# Exercise 3: Data Processing

Matthew Chandler

## Abstract

Several experimental data sets are examined and processed using MATLAB v9.9.0. 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 some medium to be inspected. The key application of interest for the work outlined here is measurement of properties of the inspection medium, which can be determined entirely from the ultrasonic signal measured using a transducer after interaction with the medium.

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

where describe the transmitter and receiver transducer characteristics, including effects like directivity, transducer frequency response characteristic; describes the time delay of the signal due to propagation; is the beam spread of the wave; is a coefficient which describes the change in amplitude of the wave from reflection from and transmission through boundaries; and is the attenuation.

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

where , , are the incident, reflected and transmitted amplitudes respectively, and is the acoustic impedance of the th medium defined as

where is acoustic pressure, is particle velocity, is the density of the medium and 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 leading to differences in density or wave velocity. Absorption results 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 [1]:

## Frequency-dependent attenuation of Perspex

The time-domain signal was obtained from a pulse-echo immersion test on a 7.8mm Perspex plate using a 2.5 MHz transducer, and is shown in fig 1.

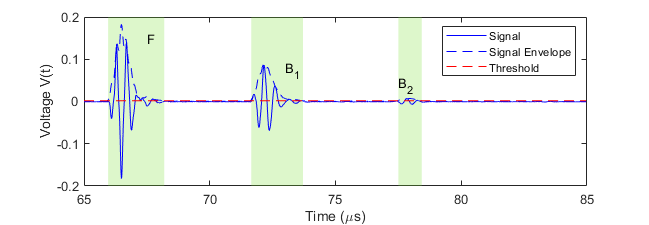


Fig. 1 Time-trace obtained from a pulse-echo immersion test on a Perspex plate. Response F is the first reflection from the front of the plate, and B1 and B2 are subsequent reverberations through the plate from the back of the plate. The envelope of the signal is shown as a dashed blue line, with the threshold shown in red.

The response from each reflection was isolated so that it could be processed independently from the others. This was done using a threshold on the signal, such that any section of data with envelope greater than the threshold ( maximum envelope voltage) was treated as a single response. These are shown as green bands in fig 1, labelled , and respectively. Using the transfer function in equation 1, the spectra of these signals can be approximated by considering the reflection and transmission of the wave through the geometry:

These frequency spectra were calculated using a Fast Fourier Transform, and are plotted in fig 2. The attenuation coefficient was then obtained by dividing the spectra ,

and is plotted in fig 3, where transmission and reflections were calculated from equation 2 assuming that the perspex block was immersed in water.

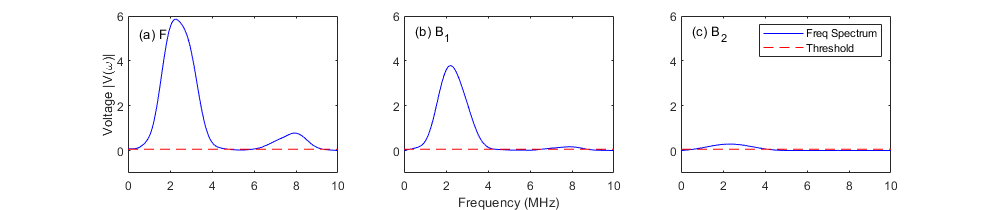


Fig. 2 Fourier transforms of voltage for each signal in fig 1. Note that only the voltage greater than a threshold ( maximum value) has been plotted, such that any frequencies with negligible value are excluded from subsequent calculation.

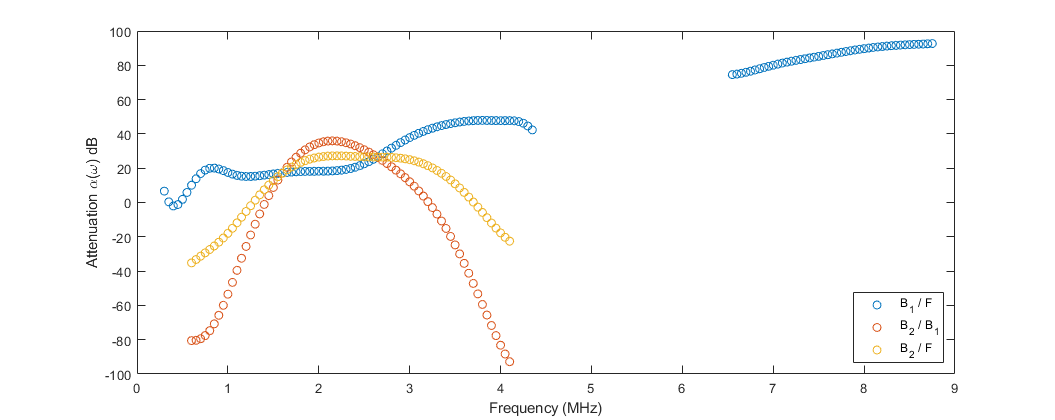


Fig. 3 Attenuation coefficient calculated from fig 2 and equation 6. Note that there is no data available for some frequencies (particularly for the data): this is because these voltages fall below the threshold plotted in fig 2, and thus are excluded from this calculation.

Of particular interest here is that the attenuation calculated from the and responses (blue circles) is close to a monotonously increasing function [3, 4]. There are some local extreme points at around , contradicting this behaviour. As these points are close to the edges, this suggests that the threshold used on the frequency spectra was not large enough. This is further supported by the fact that the attenuation plot calculated from the (orange circles) and the (yellow circles) responses are not monotonously increasing, and instead show single large maxima points. The main cause of this is that the spectrum in fig 2 is very small with respect to the other spectra, suggesting that this may fall below the value of the threshold which would be required to accurately achieve a monotonously increasing function.

Note also that this analysis has assumed that the only difference between the , and responses is the reflection coefficient and attenuation, but does not take into account the beam spreading of the wave as it passes through the medium. The beam spreading term in the transfer function (equation 1) is proportional to where 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 mode as it reflects from the geometry. In this configuration, the virtual distance will be equal to the real distance as there are no changes in wave mode.

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 the immersion fluid is assumed to be water, the standoff can be calculated (assuming that the transducer is normal to the plane of the plate) from the leading edge of response () to be . 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 in equation 6. A calculation of attenuation including beam spreading has been plotted in fig 4.

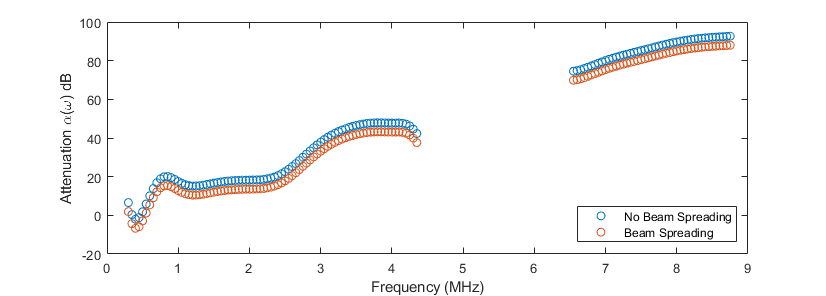


Fig. 4 Calculation of attenuation coefficient from the and responses, both including beam spreading and not for comparison. Note that the data not including beam spreading is identical to the data presented in fig 3.

It is clear from this plot that the inclusion of beam spreading in the calculation for attenuation only makes 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 10 MHz transducer. This data was filtered using a Hanning window on the frequency spectra, 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 from the back of the 10.5 mm plate was isolated in both time-traces using thresholding identical to the previous attenuation calculation: the response of interest is located at . By considering the reflection and transmission of the wave through the geometry, the frequency spectra of these responses can be written as

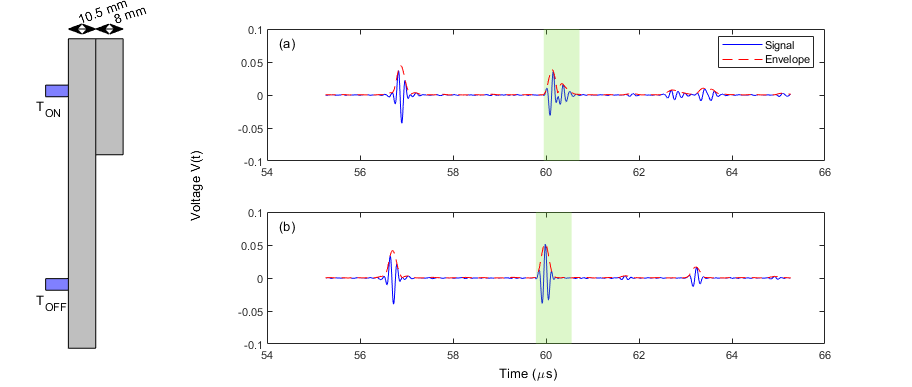


Fig. 5 Voltage time-traces from the immersion setup. (a) Time-trace taken when the transducer was positioned over a 10.5mm aluminium plate bonded to an 8mm aluminium plate with some adhesive of negligible thickness. Subscript “ON” used for future reference. (b) Time-trace taken over an unbonded region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

The spectra are plotted in fig 6a. By calculating the value of 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 . 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 , implying that the acoustic impedance of aluminium dominates over the impedance of the adhesive.

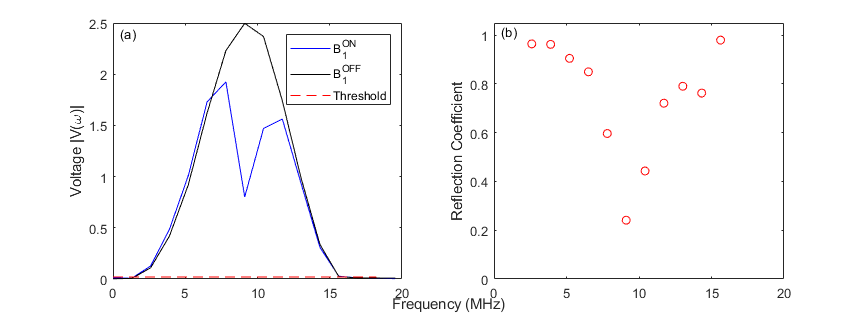


Fig. 6 (a) Frequency spectra for the bonded (blue) and unbonded (black) plate. The threshold plotted here (all data below which is excluded from subsequent calculation) is maximum value. (b) Reflection coefficient calculated by multiplying by the ratio of and .

## Automated time-domain thickness measurement

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

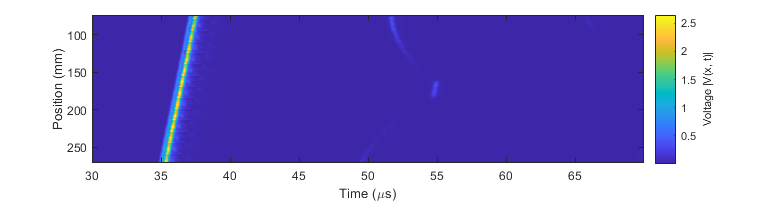


Fig. 7 B-scan taken in immersion configuration of a steel bearing casing. Absolute voltage is plotted here as a function of time and position.

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 maximum signal voltage 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 . The thickness as a function of transducer position is plotted in fig 8.

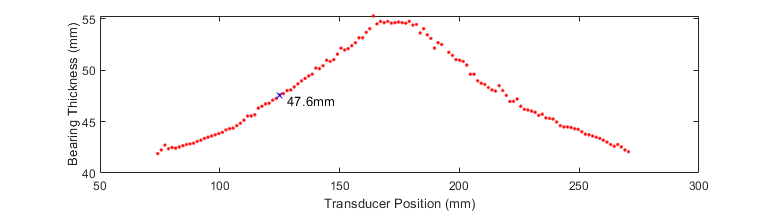


Fig. 8 Thickness of the bearing calculated from the B-scan in fig 7. This has a good agreement with the known maximum thickness of . The thickness of the bearing at transducer position is annotated as .

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 . It can be seen here that the region has a good agreement with this value, with the average maximum thickness being in this region of transducer position. In addition, the thickness of the bearing at the transducer position was found by linear interpolation to be .

## 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 maximum points at around . 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 . 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);

%write your own code here to determine the frequency-dependent attenuation

%of perspex from the data supplied

%%

% 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

% han=axes(fig,'visible','off');

% han.Title.Visible='on';

% han.XLabel.Visible='on';

% han.YLabel.Visible='on';

% ylabel(han,'Voltage |V(t)|');

% xlabel(han,'Frequency (MHz)');

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;

B\_F = 1 / sqrt(2\*standoff);

B\_B1 = 1 / sqrt(2\*standoff + 2\*d);

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 / B\_B1));

% 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(3);

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