Exercise 3: Data Processing

Matthew Chandler

ABSTRACT

Several experimental data sets were examined and processed using MATLAB v9.9.0 (R2020b). The frequency-dependent attenuation of a Perspex block was measured as a monotonously increasing function; the frequency-dependent reflection coefficient of an adhesive joint connecting two aluminium plates in immersion was measured; and an automated thickness measurement of a steel bearing casing was developed, finding the thickness to be 47.6mm at transducer position 125mm, with an overall maximum thickness of 55mm.

INTRODUCTION AND BACKGROUND

Ultrasonic Non-Destructive Testing (NDT) provides a means of examining the internal structure of some object by channelling ultrasonic waves into the inspection medium. The application of interest for the work outlined here is the measurement of properties of the medium. These can be determined entirely from the ultrasonic signal obtained from a transducer after the waves have propagated through the structure.

As the ultrasonic signal travels through the medium, its propagation through and interaction with the structure can be modelled with a transfer function $H(\omega)$ using the linear system approach [1]. The transfer function describes how the wave changes as it propagates, defined as a product of terms which act on the frequency-domain spectrum (ω) of the signal, as opposed to the time-domain signal (t). It is typically defined as

$$H(\omega) = T_x(\omega)A(\omega)BX(\omega)\Delta(\omega)R_x(\omega) \tag{1}$$

where T_x, R_x describe the transmitter and receiver transducer characteristics (including effects like directivity, transducer frequency response); Δ describes the time delay of the signal due to propagation; $B \propto \sqrt{r}^{-1}$ is the beam spread of the wave; X is a coefficient which describes the change in amplitude of the wave from reflection from and transmission through boundaries; and A is the attenuation.

The reflection and transmission coefficient X is the product of the individual reflection and transmission coefficients R, T respectively, which can be determined by considering the amplitude and pressure of the wave at a boundary. As both are continuous, these conditions can be combined to define the coefficients as

$$R \equiv \frac{A_r}{A_i} = \frac{z_1 - z_2}{z_1 + z_2} \tag{2a}$$

$$T \equiv \frac{A_t}{A_i} = \frac{2z_1}{z_1 + x_2} \tag{2b}$$

where A_i, A_r, A_t are the incident, reflected and transmitted amplitudes respectively, and z_j is the acoustic impedance of the jth medium defined as

$$z = {}^{p}/_{\dot{u}} = \rho c \tag{3}$$

where p is acoustic pressure, $\dot{u}$ is particle velocity, ρ is the density of the medium and c is the wave velocity.

Attenuation is a measure of energy loss through a system, usually due to scattering and absorption [2]. Scattering results from inhomogeneity of the medium, usually from grain boundaries separating areas of different density or wave velocity. Absorption arises from the conversion of mechanical sound energy to heat. The resulting loss of energy with propagation distance from both effects is defined by the attenuation coefficient, which is well described by an exponential decay with distance d [1]:

$$A(\omega) = e^{-\alpha(\omega)d} \tag{4}$$

where the attenuation coefficient $\alpha(\omega)$ is dependent on frequency.

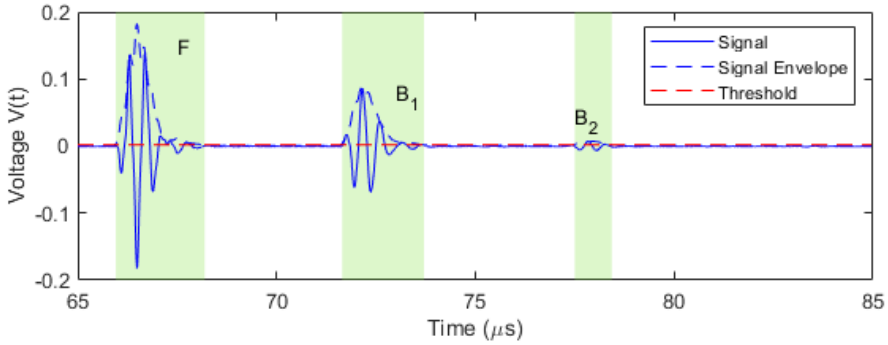

Fig. 1 Time-trace obtained from a pulse-echo immersion test on a Perspex plate. Response F is the reflection from the front of the plate, and $\mathrm{B}_1, \mathrm{B}_2$ are subsequent reverberations through the thickness, reflecting from the back of the plate. The envelope of the signal is shown as a dashed blue line, with the threshold shown in red. Independent signals obtained from thresholding are highlighted in green.

FREQUENCY-DEPENDENT ATTENUATION OF PERSPEX

A time-domain signal was obtained from a pulse-echo immersion test on a 7.8mm Perspex plate using a $2.5\mathrm{MHz}$ transducer. The raw time-trace is shown in fig 1.

The response from each reflection was isolated such that they could be processed independently. Each section of the data with voltage greater than a threshold ($0.01\times$ maximum $|V(t)|$) was treated as an individual response: these are highlighted in green in fig 1. Using the transfer function from equation 1, the spectra of these signals was approximated by considering the reflection and transmission of the wave through the geometry:

$$F(\omega) = I(\omega)\, R_{12} \tag{5a}$$

$$B_1(\omega) = I(\omega)\, T_{12} R_{21} T_{21} e^{-2\alpha(\omega)d} \tag{5b}$$

$$B_2(\omega) = I(\omega)\, T_{12} R_{21}^3 T_{21} e^{-4\alpha(\omega)d} \tag{5c}$$

These frequency spectra were calculated using a Fast Fourier Transform, and are plotted in fig 2. By dividing the spectra of F, B_1, the attenuation coefficient α can be obtained with equation

$$\Rightarrow \alpha(\omega) = -\frac{1}{2d} \ln \left| \frac{B_1(\omega)}{F(\omega)} \frac{R_{12}}{T_{12} R_{21} T_{21}} \right| \tag{6}$$

This is plotted in fig 3, where transmission and reflection coefficients were calculated using equation 2, assuming that the plate was immersed in water.

Of particular interest here is that the attenuation calculated from the F and B_1 responses (blue circles) is close to a monotonously increasing function [3, 4]. There are some local extreme points at $f \approx 0.5, 4\ \mathrm{MHz}$, contradicting this behaviour. As these points are close to the edges, this suggests that the threshold used on the frequency spectra may not have been large enough, with the error on these amplitudes contributing a more significant deviation from the true value. This is further supported by the fact that the attenuation plot calculated from the $\mathrm{B}_1, \mathrm{B}_2$ spectra (orange circles) and the F, B_2 spectra (yellow circles) are not monotonously

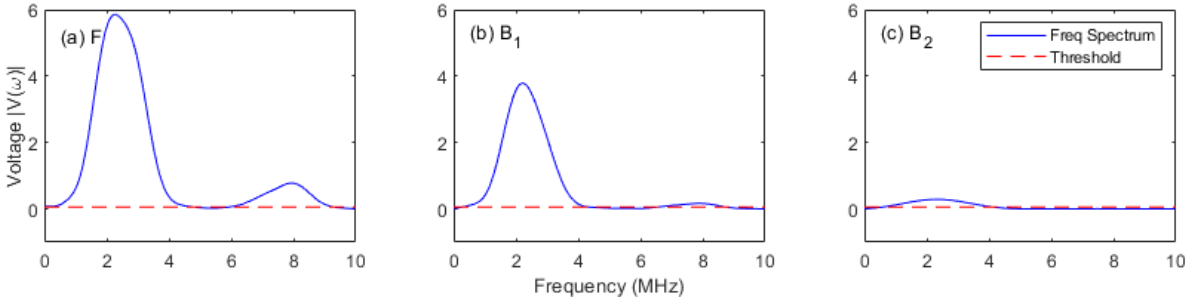

Fig. 2 Fourier transforms of voltage for each signal in fig 1. Note that only the voltage greater than a threshold ($0.1\times$ maximum $|\mathrm{B}_2|$ value) has been plotted, such that any frequencies with negligible amplitude are excluded from subsequent calculation.

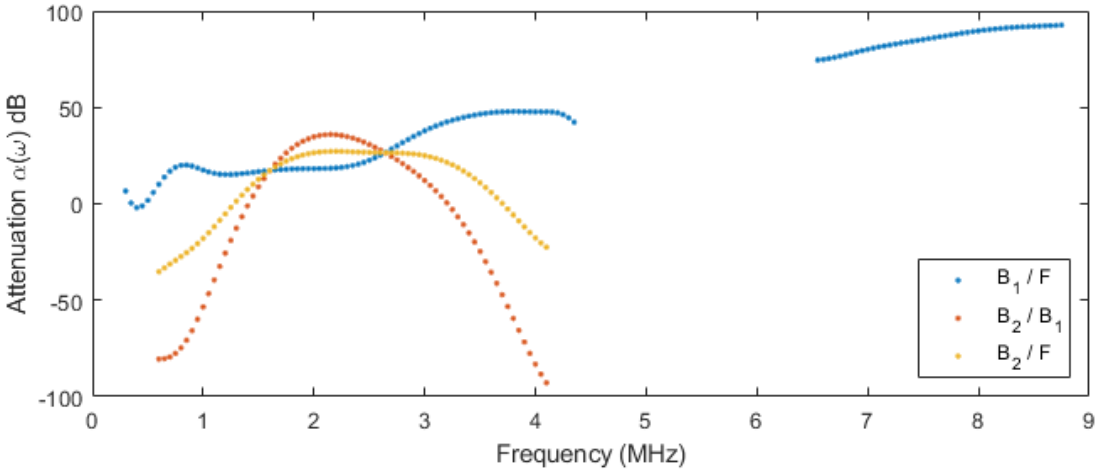

Fig. 3 Attenuation coefficient calculated from fig 2 and equation 6. Note that there is no data available for some frequencies (particularly for the 4-6MHz range in the $^{B_1}/_{F}$ data): this is because amplitude of one or both of these spectra falls below the threshold plotted in fig 2, and thus are excluded from this calculation.

increasing, and instead show single large maxima points. As the B_2 spectrum has significantly lower peak amplitude than the other spectra, it is likely that only the amplitude near the peak frequency is valid, or at $f \approx 2.25\text{MHz}$. At this point, the attenuation from the $^{B_2}/_{F}$ and $^{B_2}/_{B_1}$ spectra are very similar to the $^{B_1}/_{F}$ attenuation, further indicating that the amplitude threshold used was too low.

This analysis has assumed that the only difference between the F, B_1 and B_2 responses is the reflection coefficient and attenuation. The effect of beam spreading is not taken into account. The beam spreading term B in the transfer function (equation 1) is proportional to $\sqrt{r}^{-1}$ where r is the virtual distance the wave travels through the medium [5]. Virtual distance is defined as distinct from real distance due to changes in wave speed as it reflects from the geometry.

The beam spreading term in this experiment cannot be fully accounted for due to the immersion configuration: the standoff of the probe from the Perspex plate has not been defined. If some assumptions are made – namely, if the immersion fluid is assumed to be water, the time $t = 0\mu\text{s}$ is assumed to be the point at which the signal is transmitted, and the probe is assumed to be normal to the surface of the plate – then the standoff can be from the leading edge of response F ($t = 65.9\mu\text{s}$) to be 49.4mm. Using this as the standoff, the beam spread can now be included in the calculation for attenuation by multiplying the logarithm argument by the ratio of beam spreads $^{B_F^T B_F^R}/_{B_{B1}^T B_{B1}^R}$ in equation 6, where the T and R superscripts are used here to indicate transmit and receive paths respectively. A calculation of attenuation including beam spreading has been plotted in fig 4.

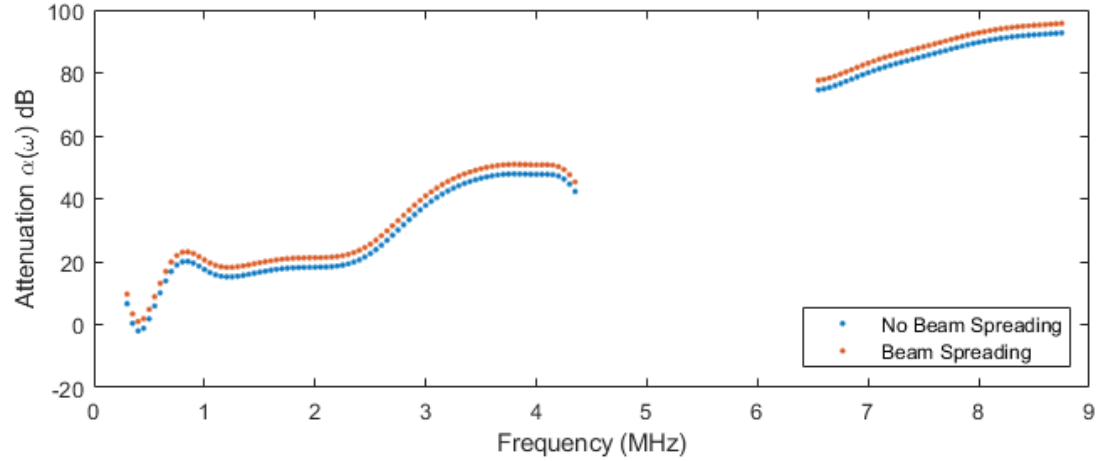

Fig. 4 Calculation of attenuation coefficient from the F and B_1 responses, both including beam spreading and not for comparison. Note that the data not including beam spreading is identical to the data presented in fig 3.

It is clear from this plot that the inclusion of beam spreading in the calculation for attenuation only makes $\sim 5\%$ difference to the overall value of attenuation. Further, as beam spreading is independent of frequency, it makes no difference to the shape of the curve, and thus not including beam spreading in the calculation is a valid approximation to make.

FREQUENCY-DEPENDENT REFLECTION COEFFICIENT FROM AN ADHESIVE JOINT

Two time-domain signals were obtained in a pulse-echo immersion experiment using a 10MHz transducer. This data was filtered using a frequency-domain Hann window centred on the frequency of the transducer: this data is shown in fig 5.

In order to identify the reflection coefficient of the adhesive joint, the response corresponding to the first reflection ($t \approx 60\mu\text{s}$) from the back of the 10.5mm plate was isolated in both time-traces using thresholding identical to the previous attenuation calculation. By considering the reflection and transmission of the wave through the geometry, the frequency spectra of these responses can be written as

$$B_1^{\text{ON}} = I(\omega)\, T_{12} R_{21}^{\text{ON}} T_{21} \tag{7a}$$

$$B_1^{\text{OFF}} = I(\omega)\, T_{12} R_{21}^{\text{OFF}} T_{21} \tag{7b}$$

$$\Rightarrow R_{21}^{\text{ON}}(\omega) = R_{21}^{\text{OFF}} \frac{B_1^{\text{ON}}(\omega)}{B_1^{\text{OFF}}(\omega)} \tag{8}$$

The spectra are plotted in fig 6a. By calculating the value of R_{21}^{OFF} for the aluminium-to-water interface as 0.838 and multiplying this by the ratio of the frequency spectra, the frequency-dependent reflection coefficient of the adhesive joint was calculated: this is shown in fig 6b.

Of particular interest in this plot is the minimum value at frequency $f \approx 9\text{MHz}$. From the definition of the reflection coefficient in equation 2, this suggests that at this frequency the numerator is minimised – in other words, the acoustic impedances of the aluminium and the adhesive are the most similar to one another. Away from this extreme point, the reflection coefficient tends towards 1, implying that the acoustic impedance of aluminium dominates over the impedance of the adhesive.

AUTOMATED TIME-DOMAIN THICKNESS MEASUREMENT

A B-scan was taken in a pulse-echo immersion configuration on a large steel bearing casing with a 5MHz transducer. This data was filtered with a Hann window centred at the transducer frequency, and has been plotted in fig 7.

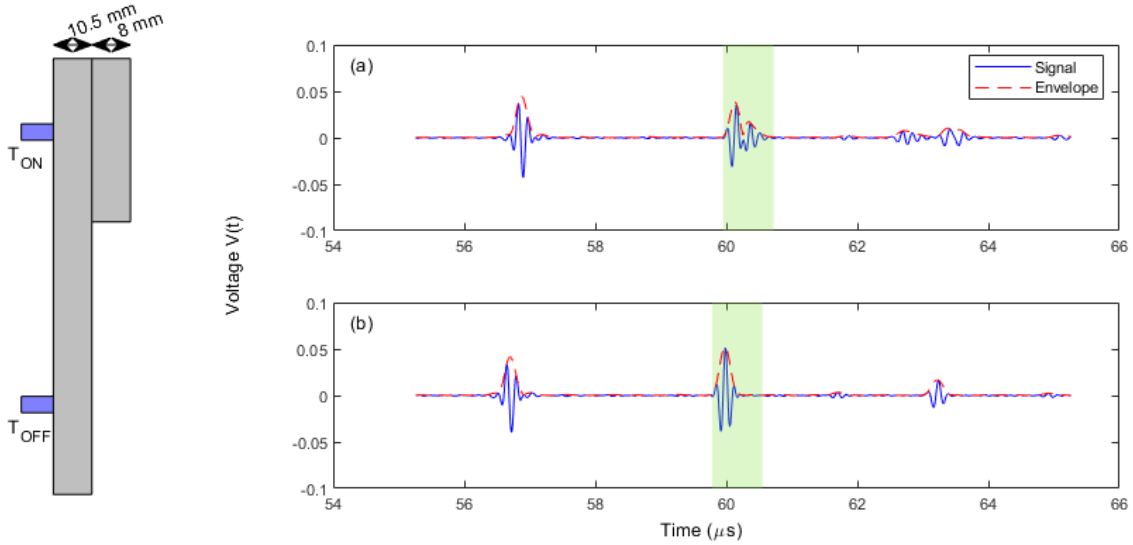

Fig. 5 Voltage time-traces from the immersion setup. (a) Time-trace taken when the transducer was positioned over a 10.5mm aluminium plate bonded to an 8mm aluminium plate with some adhesive of negligible thickness. Subscript "ON" used for reference. (b) Time-trace taken over an unbonded region of the 10.5mm plate. Subscript "OFF" used for reference.

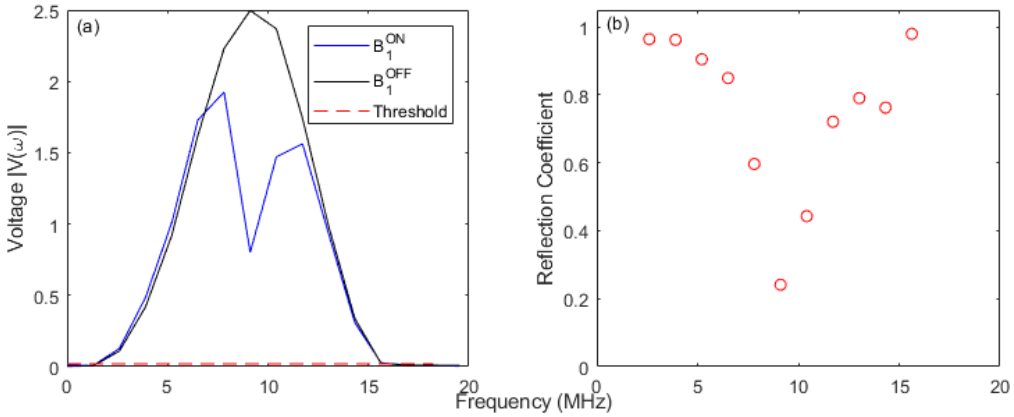

Fig. 6 (a) Frequency spectra for the bonded (blue) and unbonded (black) plate. The threshold plotted here (all data below which is excluded from subsequent calculation) is $0.01\times$ maximum $B_1^{\mathrm{ON}}(\omega)$ value. (b) Reflection coefficient calculated by multiplying R_{21}^{OFF} by the ratio of B_1^{ON} and B_1^{OFF}.

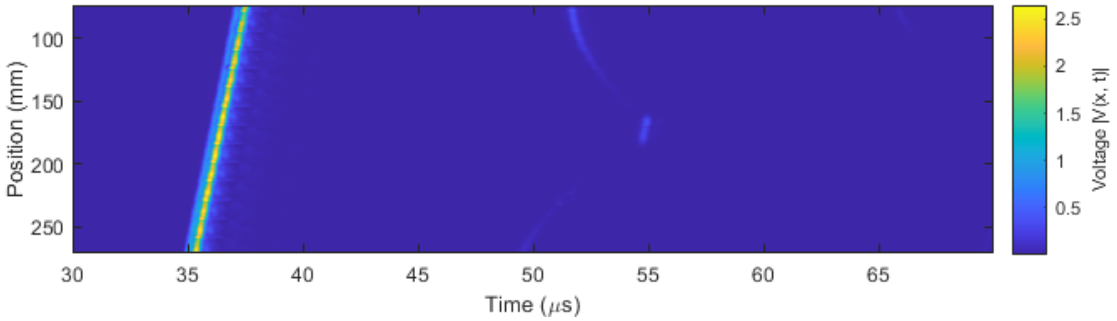

Fig. 7 B-scan taken in immersion configuration of a steel bearing casing. Absolute voltage is plotted here as a function of time and position.

In order to extract the thickness of the bearing case from this scan, the leading edge of the response at each position was determined using the same thresholding method used in previous parts, providing the time-of-flight (ToF) of the wave as it passed through the geometry and reflected from each surface. A threshold of $0.005\times$ maximum $|V(t)|$ was used to do this. After obtaining the ToF data, the thickness of the bearing was obtained simply by multiplying by the wave speed in the material. The wave was assumed to be longitudinal, which in steel has a wave speed of $5900\mathrm{ms}^{-1}$. The thickness as a function of transducer position is plotted in fig 8.

This thickness plot shows a good agreement with the provided dimensions of the geometry of the bearing casing: the maximum thickness of the component is specified to be $55\mathrm{mm}$. It can be seen here that the region $x = 162\text{-}182\mathrm{mm}$ has a good agreement with this value, with the average maximum thickness being $54.7\mathrm{mm}$. In addition, the thickness of the bearing at the transducer position $x = 125\mathrm{mm}$ was found by linear interpolation to be $47.6\mathrm{mm}$.

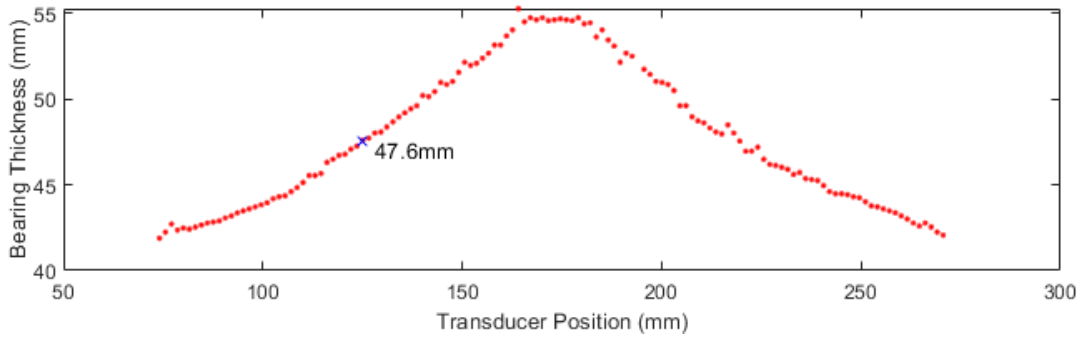

Fig. 8 Thickness of the bearing calculated from the B-scan in fig 7. This has a good agreement with the known maximum thickness of $55\mathrm{mm}$. The thickness of the bearing at transducer position $x = 125\mathrm{mm}$ is annotated as $47.6\mathrm{mm}$.

CONCLUSION

The attenuation of Perspex was found by analysing the frequency-domain of the signal obtained from an acoustic wave, and comparing each resonance of the wave as it passed through the medium. In particular, the attenuation was found to be close to a monotonously increasing function when comparing the front wall reflection with the first back wall reflection. Despite this, it showed a significant deviation from the attenuation of Perspex as it is reported in the literature. Additionally, the attenuation measured by comparing subsequent reflections of the wave proved an even poorer model, showing local maxima at $f \approx 2\mathrm{MHz}$. The reason for this is likely due to the magnitude of response of the higher-order resonances being too low, and thus it only proved a good model of attenuation close to the centre frequency of the transducer.

Additionally, the reflection coefficient of an adhesive joint was measured, and in particular was found to have a minimum at $f \approx 9\mathrm{MHz}$. The thickness of a bearing casing was measured from a B-scan, where a good agreement was found with the specified geometry of the component.

REFERENCES

1. Velichko, A., “The Frequency Domain”, *Ultrasonic NDT*, 29 Mar 2021, University of Bristol. Lecture.
2. Krautkrämer, J., & Krautkrämer, H. (1990). Attenuation of Ultrasonic Waves in Solids. In Ultrasonic Testing of Materials (pp. 108-116). Berlin: Springer.
3. O’Donnell, M., Jaynes, E. T. and Miller, J. G., “Kramers-Kronig relationship between ultrasonic attenuation and phase velocity”, *Acoust. Soc. Am.*, **69**, pp. 696-701, 1981.
4. Zellouf, D., Jayet, Y., Saint-Pierre, N., Tatibouët, J., and Baboux, J. C., “Ultrasonic spectroscopy in polymeric materials. Applications of the Kramers-Kronig relations”, *J. Appl. Phys.* **80**(5), pp. 2728-2732, 1996.
5. Budyn, N., Bevan, R. L. T., Zhang, J., Croxford, A. J. and Wilcox, P. D., "A Model for Multiview Ultrasonic Array Inspection of Small Two-Dimensional Defects," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, **66**(6), pp. 1129-1139, 2019.

APPENDIX A: MATLAB CODE (3A FREQUENCY-DEPENDENT ATTENUATION OF PERSPEX FROM AMPLITUDE SPECTRUM)

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

disp('Frequency dependent attenuation of perspex'); %display the title

%INPUTS

rho_water = 1000.0;
rho_perspex = 1180.0;
rho_air = 1.2;

c_l_perspex = 2730.0;
c_l_water = 1500.0;
c_l_air = 330.0;

d = 7.8e-3;

%file name
fname = '7_8mm_thick_perspex.mat';

%load file
load(fname);

% Isolate signals.
%   To do this, get the envelope of the voltage signal, and get anything
%   above a defined threshold as the individual signals.

fft_pts = 2^nextpow2(length(time));
voltage_spec = fft(voltage, fft_pts);
voltage_spec = voltage_spec(1:fft_pts/2);

freq_step = 1/(time(end)-time(1));
freq = [0 : freq_step : freq_step*length(voltage_spec)-1];

voltage_signal = ifft(voltage_spec, fft_pts);
voltage_envelope = abs(voltage_signal(1:length(voltage)));

threshold = max(voltage_envelope)/100;

is_signal = logical(voltage_envelope >= threshold);

%   There are exactly three signals. Collect them separately.

individual_signals = zeros(length(voltage), 3);

start_idxs = zeros(3,1);
end_idxs = zeros(3,1);

this_signal = 1;
for ii = 1:length(is_signal)-1
    if and(~is_signal(ii), is_signal(ii+1))
        % If we are at the leading edge of a signal.
        start_idx = ii;
        start_idxs(this_signal) = start_idx;
    end

    if and(is_signal(ii), ~is_signal(ii+1))
        % If we are at the trailing edge of a signal.
```

```
        end_idx = ii+1;
        end_idxs(this_signal) = end_idx;

        % Use a square window to obtain the signal.
        individual_signals(start_idx:end_idx, this_signal) = ...
            voltage(start_idx:end_idx);

        this_signal = this_signal+1;
    end
end

% Calculate individual spectra
window_1 = fn_hanning(end_idxs(1)-start_idxs(1)+1, .5, .5);
spectrum_1 = fft(voltage(start_idxs(1):end_idxs(1)) .* window_1, fft_pts);
spectrum_1 = spectrum_1(1:fft_pts/2);
window_2 = fn_hanning(end_idxs(2)-start_idxs(2)+1, .5, .5);
spectrum_2 = fft(voltage(start_idxs(2):end_idxs(2)) .* window_2, fft_pts);
spectrum_2 = spectrum_2(1:fft_pts/2);
window_3 = fn_hanning(end_idxs(3)-start_idxs(3)+1, .5, .5);
spectrum_3 = fft(voltage(start_idxs(3):end_idxs(3)) .* window_3, fft_pts);
spectrum_3 = spectrum_3(1:fft_pts/2);


%plot the data supplied
figure(1)
rectangle('Position', [time(start_idxs(1)+5)*10^6, -0.2, time(end_idxs(1))*10^6-
time(start_idxs(1))*10^6, 0.4], 'EdgeColor', 'none', 'FaceColor', [0.4660 0.8740
0.1880, .25])
rectangle('Position', [time(start_idxs(2)+5)*10^6, -0.2, time(end_idxs(2))*10^6-
time(start_idxs(2))*10^6, 0.4], 'EdgeColor', 'none', 'FaceColor', [0.4660 0.8740
0.1880, .25])
rectangle('Position', [time(start_idxs(3)+5)*10^6, -0.2, time(end_idxs(3))*10^6-
time(start_idxs(3))*10^6, 0.4], 'EdgeColor', 'none', 'FaceColor', [0.4660 0.8740
0.1880, .25])
hold on
plot(time(5:end)*10^6, real(voltage_signal(5:length(voltage))), 'b');
plot(time(5:end)*10^6, voltage_envelope(5:end), 'b--')
xlabel('Time (\mus)');
ylabel('Voltage V(t)');
text(67.5, 0.15, "F")
text(73, 0.075, "B_1")
text(77.5, 0.04, "B_2")
plot([time(5)*10^6, time(end)*10^6], [threshold, threshold], 'r--')
box on
legend('Signal', 'Signal Envelope', 'Threshold')

%%
% Break point for plotting first figure.

freq_threshold = max(abs(spectrum_3))/5;

% Divide the spectra and apply thresholding for use later on in attenuation
% calculation.
F_B1_spec = spectrum_2 ./ spectrum_1;
F_B1_spec(~and(abs(spectrum_1) > freq_threshold, abs(spectrum_2) >
freq_threshold)) = [];
F_B1_freq = freq(and(abs(spectrum_1) > freq_threshold, abs(spectrum_2) >
freq_threshold));

B1_B2_spec = spectrum_3 ./ spectrum_2;
B1_B2_spec(~and(abs(spectrum_2) > freq_threshold, abs(spectrum_3) >
freq_threshold)) = [];
```

```
B1_B2_freq = freq(and(abs(spectrum_2) > freq_threshold, abs(spectrum_3) >
freq_threshold));

F_B2_spec = spectrum_3 ./ spectrum_1;
F_B2_spec(~and(abs(spectrum_1) > freq_threshold, abs(spectrum_3) >
freq_threshold)) = [];
F_B2_freq = freq(and(abs(spectrum_1) > freq_threshold, abs(spectrum_3) >
freq_threshold));

% Plot frequency spectra.

fig = figure(2);

ax1 = subplot(1,3,1, 'box', 'on');
rectangle('Position', [time(start_idxs(1)+5), -0.2, time(end_idxs(1))-
time(start_idxs(1)), 0.4], 'EdgeColor', 'none', 'FaceColor', [0.4660 0.8740
0.1880, .25])
hold on
plot(freq(1:201)*10^-6, abs(spectrum_1(1:201)), 'b')
plot([min(freq(1:201))*10^-6, max(freq(1:201))*10^-6], [freq_threshold,
freq_threshold], 'r--')
text(5e5*10^-6, 5.25, '(a) F')
ylabel('Voltage |V(\omega)|')
box on

ax2 = subplot(1,3,2);
plot(freq(1:201)*10^-6, abs(spectrum_2(1:201)), 'b')
hold on
plot([min(freq(1:201))*10^-6, max(freq(1:201))*10^-6], [freq_threshold,
freq_threshold], 'r--')
text(5e5*10^-6, 5.25, '(b) B_1')
xlabel('Frequency (MHz)')
box on

ax3 = subplot(1,3,3);
plot(freq(1:201)*10^-6, abs(spectrum_3(1:201)), 'b')
hold on
plot([min(freq(1:201))*10^-6, max(freq(1:201))*10^-6], [freq_threshold,
freq_threshold], 'r--')
text(5e5*10^-6, 5.25, '(c) B_2')
legend('Freq Spectrum', 'Threshold')
linkaxes([ax1, ax2, ax3], 'xy')
box on

F_spec = spectrum_1(abs(spectrum_1) > freq_threshold);
F_freq = freq(abs(spectrum_1) > freq_threshold);
B1_spec = spectrum_2(abs(spectrum_2) > freq_threshold);
B1_freq = freq(abs(spectrum_2) > freq_threshold);
B2_spec = spectrum_3(abs(spectrum_3) > freq_threshold);
B2_freq = freq(abs(spectrum_3) > freq_threshold);

% Calculate reflection and transmission coefficients

z_air = rho_air * c_l_air;
z_water = rho_water * c_l_water;
z_perspex = rho_perspex * c_l_perspex;

R_12 = (z_water - z_perspex) / (z_water + z_perspex);
R_21 = (z_perspex - z_water) / (z_perspex + z_water);
T_12 = 2 * z_water / (z_water + z_perspex);
T_21 = 2 * z_perspex / (z_perspex + z_water);
```

```
% Beam spreading parameters.

standoff = time(start_idxs(1)) * 1500.0 / 2;
% γ used in beam spreading. As transducer is assumed to be normal to
% plate, cos α = cos β = 1.
gamma = c_l_water / c_l_perspex;
% Only one B_F calculated as transmit and receive beam spreads are
% identical due to single leg for ray.
B_F = 1 / sqrt(standoff);
B_B1_T = 1 / sqrt(standoff + d/gamma);
% On receive path, γ_R = 1/γ_T
B_B1_R = 1 / sqrt(d + standoff * gamma);

alpha_F_B1 = -1 / (2 * d) * log(abs(F_B1_spec * R_12 / (T_12 * R_21 * T_21)));
alpha_B1_B2 = -1 / (2 * d) * log(abs(B1_B2_spec / (R_21^2)));
alpha_F_B2 = -1 / (4 * d) * log(abs(F_B2_spec * R_12 / (T_12 * R_21^3 * T_21)));

alpha_F_B1_BS = -1 / (2 * d) * log(abs(F_B1_spec * R_12 / (T_12 * R_21 * T_21) *
B_F^2 / (B_B1_T * B_B1_R)));

% Plot Attenuation

figure(3)
scatter(F_B1_freq*10^-6, alpha_F_B1, '.');
hold on
scatter(B1_B2_freq*10^-6, alpha_B1_B2, '.');
scatter(F_B2_freq*10^-6, alpha_F_B2, '.');
xlabel('Frequency (MHz)')
ylabel('Attenuation \alpha(\omega) dB')
box on
legend('B_1 / F', 'B_2 / B_1', 'B_2 / F','Location', 'southeast')

% Plot attenuation with beam spreading

figure(4)
scatter(F_B1_freq*10^-6, alpha_F_B1, '.');
hold on
scatter(F_B1_freq*10^-6, alpha_F_B1_BS, '.');
xlabel('Frequency (MHz)')
ylabel('Attenuation \alpha(\omega) dB')
box on
legend('No Beam Spreading', 'Beam Spreading','Location', 'southeast')
```

APPENDIX B: MATLAB CODE (3B FREQUENCY-DEPENDENT REFLECTION COEFFICIENT FROM AN ADHESIVE JOINT)

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

% Load data.

fname_on = 'joint_on_adhesive.mat';
fname_off = 'joint_off_adhesive.mat';

load(fname_on)
voltage_on = voltage;
clear time voltage

load(fname_off)
voltage_off = voltage;
clear voltage fname_on fname_off

rho_alum = 2700.0;
rho_water = 1000.0;
c_l_alum = 6320.0;
c_l_water = 1500.0;

% Filter signal.

fft_pts = 2^nextpow2(length(time));
on_spec = fft(voltage_on, fft_pts);
on_spec = on_spec(1:fft_pts/2);
off_spec = fft(voltage_off, fft_pts);
off_spec = off_spec(1:fft_pts/2);

freq_step = 1/(time(end) - time(1));
freq = [0 : freq_step : freq_step*(length(on_spec) - 1)];

window = fn_hanning(length(on_spec), 10e6/freq(end), 10e6/freq(end));
on_spec = on_spec .* window;
off_spec = off_spec .* window;

on_signal = ifft(on_spec, fft_pts);
on_signal = on_signal(1:length(voltage_on));
off_signal = ifft(off_spec, fft_pts);
off_signal = off_signal(1:length(voltage_off));

% Get the appropriate response - this will be the third peak when threshold
% is set to max(signal)/50.

on_threshold = max(abs(on_signal))/50;
off_threshold = max(abs(off_signal))/50;

is_response_on = logical(abs(on_signal) > on_threshold);
is_response_off = logical(abs(off_signal) > off_threshold);

this_response_on = 1;
this_response_off = 1;
for ii = 1:length(on_signal)-1
    if and(and(~is_response_on(ii), is_response_on(ii+1)), this_response_on ==
5)
        % If we are at the leading edge of the third signal
        start_idx_on = ii;
    end
```

```
    if and(and(~is_response_off(ii), is_response_off(ii+1)), this_response_off
== 3)
         % If we are at the leading edge of the third signal
         start_idx_off = ii;
    end

    if and(and(is_response_on(ii), ~is_response_on(ii+1)), this_response_on ==
5)
       % If we are at the trailing edge of the third response
       end_idx_on = ii+1;
    end
    if and(and(is_response_off(ii), ~is_response_off(ii+1)), this_response_off
== 3)
       % If we are at the trailing edge of the third response
       end_idx_off = ii+1;
    end

    if and(is_response_on(ii), ~is_response_on(ii+1))
       % If we are at the trailing edge of any response
       this_response_on = this_response_on + 1;
    end
    if and(is_response_off(ii), ~is_response_off(ii+1))
       % If we are at the trailing edge of any response
       this_response_off = this_response_off + 1;
    end
end

len_idx = max(end_idx_on - start_idx_on, end_idx_off - start_idx_off);

% Plot voltage signals

fig = figure(1);

subplot(2,5,[2,3,4,5])
ylim([-0.05, 0.05])
plot(time*10^6, real(on_signal), 'b')
hold on
rectangle('Position', [time(start_idx_on)*10^6, -0.1,
time(start_idx_on+len_idx)*10^6-time(start_idx_on)*10^6, 0.2], 'EdgeColor',
'none', 'FaceColor', [0.4660 0.8740 0.1880, .25])
plot(time*10^6, abs(on_signal), 'r--')
legend('Signal', 'Envelope')
text(54.25, 0.08, '(a)')

subplot(2,5,[7,8,9,10])
ylim([-0.05, 0.05])
plot(time*10^6, real(off_signal), 'b')
hold on
rectangle('Position', [time(start_idx_off)*10^6, -0.1,
time(start_idx_off+len_idx)*10^6-time(start_idx_off)*10^6, 0.2], 'EdgeColor',
'none', 'FaceColor', [0.4660 0.8740 0.1880, .25])
plot(time*10^6, abs(off_signal), 'r--')
text(54.25, 0.08, '(b)')

h = axes('Position',[0 0 1 1],'Visible','off');
text(.5475, .04, 'Time (\mus)')
yl = text(.2125, .43, 'Voltage V(t)');
set(yl, 'Rotation', 90)

rectangle('Position', [0, 0, 1, 1], 'EdgeColor', [0, 0, 0, 0])
rectangle('Position', [0.105, 0.6, 0.03, 0.3], 'FaceColor', [.75, .75, .75])
rectangle('Position', [0.075, 0.1, 0.03, 0.8], 'FaceColor', [.75, .75, .75])
```

```
rectangle('Position', [0.05, 0.25, 0.025, 0.03], 'FaceColor', [.5, .5, 1])
rectangle('Position', [0.05, 0.75, 0.025, 0.03], 'FaceColor', [.5, .5, 1])
annotation('doublearrow', [0.075, 0.105], [0.925, 0.925])
annotation('doublearrow', [0.105, 0.135], [0.925, 0.925])
plate_1 = text(0.085, 0.95, '10.5 mm');
plate_2 = text(0.125, 0.95, '8 mm');
set(plate_1, 'Rotation', 20)
set(plate_2, 'Rotation', 20)
text(0.04, 0.22, 'T_{OFF}')
text(0.04, 0.72, 'T_{ON}')

% Get the frequency spectra.

refl_sig_on = on_signal(start_idx_on:start_idx_on+len_idx);
refl_sig_off = off_signal(start_idx_off:start_idx_off+len_idx);
refl_time_on = time(start_idx_on:start_idx_on+len_idx);
refl_time_off = time(start_idx_off:start_idx_off+len_idx);

fft_pts_on = 2^nextpow2(length(refl_sig_on));
fft_pts_off = 2^nextpow2(length(refl_sig_off));
refl_spec_on = fft(refl_sig_on, fft_pts_on);
refl_spec_on = refl_spec_on(1:fft_pts_on/2);
refl_spec_off = fft(refl_sig_off, fft_pts_off);
refl_spec_off = refl_spec_off(1:fft_pts_off/2);

freq_step_on = 1/(refl_time_on(end) - refl_time_on(1));
freq_on = [0 : freq_step_on : freq_step_on*(length(refl_spec_on)-1)];
freq_step_off = 1/(refl_time_off(end) - refl_time_off(1));
freq_off = [0 : freq_step_off : freq_step_off*(length(refl_spec_off)-1)];

% Calculate the theoretical reflection coefficient

z_water = rho_water * c_l_water;
z_alum = rho_alum * c_l_alum;

R_21_off = (z_alum - z_water) / (z_alum + z_water);

% Get adhesive reflection coefficient.

refl_threshold = max(abs(refl_spec_on))/100;

is_freq_on = logical(abs(refl_spec_on) > refl_threshold);
is_freq_off = logical(abs(refl_spec_off) > refl_threshold);

for ii = 1:length(is_freq_on)-1
    if and(~is_freq_on(ii), is_freq_on(ii+1))
        % If we are at the start of the on_freq peak
        start_idx_on = ii;
    end
    if and(~is_freq_off(ii), is_freq_off(ii+1))
        % If we are at the start of the off_freq peak
        start_idx_off = ii;
    end

    if and(is_freq_on(ii), ~is_freq_on(ii+1))
        % If we are at the start of the on_freq peak
        end_idx_on = ii+1;
    end
    if and(is_freq_off(ii), ~is_freq_off(ii+1))
        % If we are at the start of the off_freq peak
        end_idx_off = ii+1;
```

```
    end
end

start_idx = max(start_idx_on, start_idx_off);
end_idx = min(end_idx_on, end_idx_off);

clear start_idx_on start_idx_off end_idx_on end_idx_off

refl_spec_on_1 = refl_spec_on(start_idx:end_idx);
refl_spec_off_1 = refl_spec_off(start_idx:end_idx);
freq = freq_on(start_idx:end_idx);

R_21_on = R_21_off * real(refl_spec_on_1 ./ refl_spec_off_1);

% Plot reflection coefficient

fig = figure(2);
subplot(1,2,1)
% Only plot up to 15 MHz. Could find an automated way to do this if
% required.
plot(freq_on(1:16)*10^-6, abs(refl_spec_on(1:16)), 'b')
hold on
plot(freq_off(1:16)*10^-6, abs(refl_spec_off(1:16)), 'black')
plot([freq_on(1)*10^-6, freq_on(15)*10^-6], [refl_threshold, refl_threshold],
'r--')
box on
legend('B_1^{ON}', 'B_1^{OFF}', 'Threshold')
ylabel('Voltage |V(\omega)|')
text(.5, 2.4, '(a)')

subplot(1,2,2)
scatter(freq(2:end-1)*10^-6, R_21_on(2:end-1), 'ro')
box on
ylabel('Reflection Coefficient')
ylim([0, 1.05])
xlim([0, 20])
text(0.5, 1.0175, '(b)')

h = axes(fig,'Visible','off');
h.XLabel.Visible='on';
h.YLabel.Visible='on';
xlabel('Frequency (MHz)')
```

APPENDIX C: MATLAB CODE (3B AUTOMATED TIME-DOMAIN THICKNESS MEASUREMENT)

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

load('bearing_casing_bscan.mat')

% Filter data and get envelope.

fft_pts = 2^nextpow2(length(time));
spectra = fft(voltage, fft_pts, 1);
spectra = spectra(1:fft_pts/2, :);

df = 1/(time(end) - time(1));
freq = [0 : df : df*(fft_pts/2 - 1)];

gaussian_window = fn_gaussian(fft_pts/2, 5e6/freq(end), 5e6/freq(end));
gaussian_spectra = spectra .* gaussian_window;
hanning_window = fn_hanning(fft_pts/2, 5e6/freq(end), 5e6/freq(end));
hanning_spectra = spectra .* hanning_window;
hanning_hi_window = fn_hanning_hi_pass(length(spectra), 2*5e6/freq(end),
3*5e6/freq(end));
hanning_hi_spectra = spectra .* hanning_hi_window;
hanning_lo_window = fn_hanning_lo_pass(length(spectra), 2*5e6/freq(end),
3*5e6/freq(end));
hanning_lo_spectra = spectra .* hanning_lo_window;

g_signals = ifft(gaussian_spectra, fft_pts, 1);
g_signals = g_signals(1:length(time), :);

h_signals = ifft(hanning_spectra, fft_pts, 1);
h_signals = h_signals(1:length(time), :);

hh_signals = ifft(hanning_hi_spectra, fft_pts, 1);
hh_signals = hh_signals(1:length(time), :);

hl_signals = ifft(hanning_lo_spectra, fft_pts, 1);
hl_signals = hl_signals(1:length(time), :);

% Plot filtered data.

figure(1)
subplot(2,2,1)
imagesc(time*10^6, pos*10^3, abs(g_signals'))
xlabel('Time (\mus)')
ylabel('Position (mm)')

subplot(2,2,2)
imagesc(time*10^6, pos*10^3, abs(h_signals'))
xlabel('Time (\mus)')
ylabel('Position (mm)')

subplot(2,2,3)
imagesc(time*10^6, pos*10^3, abs(hh_signals'))
xlabel('Time (\mus)')
ylabel('Position (mm)')

subplot(2,2,4)
imagesc(time*10^6, pos*10^3, abs(hl_signals'))
xlabel('Time (\mus)')
```

```
ylabel('Position (mm)')

c = colorbar();
c.Label.String = 'Voltage |V(x, t)|';

% Work out response locations.

threshold = min(max(abs(h_signals), [], 1)) / 200;
is_response = logical(abs(h_signals) > threshold);

lengths = zeros(length(pos), 1);
start_idxs = zeros(length(pos), 3);

for ii = 1:length(pos)
    end_idxs = [0];
    this_response_start = 1;
    this_response_end = 2;
    for jj = 1:length(time)-1
        if and(and(~is_response(jj, ii), is_response(jj+1, ii)), jj >
end_idxs(end)+100)
            % If at the leading edge
            start_idxs(ii, this_response_start) = jj;
            this_response_start = this_response_start + 1;
        end
        if and(is_response(jj, ii), ~is_response(jj+1, ii))
            % If at the trailing edge
            end_idxs(this_response_end) = jj+1;
            this_response_end = this_response_end + 1;
        end
    end
end

% Calculate thickness from response locations.

bearing_thickness = zeros(length(pos), 1);
for ii = 1:length(pos)
    % Confirm that there are at least two leading edges of responses.
    if and(start_idxs(ii, 1) ~= 0, start_idxs(ii, 2) ~= 0)
        bearing_thickness(ii) = (time(start_idxs(ii, 2)) - time(start_idxs(ii,
1))) * 5900.0 / 2;
    end
end
% Remove any data where there are not two leading edges.
pos_ = pos(bearing_thickness ~= 0);
bearing_thickness(bearing_thickness == 0) = [];
h_signals_ = h_signals(:, bearing_thickness ~= 0);

thickness_at_125 = interp1(pos_, bearing_thickness, 0.125, 'linear');

% Plot thickness against transducer position.

figure(2)
scatter(pos_*10^3, bearing_thickness*10^3, 'r.')
hold on
scatter([125], [thickness_at_125*10^3], 'bx')
text(125*1.025, thickness_at_125*10^3*0.99, sprintf('%4.1fmm',
thickness_at_125*10^3))
xlabel('Transducer Position (mm)')
ylabel('Bearing Thickness (mm)')
box on
```