

Problem 1 - Modal Behaviour of a Cylindrical Room

The Matlab code for this problem is listed in Appendix 1.

Problem 1a

Table 1 lists the ten lowest resonance mode orders for the room and the respective frequency.

Index	Mode (n_x, n_θ, n_r)	Frequency [Hz]
0	0, 0, 0	0
1	1, 0, 0	17.2
2	0, 1, 0	33.5
3	2, 0, 0	34.3
4	1, 1, 0	37.6
5	2, 1, 0	48.0
6	3, 0, 0	51.5
7	0, 2, 0	55.6
8	1, 2, 0	58.2
9	3, 1, 0	61.4
10	2, 2, 0	65.3

Table 1: Resonant modes of the cylindrical room.

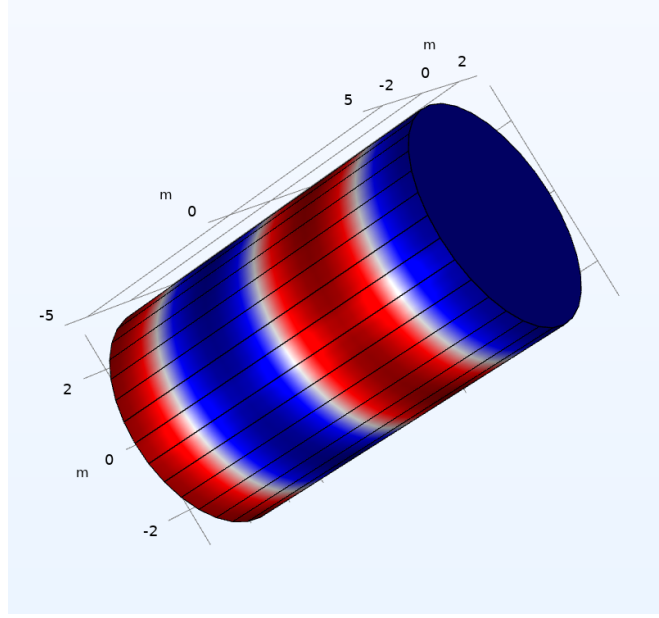
Problem 1b

The two closest modes are (3, 0, 0) and (0, 2, 0) with frequencies of 51.5 Hz and 55.6 Hz, respectively.

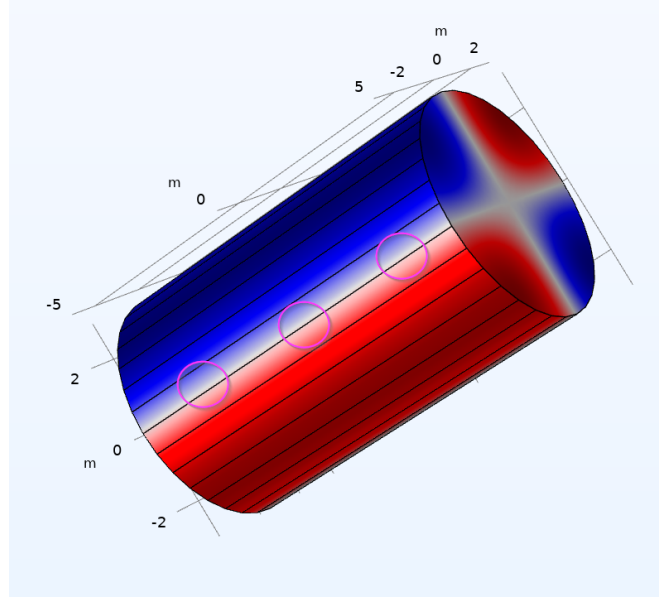
Problem 1c

Figure 1 illustrates the (3, 0, 0) and (0, 2, 0) modes. The white lines in each figure show the modal lines for that mode. **The machine can be placed where the modal lines for each mode overlap.** The figures were produced using the Room Eigenmode Simulator Version 1.1 software package.

The pink rings in Figure 1b indicate 3 possible places where the machine could be placed. These points coincide with the three modal planes shown in Figure 1a. Theoretically, there are an infinite number of places where the machine could be placed. However, placement would take into account practical considerations such as accessibility, etc.



(a) Mode (3, 0, 0)



(b) Mode (0, 2, 0)

Figure 1: Visualization of modes. (a.) Mode (3, 0, 0). (b.) Model (0, 2, 0). The pink circles in [1b](#) illustrate a few modal line intersections where the machine could be placed.

Problem 2 - Sabine Room

The Matlab code for this problem is listed in Appendix 2.

Problem 2a

The reverberant field sound pressure level is approximately 98.3 dB SPL.

Problem 2b

Figure 2 shows the direct, reverberant, and total sound pressure levels for a 25 mW, 125 Hz, broadband, omnidirectional source placed centrally in the room (i.e. the directivity factor is 1).

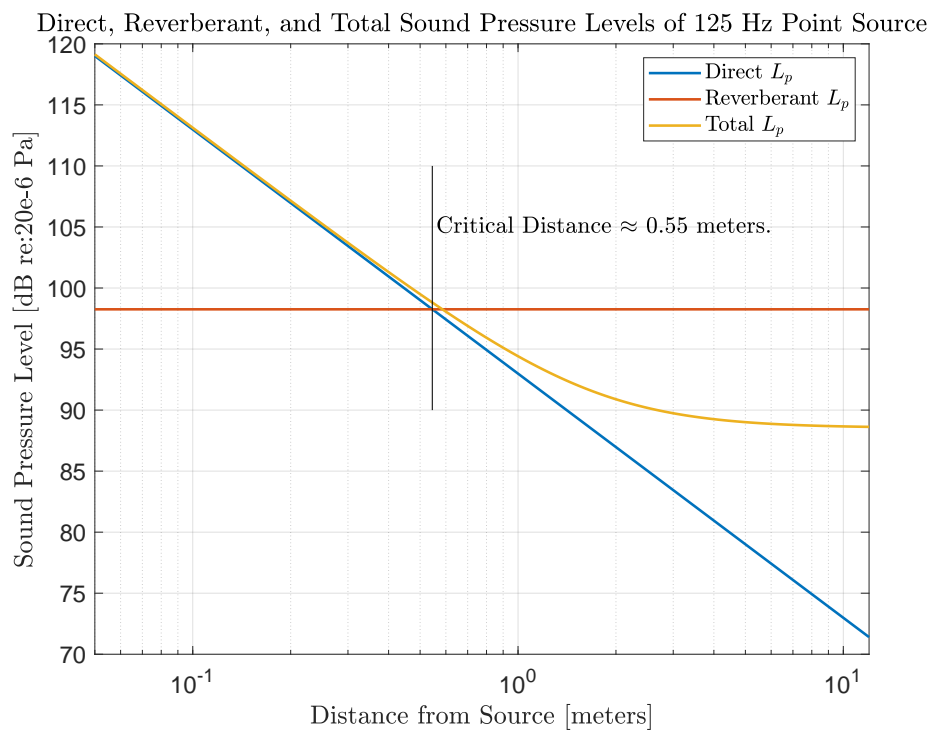


Figure 2: Sound levels for the room produced by a 25 mW, 125 Hz omnidirectional, broadband source.

Problem 3 - Transmission Loss Measurement

The Matlab code for this problem is listed in Appendix 3.

Figure 3 shows the average absorption per octave band based on the T60 data. The calibration plate isolates the receiver room and the absorption calculation does not consider the calibration plate.

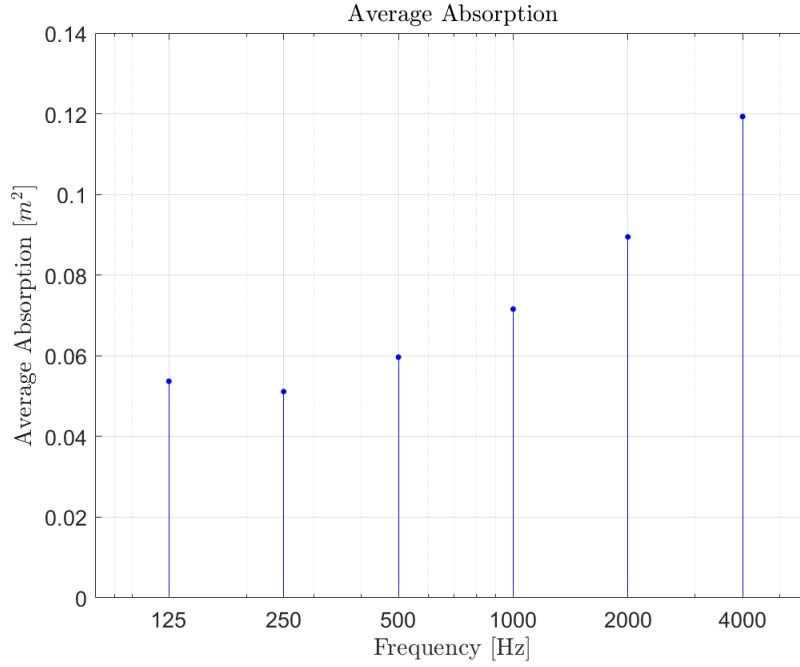


Figure 3: Average absorption per octave band.

Figure 5 shows the transmission loss per octave band.

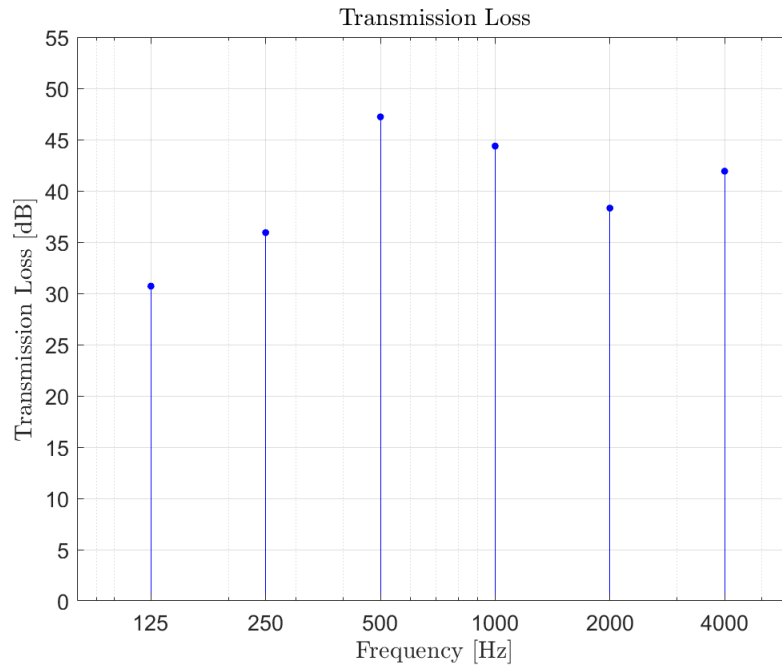


Figure 4: Transmission loss per octave band.

Problem 4 - Panel Transmission Loss

The Matlab code for this problem is listed in Appendix 4.

Problem 4a

The resonance frequency, f_o , of the galvanized steel panel is 4 Hz or $22.6 \frac{\text{radians}}{\text{s}}$.

Problem 4b

i - Critical Frequency and Coincidence Frequency at 75°

The critical frequency is 10,216 Hz and the coincidence frequency at 75° is 10,494 Hz.

ii - Transmission Loss at Angle of Incidence of 75°

ph

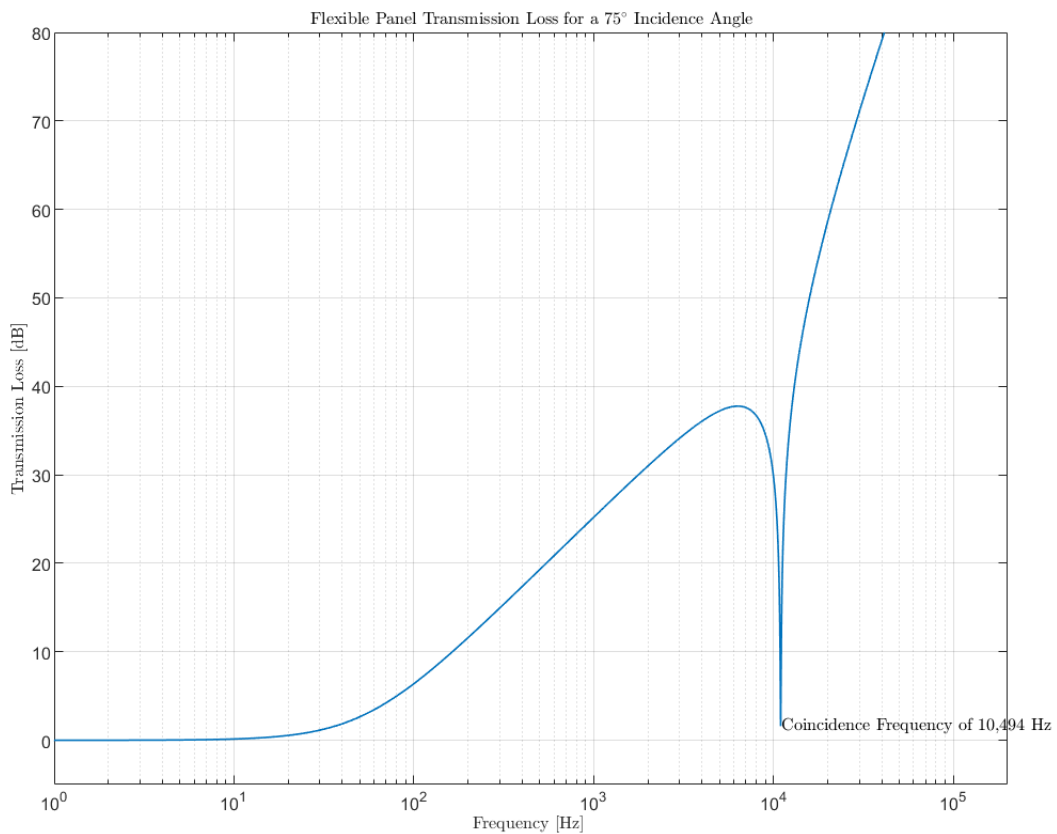


Figure 5: Transmission loss per octave band.

iii - Transmission Loss for Angles of Incidence between $0-90^\circ$

ph

iv - Diffuse Transmission Loss

ph

Problem 4c

Problem 4d

i - Critical Frequency and Coincidence Frequency at 75°

ph

ii - Transmission Loss at Angle of Incidence of 75°

ph

vspace0.25cm

iii - Transmission Loss for Angles of Incidence between $0-90^\circ$

ph

iv - Diffuse Transmission Loss

Problem 5 - Large Enclosure Design

The Matlab code for this problem is listed in Appendix 6.

ph

Problem 6 - Close-fitting Enclosure Design

The Matlab code for this problem is listed in Appendix 6.

ph

1 Appendix - Matlab Code for Problem 1

```
1
2
3
4 %% Synopsis
5
6 % Problem 1 – Modal Behaviour of a Cylindrical Room
7
8
9
10 %% Environment
11
12 close all; clear; clc;
13 % restoredefaultpath;
14
15 % addpath( genpath( '' ), '-begin' );
16 addpath( genpath( './00 Support' ), '-begin' );
17
18 % set( 0, 'DefaultFigurePosition', [ 400 400 900 400 ] ); % [ left bottom width height ]
19 set( 0, 'DefaultFigurePaperPositionMode', 'manual' );
20 set( 0, 'DefaultFigureWindowStyle', 'normal' );
21 set( 0, 'DefaultLineLineWidth', 1.5 );
22 set( 0, 'DefaultTextInterpreter', 'Latex' );
23
24 format ShortG;
25
26 pause( 1 );
27
28
29
30 %% Define Room
31
32 room.radius = 3; % m
33 room.length = 10; % m
34
35
36
37 %% Test Circular Mode Function
38
39 % psi = circular_mode_shape( 3, 1, 2, false ); % 3.7261 – CHECKED FROM CLASS (PLOT NOT CREATED)
40 % psi = circular_mode_shape( 3, 1, 2, true ); % 3.7261 – CHECKED FROM CLASS (PLOT CREATED)
41
42
43
44 %% Define Anonymous Function for the Natural Frequencies
45
46 h_natural_frequencies = @( c, nx, ntheta, nr, Lx, cylinder_radius, plot_flag ) (c/2) .* sqrt( (
    nx/Lx).^2 + (circular_mode_shape(nr, ntheta, cylinder_radius, plot_flag)/cylinder_radius)
    .^2 );
47
48
49
50 %% Calculate the Natural Frequencies
51
52 % The maximum number of radial modes is 5 (indexed from 0 to 4).
53 % The maximum number of angular modes is 8 (indexed from 0 to 7).
54
55 NX_SIZE = 20;
56 NTHETA_SIZE = 7;
57 NR_SIZE = 4;
58 natural_frequencies = nan( NX_SIZE, NTHETA_SIZE, NR_SIZE );
59
60 for nx = 0:1:NX_SIZE
61     for ntheta = 0:1:NTHETA_SIZE
62         for nr = 0:1:NR_SIZE
63             natural_frequencies( nx+1, ntheta+1, nr+1 ) = h_natural_frequencies( 343, nx, ntheta,
                nr, 10, 3, false );
64         end
65     end
66 end
67
68
69
70 %% Part a – Find 10 Lowest Resonance Frequencies
71
72 NUMBER_OF_LOWEST_FREQUENCIES = 11;
```

```

73     mode_indices = ( 1:1:NUMBER_OF_LOWEST_FREQUENCIES ).';
74
75 [ sortedValues, sortedIndices ] = sort( natural_frequencies(:) ); % 21-by-8-by-5 -> 840 elements
76
77 smallestValues = sortedValues( 1:NUMBER_OF_LOWEST_FREQUENCIES );
78 % [ mode_indices    round( smallestValues, 1 ) ]
79 %
80 % 1            0
81 % 2           17.2
82 % 3           33.5
83 % 4           34.3
84 % 5           37.6
85 % 6           47.9
86 % 7           51.5
87 % 8           55.6
88 % 9           58.2
89 % 10          61.4
90 % 11          65.3
91
92 smallestIndices = sortedIndices( 1:NUMBER_OF_LOWEST_FREQUENCIES );
93
94 [ x, y, z ] = ind2sub( size(natural_frequencies), smallestIndices );
95 % ( [ x y z ] - 1 )
96
97 % Verify the calculated mode indices.
98 h_natural_frequencies( 343, 0, 0, 0, 10, 3, false ); % 0 Hz
99 h_natural_frequencies( 343, 1, 0, 0, 10, 3, false ); % 17.2 Hz
100 h_natural_frequencies( 343, 0, 1, 0, 10, 3, false ); % 33.5 Hz
101 h_natural_frequencies( 343, 2, 0, 0, 10, 3, false ); % 34.3 Hz
102 h_natural_frequencies( 343, 1, 1, 0, 10, 3, false ); % 37.6 Hz
103 h_natural_frequencies( 343, 2, 1, 0, 10, 3, false ); % 48.0 Hz
104 h_natural_frequencies( 343, 3, 0, 0, 10, 3, false ); % 51.5 Hz
105 h_natural_frequencies( 343, 0, 2, 0, 10, 3, false ); % 55.6 Hz
106 h_natural_frequencies( 343, 1, 2, 0, 10, 3, false ); % 58.2 Hz
107 h_natural_frequencies( 343, 3, 1, 0, 10, 3, false ); % 61.4 Hz
108 h_natural_frequencies( 343, 2, 2, 0, 10, 3, false ); % 65.3 Hz
109
110
111
112 %% Part b – Two
113
114 % [ (1:11).' abs( smallestValues - 53 ) ]
115
116 temp = [ x y z ] - 1; temp( 7:8, :, : )
117
118 h_natural_frequencies( 343, 3, 0, 0, 10, 3, false ); % 51.5 Hz, (3, 0, 0)
119 h_natural_frequencies( 343, 0, 2, 0, 10, 3, false ); % 55.6 Hz, (0, 2, 0)
120
121
122
123 %% Part c
124
125 % See the report.
126
127
128
129 %% Clean-up
130
131 if ( ~isempty( findobj( 'Type', 'figure' ) ) )
132     monitors = get( 0, 'MonitorPositions' );
133     if ( size( monitors, 1 ) == 1 )
134         autoArrangeFigures( 2, 2, 1 );
135     elseif ( 1 < size( monitors, 1 ) )
136         autoArrangeFigures( 2, 2, 1 );
137     end
138 end
139
140
141 fprintf( 1, '\n\n*** Processing Complete ***\n\n' );
142
143
144
145 %% Reference(s)
146
147 % https://www.mathworks.com/matlabcentral/answers/1883747-how-to-find-the-5-minimum-values-in-a-multidimensional-matrix-and-the-indices-to-which-these-entries

```

2 Appendix - Matlab Code for Problem 2

```
1
2
3
4 %% Synopsis
5
6 % Problem 2 — Sabine Room
7
8
9
10 %% Environment
11
12 close all; clear; clc;
13 % restoredefaultpath;
14
15 % addpath( genpath( '' ), '-begin' );
16 addpath( genpath( '../00 Support' ), '-begin' );
17
18 % set( 0, 'DefaultFigurePosition', [ 400 400 900 400 ] ); % [ left bottom width height ]
19 set( 0, 'DefaultFigurePaperPositionMode', 'manual' );
20 set( 0, 'DefaultFigureWindowStyle', 'normal' );
21 set( 0, 'DefaultLineLineWidth', 1.5 );
22 set( 0, 'DefaultTextInterpreter', 'Latex' );
23
24 format ShortG;
25
26 pause( 1 );
27
28
29
30 %% Define Room
31
32 room.width = 8; % m
33 room.length = 6; % m
34 room.height = 3; % m
35 room.volume = room.width * room.length * room.height; % 144 m^3
36 room.area = 2*(room.width*room.height) + 2*(room.length*room.height) + 2*(room.width*room.
    length); % 180 m^2
37
38 alpha_average_walls_and_floor = 0.05; % For the walls and the floor.
39 alpha_average_ceiling = 0.15; % For the ceiling.
40
41 % For the 125 Hz octave band.
42
43
44
45 %% Part a — Estimate the Reverberant Sound Pressure Level
46
47 Lw = 10*log10( 25e-3 / 1e-12 ); % 103.98 dB
48
49 average_absorption_coefficient = ( (room.width*room.length)*alpha_average_ceiling + (room.width
    *room.length + 2*(room.width*room.height) + 2*(room.length*room.height) ) *
    alpha_average_walls_and_floor ) / room.area; % 0.076667 unitless
50
51 room_constant = room.area * average_absorption_coefficient / ( 1 - average_absorption_coefficient
    ); % 14.9 m^2 or Sabin
52
53
54 sound_pressure_level = Lw + 10*log10( 4 / room_constant ); % 98.3 dB
55
56
57
58 %% Part b — Calculate the Critical Distance
59
60 D0 = 1;
61
62 r = 0:0.05:12; % m
63
64 h_Lp_direct = @( Lw, D0, r ) Lw + 10*log10( D0./(4.*pi.*r.^2) );
65 h_Lp_reverberant = @( Lw, room_constant ) Lw + 10*log10( 4 ./ room_constant );
66 %
67 h_Lp_net = @( Lw, D0, r, room_constant ) Lw + 10*log10( D0./(4.*pi.*r.^2) + 4/room_constant ) +
    10*log10( 343*1.2/400 );
68
69
70 figure( ); ...
```

```

71     h1 = plot( r, h_Lp_direct( Lw, D0, r ) ); hold on;
72     h2 = plot( r, ones( size(r) ).*h_Lp_reverberant( Lw, room_constant ) );
73     h3 = plot( r, h_Lp_net( Lw, D0, r, room.volume ) ); grid on;
74     legend( [ h1, h2 h3 ], { 'Direct $L_p$', 'Reverberant $L_p$', 'Total $L_p$' }, '
        Interpreter', 'Latex' );
75
76     %
77     text( 0.56, 105, 'Critical Distance $\approx$ 0.55 meters.', 'Interpreter', 'Latex' );
78     line( [ 0.545 0.545 ], [ 90 110 ], 'Color', 'k', 'LineWidth', 0.6 );
79
80     xlabel( 'Distance from Source [meters]' ); ylabel( 'Sound Pressure Level [dB re:20e-6 Pa]' )
81     ;
82     title( 'Direct, Reverberant, and Total Sound Pressure Levels of 125 Hz Point Source' );
83     %
84     set( gca, 'XScale', 'log' );
85
86 % Estimate the critical distance (see page 84 of "06-Indoors.pdf" notes for ACS 537).
87 rc = 0.141 * sqrt( D0 * room_constant ); % 0.5451 meters
88
89 return
90 %% Clean-up
91
92 if ( ~isempty( findobj( 'Type', 'figure' ) ) )
93     monitors = get( 0, 'MonitorPositions' );
94     if ( size( monitors, 1 ) == 1 )
95         autoArrangeFigures( 2, 2, 1 );
96     elseif ( 1 < size( monitors, 1 ) )
97         autoArrangeFigures( 2, 2, 1 );
98     end
99 end
100
101 fprintf( 1, '\n\n*** Processing Complete ***\n\n' );
102
103
104
105
106 %% Reference(s)

```

3 Appendix - Matlab Code for Problem 3

```
1
2
3
4 %% Synopsis
5
6 % Question 3 – Transmission Loss Measurement
7
8
9 % Note(s):
10 %
11 %     1.) Lp1 depends on the transmission loss.
12 %
13 %         If the transmission loss is low, then more energy goes to room 2 (i.e., the receiver room
14 %         ).
15 %
16 %         The noise reduction from the source room to the receiver room.
17 %
18 %         Adding the barrier will change the level in the source room. Typically making the sound
19 %         level higher in the source room.
20
21 %% Environment
22
23 close all; clear; clc;
24 % restoredefaultpath;
25
26 % addpath( genpath( '' ), '-begin' );
27 addpath( genpath( '../00 Support' ), '-begin' );
28
29 % set( 0, 'DefaultFigurePosition', [ 400 400 900 400 ] ); % [ left bottom width height ]
30 set( 0, 'DefaultFigurePaperPositionMode', 'manual' );
31 set( 0, 'DefaultFigureWindowStyle', 'normal' );
32 set( 0, 'DefaultLineLineWidth', 1.2 );
33 set( 0, 'DefaultLineMarker', 'x' );
34 set( 0, 'DefaultLineMarkerSize', 15 );
35 % set( 0, 'DefaultAxesLineStyleOrder', { '-' '--o' '+' } );
36 set( 0, 'DefaultTextInterpreter', 'Latex' );
37
38 format ShortG;
39
40 pause( 1 );
41
42
43
44 %% Dimensions of Rooms and Panel
45
46 room.length = 4; % m
47 room.width = 4; % m
48 room.height = 4; % m
49 room.volume = room.length * room.width * room.height; % 64 m^3
50 room.area = 2*(room.length * room.width) + 2*(room.length * room.height) + 2*(room.width *
51     room.height); % 96 m^2
52
53 panel.width = 0.8; % m
54 panel.height = 0.8; % m
55 panel.area = panel.width * panel.height; % 0.64 m^2
56
57 c = 343; % m/s
58
59
60 %% Measurement Data
61
62 octave_band_frequencies = [ 125 250 500 1000 2000 4000 ].'; % Hz
63 T60 = [ 2.0 2.1 1.8 1.5 1.2 0.9 ].'; % seconds
64 spl.source_room = [ 90 95 103 105 100 93 ].'; % dB re: 20e-6 Pa
65 spl.receiver_room = [ 50 50 46 50 50 38 ].'; % dB re: 20e-6 Pa
66 % [ octave_band_frequencies T60 spl.source_room spl.receiver_room (spl.source_room - spl.
67     receiver_room) ]
68
69
70 %% Calculate Average Absorption in the Receiver Room using Reverberation Time Measurements
71
```

```

72 average_absorption = @( volume, area, c, T60 ) ( 55.25 .* volume ) ./ ( area .* c .* T60 );
73
74 receiver_room.average_absorption = average_absorption( room.volume, room.area, c, T60 );
75
76 % Assumption: Calibration panel has very high transmission loss.
77
78 figure( ); ...
79 stem( octave_band_frequencies, receiver_room.average_absorption, 'Marker', '.', 'MarkerSize',
12, 'Color', 'b' ); grid on;
80 xlabel( 'Frequency [Hz]' ); ylabel( 'Average Absorption [m^2]' );
81 title( 'Average Absorption' );
82 %
83 xticks( octave_band_frequencies ); xticklabels( num2cell( octave_band_frequencies ) );
84 set( gca, 'XScale', 'log' );
85 xlim( [ 80 6e3 ] ); ylim( [ 0 0.14 ] );
86
87
88


---


89 %% Calculate the Receiver Room Constant
90
91 % The calibration plate isolates the receiver room.
92
93 % The receiver room does not consider the calibration plate.
94
95 room_constant = @( average_absorption, area ) ( average_absorption * area ) ./ ( 1 -
average_absorption ); % Unitless
96
97 receiver_room.room_constant = room_constant( receiver_room.average_absorption, room.area );
98
99
100


---


101 %% Calculate the Transmission Loss in Each Octave Band
102
103 transmission_coefficient = @( receiver_room_pressure, source_room_pressure, panel_area,
receiver_room_constant ) ( ( receiver_room_pressure ./ source_room_pressure ) .*
receiver_room_constant ) ./ panel_area;
104
105 tau = transmission_coefficient( 10.^(spl.receiver_room./10)*20e-6, 10.^(spl.source_room./10)*20e
-6, panel.area, receiver_room.room_constant );
106
107 TL = -10*log10( tau );
108
109 figure( ); ...
110 stem( octave_band_frequencies, TL, '.', 'MarkerSize', 15, 'Color', 'b' ); grid on;
111 xlabel( 'Frequency [Hz]' ); ylabel( 'Transmission Loss [dB]' );
112 title( 'Transmission Loss' );
113 %
114 xticks( octave_band_frequencies ); xticklabels( num2cell( octave_band_frequencies ) );
115 set( gca, 'XScale', 'log' );
116 xlim( [ 80 6e3 ] ); ylim( [ 0 55 ] );
117
118
119


---


120 %% Clean-up
121
122 if ( ~isempty( findobj( 'Type', 'figure' ) ) )
123     monitors = get( 0, 'MonitorPositions' );
124     if ( size( monitors, 1 ) == 1 )
125         autoArrangeFigures( 2, 2, 1 );
126     elseif ( 1 < size( monitors, 1 ) )
127         autoArrangeFigures( 2, 2, 1 );
128     end
129 end
130
131
132 fprintf( 1, '\n\n\n*** Processing Complete ***\n\n\n' );
133
134
135


---


136 %% Reference(s)

```

4 Appendix - Matlab Code for Problem 4

```
1
2
3
4 %% Synopsis
5
6 % Problem 4 – Panel Transmission Loss
7
8
9
10 %% Environment
11
12 close all; clear; clc;
13 % restoredefaultpath;
14
15 % addpath( genpath( '' ), '-begin' );
16 addpath( genpath( '../40 Assignments/00 Support' ), '-begin' );
17
18 % set( 0, 'DefaultFigurePosition', [ 400 400 900 400 ] ); % [ left bottom width height ]
19 set( 0, 'DefaultFigurePaperPositionMode', 'manual' );
20 set( 0, 'DefaultFigureWindowStyle', 'normal' );
21 set( 0, 'DefaultLineLineWidth', 0.8 );
22 set( 0, 'DefaultTextInterpreter', 'Latex' );
23
24 format ShortG;
25
26 pause( 1 );
27
28
29
30 %% Define Panel
31
32 panel.length = 80e-2; % m
33 panel.E = 200e9; % Pa
34 panel.density = 7800; % kg/m^3
35 panel.v = 0.29; % Poisson's Ratio (unitless)
36 panel.thickness = 1.2e-3; % m
37 panel.eta = 0.001; % Loss factor (unitless)
38
39 c = 343; % m/s
40 rho0 = 1.21; % kg/m^3
41
42
43
44 %% Measured Panel Data
45
46 octave_band_frequencies = [ 63 125 250 500 1000 2000 4000 8000 ].'; % Hz
47 TL = [ 9 14 21 27 32 37 43 42 ].'; % dB
48
49
50
51 %% Problem 4a – Infinite, Rigid Panel Model with Normal Incidence
52
53 D = ( panel.E * panel.thickness.^3 ) / ( 12 * ( 1 - panel.v^2 ) ); % 31.4
54
55 ms = panel.density * panel.thickness; % 9.4 kg/m^2
56
57 wo = pi^2 / panel.length * sqrt( D / ms ); % 22.6 radians/s
58 s = wo^2 * ms; % 4,785.9 kg radians / m^2s^2
59
60 fo = wo / (2*pi); % 4 Hz
61
62
63 % Define an Anonymous function for the rigid panel with normal incidence.
64 h_tau_infinite_rigid_panel = @( f, wo, ms, s, rho0, c, eta) 4 ./ ( ( (2*pi.*f*ms - s./(2*pi.*f))
65     ./ (rho0 * c) ).^2 + ( (wo*ms*eta) ./ (rho0*c) + 2 ).^2 );
66
67
68 %% Problem 4b – Infinite, Flexible Panel Model with Random Incidence
69
70 % The panel has bending waves.
71
72
73 % Part (i.)
74
```



```

75 % The critical frequency.
76 critical_frequency = c^2 / (2*pi) * sqrt( ms / D ); % 10.22 kHz
77
78 % Verify the critical frequency using a 90 degree angle of incidence.
79 critical_frequency_verify_1 = 1./(2*pi) .* sqrt( ms / D ) .* ( c / sind( 90 ) ).^2; % 10.22 kHz
80
81 % Verify the critical frequency using the properties of the panel.
82 critical_frequency_verify_2 = c^2 / ( 1.8 * panel.thickness * sqrt( panel.E / ( panel.density * (
    1 - panel.v^2) ) ) ); % 10.3 kHz
83
84
85 % Coincidence frequency for a 75 degree angle of incidence.
86 phi = 75;
87 h_coincidence_frequency = @( ms, D, c, phi ) 1./(2*pi) * sqrt( ms / D ) .* ( c ./ sind( phi
    ) ).^2;
88 h_coincidence_frequency( ms, D, c, phi ); % 10,949 Hz
89
90
91 % Define an Anonymous function for the flexible panel with random incidence.
92 h_tau_infinite_flexible_panel = @( f, rho0, c, phi, D, eta ) ( 2*rho0.*c*secd(phi)).^2 ./ ( (2*
    rho0.*c*secd(phi) + D*eta*(2*pi.*f./c).^4./(2*pi.*f)*sind(phi)^4).^2 + ...
93 (2*pi.*f*ms - D*(2*pi.*f./c).^4./(2*pi.*f)*sind(phi)^4).^2 );
94
95
96
97 % Part (ii.) — Transmission loss for a 75 degree angle of incidence.
98 f = 0.1:1:100e3;
99
100 figure( ); ...
101 plot( f, -10*log10( h_tau_infinite_flexible_panel( f, rho0, c, phi, D, panel.eta ) ), '
    LineStyle', '-', 'Marker', 'none' ); grid on;
102 text( 11000, 1.6, 'Coincidence Frequency of 10,494 Hz' );
103 xlabel( 'Frequency [Hz]' ); ylabel( 'Transmission Loss [dB]' );
104 title( 'Flexible Panel Transmission Loss for a 75° Incidence Angle' );
105 set( gca, 'XScale', 'log' );
106 axis( [ 1 200e3 -5 80 ] );
107
108 % return
109
110 % Part (iii.)
111 f = 0.1:1:100e3;
112
113
114 eta = panel.eta;
115
116 phi_set = 0:5:90;
117
118 [ phi_set, h_coincidence_frequency( ms, D, c, phi_set ) ];
119
120 t_set = [ ];
121
122
123 h1 = figure( ); ...
124
125 hold on;
126 % stem( octave_band_frequencies ./ (wo / (2*pi) ), TL, 'LineWidth', 0.5, 'Marker', 'o', '
    MarkerSize', 8, 'MarkerEdgeColor', 'b', 'MarkerFaceColor', 'r' ); hold on;
127 stem( octave_band_frequencies, TL, 'LineWidth', 0.5, 'Marker', 'o', 'MarkerSize', 8, '
    MarkerEdgeColor', 'b', 'MarkerFaceColor', 'r' ); hold on;
128 line( [ 16e3 16e3 ], [ 0 65 ], 'Color', 'c' );
129 %
130 % plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s, rho0, c,
    panel.eta ) ), 'LineStyle', '-' );
131 plot( f, -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s, rho0, c, panel.eta ) ), '
    LineStyle', '-' );
132 %
133
134 for phi = phi_set
135 % plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_term1( rho0, c, phi ) ./ ( h_tau_term2(
    rho0, c, phi, D, eta, f ) + h_tau_term3( f, ms, D, phi ) ) ), 'Color', 'r', '
    LineStyle', '-', 'Marker', 'none' );
136 % plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_flexible_panel( f, rho0, c, phi, D
    , panel.eta ) ), 'LineStyle', '--', 'Marker', 'none' );
137
138 plot( f, -10*log10( h_tau_infinite_flexible_panel( f, rho0, c, phi, D, panel.eta ) ), '
    LineStyle', '-', 'Marker', 'none' );
139
140 t_set = [ t_set; h_tau_infinite_flexible_panel( f, rho0, c, phi, D, panel.eta ) ];

```

```

141
142     end
143
144     grid on;
145
146     xlabel( 'Frequency [Hz] ' ); ylabel( 'Transmission Loss [dB] ' );
147     title( 'Measured Panel Transmission Losses' );
148     set( gca, 'XScale', 'log' );
149     axis( [ 2e-3 100e3 -5 70 ] );
150
151     % close( h1 );
152
153
154     N_phi = size( t_set, 1 );
155
156     temp1 = t_set;
157     temp2 = sind( 2*phi_set ).';
158     % temp2 = sind( phi_set ).';
159
160     temp3 = t_set .* repmat( temp2, 1, size( temp1, 2 ) );
161
162     tau_d = 1/N_phi .* nansum( temp3, 1 );
163
164     tau_d_verify = nanmean( t_set .* temp2, 1 );
165
166     % return
167
168 %% Combined Transmission Loss Plot
169
170 figure( ); ...
171     h1 = stem( octave_band_frequencies, TL, 'LineStyle', 'none', 'Marker', 'o', 'MarkerSize', 8,
172             'MarkerFaceColor', 'r', 'MarkerEdgeColor', 'k' ); hold on;
173     h2 = plot( f, -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s, rho0, c, panel.eta ) ), '
174             'LineStyle', '-', 'Color', 'k' );
175     h3 = plot( f, -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s, rho0, c, panel.eta )
176             ./ (200*panel.eta) * ( 4*panel.length / ( panel.length^2 * critical_frequency ) ) ), '
177             'LineStyle', '-', 'Marker', 'none', 'Color', 'k' );
178     h4 = plot( f, -10*log10( tau_d ), 'LineStyle', '-', 'Marker', 'none', 'Color', 'b' );
179     h5 = line( [ 16e3 16e3 ], [ 0 65 ], 'Color', 'r' ); grid on; % 16 kHz Demarcation
180
181     legend( ...
182             [ h1, h2, h3, h4, h5 ], ...
183             'Target Transmission Loss', ...
184             'Infinite Rigid Panel', ...
185             'Infinite Flexible Panel with Diffuse Incidence', ...
186             'Finite Flexible Panel Model', ...
187             '16 kHz Demarcation', ...
188             'Location', 'NorthWest' );
189
190
191     xlabel( 'Frequency [Hz] ' ); ylabel( 'Transmission Loss [dB] ' );
192     title( 'Measured Panel Transmission Losses' );
193     set( gca, 'XScale', 'log' );
194     axis( [ 1 200e3 -5 80 ] );
195
196
197 %% Plot Data and Model — Different Side Materials
198
199 % h_tau_infinite_rigid_panel_side_materials = @( f, wo, ms, s, rho0, c, eta, n ) (4*n) ./ ( (
200 %     (2*pi*f*ms - s./(2*pi*f)) ./ (rho0 * c) ).^2 + ( (wo*ms*eta) ./ (rho0*c) + n + 1 ).^2 );
201
202 % f = 1e-2:1e-2:20e3;
203 %
204 % phi = 15;
205 % eta = panel.eta;
206 %
207 % figure( ); ...
208 %     plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel_side_materials( f, wo, ms,
209 %     s, rho0, c, panel.eta, 1 ) ), 'LineStyle', '-' ); hold on;
210 %     plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel_side_materials( f, wo, ms,
211 %     s, rho0, c, panel.eta, 1/3600 ) ), 'LineStyle', '-' );
212 %     plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel_side_materials( f, wo, ms,
213 %     s, rho0, c, panel.eta, 3600 ) ), 'LineStyle', '-' ); grid on;
214 %
215 %     legend( ...
216 %             'Same Fluid', ...
217 %             'Water to Air', ...
218 %             'Air to Water', ...

```

```

211 %             'Location', 'North' );
212 %
213 %     xlabel( 'Frequency [ $\frac{\omega}{\omega_o}$ ] ' ); ylabel( 'Transmission Loss [dB]' );
214 %     title( 'Measured Panel Transmission Losses' );
215 %     set( gca, 'XScale', 'log' );
216 %     % axis( [ 40 12e3 -5 45] );
217
218
219
220 %% Change in Stiffness
221
222 % figure( ); ...
223 %     stem( octave_band_frequencies ./ (wo / (2*pi) ), TL, 'LineWidth', 0.5, 'Marker', 'o', '
MarkerSize', 8, 'MarkerEdgeColor', 'b', 'MarkerFaceColor', 'r' ); hold on;
224 %
225 %     plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s, rho0, c,
panel.eta ) ), 'LineStyle', '-' );
226 %     plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s*100, rho0, c
, panel.eta ) ), 'LineStyle', '--' );
227 %     plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s*1e-2, rho0,
c, panel.eta ) ), 'LineStyle', '--' );
228 %
229 %     legend( ...
230 %         'Target TL Values', ...
231 %         'Infinite Rigid Panel with Normal Incidence Sound', ...
232 %         'Infinite Rigid Panel with Normal Incidence Sound (s * 100)', ...
233 %         'Infinite Rigid Panel with Normal Incidence Sound (s / 100)', ...
234 %         'Location', 'North' );
235 %
236 %     xlabel( 'Frequency [ $\frac{\omega}{\omega_o}$ ] ' ); ylabel( 'Transmission Loss [dB]' );
237 %     title( 'Measured Panel Transmission Losses - Change in Stiffness' );
238 %     set( gca, 'XScale', 'log' );
239 %     % axis( [ 40 12e3 -5 45] );
240
241
242
243 %% Change in Mass
244
245 % figure( ); ...
246 %     stem( octave_band_frequencies ./ (wo / (2*pi) ), TL, 'LineWidth', 0.5, 'Marker', 'o', '
MarkerSize', 8, 'MarkerEdgeColor', 'b', 'MarkerFaceColor', 'r' ); hold on;
247 %
248 %     plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel( f, wo, ms*100, s, rho0, c
, panel.eta ) ), 'LineStyle', '-' );
249 %     plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s, rho0, c,
panel.eta ) ), 'LineStyle', '-' );
250 %     plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel( f, wo, ms*1e-2, s, rho0,
c, panel.eta ) ), 'LineStyle', '-' );
251 %
252 %     legend( ...
253 %         'Target TL Values', ...
254 %         'Infinite Rigid Panel with Normal Incidence Sound', ...
255 %         'Infinite Rigid Panel with Normal Incidence Sound (s * 100)', ...
256 %         'Infinite Rigid Panel with Normal Incidence Sound (s / 100)', ...
257 %         'Location', 'North' );
258 %
259 %     xlabel( 'Frequency [ $\frac{\omega}{\omega_o}$ ] ' ); ylabel( 'Transmission Loss [dB]' );
260 %     title( 'Measured Panel Transmission Losses - Change in Mass' );
261 %     set( gca, 'XScale', 'log' );
262 %     % axis( [ 40 12e3 -5 45] );
263
264
265
266 %% Change in Loss Factor
267
268 % figure( ); ...
269 %     stem( octave_band_frequencies ./ (wo / (2*pi) ), TL, 'LineWidth', 0.5, 'Marker', 'o', '
MarkerSize', 8, 'MarkerEdgeColor', 'b', 'MarkerFaceColor', 'r' ); hold on;
270 %
271 %     plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s, rho0, c,
panel.eta ) ), 'LineStyle', '-' );
272 %     plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s, rho0, c,
panel.eta*1e2 ) ), 'LineStyle', ':' );
273 %     plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s, rho0, c,
panel.eta*1e-2 ) ), 'LineStyle', ':' );
274 %
275 %     legend( ...
276 %         'Target TL Values', ...

```

```

277 %             'Infinite Rigid Panel with Normal Incidence Sound', ...
278 %             'Infinite Rigid Panel with Normal Incidence Sound (eta * 100)', ...
279 %             'Infinite Rigid Panel with Normal Incidence Sound (eta / 100)', ...
280 %             'Location', 'North' );
281 %
282 %             xlabel( 'Frequency [ $\frac{\omega}{\omega_o}$ ]' ); ylabel( 'Transmission Loss [dB]' );
283 %             title( 'Measured Panel Transmission Losses - Change in Loss Factor' );
284 %             set( gca, 'XScale', 'log', 'YScale', 'log' );
285 %             % axis( [ 40 12e3 -5 45] );
286
287 % return
288
289 %% Clean-up
290
291 if ( ~isempty( findobj( 'Type', 'figure' ) ) )
292     monitors = get( 0, 'MonitorPositions' );
293     if ( size( monitors, 1 ) == 1 )
294         autoArrangeFigures( 2, 2, 1 );
295     elseif ( 1 < size( monitors, 1 ) )
296         autoArrangeFigures( 2, 2, 1 );
297     end
298 end
299
300
301 fprintf( 1, '\n\n*** Processing Complete ***\n\n' );
302
303
304
305 %% Reference(s)

```

5 Appendix - Matlab Code for Problem 5

```
1
2
3
4 %% Synopsis
5
6 % Slide 8 — Noise Reduction and Transmission Loss
7
8 % Volume of the enclosure is much bigger than the machine. Diffuse sound field in the enclosure.
9
10
11
12 %% Environment
13
14 % close all; clear; clc;
15 % restoredefaultpath;
16
17 % addpath( genpath( '' ), '-begin' );
18 addpath( genpath( '../40 Assignments/00 Support' ), '-begin' );
19
20 % set( 0, 'DefaultFigurePosition', [ 400 400 900 400 ] ); % [ left bottom width height ]
21 set( 0, 'DefaultFigurePaperPositionMode', 'manual' );
22 set( 0, 'DefaultFigureWindowStyle', 'normal' );
23 set( 0, 'DefaultLineLineWidth', 0.8 );
24 set( 0, 'DefaultTextInterpreter', 'Latex' );
25
26 format ShortG;
27
28 pause( 1 );
29
30 PRINT_FIGURES = 0;
31
32
33
34 %% Define Machine
35
36 machine.area = 3; % m^2
37 machine.absorption = 0.07; % Sabine
38 machine.D = 1; % Unitless — In air.
39
40 machine.distance = 10; % m
41
42
43
44 %% Data
45
46 octave_band_frequencies = [ 250 500 1000 2000 4000 ].'; % Hz
47 Lw = [ 105 115 106 108 119 ].'; % dB re: 1 pW
48 % [ octave_band_frequencies Lw ]
49
50 % figure( ); ...
51 % h1 = stem( octave_band_frequencies, Lw, 'Marker', '.', 'MarkerSize', 12, 'Color', 'r' );
52 % hold on;
53 % h2 = line( [ 2e2 5e3 ], [ 30 30 ] ); grid on;
54 % legend( [ h1 h2 ], 'Current Sound Pressure Levels', 'Target Sound Pressure Level', 'Location', 'North' );
55 % xlabel( 'Frequency [Hz]' ); ylabel( 'Sound Pressure Level [dB re: 20e-6 Pa]' );
56 % title( 'Sound Power Level Versus Octave Band Center Frequency' );
57 %
58 % axis( [ 150 6e3 0 140 ] );
59 % set( gca, 'XScale', 'log' );
60
61
62 %% Per Octave Band Insertion Loss
63
64 Lp_10_meters = Lw + 10*log10( machine.D / ( 4 * pi * machine.distance^2 ) ); % dB re: 20e-6 Pa
65 % [ octave_band_frequencies Lp_10_meters ]
66 %
67 % The value of R is infinite. The machine is outside in open air.
68
69 octave_band_IL = Lp_10_meters - 30;
70 % [ octave_band_frequencies octave_band_IL ]
71
72
73
```

```

74 %% Anonymous Function for Insertion Loss
75
76 h_IL_large = @( Sw, alpha_w, Si, alpha_i, TL ) 10*log10( 1 + (Sw*alpha_w + Si*alpha_i)./(Sw
+ Si)*10^(TL/10) );
77
78
79
80 %% Find Values of TL and Absorption that will Meet the Target Insertion Loss – Ground Reflecting
81
82 % Assumption(s):
83 %
84 % 1.) The ground is a hard reflecting surface
85 % 2.) The enclosure is a cube.
86 % 3.) There is no noise transmission through the ground.
87
88 enclosure.dimension = 2; % m
89 enclosure.area = 6 * enclosure.dimension^2; % 20 m^2
90
91
92 % https://www.controlnoise.com/wp-content/uploads/2022/02/Acoustic-Enclosures-Datasheet.pdf
93 % https://www.cecoenviro.com/wp-content/uploads/2023/12/Acoustic-Enclosures-8pp-A4-web.pdf
94 % https://www.controlnoise.com/product/acoustic-enclosures/
95
96
97 switch ( 3 )
98
99     case 1
100
101         % 250 Hz – QBV-2
102         alpha_w = 0.27; % From specification sheet.
103         TL = 20; % From specification sheet.
104         IL_estimates(1) = h_IL_large( enclosure.area, alpha_w, machine.area, machine.
absorption, TL );
105
106         % 500 Hz – QBV-2
107         alpha_w = 0.96; % From specification sheet.
108         TL = 29; % From specification sheet.
109         IL_estimates(2) = h_IL_large( enclosure.area, alpha_w, machine.area, machine.
absorption, TL );
110
111         % 1 kHz – QBV-2
112         % alpha_w = 1.13; % From specification sheet.
113         alpha_w = 0.99; % From specification sheet. See comment on slide 28 of Lecture 10.
114         TL = 40; % From specification sheet.
115         IL_estimates(3) = h_IL_large( enclosure.area, alpha_w, machine.area, machine.
absorption, TL );
116
117         % 2 kHz – QBV-2
118         % alpha_w = 1.08; % From specification sheet.
119         alpha_w = 0.99; % From specification sheet. See comment on slide 28 of Lecture 10.
120         TL = 50; % From specification sheet.
121         IL_estimates(4) = h_IL_large( enclosure.area, alpha_w, machine.area, machine.
absorption, TL );
122
123         % 4 kHz – QBV-2
124         alpha_w = 0.99; % From specification sheet.
125         TL = 55; % From specification sheet.
126         IL_estimates(5) = h_IL_large( enclosure.area, alpha_w, machine.area, machine.
absorption, TL );
127
128
129     case 2
130
131         % Note: Absorption values are carried over from QBV-2.
132
133         % 250 Hz – QBV-3
134         alpha_w = 0.27; % From specification sheet.
135         TL = 25; % From specification sheet.
136         IL_estimates(1) = h_IL_large( enclosure.area, alpha_w, machine.area, machine.
absorption, TL );
137
138         % 500 Hz – QBV-2
139         alpha_w = 0.96; % From specification sheet.
140         TL = 33; % From specification sheet.
141         IL_estimates(2) = h_IL_large( enclosure.area, alpha_w, machine.area, machine.
absorption, TL );
142
143         % 1 kHz – QBV-2

```

```

144 % alpha_w = 1.13; % From specification sheet.
145 alpha_w = 0.99; % From specification sheet. See comment on slide 28 of Lecture 10.
146 TL = 46; % From specification sheet.
147 IL_estimates(3) = h_IL_large( enclosure.area, alpha_w, machine.area, machine.
    absorption, TL );
148
149 % 2 kHz – QBV-2
150 % alpha_w = 1.08; % From specification sheet.
151 alpha_w = 0.99; % From specification sheet. See comment on slide 28 of Lecture 10.
152 TL = 53; % From specification sheet.
153 IL_estimates(4) = h_IL_large( enclosure.area, alpha_w, machine.area, machine.
    absorption, TL );
154
155 % 4 kHz – QBV-2
156 alpha_w = 0.99; % From specification sheet.
157 TL = 58; % From specification sheet.
158 IL_estimates(5) = h_IL_large( enclosure.area, alpha_w, machine.area, machine.
    absorption, TL );
159
160
161 case 3
162
163 % Note: Absorption values are carried over from QBV-2.
164
165 % 250 Hz – QBV-3
166 alpha_w = 0.99; % From specification sheet.
167 TL = 39; % From specification sheet.
168 IL_estimates(1) = h_IL_large( enclosure.area, alpha_w, machine.area, machine.
    absorption, TL );
169
170 % 500 Hz – QBV-2
171 alpha_w = 0.96; % From specification sheet.
172 TL = 59; % From specification sheet.
173 IL_estimates(2) = h_IL_large( enclosure.area, alpha_w, machine.area, machine.
    absorption, TL );
174
175 % 1 kHz – QBV-2
176 % alpha_w = 1.13; % From specification sheet.
177 alpha_w = 0.99; % From specification sheet. See comment on slide 28 of Lecture 10.
178 TL = 68; % From specification sheet.
179 IL_estimates(3) = h_IL_large( enclosure.area, alpha_w, machine.area, machine.
    absorption, TL );
180
181 % 2 kHz – QBV-2
182 % alpha_w = 1.08; % From specification sheet.
183 alpha_w = 0.99; % From specification sheet. See comment on slide 28 of Lecture 10.
184 TL = 67; % From specification sheet.
185 IL_estimates(4) = h_IL_large( enclosure.area, alpha_w, machine.area, machine.
    absorption, TL );
186
187 % 4 kHz – QBV-2
188 alpha_w = 0.91; % From specification sheet.
189 TL = 72; % From specification sheet.
190 IL_estimates(5) = h_IL_large( enclosure.area, alpha_w, machine.area, machine.
    absorption, TL );
191
192 end
193
194 [ octave_band_IL IL_estimates.' (octave_band_IL - IL_estimates.' ) ]
195
196
197
198 % What is the most restrictive case?
199
200
201
202 %% Find Values of TL and Absorption that will Meet the Target Insertion Loss – Ground with Cover
203
204 % Assumption(s):
205 %
206 % 1.) The ground is covered with the absorption material.
207 % 2.) The enclosure is a cube.
208
209
210
211 %% Clean-up
212
213 if ( ~isempty( findobj( 'Type', 'figure' ) ) )

```

```

214     monitors = get( 0, 'MonitorPositions' );
215     if ( size( monitors, 1 ) == 1 )
216         autoArrangeFigures( 2, 2, 1 );
217     elseif ( 1 < size( monitors, 1 ) )
218         autoArrangeFigures( 2, 2, 1 );
219     end
220 end
221
222
223 fprintf( 1, '\n\n*** Processing Complete ***\n\n' );
224
225
226
227 %% Reference(s)

```

6 Appendix - Matlab Code for Problem 6

```
1
2
3
4 %% Synopsis
5
6 % Lecture 11, Wednesday, February 19, 2025
7
8 % The compressor elevated above the ground.
9
10
11
12 %% Environment
13
14 close all; clear; clc;
15 % restoredefaultpath;
16
17 % addpath( genpath( '' ), '-begin' );
18 addpath( genpath( '../40 Assignments/00 Support' ), '-begin' );
19
20 % set( 0, 'DefaultFigurePosition', [ 400 400 900 400 ] ); % [ left bottom width height ]
21 set( 0, 'DefaultFigurePaperPositionMode', 'manual' );
22 set( 0, 'DefaultFigureWindowStyle', 'normal' );
23 set( 0, 'DefaultLineLineWidth', 0.8 );
24 set( 0, 'DefaultTextInterpreter', 'Latex' );
25
26 format ShortG;
27
28 pause( 1 );
29
30 PRINT_FIGURES = 0;
31
32
33
34 %% Define Anonymous Functions
35
36 h_RA_term_1 = @( rho0, c, S, k, delta_mu, D, w ) ( rho0*c/S ) * ( ( k * sqrt( (2*3.178e-5) / (
    rho0*w ) ) * D * 0.004 ) / (2*S) * 1.4364 );
37 h_RA_term_2 = @( rho0, c, S, k, delta_mu, D, w, h ) ( rho0*c/S ) * 0.288*k*3.178e-5*log10((4*
    S)/(pi*h^2));
38 h_RA_term_3 = @( rho0, c, S, k, delta_mu, D, w, h ) ( rho0*c/S ) * (0.5*S*k^2)/(2*pi);
39
40
41
42 %% Define Compressor
43
44 compressor.width = 1; % m
45 compressor.depth = 1; % m
46 compressor.height = 2; % m
47 compressor.area = 2*(compressor.width * compressor.depth) + 2*(compressor.width * compressor.
    height) + 2*(compressor.depth * compressor.height); % 3 m^2
48 compressor.volume = compressor.width * compressor.depth * compressor.height; % m^3
49
50 compressor.power_level = 105; % dB re: 1e-12 Watts
51 compressor.frequency = 50; % Hz
52
53 c = 343; % m/s
54
55 rho0 = 1.2; % kg/m^3 CHECK
56
57
58
59 %% Sound Level Target
60
61 sound_level_target = 82; % dB re: 20e-6 Pascals
62
63
64
65 %% Define Workshop
66
67 R = 40; % m^2 or Sabins
68
69
70
71 %% Define Close-fitting Enclosure
72
```

```

73 helmholtz_factor = (2 * pi * compressor.frequency) / c; % 0.92 m
74 %
75 % For a small enclosure, k*d << 1. Therefore d << 1.1.
76 d = 0.75;
77 d = 0.25;
78 % d = 1;
79 % helmholtz_factor * d; % 0.69
80
81 enclosure.width = compressor.width + d; % m
82 enclosure.depth = compressor.depth + d; % m
83 enclosure.height = 3; % m; compression height is 2 m
84 % enclosure.height = compressor.height + d;
85 enclosure.area = 2*(enclosure.width * enclosure.depth) + 2*(enclosure.width * enclosure.
    height) + 2*(enclosure.depth * enclosure.height);
86 enclosure.volume = enclosure.width * enclosure.depth * enclosure.height;
87
88 enclosure.E = 3.6e9; % Pascals
89 enclosure.thickness = 3.81e-2; % m
90 enclosure.density = 800; % kg/m^3
91 enclosure.poisson_ratio = 0.25; % Unitless
92
93 % Clamped boundary conditions.
94
95
96
97 %% Calculate Diffuse Sound Pressure Level
98
99 % Assume distance is beyond the critical distance, so the distance value is
100 % large and its associated term is not relevant.
101
102 sound_pressure_level = 105 + 10*log10( 4/R ); % 95 dB SPL
103
104
105
106 %% Calculate the Required Insertion Loss
107
108 target_insertion_loss = sound_pressure_level - 82 % 13 dB
109
110
111
112 %% Insertion Loss
113
114 % For the insertion loss to be high, we need:
115 %
116 % 1.) Compliance of the air to be high; volume of enclosure must be large.
117 % 2.) Compliance of each enclosure wall to be low; low area, high stiffness, edges clamped).
118 % AREA IS THE DOMINATE FACTOR OVER VOLUME.
119
120
121 % The correction factor for clamped walls. See Figure 12.4 on slide 9 of the Lecture 11 notes.
122 aspect_ratio = enclosure.height / enclosure.width; % 1.7
123 correction_factor = 2; % Approximate value read from the Figure 12.4.
124
125 bending_stiffness = ( enclosure.E * enclosure.thickness^3 ) / ( 12*( 1 - enclosure.poisson_ratio
    ^2 ) ); % 1.78e7
126 h_wall_compliance = @( wall_area, correction_factor ) ( 0.001 * wall_area^3 *
    correction_factor ) / bending_stiffness;
127
128 Ca = enclosure.volume / ( rho0 * c^2 );
129
130
131 % Top
132 top.area = enclosure.width * enclosure.depth;
133 top.aspect_ratio = max( enclosure.width, enclosure.depth ) / min( enclosure.width, enclosure.
    depth );
134 top.correction_factor = 3.8;
135 top.compliance = h_wall_compliance( top.area, top.correction_factor );
136
137
138 % Side 1
139 side_1.area = enclosure.depth * enclosure.height;
140 side_1.aspect_ratio = max( enclosure.width, enclosure.height ) / min( enclosure.width, enclosure.
    height );
141 side_1.correction_factor = 2;
142 side_1.compliance = h_wall_compliance( side_1.area, side_1.correction_factor );
143
144 % Side 2
145 side_2.compliance = side_1.compliance;

```

```

146
147
148 % Side 3
149 side_3.area = enclosure.width * enclosure.height;
150 side_3.aspect_ratio = max( enclosure.width, enclosure.height ) / min( enclosure.width, enclosure.
    height );
151 side_3.correction_factor = 2;
152 side_3.compliance = h_wall_compliance( side_3.area, side_3.correction_factor );
153
154 % Side 4
155 side_4.compliance = side_3.compliance;
156
157
158 estimated_insertion_loss = 20*log10( 1 + Ca / ( top.compliance + 2*side_1.compliance + 2* side_3.
    compliance ) ); % 59.2 dB
159
160
161
162 %% Compliance of the Air Intake
163
164 air_intake_radius = 10e-2; % m
165 air_intake_thickness = enclosure.thickness; % m
166 air_intake_frequency = 50; % Hz
167 air_intake_angular_frequency = 2*pi*air_intake_frequency; % radians/s
168
169 viscosity = 1.5e-5; % m^2/s
170
171 h = 0.3; % CHECK
172
173 f = 50;
174 term_1 = h_RA_term_1( rho0, c, pi*(air_intake_radius^2)^2/4, 2*pi*f/c, sqrt( (2 * 3.178e-5 )
    / ( 2*pi*f * rho0 ) ), pi * 0.1, 2*pi*f );
175 term_2 = h_RA_term_2( rho0, c, pi*(air_intake_radius^2)^2/4, 2*pi*f/c, sqrt( (2 * 3.178e-5 )
    / ( 2*pi*f * rho0 ) ), pi * 0.1, 2*pi*f, 0.3 );
176 term_3 = h_RA_term_3( rho0, c, pi*(0.1)^2/4, 2*pi*f/c, sqrt( (2 * 3.178e-5 ) / ( 2*pi*f *
    rho0 ) ), pi * 0.1, 2*pi*f, 0.3 );
177 impedance.real = term_1 + term_2 + term_3;
178
179
180 % Deng (1998)
181 epsilon = 1;
182 L_o = air_intake_radius * ( 1.27 / (1 + 1.92*epsilon) - 0.086 );
183
184 L_e = enclosure.thickness + 2*L_o;
185 impedance.imaginary = 1j * rho0 * (2 * pi * f) * L_e / ( pi*0.1^2/4 );
186
187
188 impedance.net = impedance.real + impedance.imaginary;
189 compliance_of_hole = 1 / impedance.net;
190 Cl = abs( compliance_of_hole );
191
192 estimated_insertion_loss
193 estimated_insertion_loss_with_hole = 20*log10( (Cl + Ca) / ( Cl + ( top.compliance + 2*side_1.
    compliance + 2* side_3.compliance ) ) )
194
195
196 13 - estimated_insertion_loss_with_hole;
197
198
199
200 critical_frequency = c^2/(2*pi)*sqrt( enclosure.density * enclosure.thickness / bending_stiffness
    );
201 %
202 % The critical frequency is 25 Hz.
203 %
204 % The frequency of the compressor is 50 Hz.
205
206
207
208
209
210 %% Clean-up
211
212 if ( ~isempty( findobj( 'Type', 'figure' ) ) )
213     monitors = get( 0, 'MonitorPositions' );
214     if ( size( monitors, 1 ) == 1 )
215         autoArrangeFigures( 2, 2, 1 );
216     elseif ( 1 < size( monitors, 1 ) )

```

```
217         autoArrangeFigures( 2, 2, 1 );
218     end
219 end
220
221
222 fprintf( 1, '\n\n*** Processing Complete ***\n\n' );
223
224
225
226 %% Reference (s)
```
