{22} = \left[A^{ij}\right]_{MM} \tag{75}$$

Where $\left[A^{ij}\right]_{pp}$ is equal to the previously defined matrices for the single membrane case. From this new information, we can now reconstruct the final force balance equation for an array of identical membranes as:

$$\begin{aligned}\left\{P^i_{es,0,p}\right\} + \left\{n^i_p\right\}\tilde{V}e^{j\omega t} &= \left[K^{ij}\right]\left\{u^i_{0,p}\right\} + \left\{P^i_{a,p}\right\} \\ &+ \left[\left[K^{ij}_{pp}\right] - \left[K^{ii}_{ss,pp}\right] - \omega^2\left[M^{ii}_{pp}\right] + j\omega\left[\zeta^{ii}_{pp}\right] + j\omega\left[Z^{ij}_{r,self,pp}\right]\right. \\ &\left. + j\omega[Z^{ij}_{r,mut,pq}]\right]\left\{\tilde{u}^i_p\right\}e^{j\omega t}\end{aligned} \tag{76}$$

$i, j = 1,2, \ldots N^*$

$p, q = 1,2, \ldots M$

Where the generic forms of self-effects and mutual effects system matrices are, respectively:

$$\left[A^{ij}_{pp}\right] = \begin{bmatrix}\left[A^{ij}\right]_{11} & 0 & \cdots & 0 \\ 0 & \left[A^{ij}\right]_{11} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \left[A^{ij}\right]_{11}\end{bmatrix}, \quad \begin{matrix} i, j = 1,2, \ldots N^* \\ p = 1,2, \ldots M \end{matrix} \tag{77}$$

$$[Z^{ij}_{r,mut,pq}] = \begin{bmatrix} 0 & \left[Z^{ij}_{r,mut}\right]_{12} & \cdots & \left[Z^{ij}_{r,mut}\right]_{1M} \\ \left[Z^{ij}_{r,mut}\right]^T_{12} & 0 & \cdots & \left[Z^{ij}_{r,mut}\right]_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ \left[Z^{ij}_{r,mut}\right]^T_{1M} & \left[Z^{ij}_{r,mut}\right]^T_{2M} & \cdots & 0 \end{bmatrix}, \quad i, j = 1,2, \ldots N^* \tag{78}$$

Notice how the definition of the mutual impedance system matrix takes advantage of the reciprocity relationship defined in equation (71).

It should be mentioned that when performing array calculations, the simplified version of the mechanical stiffness matrix that utilizes quarter symmetry is no longer valid. This is because that simplification assumes symmetric loading across the membrane. However, depending on the location of the membrane in an array, the

loading on that membrane will likely not be symmetric. This renders the ¼ symmetry stiffness matrices invalid.

On a similar note, though, symmetry in an array can be utilized to significantly decrease the problem size. If an array has ½ or ¼ symmetry, then there is redundancy in the displacement vectors being solved for. As a result, only the displacements of the partial array need to be solved for. This implies that the stiffness matrices will be of size M/2xM/2 or M/4xM/4. However, it does not imply that the mutual impedances from membranes not in the section of interest can be neglected. The final mutual impedance for the partial array must account for the effects of all membranes in the array. Fortunately, the effects that the membranes in the partial array have on the other membranes outside of the partial array do not need to be calculated. This is because those exterior membranes are forced to have the same velocity as the interior membranes by reducing the displacement vector size, making it unnecessary to calculate their corresponding stiffness matrices.

3.4. Solving the Static Analysis

Now that the force balance equation for a CMUT array has been fully defined, it becomes possible to solve for the system behavior. The first step is to solve for the static, biased solution of the CMUT. We only need to consider a single membrane, since the presence of neighboring membranes will have a negligible effect on the static displacement. If we remove all time dependent parameters from the force balance equation in (76), then we are left with:

$$\begin{aligned} \left\{P_{es,0}^{i}\right\} &= \left[K^{ij}\right]\left\{u_0^i\right\} + \left\{P_a^i\right\} \\ -\frac{\varepsilon_0 \mathrm{V}_0^2\{\Psi^{\mathrm{i}}\}}{2\left(h_{eff} + \left\{u_0^i\right\}\right)^2} &= \left[K^{ij}\right]\left\{u_0^i\right\} + \left\{P_a^i\right\} \end{aligned} \tag{79}$$

This is a non-linear, implicit equation, and thus cannot be solved directly. Instead, the problem can be solved using an iterative approach, as demonstrated in [32].

In this approach, a guess solution (e.g. null vector) for the static displacement of the membrane is substituted into the electrostatic pressure equation. The corresponding electrostatic pressure is then used to estimate a new displacement vector by substituting it into equation (79). This new displacement vector is then substituted into the electrostatic force equation, and this process is continued until the displacement of the membrane between iterations converges to a chosen tolerance. The displacement from the final solution becomes the accepted static displacement, $\{u_0^i\}$, used for future calculations. The process is as follows:

$$
\begin{aligned}
&Guess\ displacement\ vector = \left\{u_0^i\right\}^0 \\
&\left\{P_{es,0}^i\right\}^0 = -\frac{\varepsilon_0 \mathrm{V}_0^2\{\Psi^{\mathrm{i}}\}}{2\left(h_{eff} + \left\{u_0^i\right\}^0\right)^2} \\
&\left\{u_0^i\right\}^1 = \left[K^{ij}\right]^{-1}\left[\left\{P_{es,0}^i\right\}^0 - \left\{P_a^i\right\}\right] \\
&\left\{P_{es,0}^i\right\}^1 = -\frac{\varepsilon_0 \mathrm{V}_0^2\{\Psi^{\mathrm{i}}\}}{2\left(h_{eff} + \left\{u_0^i\right\}^1\right)^2} \\
&\left\{u_0^i\right\}^2 = \left[K^{ij}\right]^{-1}\left[\left\{P_{es,0}^i\right\}^1 - \left\{P_a^i\right\}\right] \\
&\qquad\qquad \vdots \\
&\left\{P_{es,0}^i\right\}^{n-1} = -\frac{\varepsilon_0 \mathrm{V}_0^2\{\Psi^{\mathrm{i}}\}}{2\left(h_{eff} + \left\{u_0^i\right\}^{n-1}\right)^2} \\
&\left\{u_0^i\right\}^n = \left[K^{ij}\right]^{-1}\left[\left\{P_{es,0}^i\right\}^{n-1} - \left\{P_a^i\right\}\right] = \{u_0^i\}
\end{aligned}
\tag{80}
$$

This method can be used to determine the collapse voltage of the CMUT as the solution will fail to converge if the bias is above the collapse voltage.

Once the static solution has been obtained, a number of parameters for the CMUT can be calculated. The static capacitance of a single membrane can be calculated as:

$$C_0 = \frac{\varepsilon_0\{\Psi^i\}}{2\left(h_{eff} + \{u_0\}\right)^2} \tag{81}$$

Recall that Ψ is simply a vector of 1's and 0's denoting which nodes correspond to the electrode area. The clamped capacitance can also be calculated to include the effects of parasitic capacitances:

$$C_{clamp} = C_0 + C_P \tag{82}$$

The parasitic capacitance is usually estimated from experimental results. The k^2 coupling coefficient can also be determined from the capacitance method described in Chapter 2:

$$\begin{aligned} k^2 &= 1 - \frac{C_0}{C_T} \\ C_T &= \frac{dQ}{dV}\Big|_{V_0, z_0} \end{aligned} \tag{83}$$

This requires that a very small voltage, ΔV, be added to the bias voltage, and the static solution resolved. This new static displacement, $\{u_{0+\Delta V}\}$, can be used to calculate the free capacitance, C_T, as:

$$C_T = \frac{C_{0+\Delta V}\Delta V - C_0 V_0}{\Delta V} \tag{84}$$

3.5. Solving the Dynamic Analysis

Once the static solution has been obtained, the linear, dynamic response of the CMUT array can be determined. Considering now only the dynamic terms in the force balance equation, we have:

$$\begin{aligned} \{n_p^i\}\tilde{V} = \Big[&\left[K_{pp}^{ij}\right] - \left[K_{ss,pp}^{ii}\right] - \omega^2\left[M_{pp}^{ii}\right] + j\omega\left[\zeta_{pp}^{ii}\right] + j\omega\left[Z_{r,self,pp}^{ij}\right] \\ &+ j\omega[Z_{r,mut,pq}^{ij}]\Big]\{\tilde{u}_p^i\} \end{aligned} \tag{85}$$

Note that $\left[K_{ss}^{ii}\right]$ and $\{n_p^i\}$ can now be fully defined using the static solution. $\left[K_{ss}^{ii}\right]$ can be estimated by displacing the static solution slightly, and determining the rate of change of electrostatic force with displacement:

$$\{K_{ss}^{i}\} = \frac{(P_{es}(\{u_0^i\} + \Delta u, V_0) - \{P_{es,0}\})}{\Delta u} \tag{86}$$

This spring softening vector is then transformed into a diagonal matrix, as discussed earlier. The dynamic problem is linear and explicitly defined, so simple matrix operations can be used to solve for the displacement directly:

$$\{\tilde{u}_p^i\} = \left[\left[K_{pp}^{ij}\right] - \left[K_{ss,pp}^{ii}\right] - \omega^2\left[M_{pp}^{ii}\right] + j\omega\left[\zeta_{pp}^{ii}\right] + j\omega\left[Z_{r,self,pp}^{ij}\right] + j\omega[Z_{r,mut,pq}^{ij}]\right]^{-1}\{n_p^i\}\tilde{V} \tag{87}$$

With this single equation, the dynamic behavior of the entire CMUT array, with fluid-coupling effects can be calculated! Note that this equation must be solved at each frequency due to the ω-dependence. From this solution, the average velocity of the membrane array can be calculated to generate a frequency response graph for center frequency and bandwidth estimations.

3.6. Optional Additions to the Model

With the completion of the basic model described above, it becomes possible to add many other features to more closely simulate experimental results. The following is just a sample of ways in which the FD/BEM model can be expanded.

3.6.1. Impedance and Admittance

Electrical impedance is a common metric used in characterization and testing of CMUT elements. The mechanical impedance of a CMUT is defined as the ratio of the force applied to its membrane, divided by the corresponding average velocity:

$$Z_{mech} = \frac{F_{mech}}{\langle \dot{u} \rangle} \tag{88}$$

However, the electrical impedance measured across the terminals of the electrical mesh (see equivalent circuit model in Chapter 2) not only includes this mechanical impedance, but also the static capacitance and the parasitic capacitance. Thus, to model the true

electrical impedance of the CMUT, the mechanical impedance should be converted to the electrical mesh using the transformer ratio, and then added in parallel with the parasitic and static capacitance:

$$\frac{1}{Z_{elec}} = \frac{1}{Z_{Cp}} + \frac{1}{Z_{C0,tot}} + \frac{n^2}{Z_{mech}}$$

$$\therefore Z_{elec} = \frac{Z_{Cp} Z_{C0,tot} Z_{mech}}{Z_{mech} Z_{Cp} + Z_{mech} Z_{C0,tot} + Z_{C0,tot} Z_{Cp}} \tag{89}$$

$$Z_{C0,tot} = \frac{1}{j\omega C_{0,tot}}, \qquad Z_{Cp} = \frac{1}{j\omega C_p}$$

To predict this parameter using the model, the frequency response in the chosen environment must first be simulated. From this, the average velocity over the relevant membranes should be calculated and utilized to calculate mechanical impedance. Then, an average transformer ratio needs to be calculated over the membranes, and then the equation above can be used to determine impedance. Note that all of these impedances and the transformer ratio should only consider the total electrode area of the membranes that were part of the element being electrically measured, and not the area that includes surrounding membranes. The impedances for the static capacitance can be calculated for a single membrane, and then multiplied by the number of membranes in the measured element to obtain the total capacitance $Z_{C0,tot}$. The parasitic capacitance is difficult to measure directly, so it is often determined by matching a simulation to experimental results. The mechanical impedance should be calculated directly, however, since velocity will vary between membranes.

Electrical admittance is very closely related to impedance and can also be calculated with the model using:

$$Y_{elec} = Y_{Cp} + Y_{C0,tot} + n^2 Y_{mech}$$

$$Y_{elec} = j\omega C_p, \qquad Y_{C0} = j\omega C_{0,tot}, \qquad Y_{mech} = \frac{\langle \dot{u} \rangle}{F_{mech}} \tag{90}$$

3.6.2. Attenuation in Fluid Media

To include effects of attenuation in the fluid, the simplest solution is to add an exponential to the expressions for self- and mutual- impedance:

$$\left[Z_{r,self}^{ij}\right] = \frac{j\rho_{fl}\omega S}{2\pi}\frac{e^{-jr_{ij}k}}{r_{ij}}e^{-\alpha r_{ij}}$$

$$\left[Z_{r,mut}^{ij}\right]_{\mathrm{pq}} = \frac{j\rho\omega S}{2\pi}\frac{e^{-j\mathrm{r}_{ij}^{pq}k}}{\mathrm{r}_{ij}^{pq}}e^{-\alpha \mathrm{r}_{ij}^{pq}}, \quad \begin{matrix} i,j = 1,2,\dots N^{*} \\ p,q = 1,2,\dots M \end{matrix} \tag{91}$$

$$\alpha = A_{fl}f^{n_{fl}}$$

f is the frequency of operation, and A_{fl} and n_{fl} are parameters regarding the viscous damping properties of the particular fluid, and can be found in literature. Note that this frequency dependent model for viscous damping may not be valid for all media.

3.6.3. Calculation of Pressure Distribution at a Specified Distance from the Array

Perhaps one of the most important additions to the model is the ability to calculate the pressure at a plane in space using only the velocity distribution of the membranes. As would be expected, the average velocity or pressure at the surface of the membranes will often times not correspond to what is measured experimentally with a hydrophone or pulse-echo. Interference effects will shape the beamwidth of the array at each frequency. Thus, at each frequency, different amounts of energy will actually reach the receiver element depending on its size, and its angle and separation distance relative to the transmitter. Furthermore, absorption effects within the fluid can filter certain frequencies more than others, and typically will degrade the signal more significantly at higher frequencies. Additionally, it is possible that cross talk effects can cause certain frequencies to have large average surface velocities, but the velocity distribution is such that it will create an evanescent wave which will die off rapidly with distance from the array.

The above mentioned filtering effects of sound wave propagation in a fluid can be accounted for by once again using the Rayleigh integral derived from a baffled point source. Only now, instead of using it indirectly to calculate impedance, we will use it directly to determine the pressure at a specific location using the known velocity distribution on the array surface. This implies one additional step to the typical model. First the dynamic response of the array should be calculated. Then a planar surface at any location in space can be meshed with N_Z total nodes, and the pressure at each node in that mesh will be calculated in a similar manner to the radiation impedance terms. If we retain the same notation as before, we can write nodal pressures values due to the effects from each membrane as:

$$[\mathrm{P}^{\mathrm{li}}_{\mathrm{r,field}}]_p = \frac{j\rho\omega S}{2\pi}\frac{e^{-j\mathrm{r}^p_{li}k}}{\mathrm{r}^p_{li}}\{\dot{u}^i\}_p = \frac{-\rho\omega^2 S}{2\pi}\frac{e^{-j\mathrm{r}^p_{li}k}}{\mathrm{r}^p_{li}}\{\tilde{u}^i\}_p, \qquad \begin{matrix} i = 1,2,\dots N^* \\ p = 1,2,\dots M \\ l = 1,2,\dots N_Z \end{matrix}$$

$$[\mathrm{P}^{\mathrm{li}}_{\mathrm{r,field}}]_p = \begin{bmatrix} \mathrm{P}^{11}_{\mathrm{r,field}} & \mathrm{P}^{12}_{\mathrm{r,field}} & \cdots & \mathrm{P}^{1N^*}_{\mathrm{r,field}} \\ \mathrm{P}^{21}_{\mathrm{r,field}} & \mathrm{P}^{22}_{\mathrm{r,field}} & \cdots & \mathrm{P}^{2N^*}_{\mathrm{r,field}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{P}^{N_Z1}_{\mathrm{r,field}} & \mathrm{P}^{N_Z2}_{\mathrm{r,field}} & \cdots & \mathrm{P}^{N_ZN^*}_{\mathrm{r,field}} \end{bmatrix}_p \tag{92}$$

$$r^p_{li} = \left|\overrightarrow{r^p_j} - \overrightarrow{r^p_l}\right|$$

This will generate a set of M matrices, where each row corresponds to the pressure at a single node in the far-field planar mesh due to each node in membrane p. See Figure 24 for an illustration of the notation regarding this far-field calculation.

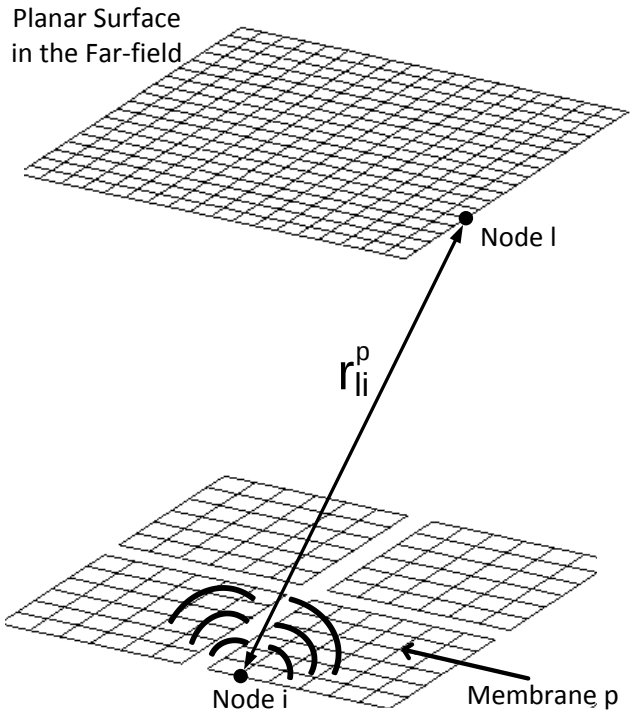


Figure 24. Illustration of the notation regarding the far-field pressure calculation

The actual pressure at each node in the planar mesh can be calculated by summing all of the matrices together, and then summing all elements in each row of that matrix. This will create a vector that is N_Z elements long, representing the total pressure at each node in the far-field plane:

$$[\mathrm{P}_{\mathrm{r,field}}^{\mathrm{li}}]_{tot} = \sum_{p=1}^{M} [\mathrm{P}_{\mathrm{r,field}}^{\mathrm{li}}]_{p}$$

$$\{\mathrm{P}_{\mathrm{r,field}}^{\mathrm{l}}\}_{tot} = \begin{Bmatrix} \sum_{i=1}^{N^*} \mathrm{P}_{\mathrm{r,field,tot}}^{\mathrm{1i}} \\ \sum_{i=1}^{N^*} \mathrm{P}_{\mathrm{r,field,tot}}^{\mathrm{2i}} \\ \vdots \\ \sum_{i=1}^{N^*} \mathrm{P}_{\mathrm{r,field,tot}}^{\mathrm{N_Z i}} \end{Bmatrix} \tag{93}$$

3.6.4. Accounting for Finite Bandwidth of Experimental Input Signals

In reality, the signal sent to the CMUT does not contain infinite bandwidth. However, the linear dynamic analysis used for this model calculates the frequency response by taking a constant AC voltage amplitude, and assuming that this same amplitude signal is transferred to the CMUT at all frequencies. In essence, it assumes that the signal sent to the CMUT has perfect bandwidth. Thus, the resultant frequency response only reflects the frequency response of the CMUT array. While this result is an important parameter to characterize, the actual signal transmitted and received by the CMUT should be modeled as well for accurate comparison with experiments.

Fortunately, the linear nature of the model makes this a very easy task to accomplish. For a given linear system, A(x), with a single input, B(x), there will be a corresponding output C(x). If we take the same linear system and input a different signal D(x), we will get a different output E(x). The relationship between this input/output and the previous input/output will be directly related to each other as:

$$\frac{B(x)}{C(x)} = \frac{D(x)}{E(x)}$$
$$\therefore D(x) = C(x) * \frac{D(x)}{B(x)} \tag{94}$$

Thus, we can take the output frequency response from the simulation (which assumed perfect bandwidth) and simply multiply it by the ratio between the frequency response of the actual input to the CMUT and the modeled input. This implies that any experimental input signal (e.g. pulse, toneburst, etc.) can be accounted for in the model by simply determining the frequency response of its temporal plot using a Fourier transform. The frequency response should then be divided by the constant magnitude frequency response of the model, and multiplied with the model's original output.

3.6.5. Inactive Membranes and Multiple Biases

In most CMUT arrays, all membranes are not actuated simultaneously. Rather, the arrays are typically divided into elements, each containing a set number of membranes, and then fired in sequence one at a time. As each one is transmitted, all surrounding elements will typically act as receivers for the reflected echo signals. These received signals can then be processed and constructed to form an image. An accurate model should simulate this exact scenario. The model, as defined earlier, is fully capable of performing such a task by modifying the electrostatic pressure term containing the transformer ratio. This term describes the mechanical pressure exerted by the AC signal. It is a vector containing the dynamic pressure for each membrane. Thus, the AC signal sent to each membrane can be turned on or off using a Boolean variable:

$$\begin{aligned} &\left\{\Upsilon_{\mathrm{p}} n_p^i\right\}\tilde{V}e^{j\omega t}, \qquad \begin{matrix} i = 1,2,\dots N^* \\ p = 1,2,\dots M \end{matrix} \\ &\Upsilon_{\mathrm{p}} = \begin{cases} 1 & membrane\ p\ is\ active \\ 0 & membrane\ p\ is\ inactive \end{cases} \end{aligned} \tag{95}$$

In this manner, the AC signal sent to each membrane can be adjusted.

This same logic can be applied to the model to allow for using different biases for different elements in the array. For instance, the dual ring FL-IVUS array has two separate rings that allow for separate Tx and Rx biases. This can be modeled by running the static simulation multiple times to solve for each biased state. This will allow the calculation of multiple transformer ratio vectors and spring softening matrices corresponding to each bias. These must then be placed in the appropriate positions within the large system matrix.

3.7. Model Limitations

For clarity, the assumptions in the model will be summarized here. The thin plate equations require that the total displacement of the membrane be small compared to its

thickness. It also requires that the lateral dimensions be large relative to the thickness of the membrane. For most CMUT membranes, these assumptions are valid. For lower aspect ratio membranes, a comparison with a 3D FEM would be recommended to verify accuracy. The stiffness equations used here assume no pre-stress exists in the membrane before biasing. The CMUTs used in the Degertekin research group have been optimized to have extremely low stress membrane material, so this assumption should be valid for our devices. Also, in order for the finite difference approach to be valid, the mesh must be relatively fine for accurate derivative approximations. This is a similar requirement for FE models.

It should also be noted that in using the thin plate approach, the model assumes that the membrane is rigidly clamped at the edge of its gap. However, actual CMUTs are clamped by the narrow regions of nitride in between membranes. This nitride is clamped to the substrate, but will allow some horizontal yielding. Thus, the thin plate assumption may add a slight amount of artificial stiffness to the membrane. This stiffening effect will be investigated in Chapter 5.

The electrostatic force, in assuming that it acts vertically on each node, neglects the effects of fringing fields. Considering the large surface area of the electrodes relative to the length of their perimeter, fringing effects are typically negligible for CMUT simulations. Nearly all of the electrostatic force is provided by the main surface of the electrodes.

Also, the Green's function derivation of the Rayleigh integral assumes that the nodes act as baffled point source radiators. However, in reality, the surface surrounding each node is not a truly infinite, rigid, flat baffle. The moving membranes introduce a small topography surface, decreasing its flatness. Furthermore, the surface is not a perfectly rigid reflector because the membranes and even the clamped nitride itself will absorb some energy from the fluid as they deflect under loading. Finally, the surface is not truly infinite either. In reality, the devices are cut from their silicon wafers and

contain edges that could create reflections at their edges that are not accounted for by the Green's function. This is even more of an issue when considering FL-IVUS dual-ring arrays, because their silicon edges will exist directly next to the CMUTs when they are in their final "donut" shape. In spite of these discrepancies, the Green's function BEM approach has been validated elsewhere [1]. Aside from contributions from the donut shape, the approach will be further validated in a subsequent chapter.

Of course, one of the most limiting assumptions comes from the linearized, dynamic analysis. In utilizing the Taylor series expansion of the electrostatic force, it was assumed that the dynamic displacements and voltages were much smaller in magnitude than their static counterparts. For receive operation, this is definitely a valid assumption as typically a large DC bias is used, and incoming pressure waves only generate very small signals. In transmit, depending on the mode of operation, this may not be a valid assumption because large voltages are sometimes needed to provide sufficient pressure output. Thus, this model will not account for any nonlinearity that would arise from dynamic operation. The first order approximation utilized in this model, however, should be sufficient for most design purposes.

CHAPTER 4

FABRICATION OF CMUT ARRAYS

In order to validate the model, experimental testing of actual CMUT arrays was necessary for direct comparison with simulation results. The following chapter describes the typical process steps involved with fabricating these arrays. It concludes by detailing the characteristics of the specific arrays that were used for model validation in the next chapter.

4.1. Standard CMUT Fabrication

CMUTs utilize microfabrication techniques that have been developed by the microprocessor industry. These processes allow for complex, densely populated arrays to be manufactured in batch quantities with high yield. Such capabilities allow CMUTs to easily adapt to the rigorous requirements of FL-IVUS applications described earlier. In recent years, there has been a significant fabrication effort to tightly integrate CMUTs with their receive circuitry to minimize parasitic capacitances and cable counts [37-42]. CMUTs are ideally suited for this type of integration since they are constructed with the same processes as their electronics. The following is a description of the fabrication process developed by the Degertekin research group at the Georgia Institute of Technology [43]. The tools used for this process are limited to what is available on-campus in the Microelectronics Research Center (MiRC). This process is CMOS compatible, with no process step requiring an operational temperature above 250°C.

A diagram showing each fabrication step is shown in Figure 25. Each step is described in detail in the following section, and numbered according to the figure shown. For more details regarding this process, please consult the cited reference.

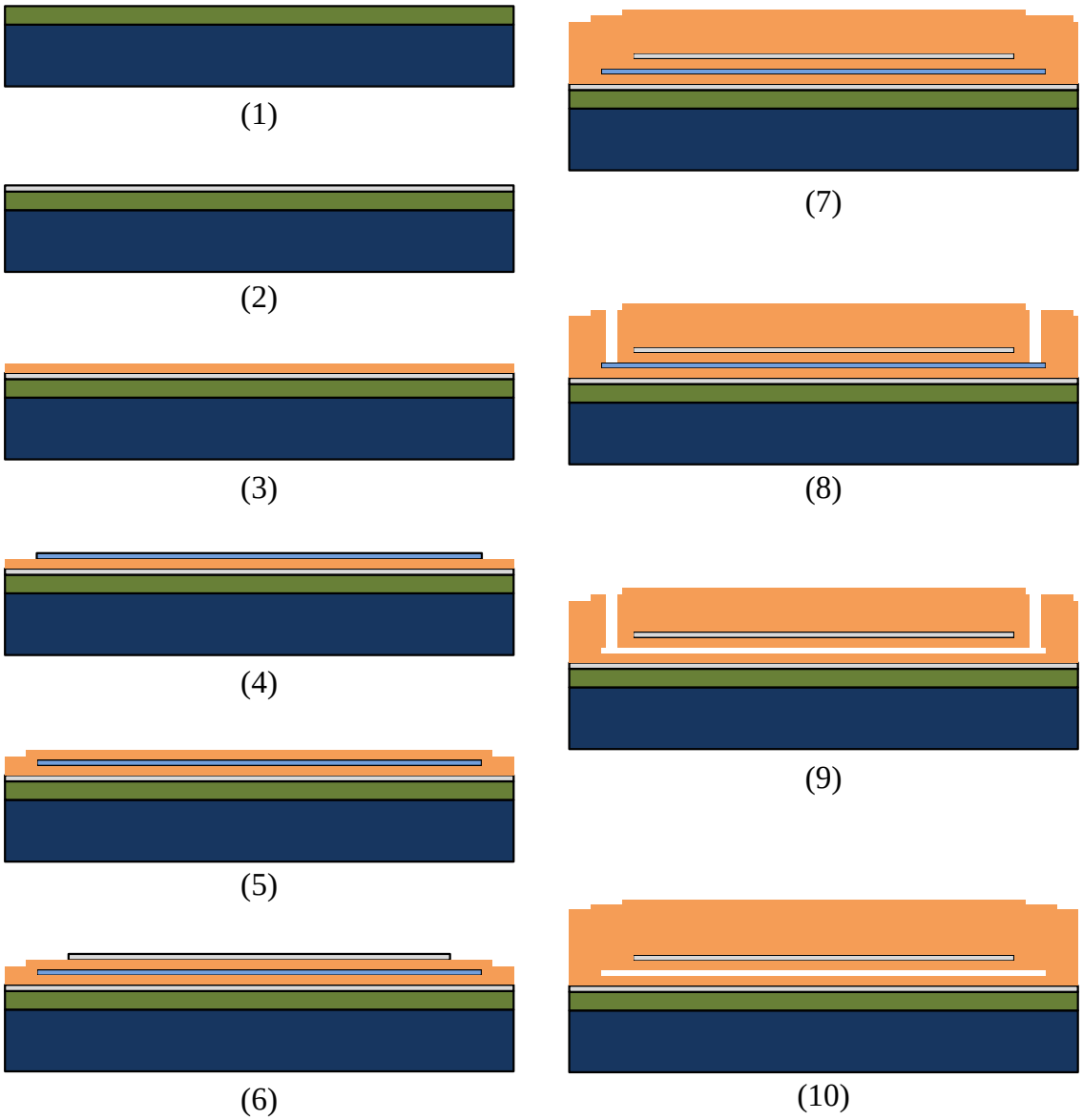


Figure 25. Diagram of the fabrication process flow used for low-temperature, CMOS-compatible CMUTs (not to scale)

4.1.1. Oxide Passivation (1)

The process begins by depositing a 3um thick layer of Silicon Dioxide onto a 500um thick Silicon substrate, using a Unaxis 250°C PECVD recipe. This layer electrically passivates the bottom electrode of the CMUT from the semi-conductive surface of the substrate. It must be thick enough to prevent any significant parasitic

capacitances from occurring between the bottom electrode of the CMUT and the bulk wafer.

4.1.2. Bottom Electrode (2)

An aluminum bottom electrode is deposited using a Unifilm DC sputtering tool and patterned. Its thickness is typically 0.12 um. This layer is then patterned using Shipley 1813 positive photoresist, and etched using Transene Aluminum Etchant Type A. This bottom electrode serves as the common electrode between all elements in a particular array. The silicon wafer could also be used as the common bias, but parasitic capacitances arising from overlapping connection lines degrade performance. With a patterned bottom electrode, the connection lines will not overlap with the biased top electrode.

4.1.3. Bottom Electrode Isolation (3)

A conformal Silicon Nitride layer is deposited over the bottom electrode using a 250°C Unaxis PECVD recipe. This layer separates the bottom electrode from the sacrificial layer to prevent it from being etched away during the release step (step (9)). Its thickness should be minimized as it contributes to the overall capacitance between the electrodes of the CMUT. This layer is approximately 0.2 um thick. PECVD nitride is typically not as dense as alternative deposition methods, so it can often suffer from pinholes. These pinholes must be alleviated to prevent chrome etchant from leaking through the isolation layer and etching away the bottom electrode during the sacrificial release step. This is done by interrupting the deposition twice to rotate the wafers, disrupting the pinhole formation process.

4.1.4. Sacrificial Layer (4)

A chrome sacrificial layer is deposited using a Unifilm DC sputtering tool and patterned. This layer defines the gap of the CMUT, and is typically 0.12 um thick. This layer is then patterned using Shipley 1813 positive photoresist, and etched using Cyantek Cr-7s chrome etchant. The lateral dimensions of this layer play a significant role in the overall stiffness of the finished CMUT. It is designed to be slightly smaller than the bottom electrode so that small misalignment errors will not change the dimensions of the layer.

To ensure a reliable, uniform release in step (9), the sacrificial layer for a single element in an array is often divided into multiple membranes. This implies the need for a thinner membrane than the larger single membrane element would have required to achieve a given center frequency.

4.1.5. Top Electrode Isolation (5)

A conformal Silicon Nitride layer is deposited over the sacrificial layer using a 250°C Unaxis PECVD recipe. This layer separates the top electrode from the sacrificial layer to prevent it from being etched away during the release step (step (9)). Its thickness should be minimized as it contributes to the overall capacitance between the electrodes of the CMUT. This layer is approximately 0.2 um thick. Once again the wafers are rotated twice during the deposition to prevent the formation of pinholes.

4.1.6. Top Electrode (6)

An aluminum top electrode is deposited using a Unifilm DC sputtering tool. Its thickness is typically 0.12 um. It is patterned using Shipley 1813 positive photoresist, and etched using Transene Aluminum Etchant Type A. The top electrode layer provides

individual biases to each element in an array, so every element has its own electrically isolated electrode.

As mentioned in step (4), individual elements are often divided up into multiple membranes to ensure reliable release. Thus, these membranes are connected together electrically using the top electrode layer to allow simultaneous actuation with a single voltage input. To minimize parasitic capacitances, the top electrodes of each membrane in an element are patterned separately with small connection lines in between, rather than using one large top electrode.

4.1.7. Membrane Formation (7)

A 1 um thick Silicon Nitride layer is deposited using the same low-temperature Unaxis PECVD recipe. This layer builds up the structure of the membrane so that it is stiff enough to prevent collapsing during the subsequent release step. It should be noted that this Unaxis recipe has been optimized to yield very low stress nitride so that the membrane stiffness is not affected significantly [43].

4.1.8. Sacrificial Etch Holes (8)

Small holes are etched through the nitride down to the sacrificial layer using a Vision RIE plasma. These holes are patterned using Shipley 1813 photoresist. This etch recipe utilizes Argon sputtering to ensure an anisotropic etch with vertical sidewalls. The sacrificial layer acts as an etch stop, preventing accidental etching of the bottom electrode isolation layer. The sacrificial layer has locations patterned into it that specifically provide space for these etch holes off to the side of each patterned sacrificial area rather than directly over the sacrificial layer. This prevents the membrane from being pinned down after the sealing step (step 10).

4.1.9. Sacrificial Release (9)

The entire wafer is submerged in a bath of room temperature CR-7s chrome etchant for at least 4 hours. The chrome etchant will gradually seep through the previously defined etch holes and etch away the sacrificial layer. The wafers are rinsed off periodically to check the progress of the etch step. Cr-7s also etches aluminum, so the bottom and top electrode isolation layers provide protection during this step to prevent them from being etched away. When the entire sacrificial layer has been removed, the wafer is removed from the etchant bath, and rinsed thoroughly under running DI water. The wafer is then placed in a bath of DI water for 5 minutes, followed by another 5 minute soak in a second bath of fresh DI water. These steps effectively rinse away the chrome etchant from the gap, and replace it with water. This 2-bath process is immediately followed by a second 2-bath process using isopropyl alcohol in place of DI water. The alcohol replaces the water in the membrane gap, and due to its high volatility will evaporate very quickly, evacuating all fluid from the gap. To speed the process up, the wafer is placed in an oven at 85°C. The temperature must not be set too high or the solvent will not be able to escape through the etch holes quickly enough, causing a pressure buildup within the gap and subsequent bursting of the membranes.

4.1.10. Membrane Sealing and Final Thickness Definition (10)

The newly released membranes are placed in the Unaxis PECVD for one last nitride deposition that will define the final thickness of the membrane. The final membrane thickness typically ranges from 2 - 3.5 um. The nitride will fill and seal off the sacrificial etch holes, effectively creating a vacuum gap in the CMUT.

4.1.11. Open Bond Pads (11)

All of the nitride deposition steps covered the entire wafer, including the bond pads which are used to access the electrodes of the CMUTs. The final step in the fabrication process (not shown in Figure 25) involves one last etching step in the Vision RIE to drill through these nitride layers down to the bottom and top electrode bond pads. The plasma etch with Argon is used once again to ensure vertical sidewalls. This step utilizes a slightly thicker Shipley 1827 photoresist because Shipley 1813 may not provide sufficient protection for this longer etch. Since the top electrode is higher than the bottom electrode, it will be slightly over etched during this step.

4.2. Post-processing of Fabricated CMUTs

After completion of cleanroom fabrication, the individual devices are diced out of the wafer using a dicing saw. The diced arrays are then fixed to a ceramic chip carrier or PCB using UV epoxy. A wirebonder is used to form electrical connection lines between the chip carrier and the device. For devices that will be tested in water, the associated wirebonded assemblies are coated in 3 um of Parylene-C using an SCS Parylene Labcoater to prevent shorting between electrical lines. It should be noted that parylene will slightly affect the membrane dynamics and should be accounted for in simulations.

4.3. CMUT on CMOS Fabrication

As mentioned earlier, CMUTs are ideally suited for tight integration with their control circuitry. The fabrication process described above was designed with this in mind, and is completely CMOS compatible due to the low processing temperatures. For devices intended for applications such as FL-IVUS, the CMUTs can be built directly on top of the CMOS chips to minimize parasitic capacitances. The circuitry includes trans-impedance amplifiers for the receive CMUT elements, high-voltage pulsers for the

transmitters, and multiplexers for switching between elements. These multiplexers allow for a significant reduction in the overall cable count. A CMP polishing and one additional patterning step are all that is required to electrically connect the CMUTs to the relevant CMOS connection pads. The following sections describe these extra required steps in more detail.

4.3.1. CMP Polishing of the Passivation Oxide

The control circuitry for the CMUTs is fabricated onto 200 mm Si wafers using a 0.35um CMOS TSMC process. Similar to the standard CMUT fabrication process, the CMOS wafers must have a PECVD oxide layer deposited over them. However, this layer serves a very important purpose in addition to providing passivation between the CMUT electrodes and the CMOS substrate. The CMOS wafers are designed and fabricated at a TSMC facility. When they are shipped back, they contain significant topography with features as large as 1 um in height that originate from underlying metal layers that were never polished. Fabricating CMUTs directly over the top of this sort of topography could at the very least result in significant non-uniformity between membranes in an array. It is more likely, however, that non-functioning membranes would result considering the fact that the 1 um surface topography is over five times as tall as the membrane gaps. Furthermore, electrical lines may suffer from open circuits if the conformal metal layers are not conformal enough to coat the highly sloped sidewalls of the topography features. Thus, to eliminate these effects, the oxide layer is polished using a commercial CMP process. To ensure that at least 3um of oxide remain on the surface after polishing, 4.5 um of oxide are deposited over the CMOS from the beginning.

4.3.2. CMOS Wafer Dicing and Backside Grinding

The CMOS wafers contain 48 repeated die, with each die containing the electronics for many different device designs. The cleanroom facilities at Georgia Tech

do not possess sufficient capabilities to process 200 mm samples. Thus, the wafers are diced by an outside vendor into 4 cm by 7 cm blocks containing 2x3 die each, as shown in Figure 26. Unfortunately, these rectangular blocks are not ideal for microfabrication because photoresist tends to pool at two opposing corners as it is spun on. These locations are denoted in Figure 26. This results in some non-functional devices in those areas as they are not fully defined in the subsequent lithography.

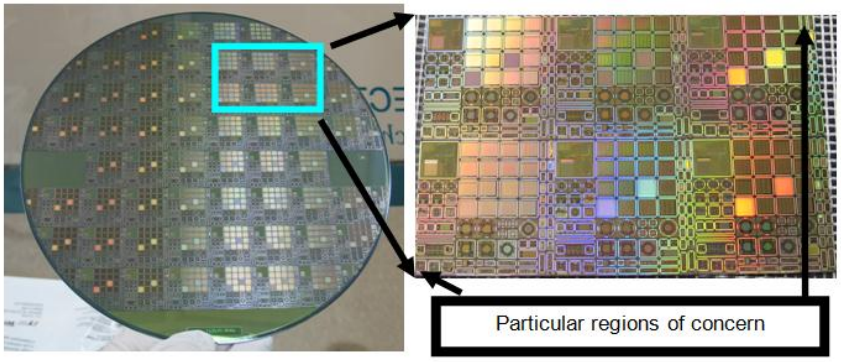


Figure 26. Photograph of the 8 inch CMOS wafer containing 48 repeated die (left) and the subsequently diced block containing 2x3 die (right).

Before being diced, the CMOS wafers go through a back-side grinding process that reduces their thickness from 550 um to 300 um. During CMUT operation, an undesired Lamb wave propagates through the bulk of the Silicon wafer and causes a sharp dip in the frequency response of the CMUTs at approximately 7.5 MHz for 550 um Silicon substrates [26]. The frequency at which this standing wave occurs is directly proportional to the thickness of the bulk substrate, and thus thinning the wafer pushes it to a higher frequency out of the band of interest.

4.3.3. Connection Etch Holes

After being polished, grinded, and diced, the CMOS wafers are ready for CMUT processing. The first and only patterning step required in addition to the standard CMUT fabrication steps involves etching small holes through the polished oxide to the

connection pads of the CMOS electronics. These holes are defined using Shipley 1827 photoresist. They are etched using an isotropic Vision RIE plasma recipe which has been optimized to provide highly sloped sidewalls [42], as shown in Figure 27. This allows the conformal metal layer of the subsequently deposited bottom electrode of the CMUT to fully coat the sidewalls of these vias, creating a reliable connection from the CMUT to the CMOS below. These connections must be placed off to the side of each element so that the CMUTs are not patterned over such large dips in the topography of the wafer.

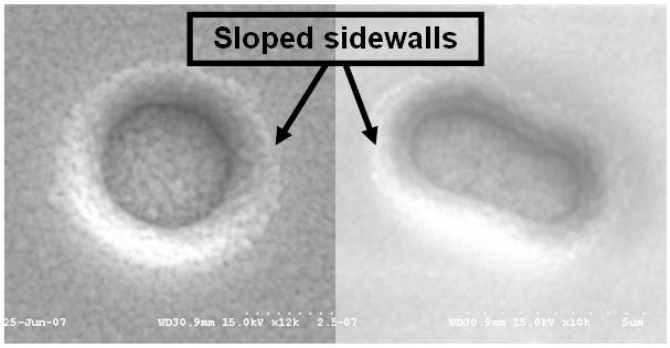


Figure 27. SEM pictures of the sloped sidewalls obtained from using the modified RIE etch recipe

After creating the connection vias, the standard CMUT fabrication process described earlier is carried out. The completed devices will contain CMUTs that are tightly integrated with their CMOS electronics. A cross-section of such a device is shown in Figure 28, demonstrating how the sloped vias are used to connect the bottom electrode of a CMUT element to the CMOS receive circuitry, and how wirebonds are used to connect to the few remaining externally accessible pads.

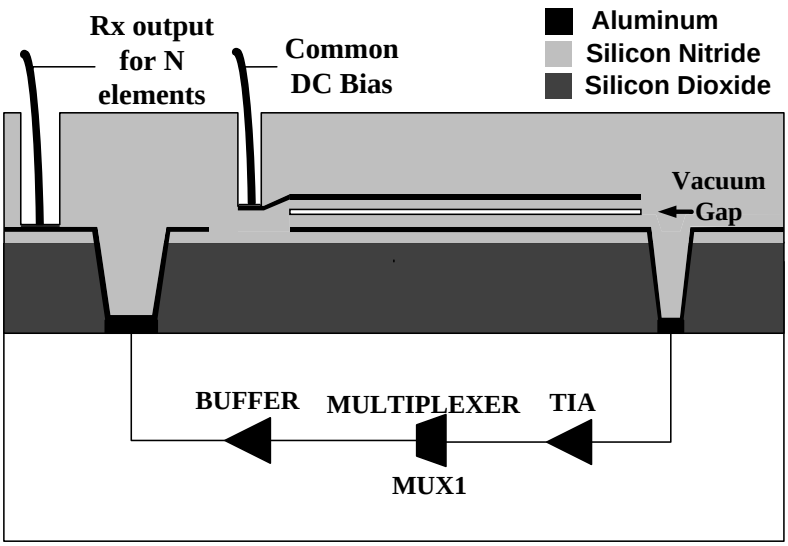


Figure 28. Cross-section of a CMUT element connected to receive circuitry using the CMUT on CMOS fabrication process

4.4. Fabricated CMUT Arrays Used For Model Testing

4.4.1. CMUT on CMOS FL-IVUS Dual Ring Array

An example of a completed FL-IVUS Dual Ring Array with fully integrated CMOS Tx and Rx electronics is shown in Figure 29. The array shown has a diameter of 2 mm, and contains 64 Tx elements and 54 Rx elements for a total of 480 individual membranes. Each membrane is a 35 um x 35 um square, with a 25 um x 25 um electrode. Note that due to the CMOS integration, not only will parasitics be minimized, but the number of required external connections has been reduced to thirteen. Also note that this is a device specifically designed for lab testing so it has not been fabricated into a true "donut" shape with the inner and outer portions of the ring removed. Future iterations of this device will include a final anisotropic through-wafer etching step that will completely remove the unwanted portions, as shown in the recently fabricated proof of concept device in Figure 30.

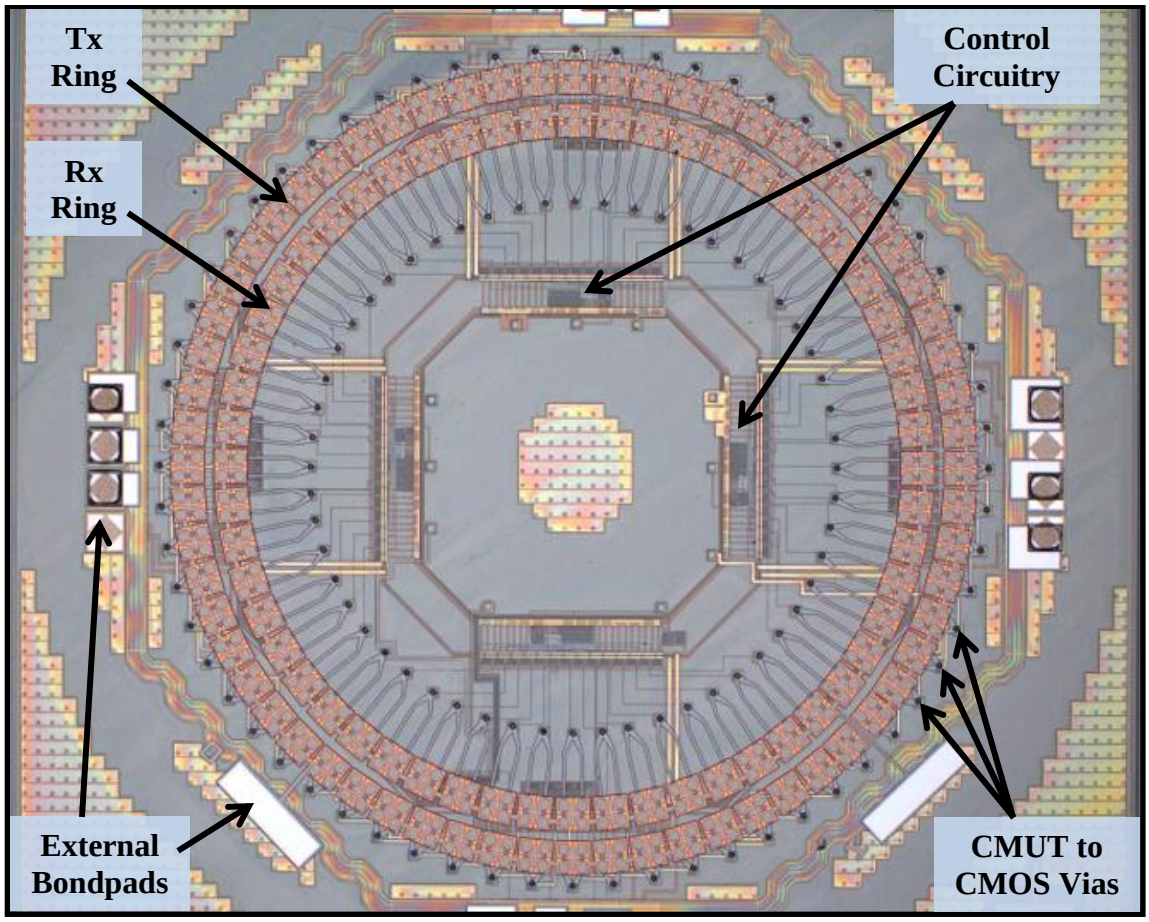


Figure 29. Fully fabricated FL-IVUS Dual Ring Array with CMUT on CMOS integration

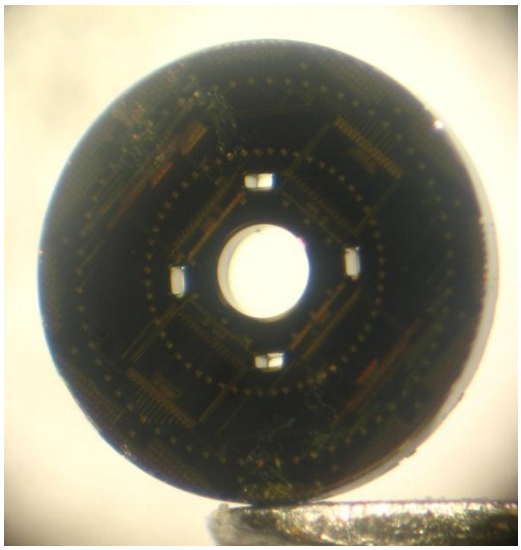


Figure 30. Proof of concept FL-IVUS CMOS chip with true annular shape (does not contain CMUT array)

4.4.2. Isolated Single Elements

Other test arrays, as shown in Figure 31, consist of a single, square element containing either 4 or 16 individual membranes. These test arrays can be used to validate the FD/BEM model's ability to predict the dynamic behavior of a single, isolated DRA element in immersion.

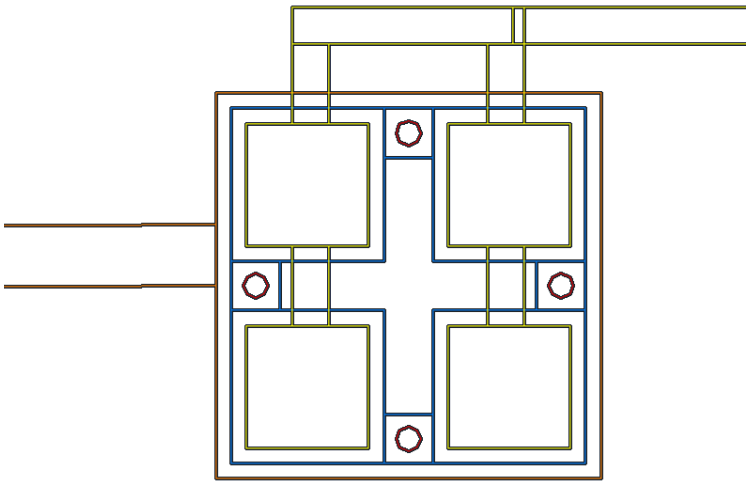


Figure 31. Illustration of single 4-membrane element array Note that only the outlines of the features are shown. Yellow – top electrode, blue – sacrificial layer, orange – bottom electrode, pink holes – sacrificial etch holes.

4.4.3. Large Linear Array with Isolated Center Element

A large linear array of 12x12 membranes was also fabricated. This array has a single, 16-membrane element at its center which has a separate signal line. This type of array will allow activation of a single element surrounded by many inactive (biased, but un-actuated) membranes, and vice versa. This design is illustrated in Figure 32. The FD/BEM model should be able to accurately predict the differences in the frequency responses between the isolated 16 membrane element described in the previous section,

and the un-isolated element in this array. Of course, this array can also be used in such a way that all membranes are pulsed simultaneously, providing an additional actuation scenario which can be used to validate the FD/BEM model.

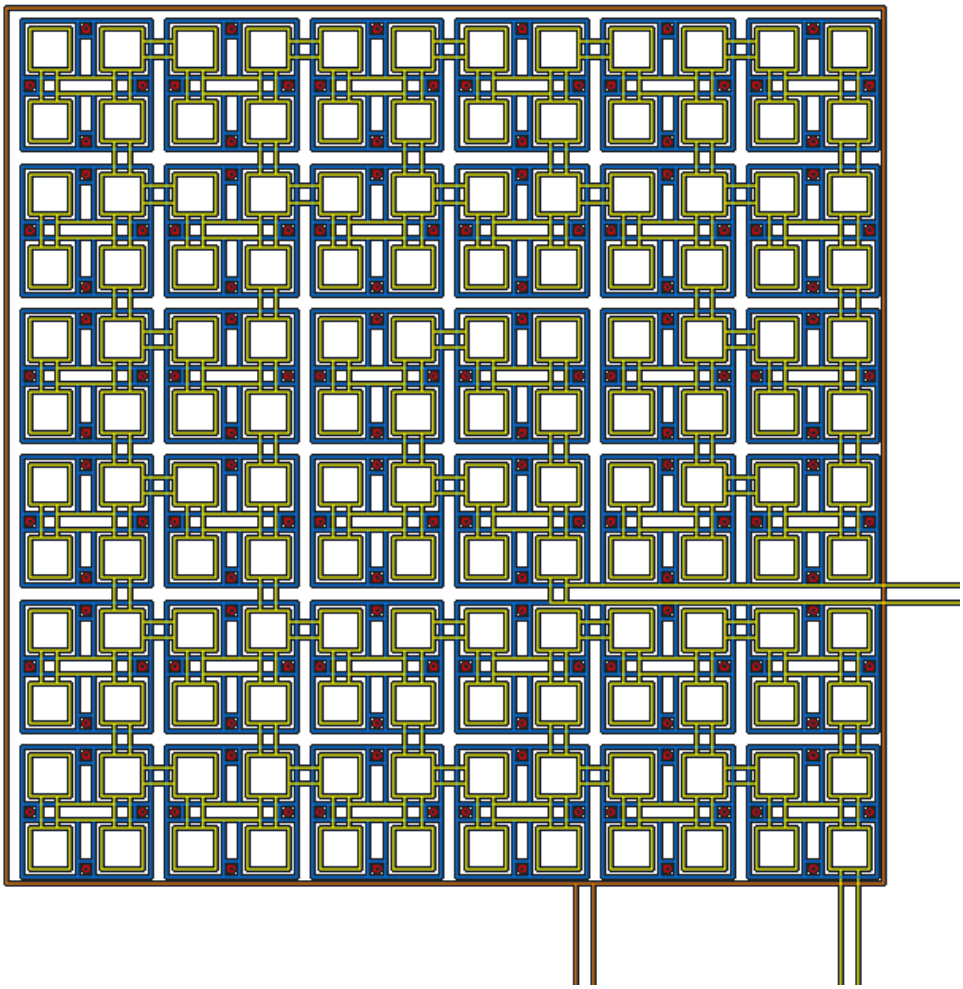


Figure 32. Illustration of linear array with isolated center element. Note that only the outlines of the features are shown. Yellow – top electrode, blue – sacrificial layer, orange – bottom electrode, pink holes – sacrificial etch holes.

CHAPTER 5

MODEL VALIDATION

In order to fully characterize the analytical model described in Chapter 3, comparison with both finite element models and experimental results is necessary. The comparisons with FEM will yield valuable information regarding how much accuracy is lost in using the thin plate and BEM simplifications. In this manner, the source of any discrepancies between experiment and the analytical model can be more specifically identified.

5.1. Comparison with FEM

A design space of 10 different membrane geometries was utilized for assessment of the model's capabilities. There are both square and rectangular membranes, with a range of aspect ratios to gain insight regarding the limits of the thin plate assumption. The geometry of the 10 membrane designs used for comparison between the FE and the FD/BEM model can be found in Table 1. Many of the chosen geometries are commonly used for most of the FL-IVUS devices manufactured by the Degertekin group. The material properties used for these simulations can be found in Table 2.

Table 1. Geometry of designs used for FEM and FD/BEM model comparisons

Design	h_{gap} (um)	h_{isol} (um)	Electrode Coverage (%)	Post Width* (um)	Lateral Dim. (umXum)	Membrane Thickness (um)	Aspect Ratio (Length/ Thick)
A	0.12	0	100	5	35x35	1.0	35.0
B	0.12	0	100	5	25x25	1.0	25.0
C	0.12	0	100	5	35x35	2.0	17.5
D	0.12	0	100	5	25x25	2.0	12.5
E	0.12	0	100	5	35x70	1.0	35.0
F	0.12	0.3	75	5	50x50	3.4	14.7
G	0.12	0.3	75	5	50x50	2.5	20.0
H	0.12	0.3	75	5	18x18	2.0	6.0
I	0.12	0.3	75	5	13x13	1.5	8.7
J	0.12	0.3	75	5	70x70	2.5	70.0

*only applies to FEM model

Table 2. Material properties used for FE and analytical models

Material	Property	Value	Units
Silicon Nitride	Young's Modulus	110	GPa
	Density	2040	kg/m^3
	Poisson's Ratio	0.22	~
	Viscous Damping Coefficient	0	N-s/m
Water	Speed of sound	1500	m/s
	Density	1000	kg/m^3
	Damping Ratio	0	dB/MHz/m

For these validation tests, parameters relevant to CMUT design such as collapse voltage and bandwidth were chosen as the comparison criteria. The results are summarized in the following sections. Note that all finite element models were created using Comsol 3.5a software.

5.1.1.1. Collapse Voltage

A comparison of collapse voltage calculations between the FE and FD/BEM approach will indicate the accuracy of the thin plate stiffness equations and the finite difference approach. The finite element model for determining collapse was modeled in 3D with the geometry shown in Figure 33. It utilized quarter symmetry, and applied a fixed boundary condition to the clamped bottom surfaces of the support posts. The vertical faces of the support posts were also given symmetry boundary conditions to model the effect of surrounding membranes. For all simulations, the bias was applied over the entire surface of the membrane (not including support posts). The electrostatic force is calculated in a manner very similar to the FD/BEM model. It applies a pressure to the bottom surface of the membrane which is a function of the vertical displacement. In essence, it neglects fringing fields and assumes that the electrostatic force acts only in the vertical direction. Previous testing has shown this to be an acceptable simplification. The model also neglects the mechanical properties of the embedded top electrode, assuming that the entire membrane thickness consist of silicon nitride. Since the electrode is typically $1/10^{th}$ to $1/20^{th}$ the thickness of the membrane, this is typically a very accurate simplification. This significantly decreases the required number of nodes in the finite element model by eliminating the fine geometrical features. An iterative solver was used to calculate the static displacement of the membrane. When the solver failed to converge, the collapse voltage had been exceeded.

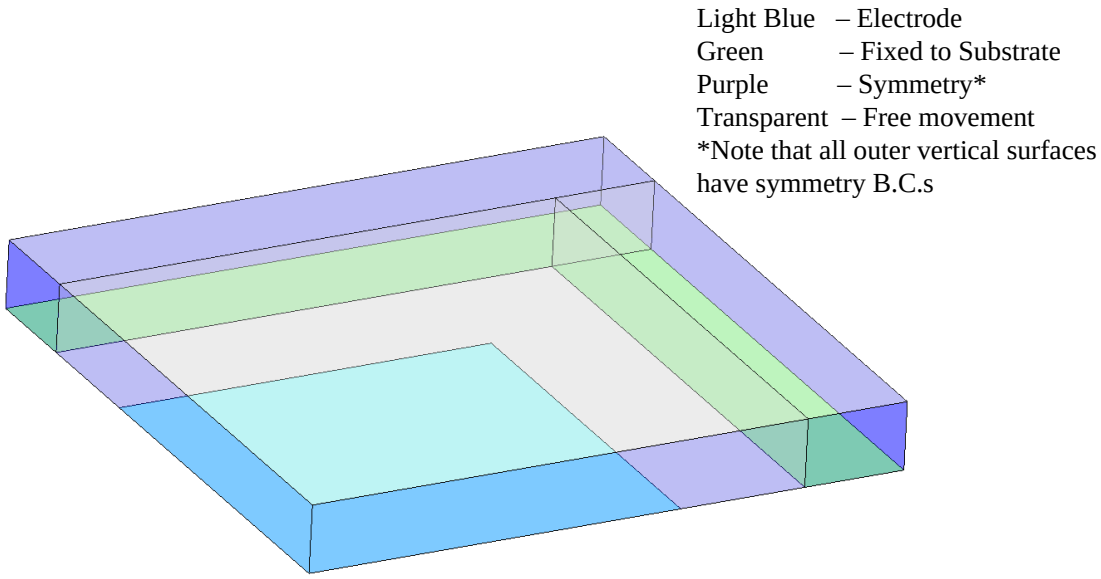


Figure 33. Geometry and boundary conditions used to calculate collapse with FEM

A summary of the results for all 10 designs can be found in Table 3. It is clear that the FD/BEM model is very accurate for determining the collapse voltage of CMUTs relative to the standard FEM method for a wide range of geometries. For most of the higher aspect ratio membranes, the two methods calculated identical estimates. Only for aspect ratios below 12.5:1 was the difference significant enough to be noteworthy. This implies that, as expected, very low aspect ratio membranes will be inaccurately stiffened using the thin-plate model.

Table 3. Results for collapse voltage calculated using FEM and FD/BEM approach

	Collapse Voltage (V)		
Design	**FEM**	**FD**	**%Diff**
A	20	20	0.00
B	39	39	0.00
C	54	55	1.85
D	101	108	6.93
E	13	13	0.00
F	97	100	3.09
G	63	63	0.00
H	305	349	14.43
I	378	434	14.81
J	33	33	0.00

5.1.1.2. Center Frequency in Vacuum

The center frequency of a single membrane in vacuum will yield more information regarding the accuracy of the thin plate assumption with respect to aspect ratio. Furthermore, the un-damped center frequency of a membrane is a very high-Q feature and thus gives a very precisely defined parameter for comparison between models. For the FE model, all of the same boundary conditions from the collapse voltage model were used. A linear frequency response was solved using a 1V AC signal (no static bias). The average velocity over the surface of the membrane was plotted at each frequency. The location of the 1st resonant peak in the graph was used as the metric for comparison between FE and the FD/BEM results. These results are summarized in Table 4.

Table 4. Results of FE and FD/BEM center frequencies for a single membrane in vacuum

	Center Freq (MHz)		
Design	**FEM**	**FD**	**%Diff**
A	9.8	9.4	3.67
B	18.8	18.3	2.66
C	18.5	18.8	1.57
D	34.5	36.8	6.67
E	6.7	6.6	0.90
F	15.0	15.6	4.07
G	11.4	11.5	0.88
H	61.5	71.0	15.45
I	87.5	102.2	16.80
J	6.0	5.9	2.17

Once again, the FD/BEM model was very accurate relative to FE calculations for most of the modeled geometries. As before, only the very low aspect ratio membranes had a noteworthy amount of error. Even for these lower aspect ratio membranes, the error in the center frequency estimate is within 10%, which is acceptable for many design applications.

5.1.1.3. Frequency Response in Water (Single Membrane)

For the remaining comparisons, the CMUT membrane behavior in immersion will be analyzed. Adding the fluid environment to the 3D FE model significantly increases the calculation time, particularly for higher frequency ranges. Consequently, the design space will be limited to only a portion of the geometries shown in Table 1.

For the FE model, a fluid space was added to the geometry used for calculating frequency response in vacuum, as shown in Figure 34. All of the same boundary conditions from the previous model were used, with one exception. With the addition of the fluid space, the fluid-structure interface needed to be defined on the top surface of the membrane. This was done by applying a pressure force on the structural elements equal

to the pressure in the fluid. Likewise, the accelerations of the fluid element nodes in contact with the membrane were set equal to the normal acceleration of the membrane. This created a 2-way coupling between the fluid and structural elements. An absorbing boundary condition was applied to the outer surface of the fluid space to model the semi-infinite nature of the fluid and to prevent reflections from creating undesired standing waves. The vertical faces of the fluid space utilized rigid wall boundary conditions to account for the quarter symmetry. The horizontal surface surrounding the CMUT also utilized a rigid wall to simulate a baffle. The fluid space was meshed with elements such that the highest frequency was resolved with at least 12 nodes per wavelength. A 1 Pa harmonic pressure was applied to the entire surface of the membrane. A linear frequency response was solved without electrostatically biasing the membrane.

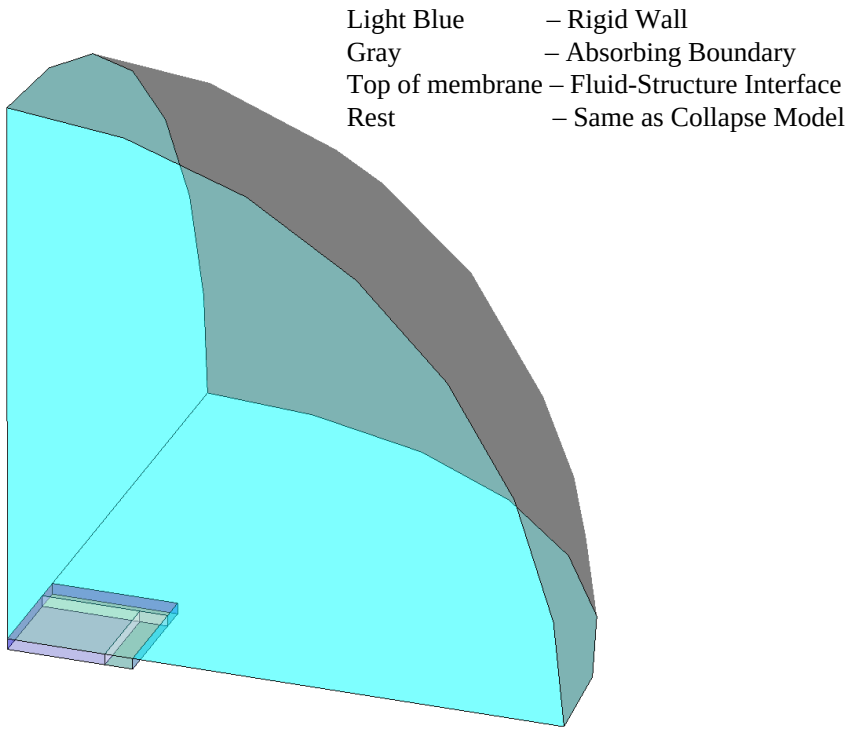


Figure 34. Geometry and boundary conditions used to calculate frequency response of single membrane in immersion with FEM

So that magnitudes could be compared between the two modeling methods, the dynamic electrostatic force vector of the pressure balance equation in the previous chapter was replaced by a uniform pressure vector of magnitude P_{dyn}. The pressure balance from Chapter 3 (Equation (87)) becomes:

$$\begin{aligned}\{\tilde{u}_p^i\} = \Big[\big[K_{pp}^{ij}\big] - \big[K_{ss}^{ii}\big] - \omega^2\big[M_{pp}^{ii}\big] + j\omega\big[\zeta_{pp}^{ii}\big] + j\omega\big[Z_{r,self,pp}^{ij}\big] \\ + j\omega[Z_{r,mut,pq}^{ij}]\Big]^{-1}\{P_{dyn,p}^i\} \\ i = 1,2,\dots N^* \\ p = 1 \\ \{P_{dyn,p}^i\} = \begin{Bmatrix}1\\1\\\vdots\\1\end{Bmatrix} P_{dyn}\end{aligned} \tag{96}$$

The velocity over the membrane surface was averaged and converted to dB for the frequency response plots. The center frequency and bandwidth of the first resonance, as well as the location of the 1st anti-resonance were used as metrics for comparison. The results from both simulation methods can be found in Table 5. As before, most modeled geometries showed close agreement between the FE and the FD/BEM model for all comparison metrics. Most importantly, the center frequency and bandwidth was very accurately predicted for all geometries except for Design D, which possessed the lowest aspect ratio. Even the results for this geometry would be sufficiently accurate for many design purposes.

Table 5. Results of FE and FD/BEM frequency sweeps for a single membrane in water

	Center Freq (MHz)			Bandwidth (%)			1st Antires. (MHz)		
Design	**FEM**	**FD/ BEM**	**%Diff**	**FEM**	**FD/ BEM**	**%Diff**	**FEM**	**FD/ BEM**	**%Diff**
A	3.6	3.6	0.00	11.6	11.2	3.45	16.9	16.8	0.59
B	8.0	8.1	1.25	15.0	16.1	7.33	34.8	35.4	1.72
C	9.1	9.5	4.40	23.2	24.0	3.45	34.8	37.4	7.47
D	19.4	21.4	10.31	19.4	21.4	10.31	-	-	-
E	2.2	2.2	0.00	9.7	9.2	5.15	5.5	5.2	4.04

A sample plot of the frequency responses calculated for Design A can be found in Figure 35. Note the close agreement between the 2 modeling methods across the entire frequency spectrum. As well, the magnitudes are essentially identical at all frequencies.

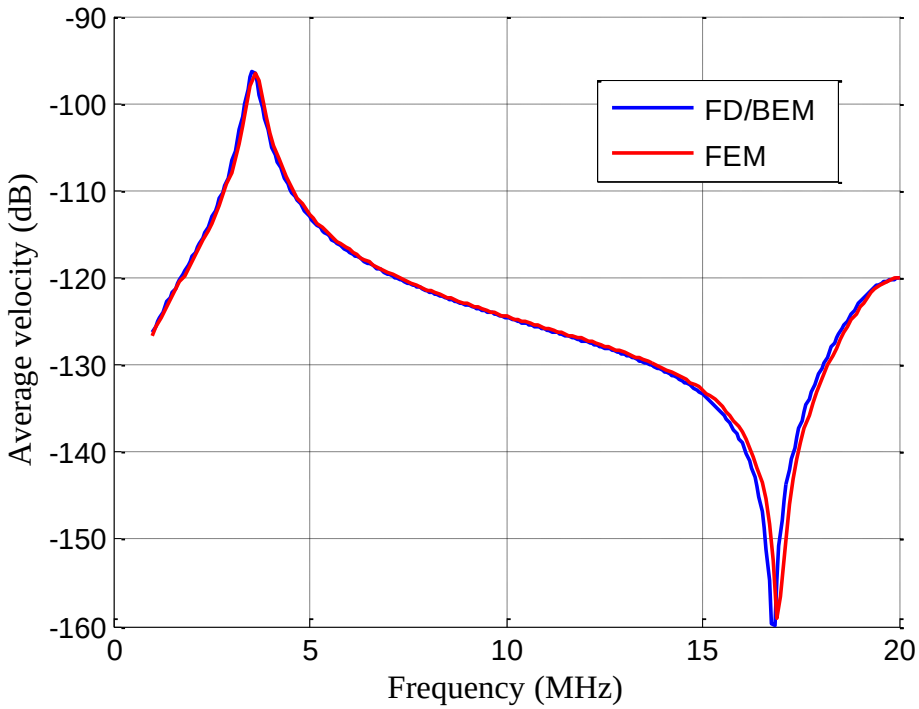


Figure 35. Frequency responses from FEM and FD/BEM model for a single membrane in water (Design A)

It should be pointed out that for lower aspect ratio membranes, it may sometimes be possible to artificially soften them by thinning the membrane thickness used for the FD/BEM model. This approach could effectively shift the center frequency closer to the actual center frequency predicted by finite element analysis. An example of this can be found in Figure 36. In this figure, the results for Design D from both FE and FD/BEM modeling techniques are shown. An additional plot is included that corresponds to the frequency response of the same Design D geometry, but with the membrane thickness thinned down by 0.14 um. As can be observed, this shifts the center frequency to the exact position predicted by finite element. Furthermore, this thinning technique results in a nearly identical bandwidth to the FEM as well. Such a process could be used for questionable aspect ratio geometries, before attempting the full array calculation. This would ensure that the individual membranes of the array would exhibit more appropriate dynamic behavior in immersion.

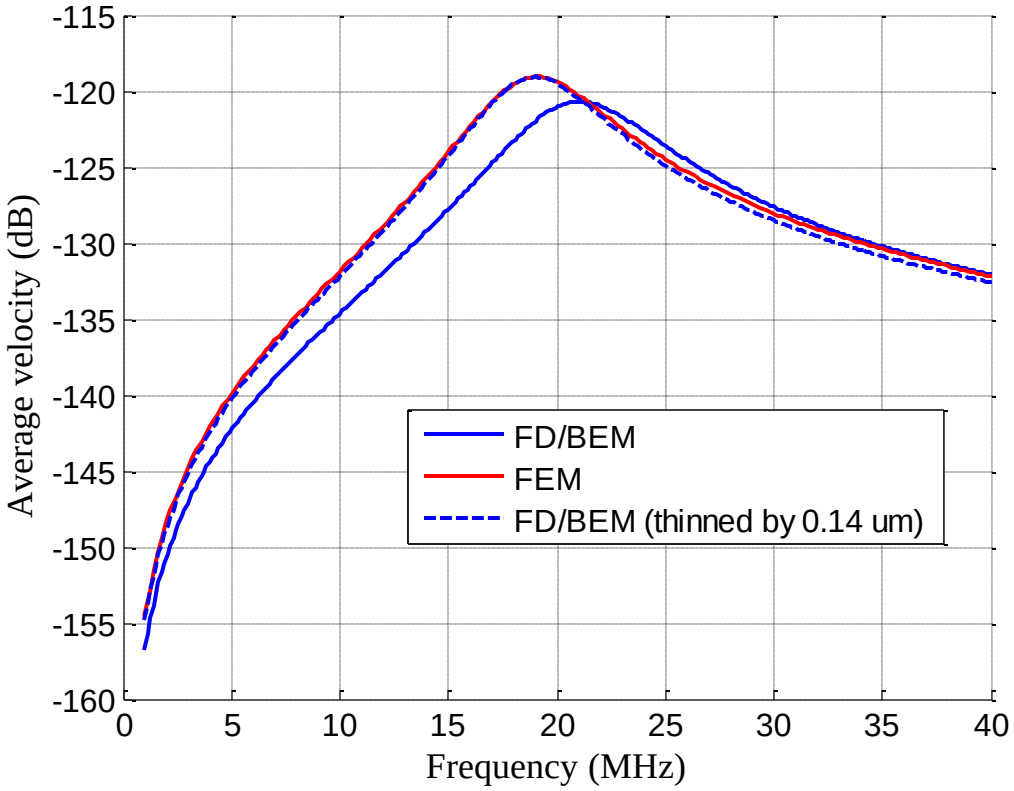


Figure 36. Frequency responses from FEM and FD/BEM model for a single membrane in water (Design D).

5.1.1.4. Frequency Response in Water (Multiple Membranes)

Because the array calculation essentially employs all of the same techniques used for the single membrane case, an additional comparison with FEM results for arrayed membranes in water will yield little additional information. However, for completeness, a simulation of 2 membranes (Design A) spaced 10 um apart was simulated in FEM to demonstrate the functionality of the full analytical calculation. The corresponding frequency response is shown in Figure 37. As expected, the two modeling techniques yield very similar results. They both clearly show a small peak at 9.5 MHz corresponding to a frequency in which mutual crosstalk effects cause a higher mode shape to occur in the membranes, as shown in Figure 38. Note that the green surface in this figure denotes a plane of symmetry. The small discrepancy between the 2 frequency

responses may be improved further with finer meshing and/or a larger fluid radius in the finite element model. However, due to processing limitations of the computer used for these simulations, it was difficult to implement a sufficiently fine mesh.

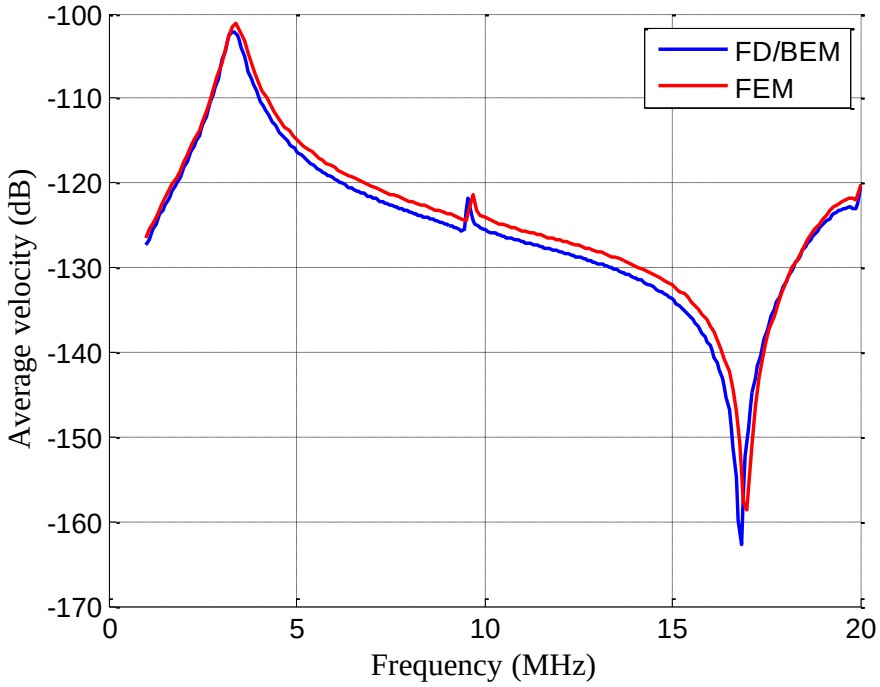


Figure 37. Frequency responses from FEM and FD/BEM model for 2 membranes in water with 10 um spacing (Design A)

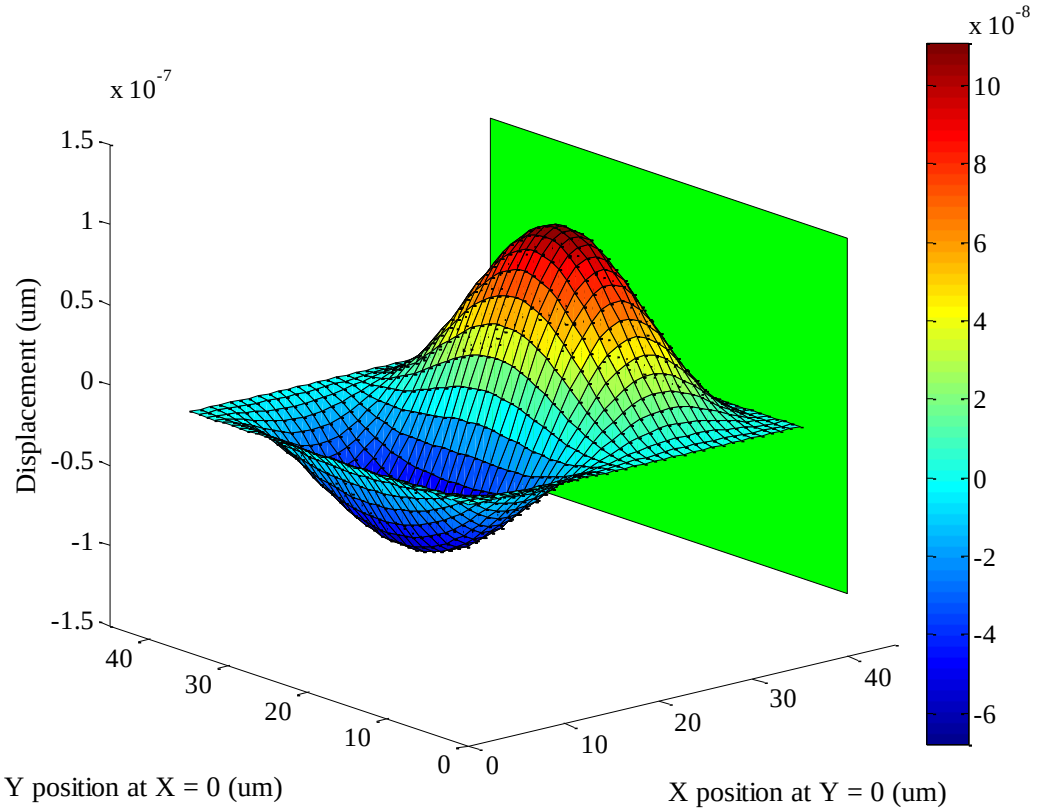


Figure 38. Displacement of 2 membrane array in water at 9.5 MHz. Green surface denotes plane of symmetry.

Regarding the artificial thinning technique mentioned in the previous section, Design D was again used for a 2 membrane array simulation. It was then thinned down by 0.14 um, and simulated again. The resultant plots from FEM and the FD/BEM model are shown in Figure 39. With respect to bandwidth and center frequency, the thinned down membrane has a much closer correlation with the FEM frequency response than when the actual membrane thickness was used. There is a small discrepancy in magnitudes, but it is minimal and may be improved with better meshing and a larger fluid radius in the FE model. This figure demonstrates the potential for utilizing the FD/BEM model even for low aspect ratio membranes. It does require an additional FE model to be solved, but the added time necessary for this step would be insignificant relative to the time saved using the FD/BEM model for the full array simulation. Obviously, this

technique must be used on a case-by-case basis, and may not be valid for certain geometries.

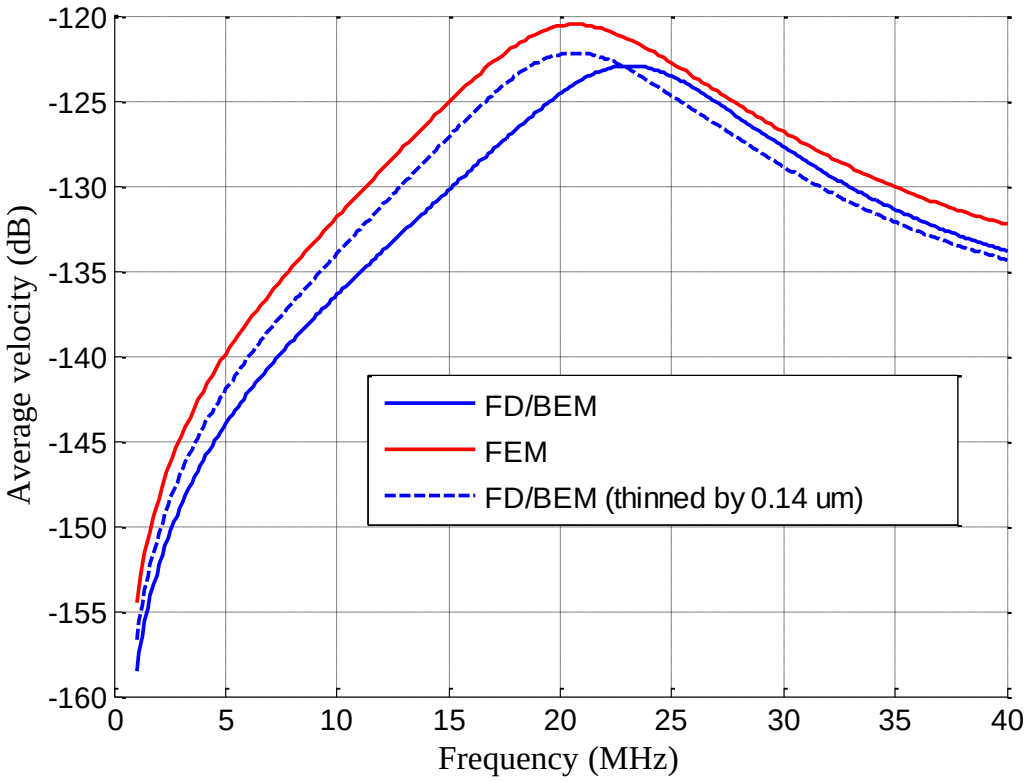


Figure 39. Frequency responses from FEM and FD/BEM model for 2 membranes in water (Design D)

5.2. Comparison with Experiment

5.2.1. Network Analyzer Measurements

To measure the impedance, admittance, and collapse voltage of the CMUT devices, the setup shown in Figure 40 was used. Each array was epoxied and wirebonded to a PCB. Ground and bias lines were soldered from the PCB to an SMA adapter. This adapter was attached to an Agilent 8753 ES Network Analyzer (NA) through a custom bias box. The bias box sums the input voltage from the NA with a static bias provided by a high voltage power supply. The NA applies a small-signal sinusoidal voltage over a

specified frequency range. It measures the electrical reflection coefficient which can then be converted to a complex-valued impedance or admittance.

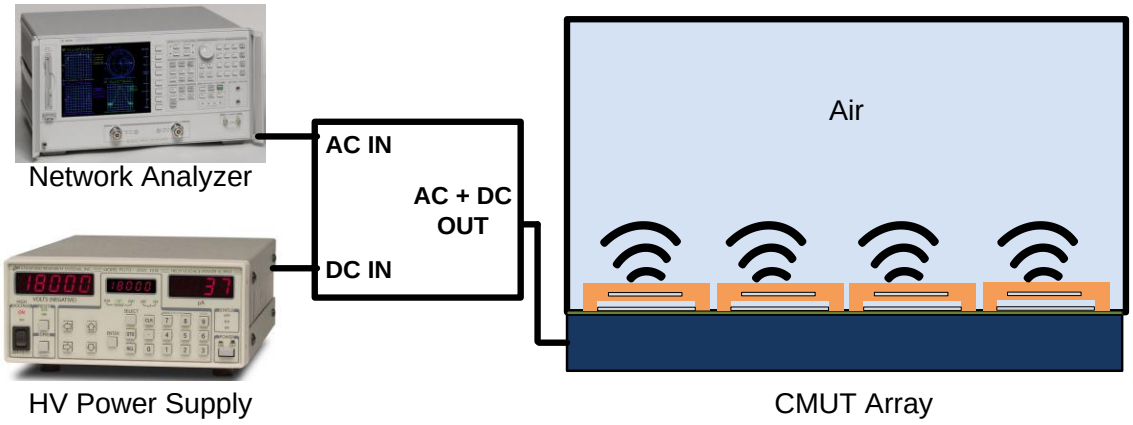


Figure 40. Illustration of experimental setup used to measure impedance and admittance

For air measurements, acoustic crosstalk is negligible, so testing multiple array designs that contain the same membrane geometry is not necessary. Thus, only two air measurements were taken corresponding to the two membrane geometries shown in Table 6.

Table 6. Membrane Geometries for Fabricated CMUT Arrays Used in all Experiments

Geometry	**h_{gap} (um)**	**h_{isol} (um)**	**Electrode Dimensions (um x um)**	**Width (um)**	**Length (um)**	**Membrane Thickness (um)**
A	0.12	0.56	25 x 25	35	35	1.9
B	0.12	0.56	20 x 20	25	25	1.9

5.2.1.1. Estimating Damping and Parasitic Capacitance

In air, the real magnitudes of impedance and admittance are both affected by membrane damping, which is a parameter that is difficult to predict and can vary widely between devices. Thus, it is typically estimated by comparing experimental results with analytical calculations. An arbitrary damping factor is then added to the analytical model

such that its predicted impedance and admittance magnitudes match what is observed experimentally.

Furthermore, the real part of impedance (both center frequency and magnitude) is also a function of parasitic capacitances and the static capacitance. Static capacitance can be predicted accurately with the FD/BEM model, but parasitics will be unique to each device and are dependent on wirebonds, soldered connections, etc. Thus, for these experiments, the damping factor was estimated by matching experimental admittance magnitudes, and the parasitic capacitances were determined by comparing simulations with impedance curves.

The estimated parasitic capacitances for Designs A and B were 2 pF and 3 pF, respectively. These estimates agree closely with actual parasitic capacitance measurements of wirebonded devices conducted previously. The damping for Designs A and B were estimated to be 2800 Pa-s/m and 6000 Pa-s/m, respectively.

5.2.1.2. Collapse Voltage

The collapse voltage of the two membrane geometries was also determined using the impedance measurements in air. As the bias on the membrane is increased, the center frequency of the impedance plot will shift lower due to spring softening, while the overall shape of the curve will remain relatively constant. However, when the membranes collapse, a large shift in the center frequency and a significant change in the shape of the frequency response will be observed. The amplitude of the center frequency will also typically decrease dramatically, because small-signal operation beyond the collapse voltage is not efficient. Experimental testing indicated a collapse voltage of 109 V for Design A and 212 V for Design B. The FD/BEM simulation accurately predicted corresponding collapse voltages of 118V and 221V. The small discrepancy between simulation and experiment can be attributed to the fact that theoretical calculations assume that the membrane will reach a stable equilibrium with the application of any

applied voltage below collapse. However, experimentally increasing the bias on a membrane introduces transients as the voltage is stepped up incrementally. Near collapse, these transients are enough to cause the membrane to swing so far beyond its stable equilibrium that it will prematurely collapse. Thus, analytical calculations tend to always predict higher collapse voltages than what is experimentally observed.

5.2.1.3. Impedance/Admittance in Air

Experimental impedance and admittance plots of the two membrane geometries can be found in the top graphs of Figure 41 - Figure 43. Corresponding simulated plots are located in the bottom graphs of these figures. A summary of the numerical values from these plots can be found in Table 7 and Table 8. The measurements taken for Design A were conducted on an array of 4x4 membranes, and the measurements for Design B were done on a 2x2 membrane array.

Measuring the impedance and admittance of multi-membrane elements tends to result in lower-Q center frequencies than would be predicted analytically. This is because fabricated devices will contain small non-uniformities in material thickness that will cause each membrane to have unique behavior. These effects are typically very small, but in air the high-Q of CMUT membranes enables these non-uniformities to be observed as slight differences in center frequencies at various biases. This in turn causes the center frequency of the whole element to be lower-Q and less defined. This effect is apparent when comparing the simulation predictions (which assume all membranes are identical) to the experimental results. For the element containing 16 membranes of Design A, there is a large discrepancy between actual and predicted bandwidths. For the 4 membrane element of Design B, this effect can still be observed, but is less emphasized. Clearly, the more membranes in an element, the larger the difference.

Note that for Design B, the actual membrane thickness was not used in simulations because, as expected, its small aspect ratio led to inaccuracies in the thin

plate approximation. From FEM comparisons, it was expected that the thin plate approximation would artificially stiffen the membrane, predicting an inaccurately high collapse voltage and a center frequency increased by 2-3 MHz for low biases. This was confirmed through experiments. Thus, the simulated membrane was "thinned" to 1.74 um for a closer match to experiments.

For both designs, there is a clear correlation between simulation and experiment. At nearly all bias voltages, the center frequencies are predicted to within 1% error from experimental measurements. Beyond approximately 90% collapse, the membrane behaviors become highly nonlinear and simulation predictions are less accurate. In Figure 43, for example, the predicted center frequency is nearly 1.5 Mhz lower than expected for 200 V bias, but at nearly 95% of collapse, this inaccuracy is expected. Other discrepancies between predictions and experiment can also be attributed to the estimations for parasitic capacitance and damping.

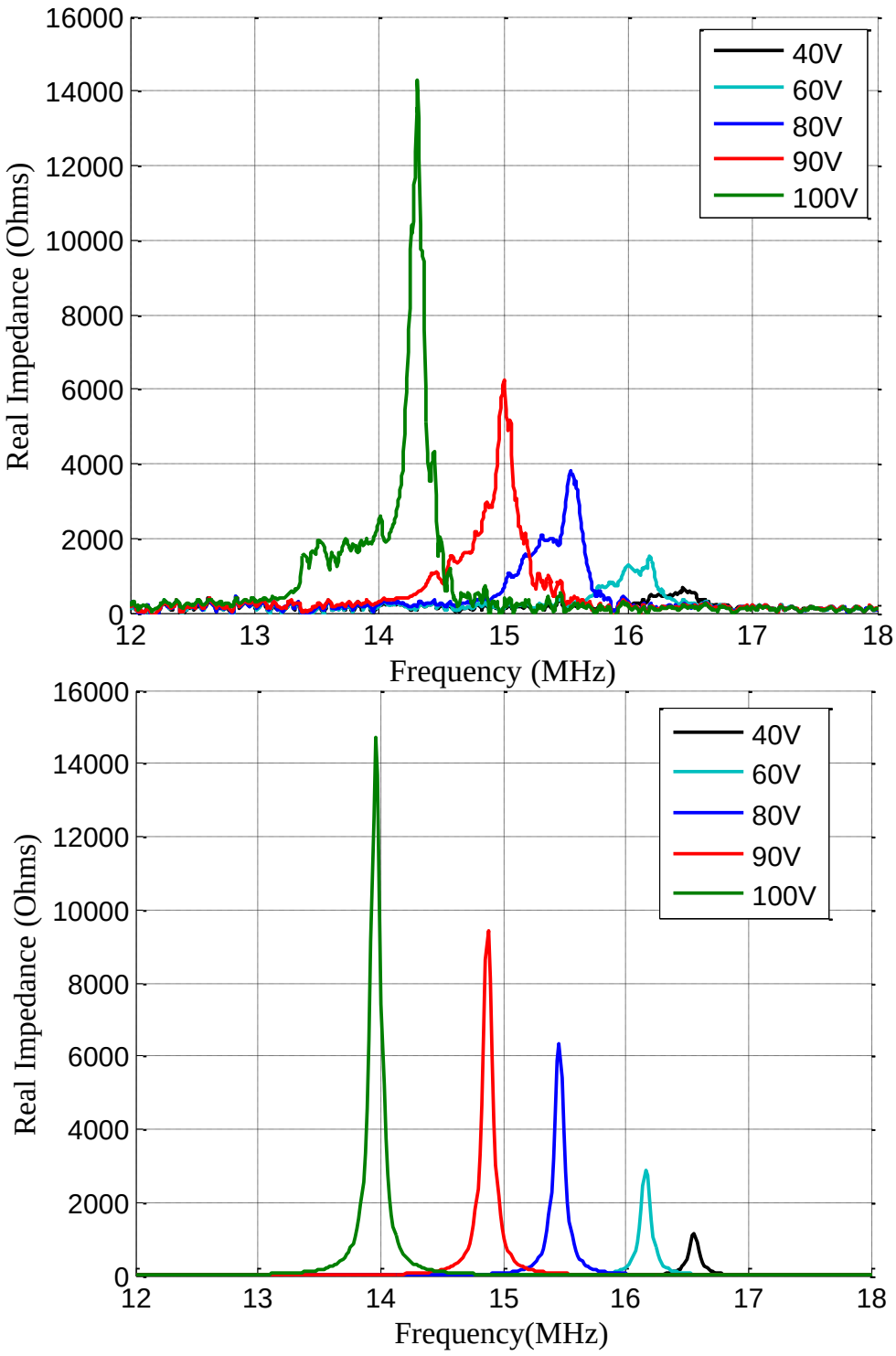


Figure 41. Impedance data from experiment (top) and from FD/BEM simulation (bottom) for varying biases. Data is for a 16 membrane element (Design A) in air.

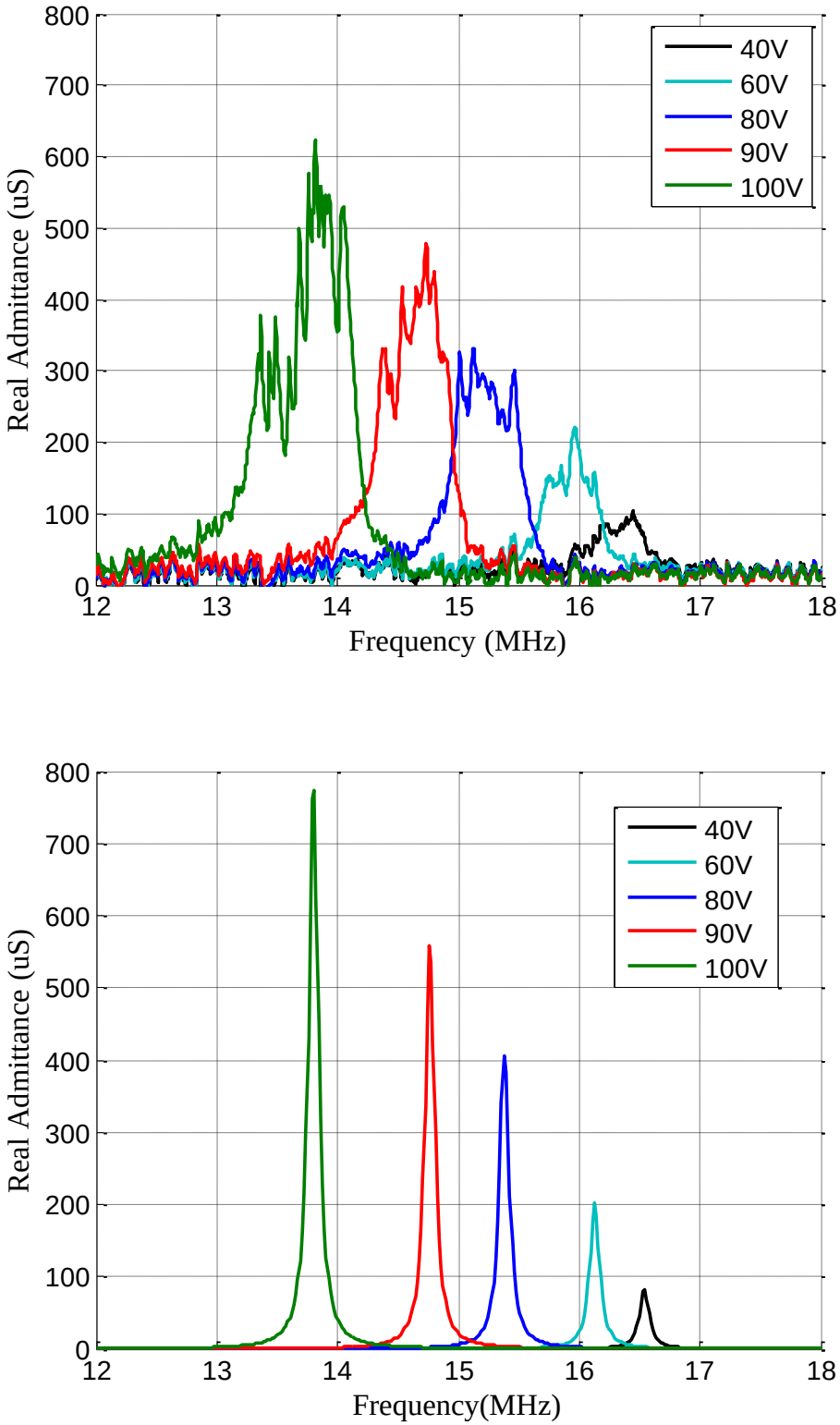


Figure 42. Admittance data from experiment (top) and from FD/BEM simulation (bottom) for varying biases. Data is for a 16 membrane element (Design A) in air.

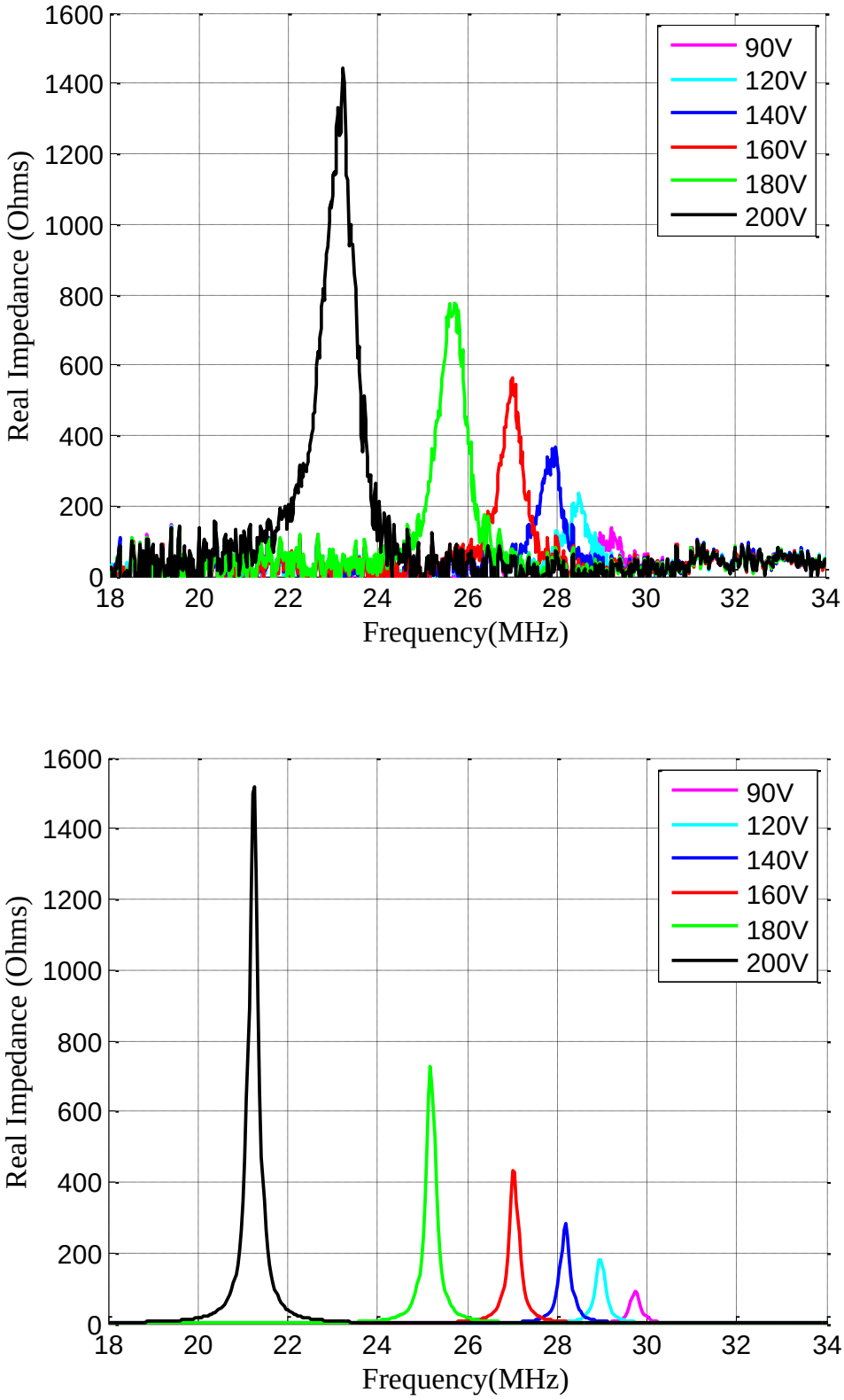


Figure 43. Admittance data from experiment (top) and from FD/BEM simulation (bottom) for varying biases. Data is for a 4 membrane element (Design B) in air.

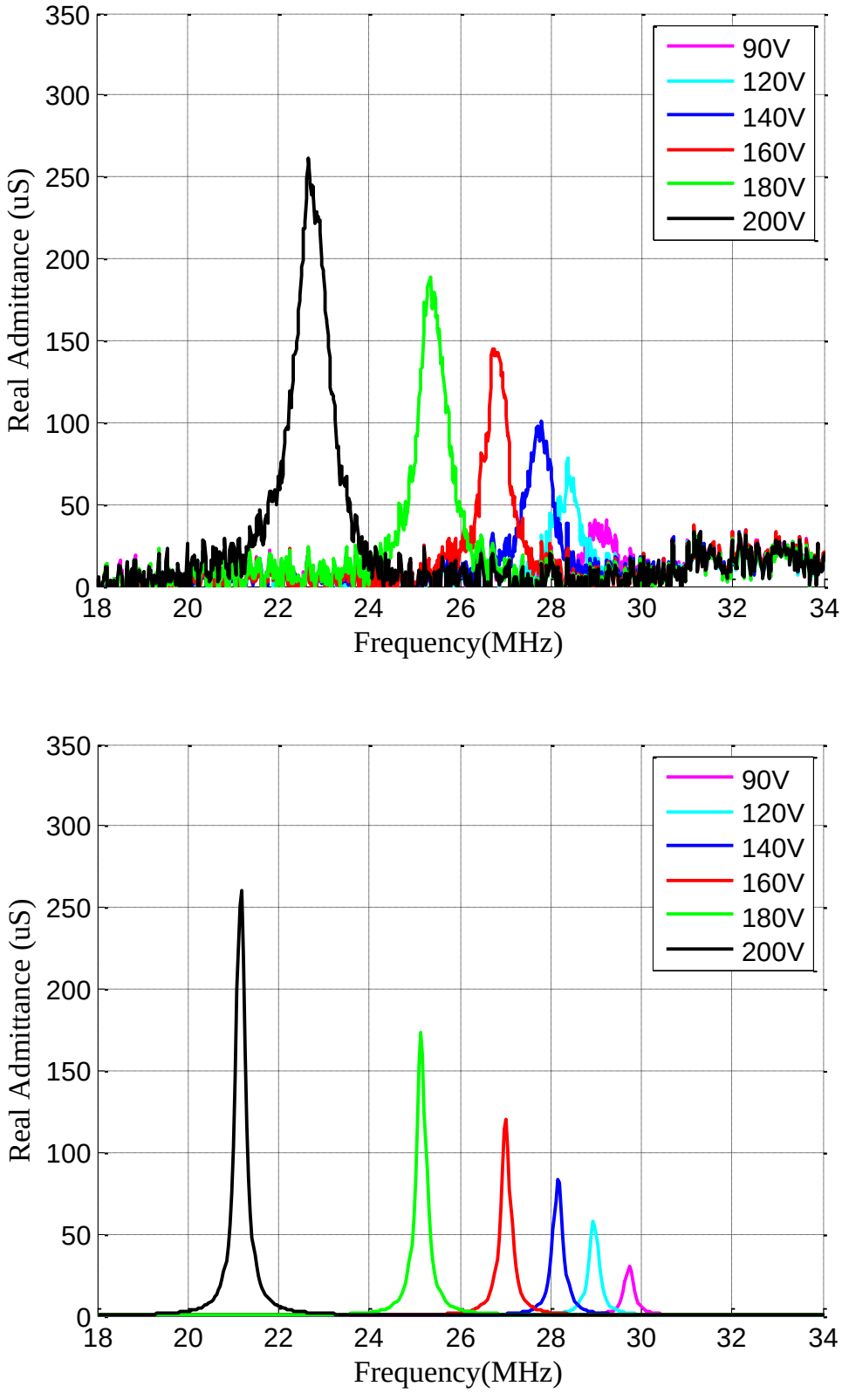


Figure 44. Impedance data from experiment (top) and from FD/BEM simulation (bottom) for varying biases. Data is for a 4 membrane element (Design B) in air.

Table 7. Summary of impedance/admittance results for Design A

	Real Impedance				Real Admittance			
	Center Frequency (MHz)		Magnitude @ Peak Frequency (Ω)		Center Frequency (MHz)		Magnitude @ Peak Frequency (Ω)	
Bias	**Exp.***	**FD/BEM**	**Exp.**	**FD/BEM**	**Exp.***	**FD/BEM**	**Exp.**	**FD/BEM**
40	16.2-16.6	16.6	665	1131	16.2-16.5	16.5	103	83
60	15.8-16.2	16.2	1500	2897	15.7-16.2	16.1	222	203
80	15.3-15.6	15.5	3809	6332	15.0-15.5	15.4	332	405
90	14.9-15.0	14.9	6235	9414	14.3-14.9	14.8	478	558
100	14.2-14.4	14.0	14300	14700	13.3-14.0	13.8	624	772

*Reported as a range encompassing all magnitudes that are greater than 50% of the peak value

Table 8. Summary of impedance/admittance results for Design A

	Real Impedance				Real Admittance			
	Center Frequency (MHz)		Magnitude @ Peak Frequency (Ω)		Center Frequency (MHz)		Magnitude @ Peak Frequency (Ω)	
Bias	**Exp.***	**FD/BEM****	**Exp.**	**FD/BEM****	**Exp.***	**FD/BEM****	**Exp.**	**FD/BEM****
90	28.9-29.5	29.7	137	92	28.8-29.4	29.7	40	31
120	28.0-28.8	29.0	235	181	28.0-28.6	28.9	78	58
140	27.6-28.0	28.2	370	284	27.4-28.1	28.2	101	83
160	26.7-27.3	27.0	565	434	26.5-27.0	27.0	145	120
180	25.3-26.0	25.2	781	728	25.1-25.8	25.1	188	175
200	22.7-23.5	21.3	1441	1517	22.3-23.0	21.2	262	261

*Reported as a range encompassing all magnitudes that are greater than 50% of the peak value

**Values correspond to the thinned down membrane simulation

5.2.2. Hydrophone Measurements

For hydrophone measurements, the setup shown in Figure 45 was used. The CMUT arrays were epoxied and wirebonded to a PCB, and subsequently coated in 3 um of Parylene. The PCBs were connected to the output of the same bias box used for the Network Analyzer experiments, and as before, the DC bias was provided by a high voltage power supply. The dynamic transmit signal was provided by an Agilent 3325a arbitrary waveform generator, which was amplified by an ENI 310-L RF Power Amplifier. The CMUT arrays were immersed in water, and their transmitted pressure waves were detected by an ONDA Corporation HGL-0085 Hydrophone which utilized an AH-2010 pre-amplifier. The output of the pre-amplifier was then sent to an oscilloscope and recorded by a PC using a GPIB connection. A Newport ESP300 Universal Motion Controller was used to control an X-Y stage and/or a rotation stage, which enabled the CMUT array to be moved in small, controlled increments with respect to the stationary hydrophone. The X-Y stage allowed for planar scans of the radiated pressure fields, and the rotation stage allowed for angular measurements.

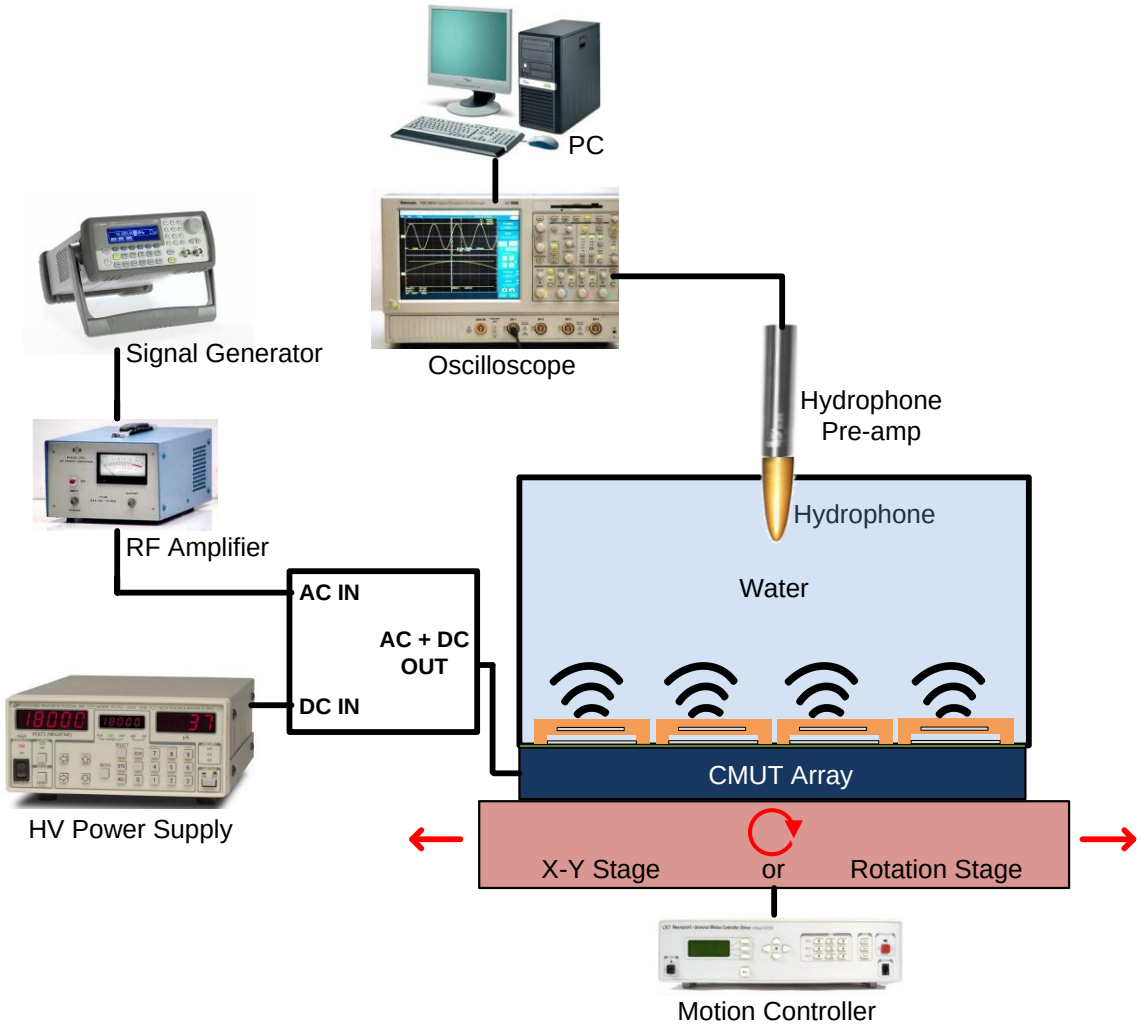


Figure 45. Illustration of experimental setup used to measure transmit signal of a CMUT with a hydrophone

To determine the frequency response of the CMUT arrays, the recorded temporal response of the hydrophone was converted to the spectral domain using an FFT. Its magnitude was adjusted by the hydrophone's calibration data, and compared with the FD/BEM results.

5.2.2.1. Simulation Methodology

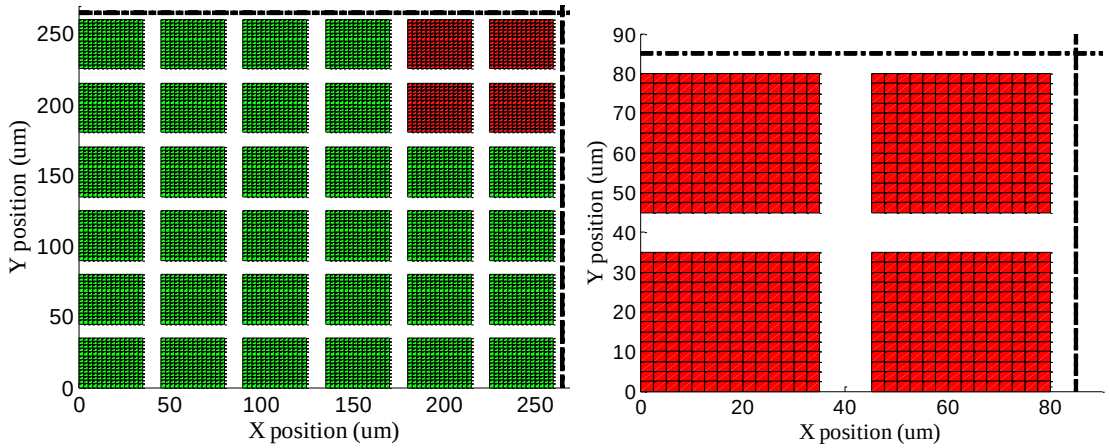
To predict the hydrophone measurement results using the FD/BEM model, the Rayleigh integral addition to the model, as described in section 3.6.3, is needed. This

allows the velocity distribution of the array to be transformed to a pressure field at a certain distance away from the array. For accurate results, it is important that the nodal mesh created at the specified distance accounts for the active area of the hydrophone receiver. Depending on the type of experiment setup being simulated, there are a few different ways to account for this.

If a frequency response experiment is being conducted in which the FFT of a single pulse is calculated, then the most accurate method would be to mesh a planar surface in space that has the dimensions of the hydrophone. The average pressure over this plane should then be averaged at each frequency for comparison with the FFT of the hydrophone output.

If planar or angular scan results from a hydrophone are being compared, then it is most convenient to create a plane or curve in space that utilizes the same spacing and number of locations (nodes) as the actual hydrophone scan. Then, when performing the Rayleigh integral, the elemental area surrounding each node over which the Rayleigh integral is performed should be set equal to the area of the hydrophone transducer. In this way, for either a planar or angular scan, the total pressure calculated at each node will be equal to what the hydrophone would measure. Note that this is only a valid approach due to the relatively small size of the hydrophone (approximately 80um x 80um). Thus, in the far-field, there should not be a significant pressure gradient across its surface. For larger receivers, a pressure gradient would occur across the transducer's surface and thus the active area of the receiver would need to be meshed and averaged at each location in space.

The modeled arrays shown in Figure 46 were used to predict the results of experimental hydrophone tests on fabricated devices with identical geometries. They include a 12x12 membrane array with an active 4x4 membrane center element, and an isolated 4x4 membrane element. The models did not account for the 3um of parylene, so some discrepancies were expected.



(a) (b)

Figure 46. Model geometries for (a) a 12x12 membrane array with an active 4x4 center element, and (b) an isolated 4x4 membrane array. These correspond to fabricated devices with identical geometries used in the experimental hydrophone tests. Note the use of quarter symmetry.

5.2.2.2. Frequency Response

The hydrophone was used to first measure a single pulse response from the different array designs. A single cycle, bipolar pulse centered at approximately 8 MHz was used to excite the arrays with a broadband signal. The FFT of each signal was then calculated and compared with the simulated results. The simulated plots were adjusted by the FFTs of the input voltage signals.

The pulse response of the 12x12 membrane array with active center element is displayed in Figure 47. The corresponding simulation prediction is shown in Figure 48. As can be observed, the key features of the experimental FFT are accurately predicted by the FD/BEM model. The large notch at 8 MHz and the small peaks on either side of it are clearly visible in Figure 48. The small discrepancy in the center frequencies is partly attributable to the fact that the FD/BEM model did not account for the 3um of parylene. From previous FEM analyses, it was determined that a parylene layer can slightly lower or raise the center frequency of a membrane by 1-2 MHz, depending on its exact thickness. The parylene layer will also add 20-30V to the collapse voltage. The other

notches in the experimental FFT that are not visible in the simulated plot could be caused by reflections from the diced edges of the silicon wafer, or lamb waves propagating through the thickness of the substrate. Neither of these is accounted for by the model.

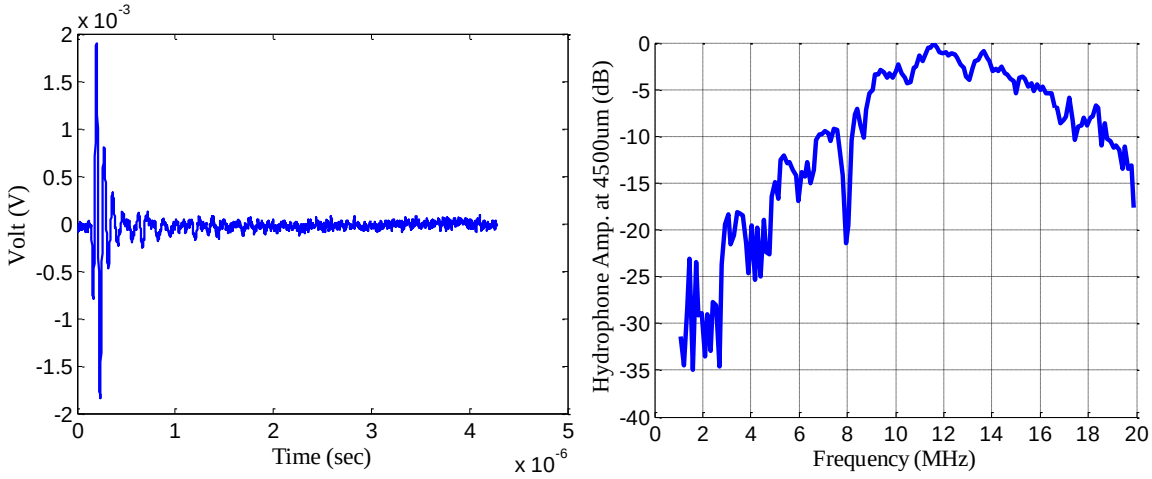


Figure 47. Temporal hydrophone response (left) for a 12x12 membrane array with active center element and its associated FFT (right).

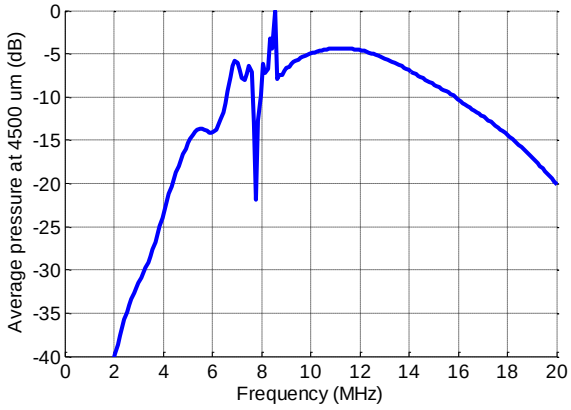


Figure 48. Simulated frequency response for a 12x12 membrane array with active 4x4 membrane center element. Plot has been adjusted by the experimental input signal.

The pulse response of the isolated 4x4 membrane array with active center element is displayed in Figure 49. The corresponding simulation prediction is shown in Figure 50. Once again, the overall shape of the experimental FFT is accurately predicted by the FD/BEM model, including the drop-off below 8 MHz, and the increased bandwidth and

lack of sharp crosstalk features as compared with the un-isolated 4x4 membrane case. There are still some small discrepancies between the two plots, but, as before, they likely exist because the FD/BEM did not account for the 3um of parylene, nor did it consider the effects of lamb waves or reflections off the diced silicon edges.

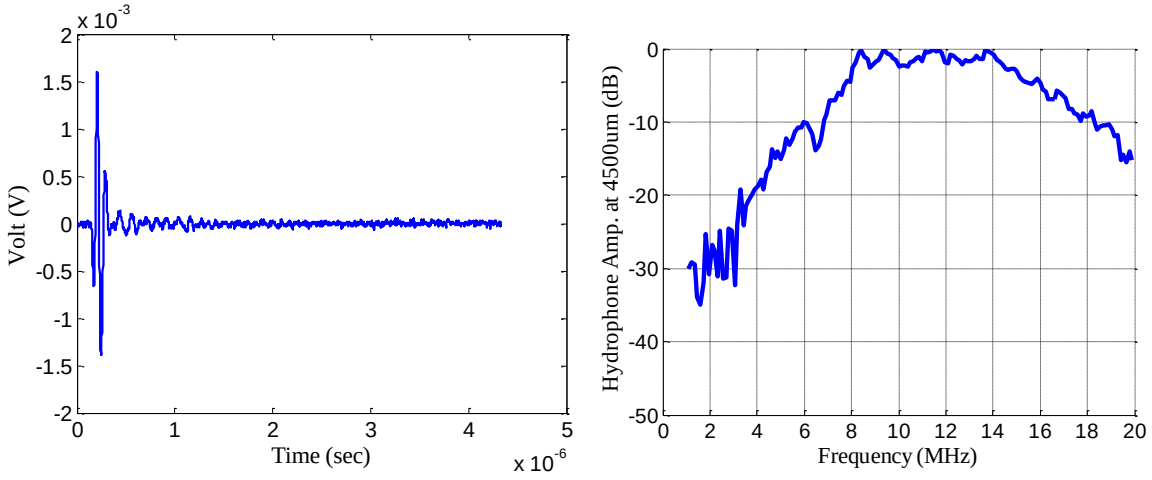


Figure 49. Temporal hydrophone response (left) for an isolated 4x4 membrane array and its associated FFT (right).

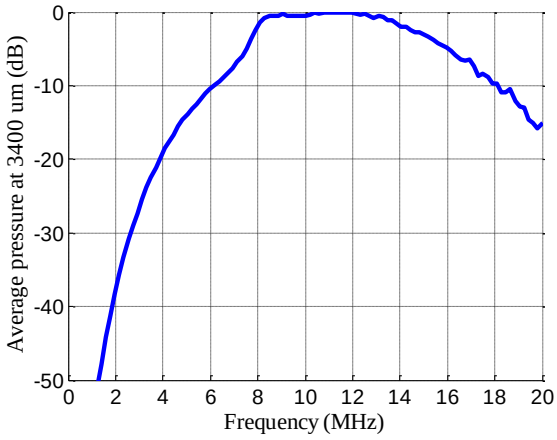


Figure 50. Simulated frequency response for an isolated 4x4 membrane array. Plot has been adjusted by the experimental input signal.

5.2.2.3. Radiation Pattern

The radiation patterns of the three array geometries were also characterized using a rotation stage. A single cycle, bipolar pulse centered at approximately 8 MHz was used to excite the arrays with a broadband signal. A large piezoelectric receiver was used to detect the pressure waves transmitted by the CMUT array, as it was rotated between 0 and 180 degrees, with 2 degree increments. Due to limited space in the experimental setup, a full 180 degree sweep was not possible, so most scans only covered approximately 140 degrees. The FFT of the received signals at each angle were then calculated, and radiation plots were generated by using the magnitudes at particular frequencies from these FFTs.

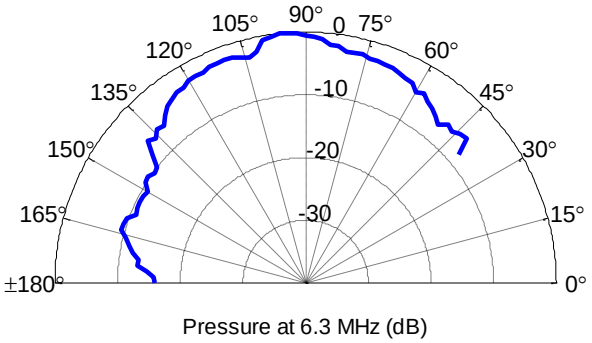
Radiation plots were generated with the FD/BEM model assuming that the piezoelectric element was small enough to neglect pressure gradients across its surface. Since the measurements were performed in the far-field, this should be an acceptable approximation. Finally, as a comparison, radiation plots were also generated for the ideal case in which crosstalk does not exist, and all active membranes vibrate in-phase with equal amplitude (inactive membranes do not move). Essentially, this serves as a metric for determining how significantly crosstalk alters the radiation pattern of a CMUT array.

The radiation plots for the isolated 4x4 membrane array can be found in Figure 51 - Figure 53, for frequencies of 6.3 MHz, 8 MHz, and 9.1 MHz, respectively. Within each figure, designations of (a), (b), and (c) are given respectively to the experimental, FD/BEM, and ideal piston plots.

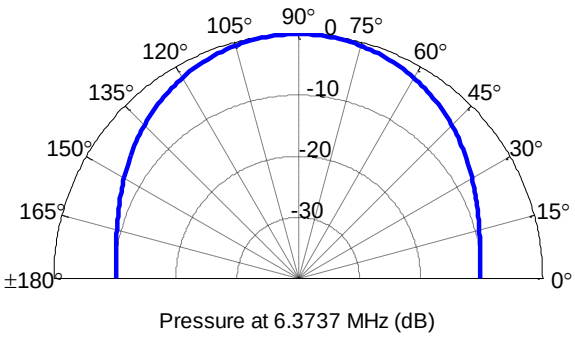
At 6.3 MHz, crosstalk effects are negligible as evidenced by the identical predictions between both the FD/BEM model and the ideal piston case. Both methods accurately predicted a relatively uniform radiation pattern with a slightly higher magnitude pressure wave being sent at 90 degrees. At 8 MHz, crosstalk is significant, and thus there is a notable difference between the ideal piston and the FD/BEM plot. The

experimental results reflect a beam pattern much closer to the FD/BEM prediction in terms of the beamwidth of the main lobe, the null at 145 degrees, and the existence of a side lobe (which the ideal piston case does not predict). A similar result can be observed at 9.1 MHz. It is interesting to observe that even for this isolated element, without any surrounding inactive membranes, crosstalk still affects the radiation pattern at certain frequencies.

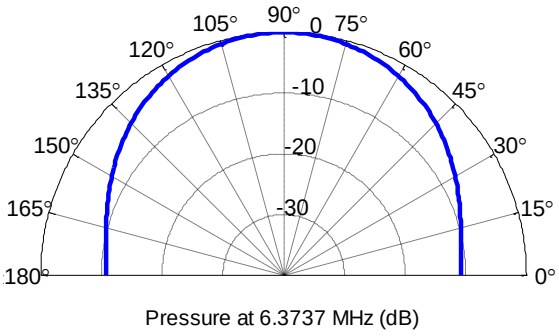
In spite of the fact that the FD/BEM model more accurately depicted the radiation plots than the ideal piston case, there were still notable differences between experiment and simulation, especially in high-crosstalk regions. These discrepancies were likely due in part to non-uniformities in the fabricated array, and the lack of a parylene layer in the simulation. Furthermore, if the array was slightly tilted during the angular scan, this could have skewed all of the features of the beam pattern.



(a)

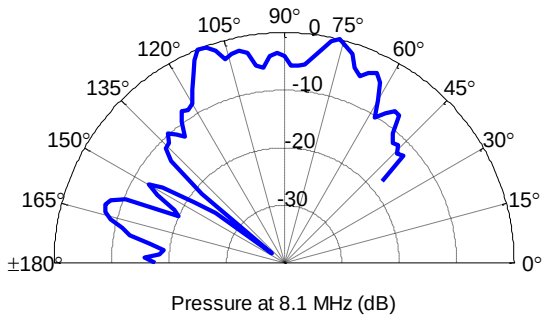


(b)

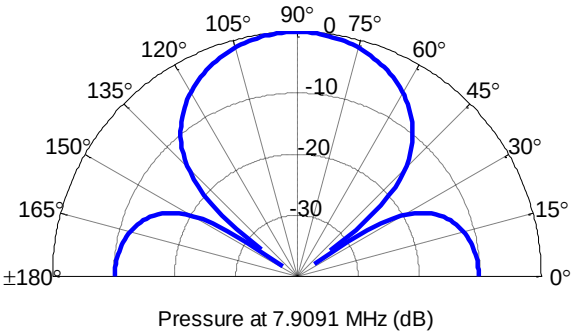


(c)

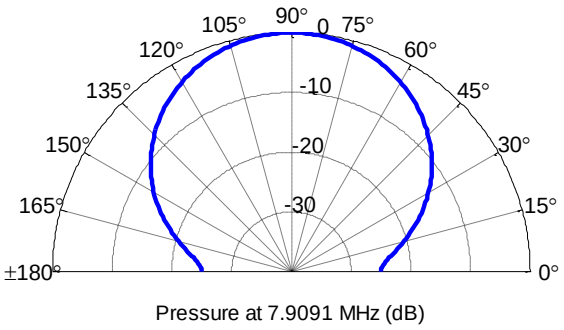
Figure 51. Angular radiation patterns for the isolated 4x4 membrane array at approximately 6.3 MHz for (a) the fabricated device, (b) the FD/BEM model, and (c) an ideal case considering all active membranes as pistons vibrating in-phase with equal amplitudes.



(a)

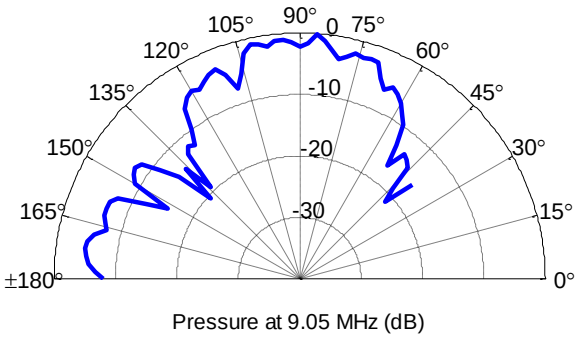


(b)

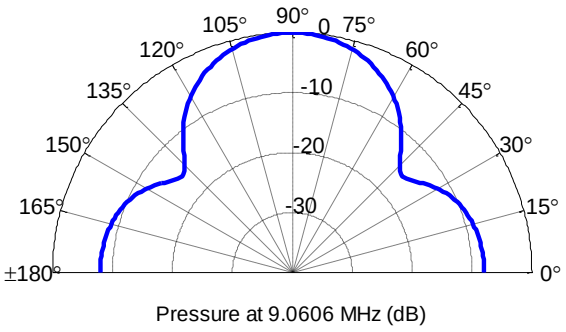


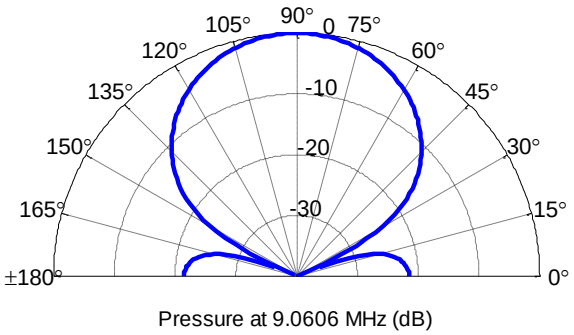
(c)

Figure 52. Angular radiation patterns for the isolated 4x4 membrane array at approximately 8 MHz for (a) the fabricated device, (b) the FD/BEM model, and (c) an ideal case considering all active membranes as pistons vibrating in-phase with equal amplitudes.



(a)





(b) (c)

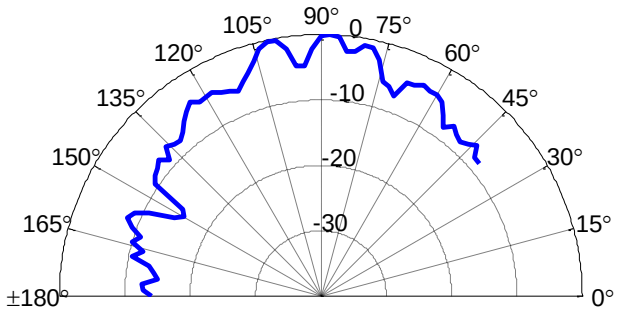
Figure 53. Angular radiation patterns for the isolated 4x4 membrane array at approximately 9.1 MHz for (a) the fabricated device, (b) the FD/BEM model, and (c) an ideal case considering all active membranes as pistons vibrating in-phase with equal amplitudes.

The radiation plots for the 12x12 membrane array with active 4x4 membrane center element can be found in Figure 54 - Figure 56, for frequencies of 8 MHz, 8.6 MHz, and 12.2 MHz, respectively. Within each figure, designations of (a), (b), and (c) are given respectively to the experimental, FD/BEM, and ideal piston plots.

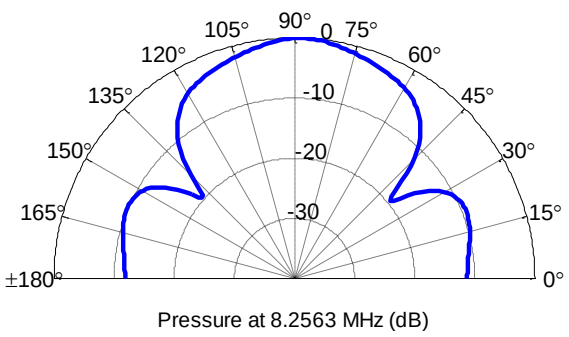
Once again, the FD/BEM model more accurately depicts the behavior of the CMUT array, particularly in regions of high crosstalk. This becomes most obvious when comparing the locations and magnitudes of nulls and side lobes. However, there are still some significant discrepancies between the model and experiment, even more so than

what was observed for the isolated 4x4 membrane element. These differences can once again be attributed to membrane non-uniformities, the lack of a parylene layer in the model, and possibly a tilted array during the angular scan. Since this array contains many more membranes than the isolated 4x4 element, non-uniformities in the array can potentially cause more drastic inconsistencies than in the previous case.

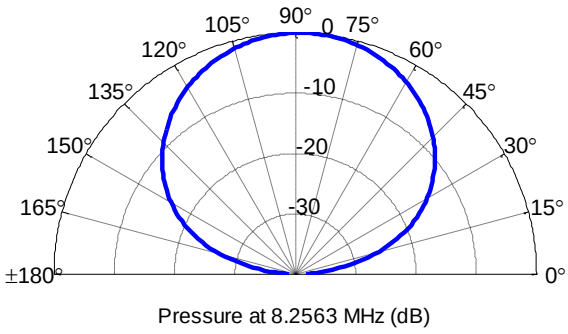
It should also be noted that in the region of high crosstalk, the beam pattern changes significantly with small changes in frequency. Furthermore, since the modeled membranes do not precisely match the fabricated ones (because of non-uniformities, no parylene layer, etc), a particular beam pattern may occur at a different frequency experimentally than in the model. Thus, the radiation patterns may not agree due to the difficulty involved with precisely matching the correct frequencies. In general, though, the FD/BEM model seems to give a more accurate depiction of the radiation patterns for CMUT arrays in terms of the main beamwidth, and the depth and locations of nulls and sidelobes.



(a)

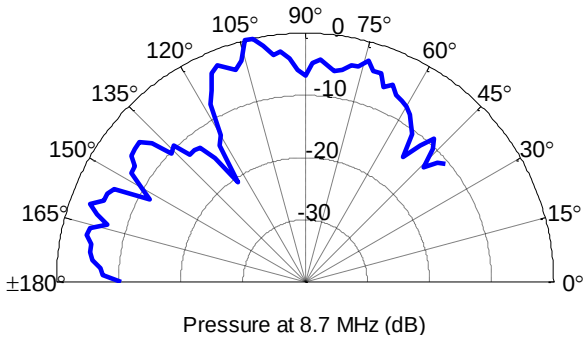


(b)

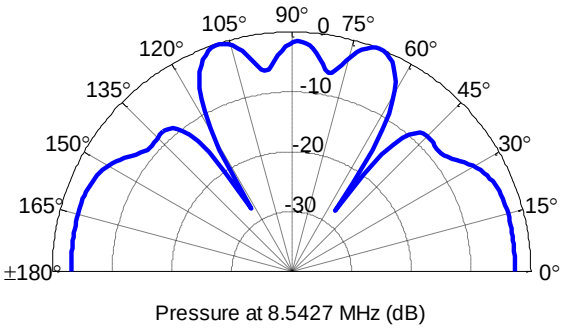


(c)

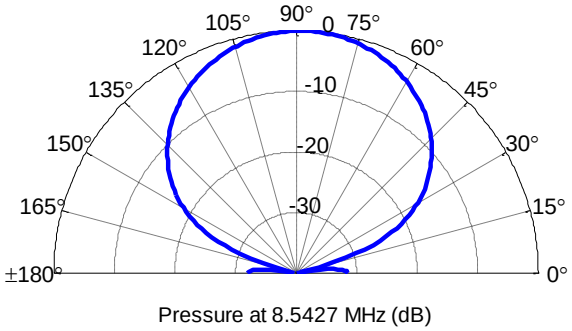
Figure 54. Angular radiation patterns for the 12x12 membrane array with active 4x4 membrane center element at approximately 8 MHz for (a) the fabricated device, (b) the FD/BEM model, and (c) an ideal case considering all active membranes as pistons vibrating in-phase with equal amplitudes.



(a)

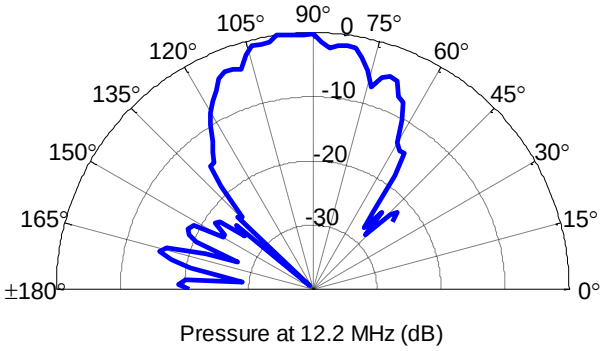


(b)

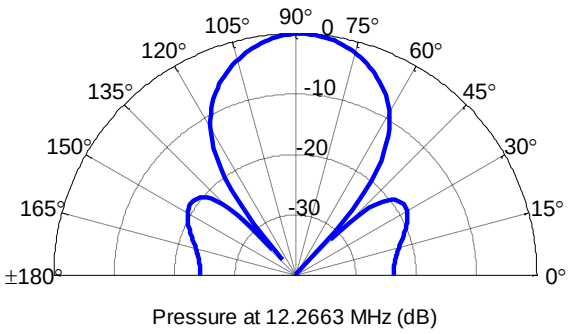


(c)

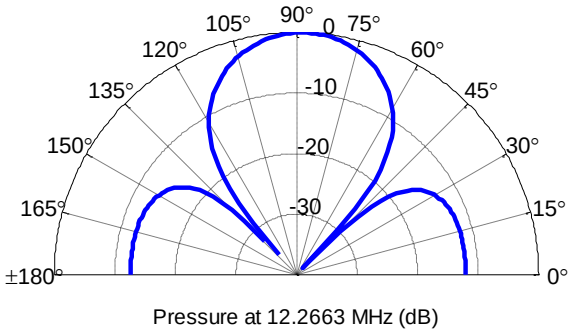
Figure 55. Angular radiation patterns for the 12x12 membrane array with active 4x4 membrane center element at approximately 8.6 MHz for (a) the fabricated device, (b) the FD/BEM model, and (c) an ideal case considering all active membranes as pistons vibrating in-phase with equal amplitudes



(a)



(b)



(c)

Figure 56. Angular radiation patterns for the 12x12 membrane array with active 4x4 membrane center element at approximately 12.2 MHz for (a) the fabricated device, (b) the FD/BEM model, and (c) an ideal case considering all active membranes as pistons vibrating in-phase with equal amplitudes

5.2.3. Noise Measurements

5.2.3.1. Background

For CMUT arrays containing a large number of elements, a simple method for characterizing the uniformity and functionality of all elements in an array is needed. The simplest method for this is to individually test the impedance of each element using a network analyzer. Unfortunately, this type of measurement requires access to both

electrodes, which is not possible once CMUTs have been integrated with CMOS amplifiers. Thus, a new method for characterizing CMUT arrays was recently proposed by Gurun et al. which uses the thermal-mechanical noise spectrum of the output from the receive amplifiers [44]. This method applies a static bias to the CMUTs, and then measures the noise spectrum from the receive element amplifiers with a spectrum analyzer. This noise spectrum, as will be explained next in this section, is directly related to the total impedance of the CMUT. Consequently, the mechanical behavior of the receive elements of a CMUT array can be characterized through noise measurements without applying a dynamic signal to the elements, as would be required for Network Analyzer measurements. Furthermore, the correlation between noise and CMUT impedance implies that noise measurements can also be used as a passive sensing mechanism for detecting environmental fluid properties.

The equivalent circuit for a CMUT receiver element is shown in Figure 57. This circuit is similar to the transmit circuit shown in Chapter 2, but replaces the AC transmit signal with a trans-impedance amplifier (TIA). It also includes a dynamic applied force which accounts for the thermal-mechanical noise generated everywhere in the surrounding environment. The magnitude of this "force source" is equivalent to:

$$F_{Noise} = \sqrt{4k_B T\left(Real(Z_{CMUT})\right)},$$
$$k_B = 1.38 * 10^{-23}\frac{m^2 kg}{s^2 K} \tag{97}$$
$$Z_{CMUT} = Z_m + Z_r$$

Where k_B is Boltzmann's constant and T is the absolute temperature of the environment. If we then then translate the velocity due to the noise in the mechanical mesh over to the electrical domain using the transformer ratio, the equivalent circuit (after some manipulation) becomes that shown in Figure 58. In this circuit:

$$Z^*_{CMUT} = Z_{CMUT} + n^2 Z_{ss}$$
$$I_{CMUT\ Noise} = \langle \dot{z} \rangle n = \frac{F_{Noise}}{|Z^*_{CMUT}|} n \tag{98}$$
$$Z_{ss} = -\frac{1}{j\omega C_0}$$

Thus, the thermal-mechanical current noise originating from the CMUT will be split between the amplifier internal impedance, the static and parasitic capacitances, and the CMUT impedance. The design of the TIA is such that its impedance will always be much lower than the static capacitance impedance up to around 20 MHz. Furthermore, since we are considering CMUT on CMOS devices, the parasitic capacitance will be very small relative to the static capacitance. Thus, most of the noise current will flow through the CMUT impedance and the amplifier. For water operation, the radiation impedance will be very large relative to the amplifier, and thus most of the noise current will flow directly through the amplifier, implying that noise measurements at the output of the TIA will be directly proportional to the CMUT impedance. For air measurements, the TIA impedance will typically still be smaller than the CMUT impedance, but will be close enough such that some noise current will be lost through the CMUT impedance. Consequently, the SNR of the TIA output will suffer proportionally.

Looking more closely at the current noise, we see that:

$$I_{CMUT\ Noise} = \frac{F_{Noise}}{|Z^*_{CMUT}|} n = \frac{\sqrt{4k_B T\left(Real(Z^*_{CMUT})\right)}}{\sqrt{Real(Z^*_{CMUT})^2 + Imag(Z^*_{CMUT})^2}} n \tag{99}$$
$$\therefore I_{CMUT\ Noise} = n\sqrt{4k_B T\left(Real(Y^*_{CMUT})\right)}$$

Where Y^*_{CMUT} is the admittance of the CMUT with spring softening. Thus, the noise spectrum measured at the output of the receive amplifier is directly proportional to the square root of the real CMUT admittance, assuming the CMUT impedance is sufficiently high. With this knowledge, it becomes possible to predict the noise output using the

FD/BEM model developed in this thesis. These calculated values can be compared with experimental results as another means of validating the accuracy of the model. In the next chapter, the model will be used to evaluate the potential of using CMUT noise measurements as a fluid sensing mechanism.

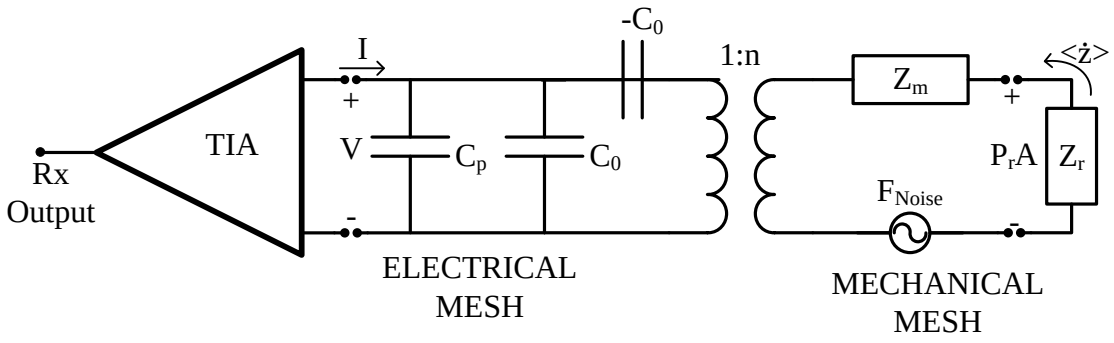


Figure 57. Equivalent electro-acoustic circuit for a CMUT receiver element (includes thermal-mechanical noise)

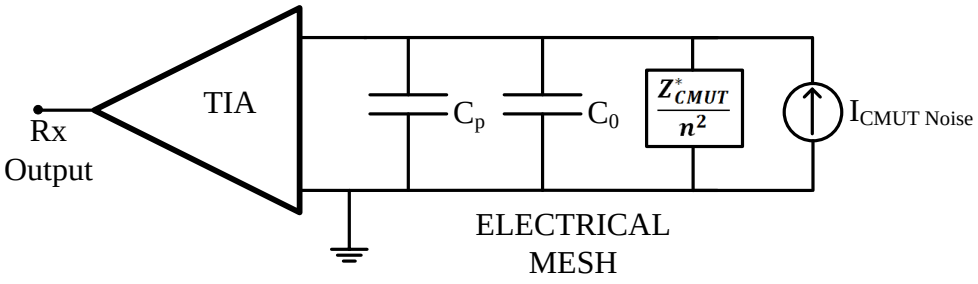


Figure 58. Equivalent electrical circuit for a CMUT receiver element with thermal-mechanical noise

5.2.3.2. Experimental Setup

To measure the noise output of the CMUTs, the setup shown in Figure 59 was used. The top electrode of the CMUT element was connected to the CMOS amplifier/MUX/buffer circuitry using the CMUT on CMOS fabrication technique. The fabricated device was epoxied to a ceramic chip carrier and coated in parylene. The bottom electrode bond pad was connected directly to a high voltage power supply. The

output of the TIA was connected to an Agilent 4395A Spectrum Analyzer. Other voltages were applied to the device to control the CMOS circuitry, but those are not relevant to the scope of this experiment.

The device used for these experiments was a dual-ring array consisting of two rings of 4-membrane elements. As shown in Figure 60, each element contained two pairs of trapezoidal membranes with slightly different sizes.

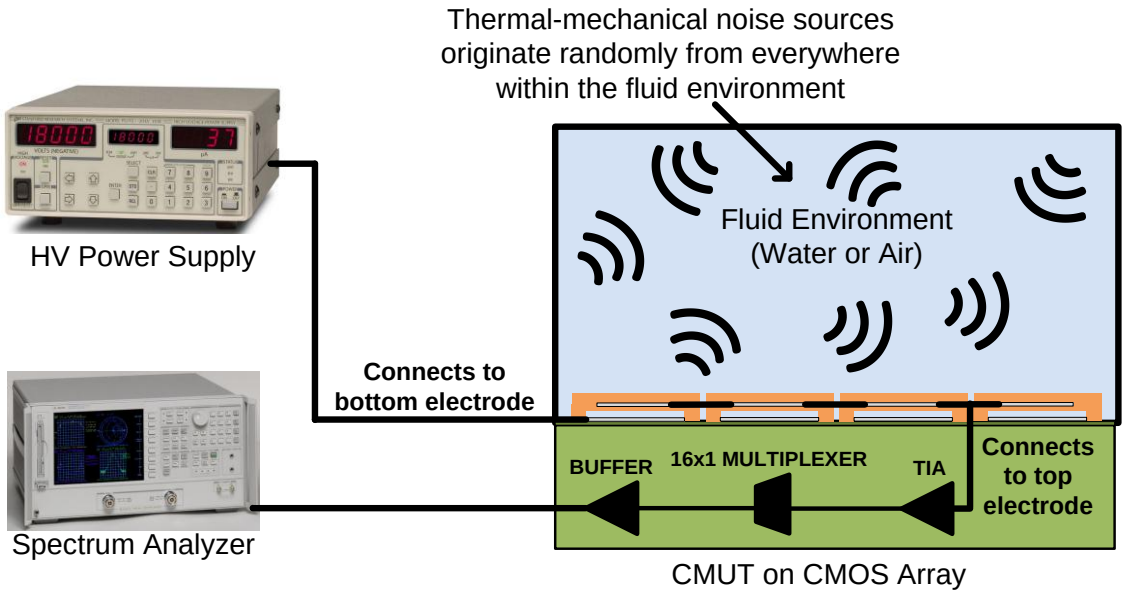


Figure 59. Illustration of experimental setup used to measure input current noise of a CMUT receiver element

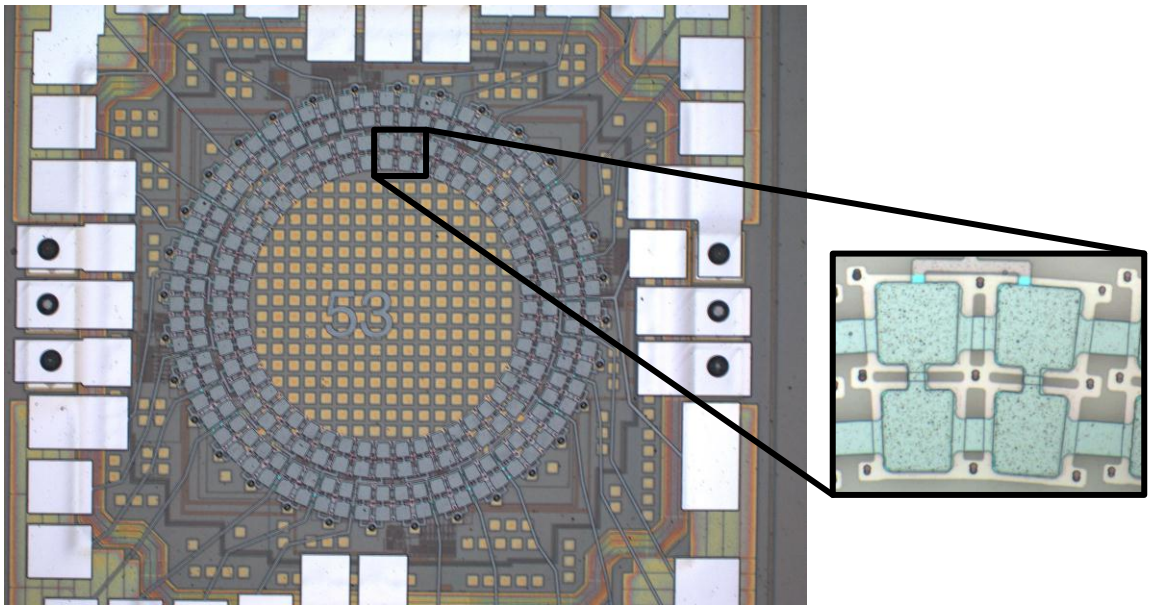


Figure 60. Fabricated device geometry used for noise measurements, demonstrating the presence of two membrane sizes in each element.

5.2.3.3. Simulation Methodology

As mentioned earlier, the device used for these experiments utilized elements that contained 4 trapezoidal membranes of 2 different sizes. Theoretically, the trapezoidal shape could have been incorporated into the FD/BEM model, but for simplification purposes, this characteristic was not considered. Instead, the model utilized the average dimensions of each membrane and assumed that they were rectangular in shape, as shown in Figure 61. Other simplifications include neglecting the curvature of the array. The main inaccuracy that this introduces is that the actual curved array has different inter-membrane spacing within each ring (smaller spacing on the inner ring and larger spacing on the outer ring). The simulated linear array utilizes identical spacing in between membranes, other than slight difference that originates from the different membrane spacing. The membrane spacing affects the periodicity of the array and thus this simplification may slightly shift the locations in the frequency response at which crosstalk notches/peaks occur.

All membranes were given the same bias, and the membranes illustrated in red were actuated with a uniform, harmonic pressure wave across their entire surface. After solving, the total force on the active element was calculated and divided by the average velocity over that same element. This produced a lumped element value for the total mechanical impedance of the element including both mechanical and radiation effects. A lumped element value for the transformer ratio was also calculated using the total electrode area and average gap under those electrodes. With these two values, Equation (102) could then be solved for the estimated input current noise measured at the output of the TIA.

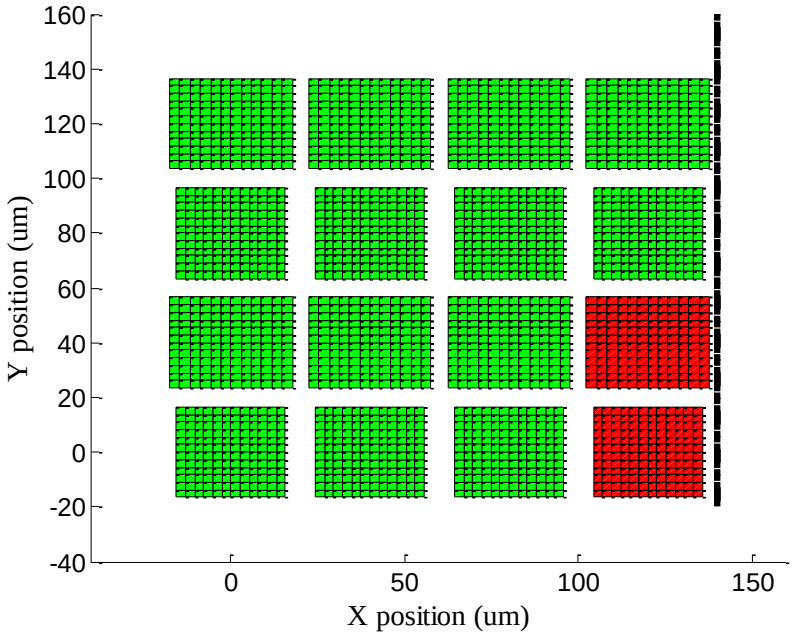


Figure 61. Model geometry used for simulating noise of a dual ring array element in water. All elements are biased. Red membranes have uniform, harmonic pressure wave applied to their surface. Dashed line denotes symmetry axis.

5.2.3.4. Noise Spectrum in Water

The noise spectrums acquired from a single CMUT on CMOS receiver element in immersion at 0 V, 70 V, and 95V bias are shown in Figure 62. The predicted noise spectrums calculated using the FD/BEM model are shown in plots (a) and (b) of Figure

63. Note that all plots are of the square of the noise current. A zoomed in version of these plots which utilizes the same axes as the experimental data is shown in (c) and (d) of Figure 63.

There are extremely high-Q, large magnitude peaks in the predicted frequency response that are not visible in the experimental results. It is expected that these peaks most likely do not show up in experiments because noise measurements require a high number of averages to eliminate extraneous noise from the data. During this process, very high-Q features tend to get eliminated. This is because even the slightest shifts in their frequency over time will result in their large magnitudes being damped out as they are averaged with many low-magnitude data points.

Other than the high-Q features, the relative shape and magnitudes of the graphs match well with experimental results in spite of the simplifications made to the simulation. Specifically, they both contain three distinct regions. The first one occurs between 5 and 10 MHz (at 95V bias) and corresponds to a region of large amplitude, high-Q peaks. The second region is located between 10 and 20 MHz and is a much broader band, smoothed peak. The final region occurs beyond 20 MHz and displays a series of notches in the frequency response. All three regions were accurately predicted by the model. Discrepancies between the results are likely a product of the simplifications made to the model geometry. However, these discrepancies are small, and thus the noise measurement comparisons further validate the accuracy and usefulness of the FD/BEM model.

Because of the close correlation between simulation and experiment, it becomes possible to utilize the model for a more thorough examination of the features in the graph and their causes. For example, for 95V bias, at 2.6 MHz, away from any notable features in the graph, the membranes in the array behave as expected. As shown in Figure 64 (a), only the element that was actuated exhibits any significant displacement.

However, as shown in Figure 64 (b)-(d), in the first region of interest a high degree of crosstalk is occurring and all membranes in the array are being excited with different phases and amplitudes. This explains the presence of large peaks and notches in the frequency spectrum. Depending on the frequency, crosstalk will either boost or decrease the noise current seen by the TIA. Many of the features in this region are very high-Q because the array displacement distribution is constantly shifting between completely different resonance modes. This in turn causes the admittance (and therefore noise spectrum) of the actuated element to dramatically change with different frequencies. Note that the two main peaks near 6 MHz and 9.5 MHz in Figure 63 (d) correspond precisely with the center frequencies that would occur if the two membranes geometries were isolated in immersion, as shown in Figure 65. This implies that much of the crosstalk in this region has higher magnitude than anywhere else in the spectral plot because the membranes are near their resonance in this location. Finally, it is also notable that in this region, the two largest magnitude, high-Q features occur because a periodic wave is being set up in a direction that is specific to a particular inter-membrane spacing. At 6.1 MHz for example, as shown in Figure 64 (b), the largest peak in the noise spectrum is created because a crosstalk wave is occurring along the second row of membranes. This row corresponds to a unique membrane pitch. Similarly at 9.7 MHz, as shown in Figure 64 (d), a crosstalk wave is propagating through the inner ring of the array.

The next region, as demonstrated in Figure 64 (e), represents the main band of the transducer array. It still exhibits crosstalk effects, but in general the effects are not as dramatic as in the previous region, likely because the frequencies have moved past the main resonances of the membrane geometries.

The final region, which occurs at approximately 20 MHz and beyond, displays a series of notches. Upon examining the array behavior at these notches, their source becomes clear. Once again, a periodic wave is being set up in a direction that is specific

to a particular inter-membrane spacing. For example, at 20 MHz, the first notch is caused by a crosstalk wave that is occurring along the second row of membranes (see Figure 64 (f)). In this case, however, the wave is creating an antisymmetric mode shape in the associated membranes which results in a null (rather than a peak, as before) in the frequency response. Similarly, the next notch that occurs at a slightly higher frequency is due to a wave propagating along the inner most ring of the array.

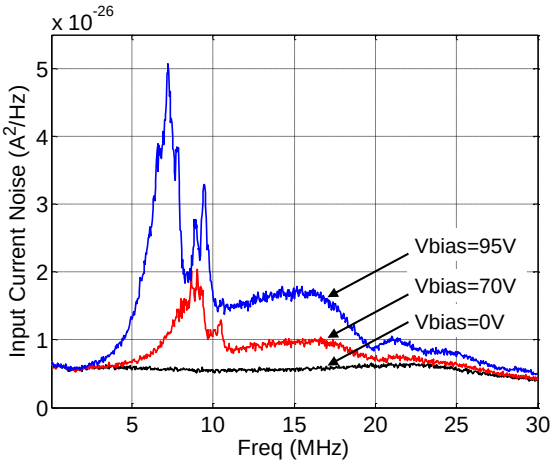


Figure 62. Experimentally measured noise spectrums of a single CMUT on CMOS receiver element in immersion at 3 different bias levels.

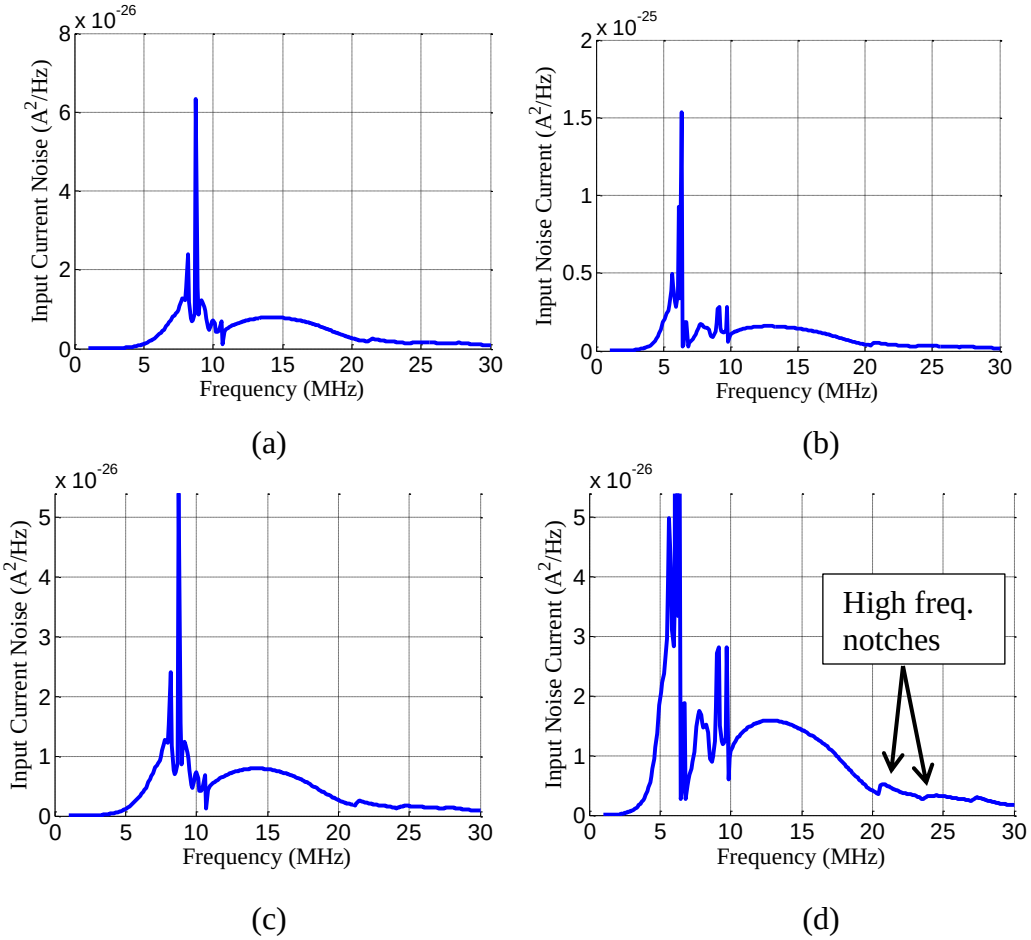


Figure 63. Simulated noise spectrums of a single CMUT on CMOS receiver element in immersion at (a) 70V and (b) 95V. (c) and (d) are the same respective plots at these 2 bias levels but with zoomed axes for better comparison with experimental data.

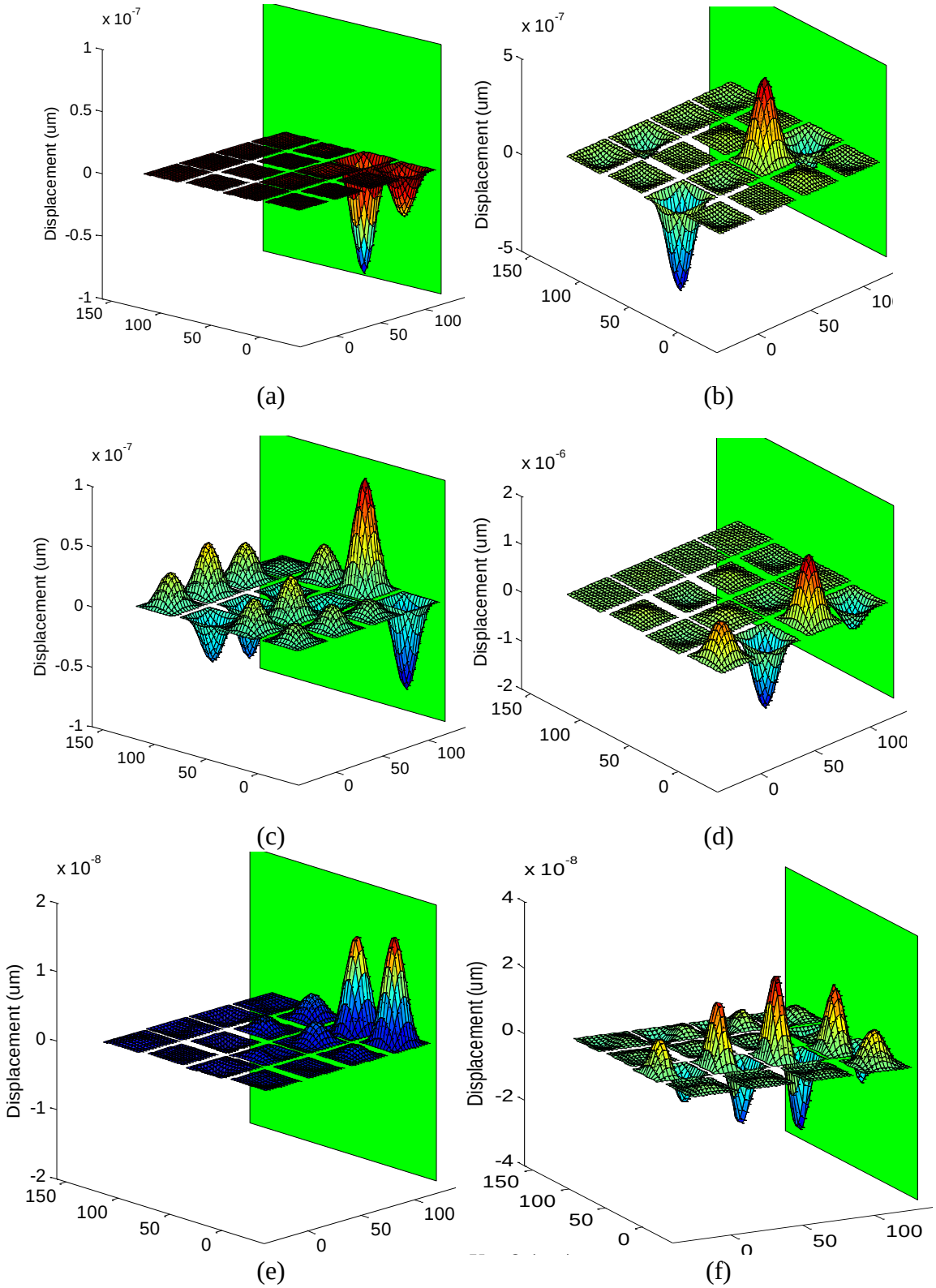


Figure 64. Simulated displacement of membranes in a dual ring array in immersion at (a) 2.5 MHz, (b) 6.1 MHz, (c) 9.7 MHz, (d) 9.73 MHz, (e) 15 MHz, and (f) 20.5 MHz for single element actuation (95 V bias). Green plane denotes symmetry.

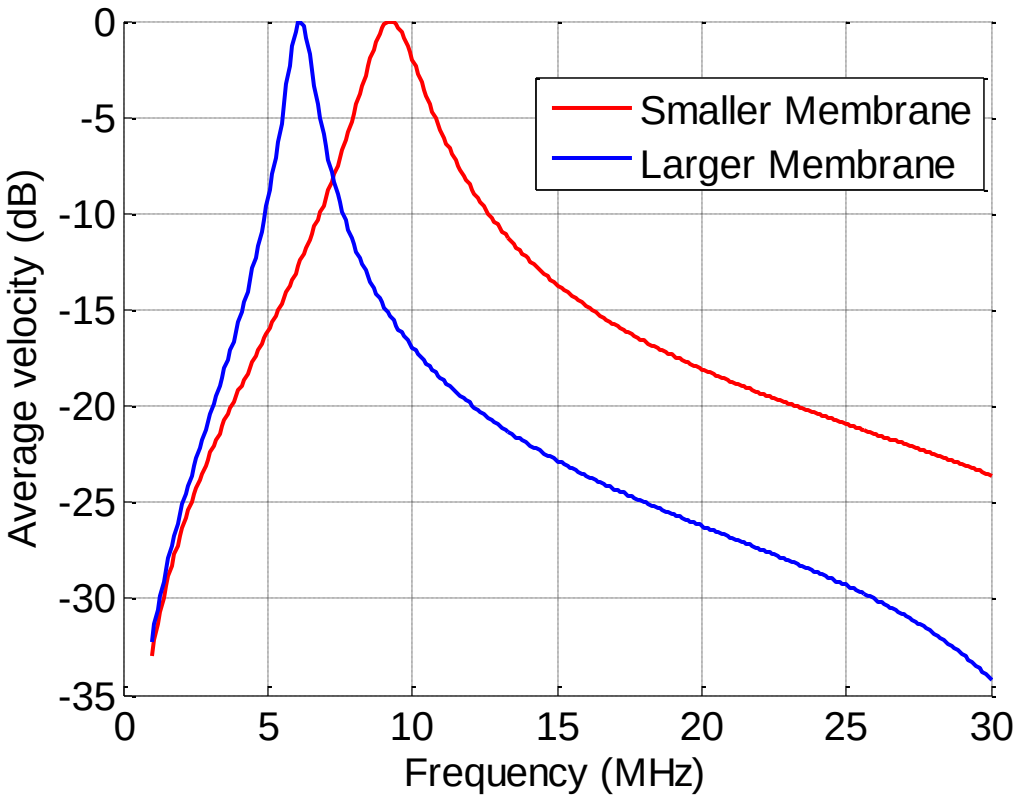


Figure 65. Average velocities of isolated membranes in water for the two membrane geometries used in the noise simulations.

CHAPTER 6

APPLICATION OF THE MODEL

6.1. Optimization of Membrane Aspect Ratio in Lateral Dimensions

Although the FD/BEM model is intended for simulation of large arrays, its efficient operation makes it ideal for optimizing single membrane performance by assessing many geometries in a very short amount of time. With this in mind, the model was utilized to determine the best rectangular membrane geometry to achieve the highest bandwidth. This was done by searching a design space of many different membrane lateral dimensions, while adjusting the membrane thickness accordingly to ensure that every design had the same center frequency in immersion. All membranes had 100% electrode coverage. The bandwidth of each geometry was then calculated using plots of average surface velocity. The results are summarized in Table 9 for an operating frequency of 9.4 MHz.

From this optimization study, it was concluded that a membrane's lateral dimensions are directly proportional to its bandwidth. In other words, increasing a membrane's size in one or both of its lateral dimensions will effectively increase its corresponding bandwidth. Another study considering 3 MHz operating frequency yielded identical results.

Figure 66 demonstrates the effects of increasing only one dimension (left) or both dimensions proportionally (right). As can be seen, when increasing only one dimension of the membrane, the bandwidth is improved in spite of the fact that anti-resonance notches are created closer to the center frequency. In fact, the bandwidth nearly triples in changing the dimensions from 18um x 18um to 18um x 100um. The notches in the plot originate from higher order mode shapes in the longer dimension. As discussed in Chapter 2, notches surrounding the main band of a transducer create ringdown effects in

the corresponding temporal response which degrades imaging performance. Thus, rectangular shaped membranes may not be optimal in spite of their bandwidth improvements. In contrast, increasing both dimensions proportionally results in a similar improvement in bandwidth without adding any notches to the main frequency band. The only difficulty introduced with large, square membranes is that they become increasingly difficult to reliably release during fabrication. Thus, there is likely an optimal size which maximizes bandwidth and can be easily fabricated.

With regards to how increasing the lateral dimensions of CMUT membranes will improve array performance, a separate study will be detailed in Section 6.2.5.

Table 9. Fractional Bandwidths (FBWs) for Different Membrane Geometries

Design	**Width (um)**	**Length (um)**	**Membrane Thickness (um)**	**Aspect Ratio in Lateral Dimensions**	**Max Aspect Ratio (Lateral/ Thick)**	**Center Freq (MHz)**	**FBW (%)**
1	18	100	1.05	5.6	95.2	9.45	36.8
2	18	70	1.00	3.9	70.0	9.38	31.6
3	18	50	0.96	2.8	52.1	9.44	25.3
4	18	30	0.85	1.7	35.3	9.40	17.8
5	18	18	0.63	1.0	28.6	9.25	13.3
6	35	35	1.97	1.0	17.8	9.30	23.3
7	45	45	3.05	1.0	14.8	9.40	28.7
8	55	55	4.30	1.0	12.8	9.41	33.9
9	35	70	2.75	2.0	25.5	9.30	34.3

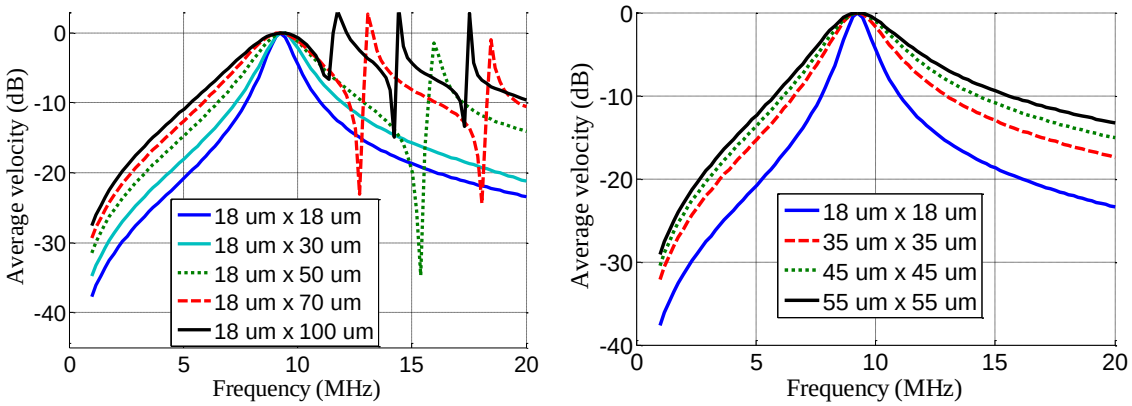


Figure 66. Effect of increasing one lateral dimension (left) or both lateral dimensions (right) on the frequency response of an isolated CMUT membrane in immersion

6.2. Design Study: Optimizing a Dual-Ring Array Element for Minimal Crosstalk

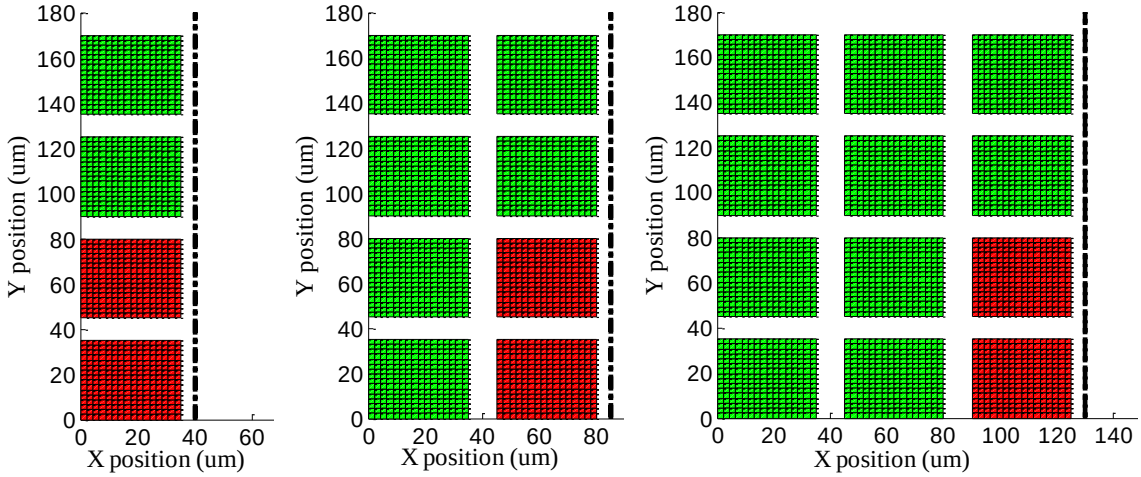
In order to highlight the capabilities of the FD/BEM model for CMUT array design applications, a study was conducted to optimize a dual-ring array element with the specific goal of minimizing unwanted acoustic crosstalk effects. The following sections will utilize the model's computational efficiency to iterate through a number of different design scenarios that hold potential for reducing crosstalk. This study is not intended to exhaustively search the design space, but is rather a preliminary investigation into potential techniques for optimizing array design.

For reference, the performance of the resultant arrays were compared with a standard DRA design which consists of 2 rings of equally spaced elements, where each element contains 4 membranes that have the same geometry of 35um x 35um x 2um. All membranes are assumed to be spaced 10um apart. The model predicts a collapse voltage of 102V and a center frequency (isolated, in immersion) of approximately 8.5 MHz for this particular geometry when biased at 75% of the collapse voltage.

6.2.1. Effects of Inactive Membranes

For efficient modeling, it is necessary to determine how many neighboring, inactive membranes are needed for accurate prediction of dynamic behavior of a single CMUT DRA element. Thus, a preliminary analysis was conducted that calculated the dynamic response of a single element with only a few neighbors, and then gradually increased the model size by increasing the number of neighboring membranes. The frequency responses of all models were then compared.

The initial model employed the geometry shown in Figure 67 (a). Note the use of half symmetry. It consisted of a single 4-membrane element, with only one other neighboring 4-membrane element positioned above it in the outer ring of the array. Both elements were given the same bias of 75% of the collapse voltage, but only the lower element was actuated with an AC voltage. With every new iteration of the model geometry, an additional column of membranes was added to the left and right of this original array of 2x4 membranes, as shown in Figure 67 (b) and (c).



(a) (b) (c)

Figure 67. Model geometries consisting of (a) 0 (b) 1 and (c) 2 inactive, neighboring membrane columns. Dashed line implies symmetry plane. These models were used for determining how many inactive columns should be included in the FD/BEM model for sufficient accuracy. All membranes are biased but only red elements are actuated with AC signal.

In this manner, models of arrays containing 2x4, 4x4, 6x4, 8x4, 10x4, and 12x4 membranes were simulated. Plots of the admittance and average velocity of the active element are shown in Figure 68 (left) and (right), respectively. From these graphs, it is clear that accurate predictions for a DRA element can be achieved over nearly the entire frequency spectrum as long as at least 2 neighboring columns are included on both sides the element. This observation also holds true when considering the inactive element directly above the active element. As long as these 2 neighboring columns are included, the smooth region surrounding the main band of interest and the higher frequency notches due to crosstalk-induced anti-symmetric mode shapes are both resolved with sufficient accuracy.

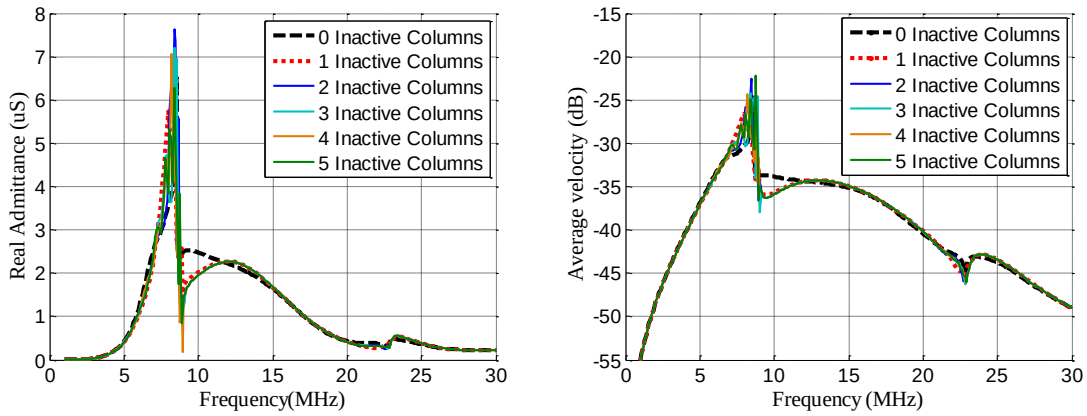


Figure 68. Simulated admittance (left) and average velocity (right) of an active DRA element with differing numbers of inactive neighboring columns.

The frequency response plots also reveal that even the largest model consisting of 5 neighboring membrane columns was insufficient to achieve a converged solution for the small region containing significant crosstalk effects near 8.5MHz, as demonstrated in Figure 69. Within this frequency range, the high-Q notches and peaks shift in both location and magnitude with every additional column. There are a number of reasons which likely contribute to this result. Primarily, this is the portion of the considered frequency range in which crosstalk effects are extreme enough to affect even the

outermost membranes of the modeled array, as shown in Figure 70 (b) and (c). In most other regions, the crosstalk effects are either negligible, as shown in Figure 70 (a), or very localized, as shown in Figure 70 (d). Thus, in these regions, the inclusion of only a few inactive membrane columns should be sufficient to accurately depict the dynamic behavior of the array. However, in the crosstalk-dominated region near 8.5MHz, all membranes in the array affect (and are affected by) the dynamic response of the active element. Thus, adding more columns adds new information to the system and can potentially change the entire behavior of the array in this frequency range. For example, additional columns introduce new inter-membrane diagonal pitches which create new potential directions for periodic crosstalk waves to occur.

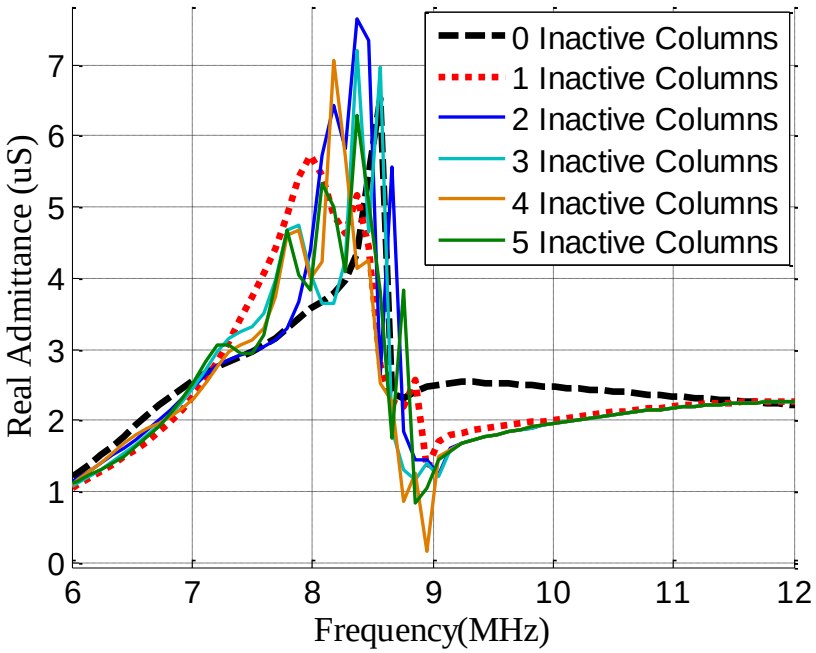


Figure 69. Zoom-in of the crosstalk dominated region of the simulated admittance for an active DRA element with differing numbers of inactive neighboring columns.

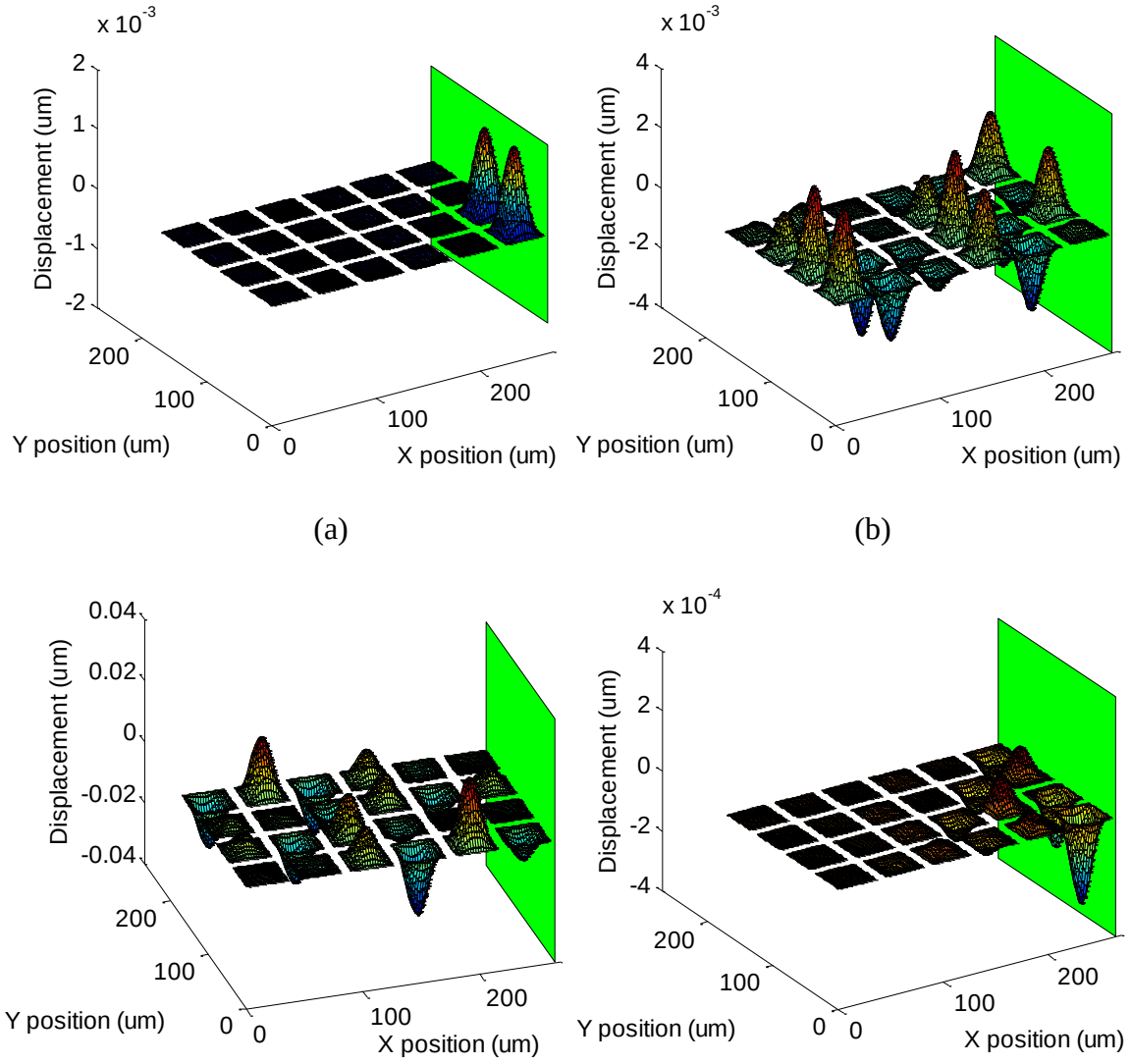


Figure 70. Simulated displacement of membranes in a dual ring array in immersion at (a) 4.8 MHz, (b) 8.1 MHz, (c) 8.8 MHz, and (d) 12 MHz for single element actuation (76 V bias). Green plane denotes symmetry.

It should be noted that the membranes on the outer columns are the least accurately modeled membranes in the array. This is because their boundary conditions do not reflect the fact that they should be surrounded on both sides with membranes. In the model, they only contain membranes on one side. This introduces an anti-symmetric loading that would not occur in reality. In most regions where crosstalk effects are relatively localized, these inaccuracies likely have minimal effect. However, in the

crosstalk dominated region, where the outermost membranes play a significant role in the dynamic response, these inaccuracies should be considered.

It is interesting that the higher frequency notch due to anti-symmetric membrane mode shapes, as shown in Figure 71, is not localized and yet can also be accurately predicted with only 2 neighboring columns. This is likely due to the very specific direction-dependence of the periodic wave that is being set up. These membrane mode shapes can only be created in directions perpendicular to the lateral dimensions of each membrane, and are essentially dependent only on the inter-membrane pitch in these directions (assuming all membranes have same geometry and bias). The inclusion of 2 inactive columns seems to be sufficient to set this pitch dimension and allow for propagation of the periodic wave. Adding more columns will allow the wave to propagate further outward, but will not change the inter-membrane pitch, and so the frequencies at which the associated notches occur will not shift significantly. Furthermore, the associated periodic wave is evanescent so the displacement magnitude of neighboring membranes dies off quickly as the pressure wave propagates out from the active element. This implies that the magnitude of the admittance or velocity for the active element will not change significantly as more inactive columns are added.

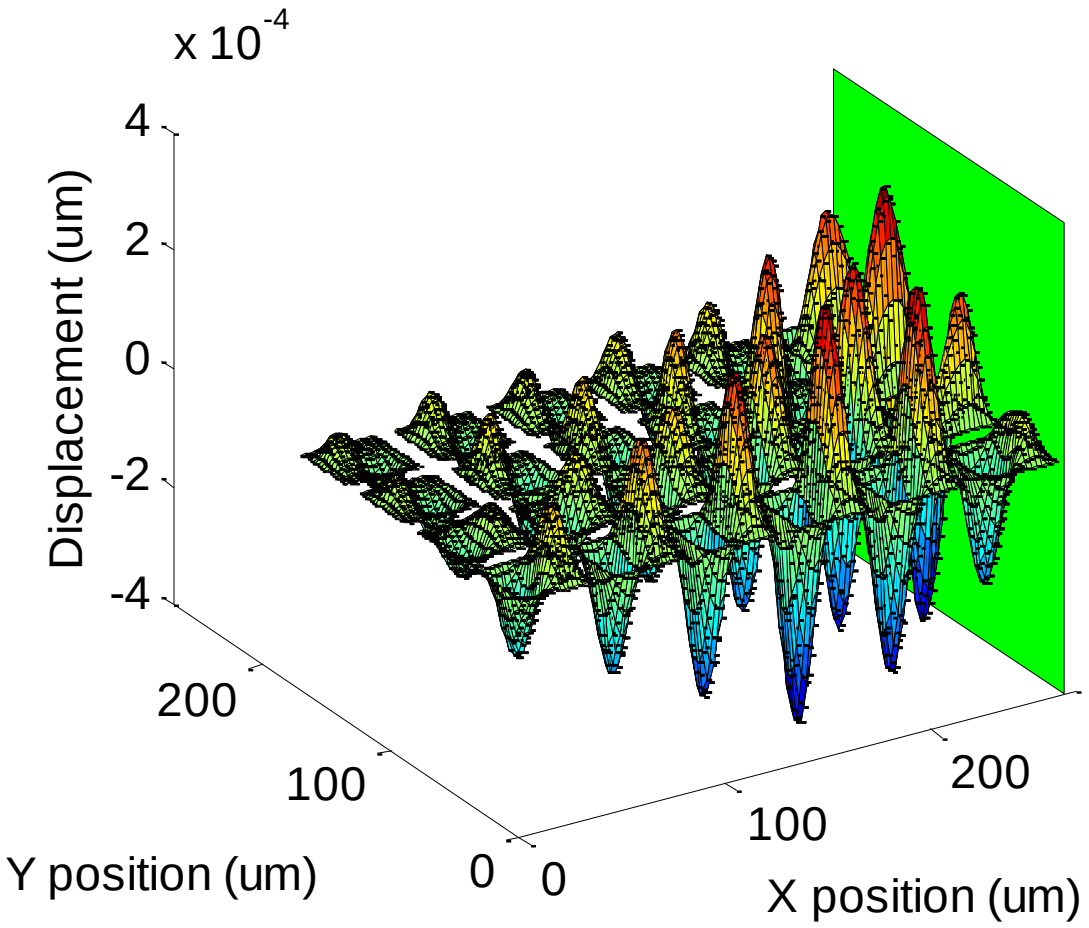


Figure 71. Simulated displacement of membranes in a dual ring array in immersion at 23 MHz for single element actuation (76 V bias). Green plane denotes symmetry.

The results from this study also imply that the predicted pressure magnitudes for a DRA element should be accurate for most portions of the frequency response as long as 2 inactive columns are included in the model. Since crosstalk is relatively localized for most of the frequency response, the total pressure output from the array will originate mainly from the active element and its closest neighbors. However, it also implies that pressure magnitudes in the crosstalk-dominated region will be incorrect when partial array models are used, because the significant contributions from outer membranes will be neglected.

For most purposes, a few neighboring columns will be sufficient to simulate the dynamic response of a DRA element. This will accurately predict the shape of the frequency response, its associated magnitudes and center frequency, and the general location of the region of high crosstalk. However, if a high degree of accuracy is needed regarding the magnitude and locations of the peaks and notches in the crosstalk-dominated region, many more membrane columns should be included in the model. Unfortunately, processing limitations of the laboratory computer used for these simulations would not allow for more than 5 neighboring columns to be considered. Thus, the exact number of columns needed for complete accuracy could not be determined. It should be noted that larger models would also have to account for the curvature of the dual-ring array, and could no longer simplify the system into equally-spaced, vertical columns.

6.2.2. Non-Periodic Membrane Spacing

Many of the high-Q peaks and notches due to acoustic crosstalk are dependent on the periodic spacing between membranes. Thus, it is logical to deduce that by interrupting this periodicity, the severity of these effects can be reduced. With this in mind, a design study was performed on a single row of 10 membranes. Each membrane had dimensions of 35um x 35um x 2um. All membranes were biased at 75% collapse but only the outer element was actuated with an AC voltage (Figure 72). The inter-membrane spacing between successive membranes (starting from the active element) was increased by a fixed number. This fixed number was set in 4 separate model iterations as 0um, 2um, 4um, and 10um.

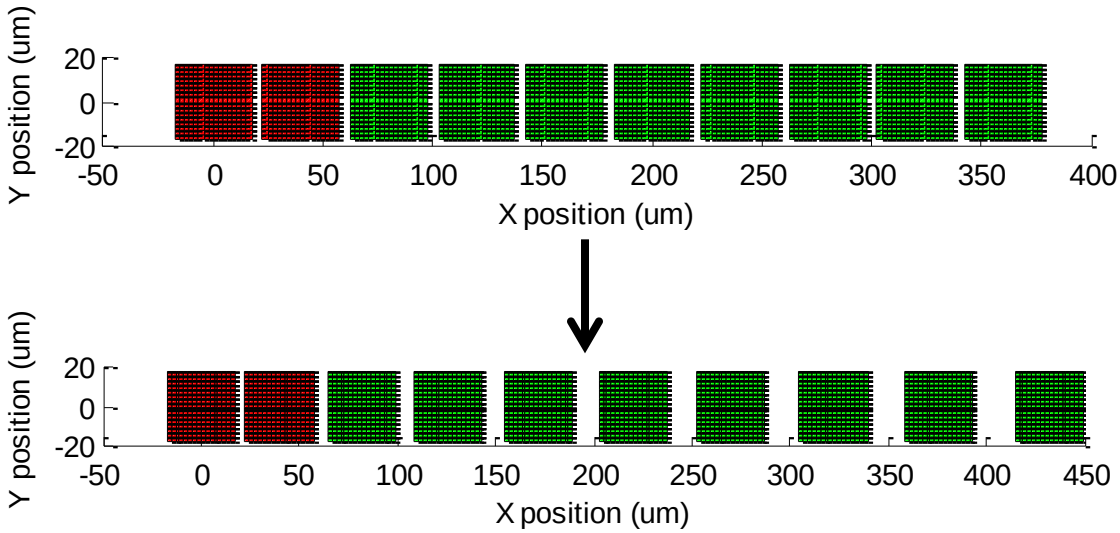


Figure 72. Model geometry for a horizontal array of 10 membranes with an active outer element. The inter-membrane pitch between successive membranes (starting from the active element) was increased by a fixed number, as shown in the bottom figure.

The average velocity plots of the active membranes in the geometry shown in Figure 72 are displayed in Figure 73, respectively. These plots demonstrate that using a non-periodic pitch may be an effective means for reducing acoustic crosstalk. Increasing the degree of periodicity disruption decreases the number and magnitude of the sharp features in the band of interest. However, it should be noted that this technique requires sacrificing active area within the array; a fact which must be carefully considered during array design. Furthermore, a relatively large spacing increase was required before a noticeable reduction in crosstalk effects was observed. Therefore, although this study indicates that there is potential in this design strategy, a more thorough analysis on a full DRA element model would be necessary to truly optimize array operation.

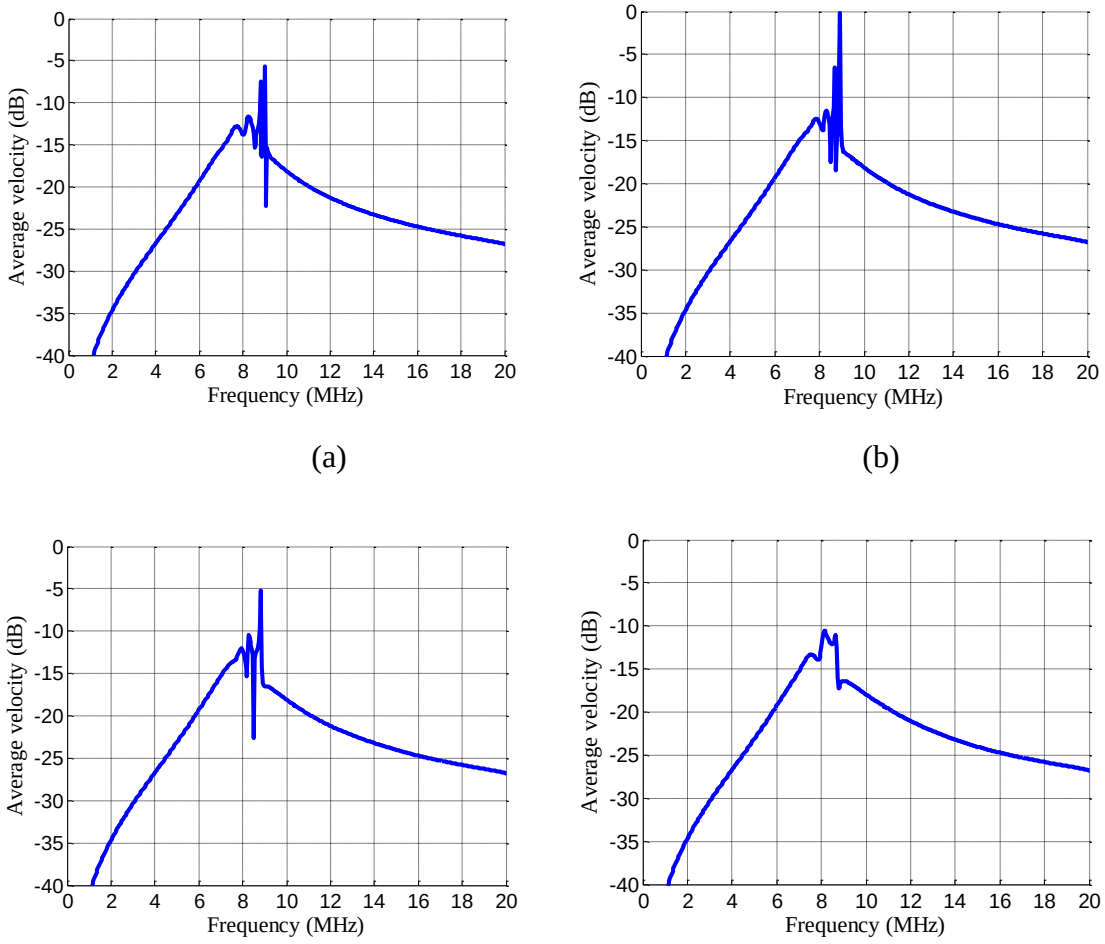


Figure 73. Simulated average velocity plots for the model geometry shown in Figure 72, when the inter-membrane spacing is increased by (a) 0um, (b) 2um, (c) 4um, and (d) 10um with every successive membrane.

6.2.3. Optimizing the Inter-Membrane Pitch

As suggested by [45], there may be an optimal inter-membrane pitch that maximizes the performance of a CMUT array. Thus, a design study was performed to evaluate the viability of utilizing pitch as an optimization parameter. A single line of 10 equally spaced membranes was considered, as shown in Figure 74. Each membrane had dimensions of 35um x 35um x 2um. With each successive model iteration, the spacing between the membranes was varied from 5um to 35um. The pitch between diagonally

aligned membranes was also considered, as shown in Figure 75. All membranes were biased at 75% collapse, but only the red membranes were actuated with an AC voltage.

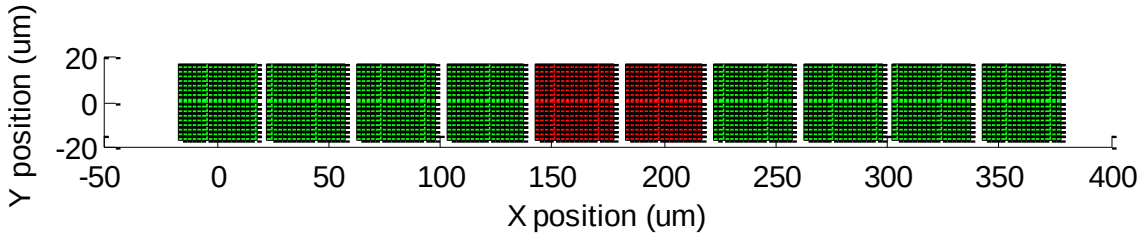


Figure 74. Model geometry used to determine an optimal pitch for a single line of 10 equally spaced membranes.

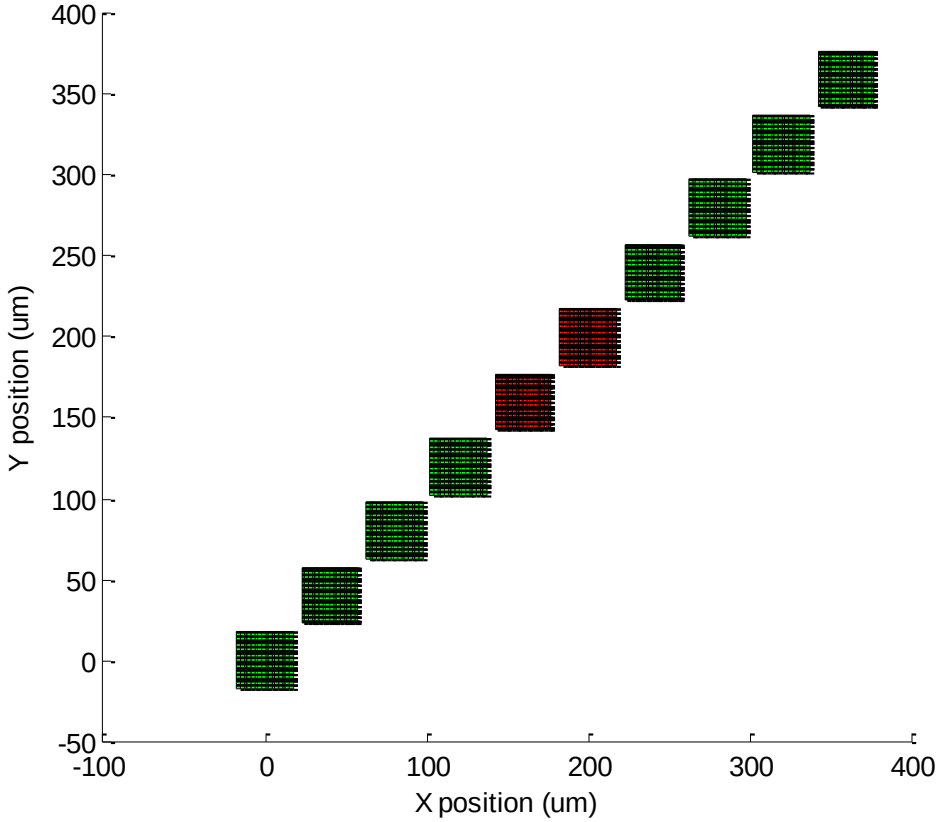


Figure 75. Model geometry used to determine an optimal pitch for a diagonal configuration of 10 equally spaced membranes.

The resultant average velocity plots for the active elements of the horizontal arrays are shown in Figure 76. The velocity plots for the diagonal array are shown in

Figure 77. These plots indicate that for a given configuration of membranes, there is an optimal inter-membrane spacing. In general, the larger the distance between membranes, the greater the reduction in crosstalk effects will be. For the horizontal array, the unwanted crosstalk peaks and notches virtually disappear when a spacing of 35um (a full membrane width) or more is utilized. For the diagonal pitch, these features vanish with a slightly smaller spacing. However, increasing the spacing decreases the fill factor of the array and will have a negative impact on the overall sensitivity of the array. Therefore, there is likely an optimal spacing which will minimize crosstalk without sacrificing too much of the active area. A more thorough analysis of a full DRA model would need to be performed to determine this optimal point.

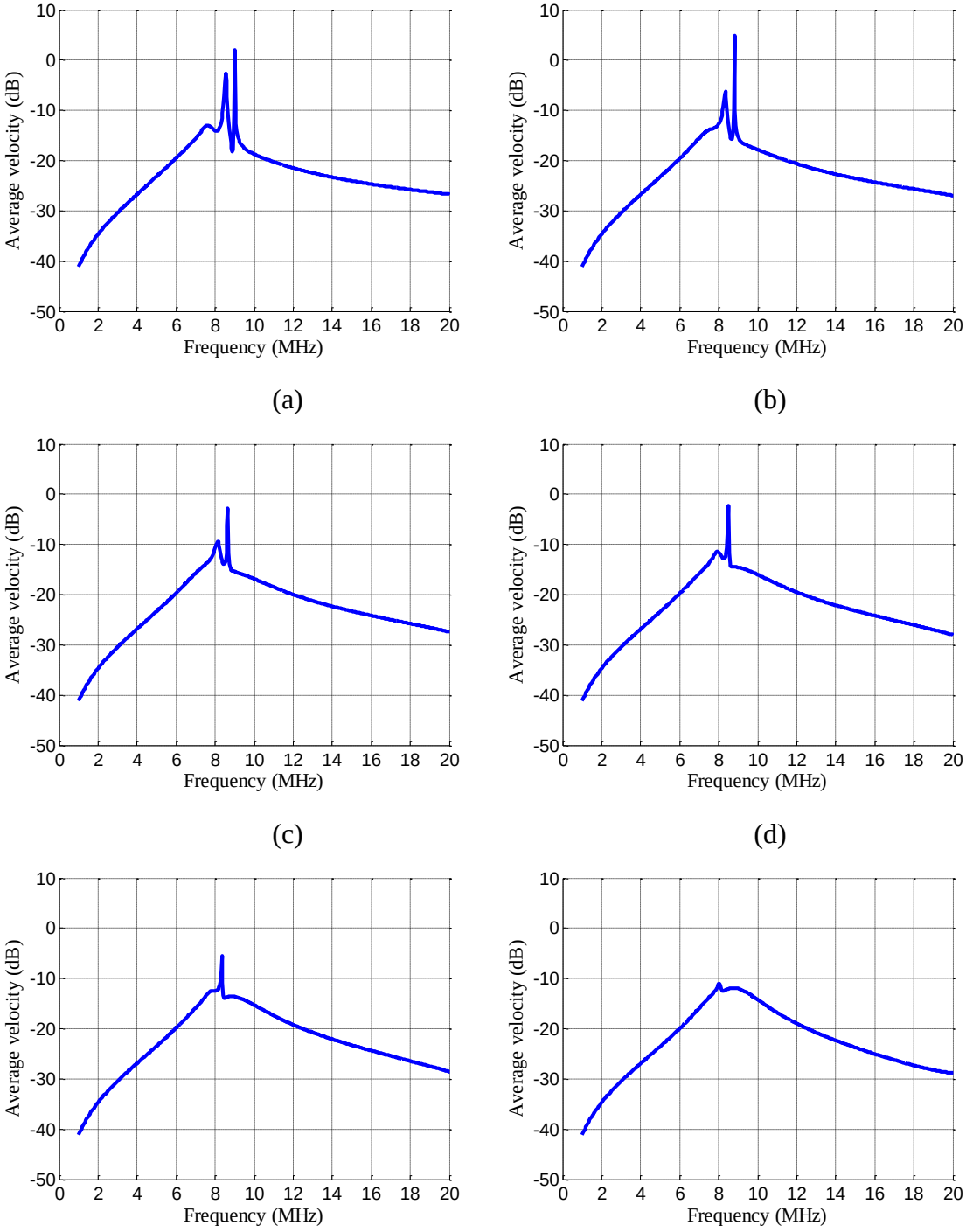


Figure 76. Real average velocity plots for inter-membrane spacings of (a) 5um, (b) 10um, (c) 15um, (d) 20um, (e) 25um, and (f) 35um for the model geometry shown in Figure 74.

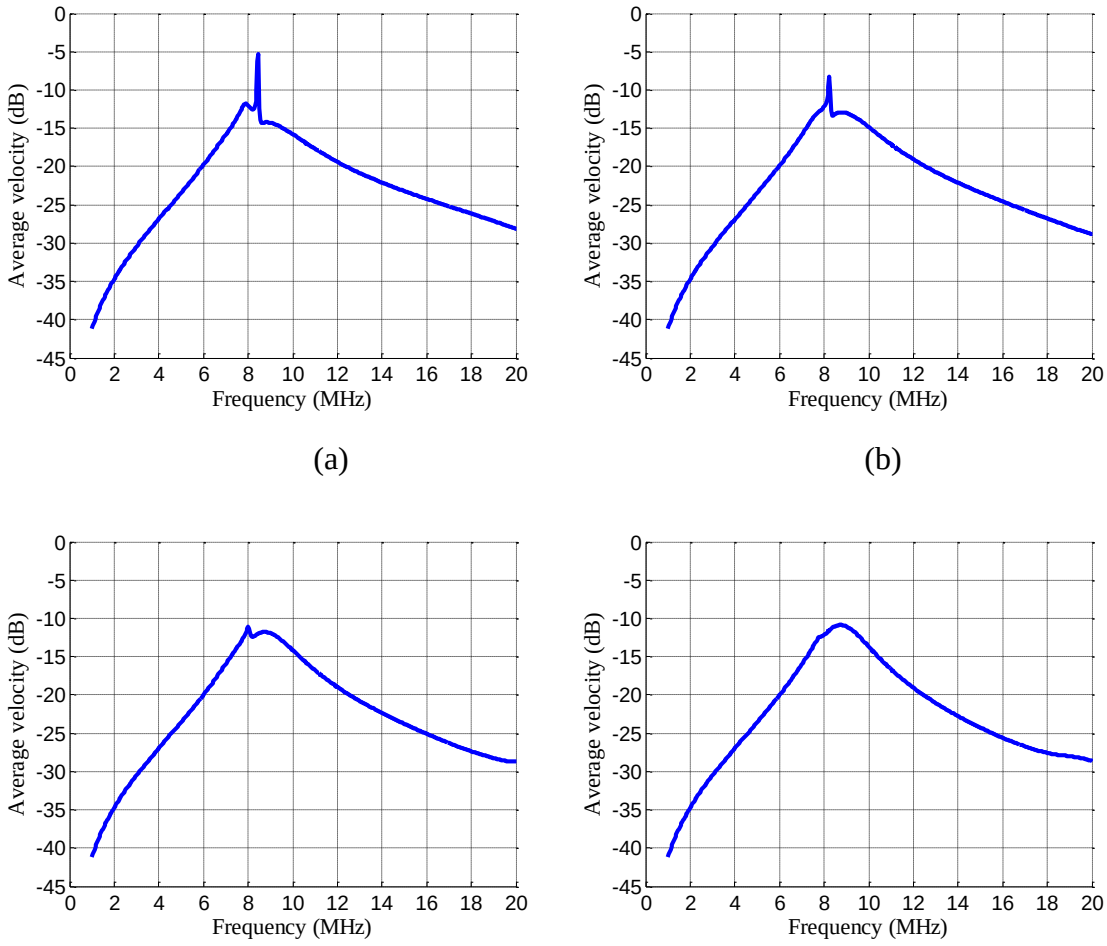


Figure 77. Average velocity plots for equal inter-membrane spacings in both X and Y directions of (a) 5um, (b) 10um, (c) 15um, and (d) 20um for the model geometry shown in Figure 75.

6.2.4. Optimizing the Transmit Element Locations

The elements of the separate Tx and Rx rings of the DRA do not have to be vertically aligned as they have been in the previous design cases. Thus, as another measure for interrupting periodicity, a design study was conducted to determine if there is an optimal orientation for the Tx ring relative to the Rx ring. This was done by shifting the upper 2 rows of the DRA model horizontally with respect to the lower 2 rows, as shown in Figure 78. 5 iterations were performed in which the top elements were shifted by 0um, 5um, 10um, 15um, and 22.5um (half the inter-membrane pitch).

The average velocity plots for a DRA with Tx ring shifted by 0um, 5um, 10um, 15um, and 22.5um can be found in Figure 79 (a) – (e), respectively. From these results, it seems that simply rotating the Tx ring with respect to the Rx ring will have little effect on array crosstalk. In fact, shifting the ring from the standard configuration slightly increases the maximum amplitude of the largest peak in the crosstalk-dominated region. In any case, this technique holds little potential for reducing crosstalk and need not be investigated further.

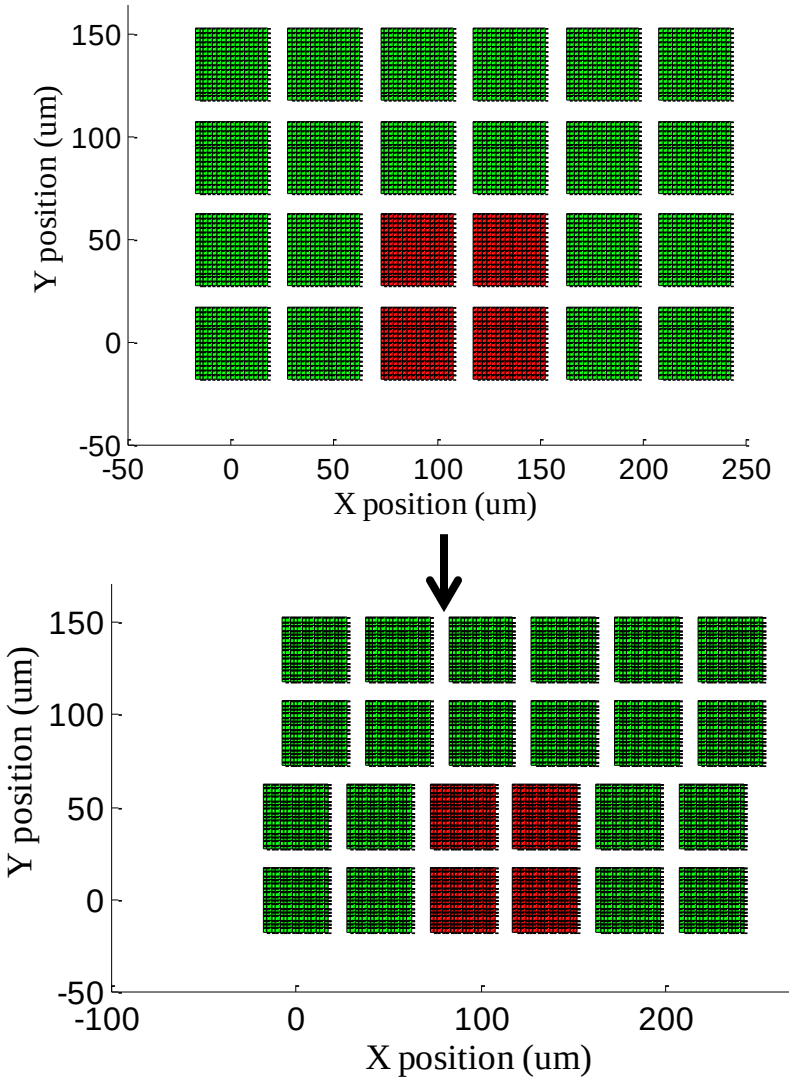
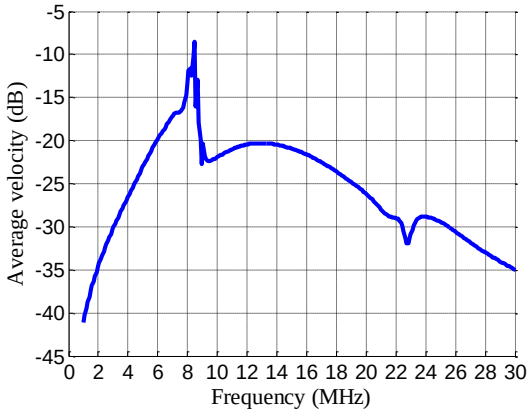
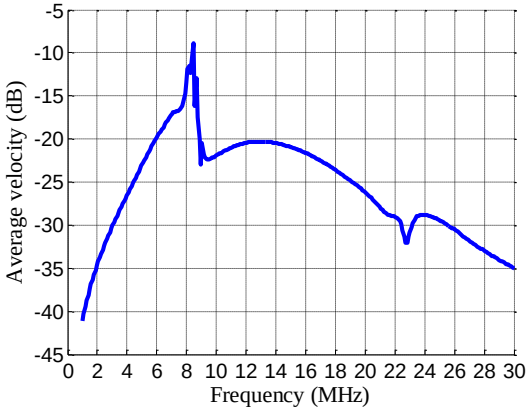


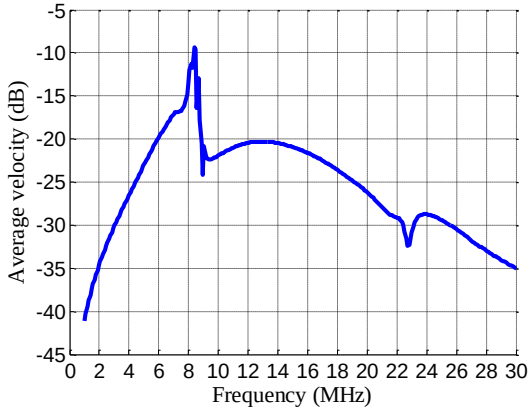
Figure 78. Model geometry used to determine an optimal orientation for the Tx ring. Each iteration of the simulations horizontally shifted the upper ring from its original position (top) to a new offset position (bottom)



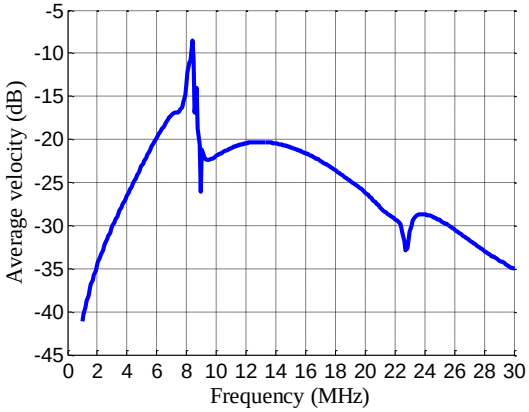
(a)



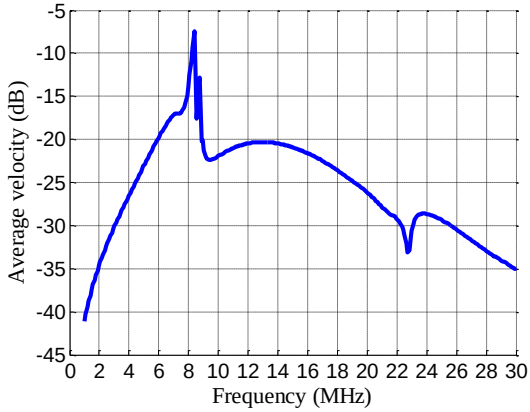
(b)



(c)



(d)



(e)

Figure 79. Average velocity plots for DRA with Tx ring shifted horizontally by (a) 0um, (b) 5um, (c) 10um, (d) 15um, and (e) 22.5um

6.2.5. Number of Membranes in an Element

The standard DRA design contains 4 membranes per element. However, this is not a rigid requirement, and it is likely that reducing the number of membranes per element will reduce crosstalk effects by reducing the total number of neighboring membranes. This design change could potentially have other beneficial side effects as well. For example, if the number of membranes per element is reduced to one, then the unwanted phenomenon of membranes within an element vibrating out of phase at specific frequencies will be eliminated. Also, utilizing fewer membranes per element would require larger membranes to fill the same area, and since we know from Section 6.1 that these membranes will have larger bandwidths, this could translate to a larger bandwidth in array operation as well. Finally, increasing the number of membranes in an element effectively decreases the active area because more space must be occupied by the membrane support posts. Thus, using fewer membranes may allow for an improvement in sensitivity, which would imply that smaller 1-membrane elements could achieve the same efficiency as larger 4-membrane elements.

To test the validity of these hypotheses, a design study was conducted on separate DRA models containing four 35um x 35um x 2um membranes, two 35um x 80um x 3um membranes, one 55um x 55um x 4.4um membrane, one 70um x 70um x 6.8um membrane, nine 20um x 20um x 0.78um membranes, and nine 25um x 25um x 1.15um membranes per element (Figure 80) to determine what improvements, if any, would result. All models utilized an inter-element pitch of 90um.

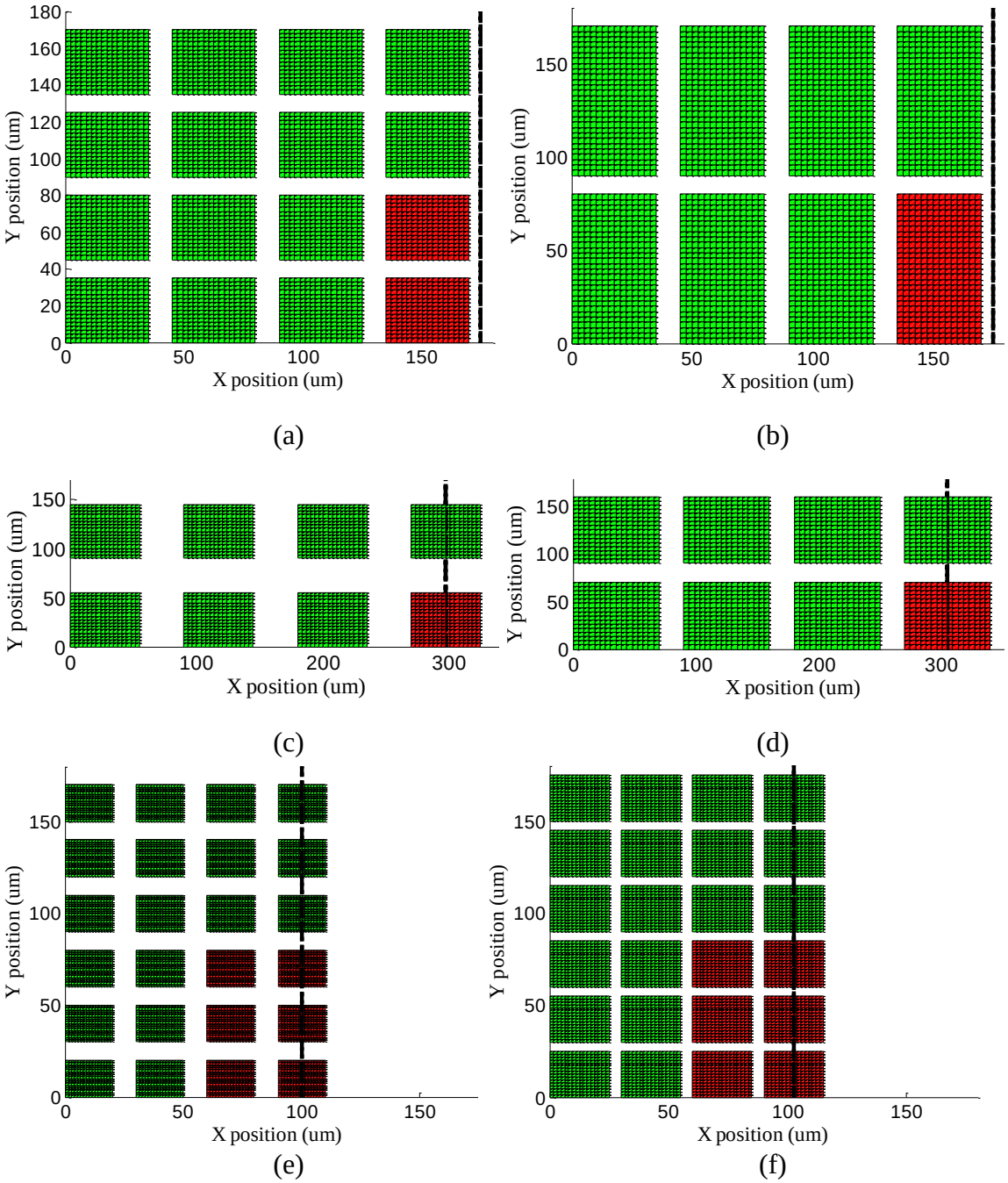


Figure 80. Model geometries used for testing the effects on crosstalk with (a) four 35um x 35um x 2um membranes, (b) two 35um x 80um x 3um membranes, (c) one 55um x 55um x 4.4um membrane, (d) one 70um x 70um x 6.8um membrane, (e) nine 20um x 20um x 0.78um membranes, and (f) nine 25um x 25um x 1.15um membranes per element. Note the use of half symmetry. All models have an inter-element spacing of 90um. Red membranes are active.

The average velocity plots for the active membranes of the respective model geometry in Figure 80 can be found in Figure 81. As would have been expected from the previous analysis conducted in Section 6.1, the rectangular membranes used for the 2-membrane-elements introduce a large peak in the frequency response due to a higher order resonance in the longer lateral dimension. As a result, this particular array design experiences a degradation in bandwidth due to the extra peak without significantly improving the original crosstalk features of the standard DRA design. Thus, 2-membrane rectangular elements are not a desirable option for reducing crosstalk in dual ring arrays.

On the other hand, both 1-membrane element arrays demonstrate a noticeable decrease in the Q-factor of the crosstalk features near 8.5 MHz as compared to the standard DRA design. Furthermore, using a small membrane for the 1-membrane elements, even though it decreases the membrane and electrode size, will result in increased velocity amplitude without significantly altering the crosstalk features. However, a drawback to this design is that the main band of the device has now shifted to the crosstalk-dominated region. Thus, even though crosstalk features have been reduced, the remaining crosstalk features cause a reduction in total bandwidth of the device when compared to the standard DRA.

Finally, using more membranes per element has a much different effect. It actually increases the bandwidth around the less crosstalk-dominated region at 15 MHz. This is partially caused by the fact that the anti-symmetric resonance notches have shifted even higher in frequency due to the smaller inter-membrane pitch. For example, Figure 81 (f) demonstrates comparable velocity magnitudes to Figure 81 (a), but with a much larger bandwidth around the main peak (and a slightly higher center frequency). However, in the crosstalk-dominated region, the sharp peaks and notches have increased in quantity. This is most likely a result of the increased number of membranes, which enables more potential resonance modes in the array.

In conclusion, the results of this design study imply that varying the number of membranes per element can have a significant impact on the frequency response of a dual ring array. Reducing the number of membranes per element decreases the sharpness of crosstalk features, but degrades the bandwidth of the device. Increasing the number of membranes effectively increases the bandwidth around the main peak, but also undesirably enhances the features in the crosstalk-dominated region. A more thorough investigation would be needed to form any final conclusions on the optimal number of membranes per element. That investigation should iterate through many different designs that vary not only the number of membranes per element, but also the membrane size, element size, inter-membrane pitch, and inter-element pitch. Furthermore, the improvements in the shape of the frequency response should be considered in conjunction with the resultant effects on other important performance indicators such as sensitivity, radiation pattern, and maximum pressure output.

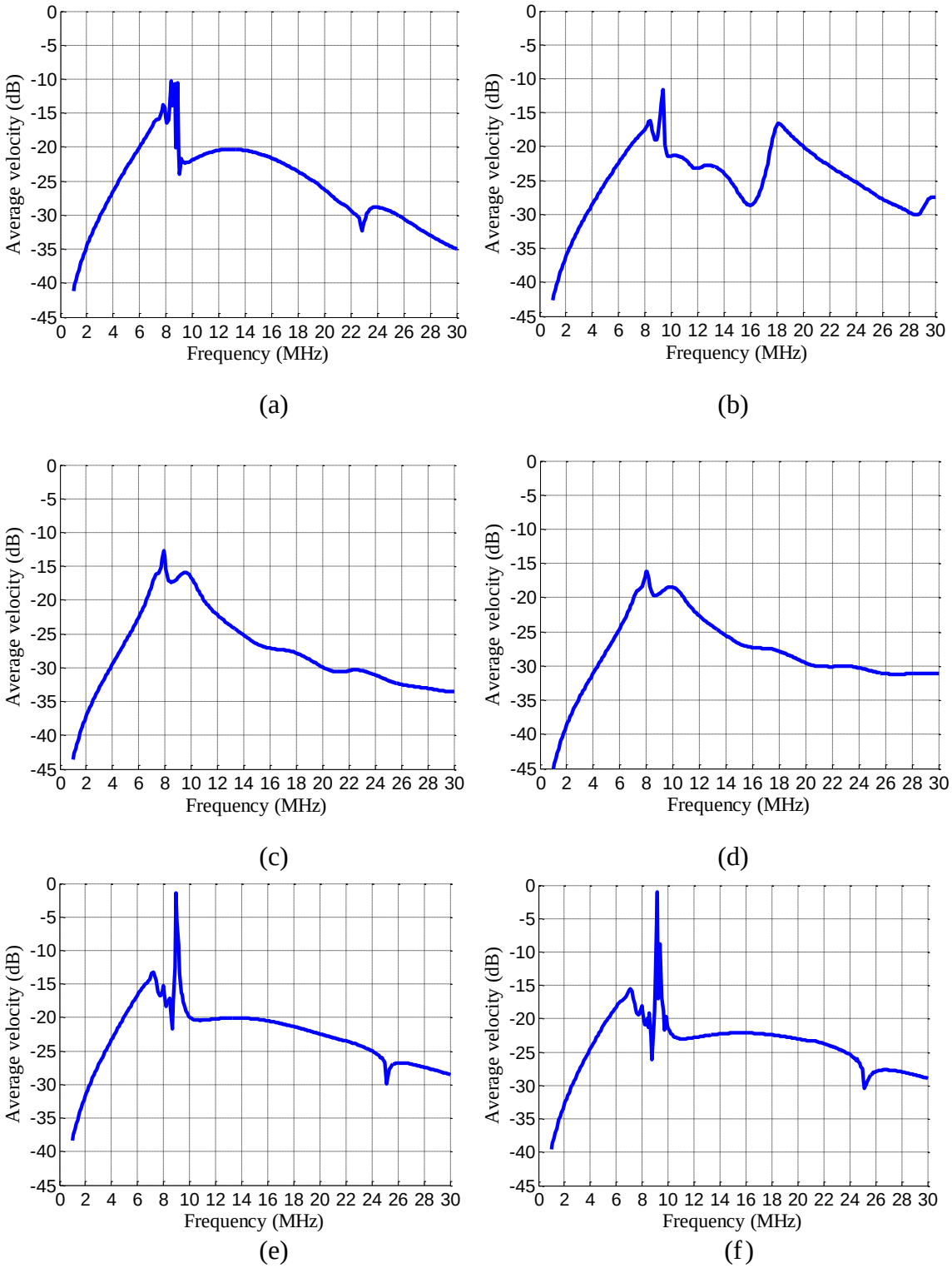


Figure 81. Average velocity plots for the respective model geometry shown in Figure 80

6.2.6. Collapsing the Transmit Ring

One method for reducing crosstalk in an array could utilize the dual-biasing capabilities of the DRA design. Collapse-mode operation of CMUTs has been suggested for generating large pressure signals in transmit [46, 47]. However, collapse-mode operation in a DRA could have the added benefit of preventing the entire outer transmit ring of membranes from contributing to acoustic crosstalk. These membranes would be relatively clamped in place, and would thus be unable to vibrate in response to neighboring pressure waves.

To determine the degree of crosstalk reduction that this mode of operation could generate, two model geometries were compared. One utilized the standard dual-ring structure, as shown in Figure 82 (left), while the other one contained only a single ring, as shown in Figure 82 (right). This second geometry assumes that the collapsed transmit ring will not contribute to acoustic crosstalk. All membranes were biased at 75% collapse and the membranes shown in red were actuated with an AC signal. The total admittance over the active element was calculated for both design cases.

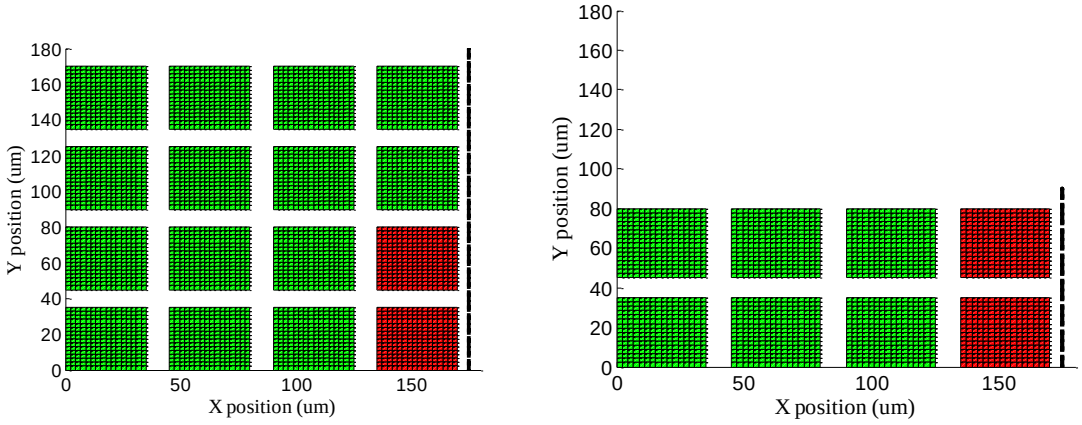


Figure 82. Model geometries for a standard dual ring array (left) and a dual ring array with a collapsed Tx ring (right).

The resultant plots of average velocity of the active element for both design cases are presented in Figure 83. The right graph in this figure shows a magnified version of the crosstalk dominated region. From these plots, it seems clear that collapsing the Tx ring will have a beneficial effect on acoustic crosstalk within the array. In the crosstalk-dominated region around 8.5 MHz, the number of peaks and notches is significantly decreased. Furthermore, the Q-factor of these features is also noticeably reduced resulting in a much smoother curve. Finally, the notch near 9 MHz was also reduced in severity by collapsing the Tx ring, which allowed for a slight improvement in bandwidth around the main peak at 12 MHz. These improvements will translate to shorter response times and less ringing in the temporal domain for the CMUT array. It should be noted that the velocity for the collapsed Tx ring case does have a larger magnitude peak in the crosstalk-dominated region compared to the standard DRA case. However, this peak has a lower Q than the peaks located on the velocity curve for the standard DRA design, which should translate to less drastic temporal domain ringing.

It is also interesting that collapsing the Tx ring had virtually no effect on the velocity curve outside of the crosstalk-dominated region. As mentioned previously, this is likely because crosstalk effects are negligible or very localized in these locations, so eliminating neighboring membranes will have minimal impact on the resultant response.

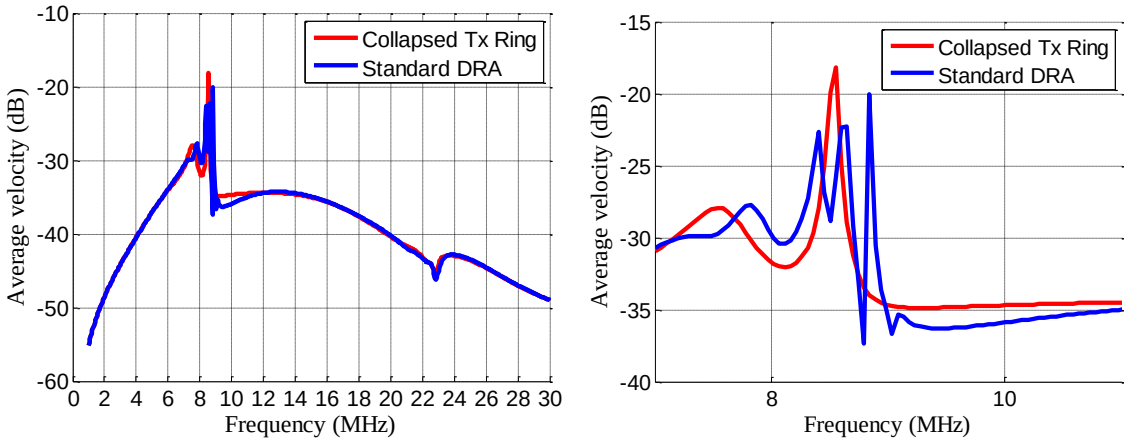


Figure 83. Simulated average velocity plots for a DRA receive element with and without collapsing the TX ring. The right graph is a zoomed-in version of the original plot (left) for better visualization of the crosstalk region.

6.3. Investigating the Potential of CMUT Noise Detection for Fluidic Sensing

In some circumstances, the sharp features created in the spectral response by crosstalk effects may be advantageous. These high Q features would be ideal for precision sensing of fluid properties, because acoustic crosstalk is directly related to the characteristics of the surrounding environment. Furthermore, as described earlier, CMUTs integrated with low noise amplifiers can be employed to detect the shifts in the crosstalk peaks using noise current measurements. This would allow for a passive sensing mechanism which would not disturb the fluid environment, and would only require a static bias on the CMUT membranes.

Since experimental tests have already verified that the FD/BEM model can accurately predict the noise current spectrum, simulations were conducted to test the capabilities of such a noise-based passive fluid sensor. The model geometry shown in Figure 84 was used for all simulations. Note the use of half symmetry. All membranes had dimensions of 35um x 35um x 2um with a 0.12um gap and 0.4um of nitride isolation. All membranes were biased at 60V, and the noise current was calculated using the method described in Section 5.2.3.3.

The precision of the noise fluid sensing mechanism was assessed by calculating the change in resonance frequency of a particular high-Q crosstalk feature with a 1% change in fluid impedance. This was accomplished by changing the speed of sound in the simulated fluid environment. The results are plotted in the left graph of Figure 85, with a zoomed-in version of the main region of interest on the right. These figures validate the assumption that noise spectrums can be used to detect minute changes in the characteristics of the fluid environment. The high-Q crosstalk features demonstrate a distinct shift in their center frequency when the speed of sound of the fluid changes by only 1%. This type of change would be difficult to detect in larger bandwidth peaks. Furthermore, this implies that these fluid properties can be detected passively, without the need for disturbing the medium.

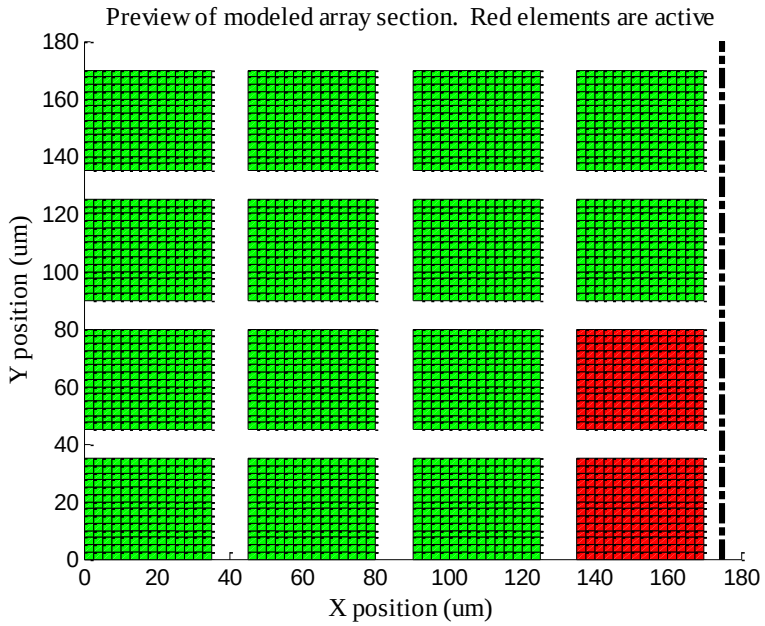


Figure 84. Model geometry used for investigating potential of using CMUT noise current measurements for passive fluid sensing. Active elements (excited with AC signal) are shown in red, while inactive elements are shown in green. Symmetry axis is shown as dashed line.

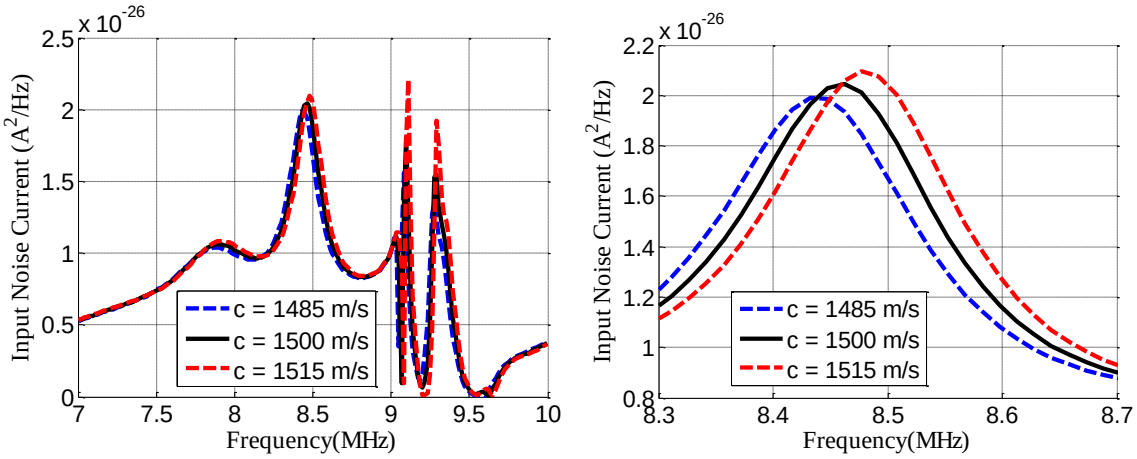


Figure 85. The effects of increasing or decreasing the speed of sound of a fluid by 1% on the location of the resonance frequency of a crosstalk feature in the noise current spectrum. Graph on right is zoomed in version of original (left).

CHAPTER 7

CONCLUSION

7.1. Summary and Conclusions

Capacitive Micromachined Ultrasonic Transducers (CMUTs) have demonstrated potential to advance the current state of medical ultrasound imaging beyond the capabilities of the currently employed piezoelectric technology. Because they are fabricated using micromachining techniques, they can easily achieve small, complex geometries, densely populated arrays, tight electronics integration, and large yields through batch processing. Consequently, these advantages and their inherently high bandwidth make them a more suitable choice for advanced ultrasound applications such as high-frequency or forward-looking IVUS compared to piezoelectric transducers.

In order to optimize CMUT transducers, an accurate, efficient model is needed. Unfortunately, acoustic crosstalk effects add significant difficulty to the task of creating such a model. Due to the mutual dependence between each membrane in an array, any accurate model must include all active membranes and many, if not all, of the surrounding inactive membranes. Finite Element Modeling (FEM) is not ideally suited for such large-scale computations. Thus, a more efficient analytical model based on the work of Meynier et al. [1] was developed and characterized in this thesis. It employs a finite difference (FD) approximation of thin plate equations to describe the stiffness of the membrane, and a Boundary Element Matrix (BEM) based on the Green's function for a baffled point source to account for fluid coupling. This reduces the modeled array to a system of force balance equations applied to a simple 2D nodal mesh, which can subsequently be solved with basic matrix operations. As a result, this model is capable of simultaneously calculating the dynamic displacement of hundreds of membranes in an array, with accuracy comparable to FEM but dramatically decreased computation time.

Comparison of calculations from FEM and the FD/BEM model for single membranes and small arrays revealed comparable accuracy to within approximately 5% difference when the lateral dimensions of the considered membranes were at least one order of magnitude larger than their corresponding thicknesses. This accuracy applied to predictions of collapse voltage, center frequency, bandwidth, pressure magnitudes, and the overall shape of the frequency response. For lower aspect ratio membranes, the thin plate equations are no longer valid and they inaccurately stiffen the membranes. The useable range of aspect ratios for the thin plate equations can be somewhat extended by artificially thinning the membranes in the FD/BEM model to match the center frequency predicted by FEM for a single membrane in immersion. This was shown to generate more accurate predictions of center frequency and bandwidth for array operation of low aspect ratio membranes.

Comparison with experimental results revealed that the FD/BEM model could accurately calculate collapse voltage, and the impedance and admittance curves at nearly all biases up to collapse. Furthermore, model calculations for the frequency response in the far-field of various array designs closely correlated with hydrophone measurements. The model accurately predicted the center frequency and the location and magnitude of notches and peaks caused by array crosstalk. It also demonstrated its accuracy in determining how crosstalk would change the radiation patterns of the array as compared to a case in which crosstalk was neglected and all active membranes vibrated as ideal pistons. Finally, experimental tests revealed that the FD/BEM model could also be used to calculate noise current spectrums.

The degree of correlation with experimental results was sufficient to justify implementing the model for various design applications. An investigation which iterated through various geometries revealed that larger lateral dimensions effectively increase the bandwidth of a single membrane. It also showed that square membranes are preferable to rectangular membranes, because higher order mode shapes in the longer dimension of a

rectangular membrane create unwanted features in the frequency response. The FD/BEM model was also utilized for a design study on dual ring array (DRA) design optimization. This study served as a preliminary investigation into the potential of various techniques that could be employed for reducing array crosstalk. The results of this study revealed that disrupting array periodicity, increasing the inter-membrane pitch, operating the transmit ring in collapse-mode, and altering the number of membranes per element can all significantly reduce the impact of acoustic crosstalk on a DRA element frequency response. However, each of these design strategies has associated drawbacks which negatively affect other performance parameters (e.g. active area, pressure output, bandwidth, etc.) of the array, and thus they must be carefully considered in a more detailed optimization study. Finally, a simple FD/BEM analysis also confirmed that CMUT arrays could potentially be implemented as passive fluid sensors by utilizing a noise current detection mechanism.

In general, the FD/BEM model has demonstrated a capability to accurately predict a number of important performance parameters of CMUT arrays. Its accuracy is comparable with FEM and its efficient modeling approach allows for simulation of much larger arrays than would be possible with FEM. Array optimization is a very complex task which requires balancing many conflicting goals while exploring an extremely large design space. It is expected that the model developed here will be an extremely useful tool in future work with this challenging design problem.

7.2. Recommendations

The FD/BEM model has been developed and characterized through comparison with both FEM and experimental results. It has proven to yield accurate predictions of CMUT array performance including crosstalk effects. However, it still has inherent limitations. Namely, it currently assumes that the membrane material is homogenous and uniform. As mentioned in Chapter 3, the model could be improved by including an

effective Poisson's ratio and flexural rigidity for layers that are uniform across the entire membrane. Non-uniform thicknesses such as mass-loading could also be accounted for with the model, but may require using step-wise laminated plate theory. Thick damping layers such as RTV could also incorporated by utilizing a different Green's function for the BEM which accounts for layered media. If these improvements were made to the model, a much larger design space could be explored during optimization.

It would also be useful to pursue an even more generic model than what was proposed here. For example, the frequency response analysis neglects non-linear effects. A transient simulation which employs an iterative solution to each time step could take advantage of the model's efficiency to allow for full non-linear analysis of CMUT arrays. This would require a reworking of the framework of equations to include time dependent effects, but would be well worth the effort.

Now that the model's accuracy has been characterized and validated, the next logical step would be to implement it for optimization of CMUT arrays. This optimization could employ some of the techniques used in the design study of Chapter 6. However, it should more thoroughly explore the design space and consider all parameters relevant to transducer performance (e.g. pressure output, sensitivity, operating voltage, bandwidth, radiation pattern, etc.) in addition to the goal of minimizing sharp crosstalk features from the frequency response. Sparse array design is an ideal application for the FD/BEM model and should be investigated in detail to determine the optimal number of elements to include in an array and their corresponding configuration.

APPENDIX A

DERIVATION OF HIGHER ORDER FD APPROXIMATIONS

Every finite difference approximation can be derived using the same method. It essentially involves generating the Taylor series expansion for values of a function at (x_{i+1},y_j), (x_{i+1},y_{j+1}), (x_{i+2},y_{j-1}), and other nearby nodes surrounding that point. Note that the i,j naming scheme denotes the indices of nearby nodes relative to a central node (x_i,y_j), as shown in Figure 86. An example of this technique was provided in Chapter 3. Note that higher order derivatives require information from more surrounding nodes than lower order derivatives. They also require more higher order terms to be included in the Taylor series expansion. Furthermore, 2 dimensional derivatives $(\frac{d^2f}{dxdy})$ will require information from surrounding nodes in 2 dimensions, whereas 1 dimensional derivatives only require Taylor series expansions for surrounding nodes that are located in the associated dimension.

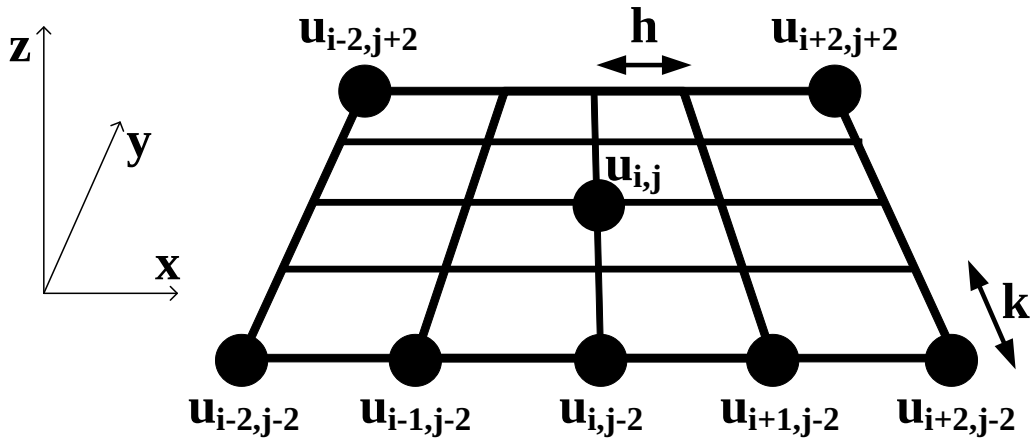


Figure 86. Index notation for finite difference approximations around a specific node, where u is the vertical displacement of a node, and i and j denote the relative location of neighboring nodes

For reference, the generic expression for a Taylor series expansion of 2 variables is:

$$
\begin{aligned}
&F\left(x_{i+ah}, y_{j+bk}\right) \\
&\quad = F + ahF_x + bkF_y + \frac{a^2}{2}h^2F_{xx} + \frac{b^2}{2}k^2F_{yy} \\
&\quad + abhkF_{xy} + \frac{a^3}{6}h^3F_{xxx} + \frac{b^3}{6}k^3F_{yyy} + \frac{a^2b}{2}h^2kF_{xxy} \\
&\quad + \frac{ab^2}{2}hk^2F_{xyy} + \frac{a^4}{24}h^4F_{xxxx} + \frac{b^4}{24}k^4F_{yyyy} + \frac{a^3b}{6}h^3kF_{xxxy} \\
&\quad + \frac{a^2b^2}{4}h^2k^2F_{xxyy} + \frac{ab^3}{6}hk^3F_{xyyy} \ldots + R(h^5)
\end{aligned}
\tag{100}
$$

$$F = F\left(x_i, y_j\right)$$

$$where \quad F_x = F_x\left(x_i, y_j\right) = \left.\frac{dF}{dx}\right|_{x=x_i, y=y_j}$$

$$\vdots$$

Note that this equation can also be used for 1-dimensional expansions by setting the appropriate incremental value to zero. By solving systems of these equations, all of the finite difference approximations necessary for the 4th order thin plate equations can be derived. The most complex derivative derivations are the 4th order derivatives of 2 dimensions ($\frac{d^4f}{dx^2dy^2}$), which require solving a system of 15 separate equations. For convenience, all finite difference approximations used in this paper are provided below.

$$F_x = \frac{1}{2h}\left[F\left(x_{i+1}, y_j\right) - F\left(x_{i-1}, y_j\right)\right] + R(h^2) \tag{101}$$

$$F_y = \frac{1}{2k}\left[F\left(x_i, y_{j+1}\right) - F\left(x_i, y_{j-1}\right)\right] + R(h^2) \tag{102}$$

$$F_{xx} = \frac{1}{h^2}\left[F\left(x_{i+1}, y_j\right) - 2F\left(x_i, y_j\right) + F\left(x_{i-1}, y_j\right)\right] + R(h^2) \tag{103}$$

$$F_{yy} = \frac{1}{k^2}\left[F\left(x_i, y_{j+1}\right) - 2F\left(x_i, y_j\right) + F\left(x_i, y_{j-1}\right)\right] + R(h^2) \tag{104}$$

$$F_{xy} = \frac{1}{4hk}\left[F\left(x_{i+1}, y_{j+1}\right) - F\left(x_{i+1}, y_{j-1}\right) - F\left(x_{i-1}, y_{j+1}\right) + F\left(x_{i-1}, y_{j-1}\right)\right] + R(h^2) \tag{105}$$

$$F_{xxx} = \frac{1}{2h^3}\left[F\left(x_{i+2}, y_j\right) - 2F\left(x_{i+1}, y_j\right) + 2F\left(x_{i-1}, y_j\right) - F\left(x_{i-2}, y_j\right)\right] + R(h^2) \tag{106}$$

$$F_{yyy} = \frac{1}{2k^3}\left[F\left(x_i, y_{j+2}\right) - 2F\left(x_i, y_{j+1}\right) + 2F\left(x_i, y_{j-1}\right) - F\left(x_i, y_{j-2}\right)\right] + R(h^2) \tag{107}$$

$$F_{xxy} = \frac{1}{6h^2k}\left[F\left(x_{i-1}, y_{j-1}\right) - F\left(x_{i+1}, y_{j+1}\right) - F\left(x_{i-1}, y_{j+1}\right) + F\left(x_{i+1}, y_{j-1}\right) + F\left(x_{i-2}, y_{j+1}\right) - F\left(x_{i+2}, y_{j-1}\right) - F\left(x_{i-2}, y_{j-1}\right) + F\left(x_{i+2}, y_{j+1}\right)\right] + R(h^2) \tag{108}$$

$$F_{xyy} = \frac{1}{6hk^2}\left[F\left(x_{i-1}, y_{j-1}\right) - F\left(x_{i+1}, y_{j+1}\right) - F\left(x_{i+1}, y_{j-1}\right) + F\left(x_{i-1}, y_{j+1}\right) + F\left(x_{i+1}, y_{j-2}\right) - F\left(x_{i-1}, y_{j+2}\right) - F\left(x_{i-1}, y_{j-2}\right) + F\left(x_{i+1}, y_{j+2}\right)\right] + R(h^2) \tag{109}$$

$$F_{xxxx} = \frac{1}{h^4}\left[F\left(x_{i+2}, y_j\right) - 4F\left(x_{i+1}, y_j\right) + 6F\left(x_i, y_j\right) - 4F\left(x_{i-1}, y_j\right) + F\left(x_{i-2}, y_j\right)\right] + R(h^2) \tag{110}$$

$$F_{yyyy} = \frac{1}{k^4}\left[F(x_i, y_{j+2}) - 4F(x_i, y_{j+1}) + 6F(x_i, y_j) - 4F(x_i, y_{j-1}) + F(x_i, y_{j-2})\right] + R(h^2) \tag{111}$$

$$\begin{aligned} F_{xxyy} = \frac{1}{9h^2k^2}\big[&F(x_{i+1}, y_{j+1}) + F(x_{i\mp1}, y_{j-1}) + F(x_{i-1}, y_{j+1}) \\ &+ F(x_{i+1}, y_{j-1}) + F(x_{i+2}, y_{j+2}) + F(x_{i-2}, y_{j-2}) \\ &+ F(x_{i+2}, y_{j-2}) + F(x_{i-2}, y_{j+2}) - F(x_{i-2}, y_{j+1}) \\ &- F(x_{i+2}, y_{j-1}) - F(x_{i-2}, y_{j-1}) - F(x_{i+2}, y_{j+1}) \\ &- F(x_{i-1}, y_{j+2}) - F(x_{i+1}, y_{j-2}) - F(x_{i-1}, y_{j-2}) \\ &- F(x_{i+1}, y_{j+2})\big] + R(h^2) \end{aligned} \tag{112}$$

APPENDIX B

DERIVATION OF FULL FD STIFFNESS EQUATION

In order to apply the finite difference method to the 4th order thin plate equations, the equations listed in Chapter 3 must be combined and expanded. They will be rewritten below for convenience.

$$K[u(x,y,t)] = \left[\frac{\partial^2 M_x}{\partial x^2} + \frac{\partial^2 M_y}{\partial y^2} + 2\frac{\partial^2 M_{xy}}{\partial x \partial y}\right][u(x,y,t)] \tag{113}$$

$$M_x = D\left(\frac{d^2u}{dx^2} + \nu\frac{d^2u}{dy^2}\right)$$

$$M_y = D\left(\frac{d^2u}{dy^2} + \nu\frac{d^2u}{dx^2}\right) \tag{114}$$

$$M_{xy} = D(1-\nu)\frac{d^2u}{dxdy}$$

Thus, expanding the moments from (114) yields:

$$\frac{\partial^2 M_x}{\partial x^2} = D_{xx}u_{xx} + 2D_x u_{xxx} + Du_{xxxx} + D_{xx}\nu u_{yy} + 2D_x v_x u_{yy} + 2D_x \nu u_{xyy} + Dv_{xx}u_{yy} + 2Dv_x u_{xyy} + D\nu u_{xxyy} \tag{115}$$

$$\frac{\partial^2 M_y}{\partial y^2} = D_{yy}u_{yy} + 2D_y u_{yyy} + Du_{yyyy} + D_{yy}\nu u_{xx} + 2D_y v_y u_{xx} + 2D_y \nu u_{xxy} + Dv_{yy}u_{xx} + 2Dv_y u_{xxy} + D\nu u_{xxyy} \tag{116}$$

$$\frac{\partial^2 M_{xy}}{\partial x \partial y} = D_{xy}u_{xy} + D_y u_{xxy} + D_x u_{xyy} + Du_{xxyy} - D_{xy}\nu u_{xy} - D_y v_x u_{xy} - D_y \nu u_{xxy} - D_x v_y u_{xy} - Dv_{xy}u_{xy} - Dv_y u_{xxy} - D_x \nu u_{xyy} - Dv_x u_{xyy} - D\nu u_{xxyy} \tag{117}$$

Note that the subscripts of the moments do not denote derivatives, but rather reference the direction in which the moment is applied to the elemental volume. Substituting these expanded equations into the original stiffness equation yields:

$$\begin{aligned} K = {} & u_{xxxx}[D] + u_{yyyy}[D] + u_{xxyy}[D] + u_{xxx}[2D_x] + u_{yyy}\left[2D_y\right] \\ & + u_{xxy}\left[2D_y\right] + u_{xyy}[2D_x] \\ & + u_{xx}\left[D_{xx} + \nu D_{yy} + D\nu_{yy} + 2D_y\nu_y\right] \\ & + u_{yy}\left[D_y + \nu D_{xx} + D\nu_{xx} + 2D_x\nu_x\right] \\ & + u_{xy}\left[2D_{xy} - 2\nu D_{xy} - 2D_y\nu_y - 2D\nu_{xy}\right] \end{aligned} \tag{118}$$

This expansion assumes that D and ν are functions of x and y. Their respective derivatives can also be approximated with finite difference equations. Note that this assumption is true only if the membrane does not have a uniform thickness (e.g. mass loading, thick electrode with partial coverage, etc.). For a uniform membrane, all of the derivatives of these two variables can be set to zero, and the final equation simplifies to:

$$K = u_{xxxx}[D] + u_{yyyy}[D] + u_{xxyy}[D] \tag{119}$$

APPENDIX C

COMMENTED MATLAB CODE

MAIN CODE

```
clear all
close
clc

global Electrode
global eps_0
global h_eq
global iterMax
global ReducedNodes
global Kmm
global TotalNodes
global StepX
global StepY
global X_m
global Y_m
global NodesX
global NodesY
global FreqList
global zAC
global vel_AC_ARRAY
global MemSpaceX
global MemSpaceY
global NumMemX
global NumMemY
global NumMemXi
global NumMemYi
global PitchX
global PitchY
global Fluid_c
global Fluid_rho
global RayleighPlane_X_cent
global RayleighPlane_Y_cent
global RayleighPlane_LengthX
global RayleighPlane_LengthY
global RayleighPlane_NodesX
global RayleighPlane_NodesY
global RayleighPlane_Zloc
global RayleighAngular_X_cent
global RayleighAngular_Y_cent
global RayleighAngular_dS
global NumNodesRad
global RayleighAngular_R
global vAC
global symy
global symx
global Fluid_alpha_props
global LiveMemsInd
global Input_pulse_FFT_interp
global TotalMems
global TotalMemsSYM

%% Constants

eps_0 = 8.85418782e-12;    %Permmitivity of free space
```

```
kT = 4.11e-21;              %Boltzmann constant * Temperature
tic;

%%  (USER INPUTS)

% Material Properties
Nitride_E    = 110e9;   %[Pa]
Nitride_v    = .22;
Nitride_rho  = 2040;    %[kg/m^3]
Nitride_eps_r = 6.3;
Aluminum_E    = 70.0e9;     %[Pa]
Aluminum_v    = 0.35;
Aluminum_rho  = 2700;       %[kg/m^3]
Parylene_E   = 3.2e9; %[Pa]
Parylene_v   = .4;
Parylene_rho = 1289;  %[kg/m^3]
Water_rho    = 1000;  %[kg/m^3]
Water_c      = 1500;  %[m/s]
Air_rho      = 1.2;   %[kg/m^3]
Air_c        = 343;   %[m/s]

damp = 6e3;     %[Pa-s/m] Damping Term

% Membrane Geometry (Square DRA devices, NEW CMUT BATCH)
Membrane_Xwidth = 35e-6;     %[um]
Membrane_Ywidth = 35e-6;     %[um]
Membrane_thick = 2e-6;     %[um]
Electrode_Xwidth = 35e-6;    %[um]
Electrode_Ywidth = 35e-6;    %[um]
BE_isol_height = 0.24e-6;    %[um]
Sac_isol = .136e-6;           %[um]
Sac_isol_height = 0.32e-6;   %[um]
TE_height = 0e-6;            %m  %Keep at 0um, may cause inaccurate
solution
Parylene_thick = 0e-6;       %m  %Keep at 0um, may cause inaccurate
solution

% Array parameters
MemSpaceX = 10e-6;
MemSpaceY = 10e-6;
NumMemX = 2;    %Number of membranes in x direction
NumMemY = 1;    %Number of membranes in y direction
LiveMemsInd = [1];  %Index of membranes that are active

                        %Membranes are numbered from left to right,
                        %start with bottom row and move up

                        %If Symmetry exists, only account for
bottom/left
                        %corner of array

% Denote if symmetry exists across x-axis (symx) or y-axis (symy)
symx = 0;  %1 = yes, 0 = no (make sure loading, geom, B.C.s, etc are
ALL SYMMETRIC!!)
```

```
symy = 0;  %1 = yes, 0 = no (make sure loading, geom, B.C.s, etc are
ALL SYMMETRIC!!)

% Simulation Options
ShowArrayPreview = 1;
CalculateDynamic = 1;
CalculateRayleighInt = 0;
plotdB = 1;
Normalize_FreqResponse = 1;
Plot_Pulse_Echo_FFT = 0;

% Actuation Parameters
P_a = 0;   %[Pa] Atmospheric Pressure
V0 = 80;   %[V] DC Bias

% Frequency Range of Interest
fstart = 1e6;
fstop  = 30e6;
fsteps  = 1000;

% Input Signal Parameters
InputType = 'toneburst'; %Choose from: 'cw', 'toneburst',
'pulse','fromfigfile','pressure'

switch lower(InputType)
    case 'cw'
        vAC = -6;   %[Vp] AC voltage
    case 'toneburst'
        Freq_cent = 15;      %MHz
        Amp = 5;            %V (peak)
        NumCycles = 1;      %Cycles
        BurstStart = .5;     %us
        Alpha = 1;        %Constant for Gaussian Window.  For no
windowing, set to 0.  Default is 2.5.
    case 'pulse'
        Amp = 1;            %V
        Pulsewidth = 15;     %ns
        PulseStart = 5;     %ns
        Alpha = 2.5;        %Constant for Gaussian Window.  For no
windowing, set to 0.  Default is 2.5.
    case 'fromfigfile'
        FigFile = 'Z:\home\Mike
Hochman\Research\Experimental_Results\HydrophoneScans\4CenterElemActive
_1mmx1mmScan\Input_12MHz_5Cyc_20mV.fig';
end

Cp = 3e-12;     %[F] Parasitic Capacitance (only affects impedance
plot) (don't set to 0!)

% Environment parameters
Vacuum = 0;
Use_water = 1;  %1 = water, 0 = air

% Damping properties
%Fluid_alpha_props = [10*100];       %Red RTV
```

```
Fluid_alpha_props = [0];              %Water
% Fluid_alpha_props = [6.43e-12 1.85];  %Oil
                                     %[] Attenuation properties [A n] --
> alpha = A*f^n

                                     %Use zero vector [0 0] or [0] to
assume no attenuation

                                     %If n-value is omitted, value is
                                     %assumed to be alpha in [dB/MHz/m]

% Rayleigh Integral Geometry (don't use if not calculating Rayleigh
Integral)
RayleighPlane_X_cent = 45e-6*12/2;       %m
RayleighPlane_Y_cent = 45e-6*12/2;       %m
RayleighPlane_LengthX = 5e-3;          %m
RayleighPlane_LengthY = 5e-3;          %m
RayleighPlane_NodesX = 11;                    %m
RayleighPlane_NodesY = 11;                    %m
RayleighPlane_Zloc = 4.5e-3;                %m

% Rayleigh Integral Geometry for Angular Plot(don't use if not
calculating Rayleigh Integral)
RayleighAngular_X_cent = 45e-6*12/2;       %m
RayleighAngular_Y_cent = 45e-6*12/2;       %m
RayleighAngular_dS     = (80e-6)^2;        %m^2
NumNodesRad = 180;                         %steps
RayleighAngular_R = 5.5e-3;                 %m

% Choose only ODD-NUMBERED node values (ensures a valid center point on
membrane)
NodesX = 19;             %Number of nodes in X direction
NodesY = 19;             %Number of nodes in X direction
iterMax = 30;    %Sets maximum iteration to avoid infinite loop

%% Calculate basic values
X_m = Membrane_Xwidth;    %Membrane X width
Y_m = Membrane_Ywidth;    %Membrane Y width
X_e = Electrode_Xwidth;     %Electrode X width
Y_e = Electrode_Ywidth;     %Electrode Y width

if Use_water == 1
    Fluid_rho = Water_rho;
    Fluid_c   = Water_c;
else
    Fluid_rho = Air_rho;
    Fluid_c     = Air_c;
end

PitchX = Membrane_Xwidth + MemSpaceX;
PitchY = Membrane_Ywidth + MemSpaceY;
Elem_x = Membrane_Xwidth*NumMemX + MemSpaceX*(NumMemX-1);
Elem_y = Membrane_Ywidth*NumMemY + MemSpaceY*(NumMemY-1);
Elem_Area = Elem_x*Elem_y;
Mem_Area = Membrane_Xwidth*Membrane_Ywidth;
```

```
Elec_Area = Electrode_Xwidth*Electrode_Ywidth;
LiveMems = length(LiveMemsInd);  %Number of membranes that are active
Elem_Area_LIVE = Mem_Area*LiveMems;
TotalMems = NumMemX*NumMemY;

h_eq0 = Sac_isol + (BE_isol_height + Sac_isol_height)/Nitride_eps_r;

StepX = X_m/(NodesX-1);    %X Step size
StepY = Y_m/(NodesY-1);    %Y Step size

TotalNodes = NodesX*NodesY;

%% These if statements correct parameters (Area, number of membranes,
etc)
% that are decreased by using symmetry
NumMemXi = NumMemX;
NumMemYi = NumMemY;
if symx
    NumMemYi = ceil(NumMemY/2);
    Elem_Area = Elem_Area*.5;
end
if symy
    NumMemXi = ceil(NumMemX/2);
    Elem_Area = Elem_Area*.5;
end

if symx || symy
    TotalMemsSYM = NumMemXi*NumMemYi;
    MEM_LIVE = [1:TotalMemsSYM]';

    MEMSYM = ones(TotalMemsSYM,1);
    if symy && mod(NumMemX,2)
        MEMSYM(NumMemXi:NumMemXi:end,1) = .5;
    end
    if symx && mod(NumMemY,2)
        MEMSYM(NumMemYi:end,1) = .5;
    end
    if symx && symy
        if mod(NumMemX,2) && mod(NumMemY,2)
            MEMSYM(end,1) = .25;
        end
    end

else
    MEM_LIVE = [1:TotalMems]';
    MEMSYM = ones(TotalMems,1);
end

%% Generate a column vector of 1's (active) and 0's (dead) to designate
which membranes are live
MEM_LIVE = [1:TotalMems]';
MEM_LIVE = ismember(MEM_LIVE, LiveMemsInd);

%% Check if number of nodes is odd
if (mod(NodesX,2) == 0) || (mod(NodesY,2) == 0)
```

```
    disp('Number of nodes in x and y direction must both be ODD to
ensure valid center point')
    disp(' ')
    disp('Choose alternate node numbers!!')
    return
end

%% Check if Z_mut will be too large for memory to handle
Z_mut_Size = (NodesX*NodesY*NumMemXi*NumMemYi)^2;
Z_mut_SizeMax = 9e7;
if Z_mut_Size > Z_mut_SizeMax
    disp('Stiffness matrix may be too large (for a computer with 4MB
RAM)!!!')
    disp(' ')
    disp('Reduce number of nodes/membranes!!')
    return
end

%% Ensure model geometry looks ok
if ShowArrayPreview
    Plot_ARRAY_Preview_FD_SYM
    button = questdlg('This is the designated model geometry.  Continue
with evaluation?');

    switch button
        case 'Yes'
            close
        otherwise
            return
    end
end

%% Calculate FFT of input signal and get user approval to proceed
if CalculateDynamic
    FreqList = (fstart:(fstop-fstart)/(fsteps-1):fstop)/1e6;
    switch lower(InputType)
        case 'cw'
            Input_pulse_FFT_interp =
Calculate_FreqResponseAdjustedByInputFFT(InputType,vAC);
        case 'toneburst'
            Input_pulse_FFT_interp =
Calculate_FreqResponseAdjustedByInputFFT(InputType,Freq_cent,Amp,NumCyc
les,Alpha,BurstStart);
        case 'pulse'
            Input_pulse_FFT_interp =
Calculate_FreqResponseAdjustedByInputFFT(InputType,Amp,Pulsewidth,Pulse
Start,Alpha);
        case 'fromfigfile'  %If loaded from fig file, generate
necessary parameters before code runs
            Input_pulse_FFT_interp =
Calculate_FreqResponseAdjustedByInputFFT(InputType,FigFile);
        otherwise
            Input_pulse_FFT_interp =
Calculate_FreqResponseAdjustedByInputFFT(InputType);

    end
```

```
    if isempty(Input_pulse_FFT_interp)  %Do not continue with code if
user did not accept FFT
        return
    end
end

%% Show user preview of Rayleigh Integral geometry and get approval to
proceed
if CalculateRayleighInt
    Plot_ARRAY_Preview_FD_SYM

    grid_space_Xz = RayleighPlane_LengthX/(RayleighPlane_NodesX-1);
    grid_space_Yz = RayleighPlane_LengthY/(RayleighPlane_NodesY-1);

    %Define mesh of planar location at given z_Rayleigh (if symmetry
exists, override center values)
    X_z = RayleighPlane_X_cent-
RayleighPlane_LengthX/2:grid_space_Xz:RayleighPlane_X_cent+RayleighPlan
e_LengthX/2;
    Y_z = RayleighPlane_Y_cent-
RayleighPlane_LengthY/2:grid_space_Yz:RayleighPlane_Y_cent+RayleighPlan
e_LengthY/2;

    [X_z Y_z] = meshgrid(X_z,Y_z);

    Z =
RayleighPlane_Zloc*ones(RayleighPlane_NodesY,RayleighPlane_NodesX);
    s = surface(X_z/1e-6,Y_z/1e-6,Z);
    set(s,'FaceColor','b')
    set(s,'FaceAlpha',.3)
    xlabel('X position (um)')
    ylabel('Y position (um)')
    title(['Preview of Rayleigh Integral geometry.  Blue plane is
location at which pressures will be calculated.'])
    xlim('auto')
    ylim('auto')


    button = questdlg('This is the geometry that will be used for the
Rayleigh Integral.  Continue with evaluation?');

    switch button
        case 'Yes'
            close
        otherwise
            return
    end
end

%% Initialize all matrices
Electrode = zeros(TotalNodes,1);

D = zeros(TotalNodes,6);          %Column 2 - 1st Derivative, Column3 -
2nd Derivative
```

```
v = zeros(TotalNodes,6);          %Column 2 - 1st Derivative, Column3 -
2nd Derivative

%Calculate effective D and v (if Parylene_thick = 0, this will generate
the
%original values of D and v.  See K. Pister reference (Ref 26 in model
binder)
y=zeros(2,1);
y(1) = Membrane_thick;
y(2) = y(1) + Parylene_thick;
Qk = [Nitride_E/(1-Nitride_v^2); Parylene_E/(1-Parylene_v^2)];
Qk_v = [Nitride_E*Nitride_v/(1-Nitride_v^2); Parylene_E*Parylene_v/(1-
Parylene_v^2)];
Bk = Qk(1)*y(1)+Qk(2)*(y(2)-y(1));
Bk_v = Qk_v(1)*y(1)+Qk_v(2)*(y(2)-y(1));
Ck = Qk(1)*y(1)^2/2+Qk(2)*(y(2)^2-y(1)^2)/2;
Ck_v = Qk_v(1)*y(1)^2/2+Qk_v(2)*(y(2)^2-y(1)^2)/2;
Dk = Qk(1)*y(1)^3/3+Qk(2)*(y(2)^3-y(1)^3)/3;
Dk_v = Qk_v(1)*y(1)^3/3+Qk_v(2)*(y(2)^3-y(1)^3)/3;
K1 = (Bk*Ck-Bk_v*Ck_v)/(Bk^2-Bk_v^2);
K2 = (Bk*Ck_v-Bk_v*Ck)/(Bk^2-Bk_v^2);
Dk_star = Dk-Ck*K1-Ck_v*K2;
Dk_v_star = Dk_v-Ck_v*K1-Ck*K2;
Dmult = Dk_star;
vmult = Dk_v_star/Dk_star;
Dmult2 = Dmult; %Can use this variable to account for 3-layer membrane
with electrode thickness
vmult2 = vmult;

countj = 0;
D(:,1) = Dmult;   %Default all D and v as Single Layer coefficients
v(:,1) = vmult;

%If electrode layer had a thickness, this for-loop could be used to
change
%values of D and v on electrode locations
for y = -(NodesY-1)/2*StepY:StepY:(NodesY-1)/2*StepY

    if y <= StepY*floor(Y_e/StepY/2+1) && y >= -
StepY*floor(Y_e/StepY/2+1)

        Electrode(countj*NodesX + ceil((X_m/2-
X_e/2)/StepX+1):countj*NodesX + floor((X_m/2+X_e/2)/StepX+1),1) = 1;

        %Multiple Layer coefficients
        D(countj*NodesX + ceil((X_m/2-X_e/2)/StepX+1):countj*NodesX +
floor((X_m/2+X_e/2)/StepX+1),1) = Dmult2;
        v(countj*NodesX + ceil((X_m/2-X_e/2)/StepX+1):countj*NodesX +
floor((X_m/2+X_e/2)/StepX+1),1) = vmult2;

    end
    countj = countj+1;
end
```

```
Electrode(1+(NodesY-1)*NodesX:TotalNodes) = [];     %Delete node rows
for fixed x edge (top)
Electrode(NodesX:NodesX:NodesX*(NodesY-1)) = [];    %Delete node rows
for fixed x edge (right)
Electrode(1:(NodesX-1)) = [];                       %Delete node rows
for fixed x edge (bottom)
Electrode(1:(NodesX-1):(NodesX-1)*(NodesY-2)) = []; %Delete node rows
for fixed x edge (left)

%% Solve for derivatives of D and v using finite differences

%Column 1   Column 2    Column 3    Column 4    Column 5    Column 6
%   D         Dx           Dy         Dxx          Dyy         Dxy
%   v         vx           vy         vxx          vyy         vxy

count = 1;
for y = -(NodesY-1)/2*StepY:StepY:(NodesY-1)/2*StepY
    for x = -(NodesX-1)/2*StepX:StepX:(NodesX-1)/2*StepX

        if (x == -StepX*(NodesX-1)/2) || (y == -StepY*(NodesY-1)/2)
%Fixed boundary (outer edge of membrane)
            D(count,2:6) = 0;
            v(count,2:6) = 0;

        elseif (x == StepX*(NodesX-1)/2) || (y == StepY*(NodesY-1)/2)
%Fixed boundary (outer edge of membrane)

            D(count,2:6) = 0;
            v(count,2:6) = 0;
        else

            D(count,2) = (D(count+1,1) - D(count-1,1))./(2*StepX);
            D(count,3) = (D(count+NodesX,1) - D(count-
NodesX,1))./(2*StepY);
            D(count,4) = (D(count+1,1) - 2*D(count,1) + D(count-
1,1))./(StepX^2);
            D(count,5) = (D(count+NodesX,1) - 2*D(count,1) + D(count-
NodesX,1))./(StepY^2);
            D(count,6) = (D(count+NodesX+1,1) - D(count-NodesX+1,1) -
D(count+NodesX-1,1) + D(count-NodesX-1,1))./(4*StepX*StepY);

            v(count,2) = (v(count+1,1) - v(count-1,1))./(2*StepX);
            v(count,3) = (v(count+NodesX,1) - v(count-
NodesX,1))./(2*StepY);
            v(count,4) = (v(count+1,1) - 2*v(count,1) + v(count-
1,1))./(StepX^2);
            v(count,5) = (v(count+NodesX,1) - 2*v(count,1) + v(count-
NodesX,1))./(StepY^2);
            v(count,6) = (v(count+NodesX+1,1) - v(count-NodesX+1,1) -
v(count+NodesX-1,1) + v(count-NodesX-1,1))./(4*StepX*StepY);
        end
        count = count + 1;
    end
end
```

```
%% Solve for stiffness matrix using finite differences
%Kmm is a square matrix with "ReducedNodes" nodes
Kmm = CalculateKmm_FD(StepX, StepY, NodesX, NodesY, TotalNodes, D, v);

%%                      Determine Static Solution
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%

% 1. Begin with initial guess for displacement matrix (0 displacement)
% 2. Then find corresponding electrostatic force
% 3. Then update displacement matrix by calculating mechanical reaction
to
% electrostatic force
% 4. Continue iterating by updating electrostatic force then
displacement
% 5. Finish iterating when change in displacement matrix goes below
certain
% threshold

ReducedNodes = (NodesX-2)*(NodesY-2);
h_eq(1:ReducedNodes,1) = h_eq0;

disp(['Calculating membrane displacement at ', num2str(V0), '
Volts...'])

[z P_e iter] = Calculate_Static_Displacement(V0,P_a);

%% Check for convergence and continue on...
if iter == iterMax
    disp('Solution did not converge!  Membrane is collapsed!')
    return      %Exit code if membrane collapsed
else
    disp(['Solution converged in ', num2str(iter), ' steps'])
    disp(' ')
end



% %%%%%%%%    Calculate free capacitance for k^2
%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

IncStep = .00001;
deltaV = V0 + IncStep*V0;

disp(['Calculating membrane displacement at ', num2str(V0), 'V + dV to
find k^2...'])

[z2 P_e2 iter2] = Calculate_Static_Displacement(deltaV,P_a);

if iter2 == iterMax
    disp('k^2 could not be calculated!  Bias voltage is too close to
collapse!')
```

```
    return
else
    disp(['Solution converged in ', num2str(iter2), ' steps'])
    disp(' ')

    %Calculate static capacitance and charge density
    Qdensity = eps_0.*Electrode*V0./(h_eq(:,1) + z(:,iter));
    C0 = sum(Qdensity.*StepX*StepY)/V0;

    Qdensity2 = eps_0.*Electrode*deltaV./(h_eq(:,1) + z2(:,iter2));
    C02 = sum(Qdensity2.*StepX*StepY)/deltaV;
    CT = (C02*deltaV - C0*V0)/(deltaV - V0);

    k2 = 1 - C0/CT;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%%                  Calculate dynamic solution parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%

if CalculateDynamic

    %Mass matrix (Diagonal), for-loop is needed only for non-uniform
    %membrane thicknesses
    M = zeros(TotalNodes,TotalNodes);
    count = 1;
    for y = -(NodesY-1)/2*StepY:StepY:(NodesY-1)/2*StepY
        for x = -(NodesX-1)/2*StepX:StepX:(NodesX-1)/2*StepX

            if y >= -StepY*floor(Y_e/StepY) && y <=
StepY*floor(Y_e/StepY)
                if x >= -StepX*floor(X_e/StepX) && x <=
StepX*floor(X_e/StepX)
                    M(count,count) = Nitride_rho*Membrane_thick +
Parylene_rho*Parylene_thick + Aluminum_rho*TE_height;
                end
            else
                M(count,count) = Nitride_rho*Membrane_thick +
Parylene_rho*Parylene_thick;
            end
            count = count + 1;
        end
    end

    M(:,1+(NodesY-1)*NodesX:TotalNodes) = [];       %Delete node
columns for fixed y edge (top)
    M(1+(NodesY-1)*NodesX:TotalNodes,:) = [];       %Delete node rows
for fixed y edge (top)
```

```
    M(NodesX:NodesX:NodesX*(NodesY-1),:) = [];      %Delete node rows
for fixed x edge (right)
    M(:,NodesX:NodesX:NodesX*(NodesY-1)) = [];      %Delete node
columns for fixed x edge (right)
    M(:,1:(NodesX-1)) = [];                         %Delete node
columns for fixed y edge (bottom)
    M(1:(NodesX-1),:) = [];                         %Delete node rows
for fixed y edge (bottom)
    M(1:(NodesX-1):(NodesX-1)*(NodesY-2),:) = [];   %Delete node rows
for fixed x edge (left)
    M(:,1:(NodesX-1):(NodesX-1)*(NodesY-2)) = [];   %Delete node
columns for fixed x edge (left)


    %Mechanical to electrical coupling matrix (Transformer Ratio)
    Kme = 2*P_e(:,iter)/V0;

    %Mechanical stiffness matrix with spring softening and damping
    z3 = z(:,iter) - .0001*z(:,iter);
    P_e3 = -.5.*Electrode.*eps_0.*V0^2./(h_eq(:) + z3).^2;
    SpringSoft = (P_e3 - P_e(:,iter))./(z3 - z(:,iter));
    Kmm_p = Kmm - diag(SpringSoft);

    %%                      Run Frequency Sweep

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
    if symx || symy
        vel_AC_ARRAY = zeros(TotalMemsSYM,fsteps);
        vel_AC = zeros(TotalMemsSYM*ReducedNodes,fsteps);
        zAC = zeros(TotalMemsSYM*ReducedNodes,fsteps);

        %Arrange matrices for array configuration
        Kme_FD = [];
        MEMSYM_FD = [];
        for mems = 1:TotalMemsSYM
            if ismember(mems, LiveMemsInd)
                Kme_FD = [Kme_FD; Kme];
            else
                Kme_FD = [Kme_FD; Kme*0];
            end
            MEMSYM_FD = [MEMSYM_FD; ones(ReducedNodes,1)*MEMSYM(mems)];

        end

    else
        vel_AC_ARRAY = zeros(TotalMems,fsteps);
        vel_AC = zeros(TotalMems*ReducedNodes,fsteps);
        zAC = zeros(TotalMems*ReducedNodes,fsteps);

        %Arrange matrices for array configuration
        Kme_FD = [];
        for mems = 1:TotalMems
            if ismember(mems, LiveMemsInd)
                Kme_FD = [Kme_FD; Kme];
```

```
            else
                Kme_FD = [Kme_FD; Kme*0];
            end

        end
            MEMSYM_FD = ones(size(Kme_FD));

    end

    zAC_avg = zeros(2,fsteps);
    vel_AC_ARRAY_avg = zeros(2,fsteps);
    vel_AC_avg = zeros(2,fsteps);
    vel_AC_avg_LIVE = zeros(1,fsteps);
    Z_mut_Lump = zeros(TotalMems, TotalMems*fsteps);
    Z_mut_Lump = mat2cell(Z_mut_Lump,
[TotalMems],[ones(1,fsteps)*TotalMems]);
    Z_mut_Lump2 = Z_mut_Lump;

    StepSize = (fstop-fstart)/(fsteps-1);

    disp(' ')
    disp('Begin calculation of dynamic solution')
    count = 1;
    for w0 = 2*pi*fstart:2*pi*StepSize:2*pi*fstop

        disp(['Calculating dynamic solution for ',
num2str(w0/(2*pi*1e6)), ' MHz...'])
        progressbar((w0-2*pi*fstart)/(2*pi*fstop-
2*pi*fstart),0,'Dynamic Solution'); % Update status bar

        Kfluid = CalculateKr2_FD(StepX, StepY, NodesX, NodesY,
ReducedNodes, w0, Vacuum, Use_water); %Update fluid radiation stiffness
matrix
        Z_lump = Kmm_p - M*w0^2 - 1i*damp*w0*eye(size(M)) +
1i*w0*Kfluid;

        [Z_mut] = CalculateZ_Array_FD_SYM(Z_lump,w0);

        zAC(:,count) = (cell2mat(Z_mut))\Kme_FD(:,1)*-1;
%Accounts for force distribution due to membrane curvature
        %     zAC(:,count) =
(cell2mat(Z_mut))\ones(size(Kme_FD(:,1)))*-1;     %Use for applying a
uniform pressure loading on all membranes


        vel_AC(:,count) = zAC(:,count)*w0*1i;
        vel_AC_avg(1,count) =
sum(vel_AC(:,count).*StepX.*StepY.*MEMSYM_FD)/(Elem_Area);
        zAC_avg(1,count) =
abs(sum(zAC(:,count).*StepX.*StepY.*MEMSYM_FD)/(Elem_Area));

        if Vacuum
            count=count+1;
            continue
```

```
        end

        count = count + 1;

    end

    %%  Calculate Rayleigh Integral for surface velocity
    if CalculateRayleighInt
        count = 1;
        PressureAtZ_avg = zeros(2,fsteps);

        for w0 = 2*pi*fstart:2*pi*StepSize:2*pi*fstop

            disp(['Calculating Rayleigh Integral for surface velocity
at ', num2str(w0/(2*pi*1e6)), ' MHz'])
            progressbar((w0-2*pi*fstart)/(2*pi*fstop-
2*pi*fstart),0,'Rayleigh Integral'); % Update status bar

            PressureAtZ = CalculateRayleigh_FDsym(vel_AC(:,count), w0);
            PressureAtZ_avg(1,count) = abs(mean(mean(PressureAtZ)));
            count = count + 1;
        end
    end

end
progressbar(1,0,''); %Close progress bar if still open

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%%                      Display Results of Simulation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%

disp(' ')
disp(' ')
toc
disp(' ')
disp(['Bias Voltage = ', num2str(V0), ' V'])
disp(' ')

disp(['Center of membrane displaced = ', num2str(z((NodesX-
1)/2+((NodesY-1)/2-1)*(NodesX-2),iter)/1e-6), ' um'])
disp(' ')

disp(['Static Capacitance is ', num2str(C0), ' Farads'])
disp(' ')
```

```
disp(['Free Capacitance is ', num2str(CT), ' Farads'])
disp(' ')

disp(['Coupling Coefficient is ', num2str(k2)])
disp(' ')

%Define a text string cell array to denote simulation properties in
graph windows
GraphText{1} = ['Membrane: ',num2str(Membrane_Xwidth/1e-
6),'x',num2str(Membrane_Ywidth/1e-6),'x',num2str(Membrane_thick/1e-
6),'um'];
GraphText{2} = ['Electrode: ',num2str(Electrode_Xwidth/1e-
6),'x',num2str(Electrode_Ywidth/1e-6),'um'];
GraphText{3} = ['Gap/Isol: ',num2str(Sac_isol/1e-
6),'/',num2str((BE_isol_height+Sac_isol_height)/1e-6),'um'];
GraphText{4} = ['Bias: ', num2str(V0),'V'];
GraphText{5} = ['P_a:    ', num2str(P_a/1e6),' MPa'];
if Vacuum
    GraphText{5} = ['Medium: ', 'Vacuum'];
elseif Use_water
    if Fluid_alpha_props == [0];
        GraphText{5} = ['Medium: ', 'Water'];
    elseif Fluid_alpha_props == [6.43e-12 1.85];        %Red RTV
        GraphText{5} = ['Medium: ', 'Oil'];
    else
        GraphText{5} = ['Medium: ', 'Attenuating Fluid'];
    end
else
    GraphText{5} = ['Medium: ', 'Air'];
end

GraphText{6} = ['Array Dim: ', num2str(NumMemX),'x', num2str(NumMemY),'
mems'];
GraphText{7} = ['Damping Coeff.: ', num2str(damp),'N-s/m'];
GraphText{8} = ['SpaceX/Y: ', num2str(MemSpaceX/1e-6),'/',
num2str(MemSpaceY/1e-6),'um'];


% Create new z vector that includes zero displacement values at
boundaries
zPLOT = zeros(TotalNodes,1);
count1 = 1;
count2 = 1;

for y = 0:StepY:Y_m
    for x = 0:StepX:X_m
        if (y==StepY*(NodesY-1)) || (x==StepX*(NodesX-1))
            zPLOT(count1,1) = 0;
        elseif (y==0) || (x==0)
            zPLOT(count1,1) = 0;
        else
            zPLOT(count1,1) = z(count2,iter);
            count2=count2+1;
        end
```

```
        count1 = count1+1;
    end
end

%Plot static membrane displacement at DC bias
figure(1)
subplot(1,2,1)
grid on
plot(0:StepX/1e-6:X_m/1e-
6,zPLOT(1+NodesX*(NodesY+1)/2:NodesX+NodesX*(NodesY+1)/2)/1e-6)
xlabel('X position at Y = 0 (um)')
ylabel('displacement (um)')
title(['Displacement of ',num2str(Membrane_Xwidth/1e-
6),'x',num2str(Membrane_Ywidth/1e-6),'x',num2str(Membrane_thick/1e-
6),'um membrane at a bias of ', num2str(V0), ' Volts, P_a = ',
num2str(P_a/1e6),' MPa'])
set(gca,'FontSize',15)
t = get(gca, 'title');
xlab = get(gca, 'xlabel');
ylab = get(gca, 'ylabel');
set(t, 'FontSize', 15)
set(xlab, 'FontSize', 15)
set(ylab, 'FontSize', 15)


%Plot 3D surface plot of static membrane displacement
[X,Y] = meshgrid(0:StepX:X_m,0:StepY:Y_m);
Z1=zeros(NodesY,NodesX);
count1 = 1;
for y = 0:StepY:Y_m
    Z1(count1,1:NodesX) = zPLOT(1+NodesX*(count1-1):NodesX*count1);
    count1 = count1 + 1;
end
subplot(1,2,2)
surf(X/1e-6,Y/1e-6,Z1./1e-6)
xlabel('X position at Y = 0 (um)')
ylabel('Y position at X = 0 (um)')
zlabel('displacement (um)')
title(['Displacement of ',num2str(Membrane_Xwidth/1e-
6),'x',num2str(Membrane_Ywidth/1e-6),'x',num2str(Membrane_thick/1e-
6),'um membrane at a bias of ', num2str(V0), ' Volts, P_a = ',
num2str(P_a/1e6),' MPa'])
if X_m < Y_m
    xlim([0 Y_m/1e-6])
    ylim([0 Y_m/1e-6])
else
    xlim([0 X_m/1e-6])
    ylim([0 X_m/1e-6])
end
%zlim([-Sac_isol/1e-6 0])
set(gca,'FontSize',15)
t = get(gca, 'title');
xlab = get(gca, 'xlabel');
ylab = get(gca, 'ylabel');
zlab = get(gca, 'zlabel');
set(t, 'FontSize', 15)
```

```
set(xlab, 'FontSize', 15)
set(ylab, 'FontSize', 15)
set(zlab, 'FontSize', 15)


if ~CalculateDynamic
    return
end

%% Plot average velocity and displacement

%%%%%%%%Plot frequency response of array displacement
%%%%%%%%%%%%%%%%%%%%%

%Calculate frequency response of entire array displacement
zAC_avg(1,:) = abs(zAC_avg(1,:));
zAC_avg(2,:) = 20*log10(zAC_avg(1,:));

%Calculate frequency response of membrane ARRAY displacement (adjusted
for input spectrum)
zAC_avg(2,:) = zAC_avg(2,:) + Input_pulse_FFT_interp - 20*log10(1);

if plotdB
    if Normalize_FreqResponse
        zAC_avg(2,:) = zAC_avg(2,:) - max(max(zAC_avg(2,:)));
    end
else
    zAC_avg(2,:) = 10.^(zAC_avg(2,:)./20);
end

figure(2)
membrane = [1+ReducedNodes*3:ReducedNodes*4];
plot(FreqList,20*log10(mean(zAC(membrane,:))))
grid on
xlabel('Frequency (MHz)')
if plotdB
    ylabel('Average displacement (dB)')
else
    ylabel('Average displacement (m)')
end
%ylim([-80 0])
title(['Frequency response of ',num2str(Membrane_Xwidth/1e-
6),'x',num2str(Membrane_Ywidth/1e-6),'x',num2str(Membrane_thick/1e-
6),'um membrane at a bias of ', num2str(V0), ' Volts, P_a = ',
num2str(P_a/1e6),' MPa'])
set(gca,'FontSize',15)
t = get(gca, 'title');
xlab = get(gca, 'xlabel');
ylab = get(gca, 'ylabel');
set(t, 'FontSize', 15)
set(xlab, 'FontSize', 15)
set(ylab, 'FontSize', 15)

%%%%%%%%Plot frequency response of array velocity
%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% ActiveElemNodes=[1+3*ReducedNodes:4*ReducedNodes
1+7*ReducedNodes:8*ReducedNodes];
% vel_AC_avg =
sum(vel_AC(ActiveElemNodes,:).*StepX.*StepY)/(2*Mem_Area);

%Calculate frequency response of entire array velocity
vel_AC_avg(1,:) = abs(vel_AC_avg(1,:));
vel_AC_avg(2,:) = 20*log10(vel_AC_avg(1,:));

%Calculate frequency response of ARRAY velocity (adjusted for input
spectrum)
vel_AC_avg(2,:) = vel_AC_avg(2,:) + Input_pulse_FFT_interp -
20*log10(1);

if plotdB
    if Normalize_FreqResponse
        vel_AC_avg(2,:) = vel_AC_avg(2,:) - max(max(vel_AC_avg(2,:)));
    end
else
    vel_AC_avg(2,:) = 10.^(vel_AC_avg(2,:)./20);
end

figure(3);
set(gcf,'toolbar','figure')
hold on
plot(FreqList,vel_AC_avg(2,:))
grid on
xlabel('Frequency (MHz)')
if plotdB
    ylabel('Average velocity (dB)')
else
    ylabel('Average velocity (m/s)')
end
%ylim([-25 0])
%xlim([2 20])
title('Frequency response of all membranes in array')
set(gca,'FontSize',15)
t = get(gca, 'title');
xlab = get(gca, 'xlabel');
ylab = get(gca, 'ylabel');
set(t, 'FontSize', 15)
set(xlab, 'FontSize', 15)
set(ylab, 'FontSize', 15)
% ylim([-45 0])
% xlim([25 50])
Xlims = get(gca,'xlim');
Ylims = get(gca,'ylim');
text((Xlims(2)-Xlims(1))*1/2+Xlims(1),(Ylims(2)-
Ylims(1))*1/3+Ylims(1),GraphText,'EdgeColor','black','BackgroundColor',
[1 1 1])

uicontrol('Style', 'pushbutton', 'String', 'Plot Displacement',...
        'Position', [250 60 100 20],...
        'Callback', 'PlotDynamicDisplacement_ARRAY_FD_SYM');        %
Pushbutton string callback
```

```
                                                                        %
that calls a MATLAB function

% CalculateBandwidth(3,1000)

%% Plot Impedance, Admittance, and Noise Spectrum

%User inputs
NumMems = 4;     %Number of membranes to calculate
admittance/impedance/noise over
ActiveElemNodes=[1+2*ReducedNodes:3*ReducedNodes
1+5*ReducedNodes:6*ReducedNodes];   %Node indices of membranes to
calculate admittance/impedance/noise over

Force = 2*sum(Kme_FD(ActiveElemNodes,:)*-1*StepX.*StepY);
vel_AC_avg_LIVE =
sum(vel_AC(ActiveElemNodes,:).*StepX.*StepY)/(2*Mem_Area);

%Calculated Values
Zm = Force./vel_AC_avg_LIVE;

Elec_Area_Live = NumMems*Elec_Area;
ElecDisp=Electrode.*z(:,iter);  %Create zero displacements for all
nodes not on electrodes
ElecDisp=ElecDisp(ElecDisp~=0); %Eliminate zero entries
Transform = V0*Elec_Area_Live*eps_0/(mean(h_eq0+ElecDisp))^2;

NoiseCurrent2=(real(Zm)./(real(Zm).^2+imag(Zm).^2))*4*kT*Transform^2;
figure(9)
hold on
plot(FreqList,abs(NoiseCurrent2))
grid on
xlabel('Frequency(MHz)')
ylabel('Input Noise Current Squared (A^2/Hz)')
%ylim([0 1.2e-25])
title('Input Noise Current vs. Frequency')
Xlims = get(gca,'xlim');
Ylims = get(gca,'ylim');
text((Xlims(2)-Xlims(1))*2/3+Xlims(1),(Ylims(2)-
Ylims(1))*2/3+Ylims(1),GraphText,'EdgeColor','black','BackgroundColor',
[1 1 1])


Z1 = Zm/Transform^2;
Z2 = 1./(1i*2*pi.*FreqList*1e6*NumMems*C0);
Z3 = 1./(1i*2*pi.*FreqList*1e6*Cp);
Ztot = Z1.*Z2.*Z3./(Z2.*Z3 + Z1.*Z3 + Z1.*Z2);

figure(10)
hold on
plot(FreqList,abs(real(Ztot)),'LineWidth',2)
grid on
xlabel('Frequency(MHz)')
ylabel('Real Impedance (Ohms)')
%ylim([0 1.2e-25])
```

```
Xlims = get(gca,'xlim');
Ylims = get(gca,'ylim');
text((Xlims(2)-Xlims(1))*2/3+Xlims(1),(Ylims(2)-
Ylims(1))*2/3+Ylims(1),GraphText,'EdgeColor','black','BackgroundColor',
[1 1 1])

figure
hold on
plot(FreqList,abs(real(1./Ztot))/1e-6,'LineWidth',2)
grid on
xlabel('Frequency(MHz)')
ylabel('Real Admittance (uS)')
%ylim([0 1.2e-25])
Xlims = get(gca,'xlim');
Ylims = get(gca,'ylim');
text((Xlims(2)-Xlims(1))*2/3+Xlims(1),(Ylims(2)-
Ylims(1))*2/3+Ylims(1),GraphText,'EdgeColor','black','BackgroundColor',
[1 1 1])

%% Plot frequency response of pressure at height z above array
if CalculateRayleighInt
    PressureAtZ_avg(2,:) = 20*log10(PressureAtZ_avg(1,:)) +
Input_pulse_FFT_interp - 20*log10(1);
    PressureAtZ_avg(2,:) = PressureAtZ_avg(2,:) -
max(PressureAtZ_avg(2,:));
    figure(12);
    set(gcf,'toolbar','figure')
    hold on
    plot(FreqList,PressureAtZ_avg(2,:),'-*')
    hold off
    grid on
    xlabel('Frequency (MHz)')
    ylabel(['Average pressure at ', num2str(RayleighPlane_Zloc/1e-6), '
um (dB)'])
    title(['Average pressure of array at ',
num2str(RayleighPlane_Zloc/1e-6), ' um vs. Frequency'])
%     ylim([-30 0])
    %xlim([2 20])
    Xlims = get(gca,'xlim');
    Ylims = get(gca,'ylim');
    text((Xlims(2)-Xlims(1))*2/3+Xlims(1),(Ylims(2)-
Ylims(1))*2/3+Ylims(1),GraphText,'EdgeColor','black','BackgroundColor',
[1 1 1])

    uicontrol('Style', 'pushbutton', 'String', 'Plot Planar Pressure
Field',...
        'Position', [250 80 140 20],...
        'Callback', 'PlotPressureAtZ_rayleigh_FD');        % Pushbutton
string callback
                                                                    %
that
                                                                    %
calls a MATLAB function

    uicontrol('Style', 'pushbutton', 'String', 'Plot Angular Beam
Pattern',...
```

```
        'Position', [250 60 140 20],...
        'Callback', 'PlotAngularRadiation');        % Pushbutton string
callback
                                                                        %
that
                                                                        %
calls a MATLAB function
    %PlotPressureAtZ_rayleigh_FD  ==> Use this function to plot 2D
    %radiation pattern at frequency designated by datatip

    %PlotAngularRadiation  ==> Use this function to plot angular
    %radiation pattern at frequency designated by datatip
    %Use 2 datatips with this function to describe a frequency range
for
    %the beam pattern calculation

end

%% Plot Pulse-Echo FFT
if Plot_Pulse_Echo_FFT
    figure(14);
    cla
    hold on
    plot(FreqList,PressureAtZ_avg(2,:)+ Input_pulse_FFT_interp -
20*log10(1));
    hold off
    grid on
    xlabel('Frequency (MHz)')
    ylabel(['Average pressure from pulse-echo FFT (dB)'])
    title(['Pulse-echo FFT of array from reflector at ',
num2str(RayleighPlane_Zloc/1e-6/2), ' um away'])
%     ylim([-30 0])
    %xlim([2 20])
    Xlims = get(gca,'xlim');
    Ylims = get(gca,'ylim');
    text((Xlims(2)-Xlims(1))*2/3+Xlims(1),(Ylims(2)-
Ylims(1))*2/3+Ylims(1),GraphText,'EdgeColor','black','BackgroundColor',
[1 1 1])
end
%%
SIMTIME = toc;
% close all
% save(['FD_Analysis_results_2MEMSinFluid_1Active_100MHz_1000samp'])
% clc
% clear all
```

SubFunction: Calculate_FreqResponseAdjustedByInputFFT

```
%This codes creates a time domain signal matching the input parameters
%The code takes the FFT of this input and then multiplies it with the
%velocity frequency response

function [Input_pulse_FFT_interp, Input_pulse_FFT_interp_complex] =
Calculate_FreqResponseAdjustedByInputFFT(InputType,A1,A2,A3,A4,A5)

global FreqList

switch lower(InputType)
    case 'cw'
        vAC = A1;
        disp(['Continuous wave was used at all frequencies with Vp =
',num2str(vAC),'V'])
        Input_pulse_FFT_interp =
20*log10(abs(vAC))*ones(size(FreqList));
        Input_pulse_FFT_interp_complex(:,1) = vAC*ones(size(FreqList));
        Input_pulse_FFT_interp_complex(:,2) =
zeros(length(FreqList),1);
        return
    case 'pulse'

        Amp = A1;                %V
        Pulsewidth = A2*1e-9;    %sec
        PulseStart = A3*1e-9;    %sec
        Alpha = A4;

        Time_step = Pulsewidth/100;  %Resolve pulse with 1000 points
        t = 0:Time_step:(PulseStart+Pulsewidth*30); %make time signal
30x as long as the pulse
        Input_pulse = ones(size(t));
        Input_pulse((t>=PulseStart) & (t<=(PulseStart+Pulsewidth))) =
Input_pulse((t>=PulseStart) &
(t<=(PulseStart+Pulsewidth))).*gausswin(length(t((t>=PulseStart) &
(t<=(PulseStart+Pulsewidth)))),Alpha)';  %Apply Guassian window (if
applicable) to toneburst
        Input_pulse(t<PulseStart) = 0;
        Input_pulse(t>(PulseStart + Pulsewidth)) = 0;
        figure(50)
        subplot(1,2,1)
        plot(t./1e-6,Input_pulse)
        grid on
        title('Input Signal in time domain')
        ylim([-Amp 2*Amp])
%         xlim([0 Pulsewidth*5])
        xlabel('time (usec)')
        ylabel('Amplitude (V)')

        disp(['Input type was pulse with amplitude of ', num2str(A1),
'V, and pulsewidth of ', num2str(A2), 'ns'])
        disp('Frequency response has been adjusted accordingly')

    case 'toneburst'
```

```
        Freq_cent = A1*1e6;  %Hz
        Amp = A2;            %V
        NumCycles = A3;      %cycles
        Alpha = A4;
        BurstStart = A5*1e-6; %ns

        lambda = 1/(Freq_cent);
        Time_step = lambda/100;                  %Resolve each
wavelength with 1000 points
        t = [0:Time_step:(BurstStart+30*lambda*NumCycles)];   %make
time signal 30x as long as the pulse
        Input_pulse = Amp*sin(2*pi*Freq_cent*t);    %Create infinite
sine wave
        Input_pulse((t>=BurstStart) &
(t<=(BurstStart+lambda*NumCycles))) = Input_pulse((t>=BurstStart) &
(t<=(BurstStart+lambda*NumCycles))).*gausswin(length(t((t>=BurstStart)
& (t<=(BurstStart+lambda*NumCycles)))),Alpha)';  %Apply Guassian window
(if applicable) to toneburst
        Input_pulse(t<BurstStart) = 0;
        Input_pulse(t>(BurstStart + lambda*NumCycles)) = 0;
        figure(50)
        subplot(1,2,1)
        plot(t/1e-6,Input_pulse)
        grid on
        title('Input Signal in time domain')
        xlabel('time (usec)')
        ylabel('Amplitude (V)')

        disp(['Input type was tone burst centered at ', num2str(A1),
'MHz, with peak amplitude of ', num2str(A2), 'V and ', num2str(A3),
'cycles'])
        disp('Frequency response has been adjusted accordingly')

    case 'fromfigfile'
        hgload(A1);  %Load file into variable

        [t,Input_pulse] = GetFigureData(gcf);
        title('Input Signal in time domain')
        xlabel('time (usec)')
        ylabel('Amplitude (V)')

        Xlims = get(gca,'xlim');
        Ylims = get(gca,'ylim');

        h=text((Xlims(2)-Xlims(1))*1/2+Xlims(1),(Ylims(2)-
Ylims(1))*1/3+Ylims(1),'Define Range Containing DC Bias
Level','EdgeColor','black','BackgroundColor',[1 1 1]);
        [t_DC,~] = ginput(2);            %Get time Range containing DC
bias
        delete(h);

        h=text((Xlims(2)-Xlims(1))*1/2+Xlims(1),(Ylims(2)-
Ylims(1))*1/3+Ylims(1),'Define Range For
FFT','EdgeColor','black','BackgroundColor',[1 1 1]);
```

```
        [t_FFT,~] = ginput(2);              %Get time range for FFT
        delete(h);

        close

        Time_step = t(2)-t(1);

        [~, t_DC_Index(1)] = min(abs(t-min(t_DC)));
        [~, t_DC_Index(2)] = min(abs(t-max(t_DC)));

        [~, t_Input(1)] = min(abs(t-min(t_FFT)));
        [~, t_Input(2)] = min(abs(t-max(t_FFT)));

        V_DC = mean(Input_pulse(t_DC_Index(1):t_DC_Index(2)));
%Calculate DC Bias level
        Input_pulse = Input_pulse(t_Input(1):t_Input(2)) - V_DC;
%Subtract DC bias from input signal, and snip to only contain
designated region
        t = t(t_Input(1):t_Input(2));  %Redefine time region to only
include designated range

        figure(50)
        subplot(1,2,1)
        plot(t/1e-6,Input_pulse)
        grid on
        title('Input Signal in time domain')
        xlabel('time (usec)')
        ylabel('Amplitude (V)')

        disp('Input pulse was described by Matlab .fig file ')
        disp('Frequency response has been adjusted accordingly')

    otherwise
        disp('Excitation input type is unknown.')
        disp('A continuous wave, 1V AC signal was applied to all
electrodes.')
        Input_pulse_FFT_interp = 20*log10(1)*ones(size(FreqList));
%Ensures that freq response will not be adjusted
        Input_pulse_FFT_interp_complex(:,1) = ones(length(FreqList),1);
        Input_pulse_FFT_interp_complex(:,2) =
zeros(length(FreqList),1);
        return
end

Fs = 1/Time_step;
L = length(Input_pulse);
NFFT = 2^(nextpow2(L)+5); % Next power of 2 from length of y
Y = fft(Input_pulse,NFFT)/L;
f = linspace(0,Fs/2,NFFT/2+1);

Input_pulse_FFT_interp_complex(1,:) =
interp1(f/1e6,real(2*Y(1:NFFT/2+1)),FreqList);  %Used for making a
time-varying movie
```

```
Input_pulse_FFT_interp_complex(2,:) =
interp1(f/1e6,imag(2*Y(1:NFFT/2+1)),FreqList);  %Used for making a
time-varying movie

Input_pulse_FFT_interp_complex = Input_pulse_FFT_interp_complex(1,:) +
1i*Input_pulse_FFT_interp_complex(2,:);  %Used for making a time-
varying movie

Input_pulse_FFT_interp = 20*log10(abs(Input_pulse_FFT_interp_complex));
% Input_pulse_FFT_interp =
interp1(f/1e6,Input_pulse_FFT_interp,FreqList);  %Used for making a
time-varying movie


subplot(1,2,2)
plot(FreqList,Input_pulse_FFT_interp-max(Input_pulse_FFT_interp))
grid on
title('FFT of Input Signal')
xlabel('Frequency (MHz)')
ylabel('Amplitude (dB)')

button = questdlg('Frequency response will be adjusted by this input
signal.  Continue with evaluation?');

switch button
    case 'Yes'
    otherwise
        Input_pulse_FFT_interp = [];
        Input_pulse_FFT_interp_complex = [];
        return
end
```

SubFunction: Calculate_Static_Displacement

```
%Calculate the static displacement with applied voltage

% 1. Begin with initial guess for displacement matrix (0 displacement)
% 2. Then find corresponding electrostatic force
% 3. Then update displacement matrix by calculating mechanical reaction
to
% electrostatic force
% 4. Continue iterating by updating electrostatic force then
displacement
% 5. Finish iterating when change in displacement matrix goes below
certain
% threshold

function [z P_e iter] = Calculate_Static_Displacement(V,P_a)

global Electrode
global eps_0
global h_eq
global iterMax
global ReducedNodes
global Kmm

%Initial guess (zero vector)
z = zeros(ReducedNodes,iterMax);       %z is "ReducedNodes" long, but
the last node is always zero
P_e = zeros(ReducedNodes,iterMax);

for iter = 1:iterMax     %Sets maximum iteration to avoid infinite loop

    disp(['Iteration #', num2str(iter)])

    if iter == 1

        P_e(:,iter) = -.5.*Electrode*eps_0.*V^2./(h_eq(:) +
z(:,iter)).^2;

    else

        z(:,iter) = Kmm\(P_e(:,iter-1) - P_a);

        P_e(:,iter) = -.5.*Electrode*eps_0.*V^2./(h_eq(:) +
z(:,iter)).^2;

        zAVG = mean(z(:,iter));
        zAVG0 = mean(z(:,iter-1));
        if abs((zAVG - zAVG0)/zAVG) < .001
            break
        end

    end

end
```

```
%Delete all non-zero columns (except initial guess)
% z=z(:,any(z));
% P_e=P_e(:,any(P_e));
%
% z=[zeros(size(z(:,1))) z];
% P_e=[zeros(size(P_e(:,1))) P_e];
```

SubFunction: CalculateBandwidth

```
function CalculateBandwidth(figNum,Resolution)

[Freqs,Magnitudes]=getfigdata(figNum);  %Get X,Y data from figure

Xlims = get(gca,'xlim');
Ylims = get(gca,'ylim');
h=text((Xlims(2)-Xlims(1))*1/2+Xlims(1),(Ylims(2)-
Ylims(1))*1/3+Ylims(1),'Define Range Containing Center
Frequency','EdgeColor','black','BackgroundColor',[1 1 1]);
[freqC_temp,~] = ginput(2);            %Get Freq Range containing
center frequency
delete(h);

[~, freqC_Index(1)] = min(abs(Freqs-min(freqC_temp)));
[~, freqC_Index(2)] = min(abs(Freqs-max(freqC_temp)));

Mag_max = max(Magnitudes(freqC_Index(1):freqC_Index(2)));
Freq_max = interp1(Magnitudes,Freqs,Mag_max);
Magnitudes = Magnitudes-Mag_max;                            %Normalize
magnitudes so search algorithm starts from 0 magnitude

for freq_H = Freq_max:(Freqs(end)-Freqs(1))/Resolution:Freqs(end)
    dB_mag = interp1(Freqs,Magnitudes,freq_H);
    if dB_mag < -3
        break
    end
end

for freq_L = Freq_max:-1*(Freqs(end)-Freqs(1))/Resolution:Freqs(1)
    dB_mag = interp1(Freqs,Magnitudes,freq_L);
    if dB_mag < -3
        break
    end
end

Freq_cent = (freq_H - freq_L)/2 + freq_L;

Bandwidth = (freq_H - freq_L)/Freq_cent*100;

disp(' ')
disp(' ')
disp(['Center Frequency = ', num2str(Freq_cent), ' MHz'])
disp(' ')
disp(['Bandwidth = ', num2str(Bandwidth), ' %'])
disp(' ')

end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
function [X,Y] = getfigdata(a)

axs = get(a, 'Children');

for ind=1:length(axs)

    if strcmp(get(axs(ind),'Type'),'axes')
        break
    end

end

pos = get(axs(ind), 'Children');

for ind=1:length(pos)
    if strcmp(get(pos(ind),'Type'),'line')
        break
    end
end


X = get(pos(ind), 'XData');
Y = get(pos(ind), 'YData');

end
```

Sub-Function: CalculateKmm_FD

```
function [Kmm] = CalculateKmm_FD(StepX, StepY, NodesX, NodesY,
TotalNodes, D, v)

% Solve for stiffness matrix using finite differences
% Kmm begins as a square matrix with "TotalNodes" nodes

Kmm = zeros(TotalNodes,TotalNodes);
Fixed = ones(TotalNodes,1);

count=1;
for y = 0:StepY:(NodesY-1)*StepY
    for x = 0:StepX:(NodesX-1)*StepX

        C1  = D(count,1)/(StepX^4);
        C2  = D(count,1)/(StepY^4);
        C3  = (D(count,1))/(9*StepX^2*StepY^2);
        C4  = (D(count,2))/(StepX^3);
        C5  = (D(count,3))/(StepY^3);
        C6  = (D(count,3))/(3*StepX^2*StepY);
        C7  = (D(count,2))/(3*StepX*StepY^2);
        C8  = (D(count,4) + v(count,1)*D(count,5) +
D(count,1)*v(count,5) + 2*D(count,3)*v(count,3))/(StepX^2);
        C9  = (D(count,5) + v(count,1)*D(count,4) +
D(count,1)*v(count,4) + 2*D(count,3)*v(count,3))/(StepY^2);
        C10 = (D(count,6) - v(count,1)*D(count,6) -
D(count,3)*v(count,2) - D(count,2)*v(count,3) -
D(count,1)*v(count,6))/(2*StepX*StepY);

        im2 = -2;
        im1 = -1;
        im0 =  0;
        ip1 =  1;
        ip2 =  2;
        jm2 = -2;
        jm1 = -1;
        jm0 =  0;
        jp1 =  1;
        jp2 =  2;

        km2m2 = C3;
        km2m1 = -C3 - C6;
        km2m0 = C1 - C4;
        km2p1 = -C3 + C6;
        km2p2 = C3;
        km1m2 = -C3 - C7;
        km1m1 = C3 + C6 + C7 + C10;
        km1m0 = -4*C1 + 2*C4 + C8;
        km1p1 = C3 - C6 + C7 - C10;
        km1p2 = -C3 - C7;
        km0m2 = C2 - C5;
        km0m1 = -4*C2 + 2*C5 + C9;
        km0m0 = 6*C1 + 6*C2 - 2*C8 - 2*C9;
        km0p1 = -4*C2 - 2*C5 + C9;
```

```
        km0p2 = C2 + C5;
        kp1m2 = -C3 + C7;
        kp1m1 = C3 + C6 - C7 - C10;
        kp1m0 = -4*C1 - 2*C4 + C8;
        kp1p1 = C3 - C6 - C7 + C10;
        kp1p2 = -C3 + C7;
        kp2m2 = C3;
        kp2m1 = -C3 - C6;
        kp2m0 = C1 + C4;
        kp2p1 = -C3 + C6;
        kp2p2 = C3;

        if (x == 0)
            im2 = 0;
            im1 = 0;
        elseif (x == StepX)
            im2 = 0;
        elseif (x > StepX) && (x < (NodesX-2)*StepX)

        elseif (x == (NodesX-2)*StepX)
            ip2 = 0;
        elseif (x == (NodesX-1)*StepX)
            Fixed(count,1) = 0; %Fixed locations are set to zero
            ip1 =  0;
            ip2 =  0;
        end

        if y == 0
            jm2 = 0;
            jm1 = 0;
        elseif y == StepY
            jm2 = 0;
        elseif (y > StepY) && (y < (NodesY-2)*StepY)

        elseif (y == (NodesY-2)*StepY)
            jp2 = 0;
        elseif (y == (NodesY-1)*StepY)
            jp1 = 0;
            jp2 = 0;
            Fixed(count,1) = 0; %Fixed locations are set to zero
        end

        Kmm(count,count+im2+jm2*NodesX) =
Kmm(count,count+im2+jm2*NodesX) + km2m2;
        Kmm(count,count+im2+jm1*NodesX) =
Kmm(count,count+im2+jm1*NodesX) + km2m1;
        Kmm(count,count+im2+jm0*NodesX) =
Kmm(count,count+im2+jm0*NodesX) + km2m0;
        Kmm(count,count+im2+jp1*NodesX) =
Kmm(count,count+im2+jp1*NodesX) + km2p1;
        Kmm(count,count+im2+jp2*NodesX) =
Kmm(count,count+im2+jp2*NodesX) + km2p2;
        Kmm(count,count+im1+jm2*NodesX) =
Kmm(count,count+im1+jm2*NodesX) + km1m2;
        Kmm(count,count+im1+jm1*NodesX) =
Kmm(count,count+im1+jm1*NodesX) + km1m1;
```

```
        Kmm(count,count+im1+jm0*NodesX) =
Kmm(count,count+im1+jm0*NodesX) + km1m0;
        Kmm(count,count+im1+jp1*NodesX) =
Kmm(count,count+im1+jp1*NodesX) + km1p1;
        Kmm(count,count+im1+jp2*NodesX) =
Kmm(count,count+im1+jp2*NodesX) + km1p2;
        Kmm(count,count+im0+jm2*NodesX) =
Kmm(count,count+im0+jm2*NodesX) + km0m2;
        Kmm(count,count+im0+jm1*NodesX) =
Kmm(count,count+im0+jm1*NodesX) + km0m1;
        Kmm(count,count+im0+jm0*NodesX) =
Kmm(count,count+im0+jm0*NodesX) + km0m0;
        Kmm(count,count+im0+jp1*NodesX) =
Kmm(count,count+im0+jp1*NodesX) + km0p1;
        Kmm(count,count+im0+jp2*NodesX) =
Kmm(count,count+im0+jp2*NodesX) + km0p2;
        Kmm(count,count+ip1+jm2*NodesX) =
Kmm(count,count+ip1+jm2*NodesX) + kp1m2;
        Kmm(count,count+ip1+jm1*NodesX) =
Kmm(count,count+ip1+jm1*NodesX) + kp1m1;
        Kmm(count,count+ip1+jm0*NodesX) =
Kmm(count,count+ip1+jm0*NodesX) + kp1m0;
        Kmm(count,count+ip1+jp1*NodesX) =
Kmm(count,count+ip1+jp1*NodesX) + kp1p1;
        Kmm(count,count+ip1+jp2*NodesX) =
Kmm(count,count+ip1+jp2*NodesX) + kp1p2;
        Kmm(count,count+ip2+jm2*NodesX) =
Kmm(count,count+ip2+jm2*NodesX) + kp2m2;
        Kmm(count,count+ip2+jm1*NodesX) =
Kmm(count,count+ip2+jm1*NodesX) + kp2m1;
        Kmm(count,count+ip2+jm0*NodesX) =
Kmm(count,count+ip2+jm0*NodesX) + kp2m0;
        Kmm(count,count+ip2+jp1*NodesX) =
Kmm(count,count+ip2+jp1*NodesX) + kp2p1;
        Kmm(count,count+ip2+jp2*NodesX) =
Kmm(count,count+ip2+jp2*NodesX) + kp2p2;

        count = count + 1;
    end
end

Kmm(:,1+(NodesY-1)*NodesX:TotalNodes) = []; %Delete node columns for
fixed y edge (top)
Kmm(1+(NodesY-1)*NodesX:TotalNodes,:) = []; %Delete node rows for fixed
y edge (top)

Kmm(NodesX:NodesX:NodesX*(NodesY-1),:) = [];       %Delete node rows
for fixed x edge (right)
Kmm(:,NodesX:NodesX:NodesX*(NodesY-1)) = [];       %Delete node columns
for fixed x edge (right)

Kmm(:,1:(NodesX-1)) = []; %Delete node columns for fixed y edge
(bottom)
Kmm(1:(NodesX-1),:) = []; %Delete node rows for fixed y edge (bottom)
```

```
Kmm(1:(NodesX-1):(NodesX-1)*(NodesY-2),:) = [];       %Delete node rows
for fixed x edge (left)
Kmm(:,1:(NodesX-1):(NodesX-1)*(NodesY-2)) = [];       %Delete node
columns for fixed x edge (left)
```

Sub-Function: CalculateKr2_FD

```
%Fluid Self Radiation Impedance Calculator

function [Kfluid] = CalculateKr2_FD(StepX, StepY, NodesX, NodesY,
ReducedNodes, w0, Vacuum, Use_water)

global Fluid_alpha_props


ReducedNodes = (NodesX-2)*(NodesY-2);
Kfluid = zeros(ReducedNodes,ReducedNodes);

if Vacuum == 0

    Water_rho    = 1000;  %[kg/m^3]
    Water_c      = 1500;  %[m/s]
    Air_rho      = 1.2;   %[kg/m^3]
    Air_c        = 343;   %[m/s]

    if Use_water == 1
        Fluid_rho = Water_rho;
        Fluid_c   = Water_c;
    else
        Fluid_rho = Air_rho;
        Fluid_c     = Air_c;
    end

    dS = StepX*StepY;
    k0 = w0/Fluid_c;

    if length(Fluid_alpha_props) == 1
        Fluid_alpha = Fluid_alpha_props(1)/8.7*(w0/2/pi/1e6);
    else
        Fluid_alpha =
Fluid_alpha_props(1)*(w0/2/pi)^Fluid_alpha_props(2);  %[Np/m]
    end

    X = zeros(ReducedNodes,1);
    Y = zeros(ReducedNodes,1);

    count = 1;          %Current node position
    for y = StepY:StepY:(NodesY-2)*StepY
        for x = StepX:StepX:(NodesX-2)*StepX
            X(count) = x;
            Y(count) = y;

            count = count + 1;
        end
    end

    [Xj Xi] = meshgrid(X);
    [Yj Yi] = meshgrid(Y);
```

```
    %When i ~= j, use Green's function for baffled point source in
semi-infinite fluid
    rij = sqrt((Xi - Xj).^2 + (Yi - Yj).^2); %Calculate distance
between node and mutual node
    Kfluid = 1i.*Fluid_rho.*w0./(2.*pi).*exp(-
1i.*rij.*k0).*dS./rij.*exp(-Fluid_alpha.*rij);


    %when i = j, approximate fluid stiffness using infinitely small
baffled piston impedance
    a_eq = sqrt(dS/pi);
    Kfluid(1:(ReducedNodes+1):end) = Fluid_rho*Fluid_c*(.5*(k0*a_eq)^2
+ 1i*8/(3*pi)*(k0*a_eq));

end
```

Sub-Function: CalculateRayleigh_FDsym

```
%% define surface velocity
function [p xlims ylims] = CalculateRayleigh_FDsym(vel_ARRAY, w0)

global X_m
global Y_m
global NodesX
global NodesY
global NumMemX
global NumMemY
global PitchX
global PitchY
global StepX
global StepY
global Fluid_c
global Fluid_rho
global RayleighPlane_X_cent
global RayleighPlane_Y_cent
global RayleighPlane_LengthX
global RayleighPlane_LengthY
global RayleighPlane_NodesX
global RayleighPlane_NodesY
global Fluid_alpha_props
global ReducedNodes
global symx
global symy

%% User Inputs
%grid_space_z0 = 5e-6;

% RayleighPlane_X_cent = (12*80*1/2)*1e-6;
% RayleighPlane_Y_cent = (12*80*3/2)*1e-6;
% RayleighPlane_LengthX = (12*80)*1e-6;
% RayleighPlane_LengthY = (12*80)*1e-6;
% RayleighPlane_NodesX = 70;
% RayleighPlane_NodesY = 70;
% RayleighPlane_Zloc = 18000e-6;

if length(Fluid_alpha_props) == 1
    Fluid_alpha = Fluid_alpha_props(1)/8.7*(w0/2/pi/1e6);
%[Np/m]
else
    Fluid_alpha = Fluid_alpha_props(1)*(w0/2/pi)^Fluid_alpha_props(2);
%[Np/m]
end

%% Define constants
grid_space_Xz = RayleighPlane_LengthX/(RayleighPlane_NodesX-1);
grid_space_Yz = RayleighPlane_LengthY/(RayleighPlane_NodesY-1);
k=w0/Fluid_c;
dS = StepX*StepY;
B=1i*k*Fluid_rho*Fluid_c/2/pi*dS;
```

```
%Define mesh of planar location at given z_Rayleigh (if symmetry
exists, override center values)
X_z = RayleighPlane_X_cent-
RayleighPlane_LengthX/2:grid_space_Xz:RayleighPlane_X_cent+RayleighPlan
e_LengthX/2;
Y_z = RayleighPlane_Y_cent-
RayleighPlane_LengthY/2:grid_space_Yz:RayleighPlane_Y_cent+RayleighPlan
e_LengthY/2;

if symx
%     Ysym = Y_m*NumMemY/2 + MemSpaceY*(NumMemY-1)/2; %Override center
value and define z-plane center based on symmetry axis
%     Y_z = Ysym-RayleighPlane_LengthY:grid_space_Yz:Ysym;

    symZERO1 = ones(ReducedNodes,1);
    symZERO2 = ones(ReducedNodes,1);
    if mod(NumMemX,2)
        count = 1;

        for y = StepY:StepY:Y_m-StepY
            %zero out all nodes on the opposite side of the axis

            %(center node on axis remains non-zero)
            symZERO2((count)*(NodesX-1)/2 + (count-1)*(NodesX-
2):(count)*(NodesX-2)) = 0;

            %(center node on axis set to zero)
            symZERO2((count)*(NodesX-3)/2 + (count-1)*(NodesX-
2):(count)*(NodesX-2)) = 0;
            count = count + 1;
        end
    end
end

if symy
%     Xsym = X_m*NumMemX/2 + MemSpaceX*(NumMemX-1)/2; %Override center
value and define z-plane center based on symmetry axis
%     X_z = Xsym-RayleighPlane_LengthX:grid_space_Xz:Xsym;

    symZERO3 = ones(ReducedNodes,1);
    symZERO4 = ones(ReducedNodes,1);
    if mod(NumMemX,2)
        %zero out all nodes on the opposite side of the axis

        %(center node on axis remains non-zero)
        symZERO3(1 + (NodesX-2)*(NodesY-1)/2:ReducedNodes) = 0;

        %(center node on axis set to zero)
        symZERO4(1 + (NodesX-2)*(NodesY-3)/2:ReducedNodes) = 0;
    end
end

xlims = [X_z(1) X_z(end)];   %Used for plotting purposes
ylims = [Y_z(1) Y_z(end)];
```

```
[X_z, Y_z, ~] = meshgrid(X_z,Y_z,ones(ReducedNodes,1));
p=X_z(:,:,1)-X_z(:,:,1);

%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                               Perform Rayleigh integral
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
% Define mesh for surface velocity
X = StepX:StepX:X_m-StepX;
Y = StepY:StepY:Y_m-StepY;
[X Y] = meshgrid(X,Y);
symZEROx=[];
symZEROy=[];
if symx
    symZEROx=symZERO1;
end
if symy
    symZEROy=symZERO3;
end
[p] =
RayleighLoop(X,Y,X_z,Y_z,vel_ARRAY,p,k,B,Fluid_alpha,symZEROx,symZEROy,
PitchX,PitchY);



% For y-axis symmetry, run Rayleigh integral loop again with reversed
location values
if symy
    Xt = fliplr(X) + PitchX*(NumMemX-1);
    PitchXt = -PitchX;
    symZEROy=symZERO4;
    [p] =
RayleighLoop(Xt,Y,X_z,Y_z,vel_ARRAY,p,k,B,Fluid_alpha,symZEROx,symZEROy
,PitchXt,PitchY);
end


% For x-axis symmetry, run Rayleigh integral loop again with reversed
location values
if symx
    Yt = flipud(Y) + PitchY*(NumMemY-1);
    PitchYt = -PitchY;
    symZEROx=symZERO2;
    [p] =
RayleighLoop(X,Yt,X_z,Y_z,vel_ARRAY,p,k,B,Fluid_alpha,symZEROx,symZEROy
,PitchX,PitchYt);
end


% For quarter symmetry, run Rayleigh integral loop again with reversed
location values
if symx && symy
```

```
    [p] =
RayleighLoop(Xt,Yt,X_z,Y_z,vel_ARRAY,p,k,B,Fluid_alpha,symZEROx,symZERO
y,PitchXt,PitchYt);
end




function [p] =
RayleighLoop(X,Y,X_z,Y_z,vel_ARRAY,p,k,B,Fluid_alpha,symZEROx,symZEROy,
PitchX,PitchY)
%% Rayleigh Integral loop is a sub-function for code simplification
global NumMemX
global NumMemY
global NumMemXi
global NumMemYi
global RayleighPlane_NodesX
global RayleighPlane_NodesY
global RayleighPlane_Zloc
global ReducedNodes
global symx
global symy

[~, ~, X] =
meshgrid(ones(RayleighPlane_NodesX,1),ones(RayleighPlane_NodesY,1),X);
[~, ~, Y] =
meshgrid(ones(RayleighPlane_NodesX,1),ones(RayleighPlane_NodesY,1),Y);

count = 1;
for j = 1:NumMemYi
    for i = 1:NumMemXi

        X_z0 = X+(i-1)*PitchX;
        Y_z0 = Y+(j-1)*PitchY;

        %Calculate Rayleigh for each combination of node locations
        r=sqrt(RayleighPlane_Zloc^2+(Y_z0-Y_z).^2+(X_z0-X_z).^2);

        %If symmetry exists and a membrane was split in half, zero out
all
        %nodes on the opposite side of the axis
        if (i == NumMemXi) && (symy)
            if mod(NumMemX,2)
                vel_ARRAY(1+(count-
1)*ReducedNodes:(count)*ReducedNodes) = symZEROy.*vel_ARRAY(1+(count-
1)*ReducedNodes:(count)*ReducedNodes);
            end
        elseif (j == NumMemYi) && (symx)
            if mod(NumMemY,2)
                vel_ARRAY(1+(count-
1)*ReducedNodes:(count)*ReducedNodes) = symZEROx.*vel_ARRAY(1+(count-
1)*ReducedNodes:(count)*ReducedNodes);
            end
        end
```

```
        [~, ~, velocity] =
meshgrid(ones(RayleighPlane_NodesX,1),ones(RayleighPlane_NodesY,1),vel_
ARRAY(1+(count-1)*ReducedNodes:(count)*ReducedNodes));

        %Sum contributions from each node as coherent sources
        p = p + sum(B.*velocity.*exp(-1i*k*r)./r.*exp(-
Fluid_alpha.*r),3); %Sums along third dimension
        count = count+1;

    end
end
```

Sub-Function: CalculateZ_Array_FD_SYM

```
%Fluid Self Radiation Impedance Calculator

function [Z_mut] = CalculateZ_Array_FD_SYM(Z_lump,w0)

global NodesX
global NodesY
global NumMemX
global NumMemY
global StepX
global StepY
global PitchX
global PitchY
global Fluid_c
global Fluid_rho
global symy
global symx
global Fluid_alpha_props
global Vacuum

ReducedNodes = (NodesX-2)*(NodesY-2);

dS = StepX*StepY;
k0 = w0/Fluid_c;

if length(Fluid_alpha_props) == 1
    Fluid_alpha = Fluid_alpha_props(1)/8.7*(w0/2/pi/1e6);
else
    Fluid_alpha = Fluid_alpha_props(1)*(w0/2/pi)^Fluid_alpha_props(2);
%[Np/m]
end



X = zeros(ReducedNodes,1);
Y = zeros(ReducedNodes,1);

count = 1;          %Current node position
for y = StepY:StepY:(NodesY-2)*StepY
    for x = StepX:StepX:(NodesX-2)*StepX
        X(count) = x;
        Y(count) = y;

        count = count + 1;
    end
end

[Xj Xi] = meshgrid(X);
[Yj Yi] = meshgrid(Y);
%% Symmetry Stuff
if symy
    NumMemXi = ceil(NumMemX/2);
```

```
    if (mod(NumMemX,2) == 0)
        SymIndX = 1;
    else
        SymIndX = 0;
    end

else
    NumMemXi = NumMemX;
end

if symx
    NumMemYi = ceil(NumMemY/2);

    if (mod(NumMemY,2) == 0)
        SymIndY = 1;
    else
        SymIndY = 0;
    end

else
    NumMemYi = NumMemY;
end
TotalMemsj = NumMemX*NumMemY;
TotalMemsi = NumMemXi*NumMemYi;

%Preallocate space for Z_mut matrix
Z_mut = zeros(TotalMemsi*ReducedNodes);
Z_mut = mat2cell(Z_mut, ones(1,TotalMemsi)*ReducedNodes,
ones(1,TotalMemsi)*ReducedNodes);

    %% Iterate through i and j membranes to calculate impedance values
    counti = 1;
    i_ind = 1;
    for ni = 1:NumMemY              %Iterate through base membranes
        disp(['     Calculating mutual impedance for membrane ',
num2str(counti), '/', num2str(TotalMemsi)])
        for mi = 1:NumMemX

            if (mi > NumMemXi) || (ni > NumMemYi)
                counti = counti + 1;
                continue
            end

            Xi2 = Xi + (mi-1)*PitchX;
            Yi2 = Yi + (ni-1)*PitchY;

            for nj = 1:NumMemY      %Iterate through mutual membranes
                for mj = 1:NumMemX

                    countj = mj + (nj-1)*NumMemX;

                    if counti == countj
                        j_ind = countj  - (nj-1)*(NumMemX-NumMemXi);
                        Z_mut{i_ind,j_ind} = Z_mut{i_ind,j_ind} +
Z_lump;
```

```
                        continue
                        %                 elseif countj < counti
                        %                     continue
                    elseif Vacuum == 1
                        continue
                    else

                        Xj2 = Xj + (mj-1)*PitchX;
                        Yj2 = Yj + (nj-1)*PitchY;

                        %Calculate total mutual radiation impedance
from neighbor membrane
                        if (mj > NumMemXi) && (nj > NumMemYi)
                            %                       - Reflect over y-axis
- Reflect over x-axis                -  Account for index shift
                            j_ind  =        countj - (2*(mj-NumMemXi)-
SymIndX)   - (2*(nj-NumMemYi)-SymIndY)*NumMemXi - (nj-1)*(NumMemX-
NumMemXi);
                            Xj2 = fliplr(Xj2);
                            Yj2 = fliplr(Yj2);
                        elseif (mj > NumMemXi)
                            %                       - Reflect over y-axis
- Account for index shift
                            j_ind =         countj - (2*(mj-NumMemXi)-
SymIndX)   - (nj-1)*(NumMemX-NumMemXi);
                            Xj2 = fliplr(Xj2);
                        elseif (nj > NumMemYi)
                            %                       - Reflect over x-axis
- Account for index shift
                            j_ind =         countj - (2*(nj-NumMemYi)-
SymIndY)*NumMemXi  - (nj-1)*(NumMemX-NumMemXi);
                            Yj2 = fliplr(Yj2);
                        else
                            j_ind = countj  - (nj-1)*(NumMemX-
NumMemXi);
                            %
Z_mut{j_ind,i_ind} = Z_mut{j_ind,i_ind} + Kfluid.'*1i*w0;  %Assumes
impedances are reciprocal
                        end

                        %When i ~= j, use Green's function for baffled
point source in semi-infinite fluid
                        rij = sqrt((Xi2 - Xj2).^2 + (Yi2 - Yj2).^2);
%Calculate distance between node and mutual node
                        Kfluid = 1i.*Fluid_rho.*w0./(2.*pi).*exp(-
1i.*rij.*k0).*dS./rij.*exp(-Fluid_alpha.*rij);
                        Z_mut{i_ind,j_ind} = Z_mut{i_ind,j_ind} +
Kfluid*1i*w0;  %Assumes uniform velocity distribution (like piston)

                    end
                end
            end
            i_ind = i_ind + 1;
            counti = counti + 1;
        end
    end
```

```
% Z_mut2{1,1} = Z_mut{2,1} + Z_mut{2,2};
% Z_mut2{1,2} = Z_mut{2,3} + Z_mut{2,4};
% Z_mut2{2,1} = Z_mut{4,1} + Z_mut{4,2};
% Z_mut2{2,2} = Z_mut{4,3} + Z_mut{4,4};
% Z_mut = Z_mut2;
```

Sub-Function: Plot_ARRAY_Preview_FD_SYM

```
function Plot_ARRAY_Preview_FD_SYM()

global StepX
global StepY
global X_m
global Y_m
global NodesX
global NodesY
global NumMemXi
global NumMemYi
global NumMemX
global NumMemY
global PitchX
global PitchY
global symx
global symy
global MemSpaceX
global MemSpaceY
global LiveMemsInd

X = 0:StepX:X_m;
Y = 0:StepY:Y_m;
[X Y] = meshgrid(X,Y);

figure
cla
hold on
count = 1;
for j = 1:NumMemYi
    for i = 1:NumMemXi
        X2 = X+(i-1)*PitchX;
        Y2 = Y+(j-1)*PitchY;

        if ismember(count, LiveMemsInd)
            Z = zeros(NodesY,NodesX);
            s = surface(X2/1e-6,Y2/1e-6,Z);
            set(s,'FaceColor','r')
        else
            Z = zeros(NodesY,NodesX);
            t = surface(X2/1e-6,Y2/1e-6,Z);
            set(t,'FaceColor','g')
        end



        count = count+1;
    end
end

xlabel('X position (um)')
ylabel('Y position (um)')
title(['Preview of modeled array section.  Red elements are active'])
```

```
if X_m*NumMemXi < Y_m*NumMemYi
    xlim([0 PitchY*NumMemYi/1e-6])
    ylim([0 PitchY*NumMemYi/1e-6])
else
    xlim([0 PitchX*NumMemXi/1e-6])
    ylim([0 PitchX*NumMemXi/1e-6])
end

if symy
    Xsym = X_m*NumMemX/2 + MemSpaceX*(NumMemX-1)/2;
    xdata = ones(4,1)*Xsym/1e-6;
    ydata = [0; 0; PitchY*NumMemY; PitchY*NumMemY]/1e-6;
    zdata = [1; -1; -1; 1];
    p1 = patch(xdata,ydata,zdata,0);
    set(p1,'FaceColor','g','FaceAlpha',.2,'LineWidth',3,'LineStyle','-
.')

end

if symx
    Ysym = Y_m*NumMemY/2 + MemSpaceY*(NumMemY-1)/2;
    ydata = ones(4,1)*Ysym/1e-6;
    xdata = [0; 0; PitchX*NumMemX; PitchX*NumMemX]/1e-6;
    zdata = [1; -1; -1; 1];
    p1 = patch(xdata,ydata,zdata,0);
    set(p1,'FaceColor','g','FaceAlpha',.2,'LineWidth',3,'LineStyle','-
.')

end
%xlim([0 1e-6])
%ylim([0 X_m/1e-6])
%zlim([-Sac_isol/1e-6 0])
```

Sub-Function: PlotAngularRadiation

```
function PlotAngularRadiation()

global FreqList
global zAC
global RayleighAngular_X_cent
global RayleighAngular_Y_cent
global RayleighAngular_dS
global NumNodesRad
global RayleighAngular_R
global Input_pulse_FFT_interp
global TotalMems
global TotalMemsSYM
global symx
global symy
global ReducedNodes
global LiveMemsInd

%These variables are used for plot formatting purposes
global angles
global pAngle

dcm_obj = datacursormode(12);
info_struct = getCursorInfo(dcm_obj);
FreqIndex = info_struct.DataIndex;
Freq = info_struct.Position(1);

prompt = {'Theta (deg):','X Center (m)', 'Y Center (m)','Elemental Area
(m^2)','Radius (m)','Number of Points:','Assume ideal piston for active
element? (1 = yes, 0 = no):'};
dlg_title = 'Inputs for Radiation Plot function';
num_lines = 1;
def =
{'0',num2str(RayleighAngular_X_cent),num2str(RayleighAngular_Y_cent),nu
m2str(RayleighAngular_dS),num2str(RayleighAngular_R),num2str(NumNodesRa
d),'0'};
answer = inputdlg(prompt,dlg_title,num_lines,def);

Theta = str2double(answer{1});
RayleighAngular_X_cent = str2double(answer{2});
RayleighAngular_Y_cent = str2double(answer{3});
RayleighAngular_dS = str2double(answer{4});
RayleighAngular_R = str2double(answer{5});
NumNodesRad = str2double(answer{6});
IdealPiston = str2double(answer{7});

%% Plot pressure cross-section at specific z location
w0 = 2*pi*Freq*1e6;

if IdealPiston
    if symx || symy
        MemNUM = TotalMemsSYM;
    else
        MemNUM = TotalMems;
```

```
    end

    vel_ARRAY = [];
    for mems = 1:MemNUM
        if ismember(mems, LiveMemsInd)
            vel_ARRAY = [vel_ARRAY;
ones(ReducedNodes,length(FreqList))];
        else
            vel_ARRAY = [vel_ARRAY;
zeros(ReducedNodes,length(FreqList))];
        end
    end

else
    vel_ARRAY = zAC(:,FreqIndex).*1i.*w0;
end



[p] = CalculateRayleigh_FDsym_Angular(vel_ARRAY, w0,Theta);

p = 20*log10(abs(p)) + Input_pulse_FFT_interp(FreqIndex) - 20*log10(1);
p = p - max(max(p));
% p = 10.^(p./20);

if IdealPiston
    figure(202)
else
figure(200);
end
angles = linspace(0,180,NumNodesRad);
pAngle = p;
mmpolar(angles*pi/180,pAngle);
mmpolar('RLimit',[-40 0])
mmpolar('TLimit',([0 180]*pi/180))
xlabel('Pressure Amplitude (dB)')
h = title(['Absolute Pressure at ', num2str(FreqList(FreqIndex)), '
MHz, ', num2str(Theta),'deg, and ',num2str(RayleighAngular_R/1e-6),' um
away from the array']);
set(h,'Position',[-0.00230415 1.2212 15])

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    dcm_obj = datacursormode(gcf);
    set(dcm_obj,'UpdateFcn',@PolarPlotFormat)
    % Click on line to select data point
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Sub-Function: PlotDynamicDisplacement_ARRAY_FD_SYM

```
function PlotDynamicDisplacement_ARRAY_FD_SYM()

global StepX
global StepY
global X_m
global Y_m
global NodesX
global NodesY
global FreqList
global zAC
global NumMemXi
global NumMemYi
global NumMemX
global NumMemY
global PitchX
global PitchY
global ReducedNodes
global symx
global symy
global MemSpaceX
global MemSpaceY

dcm_obj = datacursormode(3);
info_struct = getCursorInfo(dcm_obj);
FreqIndex = info_struct(1).DataIndex;

zACPLOT = zAC(:,FreqIndex);

X = 0:StepX:X_m;
Y = 0:StepY:Y_m;
[X Y] = meshgrid(X,Y);

figure(4)
cla
hold on
count2 = 1;
for j = 1:NumMemYi
    for i = 1:NumMemXi
        X2 = X+(i-1)*PitchX;
        Y2 = Y+(j-1)*PitchY;

        Z=zeros(NodesY,NodesX);
        count1 = 1;
        for y = 2:NodesY-1;
            Z(count1+1,2:(NodesX-1)) = real(zACPLOT(1+(NodesX-
2)*(count1-1)+ReducedNodes*(count2-1):(NodesX-
2)*count1+ReducedNodes*(count2-1)));
            count1 = count1 + 1;
        end

        Z2 = [Z];
        surface(X2/1e-6,Y2/1e-6,Z2/1e-6)
```

```
        count2 = count2+1;
    end
end

xlabel('X position (um)')
ylabel('Y position (um)')
zlabel('Displacement (um)')
title(['Real displacement of membrane at a frequency of ',
num2str(FreqList(FreqIndex)), ' MHz'])
colorbar
if X_m*NumMemXi < Y_m*NumMemYi
    xlim([0 PitchY*NumMemYi/1e-6])
    ylim([0 PitchY*NumMemYi/1e-6])
else
    xlim([0 PitchX*NumMemXi/1e-6])
    ylim([0 PitchX*NumMemXi/1e-6])
end

if symy
    Xsym = X_m*NumMemX/2 + MemSpaceX*(NumMemX-1)/2;
    xdata = ones(4,1)*Xsym/1e-6;
    ydata = [0; 0; PitchY*NumMemY; PitchY*NumMemY]/1e-6;
    zlimits = get(gca,'zlim');
    zdata = [1; -1; -1; 1]*max(abs(zlimits));
    p1 = patch(xdata,ydata,zdata,0);
    set(p1,'FaceColor','g','FaceAlpha',.2)

end

if symx
    Ysym = Y_m*NumMemY/2 + MemSpaceY*(NumMemY-1)/2;
    ydata = ones(4,1)*Ysym/1e-6;
    xdata = [0; 0; PitchX*NumMemX; PitchX*NumMemX]/1e-6;
    zlimits = get(gca,'zlim');
    zdata = [1; -1; -1; 1]*max(abs(zlimits));
    p1 = patch(xdata,ydata,zdata,0);
    set(p1,'FaceColor','g','FaceAlpha',.2)

end
%xlim([0 1e-6])
%ylim([0 X_m/1e-6])
%zlim([-Sac_isol/1e-6 0])
```

Sub-Function: PlotPressureAtZ_rayleigh_FD

```
function PlotPressureAtZ_rayleigh_FD()

global FreqList
global zAC
global RayleighPlane_X_cent
global RayleighPlane_Y_cent
global RayleighPlane_LengthX
global RayleighPlane_LengthY
global RayleighPlane_NodesX
global RayleighPlane_NodesY
global RayleighPlane_Zloc
global Input_pulse_FFT_interp

dcm_obj = datacursormode(12);
info_struct = getCursorInfo(dcm_obj);
FreqIndex = info_struct.DataIndex;
Freq = info_struct.Position(1);

prompt = {'X Center (m)', 'Y Center (m)','X Length (m)','Y Length
(m)','X Nodes:', 'Y Nodes:','Z Location (m)'};
dlg_title = 'Inputs for Radiation Plot function';
num_lines = 1;
def =
{num2str(RayleighPlane_X_cent),num2str(RayleighPlane_Y_cent),num2str(Ra
yleighPlane_LengthX),num2str(RayleighPlane_LengthY),num2str(RayleighPla
ne_NodesX),num2str(RayleighPlane_NodesY),num2str(RayleighPlane_Zloc)};
answer = inputdlg(prompt,dlg_title,num_lines,def);

RayleighPlane_X_cent = str2double(answer{1});
RayleighPlane_Y_cent = str2double(answer{2});
RayleighPlane_LengthX = str2double(answer{3});
RayleighPlane_LengthY = str2double(answer{4});
RayleighPlane_NodesX = str2double(answer{5});
RayleighPlane_NodesY = str2double(answer{6});
RayleighPlane_Zloc = str2double(answer{7});

% Plot pressure cross-section at specific z location
w0 = 2*pi*Freq*1e6;
vel_ARRAY = zAC(:,FreqIndex).*1i.*w0;
[p xlims ylims] = CalculateRayleigh_FDsym(vel_ARRAY, w0);

p = 20*log10(abs(p)) + Input_pulse_FFT_interp(FreqIndex) - 20*log10(1);
p = p - max(max(p));
% p = 10.^(p./20);


figure(13)
imagesc(xlims/1e-3,ylims/1e-3,p)
set(gca,'YDir','normal')
colormap(jet)
colorbar
xlabel('X position (mm)')
ylabel('Y position (mm)')
```

```
zlabel('Pressure (dB)')
title(['Absolute Pressure at ', num2str(FreqList(FreqIndex)), ' MHz', '
and ',num2str(RayleighPlane_Zloc/1e-6),' um away from the array'])
```

Sub-Function: progressbar

```
%this m-file modified by Michael Hochman on 8/20/11
%this m-file modified by Quan Quach on 12/12/07
%email: quan.quach@gmail.com
%Original Author: Steve Hoelzer
function [stopBar] =  progressbar(fractiondone, position, CurrentOp)

try
    if(~exist('fractiondone','var'))
        return
    end
    % Description:
    %   progressbar(fractiondone,position) provides an indication of
the progress of
    % some task using graphics and text. Calling progressbar repeatedly
will update
    % the figure and automatically estimate the amount of time
remaining.
    %   This implementation of progressbar is intended to be extremely
simple to use
    % while providing a high quality user experience.
    %
    % Features:
    %   - Can add progressbar to existing m-files with a single line of
code.
    %   - The figure closes automatically when the task is complete.
    %   - Only one progressbar can exist so old figures don't clutter
the desktop.
    %   - Remaining time estimate is accurate even if the figure gets
closed.
    %   - Minimal execution time. Won't slow down code.
    %   - Random color and position options. When a programmer gets
bored....
    %
    % Usage:
    %   fractiondone specifies what fraction (0.0 - 1.0) of the task is
complete.
    % Typically, the figure will be updated according to that value.
However, if
    % fractiondone == 0.0, a new figure is created (an existing figure
would be
    % closed first). If fractiondone == 1.0, the progressbar figure
will close.
    %   position determines where the progressbar figure appears on
screen. This
    % argument only has an effect when a progress bar is first created
or is reset
    % by calling with fractiondone = 0. The progress bar's position can
be specifed
    % as follows:
    %       [x, y]  - Position of lower left corner in normalized units
(0.0 - 1.0)
    %           0   - Centered (Default)
    %           1   - Upper right
    %           2   - Upper left
```

```
    %           3   - Lower left
    %           4   - Lower right
    %           5   - Random [x, y] position
    %   The color of the progressbar is choosen randomly when it is
created or
    % reset. Clicking inside the figure will cause a random color
change.
    %   For best results, call progressbar(0) (or just progressbar)
before starting
    % a task. This sets the proper starting time to calculate time
remaining.
    %
    % Example Function Calls:
    %   progressbar(fractiondone,position)
    %   progressbar               % Initialize/reset
    %   progressbar(0)            % Initialize/reset
    %   progressbar(0,4)          % Initialize/reset and specify
position
    %   progressbar(0,[0.2 0.7])  % Initialize/reset and specify
position
    %   progressbar(0.5)          % Update
    %   progressbar(1)            % Close
    %
    % Demo:
    %   n = 1000;
    %   progressbar % Create figure and set starting time
    %   for i = 1:n
    %       pause(0.01) % Do something important
    %       progressbar(i/n) % Update figure
    %   end
    %
    % Author: Steve Hoelzer
    %
    % Revisions:
    % 2002-Feb-27   Created function
    % 2002-Mar-19   Updated title text order
    % 2002-Apr-11   Use floor instead of round for percentdone
    % 2002-Jun-06   Updated for speed using patch (Thanks to waitbar.m)
    % 2002-Jun-19   Choose random patch color when a new figure is
created
    % 2002-Jun-24   Click on bar or axes to choose new random color
    % 2002-Jun-27   Calc time left, reset progress bar when
fractiondone == 0
    % 2002-Jun-28   Remove extraText var, add position var
    % 2002-Jul-18   fractiondone input is optional
    % 2002-Jul-19   Allow position to specify screen coordinates
    % 2002-Jul-22   Clear vars used in color change callback routine
    % 2002-Jul-29   Position input is always specified in pixels
    % 2002-Sep-09   Change order of title bar text
    % 2003-Jun-13   Change 'min' to 'm' because of built in function
'min'
    % 2003-Sep-08   Use callback for changing color instead of string
    % 2003-Sep-10   Use persistent vars for speed, modify titlebarstr
    % 2003-Sep-25   Correct titlebarstr for 0% case
    % 2003-Nov-25   Clear all persistent vars when percentdone = 100
    % 2004-Jan-22   Cleaner reset process, don't create figure if
percentdone = 100
```

```
    % 2004-Jan-27   Handle incorrect position input
    % 2004-Feb-16   Minimum time interval between updates
    % 2004-Apr-01   Cleaner process of enforcing minimum time interval
    % 2004-Oct-08   Seperate function for timeleftstr, expand to
include days
    % 2004-Oct-20   Efficient if-else structure for sec2timestr
    %
    stopBar = 0;
    persistent progfig progpatch starttime lastupdate firstIteration

    % Set defaults for variables not passed in
    if nargin < 1
        fractiondone = 0;
    end
    if nargin < 2
        position = 0;
    end

    if(~exist('CurrentOp','var'))
        CurrentOp = '';
    else
        CurrentOp = [' in ' CurrentOp];
    end

    try
        % Access progfig to see if it exists ('try' will fail if it
doesn't)
        dummy = get(progfig,'UserData');
        % If progress bar needs to be reset, close figure and set
handle to empty
        if fractiondone == 0
            delete(progfig) % Close progress bar
            progfig = []; % Set to empty so a new progress bar is
created
        end
    catch
        progfig = []; % Set to empty so a new progress bar is created
    end


    percentdone = floor(100*fractiondone);

    % Create new progress bar if needed

    if (isempty(progfig) && (isempty(firstIteration)))
        firstIteration = 1;
        % Calculate position of progress bar in normalized units
        scrsz = [0 0 1 1];
        width = scrsz(3)/4;
        height = scrsz(4)/50;
        if (length(position) == 1)
            hpad = scrsz(3)/64; % Padding from left or right edge of
screen
            vpad = scrsz(4)/24; % Padding from top or bottom edge of
screen
```

```
            left   = scrsz(3)/2 - width/2; % Default
            bottom = scrsz(4)/2 - height/2; % Default
            switch position
                case 0 % Center
                    % Do nothing (default)
                case 1 % Top-right
                    left   = scrsz(3) - width  - hpad;
                    bottom = scrsz(4) - height - vpad;
                case 2 % Top-left
                    left   = hpad;
                    bottom = scrsz(4) - height - vpad;
                case 3 % Bottom-left
                    left   = hpad;
                    bottom = vpad;
                case 4 % Bottom-right
                    left   = scrsz(3) - width  - hpad;
                    bottom = vpad;
                case 5 % Random
                    left   = rand * (scrsz(3)-width);
                    bottom = rand * (scrsz(4)-height);
                otherwise
                    warning('position must be (0-5). Reset to 0.')
            end
            position = [left bottom];
        elseif length(position) == 2
            % Error checking on position
            if (position(1) < 0) | (scrsz(3)-width < position(1))
                position(1) = max(min(position(1),scrsz(3)-width),0);
                warning('Horizontal position adjusted to fit on
screen.')
            end
            if (position(2) < 0) | (scrsz(4)-height < position(2))
                position(2) = max(min(position(2),scrsz(4)-height),0);
                warning('Vertical position adjusted to fit on screen.')
            end
        else
            error('position is not formatted correctly')
        end

        % Initialize progress bar
        progfig = figure(...
            'Units',            'normalized',...
            'Position',         [position width height],...
            'NumberTitle',      'off',...
            'Resize',           'off',...
            'MenuBar',          'none',...
            'BackingStore',     'off' );
        progaxes = axes(...
            'Position',         [0.02 0.15 0.96 0.70],...
            'XLim',             [0 1],...
            'YLim',             [0 1],...
            'Box',              'on',...
            'ytick',            [],...
            'xtick',            [] );
        progpatch = patch(...
            'XData',            [0 0 0 0],...
            'YData',            [0 0 1 1],...
```

```
            'EraseMode',        'none' );

        % enable this code if you want the bar to change colors when
the
        % user clicks on the progress bar
        %     set(progfig,  'ButtonDownFcn',{@changecolor,progpatch});
        %     set(progaxes, 'ButtonDownFcn',{@changecolor,progpatch});
        %     set(progpatch,'ButtonDownFcn',{@changecolor,progpatch});
        %     changecolor(0,0,progpatch)

        set(progpatch,'FaceColor',[.1 1 .1]);

        % Set time of last update to ensure a redraw
        lastupdate = clock - 1;

        % Task starting time reference
        if isempty(starttime) | (fractiondone == 0)
            starttime = clock;
        end

        set(progfig,'CloseRequestFcn',@closeBar);

    end

    %if the user closes the progress bar during the data processing
    %then this will erase all the variables are return 1 to the output
    if (isempty(progfig) && ~(fractiondone==0))
        delete(progfig) % Close progress bar

        % Clear persistent vars
        clear progfig progpatch starttime lastupdate firstIteration
        stopBar = 1;
        return

    end


    %Enforce a minimum time interval between updates
    %but allows for the case when the bar reaches 100% so that the user
can see
    %it
    if (etime(clock,lastupdate) < 0.01 && ~(percentdone == 100))
        return
    end

    % Update progress patch
    set(progpatch,'XData',[0 fractiondone fractiondone 0])

    % Update progress figure title bar
    if (fractiondone == 0)
        titlebarstr = ' 0%';
    else
        runtime = etime(clock,starttime);
        timeleft = runtime/fractiondone - runtime;
```

```
        timeleftstr = sec2timestr(timeleft);
        titlebarstr = sprintf('%2d%%    %s
remaining%s',percentdone,timeleftstr,CurrentOp);
    end
    set(progfig,'Name',titlebarstr)

    % Force redraw to show changes
    drawnow

    % If task completed, close figure and clear vars, then exit

    if percentdone == 100 % Task completed
        %     delete(progfig) % Close progress bar


        %change the close request function back to normal
        set(progfig,'CloseRequestFcn','closereq');
        % Clear persistent vars
        clear progfig progpatch starttime lastupdate firstIteration
        delete(gcf) % Close progress bar
        return
    end
    % Record time of this update
    lastupdate = clock;
catch
    progressbar(1)
    progressbar(1)
end
%%
% ------------------------------------------------------------------------
---------
function changecolor(h,e,progpatch)
Change the color of the progress bar patch

colorlim = 2.8; % Must be <= 3.0 - This keeps the color from being too
light
thiscolor = rand(1,3);
while sum(thiscolor) > colorlim
    thiscolor = rand(1,3);
end
set(progpatch,'FaceColor',thiscolor);



%%
% ------------------------------------------------------------------------
---------
function timestr = sec2timestr(sec)
% Convert a time measurement from seconds into a human readable string.

% Convert seconds to other units
d = floor(sec/86400); % Days
sec = sec - d*86400;
h = floor(sec/3600); % Hours
sec = sec - h*3600;
```

```
m = floor(sec/60); % Minutes
sec = sec - m*60;
s = floor(sec); % Seconds

% Create time string
if d > 0
    if d > 9
        timestr = sprintf('%d day',d);
    else
        timestr = sprintf('%d day, %d hr',d,h);
    end
elseif h > 0
    if h > 9
        timestr = sprintf('%d hr',h);
    else
        timestr = sprintf('%d hr, %d min',h,m);
    end
elseif m > 0
    if m > 9
        timestr = sprintf('%d min',m);
    else
        timestr = sprintf('%d min, %d sec',m,s);
    end
else
    timestr = sprintf('%d sec',s);
end

%%
function closeBar(src,evnt)

selection = questdlg('Do you want to stop this process?',...
    'Stop process',...
    'Yes','No','Yes');
switch selection,
    case 'Yes',
        delete(gcf)
    case 'No'
        return
end
```

REFERENCES

[1] C. Meynier, Franck Teston, and Dominique Certon, "A multiscale model for array of capacitive micromachined ultrasonic transducers," *J. Acoust. Soc. Am.*, vol. 128, pp. 2549-2561, 2010.

[2] K. D. Kochanek, M.A., Jiaquan Xu, M.D., Sherry L. Murphy, B.S., Arialdi M. Miniño, MPH, and Hsiang-Ching Kung, Ph.D, "Deaths: Preliminary Data for 2009," *National Vital Statistics Reports*, vol. 59, pp. 1-68, 2011.

[3] (May 2011). *Volcano Corporation Image Gallery*. Available: http://www.volcanocorp.com/products/gallery-eagle-eye.php

[4] J. G. Fujimoto, "Optical Coherence Tomography," in *Encyclopedia of optical engineering* vol. 2, R. G. Driggers, Ed., ed: CRC, 2003, pp. 1594-1595.

[5] (May 2011). *Boston Scientific Product Page*. Available: http://www.bostonscientific.com/Device.bsci/,,/method/DevHome/navRelId/1000.1003/seo.serve

[6] N. Bom, A.F.W. van der Steen, and C.T. Lancée, "The Technical Potential of IVUS: History and Principles," in *Vascular Ultrasound*, Y. Saijo, and A.F.W. van der Steen, Ed., ed Tokyo: Springer, 2003.

[7] (May 2011). *Stanford University Cardiovascular Core Analysis Laboratory Website*. Available: http://ccal.stanford.edu/japanese/case/case28/stills.html

[8] T. Inoue, Mayasha Ohta, and Sadayuki Takahashi, "Design of Ultrasonic Transducers with Multiple Acoustic Matching Layers for Medical Application," *IEEE Transactions on Ultrasonics, Ferroelectrics, and frequency control*, vol. UFFC-34, pp. 8-16, 1987.

[9] C. S. Desilets, John D. Fraser, and Gordon S. Kino, "The Design of Efficient Broad-Band Piezoelectric Transducers," *IEEE Transactions on Sonics and Ultrasonics*, vol. SU-25, pp. 115-125, 1978.

[10] T. R. Shrout, "Innovations in Piezoelectric Materials for Ultrasonic Transducers," in *IEEE International Symposium on the Applications of Ferroelectrics*, Santa Re, NM, USA, 2008, pp. 1-4.

[11] B. K. Courtney, Nigel R. Munce, Kevan J. Anderson, Amandeep S. Thind, General Leung, Perry E. Radau, F. Stuart Foster, I. Alex Vitkin, Robert S. Schwartz, Alexander J. Dick, Graham A. Wright, and Bradley H. Strauss, "Innovations in imaging for chronic total occlusions: a glimpse into the future of angiography's blind-spot," *European Heart Journal*, vol. 29, pp. 583-593, 2008.

[12] Y. Wang, Douglas N. Stephens, and Matthew O'Donnel, "Initial Results from a forward-viewing ring-annular ultrasound array for intravascular imaging," in *IEEE Ultrasonics Symposium on Ultrasonics*, 2003, pp. 212-215.

[13] E. D. Light, Victor Lieu, and Stephen W. Smith, "New Fabrication Techniques for Ring-Array Transducers for Real-Time 3D Intravascular Ultrasound," *Ultrasonic Imaging*, vol. 31, pp. 247-256, 2009.

[14] J. H. Rogers. (June/July 2009) Forward-Looking IVUS in Chronic Total Occlusions: A new approach to an old problem. *Cardiac Interventions Today*.

[15] D. M. Mills, and L. Scott Smith, "Real-Tme In-Vivo Imaging With Capacitive Micromachined Ultrasound Transducer (CMUT) Linear Arrays," presented at the IEEE Ultrasonics Symposium, 2003.

[16] B. Bayram, Ömer Oralkan, A. Sanli Ergun, Edward Hæggström, Goksen G. Yaralioglu, and Butrus T. Khuri-Yakub, "Capacitive Micromachined Ultrasonic Transducer Design For High Power Transmission," *IEEE Transactions on Ultrasonics, Ferroelectrics, and frequency control*, vol. 52, pp. 326-339, 2005.

[17] R. O. Guldiken, "Dual-Electrode Capacitive Micromachined Ultrasonic Transducers for Medical Ultrasound Applications," PhD Dissertation, Mechanical Engineering, Georgia Institute of Technology, 2008.

[18] D. T. Yeh, Ömer Oralkan, Ira O. Wygant, Matthew O'Donnell, and Butrus T. Khuri-Yakub, "3-D Ultrasound Imaging Using a Forward-Looking CMUT Ring Array for Intravascular/Intracardiac Applications," *IEEE Transactions on Ultrasonics, Ferroelectrics, and frequency control*, vol. 53, pp. 1202-1211, 2006.

[19] S. D. Senturia, *Microsystem Design*. Boston: Kluwer Academic Publishers Group, 2001.

[20] J. Fraser, and P. Reynolds, "Finite-element method for determination of electromechanical coupling coefficient for piezoelectric and capacitive micromachined ultrasonic transducers," in *Joint 140th meeting of ASA/NOISE-CON*, 2000.

[21] G. G. Yaralioglu, Arif Sanli Ergun, Baris Bayram, Edward Hæggström, and Butrus T. Khuri-Yakub, "Calculation and Measurement of Electromechanical Coupling Coefficient of Capacitive Micromachined Ultrasonic Transducers," *IEEE Transactions on Ultrasonics, Ferroelectrics, and frequency control*, vol. 50, pp. 449-456, 2003.

[22] W. P. Mason, *Electromechanical Transducers and Wave Filters*. New York, NY: Van Nostrand, 1948.

[23] A. S. Ergun, Goksen G. Yaralioglu, and Butrus T. Khuri-Yakub, "Capacitive Micromachined Ultrasonic Transducers: Theory and Technology," *Journal of Aerospace Engineering*, pp. 76-84, 2003.

[24] L. E. Kinsler, Austin R. Frey, Alan B. Coppens, and James V. Sanders, *Fundamentals of Acoustics*, Fourth ed.: John Wiley & Sons, Inc, 2000.

[25] G. G. Yaralioglu, Mohammed H. Badi, A. Sanli Ergun, and Butrus T. Khuri-Yakub, "Improved Equivalent Circuit and Finite Element Method Modeling of Capacitive Micromachined Ultrasonic Transducers," presented at the IEEE Ultrasonics Symposium, 2003.

[26] M. Wilm, Alexandre Reinhardt, Vincent Laude, Raphaël Armati, William Daniau, and Sylvain Ballandras, "Three-dimensional modelling of micromachined-ultrasonic-transducer arrays operating in water," *Ultrasonics*, vol. 43, pp. 457-465, 2004.

[27] J. W. Hunt, Marcel Arditi, and F. Stuart Foster, "Ultrasound Transducers for Pulse-Echo Medical Imaging," *IEEE Transactions on Biomedical Engineering*, vol. BME-30, pp. 453-481, 1983.

[28] T. L. Szabo, *Diagnostic Ultrasound Imaging*. Hartford, Connecticut: Elsevier Academic Press, 2004.

[29] D. T. Yeh, Ömer Oralkan, Arif S. Ergun, Xuefeng Zhuang, Ira O. Wygant, and Butrus T. Khuri-Yakub, "High-frequency CMUT arrays for high-resolution medical imaging," in *Medical Imaging: Ultrasonic Imaging and Signal Processing*, 2005.

[30] G. Caliano, R. Carotenuto, A. Caronti, and M. Pappalardo, "CMUT Echographic Probes: Design and Fabrication Process," presented at the IEEE Ultrasonics Symposium, 2002.

[31] A. Lohfink, and Peter-Christian Eccardt, "Linear and Nonlinear Equivalent Circuit Modeling of CMUTs," *IEEE Transactions on Ultrasonics, Ferroelectrics, and frequency control*, vol. 52, pp. 2163-2172, 2005.

[32] D. Certon, Franck Teston, and Frédéric Patat, "A Finite Difference Model For cMUT Devices," *IEEE Transactions on Ultrasonics, Ferroelectrics, and frequency control*, vol. 52, pp. 2199-2210, 2005.

[33] S. P. Timoshenko, *Theory of Plates and Shells*. Birkshire, England: Open Press University, 1964.

[34] P. C. Eccardt, P. Wagner, and S. Hansen, "Analytical Models for Micromachined Transducers - An Overview," presented at the IEEE Ultrasonics Symposium, 2006.

[35] S. C. Chapra, and Raymond P Canale, *Numerical Methods for Engineers: With Software and Programming Applications*, 4th ed.: McGraw-Hill Higher Education, 2002.

[36] A. D. Pierce, *Acoustics*. Woodbury, New York: The Acoustical Society of America, 1994.

[37] C. H. Cheng, E. M. Chow, X. Jin, S. Ergun, B.T. Khuri-Yakub, "An Efficient Electrical Addressing Method Using Through-Wafer Vias For Two-Dimensional Ultrasonic Arrays," presented at the IEEE Ultrasonic Symposium, 2000.

[38] X. Cheng, D.F. Lemmerhirt, O.D. Kripfgans, M. Zhang, C. Yang, C.A. Rich, J.B. Fowlkes, "CMUT-in-CMOS Ultrasonic Transducer Arrays With On-Chip Electronics," in *Transducers: Solid-State Sensors, Actuators and Microsystems*, 2009, pp. 1222-1225.

[39] P. C. Eccardt, K. Niederer, T. Scheiter, C. Hierold, "Surface Micromachined Ultrasound Tranducers in CMOS Technology," in *IEEE Ultrasonic Symposium*, 2000, pp. 959-962.

[40] I. O. Wygant, X. Zhuang, D.T. Yeh, O. Oralkan, A.S. Ergun, M. Karaman, B.T. Khuri-Yakub, "Integration of 2D CMUT Arrays With Front-End Electronics for Volumetric Ultrasound Imaging," *IEEE Transactions on Ultrasonics, Ferroelectrics, and frequency control*, vol. 55, 2008.

[41] R. A. Noble, R.R. Davies, M.M. Day, L. Koker, D.O. King, K.M. Brunson, A.R.D. Jones, J.S. McIntosh, D.A. Hutchins, T.J. Robertson, P. Saul, "A Cost-effective and Manufacturable Route to the Fabrication of High-Density 2D Micromachined Ultrasonic Transducer Arrays and (CMOS) Signal Conditioning Electronics on the Same Silicon Substrate," presented at the IEEE Ultrasonics Symposium, 2001.

[42] J. Zahorian, Rasim Guldiken, Gokce Gurun, Muhammad Shakeel Qureshi, Mujdat Balantekin, Paul Hasler, and F. Levent Degertekin, "Single chip CMUT arrays with integrated CMOS electronics: Fabrication Process Development and Experimental Results," in *IEEE International Ultrasonics Symposium*, 2008, pp. 386-389.

[43] J. Knight, J. McLean, and F. L. Degertekin, "Low Temperature Fabrication of Immersion Capacitive Micromachined Ultrasonic Transducers on Silicon and Dielectric Substrates," *IEEE Transactions on Ultrasonics, Ferroelectrics, and frequency control*, vol. 51, pp. 1324-1333, 2004.

[44] G. Gurun, Jaime Zahorian, Paul Hasler, and Levent Degertekin, "Thermal Mechanical Noise Based Characterization of CMUTs Using Monolithically Integrated Low Noise Receiver Electronics," presented at the IEEE Ultrasonics Symposium, San Diego, 2010.

[45] H. Lee, Jinhee Tak, and Wonkyu Moon, "Effects of mutual impedance on the radiation characteristics of transducer arrays," *J. Acoust. Soc. Am.*, vol. 115, pp. 666-678, 2004.

[46] O. Oralkan, Baris Bayram, Goksen G. Yaralioglu, A. Sanli Ergun, Mario Kupnik, David T. Yeh, Ira O. Wygant, and Butrus T. Khuri-Yakub, "Experimental Characterization of Collapse-Mode CMUT Operation," *IEEE Transactions on Ultrasonics, Ferroelectrics, and frequency control*, vol. 53, pp. 1513-1523, 2006.

[47] S. Olcum, F. Yalcin Yamaner, Ayhan Bozkurt, Hayrettin Köymen, and Abdullah Atalar, "An Equivalent Circuit Model for Transmitting Capacitive Micromachined Ultrasonic Transducers in Collapse Mode," *IEEE Transactions on Ultrasonics, Ferroelectrics, and frequency control,* vol. 58, pp. 1468-1477, 2011.