Subject: ACS 547, Noise Control Applications - Module 2 Assignment

Date: February 23, 2025 (Submitted)

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.

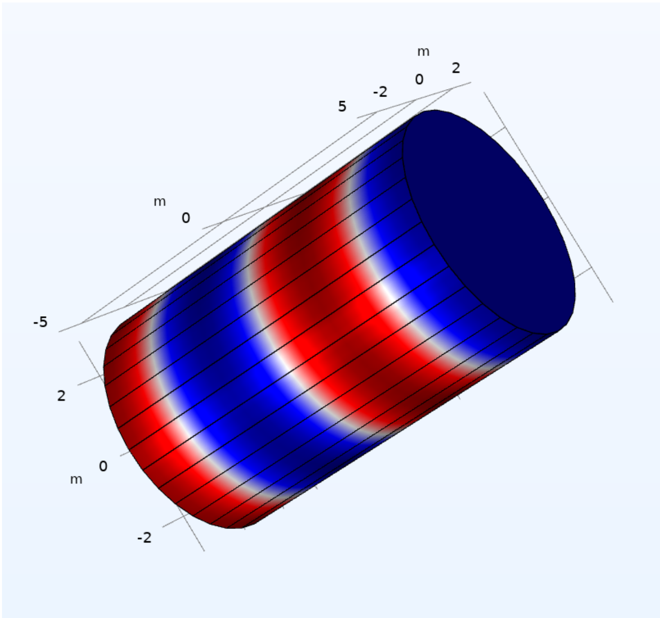
Problem 1b

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

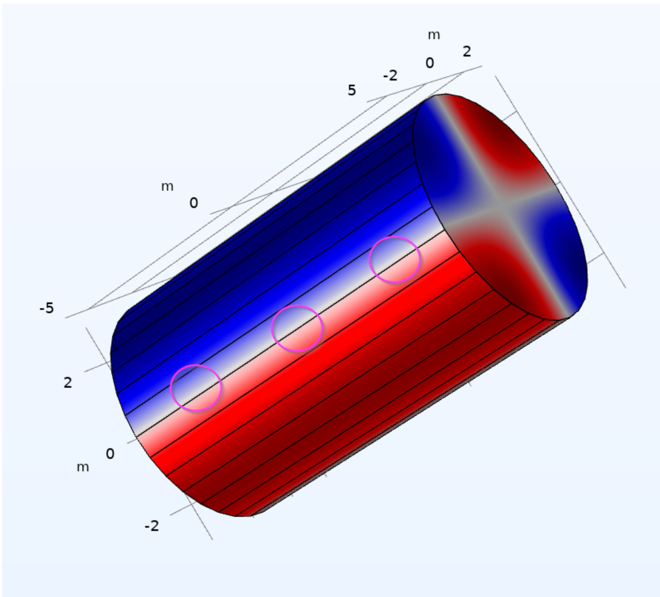
Problem 1c

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 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.

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 $\frac{\mathrm{m}^3}{\mathrm{s}}$.

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

vspace0.25cm

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

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

```matlab



%% Synopsis

% Problem 1 — Modal Behaviour of a Cylindrical Room



%% Environment

close all; clear; clc;
% restoredefaultpath;

% addpath( genpath( '' ), '-begin' );
addpath( genpath( './00 Support' ), '-begin' );

% set( 0, 'DefaultFigurePosition', [ 400 400 900 400 ] ); % [ left bottom width height ]
set( 0, 'DefaultFigurePaperPositionMode', 'manual' );
set( 0, 'DefaultFigureWindowStyle', 'normal' );
set( 0, 'DefaultLineLineWidth', 1.5 );
set( 0, 'DefaultTextInterpreter', 'Latex' );

format ShortG;

pause( 1 );



%% Define Cylindrical Room

room.radius = 3; % m
room.length = 10; % m



%% Test Circular Mode Function

% psi = circular_mode_shape( 3, 1, 2, false ); % 3.7261 — CHECKED FROM CLASS (PLOT NOT CREATED)
% psi = circular_mode_shape( 3, 1, 2, true ); % 3.7261 — CHECKED FROM CLASS (PLOT CREATED)



%% Define Anonymous Function for the Natural Frequencies

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 );



%% Calculate the Natural Frequencies

% The maximum number of radial modes is 5 (indexed from 0 to 4).
% The maximum number of angular modes is 8 (indexed from 0 to 7).

NX_SIZE = 20;
NTHETA_SIZE = 7;
NR_SIZE = 4;
    natural_frequencies = nan( NX_SIZE, NTHETA_SIZE, NR_SIZE );

for nx = 0:1:NX_SIZE
    for ntheta = 0:1:NTHETA_SIZE
        for nr = 0:1:NR_SIZE
            natural_frequencies( nx+1, ntheta+1, nr+1 ) = h_natural_frequencies( 343, nx, ntheta,
                nr, 10, 3, false );
        end
    end
end



%% Part a — Find 10 Lowest Resonance Frequencies

NUMBER_OF_LOWEST_FREQUENCIES = 11;
```

```
    mode_indices =  [ 1:1:NUMBER_OF_LOWEST_FREQUENCIES ].';

[ sortedValues, sortedIndices ] = sort( natural_frequencies(:) );  % 21-by-8-by-5 -> 840 elements

smallestValues = sortedValues( 1:NUMBER_OF_LOWEST_FREQUENCIES );
    % [ mode_indices   round( smallestValues, 1 ) ]
    %
    % 1            0
    % 2         17.2
    % 3         33.5
    % 4         34.3
    % 5         37.6
    % 6         47.9
    % 7         51.5
    % 8         55.6
    % 9         58.2
    % 10         61.4
    % 11         65.3

smallestIndices = sortedIndices( 1:NUMBER_OF_LOWEST_FREQUENCIES );

[ x, y, z ] = ind2sub( size(natural_frequencies), smallestIndices );
    [ x y z ] - 1;

% Verify the calculated mode indices.
h_natural_frequencies( 343, 0, 0, 0, 10, 3, false );  % 0 Hz
h_natural_frequencies( 343, 1, 0, 0, 10, 3, false );  % 17.2 Hz
h_natural_frequencies( 343, 0, 1, 0, 10, 3, false );  % 33.5 Hz
h_natural_frequencies( 343, 2, 0, 0 , 10, 3, false );  % 34.3 Hz
h_natural_frequencies( 343, 1, 1, 0 , 10, 3, false );  % 37.6 Hz
h_natural_frequencies( 343, 2, 1, 0 , 10, 3, false );  % 48.0 Hz
h_natural_frequencies( 343, 3, 0, 0 , 10, 3, false );  % 51.5 Hz
h_natural_frequencies( 343, 0, 2, 0 , 10, 3, false );  % 55.6 Hz
h_natural_frequencies( 343, 1, 2, 0 , 10, 3, false );  % 58.2 Hz
h_natural_frequencies( 343, 3, 1, 0 , 10, 3, false );  % 61.4 Hz
h_natural_frequencies( 343, 2, 2, 0 , 10, 3, false );  % 65.3 Hz



%% Part b - Two

[ (1:11).'  abs( smallestValues - 53 ) ]

temp = [ x y z ] - 1;  temp( 7:8, :, : )

h_natural_frequencies( 343, 3, 0, 0, 10, 3, false );  % 51.5 Hz, (3, 0, 0)
h_natural_frequencies( 343, 0, 2, 0, 10, 3, false );  % 55.6 Hz, (0, 2, 0)



%% Part c

% See the report.



%% Clean-up

if ( ~isempty( findobj( 'Type', 'figure' ) ) )
    monitors = get( 0, 'MonitorPositions' );
        if ( size( monitors, 1 ) == 1 )
            autoArrangeFigures( 2, 2, 1 );
        elseif ( 1 < size( monitors, 1 ) )
            autoArrangeFigures( 2, 2, 1 );
        end
end


fprintf( 1, '\n\n\n*** Processing Complete ***\n\n\n' );



%% Reference(s)

% 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

```matlab



%% Synopsis

% ACS 547, Sabine Room Milestone

% See slide 22 on "Lecture 07 - Sabine rooms - Filled.pptx".



%% Environment

close all; clear; clc;
% restoredefaultpath;

% addpath( genpath( '' ), '-begin' );
addpath( genpath( '../00 Support' ), '-begin' );

% set( 0, 'DefaultFigurePosition', [  400  400  900  400  ] );  % [ left bottom width height ]
set( 0, 'DefaultFigurePaperPositionMode', 'manual' );
set( 0, 'DefaultFigureWindowStyle', 'normal' );
set( 0, 'DefaultLineLineWidth', 1.5 );
set( 0, 'DefaultTextInterpreter', 'Latex' );

format ShortG;

pause( 1 );

PRINT_FIGURES = 0;



%% Information

room.width = 8;  room.length = 6;  room.height = 3;  % meters
    room_volume = room.width * room.length * room.height;  % 144 m^3
    room_area = 2*(room.width*room.height)  +  2*(room.length*room.height)  + 2*(room.width*room.length);  % 180 m^2

alpha_average_walls_and_floor = 0.05;  % For the walls and the floor.
alpha_average_ceiling = 0.15;  % For the ceiling.
%
% For the 125 Hz octave band.



%% Part a - Estimate the reverberant sound pressure level.

Lw = 10*log10( 25e-3 / 1e-12 );  % 103.98 dB

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

room_constant = room_area * average_absorption_coefficient / ( 1 - average_absorption_coefficient );  % 14.9 m^2 or Sabines


sound_pressure_level = Lw + 10*log10( 4 / room_constant );  % 98.3 dB



%% Part b

D0 = 1;

r = 0:0.05:12;  % meters

h_Lp_direct = @( Lw, D0, r )  Lw + 10*log10( D0./(4.*pi.*r.^2) );
h_Lp_reverberant = @( Lw, room_constant )  Lw + 10*log10( 4 ./ room_constant );
%
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 );

```

```
figure( ); ...
    plot( r, h_Lp_direct( Lw, D0, r ) );  hold on;
    plot( r, ones( size(r) ).*h_Lp_reverberant( Lw, room_constant) );
    plot( r, h_Lp_net( Lw, D0, r, room_volume ));  grid on;
        legend( 'Direct $L_p$', 'Reverberant $L_p$', 'Total $L_p$', 'Interpreter', 'Latex' );
    %
    text( 0.545, 100, 'Critical Distance $\approx$ 0.55 meters.', 'Interpreter', 'Latex' );
    %
    xlabel( 'Distance from Source [meters]' );  ylabel( 'Sound Pressure Level [dB re:20e-6 Pascals]' );
    title( 'Sound Pressure Components from Direct and Reverberant Fields from 125 Hz Point Source' );
    %
    set( gca, 'XScale', 'log' );


% Estimate the critical distance (see page 84 of "06-Indoors.pdf" notes for ACS 537).
rc = 0.141 * sqrt( D0 * room_constant );  % 0.5451 meters



%% Clean-up

if ( ~isempty( findobj( 'Type', 'figure' ) ) )
    monitors = get( 0, 'MonitorPositions' );
        if ( size( monitors, 1 ) == 1 )
            autoArrangeFigures( 2, 2, 1 );
        elseif ( 1 < size( monitors, 1 ) )
            autoArrangeFigures( 2, 2, 1 );
        end
end


fprintf( 1, '\n\n\n*** Processing Complete ***\n\n\n' );



%% Reference(s)

%% Overall, A-weighted Level

% Note(s):
%
%   The above analysis was done using the unweighted sound level for the 500 Hz octave band.
%
%   If an overall level is to be calculated (i.e., across a set of octave bands), then this analysis
%   must be done for all octave band center frequencies.  Once the unweighted sound pressure
%   levels at the location are determned, the respective octave band A-weighting offsets are
%   applied.
%
%   The overall A-weighted sound pressure levels is then calculated logarithmically add using the
%   expression,
%
%   10*log10( sum( 10^(Lp_a/10) )
```

3 Appendix - Matlab Code for Problem 3

```
%% Synopsis

% Question 3 - Transmission Loss Measurement


% Reference(s):
%
% Slide 8 - Noise Reduction and Transmission Loss


% Assumptions:
%
%   1.)  Lp1 depends on the transmission loss.
%
%        If the transmission loss is low then more energy goes to room 2 (i.e., the receiver room)
%        .
%
%        The noise reduction from the source room to the receiver.
%
%        Adding the barrier will change the level in the source room.  Typically making the sound
%        level higher in the source room.



%% Environment

close all; clear; clc;
% restoredefaultpath;

% addpath( genpath( '' ), '-begin' );
addpath( genpath( '../00 Support' ), '-begin' );

% set( 0, 'DefaultFigurePosition', [  400  400  900  400  ] );  % [ left bottom width height ]
set( 0, 'DefaultFigurePaperPositionMode', 'manual' );
set( 0, 'DefaultFigureWindowStyle', 'normal' );
set( 0, 'DefaultLineLineWidth', 1.0 );
set( 0, 'DefaultLineMarker', 'x' );
set( 0, 'DefaultLineMarkerSize', 15 );
% set( 0, 'DefaultAxesLineStyleOrder', { '-' '--o' '+' } );
set( 0, 'DefaultTextInterpreter', 'Latex' );

format ShortG;

pause( 1 );

PRINT_FIGURES = 0;



%% Dimensions of Rooms and Panel

room.length = 4;  % meters
room.width = 4;  % meters
room.height = 4;  % meters
    room.volume = room.length * room.width * room.height;  % 64 m^2
    room.area = 2*(room.length * room.width) + 2*(room.length * room.height) + 2*(room.width *
        room.height);  % 96 m^2

panel.width = 0.8;  % meters
panel.height = 0.8;  % meters
    panel.area = panel.width * panel.height;  % 0.64 m^2



%% Data

octave_band_frequencies = [ 125  250  500  1000  2000  4000 ].';  % Hz
T60 = [ 2.0  2.1  1.8  1.5  1.2  0.9 ].';  % seconds
spl.source_room = [ 90  95  103  105  100  93 ].';  % dB re: 20e-6 Pascals
spl.receiver_room = [ 50  50  46  50  50  38 ].';  % dB re: 20e-6 Pascals

c = 343;  % meters per second
```

```matlab



%% Pressure Difference

spl.delta = spl.source_room - spl.receiver_room;

% figure( ); ...
%     stem( octave_band_frequencies, spl.delta, 'Marker', '.', 'MarkerSize', 12, 'Color', 'r' ); grid on;
%     xlabel( 'Frequency [Hz]' ); ylabel( 'Transmission Loss [dB]' );
%     title( 'Pressure Difference Versus Octave Band Center Frequency' );
%     %
%     xticks( octave_band_frequencies ); xticklabels( num2cell( octave_band_frequencies ) );
%     set( gca, 'XScale', 'log' );
%     xlim( [ 80 6e3 ] ); ylim( [ 0 65 ] );

% https://www.mathworks.com/matlabcentral/answers/413686-how-to-set-log-scale-range



%% Determine Average Absorption in the Receiver Room using Reverberation Time Measurements

average_absorption = @( volume, area, c, T60 ) ( 55.25 .* volume ) ./ ( area .* c .* T60 );

receiver_room.average_absorption = average_absorption( room.volume, room.area, c, T60 );

% Assumption: Calibration panel has very high transmission loss.

% figure( ); ...
%     stem( octave_band_frequencies, receiver_room.average_absorption, 'Marker', '.', 'MarkerSize', 12, 'Color', 'r' ); grid on;
%     xlabel( 'Frequency [Hz]' ); ylabel( 'Average Absorption [Sabine]' );
%     title( 'Average Absorption in Receiver Room Versus Octave Band Center Frequency' );
%     %
%     xticks( octave_band_frequencies ); xticklabels( num2cell( octave_band_frequencies ) );
%     set( gca, 'XScale', 'log' );
%     xlim( [ 80 6e3 ] ); ylim( [ 0 0.14 ] );



%% Determine Receiver Room Constant

room_constant = @( average_absorption, area ) ( average_absorption * area ) ./ ( 1 - average_absorption ); % Unitless

receiver_room.room_constant = room_constant( receiver_room.average_absorption, room.area );



%% Determine the Transmission Loss in Each Octave Band

% The calibration plate isolate the receiver room and the area of the
% receiver room does not consider the calibration plate.

transmission_coefficient = @( receiver_room_pressure, source_room_pressure, panel_area, receiver_room_constant ) ( ( receiver_room_pressure ./ source_room_pressure ) .* receiver_room_constant ) ./ panel_area;

tau = transmission_coefficient( 10.^(spl.receiver_room./10)*20e-6, 10.^(spl.source_room/10)*20e-6, panel.area, receiver_room.room_constant );

TL = -10*log10( tau );

figure( ); ...
    stem( octave_band_frequencies, TL, '.', 'MarkerSize', 15, 'Color', 'r' ); grid on;
    xlabel( 'Frequency [Hz]' ); ylabel( 'Transmission Loss [dB]' );
    title( 'Transmission Loss Per Octave Band' );
    %
    xticks( octave_band_frequencies ); xticklabels( num2cell( octave_band_frequencies ) );
    set( gca, 'XScale', 'log' );
    xlim( [ 80 6e3 ] ); ylim( [ 0 55 ] );



%% Validation

TL_verify = spl.source_room - spl.receiver_room + 10*log10( panel.area ./ receiver_room.
```

```
    room_constant )

return

%% Clean-up

if ( ~isempty( findobj( 'Type', 'figure' ) ) )
    monitors = get( 0, 'MonitorPositions' );
        if ( size( monitors, 1 ) == 1 )
            autoArrangeFigures( 2, 2, 1 );
        elseif ( 1 < size( monitors, 1 ) )
            autoArrangeFigures( 2, 2, 1 );
        end
end


fprintf( 1, '\n\n\n*** Processing Complete ***\n\n\n' );



%% Reference(s)
```

4 Appendix - Matlab Code for Problem 4

```
%% Synopsis

% Slide 8 - Noise Reduction and Transmission Loss



%% Environment

% close all; clear; clc;
% restoredefaultpath;

% addpath( genpath( '' ), '-begin' );
addpath( genpath( '../40 Assignments/00 Support' ), '-begin' );

% set( 0, 'DefaultFigurePosition', [ 400 400 900 400 ] ); % [ left bottom width height ]
set( 0, 'DefaultFigurePaperPositionMode', 'manual' );
set( 0, 'DefaultFigureWindowStyle', 'normal' );
set( 0, 'DefaultLineLineWidth', 0.8 );
set( 0, 'DefaultTextInterpreter', 'Latex' );

format ShortG;

pause( 1 );

PRINT_FIGURES = 0;



%% Parameters

c = 343; % m/s
rho0 = 1.21; % kg

panel.length = 80e-2; % meters
%
panel.E = 200e9; % Pascals
panel.density = 7800; % kg / m^3
panel.v = 0.29; % Poisson's Ratio (unitless)
panel.thickness = 1.2e-3; % m
panel.eta = 0.001; % Loss factor (unitless)



%% Panel Data

octave_band_frequencies = [ 63 125 250 500 1000 2000 4000 8000 ].'; % Hz
TL = [ 9 14 21 27 32 37 43 42 ].'; % dB


% figure( ); ...
%     stem( octave_band_frequencies, TL, 'LineWidth', 0.5, 'Marker', 'o', 'MarkerSize', 8, 'MarkerEdgeColor', 'b', 'MarkerFaceColor', 'r' ); grid on;
%     xlabel( 'Frequency [Hz] ' ); ylabel( 'Transmission Loss [dB]' );
%     title( 'Measured Panel Transmission Losses' );
%     set( gca, 'XScale', 'log' );
%     axis( [ 40 12e3 -5 45] );



%% Problem 4a - Infinite, Rigid Panel Model with Normal Incidence

D = ( panel.E * panel.thickness.^3 ) / ( 12 * ( 1 - panel.v^2 ) ); % 31.4

% From lecture 9 on Wednesday, February 12, 2025, the equivalent bending
% moment of the panel is a half-wavelength.
%
wavelength = 2 * panel.length; % 1.6 meters

ms = panel.density * panel.thickness;

wo = pi^2 / panel.length * sqrt( D / ms );
    s = wo^2 * ms;
```

```
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 );

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 );



%% Problem 4b - Infinite, Flexible Panel Model with Random Incidence

% Panel has bending waves.


h_tau_term1 = @( rho0, c, phi )  ( 2*rho0*c*secd(phi) ).^2;  % Checked
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
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


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 );

% return

%% Plot Data and Model - Air on Both Sides

% 75 degrees from normal to panel.

h_c_bending_wave = @( D, f, ms )  ( (D*(2*pi.*f).^2) ./ ms ).^0.25;
%
%   Proportional to the square-root of frequency.
%
%       Small wavelenths (high frequencies;  arrive first) travel faster than long wavelength (low frequencies;  arrive later).

phi = 75;
    h_coincidence_frequency =  @( ms, D, c, phi )  1./(2*pi) * sqrt( ms / D ) .* ( c ./ sind( phi )).^2;  % 10,949 Hz

    h_coincidence_frequency( ms, D, c, phi );  % 10,949 Hz

    [ (0:15:90).'  h_coincidence_frequency( ms, D, c, [ 0:15:90 ] ).' ];


    critical_frequency =  1./(2*pi) .* sqrt( ms / D ) .* ( c / sind( 90 )).^2  % 10,216 Hz
        critical_frequency_verify_1 = c^2 / (2*pi) * sqrt( ms / D );  % lowest coincidence frequency
        critical_frequency_verify_2 = c^2 / ( 1.8 * panel.thickness * sqrt( panel.E / ( panel.density * ( 1 - panel.v^2) ) ) );




% f = 1e-2:1e-2:40e3;
f = 1e-2:1:100e3;


eta = panel.eta;

phi_set = 0:10:90;

[ phi_set.'  h_coincidence_frequency( ms, D, c, phi_set ).' ]

t_set = [ ];


h1 = figure( ); ...

    hold on;
    % stem( octave_band_frequencies ./ (wo / (2*pi) ), TL, 'LineWidth', 0.5, 'Marker', 'o', 'MarkerSize', 8, 'MarkerEdgeColor', 'b', 'MarkerFaceColor', 'r' );  hold on;
    stem( octave_band_frequencies, TL, 'LineWidth', 0.5, 'Marker', 'o', 'MarkerSize', 8, 'MarkerEdgeColor', 'b', 'MarkerFaceColor', 'r' );  hold on;
    line( [ 16e3 16e3 ], [ 0 65 ], 'Color', 'c' );
    %
```

```
    % plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s, rho0, c, panel.eta ) ), 'LineStyle', '-' );
    plot( f, -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s, rho0, c, panel.eta ) ), 'LineStyle', '-' );
    %

    for phi = phi_set
        % 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' );
        % plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_flexible_panel( f, rho0, c, phi, D , panel.eta ) ), 'LineStyle', '--', 'Marker', 'none' );
        plot( f, -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' );
        % plot( f, -10*log10( h_tau_infinite_flexible_panel( f, rho0, c, phi, D, panel.eta ) ), 'LineStyle', '--', 'Marker', 'none' );

        t_set = [ t_set;  h_tau_infinite_flexible_panel( f, rho0, c, phi, D, panel.eta ) ];

    end

    grid on;

    xlabel( 'Frequency [$\frac{\omega}{\omega_o}$] ' );  ylabel( 'Transmission Loss [dB]' );
    title( 'Measured Panel Transmission Losses' );
    set( gca, 'XScale', 'log' );
    axis( [ 2e-3 100e3 -5 70 ] );

    close( h1 );


N_phi = size( t_set, 1 );

temp1 = t_set;
temp2 = sind( 2*phi_set ).';
% temp2 = sind( phi_set ).';

temp3 = t_set .* repmat( temp2, 1, size( temp1, 2 ) );

tau_d = 1/N_phi .* nansum( temp3, 1 );

tau_d_verify = nanmean( t_set .* temp2, 1 );



h2 = figure( ); ...

    hold on;

    stem( octave_band_frequencies, TL, 'LineWidth', 0.5, 'Marker', 'o', 'MarkerSize', 8, 'MarkerEdgeColor', 'b', 'MarkerFaceColor', 'r' );  hold on;

    line( [ 16e3 16e3 ], [ 0 65 ], 'Color', 'b', 'LineWidth', 1.5 );

    plot( f, -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s, rho0, c, panel.eta ) ), 'LineStyle', '-' );

    for phi = phi_set
        plot( f, -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' );
        % plot( f, -10*log10( h_tau_infinite_flexible_panel( f, rho0, c, phi, D, panel.eta ) ), 'LineStyle', '--', 'Marker', 'none' );
    end

    plot( f, -10*log10( tau_d ), 'LineStyle', '-', 'Marker', 'none', 'Color', 'm', 'LineWidth', 1.2 );
    plot( f, -10*log10( tau_d_verify ), 'LineStyle', '-', 'Marker', 'none', 'Color', 'k', 'LineWidth', 1.2 );


    plot( f, -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s, rho0, c, panel.eta ) ./ panel.eta * ( 4*panel.length / ( panel.length^2 * critical_frequency ) ) ), 'LineStyle', '-', 'Marker', 'none', 'Color', 'k', 'LineWidth', 1.2 );

    grid on;

```

```matlab
    xlabel( 'Frequency [$\frac{\omega}{\omega_o}$] ' );  ylabel( 'Transmission Loss [dB]' );
    title( 'Measured Panel Transmission Losses' );
    set( gca, 'XScale', 'log' );
    axis( [ 2e-3 200e3 -5 90 ] );

    close( h2 );

% return

%% Final Plot

figure( ); ...

    hold on;

    stem( octave_band_frequencies, TL, 'LineWidth', 0.5, 'Marker', 'o', 'MarkerSize', 8, 'MarkerEdgeColor', 'b', 'MarkerFaceColor', 'r' );  hold on;

    plot( f, -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s, rho0, c, panel.eta ) ), 'LineStyle', '-' );

    plot( f, -10*log10( tau_d ), 'LineStyle', '-', 'Marker', 'none', 'Color', 'm', 'LineWidth', 1.2 );
   % plot( f, -10*log10( tau_d_verify ), 'LineStyle', '-', 'Marker', 'none', 'Color', 'k', 'LineWidth', 1.2 );

    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 );

    line( [ 16e3 16e3 ], [ 0 65 ], 'Color', 'b', 'LineWidth', 1.5 );

    grid on;

    legend( ...
        'Target Transmission Loss', ...
        'Infinite Rigid Panel', ...
        'Infinite Flexible Panel with Diffuse Incidence', ...
        'Finite Flexible Panel Model' );

    xlabel( 'Frequency [$\frac{\omega}{\omega_o}$] ' );  ylabel( 'Transmission Loss [dB]' );
    title( 'Measured Panel Transmission Losses' );
    set( gca, 'XScale', 'log' );
    axis( [ 2e-3 200e3 -5 90 ] );



%% Plot Data and Model - Different Side Materials

% f = 1e-2:1e-2:20e3;
%
% phi = 15;
% eta = panel.eta;
%
%
% figure( ); ...
%     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;
%     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', '-' );
%     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;
%         %
%         legend( ...
%             'Same Fluid', ...
%             'Water to Air', ...
%             'Air to Water', ...
%             'Location', 'North' );
%     %
%     xlabel( 'Frequency [$\frac{\omega}{\omega_o}$] ' );  ylabel( 'Transmission Loss [dB]' );
%     title( 'Measured Panel Transmission Losses' );
%     set( gca, 'XScale', 'log' );
%     % axis( [ 40 12e3 -5 45] );



%% Change is Stiffness

```

```
% figure( ); ...
%     stem( octave_band_frequencies ./ (wo / (2*pi) ), TL, 'LineWidth', 0.5, 'Marker', 'o', 'MarkerSize', 8, 'MarkerEdgeColor', 'b', 'MarkerFaceColor', 'r' );  hold on;
%     %
%     plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s, rho0, c, panel.eta ) ), 'LineStyle', '-' );
%     plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s*100, rho0, c, panel.eta ) ), 'LineStyle', '--' );
%     plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s*1e-2, rho0, c, panel.eta ) ), 'LineStyle', '--' );
%         %
%         legend( ...
%             'Target TL Values', ...
%             'Infinite Rigid Panel with Normal Incidence Sound', ...
%             'Infinite Rigid Panel with Normal Incidence Sound (s * 100)', ...
%             'Infinite Rigid Panel with Normal Incidence Sound (s / 100)', ...
%             'Location', 'North' );
%     %
%     xlabel( 'Frequency [$\frac{\omega}{\omega_o}$] ' );  ylabel( 'Transmission Loss [dB]' );
%     title( 'Measured Panel Transmission Losses - Change in Stiffness' );
%     set( gca, 'XScale', 'log' );
%     % axis( [ 40 12e3 -5 45] );



%% Change in Mass

% figure( ); ...
%     stem( octave_band_frequencies ./ (wo / (2*pi) ), TL, 'LineWidth', 0.5, 'Marker', 'o', 'MarkerSize', 8, 'MarkerEdgeColor', 'b', 'MarkerFaceColor', 'r' );  hold on;
%     %
%     plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel( f, wo, ms*100, s, rho0, c, panel.eta ) ), 'LineStyle', '-' );
%     plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s, rho0, c, panel.eta ) ), 'LineStyle', '-' );
%     plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel( f, wo, ms*1e-2, s, rho0, c, panel.eta ) ), 'LineStyle', '-' );
%         %
%         legend( ...
%             'Target TL Values', ...
%             'Infinite Rigid Panel with Normal Incidence Sound', ...
%             'Infinite Rigid Panel with Normal Incidence Sound (s * 100)', ...
%             'Infinite Rigid Panel with Normal Incidence Sound (s / 100)', ...
%             'Location', 'North' );
%     %
%     xlabel( 'Frequency [$\frac{\omega}{\omega_o}$] ' );  ylabel( 'Transmission Loss [dB]' );
%     title( 'Measured Panel Transmission Losses - Change in Mass' );
%     set( gca, 'XScale', 'log' );
%     % axis( [ 40 12e3 -5 45] );



%% Change in Loss Factor

% figure( ); ...
%     stem( octave_band_frequencies ./ (wo / (2*pi) ), TL, 'LineWidth', 0.5, 'Marker', 'o', 'MarkerSize', 8, 'MarkerEdgeColor', 'b', 'MarkerFaceColor', 'r' );  hold on;
%     %
%     plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s, rho0, c, panel.eta ) ), 'LineStyle', '-' );
%     plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s, rho0, c, panel.eta*1e2 ) ), 'LineStyle', ':' );
%     plot( f ./ (wo / (2*pi) ), -10*log10( h_tau_infinite_rigid_panel( f, wo, ms, s, rho0, c, panel.eta*1e-2 ) ), 'LineStyle', ':' );
%         %
%         legend( ...
%             'Target TL Values', ...
%             'Infinite Rigid Panel with Normal Incidence Sound', ...
%             'Infinite Rigid Panel with Normal Incidence Sound (eta * 100)', ...
%             'Infinite Rigid Panel with Normal Incidence Sound (eta / 100)', ...
%             'Location', 'North' );
%     %
%     xlabel( 'Frequency [$\frac{\omega}{\omega_o}$] ' );  ylabel( 'Transmission Loss [dB]' );
%     title( 'Measured Panel Transmission Losses - Change in Loss Factor' );
%     set( gca, 'XScale', 'log', 'YScale', 'log' );
%     % axis( [ 40 12e3 -5 45] );
```

```
%% Clean-up

% return

if ( ~isempty( findobj( 'Type', 'figure' ) ) )
    monitors = get( 0, 'MonitorPositions' );
        if ( size( monitors, 1 ) == 1 )
            autoArrangeFigures( 2, 2, 1 );
        elseif ( 1 < size( monitors, 1 ) )
            autoArrangeFigures( 2, 2, 1 );
        end
end


fprintf( 1, '\n\n\n*** Processing Complete ***\n\n\n' );



%% Reference(s)
```

5 Appendix - Matlab Code for Problem 5

```
%% Synopsis

% Slide 8 - Noise Reduction and Transmission Loss

% Volume of the enclosure is much bigger than the machine.  Diffuse sound field in the enclosure.



%% Environment

% close all; clear; clc;
% restoredefaultpath;

% addpath( genpath( '' ), '-begin' );
addpath( genpath( '../40 Assignments/00 Support' ), '-begin' );

% set( 0, 'DefaultFigurePosition', [  400  400  900  400  ] );  % [ left bottom width height ]
set( 0, 'DefaultFigurePaperPositionMode', 'manual' );
set( 0, 'DefaultFigureWindowStyle', 'normal' );
set( 0, 'DefaultLineLineWidth', 0.8 );
set( 0, 'DefaultTextInterpreter', 'Latex' );

format ShortG;

pause( 1 );

PRINT_FIGURES = 0;



%% Define Machine

machine.area = 3;  % m^2
machine.absorption = 0.07;  % Sabine
machine.D = 1;  % Unitless - In air.

machine.distance = 10;  % m



%% Data

octave_band_frequencies = [ 250  500  1000  2000  4000 ].';  % Hz
Lw = [ 105  115  106  108  119 ].';  % dB re: 1 pW
    % [ octave_band_frequencies  Lw ]

% figure( ); ...
%     h1 = stem( octave_band_frequencies, Lw, 'Marker', '.', 'MarkerSize', 12, 'Color', 'r' );
    hold on;
%     h2 = line( [ 2e2 5e3 ], [ 30 30 ] );  grid on;
%         legend( [ h1 h2 ], 'Current Sound Pressure Levels', 'Target Sound Pressure Level', '
    Location', 'North' );
%     xlabel( 'Frequency [Hz]' );  ylabel( 'Sound Pressure Level [dB re: 20e-6 Pa]' );
%     title( 'Sound Power Level Versus Octave Band Center Frequency' );
%     %
%     axis( [ 150 6e3  0 140 ] );
%     set( gca, 'XScale', 'log' );



%% Per Octave Band Insertion Loss

Lp_10_meters = Lw  +  10*log10( machine.D /( 4 * pi * machine.distance^2 ) );  % dB re: 20e-6 Pa
    % [ octave_band_frequencies  Lp_10_meters ]
%
%  The value of R is infinite.  The machine is outside in open air.

octave_band_IL = Lp_10_meters - 30;
    % [ octave_band_frequencies  octave_band_IL ]
```

```
%% Anonymous Function for Insertion Loss

h_IL_large = @( Sw, alpha_w, Si, alpha_i, TL ) 10*log10( 1 + (Sw*alpha_w + Si*alpha_i)./(Sw
    + Si)*10^(TL/10) );



%% Find Values of TL and Aborption that will Meet the Target Insertion Loss - Ground Reflecting

% Assumption(s):
%
% 1.) The ground is a hard reflecting survice
% 2.) The enclosure is a cube.
% 3.) There is no noise transmission through the ground.

enclosure.dimension = 2; % m
    enclosure.area = 6 * enclosure.dimension^2; % 20 m^2


% https://www.controlnoise.com/wp-content/uploads/2022/02/Acoustic-Enclosures-Datasheet.pdf
% https://www.cecoenviro.com/wp-content/uploads/2023/12/Acoustic-Enclosures-8pp-A4-web.pdf
% https://www.controlnoise.com/product/acoustic-enclosures/


switch ( 3 )

    case 1

        % 250 Hz - QBV-2
        alpha_w = 0.27; % From specification sheet.
        TL = 20; % From specification sheet.
            IL_estimates(1) = h_IL_large( enclosure.area, alpha_w, machine.area, machine.
                absorption, TL );

        % 500 Hz - QBV-2
        alpha_w = 0.96; % From specification sheet.
        TL = 29; % From specification sheet.
            IL_estimates(2) = h_IL_large( enclosure.area, alpha_w, machine.area, machine.
                absorption, TL );

        % 1 kHz - QBV-2
        % alpha_w = 1.13; % From specification sheet.
        alpha_w = 0.99; % From specification sheet. See comment on slide 28 of Lecture 10.
        TL = 40; % From specification sheet.
            IL_estimates(3) = h_IL_large( enclosure.area, alpha_w, machine.area, machine.
                absorption, TL );

        % 2 kHz - QBV-2
        % alpha_w = 1.08; % From specification sheet.
        alpha_w = 0.99; % From specification sheet. See comment on slide 28 of Lecture 10.
        TL = 50; % From specification sheet.
            IL_estimates(4) = h_IL_large( enclosure.area, alpha_w, machine.area, machine.
                absorption, TL );

        % 4 kHz - QBV-2
        alpha_w = 0.99; % From specification sheet.
        TL = 55; % From specification sheet.
            IL_estimates(5) = h_IL_large( enclosure.area, alpha_w, machine.area, machine.
                absorption, TL );


    case 2

        % Note: Abosrption values are carried over from QBV-2.

        % 250 Hz - QBV-3
        alpha_w = 0.27; % From specification sheet.
        TL = 25; % From specification sheet.
            IL_estimates(1) = h_IL_large( enclosure.area, alpha_w, machine.area, machine.
                absorption, TL );

        % 500 Hz - QBV-2
        alpha_w = 0.96; % From specification sheet.
        TL = 33; % From specification sheet.
            IL_estimates(2) = h_IL_large( enclosure.area, alpha_w, machine.area, machine.
                absorption, TL );

        % 1 kHz - QBV-2
```

```
        % alpha_w = 1.13;  % From specification sheet.
        alpha_w = 0.99;  % From specification sheet.  See comment on slide 28 of Lecture 10.
        TL = 46;  % From specification sheet.
            IL_estimates(3) = h_IL_large( enclosure.area, alpha_w, machine.area, machine.absorption, TL );

        % 2 kHz - QBV-2
        % alpha_w = 1.08;  % From specification sheet.
        alpha_w = 0.99;  % From specification sheet.  See comment on slide 28 of Lecture 10.
        TL = 53;  % From specification sheet.
            IL_estimates(4) = h_IL_large( enclosure.area, alpha_w, machine.area, machine.absorption, TL );

        % 4 kHz - QBV-2
        alpha_w = 0.99;  % From specification sheet.
        TL = 58;  % From specification sheet.
            IL_estimates(5) = h_IL_large( enclosure.area, alpha_w, machine.area, machine.absorption, TL );


    case 3

        % Note:  Abosrption values are carried over from QBV-2.

        % 250 Hz - QBV-3
        alpha_w = 0.99;  % From specification sheet.
        TL = 39;  % From specification sheet.
            IL_estimates(1) = h_IL_large( enclosure.area, alpha_w, machine.area, machine.absorption, TL );

        % 500 Hz - QBV-2
        alpha_w = 0.96;  % From specification sheet.
        TL = 59;  % From specification sheet.
            IL_estimates(2) = h_IL_large( enclosure.area, alpha_w, machine.area, machine.absorption, TL );

        % 1 kHz - QBV-2
        % alpha_w = 1.13;  % From specification sheet.
        alpha_w = 0.99;  % From specification sheet.  See comment on slide 28 of Lecture 10.
        TL = 68;  % From specification sheet.
            IL_estimates(3) = h_IL_large( enclosure.area, alpha_w, machine.area, machine.absorption, TL );

        % 2 kHz - QBV-2
        % alpha_w = 1.08;  % From specification sheet.
        alpha_w = 0.99;  % From specification sheet.  See comment on slide 28 of Lecture 10.
        TL = 67;  % From specification sheet.
            IL_estimates(4) = h_IL_large( enclosure.area, alpha_w, machine.area, machine.absorption, TL );

        % 4 kHz - QBV-2
        alpha_w = 0.91;  % From specification sheet.
        TL = 72;  % From specification sheet.
            IL_estimates(5) = h_IL_large( enclosure.area, alpha_w, machine.area, machine.absorption, TL );

end


[ octave_band_IL   IL_estimates.'   (octave_band_IL - IL_estimates.') ]


% What is the most restrictive case?



%% Find Values of TL and Aborption that will Meet the Target Insertion Loss - Ground with Cover

% Assumption(s):
%
%   1.)  The ground is covered with the absorption material.
%   2.)  The enclosure is a cube.



%% Clean-up

if ( ~isempty( findobj( 'Type', 'figure' ) ) )
```

```
    monitors = get( 0, 'MonitorPositions' );
        if ( size( monitors, 1 ) == 1 )
            autoArrangeFigures( 2, 2, 1 );
        elseif ( 1 < size( monitors, 1 ) )
            autoArrangeFigures( 2, 2, 1 );
        end
end


fprintf( 1, '\n\n\n*** Processing Complete ***\n\n\n' );



%% Reference(s)
```

6 Appendix - Matlab Code for Problem 6

```
%% Synopsis

% Lecture 11, Wednesday, February 19, 2025

% The compressor elevated above the ground.



%% Environment

close all; clear; clc;
% restoredefaultpath;

% addpath( genpath( '' ), '-begin' );
addpath( genpath( '../40 Assignments/00 Support' ), '-begin' );

% set( 0, 'DefaultFigurePosition', [ 400 400 900 400 ] ); % [ left bottom width height ]
set( 0, 'DefaultFigurePaperPositionMode', 'manual' );
set( 0, 'DefaultFigureWindowStyle', 'normal' );
set( 0, 'DefaultLineLineWidth', 0.8 );
set( 0, 'DefaultTextInterpreter', 'Latex' );

format ShortG;

pause( 1 );

PRINT_FIGURES = 0;



%% Define Anonymous Functions

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 );
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));
h_RA_term_3 = @( rho0, c , S, k, delta_mu, D, w, h ) ( rho0*c/S ) * (0.5*S*k^2)/(2*pi);



%% Define Compressor

compressor.width = 1; % m
compressor.depth = 1; % m
compressor.height = 2; % m
    compressor.area = 2*(compressor.width * compressor.depth) + 2*(compressor.width * compressor.
        height) + 2*(compressor.depth * compressor.height); % 3 m^2
    compressor.volume = compressor.width * compressor.depth * compressor.height; % m^3

compressor.power_level = 105; % dB re: 1e-12 Watts
compressor.frequency = 50; % Hz

c = 343; % m/s

rho0 = 1.2; % kg/m^3 CHECK



%% Sound Level Target

sound_level_target = 82; % dB re: 20e-6 Pascals



%% Define Workshop

R = 40; % m^2 or Sabins



%% Define Close-fitting Enclosure

```

```
helmholtz_factor = (2 * pi * compressor.frequency) / c;  % 0.92 m
%
%  For a small enclosure, k*d << 1.  Therefore d << 1.1.
d = 0.75;
d = 0.25;
% d = 1;
    % helmholtz_factor * d;  % 0.69

enclosure.width = compressor.width + d;  % m
enclosure.depth = compressor.depth + d;  % m
enclosure.height = 3;  % m;  compression height is 2 m
% enclosure.height = compressor.height + d;
    enclosure.area = 2*(enclosure.width * enclosure.depth) + 2*(enclosure.width * enclosure.height) + 2*(enclosure.depth * enclosure.height);
    enclosure.volume = enclosure.width * enclosure.depth * enclosure.height;

enclosure.E = 3.6e9;  % Pascals
enclosure.thickness = 3.81e-2;  % m
enclosure.density = 800;  % kg/m^3
enclosure.poisson_ratio = 0.25;  % Unitless

% Clamped boundary conditions.



%% Calculate Diffuse Sound Pressure Level

% Assume distance is beyond the critical distance, so the distance value is
% large and its associated term is not relevant.

sound_pressure_level = 105  +  10*log10( 4/R );  % 95 dB SPL



%% Calculate the Required Insertion Loss

target_insertion_loss = sound_pressure_level - 82  % 13 dB



%% Insertion Loss

% For the insertion loss to be high, we need:
%
%   1.)  Compliance of the air to be high;  volume of enclosure must be large.
%   2.)  Compliance of each enclosure wall to be low;  low area, high stiffness, edges clamped).
%             AREA IS THE DOMINATE FACTOR OVER VOLUME.


% The correction factor for clamped walls.  See Figure 12.4 on slide 9 of the Lecture 11 notes.
aspect_ratio = enclosure.height / enclosure.width;  % 1.7
    correction_factor = 2;  % Approximate value read from the Figure 12.4.

bending_stiffness = ( enclosure.E * enclosure.thickness^3 ) / ( 12*( 1 - enclosure.poisson_ratio^2 ) );  % 1.78e7
    h_wall_compliance = @( wall_area, correction_factor )  ( 0.001 * wall_area^3 * correction_factor ) / bending_stiffness;

Ca = enclosure.volume / ( rho0 * c^2 );


% Top
top.area = enclosure.width * enclosure.depth;
top.aspect_ratio = max( enclosure.width, enclosure.depth ) / min( enclosure.width, enclosure.depth );
top.correction_factor = 3.8;
    top.compliance = h_wall_compliance( top.area, top.correction_factor );


% Side 1
side_1.area = enclosure.depth * enclosure.height;
side_1.aspect_ratio = max( enclosure.width, enclosure.height ) / min( enclosure.width, enclosure.height );
side_1.correction_factor = 2;
    side_1.compliance = h_wall_compliance( side_1.area, side_1.correction_factor );

% Side 2
side_2.compliance = side_1.compliance;
```

```matlab


% Side 3
side_3.area = enclosure.width * enclosure.height;
side_3.aspect_ratio = max( enclosure.width, enclosure.height ) / min( enclosure.width, enclosure.height );
side_3.correction_factor = 2;
    side_3.compliance = h_wall_compliance( side_3.area, side_3.correction_factor );

% Side 4
side_4.compliance = side_3.compliance;


estimated_insertion_loss = 20*log10( 1 + Ca / ( top.compliance + 2*side_1.compliance + 2* side_3.compliance ) );  % 59.2 dB



%% Compliance of the Air Intake

air_intake_radius = 10e-2;  % m
air_intake_thickness = enclosure.thickness;  % m
air_intake_frequency = 50;  % Hz
    air_intake_angular_frequency = 2*pi*air_intake_frequency;  % radians/s

viscosity = 1.5e-5;  % m^2/s

h = 0.3;  % CHECK

f = 50;
    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 );
    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 );
    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 );
        impedance.real = term_1 + term_2 + term_3;


% Deng (1998)
epsilon = 1;
    L_o = air_intake_radius * ( 1.27 / (1 + 1.92*epsilon) - 0.086 );

L_e =  enclosure.thickness + 2*L_o;
    impedance.imaginary = 1j * rho0 * (2 * pi * f) * L_e / ( pi*0.1^2/4 );


impedance.net = impedance.real + impedance.imaginary;
    compliance_of_hole = 1 / impedance.net;
        Cl = abs( compliance_of_hole );

estimated_insertion_loss
estimated_insertion_loss_with_hole = 20*log10( (Cl + Ca)  / ( Cl + ( top.compliance + 2*side_1.compliance + 2* side_3.compliance ) ) )


13 - estimated_insertion_loss_with_hole;



critical_frequency = c^2/(2*pi)*sqrt( enclosure.density * enclosure.thickness / bending_stiffness );
%
% The critical frequency is 25 Hz.
%
% The frequency of the compressor is 50 Hz.





%% Clean-up

if ( ~isempty( findobj( 'Type', 'figure' ) ) )
    monitors = get( 0, 'MonitorPositions' );
        if ( size( monitors, 1 ) == 1 )
            autoArrangeFigures( 2, 2, 1 );
        elseif ( 1 < size( monitors, 1 ) )
```

```
                autoArrangeFigures( 2, 2, 1 );
        end
end


fprintf( 1, '\n\n\n*** Processing Complete ***\n\n\n' );



%% Reference(s)
```