Exercise 5: Array Design

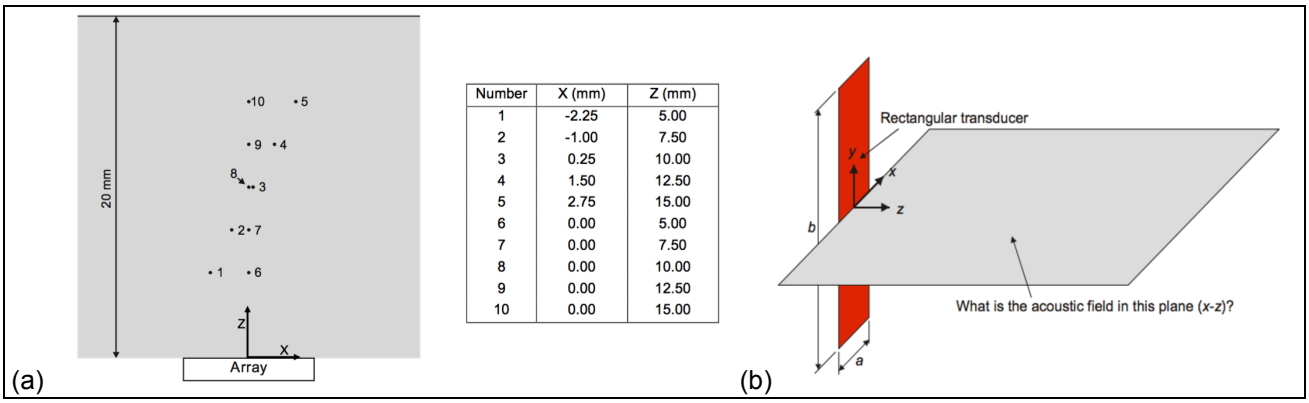
Aidan Scannell

ABSTRACT

A linear transducer array was developed and optimised with the ability to resolve ten target points. The time-domain signals were simulated and used in three post-processing imaging algorithms, TFM, focused B scan and plane B scan. A transducer with frequency = 12.5 MHz, element width = 0.05 mm, inter-element spacing = 0.05 mm and 64 elements was developed that was capable of resolving all of the target points using the TFM imaging algorithm. The focused B scan resolved all of the points except 3 & 8 and the plane B scan couldn't resolve any of the points laterally.

INTRODUCTION

This report will discuss the design of an ultrasonic phased array for operation into the human body. The testing medium consisted of 10 points requiring inspection (shown in figure 1) and a back wall 20mm away in the z direction. Huygen's principle was used to generate a calibration array that enabled appropriate parameters to be selected and used for the array simulation. The time domain signals were then simulated and used in three post-processing algorithms.



Number	X (mm)	Z (mm)
1	-2.25	5.00
2	-1.00	7.50
3	0.25	10.00
4	1.50	12.50
5	2.75	15.00
6	0.00	5.00
7	0.00	7.50
8	0.00	10.00
9	0.00	12.50
10	0.00	15.00

Fig. 1 (a) Diagram and table showing the positions of the target points (b) Transducer of size a x b where b >> a [1]

LIMITATIONS AND ASSUMPTIONS

- It was assumed that the human body had the same properties as water.
- The maximum number of elements was 64.
- The minimum gap between elements was 0.05 mm due to the manufacturing process.
- It was assumed that the transducer elements were infinitely long.
- It was assumed that the points being inspected were infinitely long compared to the x-z plane.
- The velocity of sound in water was assumed to be 1500 ms^{-1}
- Assumed water was isotropic

EXERCISE 1: 2D MATLAB SIMULATION PREDICATING BEAM PROFILE OF LINEAR ARRAY

A 2D Matlab simulation was produced to predict the beam profile from a linear array operating into water.

Huygen's principle was used reconstruct the pressure field from the transducer, which was modelled as a discrete number of point sources where each source's field could be summed. The principle of super position of waves can be applied to waves travelling through the same medium at the same time. As they pass through each other without being disturbed the net displacement at any point in space or time is equal to the sum of all the separate waves. This leads to points of constructive and destructive interference when the waves are in-phase and opposite-phase respectively. Superposition works because wave equation is linear.

Figure 1 (b) shows a transducer element with width a and length b where b was assumed to be much greater than a. Dimension y was assumed to be infinite for the transducer and the point targets, which enabled analyses in 2D using Huygen's principle. A two dimensional grid was created to model the x and z dimensions shown on Figure 1 (b).

In order to calculate the field produced by the transducer array, the distance from each point source (j) to each grid position was calculated using Equation 1,

$$r_j = \sqrt{(x - x_j)^2 + z^2} \tag{1}$$

where x_j is the x coordinate of source j and r_j is the distance from point source j to grid position (x,z). The field from each point source was then calculated using Equation 2, the wave equation in cylindrical form,

$$P_j(x,z) = A\,\frac{1}{\sqrt{r_j}}e^{i(kr_j-\omega t)} \tag{2}$$

where P_j is the field from point source j, A is a complex number representing the size and phase of the source, r_j is distance from source j to position (x,z) and the other symbols have their usual meaning. Equation 2 assumes a line source and that the material is isotropic, which is reasonable for our application as physical transducers produce line sources and the testing medium is water. The $\frac{1}{\sqrt{r}}$ term represents the wave getting smaller as it moves further away from the source, which holds true with conservation of energy. Every transducer has its own directivity pattern, which was included in the simulation using the directivity function D_f, which represents how the pressure varies with angle and is given by Equation 3,

$$D_f(\theta) = \frac{\sin(0.5\,k\,a\,\sin(\theta))}{0.5\,k\,a\,\sin(\theta)} \tag{3}$$

where a is the element width and θ is the angle between r and the z axis. The overall pressure field from the transducer array was then calculated using Equation 3, utilising the principle of superposition of waves,

$$P(x,z) = \sum_{j=1}^{n} D_{fj}\,\frac{A}{\sqrt{r_j}}e^{i(kr_j-\omega t)} \tag{4}$$

where all the symbols have their usual meaning. Figure 2 shows the simulated beam profile from the linear array that was produced in Matlab.

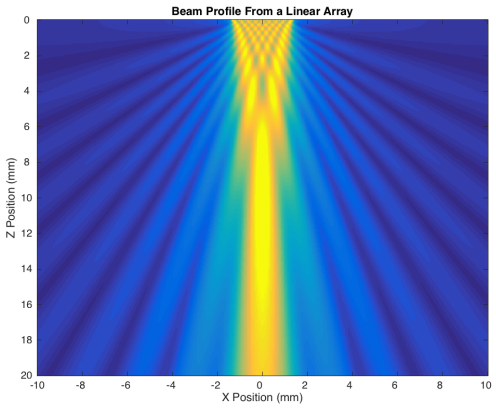


Fig. 2 Simulated beam profile from the linear array

The numerical solution is accurate within the near field but the analytical solution is surprisingly not. This is due to the fact that solving the integral required an approximation, which does not hold true for the near field.

EXERCISE 2: DESIGN AND OPTIMISATION OF LINEAR ARRAY

The linear array that was designed in Exercise 1 was then optimised for the target points shown in Figure 1. It is desirable for the transducer array to focus at the point of interest and converge the acoustic energy at the desired depth. This exercise acted as a calibration enabling an appropriate array to be designed that was capable of detecting the required features for Exercises 3 and 4. In order to inspect all of the target points the transducer array was required to focus the beam, which was achieved by steering and focussing the beam. Delays represent the time delay between each transducer being fired and were used to steer the beam a required angle. B_j represents the weight and phase delay of element j that was calculated using Equation 5,

$$B_j = e^{i\omega(d_j - d_0)/c} \tag{5}$$

where d_j is the distance from element j to a specific focal point, d_0 is the distance from the central element to focal point and c is the velocity of sound in water. The field from a transducer focusing at a point was calculated using Equation 6,

$$P(x,z) = B_j D_f(\emptyset_j)\,\frac{1}{\sqrt{r_j}}e^{i(kr_j-\omega t)} \tag{6}$$

where all symbols have their usual meanings. The transducers effective sensitivity is dependant on the beam width at the point of interest; the smaller the beam width the more energy that can be reflected by a small flaw and thus the more easily they can be detected. The transducer can only focus within the near field, which is defined by Equation 7,

$$NF = \frac{D^2}{4\lambda} \tag{7}$$

where D is the transducer width, λ is the wavelength and NF is the near field distance. There are three types of lobes, main lobes in the direction of the beam, side lobes, which appear either side of the beam direction (representing leaked energy) and grating lobes. These need to be considered when optimising the transducer. The goal of optimisation is to obtain good beam directivity and steerability, which was obtained by reducing the main lobe width, suppressing side lobe amplitudes and eliminating grating lobes. This was achieved by selecting an appropriate number of elements, element spacing, element size and wavelength.

Figure 3 is a plot of directivity against angle for a slice through the grid half way along the z axis for different $\frac{a}{\lambda}$ ratios. It can be seen that for low $\frac{a}{\lambda}$ ratios the directivity was fairly constant, so the transducer can therefore be assumed to be a set of discrete point sources. The $\frac{a}{\lambda}$ ratios used for this exercise were small enough to assume each element was a discrete point source.

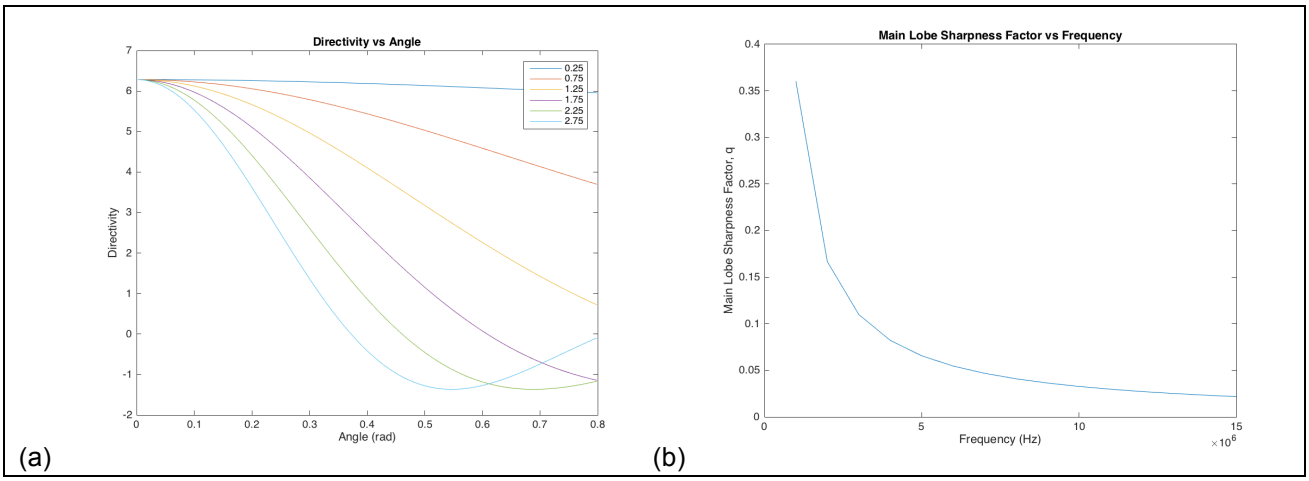


Fig. 3 (a) Directivity for various $\frac{a}{\lambda}$ ratios (b) Relationship between q and the frequency

The transducers steering performance was characterised by the main lobe sharpness factor, q, and the amount of side leaking energy [2]. The largest angle that the beam was required to steer was 0.423 rad when focusing at target 1. At any given angle the beam width could be quantified by the main lobe sharpness factor given by Equation 8, which defines the width of the lobe in the steering direction,

$$q = \frac{1}{\pi}\left[sin^{-1}\left(\sin(\theta_s) + \frac{\lambda}{Nd}\right) - sin^{-1}(\sin(\theta_s) - \frac{\lambda}{Nd}\right) \tag{8}$$

where N is the number of elements, d is the inter-element spacing, θ_s is the steering angle and the other symbols have their usual meaning. In order to improve the performance of the array the main lobe sharpness needed to be reduced. A small main lobe sharpness factor meant that the beam had good directivity as it was narrow and sharply defined [3]. Figure 3 (b) shows the relationship between frequency and the main lobe sharpness factor for the transducer set up with N = 64, θ_s = 0.423 and d = 0.10 mm and it can be seen that after 7.5 MHz the reduction in q for increased frequencies is minimal. The frequency of the transducer was set to this value as it provided a good main lobe sharpness factor, provided a long enough near field, provide acceptable axial resolution and because it reduced grating lobes, which will be discussed later.

Figure 4 (a) shows that as the steering angle is increased the main lobe sharpness factor increases meaning reduced array performance. It can be seen that for a 0.423 rad steering angle the main lobe sharpness remains low for the transducer set up with N = 64, f = 7.5 MHz and d = 0.10 mm.

It can be seen from Equation 8 that increasing the number of elements is desirable for reducing the main lobe sharpness factor but it results in a larger array. For a given element width and steering angle the main lobe sharpness factor can be determined as a function of the number of elements. This is shown in Figure 4 (b) with a transducer set up of θ_s = 0.423 rad, f = 7.5 MHz and d = 0.10 mm. It can be seen that increasing the number of elements increases the arrays performance but the transducer was limited to having 64 elements with a spacing of 0.05 mm. Figure 4 (b) can be used to select a reasonable number of elements that provides a low enough main lobe sharpness factor while still being physically feasible. 64 elements were selected for this application.

The main lobe sharpness factor can also be reduced by increasing the $\frac{d}{\lambda}$ ratio, which is achieved by either increasing the frequency of the transducer elements or by increasing the inter-element spacing, as seen in figure 5 (a). Increasing the inter-element spacing is more beneficial than increasing the element width so the element width was set to 0.05 mm (the minimum physically possible) so that the inter-element width could be

increased to the limit of grating lobes. This element width also resulted in an $\frac{a}{\lambda}$ ratio of 0.25, which meant that the elements could be assumed to be discrete point sources from Figure 3 (a). The inter-element spacing was then increased to 0.10 mm where grating lobes started to appear. Increasing the frequency has a limit due to the capabilities of the piezo-elements and leads to grating lobes as well. Grating lobes can act like a second beam that can focus on points away from the target location and introduce large amounts of noise while also leaking energy. Any scattering points that the grating lobes focus on will result in the elements receiving undesired signals.

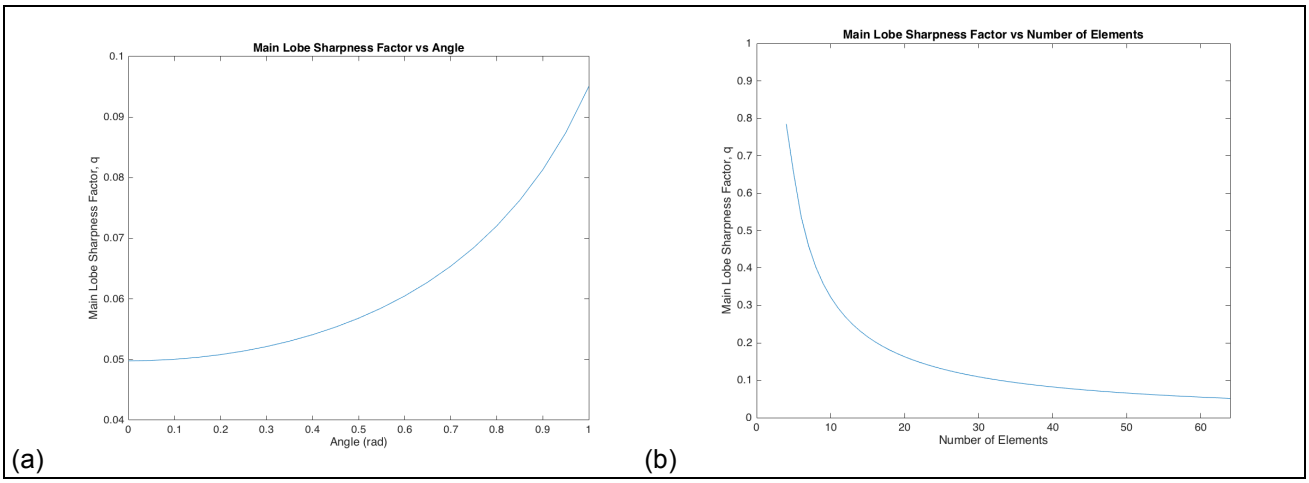


Fig. 4 (a) Relationship between q and the steering angle (b) Relationship between q and the number of elements

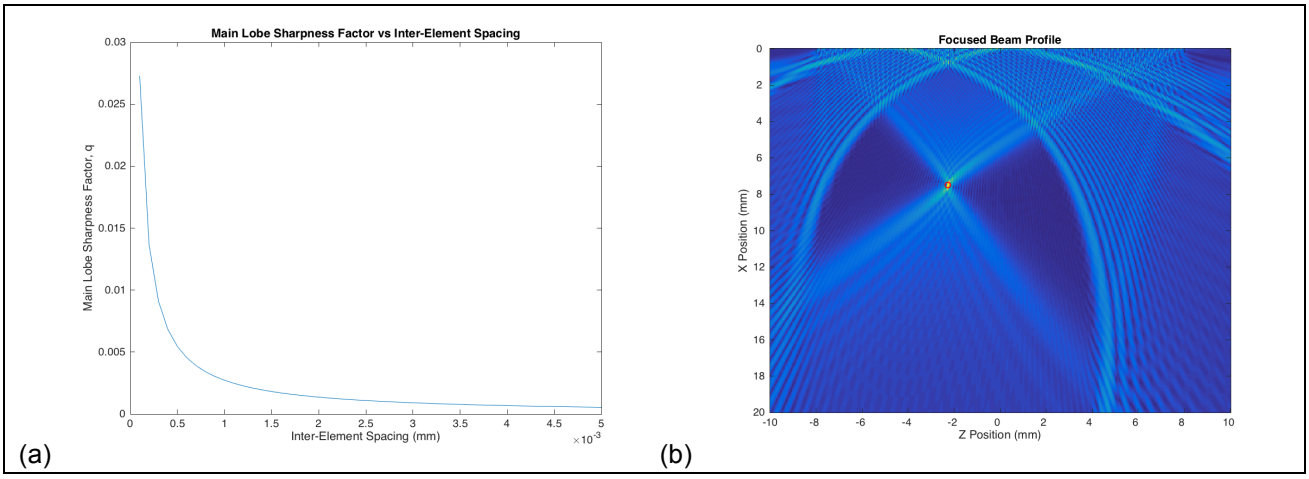


Fig. 5 (a) Relationship between q and the inter-element spacing (b)

There is therefore a trade off between grating lobes and optimising the beam. An example of severe grating lobes is shown in Figure 5 (b), where they interact with points within the target locations. For this particular application, where the target locations are known, it is possible to obtain an optimised transducer setup where the grating lobes do not interact with the target points. The optimised array with reduced grating lobes and main lobe sharpness factor is shown in Figure 5 (a) and the set up is displayed in Fable 1. The largest pressure from the grating lobes was found and compared to that of the beam. If the amplitude is small compared to the beam then any signals received from the grating lobes will be negligible. The maximum amplitude can be seen in table 1 and was one fifth of the focused beam's amplitude, so can therefore be considered negligible.

The axial resolution is defined as the ability to detect points close together along the transducer beam length. The spatial pulse length is equal to the product of the wavelength and the number of cycles in the pulse. Lower spatial pulse lengths provide better axial resolution as the axial resolution is equal to half of the spatial pulse length. The axial resolution cannot be better than half of the pulse length because of the overlap of returning echoes reflected off interface space close together [6]. For the 5-cycle toneburst that was used and the 7.5MHz frequency the axial resolution was 0.5mm, which was deemed acceptable.

The near field was large enough as the transducer was not required to focus further than 20 mm away and the main lobe sharpness factor was reduced significantly compared to other set ups. Figures 6 (a) and 6 (b) show the optimised transducer focusing on target point 10 (the furthest target point) and target point 1 (the point with the largest steering angle) respectively.

Tab. 1 Optimised transducer setup (focusing at point 1)

Frequency (MHz)	Number of Elements, N	Element Width (mm)	Inter-Element Spacing (mm)	Main Lobe Sharpness Factor, q	Near Field Distance, NF (mm)	$\frac{a}{\lambda}$	Grating Lobe Amplitude/Focused Beam Amplitude
7.5	64	0.05	0.10	0.0218	112	0.25	0.21

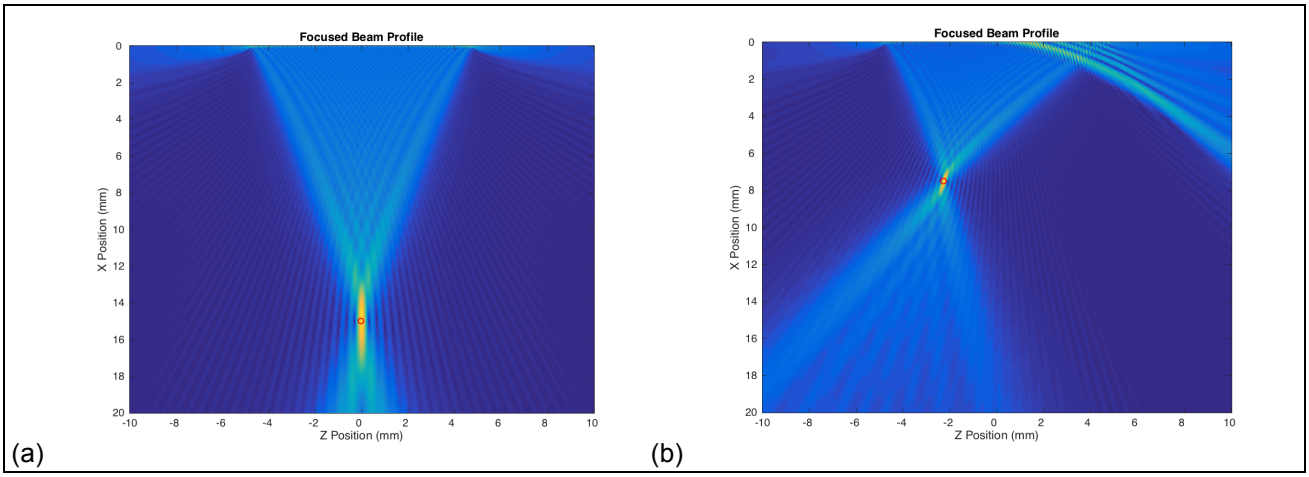


Fig. 6 (a) Focused beam on target 10 (b) Focused beam on target 1

EXERCISE 3: SIMULATE TIME-DOMAIN SIGNALS

A Matlab simulation was produced to simulate the complete set of time-domain signals received from all transmitter-receiver pairs when the array was used to inspect the sample shown in figure 1. The transducer elements received a 5-cycle toneburst input signal with a centre frequency defined by the transducer setup. Equation 8 represents the frequency domain signal transmitted from element i and received by element j for the n target points,

$$F_{ij}(\omega) = \sum_1^n F_0(\omega)\frac{1}{\sqrt{R_t R_r}} D_f(\varphi_t,\omega) D_f(\varphi_r,\omega) S(\varphi_t,\varphi_r,\omega) e^{-i\omega\tau} \tag{8}$$

where $F_{ij}(\omega)$ is the frequency response for signal from transmitter i and receiver j pair, $F_0(\omega)$ is the frequency domain 5 cycle toneburst input signal determined by taking the Fourier transform of the time domain signal, $D_f(\varphi_t,\omega)$ is the directivity function of the transmitter, $D_f(\varphi_r,w)$ is the directivity function of the receiver, $S(\varphi_t,\varphi_r,\omega)$ is the scattering amplitude of the target points equal to 0.01, R_t is the distance from transmitter i to the focal point, R_r is the distance from receiver j to the focal point and τ is the time delay for transmitter – receiver pair ij. Each transmitting elements signal was delayed in time in order to simulate the beam steering and focusing, which was represented in the frequency domain by the $e^{-i\omega\tau}$ term. The time delay was determined by calculating the distance from transmitter i to the target point plus the distance from the focal point to receiver j as can bee seen in Equation 9,

$$\tau_{ij} = \frac{\left(\sqrt{((x_{ref}-x_i)^2+z_{ref}{}^2)}+\sqrt{((x_{ref}-x_j)^2+z_{ref}{}^2)}\right)}{c} \tag{9}$$

where all the symbols have their usual meaning. The directivities were calculated using equation 3 in Exercise 1. The $\frac{1}{\sqrt{R_t R_r}}$ term represents the beam spread of the signal as it propagates through the water, which is used to simulate the signal's amplitude decreasing as the beam spreads out, for conservation of energy. The attenuation was assumed negligible, which is acceptable as the medium was water and the signals were not travelling very far. Equation 10 represents the frequency domain signal transmitted from element i and received by element j for the back wall that was treated as a perfect reflector, $R_c(\varphi)$ = 1,

$$F_{ij}(\omega) = F_0(\omega)\frac{1}{\sqrt{R_t+R_r}} D_f(\varphi_t,\omega) D_f(\varphi_r,\omega) R_c(\varphi) e^{-i\omega\tau} \tag{10}$$

where the symbols have the same meaning as before. Notice that the beam spread term is now represented as $\frac{1}{\sqrt{R_t+R_r}}$. These reflected signals were recorded for each transmitter-receiver pair for each focal point using 3-dimensional matrices, which were then superimposed for each target point. The back wall signals were then added as well. 3D matrices were utilised to enable robustness as it enabled larger grid sizes and more frequency points to be used without encountering memory problems, although it lead to longer computation

times for smaller grid sizes. The superimposed frequency domain signals were then converted to the time domain using the inverse Fourier transform.

Figure 7 (a) shows the time trace that was obtained for transmitter 1 and receiver 1. Ten signals can be seen; the last signal is the largest and represents the reflection from the back wall, its large amplitude is due to the reflection coefficient of 1 that lead to all of the energy being reflected back to the receiver. The other 9 signals correspond to each of the focal points except where two focal points (target points 3 & 8) were combined at time ~0.14 μs. The amplitude of the signals decrease as the time increases, which holds true with Equation 8 where the $\frac{1}{\sqrt{R_t R_r}}$ term represents the beam spread.

Symmetry meant that the signals only needed to be calculated for each transmitter-receiver pair once as the signal from transmitter i to receiver j will be identical as the signal from transmitter j to receiver i, which reduced the number of signals n, that needed to be recorded from n^2 to $\frac{n\,(n+1)}{2}$.

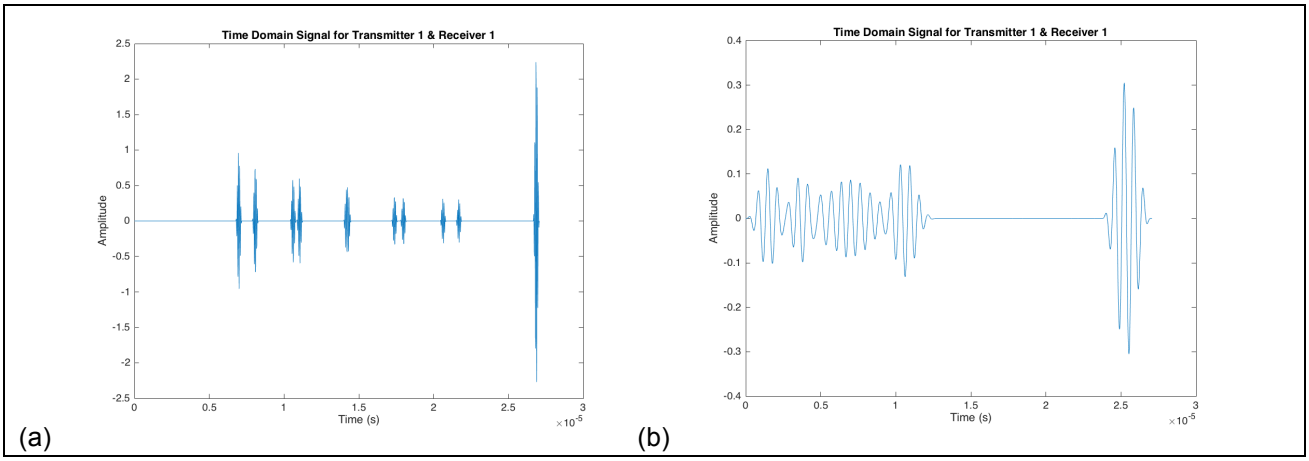


Fig. 7 (a) Time domain signal from transmitter 1, receiver 1 (b) Low f_s time domain signal from transmitter 1, receiver 1

The frequency domain was used because the directivity is dependant on frequency and therefore made analysis in the frequency domain easier. The sampling frequency was required to be at least twice the largest frequency within the signal in order to prevent aliasing, seen in Figure 7 (b) but was made approximately ten times larger in order to reconstruct the signal acceptably. In real life these signals could be obtained using various transducer set ups, which depend upon the post-processing imaging algorithm selected.

EXERCISE 4

The simulated time-domain signals from exercise 3 were processed using three imaging algorithms, the total focusing method, plane B scan and focused B scan. The optimised array from Exercise 2 was capable of detecting target points 3 and 8, demonstrating how well optimised it is. However, it was found that an optimised array with a higher frequency was more capable of separating the points on the image, indicating better resolution at higher frequencies. The array was re-optimised following the approach in exercise 2 with a transducer setup shown in Table 3. The post-processing algorithms were then implemented.

Tab. 2 Optimised transducer setup (focusing at point 1)

Frequency (MHz)	Number of Elements, N	Element Width (mm)	Inter-Element Spacing (mm)	Main Lobe Sharpness Factor, q	Near Field Distance, NF (mm)	$\frac{a}{\lambda}$	Grating Lobe Amplitude/Focused Beam Amplitude
12.5	64	0.05	0.05	0.0262	82	0.42	0.20

Total Focusing Method

The total focusing method required the beam to be focused at every grid point and thus was computationally expensive. Increasing the number of points within the grid improved the resolution but increased computational time. TFM requires a high number of transmit delay cycles and the computational time for image processing is high. If all of the matrix data is captured and then post-processed then computation time is the only limiting factor. Equation 11 was used to calculate the required transmitter delays for each grid point $\boldsymbol{r}$,

$$\tau_{ij}(\boldsymbol{r}) = \frac{|\boldsymbol{r}-\boldsymbol{r_i}|}{c} + \frac{|\boldsymbol{r_j}-\boldsymbol{r}|}{c} \quad (11)$$

where $\boldsymbol{r_i}$ is the distance from transmitter element i to grid point $\boldsymbol{r}$, $\boldsymbol{r_j}$ is the distance from receiver element j to grid point $\boldsymbol{r}$, $\boldsymbol{r}$ is the distance from central element the to grid point and τ_{ij} is the delay for transmitter-receiver pair ij. Symmetry meant that the delays only needed to be calculated for each element to each grid point and not the full set of transmitter-receiver pairs, which reduced the computation time. The delays were then used to extract the corresponding amplitude from the time-domain signals at time $\tau_{ij}(\boldsymbol{r})$ for all of the transmitter-receiver pairs at a specific grid point. These values were then summed for each grid point to determine the intensity as shown by Equation 12,

$$I(\boldsymbol{r}) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}(\boldsymbol{r}) f_{ij}(\tau_{ij}(\boldsymbol{r})) \quad (12)$$

where $a_{ij}(\boldsymbol{r})$ is assumed to be 1, $f_{ij}(\tau_{ij}(\boldsymbol{r}))$ is the amplitude of the time-domain signal at time $\tau_{ij}(\boldsymbol{r})$ for transmitter i and receiver j and $I(\boldsymbol{r})$ is intensity at grid point $\boldsymbol{r}$. The resulting intensity was then interpolated to smooth the signal by increasing the grid size from 200 x 200 to 600 x 600. The resulting image can be seen in figure 8 (a) and figure 8 (b) shows target points 8 and 3 clearly resolved.

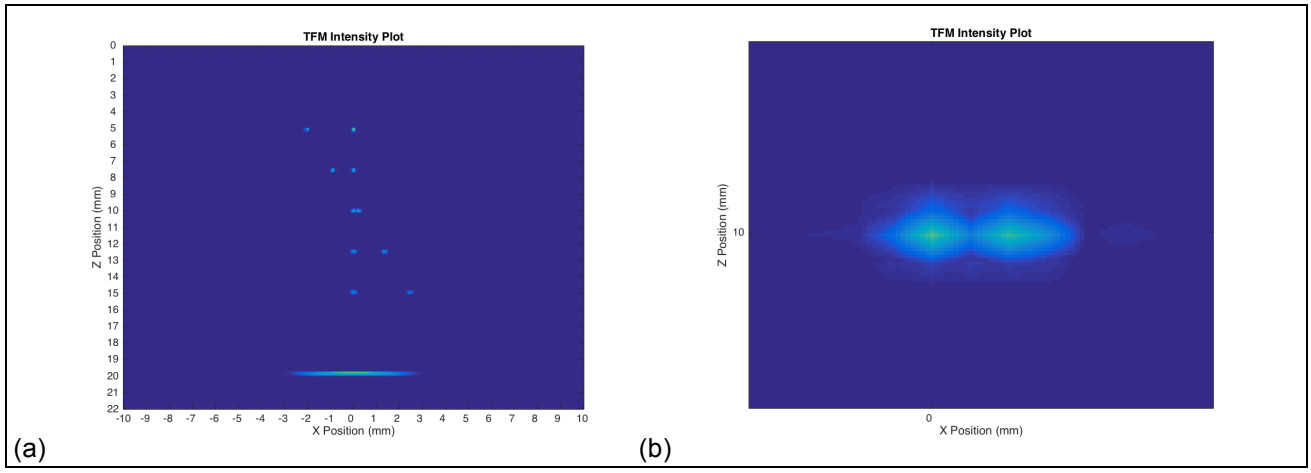


Fig. 8 (a) TFM intensity plot (B) TFM target points 3 and 8

Focused B Scan

The focused B scan image algorithm was then implemented, representing an aperture from the array focusing a beam at every grid point within the sample for multiple apertures. The data can be collected using a curved element or lens, or by utilising a transducer array with delays, to focus a set number of elements (aperture) on each grid point. The aperture is then stepped along each position on the array to obtain data for all of the grid points. This theoretically resulted in a lateral resolution of 0.1 mm (the element pitch) for the transducer setup in table 2 and a resolution of 0.15 mm for the transducer setup in table 1 from Exercise 2.

The intensity was calculated using Equation 12 where the signals were only summed for transmitter - receiver pairs that satisfied $\left|x_{tx,rx} - x\right| \leq \frac{D}{2}$, where D is the width of the aperture. The resulting intensity was then interpolated to smooth the signal by increasing the grid size from 200 x 200 to 600 x 600. The resulting image can be seen in Figure 9 (a) and Figure 9 (b) shows target points 8 and 3 merged into one point.

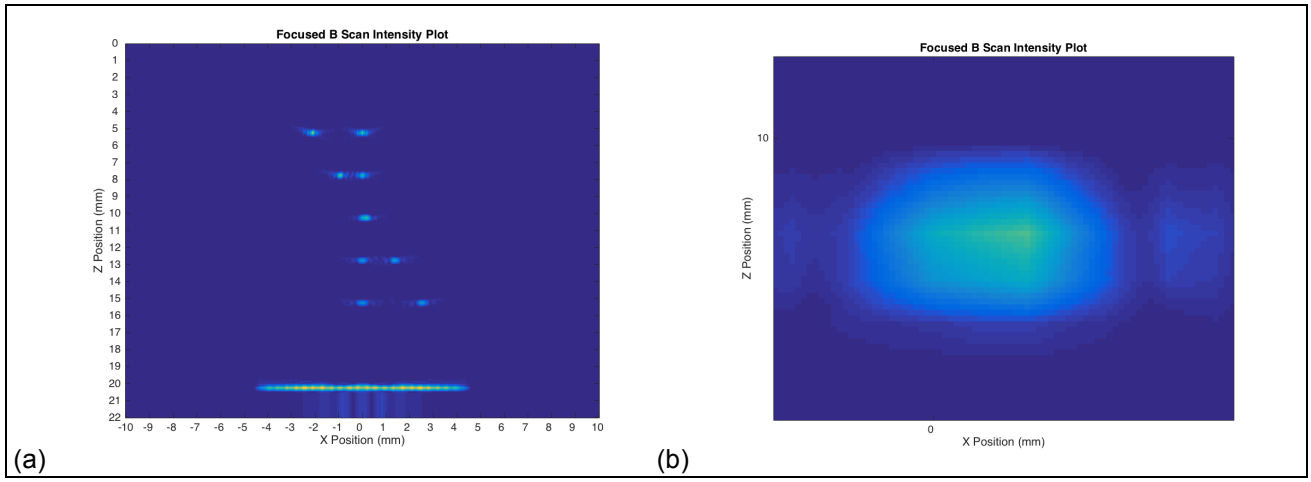


Fig. 9 (a) Focused B scan intensity plot (B) Focused B scan target points 3 and 8

Plane B Scan

The plane B scan post-processing imaging algorithm was then implemented, representing an aperture from the array being pulsed and then stepped along the array. A single element would have poor lateral resolution and sensitivity due to beam divergence and small element size respectively. This is overcome by using an aperture to obtain the time domain signals along the array, which are then combined to generate the complete B scan image. The intensity of the plane B scan was calculated using Equation 13,

$$I(\boldsymbol{r}) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}(\boldsymbol{r}) f_{ij}\left(\frac{2z}{c}\right) \tag{13}$$

where z is the normal distance from the transducer array to the grid point and c is the velocity of sound in water. The signals were only summed for transmitter - receiver pairs that satisfied $\left|x_{tx,rx} - x\right| \leq \frac{D}{2}$, where D is the width of the aperture. The resulting intensity was then interpolated to smooth the signal by increasing the grid size from 200 x 200 to 600 x 600. The resulting image can be seen in Figure 10 (a) and Figure 10 (b) shows target points 8 and 3 merged into a single line.

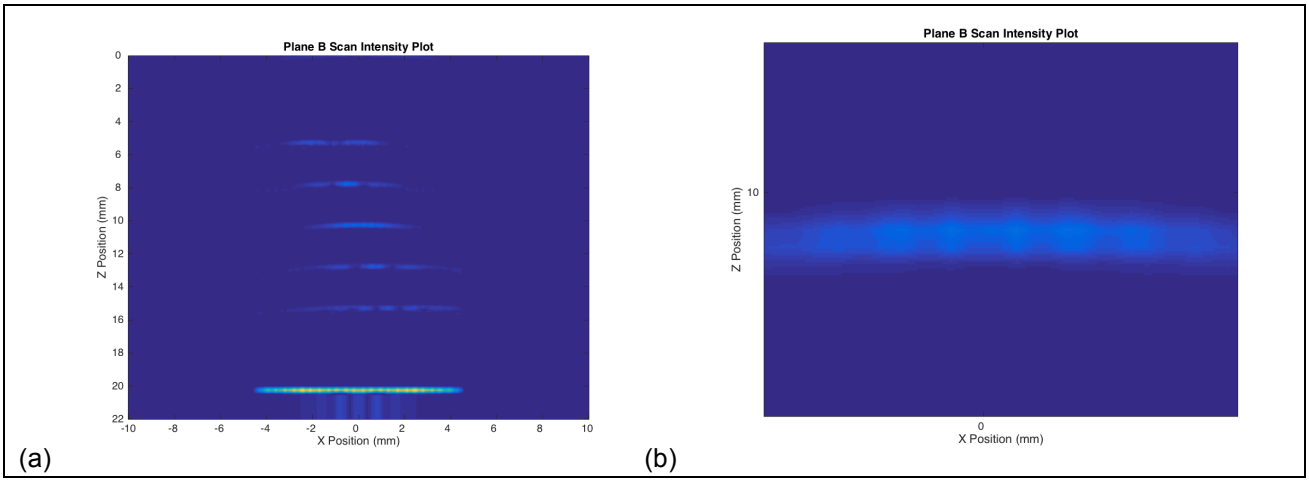


Fig. 10 (a) Plane B scan intensity plot (B) Plane B scan target points 3 and 8

Measuring Array Performance

An array performance indicator (API) as seen in [5] was used to quantitatively compare the different imaging algorithms. 'The API is a dimensionless measure of the spatial size of a point spread function. It is defined as the area, A_{-6dB}, within which the point spread function is greater than -6 dB down from its maximum value, normalised to the square of the wavelength' [5]. The API is given by Equation 14,

$$API = \frac{A_{-6dB}}{\lambda^2} \tag{14}$$

where λ is the wavelength. A_{-6dB} was determined by normalising the intensity and setting a cut off limit so that only the point maxima defining each focal point were left, as seen in Figure 11 (a). The area around these points were then interpolated and normalised once more so that any values less than half the peak could be set to zero. The area surrounding the point was then equivalent to A_{-6dB} and Figure 11 (b) shows this for target points 3 & 8 for the total focusing method.

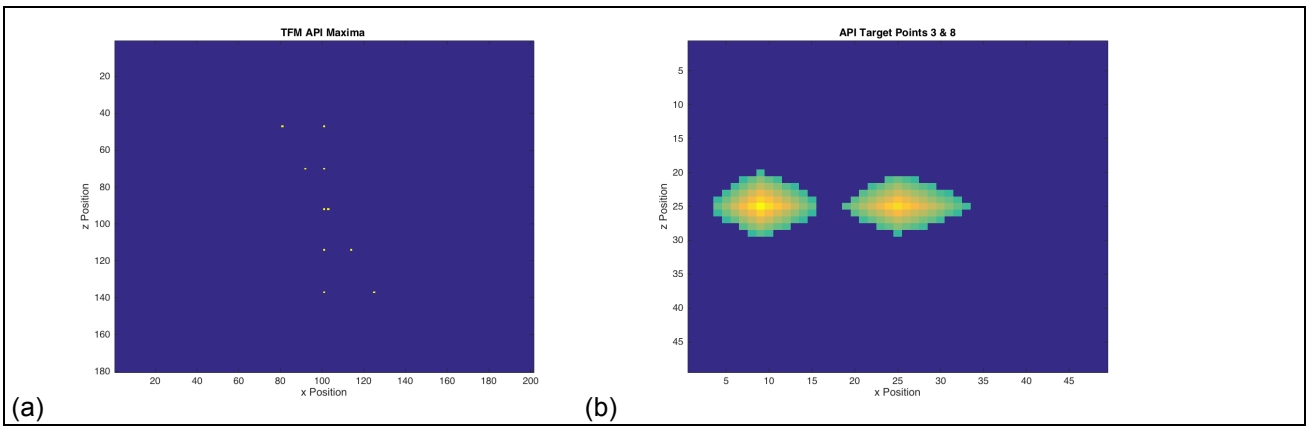


Fig. 11 (a) TFM API Maxima (B) TFM API target points 8 & 3

Due to the plane B scan's poor image it was hard to obtain API's for all of the target points so the API's of point 6 were calculated for each imaging algorithm. The parameters of the transducer and grid that were used for the API calculations are shown in Table 3. The focused B scan's and the plane B scan's apertures were set to 5 elements wide for the comparison between the three imaging algorithms.

The API results are shown in Table 4 where it can be observed that the TFM algorithm performed the best, followed by the focused B scan and then the plane B scan. The number of target points that each algorithm was able to detect was also recorded in Table 3 to provide a separate measure of each algorithms performance. The main lobe sharpness factor, q, was also calculated for each transducer set up focusing at target point 6 as a measure of how well the beam was focused.

Tab. 3 Transducer and grid setup for API calculations

Frequency (MHz)	Number of Elements (TFM), N	Element Width (mm)	Inter-Element Spacing (mm)	Grid Size	Aperture Size (B Scans) (No. of Elements)
12.5	64	0.05	0.05	0.0262	5

Lower API values indicate that the points are more pronounced and thus the algorithm is better at detecting points closer together. Figures 12 (a), (b) and (c) show the -6dB area around target point 6 for each imaging algorithm. It can be seen that the TFM algorithm provided a well defined area both laterally and axially, which would be expected as every point in the array focused on the target point, meaning that the beam had a good main lobe sharpness factor, q, resulting in good lateral resolution. All three algorithms have good axial resolution as this is obtained by setting the correct pulse length. The lateral resolution is good for the focused B scan although it cannot distinguish between target points 3 & 8. The lateral resolution of the plane B scan was very poor because no focusing was used. The main lobe sharpness factors for each algorithm are shown in Table 4 and suggest that the lower the value the better the lateral resolution.

Tab. 4 API values for post-processing imaging algorithms

	TFM	Focused B Scan	Plane B Scan
API	0.46	1.1705	1.86
Number of Target Points Detected	10/10	9/10 (Merged 3 & 8)	N/A (resolved to 5 lines)
Main Lobe Sharpness Factor, q	0.026	0.37	N/A - didn't focus

The TFM algorithm was able to detect and resolve every target point and had the lowest API, indicating that it was the best performing algorithm. The focused B scan was able to resolve and detect 8 of the points but was not able to distinguish between target points 3 & 8. The focused B scan can be improved by increasing the aperture size to 30 elements where it is able to distinguish target points 3 & 8. With an aperture of 30 elements the focused B scan's near field becomes 75mm meaning that it can focus within 75 mm as apposed to 2 mm when the aperture was 5 elements wide. When the TFM algorithm used only 30 elements it was unable to distinguish target points 3 & 8. When the focused B scan uses an aperture 30 elements wide the API is

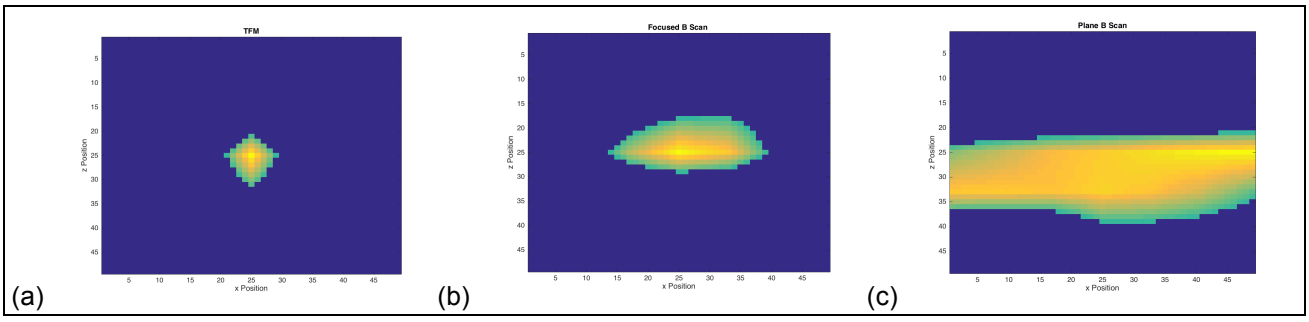


Fig. 12 (a) TFM A_{-6dB} (b) Focused B Scan A_{-6dB} (c) Plane B Scan A_{-6dB}

There are other performance indicators of the algorithms that need to be considered as well as their ability to detect and resolve the points. The computation time of each algorithm is extremely important as it affects the ability to use the algorithms to display images while the transducer is being held in place. It also affects how large an area can be inspected as for a required grid spacing, increasing the inspection area increases the

number of grid points, which results in even larger computation times. The computation time for each algorithm is expressed for 16, 32 and 64 elements in Table 5 but it should be noted that algorithms should be quantified by the number of operations as apposed to their computation time as it varies from computer to computer.

Tab. 5 Computation time and number of operations for post-processing imaging algorithms

	TFM			Plane B Scan			Focused B Scan		
Number of Elements	16	32	64	16	32	64	16	32	64
Computation Time (s)	2.31	12.11	53.94	0.75	2.77	12.37	0.62	2.50	10.23
Number of Operations	$\sim 10^7$	$\sim 4 \times 10^7$	$\sim 16 \times 10^7$	$\sim 2 \times 10^5$	$\sim 8 \times 10^5$	$\sim 34 \times 10^5$	$\sim 2 \times 10^5$	$\sim 8 \times 10^5$	$\sim 34 \times 10^5$

It can be seen that the TFM algorithm requires significantly more operations that both the plane B scan and the focused B scan. Increasing the number of elements by X increases the number of operations by 2^X. Increasing the size of the grid will have the same affect because all of the algorithms have used nested loops to iterate through all of the x and z grid positions. Increasing the grid size or reducing the grid spacing will therefore have a significant impact on the computation time of the algorithms. The 64 element focused B scan had 50 times less operations than the 64 TFM algorithm and therefore has a significantly lower computation time.

Array transducers are more flexible than conventional transducers as they can perform the roles of many of them. They can perform electronic scanning using delays, which is more reliable than manual and mechanical scanning. Array transducers are also able to focus at multiple depths, which conventional transducers cannot and are capable of advanced post-processing techniques.

CONCLUSION

There are a number of parameters that must be considered when selecting a transducer for a specific application. No time requirement was given for the transducer so it can be concluded that if full matrix capture is possible then the total focusing method is the best post-processing imaging algorithm to use. It provides the best resolution both axially and laterally and was the only algorithm capable of distinguishing between target points 3 & 8. However, the algorithm is extremely computationally expensive and if this is a requirement then the focused B scan should be used at it's performance approaches that of TFM as it's aperture size is increased but is less computationally expensive.

For the TFM post-processing imaging algorithm the following transducer set up allowed all of the target points to be inspected.

Tab. 6 Optimised transducer setup (focusing at point 1)

Frequency (MHz)	Number of Elements, N	Element Width (mm)	Inter-Element Spacing (mm)
12.5	64	0.05	0.05

REFERENCES

1. University of Bristol, "Huygen's Principle Handout", http://www.bris.ac.uk/currentstudents/, accessed 6 December 2015.
2. R.S.C. Monkhouse, P.D. Wilcox and P. Cawley in Review of Progress in QNDE, vol. 16, eds. D.O. Thompson and D.E. Chimenti (Plenum Press, New York, 1997), p.877.
3. Yijun shi, Modelling of acoustic waves for linear phased arrays, 1998, p.42.
4. Li Li, Xinliang Yu, Fangqin Li and Baojia Chen, Parameter Optimization of Linear Phased Array Transducer for Defect Detection, The Open Automation and Control Systems Journal, 2014, 6, 488-492
5. Caroline Holmes, Bruce W. Drinkwater*, Paul D. Wilcox, Post-processing of the full matrix of ultrasonic transmit–receive array data for non-destructive evaluation, NDT&E International 38 (2005) 701–711
6. Thomas G. Nyland, John S. Mattoon, Small Animal Diagnostic Ultrasound Dec 2001, p.3
7. Axial and Lateral Resolution, http://www.isradiology.org/isr/docs_books/basic/Chapter8.pdf, Accessed 17th December 2015

Exercise 1

```
clear; %clear all variables from memory
close all; %close all windows
clc; %clear command window

%-----------------------------------------------------------------
%SETUP TRANSDUCER
%-----------------------------------------------------------------
%Ultrasonic wave parameters
velocity = 1500;
frequency = 5e6;
wavelength = velocity/frequency;
k = 2*pi/wavelength;
w=2*pi*frequency;

%Transducer array parameters
transducer_elements = 20;
element_width = 0.09e-3;
spacing = 0.05e-3;
transducer_pitch = element_width + spacing;
transducer_width = transducer_pitch * (transducer_elements-1);
a = element_width;
sources_per_wavelength = transducer_elements / (transducer_width/wavelength);

%Setup Transducer Sources
source_x_positions = linspace((-
transducer_width/2),(transducer_width/2),transducer_elements);

%-----------------------------------------------------------------
%SETUP GRID
%-----------------------------------------------------------------
grid_size = 20e-3;
grid_pts = 500;
x = transpose([-grid_size/2:grid_size/grid_pts:grid_size/2]);
z = [0:grid_size/grid_pts:grid_size];

%-----------------------------------------------------------------
%SETUP MESHGRID
%-----------------------------------------------------------------
[mx,mz,ms] = meshgrid(x,z,source_x_positions);

%-----------------------------------------------------------------
%CREATE MATRICES AND VARIABLES
%-----------------------------------------------------------------
p = zeros([length(x)]); %prepare output matrix
A = 1; %complex number representing the size and phase of the source
t=0;

tic
%-----------------------------------------------------------------
%CALCULATE THE FIELD
%-----------------------------------------------------------------
%Calculate r (distance from point source)
r = sqrt((mx-ms).^2 + mz.^2);

%Calculate p for all x and z
p = A .* r.^(-0.5) .* exp(1i.*(k.*r-w.*t));

%Calculate Phi
angle = acos(mz./r);

%Calculate the Directivity Function
```

```
Df = sin(0.5*k*a*sin(angle))./(0.5*k*a*sin(angle));

%Calculate Field P
P_xz = Df .* p;

%----------------------------------------------------------------------
%PLOT FIELD
%----------------------------------------------------------------------
p_xz = sum(P_xz,3);
p_xz = abs(p_xz);

%Create Figure
figure(1)
imagesc(p_xz)
hold on
title('Focused Beam Profile From a Linear Array')
xticklabels = -10:2:10;
xticks = linspace(1, size(p_xz, 2), numel(xticklabels));
yticklabels = 0:2:20;
yticks = linspace(1, size(p_xz, 2), numel(yticklabels));
set(gca, 'XTick', xticks, 'XTickLabel', xticklabels)
set(gca, 'YTick', yticks, 'YTickLabel', yticklabels)
xlabel('X Position (mm)');
ylabel('Z Position (mm)');
caxis([0 150])
```

Exercise 2

```
clear; %clear all variables from memory
close all; %close all windows
clc; %clear command window

%-----------------------------------------------------------------
%SETUP TRANSDUCER
%-----------------------------------------------------------------
%Ultrasonic wave parameters
velocity = 1500;
frequency = 7.5e6;
wavelength = velocity/frequency;
k = 2*pi/wavelength;
w=2*pi*frequency;

%Transducer array parameters
transducer_elements = 64;
N = transducer_elements;
element_width = 0.05e-3;
spacing = 0.1e-3;
transducer_pitch = element_width + spacing;
transducer_width = transducer_pitch * (transducer_elements-1);
a = element_width;
sources_per_wavelength = transducer_elements / (transducer_width/wavelength);

%Setup Transducer Sources
source_x_positions = linspace((-
transducer_width/2),(transducer_width/2),transducer_elements);

%-----------------------------------------------------------------
%SETUP GRID
%-----------------------------------------------------------------
grid_size = 20e-3;
grid_pts = 500;
x = transpose([-grid_size/2:grid_size/grid_pts:grid_size/2]);
z = [0:grid_size/grid_pts:grid_size];

%-----------------------------------------------------------------
%SETUP MESHGRID
%-----------------------------------------------------------------
[mx,mz,ms] = meshgrid(x,z,source_x_positions);

%-----------------------------------------------------------------
%DEFINE FOCAL POINTS
%-----------------------------------------------------------------
for focal_points = [-2.25 -1.00 0.25 1.50 2.75 0.00 0.00 0.00 0.00 0.00; 5.00
7.50 10.00 12.50 15.00 5.00 7.50 10.00 12.50 15.00]/1000
focal_points_index = round(transpose(grid_pts * (focal_points)/grid_size));
focal_points_index(1) = focal_points_index(1) + grid_pts/2;

%-----------------------------------------------------------------
%CHECK SETUP
%-----------------------------------------------------------------
near_field = transducer_width^2/(4*wavelength);
a_over_lambda = a/wavelength;
%-----------------------------------------------------------------
%CREATE MATRICES AND VARIABLES
%-----------------------------------------------------------------
p = zeros([length(x)]); %prepare output matrix
A = 1; %complex number representing the size and phase of the source
t=0;

%-----------------------------------------------------------------
```

```
%CALCULATE THE FIELD
%-------------------------------------------------------------------
%Calculate r (distance from point source)
r = sqrt((mx-ms).^2 + mz.^2);

%Calculate p for all x and z
p = A .* r.^(-0.5) .* exp(1i.*(k.*r-w.*t));

%Calculate Phi
angle = acos(mz./r);

%Calculate the Directivity Function
Df = sin(0.5*k*a*sin(angle))./(0.5*k*a*sin(angle));

%Calculate Time Delay
for ii = 1: length(source_x_positions)
    d0(ii) = sqrt(focal_points(2)^2 + focal_points(1)^2);
    dj(ii) = sqrt(focal_points(2)^2 + (focal_points(1) -
(source_x_positions(ii)) )^2);
    tj(ii) = (dj(ii)-d0(ii))/velocity;
    B(ii) = exp(-1i*w*tj(ii));
end

%Calculate Field P
for ii = 1 : length(source_x_positions)
    P_xz(:,:,ii) = B(ii) .* Df(1,ii) .* p(:,:,ii);
end
p_xz = sum(P_xz,3);
p_xz = abs(p_xz);

%Angle for
angle_q = 0.4229;

%-------------------------------------------------------------------
%CALCULATE MAIN LOBE SHARPNESS FACTOR, q
%-------------------------------------------------------------------
q = (1/pi)*((asin(sin(angle_q) + (wavelength/(transducer_elements*spacing))))
...
    - asin(sin(angle_q) - (wavelength/(transducer_elements*spacing))))

%-------------------------------------------------------------------
%CALCULATE GRATING LOBE INTENSITY
%-------------------------------------------------------------------
max_grating_lobe_I = max(max(p_xz(:,1:100)));
max_beam_I = max(max(p_xz(1:end,100:end)));

grating_lobe_ratio = max_grating_lobe_I / max_beam_I;

%-------------------------------------------------------------------
%OPTIMISATION TABLE
%-------------------------------------------------------------------
 table(a_over_lambda,near_field,q,grating_lobe_ratio)

%-------------------------------------------------------------------
%PLOT FIELD
%-------------------------------------------------------------------
figure(1)
imagesc(p_xz)
hold on
title('Focused Beam Profile')
xticklabels = -10:2:10;
xticks = linspace(1, size(p_xz, 2), numel(xticklabels));
yticklabels = 0:2:20;
```

```
yticks = linspace(1, size(p_xz, 2), numel(yticklabels));
set(gca, 'XTick', xticks, 'XTickLabel', xticklabels)
set(gca, 'YTick', yticks, 'YTickLabel', yticklabels)
xlabel('Z Position (mm)');
ylabel('X Position (mm)');
scatter(focal_points_index(1),focal_points_index(2),30,'r')
pause(1)

end
```

Exercise 3

```
clear; %clear all variables from memory
close all; %close all windows
clc; %clear command window
tic

%-----------------------------------------------------------------
%SETUP TRANSDUCER
%-----------------------------------------------------------------
%Ultrasonic wave parameters
velocity = 1500;
frequency = 12.5e6;
wavelength = velocity/frequency;
k = 2*pi/wavelength;
w=2*pi*frequency;
distance_to_wall = 20/1000;

%Transducer array parameters
transducer_elements = 16;
element_width = 0.05e-3;
spacing = 0.05e-3;
transducer_pitch = element_width + spacing;
transducer_width = transducer_pitch * (transducer_elements-1);
a = element_width;
sources_per_wavelength = transducer_elements / (transducer_width/wavelength);

%Setup Transducer Sources
source_x_positions = linspace((-
transducer_width/2),(transducer_width/2),transducer_elements);

%-----------------------------------------------------------------
%SETUP GRID
%-----------------------------------------------------------------
grid_size = 22e-3;
grid_pts = 200;
x = transpose([-grid_size/2:grid_size/grid_pts:grid_size/2]);
z = [0:grid_size/grid_pts:grid_size];

%-----------------------------------------------------------------
%SETUP MESHGRID
%-----------------------------------------------------------------
[mx,mz,ms] = meshgrid(x,z,source_x_positions);

%-----------------------------------------------------------------
%DEFINE FOCAL POINTS
%-----------------------------------------------------------------
point_reflectors = [-2.25 -1.00 0.25 1.50 2.75 0.00 0.00 0.00 0.00 0.00; 5.00
7.50 10.00 12.50 15.00 5.00 7.50 10.00 12.50 15.00];
point_reflectors = point_reflectors/1000;

%-----------------------------------------------------------------
%RETRIEVE INDEXES OF FOCAL POINTS
%-----------------------------------------------------------------
focal_points_index = round(transpose(grid_pts * (point_reflectors)/grid_size));
focal_points_index(:,1) = focal_points_index(:,1) + grid_pts/2;
back_wall_index = round(transpose(grid_pts * (distance_to_wall)/grid_size));

%-----------------------------------------------------------------
%CREATE MATRICES AND VARIABLES
%-----------------------------------------------------------------
A = 1; %complex number representing the size and phase of the source
A_scatter = 0.01;
focal_points = transpose(point_reflectors); %Create Matrix of All Focal Points
```

```
%--------------------------------------------------------------------
%SETUP HANNING WINDOW
%--------------------------------------------------------------------
r = sqrt((mx-ms).^2 + mz.^2);
N = 5;
fft_points = 2048;
time_centre_pulse = (N/2)*wavelength/velocity;
pulse_time = (N*wavelength)/velocity;
max_time = 2*distance_to_wall/velocity + pulse_time;
%max_time = 2*r(end,grid_pts/2,1)/ velocity + pulse_time;
time = linspace(0,max_time,fft_points);
peak_pos_fract = (pulse_time./2)./max_time;
window = fn_hanning(fft_points, peak_pos_fract, peak_pos_fract);
input_signal = window.' .* sin(2.*pi.*frequency.*time - (time_centre_pulse));

%--------------------------------------------------------------------
%CALCULATE FFT OF HANNING WINDOW
%--------------------------------------------------------------------
fftpoints = 2.^nextpow2(length(time));
fourier_transform = fft(input_signal,fftpoints);
fourier_transform = fourier_transform(1:end/2);
frequency_sample = 1./(max(gradient(time)));
frequency_step = frequency_sample./fftpoints;
frequency_fourier =
[0:frequency_step:frequency_step.*(length(fourier_transform)-1)];

%--------------------------------------------------------------------
%CREATE 3D MATRICES FOR VARIABLES
%--------------------------------------------------------------------
fourier_transform = repmat(fourier_transform, [transducer_elements 1
transducer_elements]);
fourier_transform = permute(fourier_transform, [3 1 2]);

k = 2.*pi.*frequency_fourier./velocity;
k_a = repmat(k, [transducer_elements 1]);
k_b = repmat(k, [transducer_elements 1 transducer_elements]);
k_b = permute(k_b, [3 1 2]);
k_a = permute(k_a, [3 1 2]);

H_txrx =
zeros(transducer_elements,transducer_elements,length(frequency_fourier));
for ii = 1 : length(focal_points)
    %--------------------------------------------------------------------
    %CALCULATE DISTANCES
    %--------------------------------------------------------------------
    R_r = sqrt( (focal_points(ii,1) - source_x_positions(:)).^2 +
(focal_points(ii,2)).^2);
    R_repmat = repmat(R_r, [1 transducer_elements]);
    R_permute = permute(R_repmat, [2 1]);
    d_txrx = R_permute + R_repmat;
    d_txrx = repmat(d_txrx, [1 1 size(frequency_fourier,2)]);

    %--------------------------------------------------------------------
    %COMPLEX SPECTRUM OF PHASE SHIFTED SIGNAL
    %--------------------------------------------------------------------
    G_txrx = fourier_transform .* exp(-1i .* k_b .* d_txrx);

    %--------------------------------------------------------------------
    %CALCULATE AMPLITUDE
    %--------------------------------------------------------------------
    A_txrx = A_scatter ./ sqrt(R_repmat .* R_permute);
    A_txrx = repmat(A_txrx, [1 1 length(fourier_transform)]);
```

```
    %-------------------------------------------------------------------
    %CALCULATE ANGLE
    %-------------------------------------------------------------------
    angle_points = atan( (focal_points(ii,1) - source_x_positions(:)) ./
focal_points(ii,2));
    angle_points = repmat(angle_points, [1 transducer_elements
length(fourier_transform)]);
    angle_points(isnan(angle_points))=1;

    %-------------------------------------------------------------------
    %CALCULATE DIRECTIVITY
    %-------------------------------------------------------------------
    ptx = sin( 0.5 .* k_b .* a .* sin(angle_points))./( 0.5 .* k_b .* a .*
sin(angle_points));
    prx = permute(ptx, [2 1 3]);

    %-------------------------------------------------------------------
    %CALCULATE RESULTING SPECTRUM
    %-------------------------------------------------------------------
    H_txrx_old = A_txrx .* G_txrx;
    H_txrx = H_txrx + H_txrx_old;
end

%-------------------------------------------------------------------
%SAME AGAIN FOR BACK WALL
%-------------------------------------------------------------------
r_wall = repmat(source_x_positions, [transducer_elements 1]);
r_wall = (r_wall - permute(r_wall, [2 1])) ./ 2;
r_wall = sqrt(r_wall.^2 + distance_to_wall.^2);
r_wall = 2 .* r_wall;
d_txrx_wall = repmat(r_wall, [1 1 length(fourier_transform)]);

A_txrx_wall = A ./ sqrt(d_txrx_wall);

G_txrx_wall = fourier_transform .* exp(-1i .* k_b .* d_txrx_wall);

%-------------------------------------------------------------------
%DIRECTIVITIES
%-------------------------------------------------------------------
angle_points_wall = acos( (distance_to_wall) ./ (d_txrx_wall ./ 2));
p_wall = sin( 0.5 .*  k_b .* a .* sin(angle_points_wall))./( 0.5 .* k_b .* a .*
sin(angle_points_wall));
p_wall(isnan(p_wall))=1;
p_t_wall = permute(p_wall, [2 1 3]);

H_txrx_wall =  p_wall .* p_t_wall .* A_txrx_wall .* G_txrx_wall;

%-------------------------------------------------------------------
%SUM SIGNALS FOR POINTS AND WALL
%-------------------------------------------------------------------
H_txrx =   H_txrx + H_txrx_wall;

%-------------------------------------------------------------------
%CALCULATE TIME DOMAIN SIGNAL
%-------------------------------------------------------------------
time_domain = ifft(H_txrx,size(time,2),3);

trans = 1;
rec = 1;

plot_dat = time_domain(:,:,1:length(time));

plot_dat = plot_dat(trans,rec,:);
```

```
plot_dat = (permute(plot_dat, [3 1 2]));
hold off
plot(time(1:length(plot_dat)),(plot_dat));
hold on

title('Time Domain Signal for Transmitter 1 & Receiver 1')
xlabel('Time (s)');
ylabel('Amplitude');
caxis([0 150])
```

Exercise 4 – TFM

```
Exercise_5_3
tic
%-----------------------------------------------------------------------
%CALCULATE DELAYS
%-----------------------------------------------------------------------
max_time = 2*distance_to_wall/velocity;
r = sqrt((mx-ms).^2 + mz.^2);
d_txrx_repmat = repmat(r, [1 1 1 transducer_elements]);
d_txrx_repmat = d_txrx_repmat + permute(d_txrx_repmat, [1 2 4 3]);
d_txrx_new = (d_txrx_repmat) ./ velocity;

%-----------------------------------------------------------------------
%FIND TIME INDEXES
%-----------------------------------------------------------------------
% index = round((d_txrx_new ./ max_time) .* fft_points);
% index(index==0) = 1;
% index(index>fft_points) = fft_points;

index = round((d_txrx_new ./ max_time) .* fft_points);
index(index==0) = 1;
index(index>fft_points) = fft_points;

%-----------------------------------------------------------------------
%INITIALISE INTENSITY MATRIX
%-----------------------------------------------------------------------
I =
zeros(length(x),length(z),length(source_x_positions),length(source_x_positions))
;

%-----------------------------------------------------------------------
%LOOP FOR INTENSITY
%-----------------------------------------------------------------------
aa=0;
for ii = 1 : length(x)
    for jj = 1 : length(z)
        for kk = 1 : length(source_x_positions)
            for ll = 1 : length(source_x_positions)
                I(ii,jj,kk,ll) = time_domain(kk,ll,index(ii,jj,kk,ll));
                aa = aa + 1;
            end
        end
    end
end
toc
I = sum(I,3);
I = sum(I,4);

%-----------------------------------------------------------------------
%DETERMINE API
%-----------------------------------------------------------------------
I_API = abs(I/max(max(I)));
API_cutoff = I_API;
API_cutoff(API_cutoff <= 0.2) = 0;

maxima = imregionalmax(API_cutoff(1:grid_pts-50,:));
[row col] = find(maxima);

figure(length(row)+1)
imagesc(maxima)
tol = 3;
interp_val = 3;
pixel_size = (grid_size/grid_pts)^2 ./ (interp_val * 2^(interp_val-1) - 1)^2;
```

```
for ii = 1 : length(row)
    API = I_API(row(ii)-tol:row(ii)+tol,col(ii)-tol:col(ii)+tol);
    interp_I = interp2(API,interp_val);
    interp_I = interp_I/max(max(interp_I));
    interp_I(interp_I < 0.5) = 0;
    area_data = regionprops('struct',im2bw(interp_I),'Centroid','Area');
    data(ii,1) = row(ii) * grid_pts / grid_size;
    data(ii,2) = col(ii) * grid_pts / grid_size;
    data(ii,3) = area_data(1).Area * pixel_size;
    data(ii,4) = data(ii,3) / wavelength^2;

    figure(ii)
    imagesc(abs(interp_I))
end

I_2 = interp2(I,3);

%-----------------------------------------------------------------------
%PLOT INTENSITY
%-----------------------------------------------------------------------
hold off
figure(length(row)+2)
imagesc(abs(I_2))
hold on
title('TFM Intensity Plot')
xticklabels = -10:1:10;
xticks = linspace(1, size(I_2, 2), numel(xticklabels));
yticklabels = 0:1:22;
yticks = linspace(1, size(I_2, 2), numel(yticklabels));
set(gca, 'XTick', xticks, 'XTickLabel', xticklabels)
set(gca, 'YTick', yticks, 'YTickLabel', yticklabels)
xlabel('X Position (mm)');
ylabel('Z Position (mm)');
```

Exercise 4 – Focused B Scan

```
Exercise_5_3
tic
%-----------------------------------------------------------------------
%CALCULATE DELAYS
%-----------------------------------------------------------------------
r = sqrt((mx-ms).^2 + mz.^2);
d_txrx_repmat = repmat(r, [1 1 1 transducer_elements]);
d_txrx_repmat = d_txrx_repmat + permute(d_txrx_repmat, [1 2 4 3]);
d_txrx_new = (d_txrx_repmat) ./ velocity;

%-----------------------------------------------------------------------
%FIND TIME INDEXES
%-----------------------------------------------------------------------
index = round((d_txrx_new ./ max_time) .* length(time));
index(index==0) = 1;
index(index>length(time)) = length(time);

%-----------------------------------------------------------------------
%APERTURE
%-----------------------------------------------------------------------
D = transducer_pitch * 5 - spacing;

%-----------------------------------------------------------------------
%INITIALISE INTENSITY MATRIX
%-----------------------------------------------------------------------
I =
zeros(length(x),length(z),length(source_x_positions),length(source_x_positions))
;

%-----------------------------------------------------------------------
%LOOP FOR INTENSITY
%-----------------------------------------------------------------------
aa = 0;
for ii = 1 : length(x)
    for jj = 1 : length(z)
        for kk = 1 : length(source_x_positions)
            for ll = 1 : length(source_x_positions)
                allow = abs(source_x_positions(kk) - mx(1,jj,1));
                if D/2 >= allow
                    I(ii,jj,kk,ll) = time_domain(kk,ll,index(ii,jj,kk,ll));
                    aa = aa + 1;
                end
            end
        end
    end
end
toc
I = sum(I,3);
I = sum(I,4);

%-----------------------------------------------------------------------
%DETERMINE API
%-----------------------------------------------------------------------
I_API = abs(I/max(max(I)));
API_cutoff = I_API;
API_cutoff(API_cutoff <= 0.2) = 0;

maxima = imregionalmax(API_cutoff(1:grid_pts-20,:));
[row col] = find(maxima);

hold off
figure(length(row)+1)
```

```
hold on
imagesc(maxima)
title('Focused B Scan API Maxima')
xlabel('x Position')
ylabel('z Position')
tol = 3;
interp_val = 3;
pixel_size = (grid_size/grid_pts)^2 ./ (interp_val * 2^(interp_val-1) - 1)^2;

for ii = 1 : length(row)
    API = I_API(row(ii)-tol:row(ii)+tol,col(ii)-tol:col(ii)+tol);
    interp_I = interp2(API,interp_val);
    interp_I = interp_I/max(max(interp_I));
    interp_I(interp_I < 0.5) = 0;
    area_data = regionprops('struct',im2bw(interp_I),'Centroid','Area');
    data(ii,1) = row(ii) * grid_pts / grid_size;
    data(ii,2) = col(ii) * grid_pts / grid_size;
    data(ii,3) = area_data(1).Area * pixel_size;
    data(ii,4) = data(ii,3) / wavelength^2;

    figure(ii)
    imagesc(abs(interp_I))
end

I_2 = interp2(I,3);

%-----------------------------------------------------------------------
%PLOT INTENSITY
%-----------------------------------------------------------------------
hold off
figure(length(row)+2)
imagesc(abs(I))
hold on
title('Focused B Scan Intensity Plot')
xticklabels = -10:1:10;
xticks = linspace(1, size(I, 2), numel(xticklabels));
yticklabels = 0:1:22;
yticks = linspace(1, size(I, 2), numel(yticklabels));
set(gca, 'XTick', xticks, 'XTickLabel', xticklabels)
set(gca, 'YTick', yticks, 'YTickLabel', yticklabels)
xlabel('X Position (mm)');
ylabel('Z Position (mm)');
```

Exercise 4 – Plane B Scan

```
tic
Exercise_5_3

%-----------------------------------------------------------------------
%CALCULATE B SCAN TIME
%-----------------------------------------------------------------------
time_B_scan = (2.*mz)./velocity;
d_txrx_repmat = repmat(time_B_scan, [1 1 1 transducer_elements]);

%-----------------------------------------------------------------------
%FIND TIME INDEXES
%-----------------------------------------------------------------------
index = round((d_txrx_repmat ./ max_time) .* fft_points);
index(index==0) = 1;
index(index>fft_points) = fft_points;

%-----------------------------------------------------------------------
%APERTURE
%-----------------------------------------------------------------------
D = transducer_pitch * 5 - spacing;

%-----------------------------------------------------------------------
%INITIALISE INTENSITY MATRIX
%-----------------------------------------------------------------------
I =
zeros(length(x),length(z),length(source_x_positions),length(source_x_positions))
;

%-----------------------------------------------------------------------
%LOOP FOR INTENSITY
%-----------------------------------------------------------------------
aa=0;
for ii = 1 : length(x)
    for jj = 1 : length(z)
        for kk = 1 : length(source_x_positions)
            for ll = 1 : length(source_x_positions)
                allow = abs(source_x_positions(kk) - mx(1,jj,1));
                if D/2 >= allow
                    I(ii,jj,kk,ll) = time_domain(kk,ll,index(ii,jj,kk,ll));
                    aa = aa + 1;
                end
            end
        end
    end
end
toc
I = sum(I,3);
I = sum(I,4);

%-----------------------------------------------------------------------
%DETERMINE API
%-----------------------------------------------------------------------
I_API = abs(I/max(max(I)));

API_cutoff = I_API;
API_cutoff(API_cutoff <= 0.1) = 0;

maxima = imregionalmax(API_cutoff(1:grid_pts-20,10:grid_pts-10));
[row col] = find(maxima);

figure(length(row)+1)
hold off
```

```
imagesc(maxima)
hold on
title('Focused B Scan API Maxima')
xlabel('x Position')
ylabel('z Position')
hold off
tol = 3;
interp_val = 3;
pixel_size = (grid_size/grid_pts)^2 ./ (interp_val * 2^(interp_val-1) - 1)^2;

for ii = 1 : length(row)
    API = I_API(row(ii)-tol:row(ii)+tol,col(ii)-tol:col(ii)+tol);
    interp_I = interp2(API,interp_val);
    interp_I = interp_I/max(max(interp_I));
    interp_I(interp_I < 0.5) = 0;
    %imagesc(abs(interp_I))
    %API(API>=0.5) = 1;
    area_data = regionprops('struct',im2bw(interp_I),'Centroid','Area');
    data(ii,1) = row(ii) * grid_pts / grid_size;
    data(ii,2) = col(ii) * grid_pts / grid_size;
    data(ii,3) = area_data(1).Area * pixel_size;
    data(ii,4) = data(ii,3) / wavelength^2;

    figure(ii)
    imagesc(abs(interp_I))
end


I_2 = interp2(I,3);
%-----------------------------------------------------------------------
%PLOT INTENSITY
%-----------------------------------------------------------------------
hold off
figure(length(row)+2)
imagesc(abs(I_2))
hold on
title('Plane B Scan Intensity Plot')
xticklabels = -10:2:10;
xticks = linspace(1, size(I_2, 2), numel(xticklabels));
yticklabels = 0:2:grid_size*1000;
yticks = linspace(1, size(I_2, 2), numel(yticklabels));
set(gca, 'XTick', xticks, 'XTickLabel', xticklabels)
set(gca, 'YTick', yticks, 'YTickLabel', yticklabels)
xlabel('X Position (mm)');
ylabel('Z Position (mm)');
```