Exercise 5: Array Design

Matthew Chandler

ABSTRACT

An ultrasonic phased array was designed using MATLAB v9.9.0 (R2020b) for the purpose of inspecting a simulated water tank containing a series of defects. In achieving this design, the beam profile from a linear array was predicted and optimised. The full matrix of time-domain ultrasonic signals was then simulated for the phased array inspecting a sample with known defect locations. These time-traces were then focussed using the total focussing method (TFM), to inspect the defects in the sample.

INTRODUCTION AND BACKGROUND

Ultrasonic phased arrays have several advantages over monolithic transducers when inspecting a component for defects, including flexibility (capable of performing many functions), reliability (electronic scanning over a region of interest as opposed to mechanical scanning) and range of application: many functions commonly performed with an array are not possible to achieve with a single transducer, e.g. setting a variable focal length during or post scan. As a result, they are a staple of ultrasonic non-destructive testing (NDT). Of particular interest to the work described in this report is the modelling of ultrasonic phased arrays in a simulated environment rather than the physical implementation and experimentation: models provide evidence that the physics of the system to be inspected is well understood, and provides a much cheaper and more accessible means for evaluating the performance of an array than physical prototyping.

In 2D, the acoustic pressure field P at coordinates (x, z) due to a transducer can be found by dividing the source into n elements and adding up the individual contributions [1]. It is given by equation

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

where r_j is the distance from the jth element in the probe to the coordinates (x, z) defined as $r_j = \sqrt{\left(x - x_j\right)^2 + \left(z - z_j\right)^2}$ at time t, k and ω are the wavenumber and frequency of the wave respectively and A is a complex number expressing the amplitude and phase of the source. This model assumes that the length of the transducer in the out-of-plane direction is much longer than the length in the plane. A phased array can be modelled using the same equation, modelling each individual transducer within the array as an element.

Common methods of inspection with arrays include plane, focussed and sector scans [2], which delays the signal from each element such that the shape of the ultrasonic beam generated is changed. If the complete set of time-domain signals is collected independently from every pair of transmitter-receivers in the array (the collection of which is called full-matrix capture, FMC), any of these inspection methods can be used in post-processing to produce an image, $I(\boldsymbol{r})$, given by

$$I(x,z) = \sum_{i,j=1}^{N} a_{ij} f_{ij}\left(\tau_{ij}(x,z)\right) \tag{2}$$

where f_{ij} is the time-trace recorded by the jth element in the array on the ith transmitting element (i.e. the (i, j) element in the FMC data), a_{ij} is the amplitude applied to the time-trace obtained by the jth receiving element from the signal broadcast by the ith transmitting element, and τ_{ij} is the delay applied to the ith transmitting or jth receiving element. Both a, τ are dependent on the method of inspection.

The total focussing method (TFM) is an imaging method applied in post-processing after collection of FMC data, where the beam is focussed at every point in the image. This is achieved by finding the time taken for the wave to travel from the ith transmitter to the point (x, z) in the image and then to the jth receiver,

$$\tau_{ij}(\boldsymbol{r}) = \frac{|\boldsymbol{r} - \boldsymbol{r}_i|}{c} + \frac{|\boldsymbol{r} - \boldsymbol{r}_j|}{c} \tag{3}$$

where c is the speed of the wave in the medium, $\boldsymbol{r} = (x, z)$, $\boldsymbol{r}_i, \boldsymbol{r}_j$ the position of the transmitter and receiver respectively.

It is of note that the TFM occurs entirely in post-processing. As a result, it is not required that the data is obtained experimentally: it can be obtained from a simulation. The work described here used a ray-tracing method [3] to obtain the path of the wave from transducer to scatterer, and simulated the FMC data using a linear system approach [4]. As the waves of interest pass through one medium only, the ray is a simple line-segment drawn from the transducer element to the scatterer.

LINEAR BEAM PROFILE

The inspection target studied in this report was a calibration sample for an ultrasonic phased array designed to operate into a human body. The sample consists of a series of point targets suspended in a water tank. Using this sample, the properties of the phased array were designed such that the point targets are well resolved.

Initially, the beam profile from a linear array into water was predicted using a 2D MATLAB program. This was done using equation 1, where the amplitude was defined as $A_j = e^{i\omega t_j} D_f(\phi_j)$ for the jth array element. In this definition, t_j is the time taken for the wave to travel from the jth element to the focal point; ϕ_j is the angle that the ray makes with the probe element normal, and D_f is the directivity function of the array element defined by equation

$$D_f(\phi) = a\,\mathrm{sinc}\left(\frac{1}{2}ka\sin\phi\right) \tag{4}$$

where a is the array element width, with pitch $p = a + g$, the separation between adjacent elements. The resulting profiles for a range of transducer frequencies and element pitch values are shown in fig 1.

The resulting beam profiles indicate that over the range examined, the focus matches with the focal point fairly well, particularly for figs 1(c, e-i). It can be inferred that the transducer centre frequency f appears to influence the lateral width of the focus, and the pitch influences the length of the focus when number of elements and

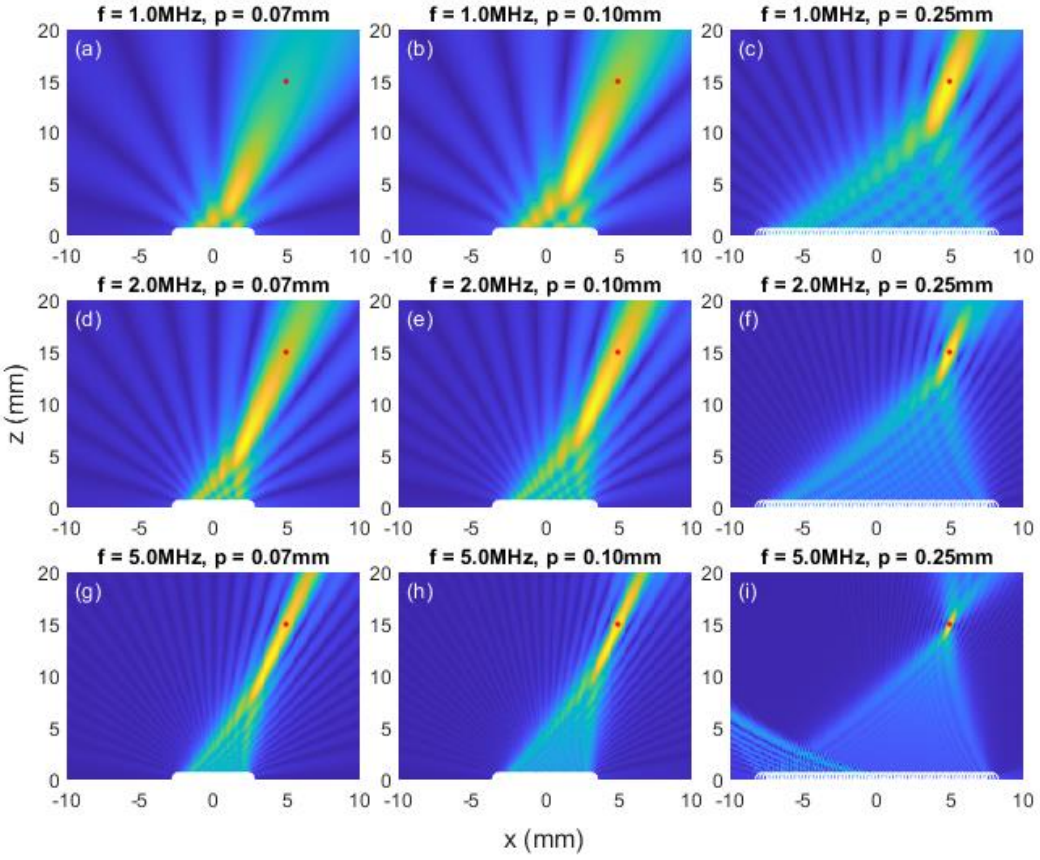

Fig. 1 (a-i) Beam profiles for transducer frequencies $f = 1, 2, 5\mathrm{MHz}$ and pitch values $p = 0.075, 0.10, 0.25\mathrm{mm}$. Array element locations are shown as white circles, and the beam focal point, chosen arbitrarily to be at coordinates $(5\mathrm{mm}, 15\mathrm{mm})$ is shown as a red circle. Number of elements in all cases is $n = 64$; element separation is $g = 0.05\mathrm{mm}$. A colour bar has not been included as absolute scale of the amplitude is not important, and each subfigure has independent limits.

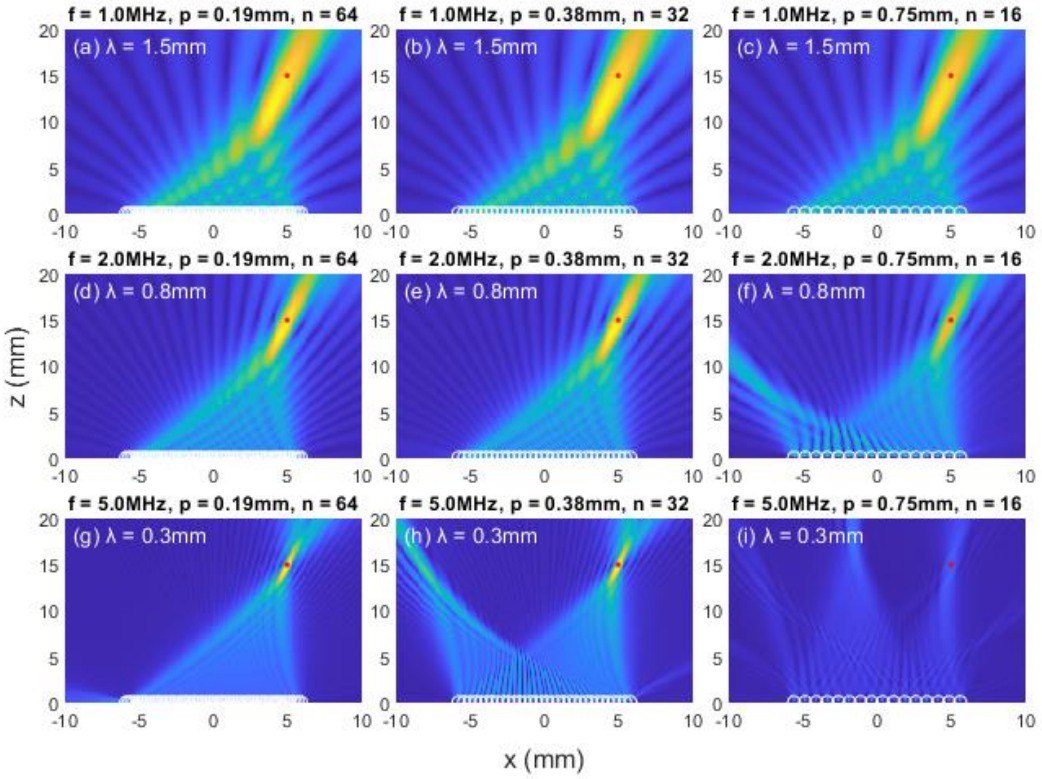

Fig. 2 (a-i) Beam profiles for transducer frequencies $f = 1, 2, 5\text{MHz}$, pitch values $p = 0.19, 0.38, 0.75\text{mm}$ and $n = 64, 32, 16$. Array element locations are shown as white circles, and the beam focal point is shown as a red circle. Element separation in all cases is 0.05mm. Note that the total length of the array in all cases is $L = pn = 12\text{mm}$. Note that each subfigure is plotted independently of each other.

point for small pitch values at a constant frequency. Additionally, the effect of the total number of elements used was studied as centre frequency was varied: the resulting beam profiles are shown in fig 2. This was done by fixing the total array length as constant $L = pn = 12\text{mm}$, and varying both the pitch and number of elements in the array. In these plots, the overall shape of the focus appears to remain roughly constant (i.e. aspect ratio), with its size varying depending frequency and total array length. This was tested quantitatively by measuring the aspect ratio (AR) of the focus for all 18 configurations examined in figs 1 and 2 – this is shown in fig 3. It can be seen in figs 3a-i that the AR has much more variation than in figs 3j-r, providing

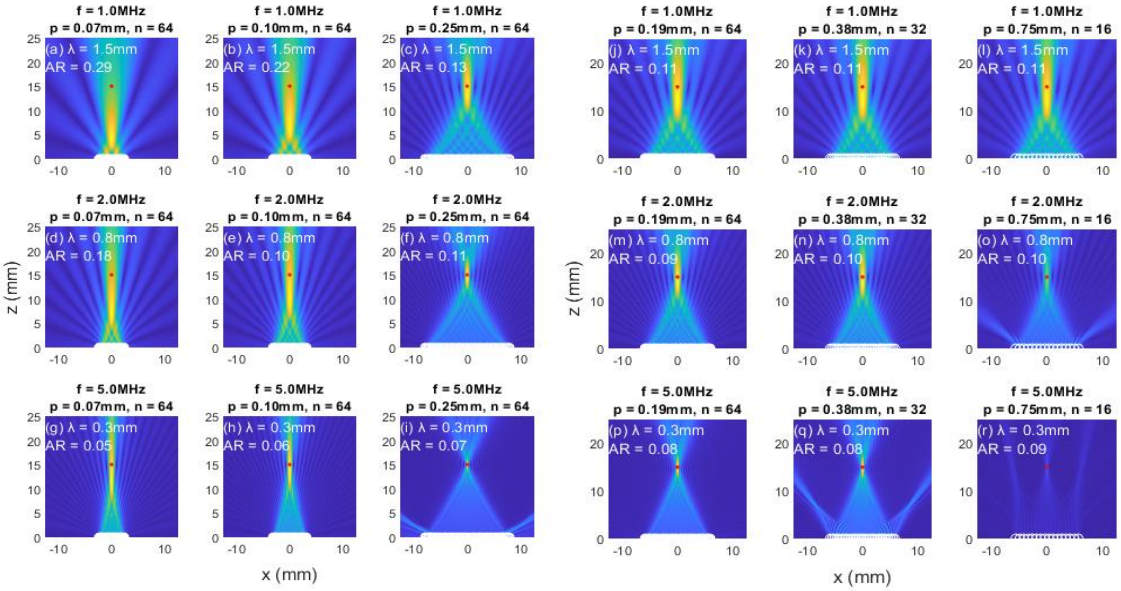

Fig. 3 (a-l; LHS) Examination of aspect ratio (AR) for configurations plotted in fig 1. There is a significant variation in AR values across these plots. (j-r; RHS) Examination of AR for configurations plotted in fig 2. The AR values are much more consistent in these plots.

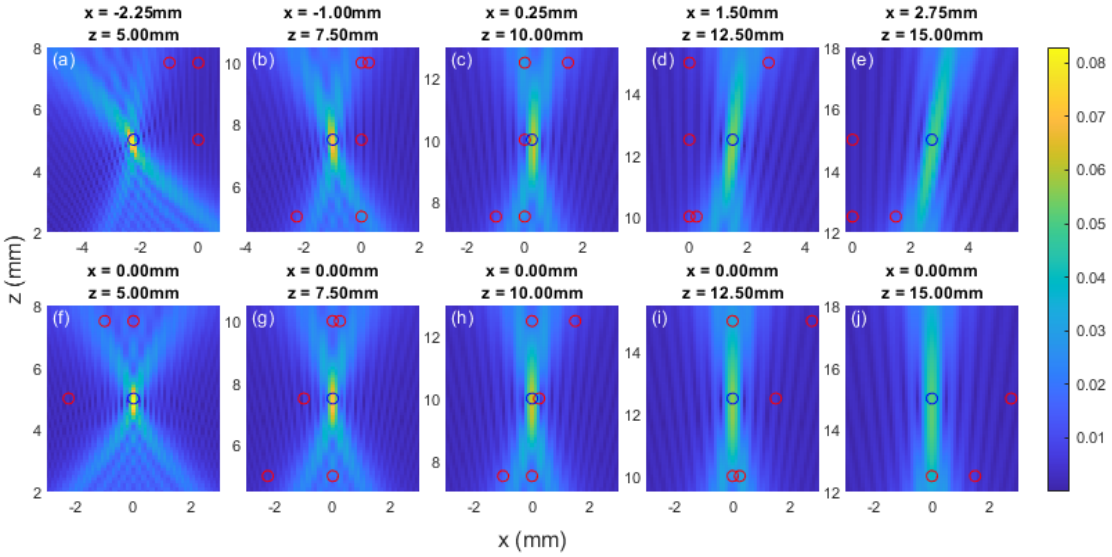

Fig. 4 Focal points set at each of the defect locations in the calibration sample, with transducer frequency $f = 5\text{MHz}$, pitch $p = 0.15\text{mm}$, total elements $n = 64$ and element separation $g = 0.05\text{mm}$. Note that the full region of interest has not been plotted, instead the focus has been magnified to inspect how well each point is resolved. The subfigures are each plotted on a common colour bar scale.

evidence that the overall length of the array has a strong influence over the shape of the focus, and the frequency has a strong influence over the size of the focus region.

In figs 2(f-i), there are grating lobes present in the $x < 0\text{mm}$ region. In the literature [2, 5], it is discussed that grating lobes arise as a result of the periodic boundary conditions placed on each element in the array in equation 1: a constant element pitch produces an infinite set of responses in the spatial Fourier domain. If multiple responses exist in the region of Fourier space which maps to physical space, then this is manifested as grating lobes in the resulting image. It is determined in the literature that grating lobes are suppressed when the pitch of the array $p < 0.5\lambda$.

The beam profiles shown in fig 2 display a good agreement with this condition, as it is satisfied in all cases where grating lobes are not present. In particular, fig 2g shows a very small grating lobe for $x < -5\text{mm}$ and small z, and it can be seen that the pitch is only slightly larger than half of the wavelength of the wave. Fig 2i shows the most significant grating lobes, where $p > 2\lambda$, the most significant deviation from the condition.

To resolve the defects in the calibration sample to a high resolution, figs 1-2 suggested that a high frequency and number of elements should be used, with pitch set to half the wavelength. The specification of the array required no more than 64 elements, with the gap between elements being at least 0.05mm. The smallest separation between adjacent scatterers in the medium is 0.25mm in the x-direction, and 2.5mm in the z-direction. Taking the largest frequency examined $f = 5\text{MHz}$, in order to achieve as fine a resolution as possible, the beam profiles with focal points set to the scatterer locations and these array specifications are plotted in fig 4. It can be seen in this figure that the points which are the least likely to be resolved individually are the points in fig 4c and 4h, and the points in fig 4i and 4j. This is because the focus is much longer in the z-direction than in the x-direction, and becomes longer as the focal point moves away from the array in the z-direction. In subsequent analysis, these points will be studied when examining the array's performance.

TIME-TRACE SIMULATION AND IMAGING ALGORITHMS

The FMC time-traces were simulated using a ray-based linear system approach [3, 4]. The array design used was a 64-element array with a pitch $p = 0.15\text{mm}$, centre frequency 5MHz and separation between elements 0.05mm. The calibration sample specified consists of a water tank with 10 point samples with locations defined in fig 5a, with the back wall of the tank located at 20mm from the array, which was treated as a perfect reflector (reflection coefficient $R \equiv 1$). The input signal was a 5-cycle toneburst, and the scattered signal was treated as $0.01u_{\text{inc}}(\omega)$. Directivity (equation 4) and beam spreading ($\propto \sqrt{d}^{-1}$, d is distance from array element to scatterer) effects were also modelled. The FMC time-traces are plotted in fig 5b.

Several different imaging algorithms were used to process the time-traces into an image: plane scan, focussed scan, sector scan and TFM. The intensity at any point in the image is defined using equation 2, where the delay time τ used to lookup the signal intensity in f_{ij} will be defined depending on the algorithm used.

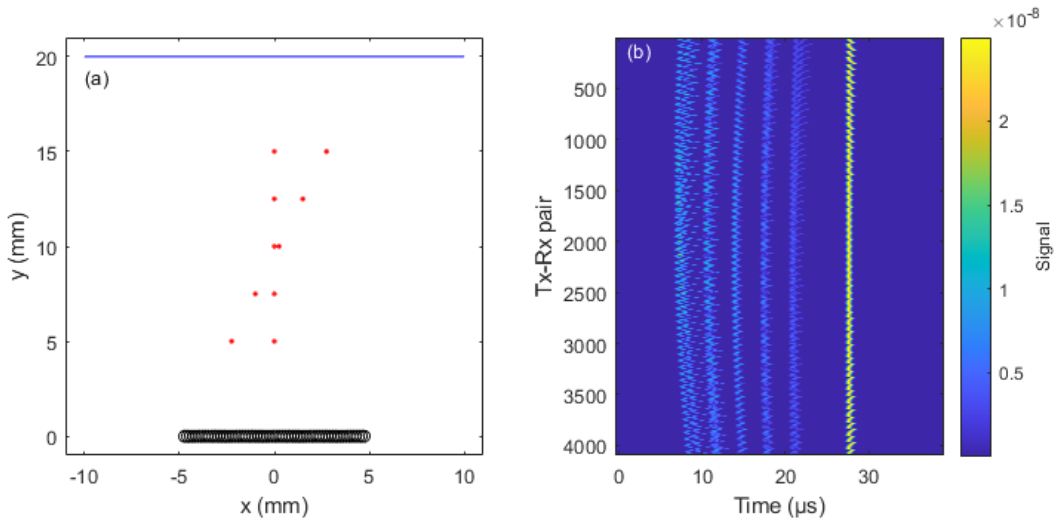

Fig. 5 (a) Geometry of the water tank. Back wall (blue line) is located 20mm away from the phased array (black circles). Ten scatterers (red circles) are spread throughout the tank. (b) Full matrix of time-traces obtained from all transmit-receive transducer pairs. The T_x-R_x index on the y-axis iterates through receivers first, transmitters second. For example, index 65 corresponds to $T_x = 2$, $R_x = 1$, etc. Scatterer signals are visible from $7\mu\text{s} < t < 23\mu\text{s}$; back wall signal visible at $t \cong 28\ \mu\text{s}$.

Additionally, for plane and focus scans, the amplitude term a is used to meet the summation conditions specified in the following equations. The final intensity for the plane, focussed, sector scans and TFM are covered in more detail by Holmes *et al.* [2], and are summarised below

$$I_{\text{Plane}}(x,z) = \left| \sum_{|x_{i,j}-x| \le D/2} f_{ij}\left(2z/c\right) \right| \tag{5}$$

$$I_{\text{Focus}}(x,z) = \left| \sum_{|x_{i,j}-x| \le D/2} f_{ij}\left(\frac{\sqrt{(x_i-x)^2+(z_i-z)^2} + \sqrt{(x_j-x)^2+(z_j-z)^2}}{c} \right) \right| \tag{6}$$

$$I_{\text{Sector}}(r,\theta) = \left| \sum_{i,j} f_{ij}\left(\frac{2r + x_i \sin\theta + x_j \sin\theta}{c} \right) \right| \tag{7}$$

$$I_{\text{TFM}}(x,z) = \left| \sum_{i,j} f_{ij}\left(\frac{\sqrt{(x_i-x)^2+(z_i-z)^2} + \sqrt{(x_j-x)^2+(z_j-z)^2}}{c} \right) \right| \tag{8}$$

In application, $z_{i,j} \equiv 0$, the aperture length D was defined such that 16 elements out of 64 were fired, and the sector scan produces an image in polar coordinates rather than cartesian coordinates.

The resulting images obtained from these algorithms are plotted in fig 6. From these plots, it appears as though the sector scan (fig 6c) is the least well able to distinguish the individual point scatterers. The back wall is captured fairly well, as a single high-intensity point which decays horizontally. From this image, a “smearing” of scatterers in the azimuthal θ-direction is apparent. This is believed to be due to the fact that all 64 elements were used to sweep through the sector: as the scatterers are close to the array, the angle at which the scatterer will be viewed by one element will be different for another. An avenue of further work exists in exploring how to resolve point scatterers more effectively with a sector scan; the work described in this report focusses on the plane and focussed scans, and particularly on the TFM.

In each of the plane, focussed and TFM scans (figs 6a, 6b and 6d), all scatterers but the two located at approximate coordinates $(0\text{mm}, 10\text{mm})$ are individually resolved for this initial image. The two scatterers located at $(0\text{mm}, 12.5\text{mm})$ and $(0\text{mm}, 15\text{mm})$ are well resolved in all of the images, indicating that these algorithms are quite sensitive in the z-direction despite the length of the focus region displayed in fig 4. The TFM image appears to resolve all scatterers best, followed by the focussed image, and finally the plane scan. In the plane and focussed images, it is clear that as scatterer distance from the array increases, the ability to

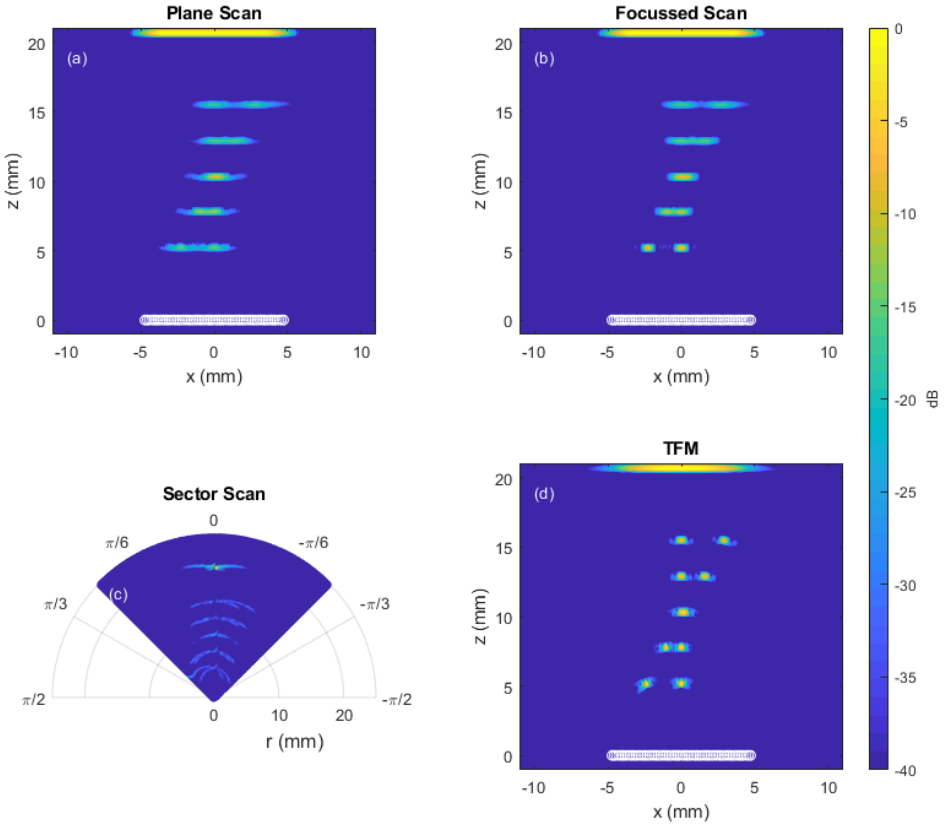

Fig. 6 Intensity plots produced from imaging algorithms defined in equations 5-8. All images produced from the same simulated FMC data set described above.

focus individual scatterers decreases in the x-axis: scatterers with equal z position become harder to distinguish for increasing z. In the TFM image, the point-like shape of scatterers remains roughly the same throughout the full depth of the image. Additionally, all three of these images resolve the back wall of the water tank fairly well for the full width of the array.

In order to fully resolve all ten point scatterers, the TFM was used due to its ability to focus scatterers well regardless of depth into the tank. Attention was concentrated on the two scatterers at approximate coordinates $(0\text{mm}, 10\text{mm})$. Previously, it was predicted from the beam profile modelling that by increasing the overall length of the array and increasing the frequency of the transducer, a higher resolution of the medium could be obtained. For a particular frequency, it has been discussed that for $p < 0.5\lambda$ grating lobes are suppressed, and thus the largest "allowed" pitch for a particular frequency is $p = 0.5\,{}^{c}/_{f}$. The strength of this inequality was examined for the array design proposed by increasing probe pitch above this limit in fig 7.

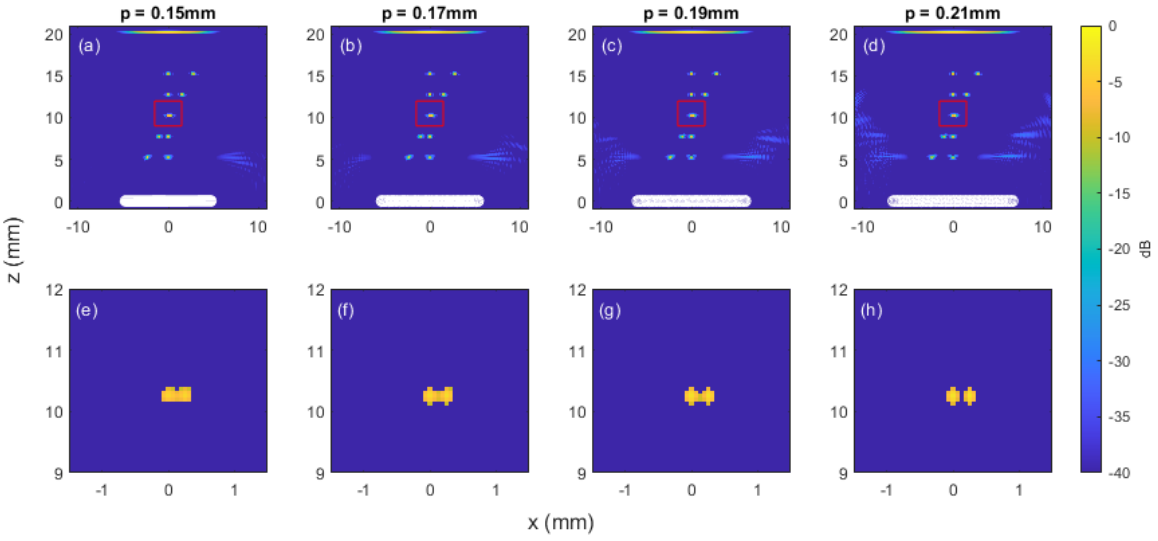

Fig. 7 Intensity of TFM images when a 5MHz probe is used for increasing element pitch. Figs 7e-h are zoomed in on the region indicated in the subfigures above, to indicate how well adjacent scatterers are resolved. Note that in these zoomed images, a filter was applied at -10dB such that only signals greater than this threshold were plotted.

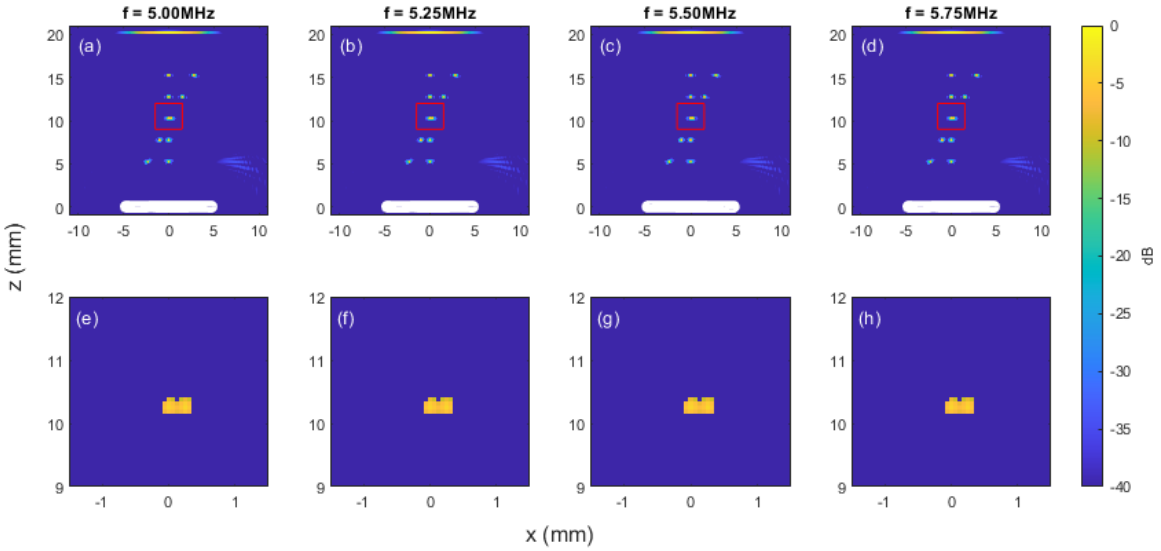

Fig. 8 Intensity of TFM images when a 0.15mm pitch probe is used for increasing frequency. Figs 8e-h are zoomed in on the region indicated in the subfigures above, to indicate how well adjacent scatterers are resolved. Note that in these zoomed images, a filter was applied at -10dB such that only signals greater than this threshold were plotted.

It can be seen in fig 7 that as pitch is increased, the presence of grating lobes increases, however the two adjacent scatterers are more easily distinguished. In this example, the presence of grating lobes may be considered as acceptable because they do not coincide with any scatterer locations, and their amplitude is very small compared to the scatterers. In a more general case, this may result in artefacts in the image which are interpreted as defects, or which obscure defects which are present. Despite this, it is the only array design tested which has been able to distinguish all individual defects to a -10dB level.

Frequency was also increased for a constant pitch of 0.15mm to study the presence of grating lobes: the TFM images are plotted in fig 8. It can be seen that over the frequency range examined, grating lobes impacted on the TFM images less than in fig 7 as pitch was increased, however adjacent scatterers were not resolved as well for the larger frequency in fig 8h than the large pitch in fig 7h. To suppress these grating lobes, the pitch could be set equal to the limit discussed above. Commonly used phased arrays in research contexts have pitch varying between 0.15mm-0.6mm [2, 3], thus the ability to reduce pitch as frequency increases might be difficult to achieve in practice.

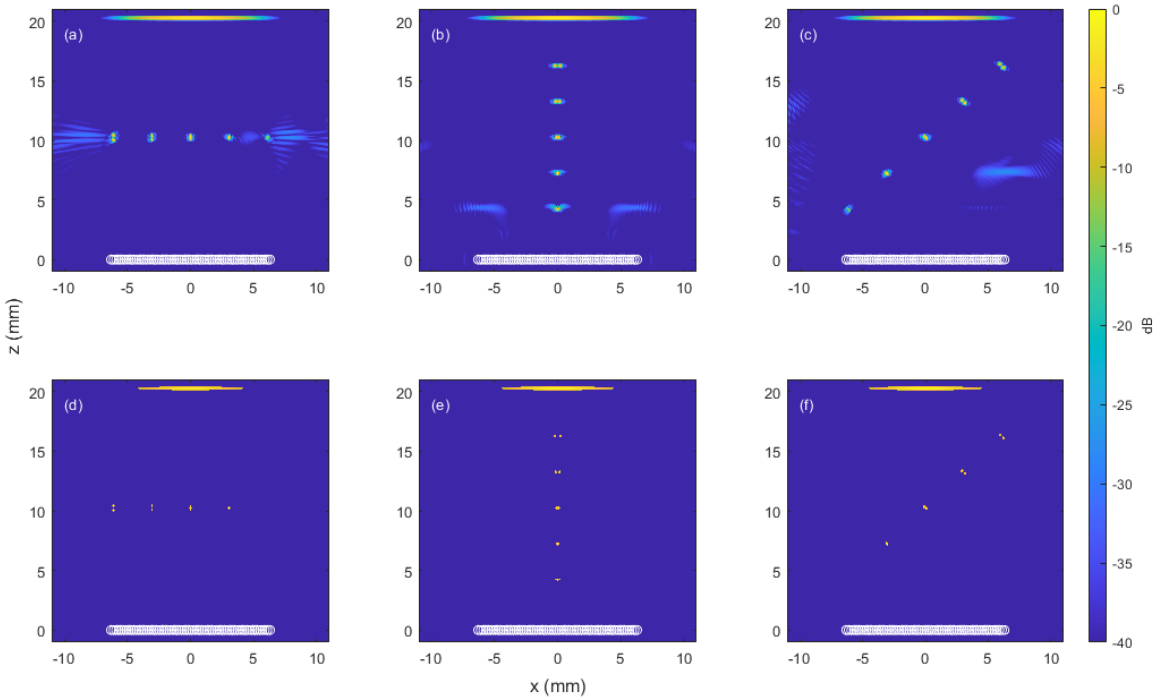

Fig. 9 Intensity of TFM images with scatterers separated by distances $0.05\text{mm}, 0.10\text{mm}, 0.20\text{mm}, 0.30\text{mm}, 0.40\text{mm}$. Fig 9a shows decreasing separation in the x-axis, fig 9b shows increasing separation in the z-axis, and fig 9c shows increasing separation in the increasing diagonal. Figs 9d-f shows corresponding plot with filter -10dB, to determine whether array is able to distinguish between adjacent scatterers.

The best image obtained used a 64-element array with pitch 0.20mm and centre-frequency 5MHz. These parameters are within the range of arrays used in research contexts, however the TFM image of the calibration sample did contain some grating lobes. To examine how well this array was able to resolve adjacent scatterers, a set of samples were examined with point scatterers separated by distances $0.05\text{mm}, 0.10\text{mm}, 0.20\text{mm}, 0.30\text{mm}, 0.40\text{mm}$. These are plotted in fig 9, with scatterers arranged in horizontally, vertically and diagonally to examine the array's performance in different directions. In all of these plots, grating lobes are present, but never coincide with a scatterer such that its location is obscured. It is also noteworthy that the scatterers separated by a distance of 0.05mm has maximum signal $< -10\text{dB}$ in the horizontal and diagonal arrangement, as they do not appear in the filtered plots: while initially unexpected (adjacent scatterers might be expected to interfere constructively) this likely arises from a destructive interference between the scatterers. In the horizontal arrangement in fig 9d, the array is unable to resolve scatterers closer than 0.30mm, whereas in the vertical and diagonal arrangement scatterers closer than 0.20mm cannot be resolved separately. In other words, this array is better able to resolve scatterers separated in the x-direction than in the z-direction, matching the shape of the focus region in the beam profile scans in fig 4.

CONCLUSION

The performance of ultrasonic phased arrays was examined by varying the total number of elements, element pitch and transducer frequency using a Huygens' model for the beam profile of the array. It was determined that the focussing ability of the array is largely controlled by the frequency, as well as the overall length of the array: it was found that increasing frequency or increasing array length would reduce the focal area to tighten the focus and increase the overall resolution of the image processed. In particular, good agreement was found with the threshold for pitch presented in the literature [2], namely that grating lobes are suppressed for pitch $p < 0.5\lambda$.

An array was designed such that a calibration sample consisting of a set of point scatterers located in a water tank could be scanned, and each scatterer identified independently of the others. After a small amount of optimisation, a 5MHz, 64-element array with pitch 0.20mm and 0.05mm separation between elements was found to be capable of distinguishing all scatterers in the sample. The smallest separation between scatterers in the sample was 0.25mm in the x-direction. On further examination, it was found that this array was unable to resolve scatterers to this level in the z-direction, but was capable of distinguishing up to a 0.30mm separation in this direction.

REFERENCES

1. Wilcox, P., "Ultrasonic Arrays I," *Ultrasonic NDT*, 31 Mar 2021, University of Bristol. Lecture.
2. Holmes, C., Drinkwater, B. W. and Wilcox, P., "Post-processing of the full matrix of ultrasonic transmit-receive array data for non-destructive evaluation," *NDT&E Intl.*, **38**, pp. 701-711, 2005.
3. Budyn, N. S., "Imaging and defect characterisation using multi-view ultrasonic data in nondestructive evaluation," EngD thesis, University of Bristol, pp. 23-40, 2020 .
4. Velichko, A., "The Frequency Domain," *Ultrasonic NDT*, 29 Mar 2021, University of Bristol. Lecture.
5. Pompei, F. and Wooh, S-C., "Phased array element shapes for suppressing grating lobes," *J. Acoust. Soc. Am.* **111**(5), pp.2040-2048, 2002.

APPENDIX A: MATLAB CODE (LINEAR BEAM PROFILE)

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

%% f vs n study for constant array length

%% Plotting parameters
n = 3;
m = 3;
subfig_label = 'abcdefghijklmnopqrstuvwxyz';
kk = 1;
t = tiledlayout(m,n);

%% Inputs
centre_frequency = [1e6, 2e6, 5e6];
el_pitch = [.1875e-3, .375e-3, .75e-3];
el_separation = .05e-3;
num_els = [64, 32, 16];
velocity_L = 1500;
backwall_distance = 25e-3;
grid_pts = 351;

focal_pt_x = 0.0e-3;
focal_pt_z = 15.00e-3;

p = zeros(n*m, grid_pts, grid_pts);

% Generate grid.
x = linspace(-backwall_distance/2, backwall_distance/2, grid_pts);
z = linspace(0, backwall_distance, grid_pts);
dx = x(2) - x(1);
dz = z(2) - z(1);
[X, Z] = meshgrid(x, z);

% Get focal_pt indices
focal_pt_ii = round(focal_pt_x / dx) + round(grid_pts / 2);
focal_pt_kk = round(focal_pt_z / dz);

for ii = 1:n
    for jj = 1:m

        %% Parameters
        wavelength = velocity_L / centre_frequency(ii);
        k = 2 * pi / wavelength;
        omega = 2 * pi * centre_frequency(ii);
        el_width = el_pitch(jj) - el_separation;

        % Mid-coordinates of each element.
        source_x_positions = linspace(0, el_pitch(jj)*(num_els(jj)-1),
num_els(jj));
        source_x_positions = source_x_positions - mean(source_x_positions);

        % Adjust X, Z, source_x_positions for vectorisation.
        X_ = repmat(X, 1, 1, length(source_x_positions));
        Z_ = repmat(Z, 1, 1, length(source_x_positions));
        source_positions = reshape(source_x_positions, 1, 1,
length(source_x_positions));

        %% Beam Profile of linear array
```

```
        % Calculate distance from source to point.
        r_j = sqrt((X_ - source_positions).^2 + (Z_ - .01e-3).^2);
        % Calculate angles made from element to pixel location.
        phi = acos(Z_ ./ r_j);
        directivity_f = el_width * sinc(k * el_width / (2 * pi) * sin(phi));
        % Array time delays. Get the list of distances to this point for all
els.
        d_j = squeeze(r_j(focal_pt_kk, focal_pt_ii, :));
        t_j = (d_j - d_j(round(num_els(jj)/2))) / velocity_L;
        % Weight-and-phase delay.
        B_j = repmat(reshape((exp(- 1i * omega * t_j)), 1, 1, num_els(jj)),
length(z), length(x), 1);

        % Calculate pressure field. Sum over elements.
        p(kk, :, :) = sum( ...
            1./sqrt(r_j) .* exp(1i*(k * r_j - omega * 0)) .* directivity_f .*
B_j, ...
            3 ...
        );
        kk = kk + 1;
    end
end

kk = 1;

for ii = 1:n
    for jj = 1:m
        % Get aspect ratios. Note: this method only works when focal_pt_x
        % = 0, and will likely change for different focal_pt_z. Keep z
        % constant to make comparison valid.
        this_p = abs(reshape(p(kk, 11:end, :), grid_pts-10, grid_pts));
        max_ = max(this_p, [], 'all');
        dB_p = 20 * log10(this_p ./ max_);
        is_focus = logical(dB_p > -3);

        is_x = x(any(is_focus, 1))*10^3;
        is_z = z(any(is_focus, 2))*10^3;

        aspect_ratio = (is_x(end) - is_x(1)) / (is_z(end) - is_z(1));

        source_x_positions = linspace(0, el_pitch(jj)*(num_els(jj)-1),
num_els(jj));
        source_x_positions = (source_x_positions - mean(source_x_positions)) *
10^3;
        nexttile;
        hold on
        box on
        imagesc(x*10^3, z*10^3, abs(reshape(p(kk, :, :), grid_pts, grid_pts)))
        scatter(source_x_positions, zeros(num_els(jj),1), 'wo')
        scatter(focal_pt_x*10^3, focal_pt_z*10^3, 'r.')
        title(sprintf('f = %2.1fMHz\np = %3.2fmm, n = %d',
centre_frequency(ii)*10^-6, el_pitch(jj)*10^3, num_els(jj)))
        text(-12.5, 21, sprintf('(%s) λ = %2.1fmm\nAR = %3.2f',
subfig_label(kk), 10^3*velocity_L / centre_frequency(ii), aspect_ratio),
'Color', 'white')
        xlim([x(1)*10^3, x(end)*10^3])
        ylim([z(1)*10^3, z(end)*10^3])
        kk = kk + 1;
    end
end
xlabel(t, 'x (mm)')
ylabel(t, 'z (mm)')
```

```
%% Beam profile for calibration sample

%% Plotting parameters
n = 5;
m = 2;
subfig_label = 'abcdefghijklmnopqrstuvwxyz';
kk = 1;
figure(2)
t = tiledlayout(m,n);

%% Inputs
centre_frequency = 5e6;
el_pitch = .15e-3;
el_separation = .05e-3;
num_els = 64;
velocity_L = 1500;
backwall_distance = 20e-3;
grid_pts = 351;

focal_pt_x = [-2.25e-3, ...
              -1.00e-3, ...
               0.25e-3, ...
               1.50e-3, ...
               2.75e-3, ...
               0.0, ...
               0.0, ...
               0.0, ...
               0.0, ...
               0.0];
focal_pt_z = [ 5.00e-3, ...
               7.50e-3, ...
              10.00e-3, ...
              12.50e-3, ...
              15.00e-3, ...
              5.00e-3, ...
               7.50e-3, ...
              10.00e-3, ...
              12.50e-3, ...
              15.00e-3];

p = zeros(n*m, grid_pts, grid_pts);

for ii = 1:n*m

    %% Parameters
    wavelength = velocity_L / centre_frequency;
    k = 2 * pi / wavelength;
    omega = 2 * pi * centre_frequency;
    el_width = el_pitch - el_separation;
    % Generate grid.
    x = linspace(-backwall_distance/2, backwall_distance/2, grid_pts);
    z = linspace(0, backwall_distance, grid_pts);
    dx = x(2) - x(1);
    dz = z(2) - z(1);
    [X, Z] = meshgrid(x, z);

    % Get focal_pt indices
    focal_pt_ii = round(focal_pt_x(ii) / dx) + round(grid_pts / 2);
    focal_pt_kk = round(focal_pt_z(ii) / dz);
```

```
    % Mid-coordinates of each element.
    source_x_positions = linspace(0, el_pitch*(num_els-1), num_els);
    source_x_positions = source_x_positions - mean(source_x_positions);

    % Adjust X, Z, source_x_positions for vectorisation.
    X = repmat(X, 1, 1, length(source_x_positions));
    Z = repmat(Z, 1, 1, length(source_x_positions));
    source_positions = reshape(source_x_positions, 1, 1,
length(source_x_positions));

    %% Beam Profile of linear array

    % Calculate distance from source to point.
    r_j = sqrt((X - source_positions).^2 + (Z - .01e-3).^2);
    % Calculate angles made from element to pixel location.
    phi = acos(Z ./ r_j);
    directivity_f = el_width * sinc(k * el_width / (2 * pi) * sin(phi));
    % Array time delays. Get the list of distances to this point for all els.
    d_j = squeeze(r_j(focal_pt_kk, focal_pt_ii, :));
    t_j = (d_j - d_j(round(num_els/2))) / velocity_L;
    % Weight-and-phase delay.
    B_j = repmat(reshape((exp(- 1i * omega * t_j)), 1, 1, num_els), length(z),
length(x), 1);

    % Calculate pressure field. Sum over elements.
    p(ii, :, :) = sum( ...
        1./sqrt(r_j) .* exp(1i*(k * r_j - omega * 0)) .* directivity_f .* B_j,
...
        3 ...
    );

end

for ii = 1:length(focal_pt_x)
    nexttile;
    hold on
    box on
    imagesc(x*10^3, z*10^3, abs(reshape(p(ii, :, :), grid_pts, grid_pts)),
[min(abs(p), [], 'all'), max(abs(p), [], 'all')])
    scatter(source_x_positions*10^3, zeros(num_els,1), 'wo')
    scatter(focal_pt_x([1:length(focal_pt_x)]~=ii)*10^3,
focal_pt_z([1:length(focal_pt_x)]~=ii)*10^3, 'ro')
    scatter(focal_pt_x([1:length(focal_pt_x)]==ii)*10^3,
focal_pt_z([1:length(focal_pt_x)]==ii)*10^3, 'bo')
    title(sprintf('x = %3.2fmm\nz = %3.2fmm', focal_pt_x(ii)*10^3,
focal_pt_z(ii)*10^3))
    text(10^3*focal_pt_x(ii) - 2.8, 10^3*focal_pt_z(ii) + 2.7, sprintf('(%s)',
subfig_label(ii)), 'Color', 'white')
    xlim([10^3*focal_pt_x(ii) - 3, 10^3*focal_pt_x(ii) + 3])
    ylim([10^3*focal_pt_z(ii) - 3, 10^3*focal_pt_z(ii) + 3])
end

t.XLabel.String = 'x (mm)';
t.YLabel.String = 'z (mm)';
t.TileSpacing = 'tight';
cb = colorbar;
cb.Layout.Tile = 'east';
```

APPENDIX B: MATLAB CODE (TIME-TRACE SIMULATION AND IMAGING ALGORITHMS)

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

%% Inputs

num_els = 64;
centre_freq = 5e6;
el_pitch = .15e-3;
el_sep = .05e-3;
v_L = 1500.0;
backwall_dist = 20.0e-3;
grid_pts = 251;
imaging_aperture = el_pitch * num_els / 4;
scat_coords = [[-2.25e-3,  5.00e-3]; ...
               [-1.00e-3,  7.50e-3]; ...
               [ 0.25e-3, 10.00e-3]; ...
               [ 1.50e-3, 12.50e-3]; ...
               [ 2.75e-3, 15.00e-3]; ...
               [ 0.00e-3,  5.00e-3]; ...
               [ 0.00e-3,  7.50e-3]; ...
               [ 0.00e-3, 10.00e-3]; ...
               [ 0.00e-3, 12.50e-3]; ...
               [ 0.00e-3, 15.00e-3]];

% Miscellaneous extra params.
num_cycles = 5;
oversample = 10; % How often per cycle the signal is sampled.

%% Parameters

lambda = v_L / centre_freq;

probe_coords = zeros(num_els, 2);
probe_coords(:, 1) = linspace(0, el_pitch*(num_els-1), num_els);
probe_coords(:, 1) = probe_coords(:, 1) - mean(probe_coords(:, 1));

[tx, rx] = meshgrid([1:num_els]);
tx = reshape(tx, length(tx)^2, 1);
rx = reshape(rx, length(rx)^2, 1);

%% Get input time signal and freq spectrum

max_time = 2.1 * sqrt(2 * backwall_dist^2) / v_L;
dt = 1 / centre_freq / oversample;
time = [0 : dt : max_time];

half_pulse = 5 / (2 * centre_freq);

input_signal = sin(2 * pi * centre_freq * time)' .* ...
               fn_hanning(length(time), half_pulse / max_time, half_pulse /
max_time);

% Bring peak of input pulse to time = 0.
time = time - half_pulse;



fft_pts = 2^nextpow2(length(time));
```

```
df = 1 / max_time;
freq = [0 : df : df*(fft_pts/2-1)];
omega = 2 * pi * freq;

input_spectrum = fft(input_signal, fft_pts);
input_spectrum = input_spectrum(1:fft_pts/2);

%% Do ray tracing

scat_dists_tx = sqrt((probe_coords(tx, 1) - scat_coords(:, 1)').^2 + ...
                     (probe_coords(tx, 2) - scat_coords(:, 2)').^2);
scat_dists_rx = sqrt((probe_coords(rx, 1) - scat_coords(:, 1)').^2 + ...
                     (probe_coords(rx, 2) - scat_coords(:, 2)').^2);
scat_times = (scat_dists_tx + scat_dists_rx) / v_L;

bw_dists = (sqrt((probe_coords(tx, 1)' - probe_coords(rx, 1)').^2 + ...
                 (2*(backwall_dist - probe_coords(tx, 2)')).^2));
bw_times = bw_dists / v_L;

% Get angles for directivity.
scat_theta_tx = acos((scat_coords(:, 2)' - probe_coords(tx, 2)) ./ ...
                     sqrt((probe_coords(tx, 1) - scat_coords(:, 1)').^2 + ...
                          (probe_coords(tx, 2) - scat_coords(:, 2)').^2));
scat_theta_rx = acos((scat_coords(:, 2)' - probe_coords(rx, 2)) ./ ...
                     sqrt((probe_coords(tx, 1) - scat_coords(:, 1)').^2 + ...
                          (probe_coords(tx, 2) - scat_coords(:, 2)').^2));

bw_theta_tx = acos((2*(backwall_dist - probe_coords(tx, 2)')) ./ bw_dists);
bw_theta_rx = acos((2*(backwall_dist - probe_coords(rx, 2)')) ./ bw_dists);

%% Propagate signal

% Get scatterer and backwall signal amplitudes.

% Directivity includes both transmit and receive paths. N.B. this function
% takes into account that in Matlab, sinc(x) := sin(πx)/( πx)
scat_dir = (el_pitch - el_sep)^2 * ...
           sinc((el_pitch - el_sep) / lambda * sin(scat_theta_tx)) .* ...
           sinc((el_pitch - el_sep) / lambda * sin(scat_theta_rx));
% Amplitude includes scattering, directivity and beam spreading.
scat_amp = 0.01 * scat_dir ./ (sqrt(scat_dists_tx) .* sqrt(scat_dists_rx));

bw_dir = (el_pitch - el_sep)^2 * ...
         sinc((el_pitch - el_sep) / lambda * sin(bw_theta_tx)) .* ...
         sinc((el_pitch - el_sep) / lambda * sin(bw_theta_rx));
bw_amp = bw_dir ./ sqrt(bw_dists); % Perfect reflector => R = 1.



output_spectra = 0;
% Scatterers
for ss = 1:size(scat_coords, 1)
    output_spectra = output_spectra + ( ...
        spdiags(input_spectrum, 0, length(freq), length(freq)) * ...
        exp(-1i * omega' * scat_times(:, ss)') * ...
        spdiags(scat_amp(:, ss), 0, length(tx), length(tx)) ...
    );
end

% Backwall
```

```
output_spectra = output_spectra + ( ...
    spdiags(input_spectrum, 0, length(freq), length(freq)) * ...
    exp(-1i * omega' * bw_times(:)') * ...
    spdiags(bw_amp(:), 0, length(tx), length(tx)) ...
);

FMC_data = ifft(output_spectra, fft_pts, 1);
FMC_data = FMC_data(1:length(time), :);
FMC_dt = 1 / (fft_pts * abs(freq(2) - freq(1)));
FMC_time = [0 : FMC_dt : FMC_dt * (length(time)-1)];

% End of simulation of wave.

%% Imaging - TFM

x = linspace(- backwall_dist / 2 - 1e-3, backwall_dist / 2 + 1e-3, grid_pts);
z = linspace(-1e-3, backwall_dist + 1e-3, grid_pts);
[X, Z] = meshgrid(x, z);

%% Plane

Plane_lookup_times = 2 * Z(:)' / v_L;
% Work out if each point is below the aperture for each tr pair
is_in_aperture = logical( ...
    abs(probe_coords(:, 1) - X(:)') <= imaging_aperture / 2 ...
);

Im_Plane = 0;
tr = 1;
for ti = 1:num_els
    for ri = 1:num_els
        % Skip all tr pairs where there are no contributions.
        if and(any(is_in_aperture(ti, :)), any(is_in_aperture(ri, :)))
            Im_Plane = Im_Plane + reshape( ...
    ... % Only contribute if this point in the image is below both tx and rx.
                is_in_aperture(ti, :) .* is_in_aperture(ri, :) .* ...
    ... % Interpolate to find the data.
                interp1(FMC_time, FMC_data(:, tr), Plane_lookup_times, 'linear',
0), ...
                size(X, 1), size(X, 2) ...
            );
        end
        tr = tr + 1;
    end
end

max_ = max(abs(Im_Plane), [], 'all');
dB_Plane = 20 * log10(abs(Im_Plane) ./ max_);

%% Focussed

Focus_lookup_times = (sqrt((probe_coords(tx, 1) - X(:)').^2 + ...
                           (probe_coords(tx, 2) - Z(:)').^2) + ...
                      sqrt((probe_coords(rx, 1) - X(:)').^2 + ...
                           (probe_coords(rx, 2) - Z(:)').^2) ...
    ) / v_L;

Im_Focus = 0;
tr = 1;
for ti = 1:num_els
    for ri = 1:num_els
        if and(any(is_in_aperture(ti, :)), any(is_in_aperture(ri, :)))
```

```
            Im_Focus = Im_Focus + reshape( ...
                is_in_aperture(ti, :) .* is_in_aperture(ri, :) .* ...
                interp1(FMC_time, FMC_data(:, tr), Focus_lookup_times(tr, :),
'linear', 0), ...
                size(X, 1), size(X, 2));
        end
        tr = tr + 1;
    end
end

max_ = max(abs(Im_Focus), [], 'all');
dB_Focus = 20 * log10(abs(Im_Focus) ./ max_);

%% Sector

r = linspace(0, 1.1 * sqrt(5)/2 * backwall_dist, grid_pts);
theta = linspace(-pi/4, pi/4, grid_pts);
[R, Theta] = meshgrid(r, theta);

Sector_lookup_times = ( ...
    2*R(:)' + probe_coords(tx, 1) * sin(Theta(:)') + probe_coords(rx, 1) *
sin(Theta(:)') ...
) / v_L;
tau_Sector = reshape(Sector_lookup_times, num_els^2, size(R, 1), size(R, 2));

Im_Sector = 0;
for tr = 1:num_els^2
    Im_Sector = Im_Sector + ...
        interp1(FMC_time, FMC_data(:, tr), squeeze(tau_Sector(tr, :, :)),
'linear', 0);
end

max_ = max(abs(Im_Sector), [], 'all');
dB_Sector = 20 * log10(abs(Im_Sector) ./ max_);
dB_Sector(dB_Sector < -40) = -40;

%% TFM

% TFM lookup times are identical to focussed lookup times. Difference is
% that we focus everywhere, instead of just below the aperture.

Im_TFM = 0;
tau_TFM = reshape(Focus_lookup_times, num_els^2, size(X, 1), size(X, 2));
for tr = 1:num_els^2
    Im_TFM = Im_TFM + ...
        interp1(FMC_time, FMC_data(:, tr), squeeze(tau_TFM(tr, :, :)), 'linear',
0);
end

max_ = max(abs(Im_TFM), [], 'all');
dB_TFM = 20 * log10(abs(Im_TFM) ./ max_);

%% Plotting

figure(1)
subplot(1,2,1)
xlim([min(x)*10^3, max(x)*10^3])
ylim([min(z)*10^3, max(z)*10^3])
hold on
plot([-backwall_dist*10^3/2, backwall_dist*10^3/2], [backwall_dist*10^3,
backwall_dist*10^3], 'b')
```

```
scatter(probe_coords(:, 1)*10^3, probe_coords(:, 2)*10^3, 'ko')
scatter(scat_coords(:, 1)*10^3, scat_coords(:, 2)*10^3, 'r.')
xlabel('x (mm)')
ylabel('y (mm)')
box on
text(-10, 19, '(a)')
subplot(1,2,2)
imagesc(time*10^6, [1:num_els^2], abs(FMC_data'))
xlabel('Time (μs)')
ylabel('Tx-Rx pair')
box on
text(1, 120, '(b)', 'Color', 'w')
cb = colorbar();
cb.Label.String = 'Signal';

figure(2)
tiledlayout(2, 2)
nexttile;
imagesc(x*10^3, z*10^3, dB_Plane, [-40, 0])
ax = gca();
set(ax, 'YDir', 'normal')
hold on
scatter(probe_coords(:,1)*10^3, probe_coords(:,2)*10^3, 'wo')
xlabel('x (mm)')
ylabel('z (mm)')
text(-10, 19, '(a)', 'Color', 'w')
title('Plane Scan')

nexttile;
imagesc(x*10^3, z*10^3, dB_Focus, [-40, 0])
ax = gca();
set(ax, 'YDir', 'normal')
hold on
scatter(probe_coords(:,1)*10^3, probe_coords(:,2)*10^3, 'wo')
xlabel('x (mm)')
ylabel('z (mm)')
text(-10, 19, '(b)', 'Color', 'w')
title('Focussed Scan')

nexttile;
dBScaled = round((dB_Sector(:)+40));
map = colormap(parula(41));
polarscatter(Theta(:), R(:)*10^3, [], map(dBScaled+1,:), 'filled')
thetalim([-90, 90])
ax = gca;
ax.ThetaAxisUnits = 'radians';
ax.ThetaZeroLocation = 'top';
ax.RAxis.Label.String = 'r (mm)';
text(pi/4, 23, '(c)', 'Color', 'w')
title('Sector Scan')

nexttile;
imagesc(x*10^3, z*10^3, dB_TFM, [-40, 0])
ax = gca();
set(ax, 'YDir', 'normal')
hold on
scatter(probe_coords(:,1)*10^3, probe_coords(:,2)*10^3, 'wo')
xlabel('x (mm)')
ylabel('z (mm)')
text(-10, 19, '(d)', 'Color', 'w')
title('TFM')

cb = colorbar();
```

```
cb.Label.String = 'dB';
cb.Layout.Tile = 'east';
```