

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.

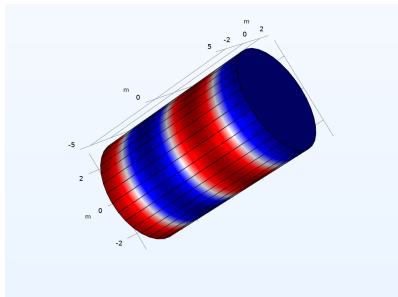
Index	Mode (n_x, n_θ, n_r)	Frequency [Hz]
0	0, 0, 0	0
1	1, 0, 0	17.2
2	0, 1, 0	33.5
3	2, 0, 0	34.3
4	1, 1, 0	37.6
5	2, 1, 0	48.0
6	3, 0, 0	51.5
7	0, 2, 0	55.6
8	1, 2, 0	58.2
9	3, 1, 0	61.4
10	2, 2, 0	65.3

Table 1: Resonant modes of the cylindrical room.

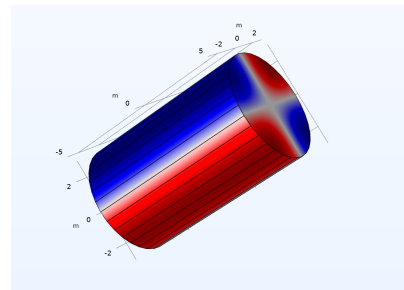
Problem 1b

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



(a) A subfigure



(b) A subfigure

Figure 1: A figure with two subfigures

Problem 2 - Sabine Room

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

For these muffler comparisons, the following assumptions were made:

- There is no flow.
- There are no resistive terms.
- The load impedance was not included because the transmission loss does not require it.

Problem 2a

Figure ?? shows the transmission loss profiles for a simple expansion chamber, a double-tuned expansion chamber, and a cascaded double-tuned expansion chamber muffler.

The peaks for the simple expansion chamber (red, dashed line) are approximately 22 dB and occur at frequencies with a wavelength that is a quarter of the length of the expansion chamber. Minimal loss occurs at half wavelength multiples.

The addition of the extension tube inside the muffler produces a quarter wavelength resonator. The side branch of Ji (2005; Slide 11, Lecture 3 notes) was used to calculate L_o . For the cascaded double-tuned expansion chamber, the extension tubes produce a secondary quarter wavelength resonator.

As noted in office hours, there is no damping which produces artificially high resonances.

Problem 2b

Figure ?? shows the transmission loss profiles for a cascaded double-tuned expansion chamber and a modified version of this muffler.

Two modifications were made to the original muffler:

1. The left 3" extension tube in the left chamber was shortened to 2" inches, making the respective muffler section 1" longer.
2. The left 3" extension tube in the right chamber was lengthened to 4", making the respective muffler section 1" shorter.

These modifications change the symmetry of the cascaded system, and allow the resonate frequencies to be changed independently.

Problem 3 - Transmission Loss Measurement

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

Table 2 lists the length of the pipe section and the mouthpiece.

Item	Length [mm]
Pipe	145
Mouthpiece	90

Table 2: Calculated length of the pipe and length of the mouthpiece.

Problem 4 - Panel Transmission Loss

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

Problem 4a

Table 3 lists the Mach numbers for each pipe section. The flow rate is $0.017462 \frac{\text{m}^3}{\text{s}}$.

Pipe	Area [m ²]	Mach Number [unitless]
Inlet	0.000507	-0.10047
Outlet	0.00811	-0.0062795

Table 3: Calculated Mach numbers.

Problem 4b

Figure ?? shows the transmission loss profiles.

The addition of flow to the intake system introduces a slight phase delay, a lower overall level of loss (approximately 22 dB), and greater loss at the dips. The phase delay is easier to see at respective dips in the loss profile.

i - Critical Frequency and Coincidence Frequency at 75°

ph

ii - Transmission Loss at Angle of Incidence of 75°

ph

iii - Transmission Loss for Angles of Incidence between 0-90°

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

$\text{vspace}0.25\text{cm}$

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 PRINT_FIGURES = 0;
29
30
31
32 %% Define Cylindrical Room
33
34 room.radius = 3; % m
35 room.length = 10; % m
36
37
38
39 %% Test Circular Mode Function
40
41 % psi = circular_mode_shape( 3, 1, 2, false ); % 3.7261 – CHECKED FROM CLASS (PLOT NOT CREATED)
42 % psi = circular_mode_shape( 3, 1, 2, true ); % 3.7261 – CHECKED FROM CLASS (PLOT CREATED)
43
44
45
46 %% Define Anonymous Function for the Natural Frequencies
47
48 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 );
49
50
51
52 %% Calculate the Natural Frequencies
53
54 % The maximum number of radial modes is 5 (indexed from 0 to 4).
55 % The maximum number of angular modes is 8 (indexed from 0 to 7).
56
57 NX_SIZE = 20;
58 NTHETA_SIZE = 7;
59 NR_SIZE = 4;
60 natural_frequencies = nan( NX_SIZE, NTHETA_SIZE, NR_SIZE );
61
62 for nx = 0:1:NX_SIZE
63     for ntheta = 0:1:NTHETA_SIZE
64         for nr = 0:1:NR_SIZE
65             natural_frequencies( nx+1, ntheta+1, nr+1 ) = h_natural_frequencies( 343, nx, ntheta,
66                 nr, 10, 3, false );
67         end
68     end
69 end
70
71
72 %% Part a – Find 10 Lowest Resonance Frequencies
```



```

73
74 NUMBER_OF_LOWEST_FREQUENCIES = 11;
75 mode_indices = [ 1:1:NUMBER_OF_LOWEST_FREQUENCIES ].';
76
77 [ sortedValues, sortedIndices ] = sort( natural_frequencies(:) ); % 21-by-8-by-5 -> 840 elements
78
79 smallestValues = sortedValues( 1:NUMBER_OF_LOWEST_FREQUENCIES );
80 % [ mode_indices round( smallestValues, 1 ) ]
81 %
82 % 1 0
83 % 2 17.2
84 % 3 33.5
85 % 4 34.3
86 % 5 37.6
87 % 6 47.9
88 % 7 51.5
89 % 8 55.6
90 % 9 58.2
91 % 10 61.4
92 % 11 65.3
93
94
95 smallestIndices = sortedIndices( 1:NUMBER_OF_LOWEST_FREQUENCIES );
96
97 [ x, y, z ] = ind2sub( size(natural_frequencies), smallestIndices );
98 [ x y z ] - 1;
99
100
101 % Verify the calculated mode indices.
102 h_natural_frequencies( 343, 0, 0, 0, 10, 3, false ); % 0 Hz
103 h_natural_frequencies( 343, 1, 0, 0, 10, 3, false ); % 17.2 Hz
104 h_natural_frequencies( 343, 0, 1, 0, 10, 3, false ); % 33.5 Hz
105 h_natural_frequencies( 343, 2, 0, 0, 10, 3, false ); % 34.3 Hz
106 h_natural_frequencies( 343, 1, 1, 0, 10, 3, false ); % 37.6 Hz
107 h_natural_frequencies( 343, 2, 1, 0, 10, 3, false ); % 48.0 Hz
108 h_natural_frequencies( 343, 3, 0, 0, 10, 3, false ); % 51.5 Hz
109 h_natural_frequencies( 343, 0, 2, 0, 10, 3, false ); % 55.6 Hz
110 h_natural_frequencies( 343, 1, 2, 0, 10, 3, false ); % 58.2 Hz
111 h_natural_frequencies( 343, 3, 1, 0, 10, 3, false ); % 61.4 Hz
112 h_natural_frequencies( 343, 2, 2, 0, 10, 3, false ); % 65.3 Hz
113
114
115
116 %% Part b – Two
117
118 [ (1:11).', abs( smallestValues - 53 ) ]
119
120 temp = [ x y z ] - 1;
121 temp( 7:8, :, : )
122
123 % Modes:
124 % ( 3, 0, 0 ) and ( 0, 2, 0 )
125
126 h_natural_frequencies( 343, 3, 0, 0, 10, 3, true ); % 51.5 Hz
127
128 h_natural_frequencies( 343, 0, 2, 0, 10, 3, true ); % 55.6 Hz
129
130
131
132 %% Part c
133
134 % For mode ( 3, 0, 0 ), place the source in the center of the cylinder at 5 meters.
135
136 % For mode ( 0, 2, 0 ), place the source in the center of the cylinder.
137
138
139
140 %% Clean-up
141
142 if ( ~isempty( findobj( 'Type', 'figure' ) ) )
143     monitors = get( 0, 'MonitorPositions' );
144     if ( size( monitors, 1 ) == 1 )
145         autoArrangeFigures( 2, 2, 1 );
146     elseif ( 1 < size( monitors, 1 ) )
147         autoArrangeFigures( 2, 2, 1 );
148     end
149 end
150

```

```

151
152 fprintf( 1, '\n\n*** Processing Complete ***\n\n\n' );
153
154
155
156 %% Reference(s)
157
158 % See slide 32 on "Lecture 06 – Room Modes – Filled.pptx".
159
160 % 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 % ACS 547, Sabine Room Milestone
7
8 % See slide 22 on "Lecture 07 - Sabine rooms - Filled.pptx".
9
10
11
12 %% Environment
13
14 close all; clear; clc;
15 % restoredefaultpath;
16
17 % addpath( genpath( '' ), '-begin' );
18 addpath( genpath( '../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', 1.5 );
24 set( 0, 'DefaultTextInterpreter', 'Latex' );
25
26 format ShortG;
27
28 pause( 1 );
29
30 PRINT_FIGURES = 0;
31
32
33
34 %% Information
35
36 room.width = 8; room.length = 6; room.height = 3; % meters
37 room_volume = room.width * room.length * room.height; % 144 m^3
38 room_area = 2*(room.width*room.height) + 2*(room.length*room.height) + 2*(room.width*room.length); % 180 m^2
39
40 alpha_average_walls_and_floor = 0.05; % For the walls and the floor.
41 alpha_average_ceiling = 0.15; % For the ceiling.
42 %
43 % For the 125 Hz octave band.
44
45
46
47 %% Part a - Estimate the reverberant sound pressure level.
48
49 Lw = 10*log10( 25e-3 / 1e-12 ); % 103.98 dB
50
51 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
52
53 room_constant = room_area * average_absorption_coefficient / ( 1 - average_absorption_coefficient ); % 14.9 m^2 or Sabines
54
55
56 sound_pressure_level = Lw + 10*log10( 4 / room_constant ); % 98.3 dB
57
58
59
60 %% Part b
61
62 D0 = 1;
63
64 r = 0:0.05:12; % meters
65
66 h_Lp_direct = @( Lw, D0, r ) Lw + 10*log10( D0./(4.*pi.*r.^2) );
67 h_Lp_reverberant = @( Lw, room_constant ) Lw + 10*log10( 4 ./ room_constant );
68 %
69 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 );
70
```

```

71
72 figure( ); ...
73 plot( r, h_Lp_direct( Lw, D0, r ) ); hold on;
74 plot( r, ones( size(r) ).*h_Lp_reverberant( Lw, room_constant ) );
75 plot( r, h_Lp_net( Lw, D0, r, room_volume ) ); grid on;
76 legend( 'Direct $L_p$', 'Reverberant $L_p$', 'Total $L_p$', 'Interpreter', 'Latex' );
77 %
78 text( 0.545, 100, 'Critical Distance $\approx$ 0.55 meters.', 'Interpreter', 'Latex' );
79 %
80 xlabel( 'Distance from Source [meters]' ); ylabel( 'Sound Pressure Level [dB re:20e-6
Pascals]' );
81 title( 'Sound Pressure Components from Direct and Reverberant Fields from 125 Hz Point Source
' );
82 %
83 set( gca, 'XScale', 'log' );
84
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
90
91 %% Clean-up
92
93 if ( ~isempty( findobj( 'Type', 'figure' ) ) )
94     monitors = get( 0, 'MonitorPositions' );
95     if ( size( monitors, 1 ) == 1 )
96         autoArrangeFigures( 2, 2, 1 );
97     elseif ( 1 < size( monitors, 1 ) )
98         autoArrangeFigures( 2, 2, 1 );
99     end
100 end
101
102
103 fprintf( 1, '\n\n*** Processing Complete ***\n\n' );
104
105
106
107 %% Reference(s)
108
109 %% Overall, A-weighted Level
110
111 % Note(s):
112 %
113 % The above analysis was done using the unweighted sound level for the 500 Hz octave band.
114 %
115 % If an overall level is to be calculated (i.e., across a set of octave bands), then this
analysis
116 % must be done for all octave band center frequencies. Once the unweighted sound pressure
117 % levels at the location are determined, the respective octave band A-weighting offsets are
118 % applied.
119 %
120 % The overall A-weighted sound pressure levels is then calculated logarithmically add using the
121 % expression ,
122 %
123 %  $10 \cdot \log_{10} \left( \sum ( 10^{(Lp_a/10)} ) \right)$ 

```

3 Appendix - Matlab Code for Problem 3

```
1
2
3
4 %% Synopsis
5
6 % Question 3 – Transmission Loss Measurement
7
8
9 % Reference(s):
10 %
11 % Slide 8 – Noise Reduction and Transmission Loss
12
13
14 % Assumptions:
15 %
16 % 1.) Lp1 depends on the transmission loss.
17 %
18 % If the transmission loss is low then more energy goes to room 2 (i.e., the receiver room)
19 %
20 % The noise reduction from the source room to the receiver.
21 %
22 % Adding the barrier will change the level in the source room. Typically making the sound
    level higher in the source room.
23
24
25
26 %% Environment
27
28 close all; clear; clc;
29 % restoredefaultpath;
30
31 % addpath( genpath( '' ), '-begin' );
32 addpath( genpath( '../00 Support' ), '-begin' );
33
34 % set( 0, 'DefaultFigurePosition', [ 400 400 900 400 ] ); % [ left bottom width height ]
35 set( 0, 'DefaultFigurePaperPositionMode', 'manual' );
36 set( 0, 'DefaultFigureWindowStyle', 'normal' );
37 set( 0, 'DefaultLineLineWidth', 1.0 );
38 set( 0, 'DefaultLineMarker', 'x' );
39 set( 0, 'DefaultLineMarkerSize', 15 );
40 % set( 0, 'DefaultAxesLineStyleOrder', { '-' '--o' '+' } );
41 set( 0, 'DefaultTextInterpreter', 'Latex' );
42
43 format ShortG;
44
45 pause( 1 );
46
47 PRINT_FIGURES = 0;
48
49
50
51 %% Dimensions of Rooms and Panel
52
53 room.length = 4; % meters
54 room.width = 4; % meters
55 room.height = 4; % meters
56 room.volume = room.length * room.width * room.height; % 64 m^2
57 room.area = 2*(room.length * room.width) + 2*(room.length * room.height) + 2*(room.width *
    room.height); % 96 m^2
58
59 panel.width = 0.8; % meters
60 panel.height = 0.8; % meters
61 panel.area = panel.width * panel.height; % 0.64 m^2
62
63
64
65 %% Data
66
67 octave_band_frequencies = [ 125 250 500 1000 2000 4000 ].'; % Hz
68 T60 = [ 2.0 2.1 1.8 1.5 1.2 0.9 ].'; % seconds
69 spl.source_room = [ 90 95 103 105 100 93 ].'; % dB re: 20e-6 Pascals
70 spl.receiver_room = [ 50 50 46 50 50 38 ].'; % dB re: 20e-6 Pascals
71
72 c = 343; % meters per second
```

```

73
74
75
76 %% Pressure Difference
77
78 spl.delta = spl.source_room - spl.receiver_room;
79
80 % figure( ); ...
81 %     stem( octave_band_frequencies, spl.delta, 'Marker', '.', 'MarkerSize', 12, 'Color', 'r' );
82 %     grid on;
83 %     xlabel( 'Frequency [Hz]' ); ylabel( 'Transmission Loss [dB]' );
84 %     title( 'Pressure Difference Versus Octave Band Center Frequency' );
85 %     %
86 %     xticks( octave_band_frequencies ); xticklabels( num2cell( octave_band_frequencies ) );
87 %     set( gca, 'XScale', 'log' );
88 %     xlim( [ 80 6e3 ] ); ylim( [ 0 65 ] );
89 % https://www.mathworks.com/matlabcentral/answers/413686-how-to-set-log-scale-range
90
91
92
93 %% Determine Average Absorption in the Receiver Room using Reverberation Time Measurements
94
95 average_absorption = @( volume, area, c, T60 ) ( 55.25 .* volume ) ./ ( area .* c .* T60 );
96
97 receiver_room.average_absorption = average_absorption( room.volume, room.area, c, T60 );
98
99 % Assumption: Calibration panel has very high transmission loss.
100
101 % figure( ); ...
102 %     stem( octave_band_frequencies, receiver_room.average_absorption, 'Marker', '.', 'MarkerSize', 12, 'Color', 'r' ); grid on;
103 %     xlabel( 'Frequency [Hz]' ); ylabel( 'Average Absorption [Sabine]' );
104 %     title( 'Average Absorption in Receiver Room Versus Octave Band Center Frequency' );
105 %     %
106 %     xticks( octave_band_frequencies ); xticklabels( num2cell( octave_band_frequencies ) );
107 %     set( gca, 'XScale', 'log' );
108 %     xlim( [ 80 6e3 ] ); ylim( [ 0 0.14 ] );
109
110
111
112 %% Determine Receiver Room Constant
113
114 room_constant = @( average_absorption, area ) ( average_absorption * area ) ./ ( 1 -
    average_absorption ); % Unitless
115
116 receiver_room.room_constant = room_constant( receiver_room.average_absorption, room.area );
117
118
119
120 %% Determine the Transmission Loss in Each Octave Band
121
122 % The calibration plate isolate the receiver room and the area of the
123 % receiver room does not consider the calibration plate.
124
125 transmission_coefficient = @( receiver_room_pressure, source_room_pressure, panel_area,
    receiver_room_constant ) ( ( receiver_room_pressure ./ source_room_pressure ) .*
    receiver_room_constant ) ./ panel_area;
126
127 tau = transmission_coefficient( 10.^(spl.receiver_room./10)*20e-6, 10.^(spl.source_room./10)*20e-6, panel.area, receiver_room.room_constant );
128
129 TL = -10*log10( tau );
130
131 figure( ); ...
132 %     stem( octave_band_frequencies, TL, '.', 'MarkerSize', 15, 'Color', 'r' ); grid on;
133 %     xlabel( 'Frequency [Hz]' ); ylabel( 'Transmission Loss [dB]' );
134 %     title( 'Transmission Loss Per Octave Band' );
135 %     %
136 %     xticks( octave_band_frequencies ); xticklabels( num2cell( octave_band_frequencies ) );
137 %     set( gca, 'XScale', 'log' );
138 %     xlim( [ 80 6e3 ] ); ylim( [ 0 55 ] );
139
140
141
142 %% Validation
143
144 TL_verify = spl.source_room - spl.receiver_room + 10*log10( panel.area ./ receiver_room.

```

```

145     room_constant )
146 return
147
148 %% Clean-up
149
150 if ( ~isempty( findobj( 'Type', 'figure' ) ) )
151     monitors = get( 0, 'MonitorPositions' );
152     if ( size( monitors, 1 ) == 1 )
153         autoArrangeFigures( 2, 2, 1 );
154     elseif ( 1 < size( monitors, 1 ) )
155         autoArrangeFigures( 2, 2, 1 );
156     end
157 end
158
159
160 fprintf( 1, '\n\n*** Processing Complete ***\n\n' );
161
162
163
164 %% Reference(s)

```

4 Appendix - Matlab Code for Problem 4

```
1
2
3
4 %% Synopsis
5
6 % Slide 8 — Noise Reduction and 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 PRINT_FIGURES = 0;
29
30
31
32 %% Parameters
33
34 c = 343; % m/s
35 rho0 = 1.21; % kg
36
37 panel.length = 80e-2; % meters
38 %
39 panel.E = 200e9; % Pascals
40 panel.density = 7800; % kg / m^3
41 panel.v = 0.29; % Poisson's Ratio (unitless)
42 panel.thickness = 1.2e-3; % m
43 panel.eta = 0.001; % Loss factor (unitless)
44
45
46
47 %% Panel Data
48
49 octave_band_frequencies = [ 63 125 250 500 1000 2000 4000 8000 ].'; % Hz
50 TL = [ 9 14 21 27 32 37 43 42 ].'; % dB
51
52
53 % figure( ); ...
54 % stem( octave_band_frequencies, TL, 'LineWidth', 0.5, 'Marker', 'o', 'MarkerSize', 8, '
55 % MarkerEdgeColor', 'b', 'MarkerFaceColor', 'r' ); grid on;
56 % xlabel( 'Frequency [Hz]' ); ylabel( 'Transmission Loss [dB]' );
57 % title( 'Measured Panel Transmission Losses' );
58 % set( gca, 'XScale', 'log' );
59 % axis( [ 40 12e3 -5 45] );
60
61
62 %% Problem 4a — Infinite, Rigid Panel Model with Normal Incidence
63
64 D = ( panel.E * panel.thickness.^3 ) / ( 12 * ( 1 - panel.v^2 ) ); % 31.4
65
66 % From lecture 9 on Wednesday, February 12, 2025, the equivalent bending
67 % moment of the panel is a half-wavelength.
68 %
69 wavelength = 2 * panel.length; % 1.6 meters
70
71 ms = panel.density * panel.thickness;
72
73 wo = pi^2 / panel.length * sqrt( D / ms );
74 s = wo^2 * ms;
```



```

75
76 h_tau_infinite_rigid_panel = @( f, wo, ms, s, rho0, c, eta ) 4 ./ ( ( (2*pi.*f*ms - s./(2*pi.*f))
    ./ (rho0 * c) ).^2 + ( (wo*ms*eta) ./ (rho0*c) + 2 ).^2 );
77
78 h_tau_infinite_rigid_panel_side_materials = @( f, wo, ms, s, rho0, c, eta, n ) (4*n) ./ ( ( (2*
    pi.*f*ms - s./(2*pi.*f)) ./ (rho0 * c) ).^2 + ( (wo*ms*eta) ./ (rho0*c) + n + 1 ).^2 );
79
80
81
82 %% Problem 4b — Infinite, Flexible Panel Model with Random Incidence
83
84 % Panel has bending waves.
85
86
87 h_tau_term1 = @( rho0, c, phi ) ( 2*rho0*c*secd(phi) ).^2; % Checked
88 h_tau_term2 = @( rho0, c, phi, D, eta, f ) ( 2*rho0*c*secd(phi) + (D*eta*(2*pi.*f./c).^4)/(2*pi
    .*f) .* sind(phi).^4 ).^2; % Checked
89 h_tau_term3 = @( f, ms, D, phi ) ( 2*pi.*f*ms - D*(2*pi.*f./c).^4/(2*pi.*f) .* sind(phi).^4 )
    .^2; % Checked
90
91
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 + ...
    (2*pi.*f*ms - D*(2*pi.*f./c).^4/(2*pi.*f)*sind(phi)^4).^2 );
93
94
95 % return
96
97 %% Plot Data and Model — Air on Both Sides
98
99 % 75 degrees from normal to panel.
100
101 h_c_bending_wave = @( D, f, ms ) ( (D*(2*pi.*f).^2) ./ ms ).^0.25;
102 %
103 % Proportional to the square-root of frequency.
104 %
105 % Small wavelenths (high frequencies; arrive first) travel faster than long wavelength (
    low frequencies; arrive later).
106
107 phi = 75;
108 h_coincidence_frequency = @( ms, D, c, phi ) 1./(2*pi) * sqrt( ms / D ) .* ( c ./ sind( phi
    ) ).^2; % 10,949 Hz
109
110 h_coincidence_frequency( ms, D, c, phi ); % 10,949 Hz
111
112 [ (0:15:90).' h_coincidence_frequency( ms, D, c, [ 0:15:90 ] ).' ];
113
114
115 critical_frequency = 1./(2*pi) .* sqrt( ms / D ) .* ( c / sind( 90 ) ).^2 % 10,216 Hz
116 critical_frequency_verify_1 = c^2 / (2*pi) * sqrt( ms / D ); % lowest coincidence
    frequency
117 critical_frequency_verify_2 = c^2 / ( 1.8 * panel.thickness * sqrt( panel.E / ( panel.
    density * ( 1 - panel.v^2 ) ) ) );
118
119
120
121
122 % f = 1e-2:1e-2:40e3;
123 f = 1e-2:1:100e3;
124
125
126 eta = panel.eta;
127
128 phi_set = 0:10:90;
129
130 [ phi_set.' h_coincidence_frequency( ms, D, c, phi_set ).' ]
131
132 t_set = [ ];
133
134
135 h1 = figure( ); ...
136
137 hold on;
138 % stem( octave_band_frequencies ./ (wo / (2*pi) ), TL, 'LineWidth', 0.5, 'Marker', 'o', '
    MarkerSize', 8, 'MarkerEdgeColor', 'b', 'MarkerFaceColor', 'r' ); hold on;
139 stem( octave_band_frequencies, TL, 'LineWidth', 0.5, 'Marker', 'o', 'MarkerSize', 8, '
    MarkerEdgeColor', 'b', 'MarkerFaceColor', 'r' ); hold on;
140 line( [ 16e3 16e3 ], [ 0 65 ], 'Color', 'c' );
141 %

```

```

142 % plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s, rho0, c,
143 panel.eta ) ), 'LineStyle', '-' );
144 plot( f, -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s, rho0, c, panel.eta ) ), '
145 LineStyle', '-' );
146 %
147 for phi = phi_set
148     % plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_term1( rho0, c, phi ) ./ ( h_tau_term2(
149 rho0, c, phi, D, eta, f ) + h_tau_term3( f, ms, D, phi ) ) ), 'Color', 'r', 'LineStyle', '-',
150 'Marker', 'none' );
151     % plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_flexible_panel( f, rho0, c, phi, D
152 , panel.eta ) ), 'LineStyle', '--', 'Marker', 'none' );
153     plot( f, -10*log10( h_tau_term1( rho0, c, phi ) ./ ( h_tau_term2( rho0, c, phi, D, eta, f
154 ) + h_tau_term3( f, ms, D, phi ) ) ), 'Color', 'r', 'LineStyle', '-', 'Marker', 'none' );
155     % plot( f, -10*log10( h_tau_infinite_flexible_panel( f, rho0, c, phi, D, panel.eta ) ), '
156 LineStyle', '--', 'Marker', 'none' );
157
158     t_set = [ t_set; h_tau_infinite_flexible_panel( f, rho0, c, phi, D, panel.eta ) ];
159
160 end
161
162 grid on;
163
164 xlabel( 'Frequency [ $\frac{\omega}{\omega_o}$ ] ' ); ylabel( 'Transmission Loss [dB] ' );
165 title( 'Measured Panel Transmission Losses' );
166 set( gca, 'XScale', 'log' );
167 axis( [ 2e-3 100e3 -5 70 ] );
168
169 close( h1 );
170
171 N_phi = size( t_set, 1 );
172
173 temp1 = t_set;
174 temp2 = sind( 2*phi_set ).';
175 % temp2 = sind( phi_set ).';
176
177 temp3 = t_set .* repmat( temp2, 1, size( temp1, 2 ) );
178
179 tau_d = 1/N_phi .* nansum( temp3, 1 );
180
181 tau_d_verify = nanmean( t_set .* temp2, 1 );
182
183
184 h2 = figure( ); ...
185
186 hold on;
187
188 stem( octave_band_frequencies, TL, 'LineWidth', 0.5, 'Marker', 'o', 'MarkerSize', 8, '
189 MarkerEdgeColor', 'b', 'MarkerFaceColor', 'r' ); hold on;
190
191 line( [ 16e3 16e3 ], [ 0 65 ], 'Color', 'b', 'LineWidth', 1.5 );
192
193 plot( f, -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s, rho0, c, panel.eta ) ), '
194 LineStyle', '-' );
195
196 for phi = phi_set
197     plot( f, -10*log10( h_tau_term1( rho0, c, phi ) ./ ( h_tau_term2( rho0, c, phi, D, eta, f
198 ) + h_tau_term3( f, ms, D, phi ) ) ), 'Color', 'r', 'LineStyle', '-', 'Marker', 'none' );
199     % plot( f, -10*log10( h_tau_infinite_flexible_panel( f, rho0, c, phi, D, panel.eta ) ), '
200 LineStyle', '--', 'Marker', 'none' );
201 end
202
203 plot( f, -10*log10( tau_d ), 'LineStyle', '-', 'Marker', 'none', 'Color', 'm', 'LineWidth',
204 1.2 );
205 plot( f, -10*log10( tau_d_verify ), 'LineStyle', '-', 'Marker', 'none', 'Color', 'k', '
206 LineWidth', 1.2 );
207
208 plot( f, -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s, rho0, c, panel.eta ) ./ panel.
209 eta * ( 4*panel.length / ( panel.length^2 * critical_frequency ) ) ), 'LineStyle', '-', '
210 Marker', 'none', 'Color', 'k', 'LineWidth', 1.2 );
211
212 grid on;
213
214 xlabel( 'Frequency [ $\frac{\omega}{\omega_o}$ ] ' ); ylabel( 'Transmission Loss [dB] ' );
215 title( 'Measured Panel Transmission Losses' );

```

```

205     set( gca, 'XScale', 'log' );
206     axis( [ 2e-3 200e3 -5 90 ] );
207
208     close( h2 );
209
210 % return
211
212 %% Final Plot
213
214 figure( ); ...
215
216     hold on;
217
218     stem( octave_band_frequencies, TL, 'LineWidth', 0.5, 'Marker', 'o', 'MarkerSize', 8, '
MarkerEdgeColor', 'b', 'MarkerFaceColor', 'r' ); hold on;
219
220     plot( f, -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s, rho0, c, panel.eta ) ), '
LineStyle', '-' );
221
222     plot( f, -10*log10( tau_d ), 'LineStyle', '-', 'Marker', 'none', 'Color', 'm', 'LineWidth',
1.2 );
223 % plot( f, -10*log10( tau_d_verify ), 'LineStyle', '-', 'Marker', 'none', 'Color', 'k', '
LineWidth', 1.2 );
224
225     plot( f, -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s, rho0, c, panel.eta ) ./ (200*
panel.eta) * ( 4*panel.length / ( panel.length^2 * critical_frequency ) ) ), 'LineStyle', '-',
'Marker', 'none', 'Color', 'k', 'LineWidth', 1.2 );
226
227     line( [ 16e3 16e3 ], [ 0 65 ], 'Color', 'b', 'LineWidth', 1.5 );
228
229     grid on;
230
231     legend( ...
232         'Target Transmission Loss', ...
233         'Infinite Rigid Panel', ...
234         'Infinite Flexible Panel with Diffuse Incidence', ...
235         'Finite Flexible Panel Model' );
236
237     xlabel( 'Frequency [ $\frac{\omega}{\omega_o}$ ]' ); ylabel( 'Transmission Loss [dB]' );
238     title( 'Measured Panel Transmission Losses' );
239     set( gca, 'XScale', 'log' );
240     axis( [ 2e-3 200e3 -5 90 ] );
241
242
243
244 %% Plot Data and Model – Different Side Materials
245
246 % f = 1e-2:1e-2:20e3;
247 %
248 % phi = 15;
249 % eta = panel.eta;
250 %
251 %
252 % figure( ); ...
253 %     plot( f ./ ( wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel_side_materials( f, wo, ms,
s, rho0, c, panel.eta, 1 ) ), 'LineStyle', '-' ); hold on;
254 %     plot( f ./ ( wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel_side_materials( f, wo, ms,
s, rho0, c, panel.eta, 1/3600 ) ), 'LineStyle', '-' );
255 %     plot( f ./ ( wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel_side_materials( f, wo, ms,
s, rho0, c, panel.eta, 3600 ) ), 'LineStyle', '-' ); grid on;
256 %
257 %     legend( ...
258 %         'Same Fluid', ...
259 %         'Water to Air', ...
260 %         'Air to Water', ...
261 %         'Location', 'North' );
262 %
263 %     xlabel( 'Frequency [ $\frac{\omega}{\omega_o}$ ]' ); ylabel( 'Transmission Loss [dB]' );
264 %     title( 'Measured Panel Transmission Losses' );
265 %     set( gca, 'XScale', 'log' );
266 %     % axis( [ 40 12e3 -5 45] );
267
268
269
270 %% Change in Stiffness
271
272 % figure( ); ...
273 %     stem( octave_band_frequencies ./ ( wo / (2*pi) ), TL, 'LineWidth', 0.5, 'Marker', 'o', '

```

```

MarkerSize', 8, 'MarkerEdgeColor', 'b', 'MarkerFaceColor', 'r' ); hold on;
274 %
275 %     plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s, rho0, c,
panel.eta ) ), 'LineStyle', '-' );
276 %     plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s*100, rho0, c
, panel.eta ) ), 'LineStyle', '--' );
277 %     plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s*1e-2, rho0,
c, panel.eta ) ), 'LineStyle', '-.-' );
278 %
279 %     legend( ...
280 %         'Target TL Values', ...
281 %         'Infinite Rigid Panel with Normal Incidence Sound', ...
282 %         'Infinite Rigid Panel with Normal Incidence Sound (s * 100)', ...
283 %         'Infinite Rigid Panel with Normal Incidence Sound (s / 100)', ...
284 %         'Location', 'North' );
285 %
286 %     xlabel( 'Frequency [ $\frac{\omega}{\omega_o}$ ] ' ); ylabel( 'Transmission Loss [dB]' );
287 %     title( 'Measured Panel Transmission Losses - Change in Stiffness' );
288 %     set( gca, 'XScale', 'log' );
289 %     % axis( [ 40 12e3 -5 45] );
290
291
292


---


293 %% Change in Mass
294
295 % figure( ); ...
296 %     stem( octave_band_frequencies ./ (wo / (2*pi) ), TL, 'LineWidth', 0.5, 'Marker', 'o', '
MarkerSize', 8, 'MarkerEdgeColor', 'b', 'MarkerFaceColor', 'r' ); hold on;
297 %
298 %     plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel( f, wo, ms*100, s, rho0, c
, panel.eta ) ), 'LineStyle', '-' );
299 %     plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s, rho0, c,
panel.eta ) ), 'LineStyle', '-' );
300 %     plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel( f, wo, ms*1e-2, s, rho0,
c, panel.eta ) ), 'LineStyle', '-.-' );
301 %
302 %     legend( ...
303 %         'Target TL Values', ...
304 %         'Infinite Rigid Panel with Normal Incidence Sound', ...
305 %         'Infinite Rigid Panel with Normal Incidence Sound (s * 100)', ...
306 %         'Infinite Rigid Panel with Normal Incidence Sound (s / 100)', ...
307 %         'Location', 'North' );
308 %
309 %     xlabel( 'Frequency [ $\frac{\omega}{\omega_o}$ ] ' ); ylabel( 'Transmission Loss [dB]' );
310 %     title( 'Measured Panel Transmission Losses - Change in Mass' );
311 %     set( gca, 'XScale', 'log' );
312 %     % axis( [ 40 12e3 -5 45] );
313
314
315


---


316 %% Change in Loss Factor
317
318 % figure( ); ...
319 %     stem( octave_band_frequencies ./ (wo / (2*pi) ), TL, 'LineWidth', 0.5, 'Marker', 'o', '
MarkerSize', 8, 'MarkerEdgeColor', 'b', 'MarkerFaceColor', 'r' ); hold on;
320 %
321 %     plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s, rho0, c,
panel.eta ) ), 'LineStyle', '-' );
322 %     plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s, rho0, c,
panel.eta*1e2 ) ), 'LineStyle', ':' );
323 %     plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s, rho0, c,
panel.eta*1e-2 ) ), 'LineStyle', ':' );
324 %
325 %     legend( ...
326 %         'Target TL Values', ...
327 %         'Infinite Rigid Panel with Normal Incidence Sound', ...
328 %         'Infinite Rigid Panel with Normal Incidence Sound (eta * 100)', ...
329 %         'Infinite Rigid Panel with Normal Incidence Sound (eta / 100)', ...
330 %         'Location', 'North' );
331 %
332 %     xlabel( 'Frequency [ $\frac{\omega}{\omega_o}$ ] ' ); ylabel( 'Transmission Loss [dB]' );
333 %     title( 'Measured Panel Transmission Losses - Change in Loss Factor' );
334 %     set( gca, 'XScale', 'log', 'YScale', 'log' );
335 %     % axis( [ 40 12e3 -5 45] );
336
337
338


---


339 %% Clean-up

```

```

340
341 % return
342
343 if ( ~isempty( findobj( 'Type', 'figure' ) ) )
344     monitors = get( 0, 'MonitorPositions' );
345     if ( size( monitors, 1 ) == 1 )
346         autoArrangeFigures( 2, 2, 1 );
347     elseif ( 1 < size( monitors, 1 ) )
348         autoArrangeFigures( 2, 2, 1 );
349     end
350 end
351
352
353 fprintf( 1, '\n\n*** Processing Complete ***\n\n' );
354
355
356
357 %% 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 survice
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)
```
