Modeling Insertion Loss of Engine Enclosures

Zac Jensen

A senior thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Bachelor of Science

Dr. Scott D. Sommerfeldt, Advisor

Department of Physics and Astronomy

Brigham Young University

April 2013

ABSTRACT

Modeling Insertion Loss of Engine Enclosures

Zac Jensen

Department of Physics and Astronomy

Bachelor of Science

Acoustic enclosures are commonly used to attenuate the noise radiated from a sound source. A model to accurately predict the insertion loss of an enclosure can quickly become complicated for complex configurations and have uncertainties over different frequency ranges. In many such situations, a reliable, simple method of predicting insertion loss with reasonable accuracy would be valuable if a quick estimation is needed for a certain enclosure. An insertion loss model was tested using a Design of Experiments (DOE) approach, which incorporated a range of apertures, absorptive treatment, and obstructions, with varying surface areas and positions in order to span a large design space. This paper discusses the insertion loss measurements of the DOE test configurations and compares the measurements with an attempted fit of all 36 configurations. Nearly 80% of the configurations fell below an error of 2.5 dB, and 100% fell below 4.5 dB, which was deemed to provide a reasonable rough estimate of insertion loss.

This research was sponsored by Caterpillar Inc., under the terms of the Caterpillar Master Sponsored Research Agreement and contains joint intellectual property. Do not distribute without the permission of Caterpillar Inc.

Contents

Section 1

Introduction

Acoustic enclosures are often used to reduce noise radiated from a source. Enclosures are especially useful for attenuating engine noise. In the case of an engine enclosure, there may be certain operational noise standards or requirements which must be met. The ability to predict how well an enclosure attenuates noise is valuable in these situations. The prediction can be even more valuable should there be certain design requirements for the enclosure, such as a mandatory placement of ventilation (aperture). A system that predicts the performance of an enclosure allows engineers to optimize an enclosure design to meet all of the necessary design requirements as well as the noise standards.

Caterpillar, Inc. (CAT) has developed a program called Sonic + which models the acoustic properties of enclosures. In this work, a simple model was developed for predicting the sound power radiated from an enclosure, which will be incorporated into Sonic +. The sound power radiated from the enclosure can be used to calculate insertion loss (IL), which is a measurement of how well the enclosure attenuates noise. The insertion loss of an enclosure is defined as the difference between the sound power level of the noise source outside of the enclosure and the sound power level of the noise source radiated by the enclosure with the source inside (Eq. 2.8). If an enclosure attenuates noise very well, it will have a high insertion loss.

1.1 Modeling Enclosure Performance

There are many methods that exist to model enclosure performance, and more specifically insertion loss. These methods may be accurate, but they can also take a considerable amount of computation time. Carter shows that insertion loss can be predicted using finite element modeling, and that source to aperture distance, aperture size and absorption affect insertion loss the most for an enclosure configuration [1]. Statistical Energy Analysis (SEA) is an approach used by Ming and Pan to model insertion loss for two enclosures with varying absorption properties [2]. They found that the amount of absorption in the enclosure affected the insertion loss greatly. Their experiment was limited to rectangular boxes, with no variation of source position or enclosure configuration. A model was developed by Sgard *et al.* using SEA and image source methods [3]. Their experiment consisted of two different enclosures, a rectangular box, and an L-shaped box, with varying source positions, enclosure materials, and an aperture element opened or closed. The image source method was comparable to other common methods in predicting insertion loss. The method discussed in this paper uses a ray-tracing approach to determine the sound power that leaves the enclosure before and after the first reflection. This approach allows for multiple source positions, apertures, absorption panels, as well as obstructions inside the enclosure.

1.2 Scope

The goal of the project was to develop and analyze predictions from a simple model of insertion loss for several enclosure configurations and compare the results to measured insertion loss data.

Usage guidelines have been developed for the simple model to help CAT engineers use the model and understand the confidence levels of the insertion loss predictions. The usage guidelines are explained in detail in section 4.

The main frequency range of interest here for noise due to a tractor engine is 1 to 8 kHz. An extension of the model to incorporate lower frequencies was briefly investigated in the early stages, but was deemed to currently be of secondary concern so it was not fully developed. Rather, it was decided to focus on developing a satisfactory model without low frequencies included. Details of the low frequency investigation are given in section 5.3.

Section 2

A Simple Model

The model uses concepts associated with ray tracing to determine how much sound power will be absorbed, reflected or transmitted by any surface within the enclosure. The surfaces of the enclosure are discretized into elements using a mesh with dimension dx. (For simplicity, square elements of length and width dx are assumed in the discussion, although the model is not limited to square elements.) The size used for all of the calculations discussed here was dx=0.2 m, which is a relatively large mesh size. This was used in order to accelerate calculations of each configuration, without sacrificing the desired level of accuracy. Using a dx of 0.2 m, an enclosure calculation only takes a few seconds. A mesh element with a dx of 0.1 to 0.3 m will provide a similar range of errors, with 0.1 m being slightly more accurate and 0.3 m slightly less accurate. It was found that the results using a coarse mesh provide very good qualitative results, so an efficient method of investigation would be to begin with a coarse grid to investigate a potentially large number of possible enclosure designs, and then to use a finer mesh to further investigate the top few designs that result from that process.

Each element in the model has a corresponding absorption and transmission coefficient, which is used to determine the amount of sound power absorbed, transmitted and reflected. There are four types of elements in the enclosure model: apertures, absorption panels, obstruction sides, and the sides of the enclosure. Solid angles are used to determine the percentage of sound power that will be incident upon an element.

2.1 Solid Angle

The solid angle is a measure of how much a point (or element) “sees” an object, or how large the two-dimensional angle is that the object subtends at that point. In spherical coordinates, the solid angle is measured by integrating over the azimuthal angle and the polar angle of the object [4].

$$\Omega = \iint_S \sin\theta \, d\theta d\phi \qquad (2.1)$$

A point radiating into free space has a total solid angle of 4π, a complete sphere. A point located on a rigid plane has a solid angle of 2π, half of a sphere. Because the elements used in the enclosure appear as rectangular elements, an approximation can be used to determine the solid angle to avoid the integration. For the approximation to work, the point is considered to be at the origin of a sphere with one corner of the rectangle along the x-axis. Figure 2.1 show the resulting geometry. The dimensions of the element are a and b, and c is the distance from the point to the corner of the object.

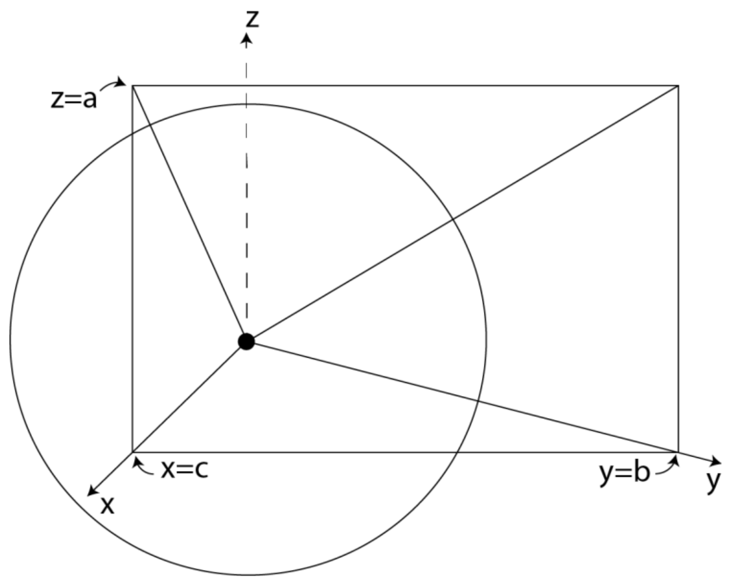


Figure 2.1 – A rectangular object with a corner opposite the origin of the sphere. The dimensions of the rectangle are a x b, and c is the distance from the corner to the origin. The solid angle subtended by the rectangle is given by Eq. 2.2.

Using spherical triangles and a few approximations, Khadjavi shows that the solid angle of the point to the element is approximated by [5]:

$$\sin \Omega = \frac{ab}{\sqrt{a^2+c^2}+\sqrt{b^2+c^2}} \qquad (2.2)$$

Figure 2.2 shows an example of calculating the solid angle of an element. The source is set at the origin, and the left corner of the element is at (0, y, 0). The element is discretized into mesh elements with dimension dx. If the dx does not fit exactly in the dimensions, a rectangle (with width less than dx) is created to incorporate the rest of the element into a smaller mesh element. Figure 2.2 shows the process for calculating the solid angle for a mesh element. The solid angle of the mesh element is approximated by using the solid angles of four rectangles that are added and subtracted to yield to solid angle for just the mesh element.

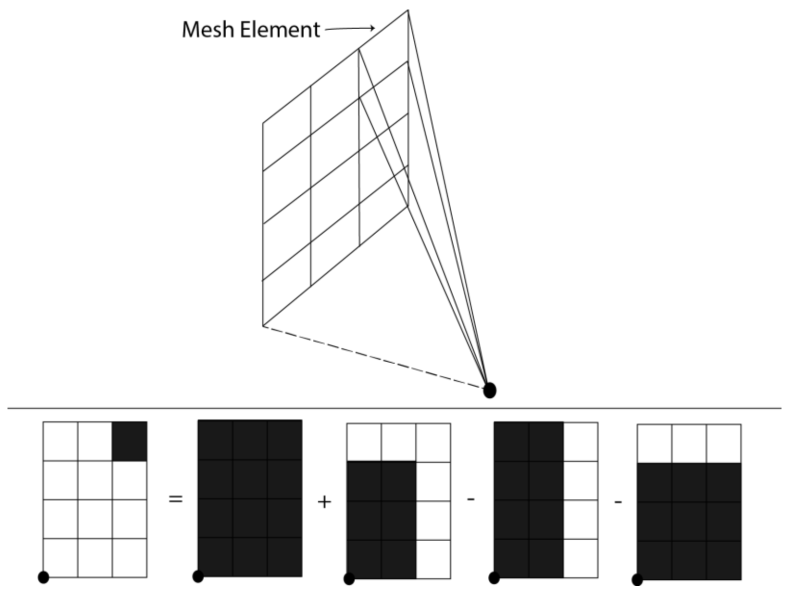


Figure 2.2 – The solid angle of a mesh element is shown. As long as the source is opposite a corner of the rectangle, the solid angle of the mesh element is the sum of the solid angles of different rectangles. The complete solid angle of the source to the entire element is the sum of each of the mesh element solid angles.

The insertion loss model uses two different solid angles, the solid angle of the source to each respective element, Ω_{Si} (Figure 2.3), and the solid angle from one element to another element, Ω_{kj} (Figure 2.4). The first solid angle is calculated using the approximation mentioned above. The source is reoriented and used as a point at the origin. Because obstructions are included in the enclosure, it is necessary to determine if there is an obstruction between the source and a mesh element. If there is an obstruction between the source and the mesh element (i.e. the source can't "see" the mesh element), the solid angle of that mesh element is not calculated. The solid angle from the source to an element is then the sum of the solid angles of all unobstructed mesh elements.

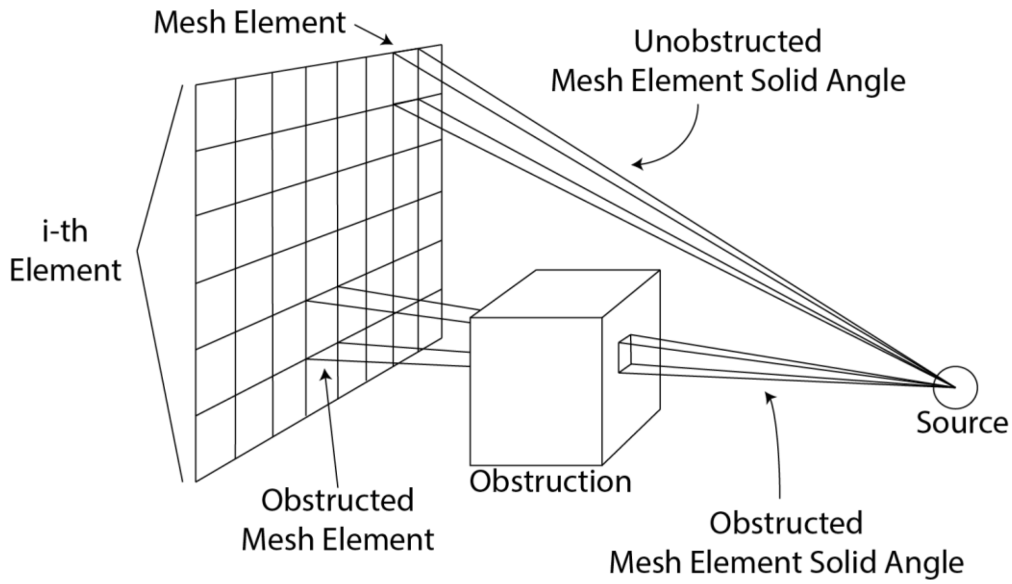


Figure 2.3 – The solid angle measured from the source to a mesh element. If there is an obstruction (or another element) between the source and a mesh element, the solid angle from the source to the obstructed mesh element is not calculated. The solid angle from the source to an element is the sum of the solid angles from the source to each unobstructed mesh element.

The second solid angle calculation, element to element, is calculated in a manner similar to the source to element solid angle, with one main difference. Both the j-th and the k-th elements are discretized into a mesh. The solid angle is then calculated from the center of the k-th mesh element to each unobstructed j-th mesh element. These are then summed and the solid angle is calculated from the center of the next mesh element of the k-th element to each of the unobstructed mesh elements of the j-th element. The total solid angle from the k-th element to the j-th element is the sum of each of these mesh element solid angles.

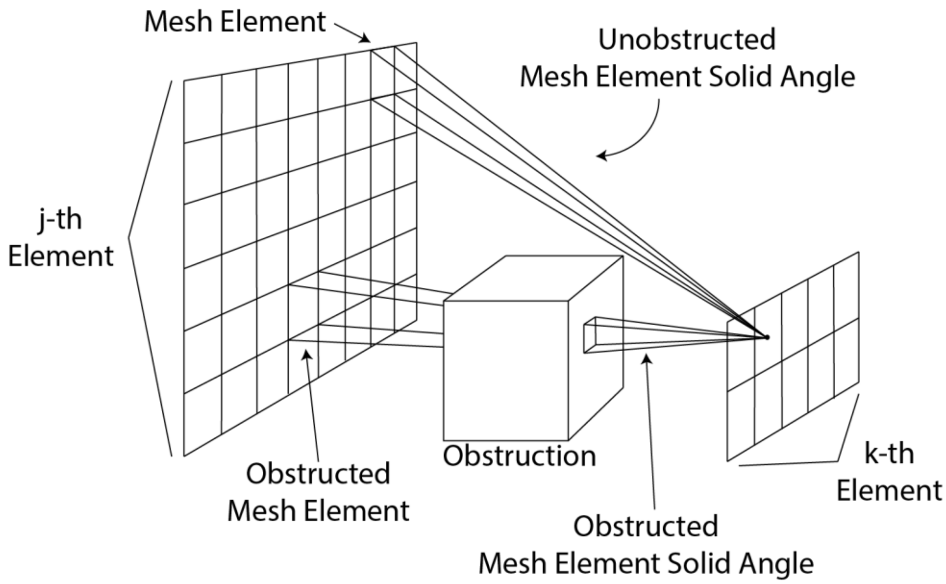


Figure 2.4 – The solid angle measured from the k-th element to the j-th element. The approach is similar to the approach for the solid angle from a source to an element, using the center of the mesh element for the k-th element as the point for the solid angle.

2.2 The Fundamental Equation

A simple model for insertion loss has been developed by CAT and BYU, and is based on a fundamental equation to estimate the insertion loss. This fundamental equation appears as Eq. 3.6 and requires that the enclosure geometry is known, as well as the absorption (α) and transmission (τ) coefficients of each surface.

In the current implementation, it is assumed that there is no reflection when energy reaches an aperture, so all of the energy at the aperture is transmitted (τ=1 and α=0). Obstruction elements are assumed to have τ=0. Otherwise, when energy reaches another element, the transmission through that element can be given by τ_j, the transmission coefficient of the j-th element. The ratio of the solid angle from the source to the j-th element (Ω_{Sj}) to the total solid angle of a point (4π) gives the fractional amount of direct-radiated power that will be incident on the element. Eq. 2.3 determines the direct power transmitted (Π_{direct}) through all elements on the first reflection as a fraction of the total sound power of the source (Π).

$$\Pi_{direct} = \Pi \sum_j \frac{\Omega_{Sj}\tau_j}{4\pi} \quad (2.3)$$

A reverberant field term is used to estimate the sound power that transmits from the enclosure after the first reflection. At the first reflection, sound power will be transmitted, absorbed, or reflected. Eq. 2.4 is the "reverberant" power term, and becomes the power available for the reverberant field term. It determines the fraction of total sound power (Π) left in the enclosure after the absorption and transmission of energy through each element on the first reflection.

$$\Pi\left[1 - \frac{\sum_i \Omega_{Si}(\alpha_i + \tau_i)}{4\pi}\right] \quad (2.4)$$

The next part of the reverberant field term determines the fraction of the reverberant power that transmits out of the enclosure (through absorption or radiation) after the first reflection. The complete reverberant field term is given by Eq. 2.5, where the three sums are over all elements (except for the k≠ j term in the third sum).

$$\Pi_{reverb} = \Pi\left[1 - \frac{\sum_i \Omega_{Si}(\alpha_i + \tau_i)}{4\pi}\right] \sum_j \sum_{k \neq j} \frac{(1-\alpha_k-\tau_k)\Omega_{kj} S_k \tau_j}{2\pi S(\bar{\alpha}+\bar{\tau})} \qquad (2.5)$$

This expression assumes that the reverberant energy is a diffuse field. The $(1-\alpha_k-\tau_k)$ term is the fraction of energy that reflects from the k-th element. $S_k$is the surface area of the k-th element, and S is the total surface area of the enclosure (including apertures). The spatially averaged absorption coefficient $\bar{\alpha} = \frac{\sum S_i \alpha_i}{S}$ and transmission coefficient $\bar{\tau} = \frac{\sum S_i \tau_i}{S}$ are in the denominator multiplied by surface area. This fractional energy is then multiplied by $\Omega_{kj}/2\pi$, the fraction of solid angle that the k-th element "sees" the j-th element. It is then multiplied by τ_j the transmission coefficient of the j-th element. This gives the amount of reverberant energy that escapes the enclosure through the j-th element. Only one reflection was considered directly in this model, with all subsequent reflections being lumped together in the reverberant term. Other ray tracing models may follow more than one reflection, but for simplicity this model considers only the first reflection. It is assumed that this will give the desired accuracy without a considerable amount of computation time.

The total energy radiated from the enclosure can be found by summing the direct and reverberant field terms ($\Pi_{rad} = \Pi_{direct} + \Pi_{reverb}$).

$$\Pi_{rad} = \Pi \left[\sum_j \frac{\Omega_{Sj}\tau_j}{4\pi} + \left(1 - \frac{\sum_i \Omega_{Si}(\alpha_i+\tau_i)}{4\pi}\right) \sum_j \sum_{k\neq j} \frac{(1-\alpha_k-\tau_k)\Omega_{kj} S_k \tau_j}{2\pi S(\overline{\alpha}+\overline{\tau})}\right] \quad (2.6)$$

At this point, if the source power (Π) is known, then the power radiated from the enclosure (Π_{rad}) can be found using Eq. 2.6. The physical interpretation of each term is outlined in Fig. 2.5.

$$\Pi_{rad} = \Pi \left[\overbrace{\sum_j \frac{\Omega_{Sj}\tau_j}{4\pi}}^{\text{Direct Field Term}} + \overbrace{\underbrace{\left(1 - \frac{\sum_i \Omega_{Si}(\alpha_i+\tau_i)}{4\pi}\right)}_{\text{Fraction of energy remaining in the enclosure after one reflection}} \sum_j \sum_{k\neq j} \frac{(1-\alpha_k-\tau_k)\Omega_{kj} S_k \tau_j}{2\pi S(\overline{\alpha}+\overline{\tau})}}^{\text{Reverberant Field Term}}\right]$$

Solid Angle from the source to the j-th element

Transmission coefficient of the j-th element

Solid Angle from the source to the i-th element.

Absorption and transmission coefficients of the i-th element

Fraction of energy reflecting off of the k-th element

Solid Angle from the k-th element to the j-th element.

Surface area of k-th element

Total Source Power

Total surface area of enclosure, including apertures

Spatially averaged absorption and transmission coefficients

Transmission coefficient of the j-th element

Figure 2.5 – The fundamental equation (Eq. 2.6) is shown, and the physical interpretation of each term is outlined.

The power radiated from any single element (besides obstructions) can be calculated by replacing the sum over the index j with the single element or elements desired. In other words, if an enclosure has one aperture, and it is necessary to calculate the sound power radiated only from that aperture (and not the power radiated by transmission through the sides of the enclosure), it can be done by replacing the sum over j elements with one element, which we will label Ap. Eq. 2.7 is the power radiated from the aperture element, (where N is the total number of elements). If an enclosure has more than one aperture element, and the power radiated from all apertures is desired, then Π_{Ap} would be the sum of the power radiated from each aperture. A similar process can be used for any other element of the enclosure.

$$\Pi_{Ap} = \Pi\left[\left(\frac{\Omega_{SAp}\tau_{Ap}}{4\pi}\right) + \left(1 - \frac{\sum_i^N \Omega_{Si}(\alpha_i+\tau_i)}{4\pi}\right)\sum_{k\neq Ap}^N \frac{(1-\alpha_k-\tau_k)\Omega_{kAp}S_k\tau_{Ap}}{2\pi S(\bar{\alpha}+\bar{\tau})}\right] \quad (2.7)$$

Insertion loss is defined by the difference of the sound power level of the source without the enclosure and the sound power level of the source inside the enclosure. Because of the rules of logarithms, this can be manipulated to be the sound power level of the ratio of the total sound power divided by the sound power radiated out of the enclosure.

$$IL = 10\log_{10}\Pi - 10\log_{10}\Pi_{rad} = 10\log_{10}\frac{\Pi}{\Pi_{rad}} \quad (2.8)$$

Because there is a source power term (Π) in the sound power radiated from the enclosure (Π_{rad}), the sound power ratio ($\frac{\Pi}{\Pi_{rad}}$) can be simplified so that the sound power term (Π) cancels out. This is based on the assumption that the power radiated by the source inside the enclosure is the same as the power radiated by the source outside of the enclosure. The full formula for the ratio is:

$$\frac{\Pi}{\Pi_{rad}} = \frac{1}{\sum_j \frac{\Omega_{Sj}\tau_j}{4\pi} + \left[1 - \frac{\sum_i \Omega_{Si}(\alpha_i+\tau_i)}{4\pi}\right]\sum_j \sum_{k\neq j} \frac{(1-\alpha_k-\tau_k)\Omega_{kj}S_k\tau_j}{2\pi S(\bar{\alpha}+\bar{\tau})}} \quad (2.9)$$

Insertion loss can then be calculated as:

$$IL = 10\log_{10}\frac{1}{\sum_j \frac{\Omega_{Sj}\tau_j}{4\pi} + \left[1 - \frac{\sum_i \Omega_{Si}(\alpha_i+\tau_i)}{4\pi}\right]\sum_j \sum_{k\neq j} \frac{(1-\alpha_k-\tau_k)\Omega_{kj}S_k\tau_j}{2\pi S(\bar{\alpha}+\bar{\tau})}} \quad (2.10)$$

2.3 Fit to Experimental Data

Eq. 2.10 is the theoretical equation used to predict the insertion loss of an enclosure with a single source and the elements discussed. A design of experiments (DOE) was performed with a wide range of enclosure arrangements. Two different methods of fitting the model to the experimental data were tested.

The first method used assumes that the equation for insertion loss is a quadratic equation, with a DC term, a linear term and a squared term.

$$IL = a1 + a2 * \log_{10} \frac{\Pi}{\Pi_{rad}} + a3 * \left(\log_{10} \frac{\Pi}{\Pi_{rad}}\right)^2 \qquad (2.11)$$

The constants in Eq. 2.11 were found by fitting the model to the experimental data using a least squares method (LS). The sum of the squares of the residuals (the difference between the model and the measurement) is minimized in a least squares fit [6]. This optimizes the curve (in this case the curve is Eq. 2.11) to fit the experimental data. For a perfect model, the constants should be a1=0, a2=10, and a3=0. It was desired that the least squares fit should give similar coefficients.

The second method was a fit in the sound power (Watts) domain instead of the insertion loss domain (dB). Because Sonic + does all of its calculations in the power domain, it was decided that this may be a more useful way to fit the data. The fit was developed based on trends seen in the data, specifically regarding the solid angle of the source to all apertures (Ω_{SAp}) and the solid angle of the obstructions to apertures (Ω_{OAp}). This method was determined to be superior to the least squares method of fitting and a more detailed analysis will be given in Sec. 4 of this paper.

Section 3

Experimental Methods

3.1 Design of Experiments

The design of experiments (DOE) was developed in order to cover a large range of possible enclosure configurations. The DOE included a single source and multiple elements: apertures, absorptions, obstruction sides, and the enclosure walls. The size of the enclosure or the elements could not change, but the placement of the source and the elements was varied into 24 different configurations. After the first 24 configurations were measured, it was necessary to include 12 more configurations which were designed to test the trends seen in the first 24 measurements. Table 3.1 shows the range of parameters covered in the DOE. The solid angle percentage in the table is defined as $\sum\Omega_{Si}/4\pi * 100$, the sum of the fractional solid angle of the same elements. In other words, if there were three aperture elements, the solid angle percentage would be the sum of the solid angles to all three apertures, divided by 4π for the fraction, then multiplied by 100 to get a percentage. Since the obstructions take up a certain volume in the enclosure, the volume percentage is a sum of the total volume of all obstructions, divided by the volume of the enclosure, then multiplied by 100. For absorption panels, surface area was used, such that the percentage of absorption surface area was 100 multiplied by the total surface area of all absorption panels, divided by the surface area of the entire enclosure. Aperture surface area percentage was calculated likewise.

	Obstruction	**Absorption**	**Aperture**
Solid Angle (Source to Element) %	0% to 50%	0% to 40%	0% to 30%
Volume / Surface Area %	0% to 55%	8% to 22%	4% to 15%

Table 3.1 – The range spanned for each element in two different categories, solid angle source to the element, and volume or surface area percentage of the element to the entire enclosure.

3.1.1 Element Size and Placement

Enclosure

The enclosure was made out of ¾" MDF. The absorption and transmission coefficients were obtained from published values for MDF. The dimensions of the enclosure were 1.95 m x 0.745 m x 1.2 m, in order to roughly approximate a typical CAT engine enclosure (see Fig. 3.1).

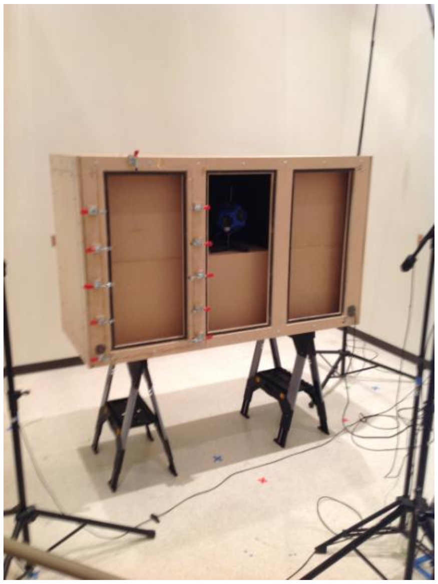


Figure 3.1 – The experimental enclosure with all five obstructions inside, and three apertures open. The enclosure was placed on sawhorses 0.91 m from the floor in order to approximate operating position.

Apertures

The aperture panels were MDF panels that could be removed from the side, front, and back of the enclosure. There were two different sizes of apertures. The apertures on the side of the enclosure were 0.457 m x 1.02 m while the aperture on the front or back of the enclosure was 0.584 m x 0.991 m (see Fig. 3.2). If an aperture was desired at a certain place on the enclosure, the panel at that spot would be removed. If no aperture was desired at that place, the aperture panel would be bolted back in to the enclosure. A gasket on the aperture panel served to seal the panel and minimize leakage through the edges of the panel.

Figure 3.2 – Aperture panels used to close a side aperture (left) and front aperture (right).

Absorption Panels

The absorption panels had dimensions 0.61 m x 1.17 m and were placed on the enclosure walls (see Fig. 3.3). The absorption coefficients for the absorption panels were experimentally obtained at BYU.

Figure 3.3 – Absorption panel.

Obstructions

The obstructions were also made of ¾" MDF. They were cubes with side dimensions of 0.57 m (see Fig. 3.4). The obstructions were placed inside the enclosure in varying arrangements. If an obstruction was stacked on top of another, they were modeled as one obstruction.

Figure 3.4 – Obstruction element.

Source

The source used was a dodecahedron source built at BYU. It was necessary for the source to be small and behave somewhat like a point source (see Fig. 3.5). The dodecahedron was used because it was small enough to fit into the enclosure around the obstructions, and because CAT uses similar sources for testing. The source was placed on a tripod-like stand inside the enclosure, either on the floor of the enclosure or on top of an obstruction.

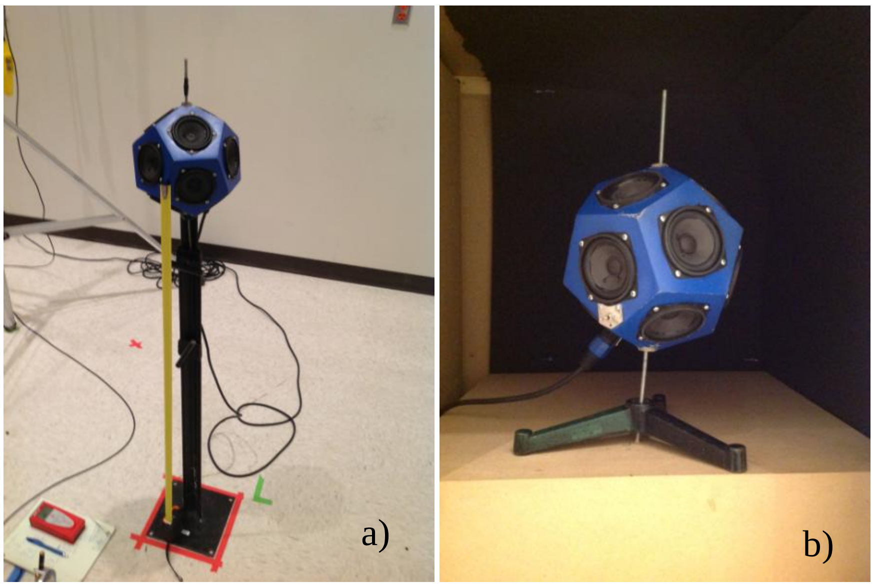


Figure 3.5 – Dodecahedron a) outside of enclosure and b) inside the enclosure. All source locations are relative to the center of the dodecahedron.

3.2 Sound Power Measurements

Sound power measurements were taken in a certified reverberation chamber. The measurements were taken as near as possible to the requirements of ISO 3741:2010 [7]. There were a few requirements that could not be met because of space restrictions. In the standard, sound power measurements are calculated by using at least 6 microphones to obtain a spatially averaged sound pressure level (SPL). The sound power level is then obtained using meteorological conditions of the measurement, as well as the reverberation time (T_{60}) and dimensions of the room. The equation in the standard uses the room and test conditions and adds to or subtracts from the SPL in decibels. From the rule of logarithms, this is effectively the same as multiplying the spatially averaged pressure by a constant (χ). In other words, according to ISO 3741, the sound power level is given by the following equation, where $\bar{p}$ is the spatially averaged pressure:

$$\Pi = \bar{p}^2\chi \qquad (3.1)$$

Using Eq. 3.1, Eq. 2.8 can be re-written for a different way of calculating insertion loss, given in Eq. 3.2, where $\overline{p_{rad}}$ is the averaged pressure radiated from the enclosure.

$$IL = 10 * \log_{10}\frac{\Pi}{\Pi_{rad}} = 10 * \log_{10}\frac{\bar{p}^2\chi}{\overline{p_{rad}}^2\chi} \ (3.2)$$

For insertion loss measurements, it was necessary to measure the sound power of the source outside of the enclosure, and then place the source inside the enclosure and measure the sound power radiated from the enclosure. If the meteorological and room conditions are the same for both the source and enclosure measurements, then the power coefficient χ is the same for both cases. This term cancels out in Eq. 3.2 and the insertion loss can be calculated by the following equation:

$$IL = 10 * \log_{10}\frac{\bar{p}^2}{\overline{p_{rad}}^2} = 10 * \log_{10}\bar{p}^2 - 10 * \log_{10}\overline{p_{rad}}^2 \ (3.3)$$

The insertion loss measurements were calculated using Eq. 3.3, as long as the power coefficient (χ) remained the same during both the outside of enclosure and inside enclosure measurements. Figure 3.6 shows the SPL of the source outside the enclosure for each of the 36 configurations. The measurements were very uniform, which was expected because the meteorological and room conditions remained the same. Figure 3.6b shows the 36 SPLs radiated from each of the enclosure configurations. These measurements were subtracted from the measurements in Figure 3.6a in order to obtain the insertion loss for each configuration.

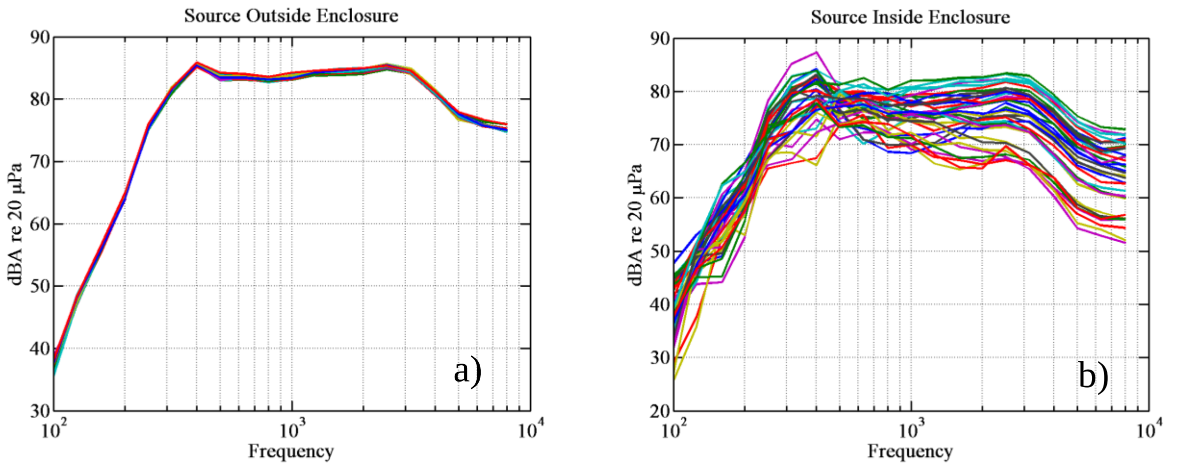


Figure 3.6 – a) The spatially averaged sound pressure level of the source outside of the enclosure for each enclosure configuration. b) The spatially averaged sound pressure level radiated from the enclosure for each enclosure configuration.

3.3 Experimental Setup

3.3.1 Reverberation Chamber

The reverberation chamber has dimensions of approximately 5 m x 6 m x 7 m, with the height as the longest dimension. The nature of the experiment and the layout of the chamber made space an issue in the setup. The setup was not able to comply with all of the ISO requirements, but came as close as possible. The requirements for the source state that it should be placed in normal working position or on the ground if this is not possible. The source should also be 1.5 m from hard surfaces (besides the ground). Because the insertion loss measurements measured both the source outside the enclosure and the source inside the enclosure, the experiment effectively had two source positions. When the source is outside the enclosure, it needs to be 1.5 meters from all hard surfaces, including the enclosure. When the source is inside the enclosure, the entire enclosure is considered the source and must be 1.5 meters from hard surfaces. There was not sufficient space in the chamber for the enclosure to fit and for the dodecahedron to be

1.5 m away from hard surfaces. One end of the enclosure violated the 1.5 m boundary requirement by 7.4 cm. This end did not have an open aperture, so it was determined that this would be the best way to break the standard requirements. Figure 3.7 shows the setup inside the reverberation chamber.

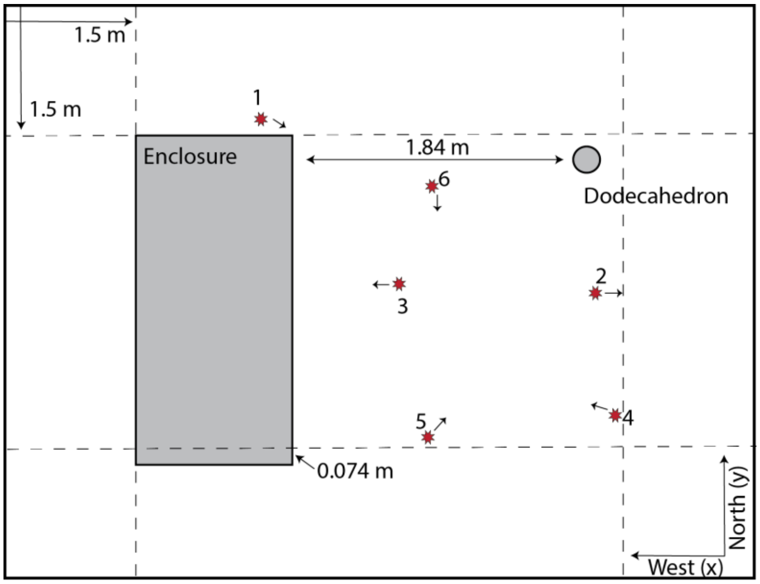


Figure 3.7 – The layout of the reverberation chamber. The edge of the enclosure is past the 1.5 m requirement by about 7.4 cm. The red stars represent the six microphones; the arrows represent the relative direction the microphone is pointed. The bottom right corner represents the origin (0,0,0) with the x-direction to the west and the y direction to the north.

3.3.2 Microphone Placements

ISO 3741:2010 requires at least six microphones to be randomly placed around the source. There are several criteria to determine how far from the source each microphone should be to ensure each microphone is in the reverberant field. Each microphone was placed a distance of at least 1.7 m away from the source. The microphones also need to be at least half a wavelength away from each other. At 100 Hz, this distance is 1.75 m and the microphones are approximately that far apart. The measurements meet the standard from 1000 Hz to 8000 Hz. The reverberation time (T_{60}) of the chamber was different with the enclosure inside, and so the low frequency validity of the measurements extends to approximately 300 Hz. Only the 1000 to 8000 Hz measurements will be considered in this paper. Figure 3.7 shows the microphone positions and Table 3.2 shows the microphone distances and angles relative to the dodecahedron position, without the dodecahedron in the stand.

Dodecahedron (Acoustic Center)	X: 1.68 m	Y: 3.34 m	Z: 1.12 m
Microphone Reference Position	X: 1.68 m	Y: 3.34 m	Z: 1.02 m
--------------	**Azimuthal Angle**	**Altitude Angle**	**Distance**
Microphone 1	7.5^o	56.5^o	4.18 m
Microphone 2	254^o	70^o	1.98 m
Microphone 3	340^o	59.5^o	3.35 m
Microphone 4	255.5^o	26^o	2.25 m
Microphone 5	301^o	50^o	4.19 m
Microphone 6	355.5^o	78^o	4.72 m

Table 3.2 – Source and microphone positions for reverberation chamber experimental setup. Microphone positions are azimuthal and altitude angles relative to the dodecahedron stand position, which is X: 1.68 m, Y: 3.34 m, and Z: 1.02 m, about 4 inches lower than the acoustic center of the dodecahedron. The azimuthal angle is 0^o facing the West wall and positive degrees are clockwise from that position (90^o is facing the North wall). The altitude angle is 0^o when pointed directly at the West wall and 90^o when pointed at the ceiling.

Section 4

Results and Analysis

4.1 Overall Error Definition

The most important factor in analyzing the data was determining how well the model works compared to the actual measurements. An overall error term was defined in order to more easily compare the 36 configurations. A residual was found by subtracting the insertion loss predicted by the model from the measured insertion loss at each third octave band (Eq. 4.1). An overall error was then determined by the average of the absolute value of the residuals across the frequency range of interest (Eq. 4.2, F is the number of third octave bands in the frequency range). Although the overall error is always a positive value, if the majority (50% or more) of residual errors over the frequency range were negative, then the overall error value was changed to a negative number. This made it possible to tell if the model underestimated or overestimated insertion loss for the most part. Because the large part of the research was to determine how well the model works compared to measured data, it became important to know trends in the model, such as over- and underestimation. Figure 4.1 shows the residual and overall error graphically for configuration 2 (see Sec. 5.3 for a description of the configuration).

$$Residual\ Error = IL_{measured} - IL_{model} \quad (4.1)$$

$$Overall\ Error = \frac{1}{F}\sum_{i=1}^{F} Abs(Residual\ Error)_i \quad (4.2)$$

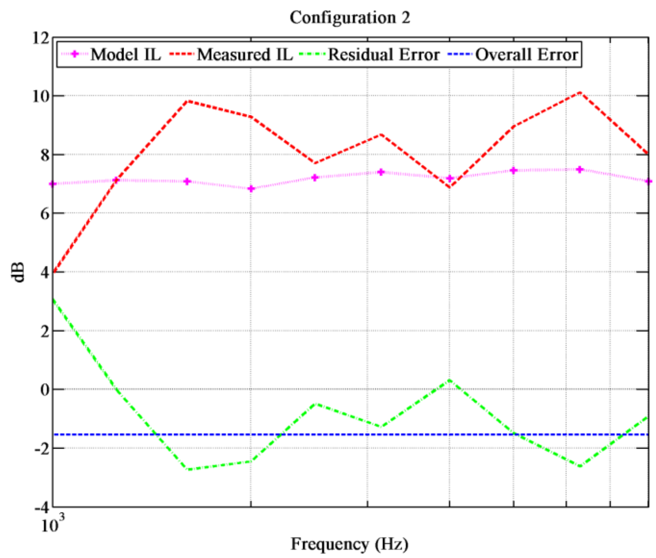


Figure 4.1 – The error measured for the 2nd configuration. The majority of residual error (green) points are negative, giving a negative overall error (blue). This indicates that the model underestimates insertion loss compared to the measured values.

4.2 Characteristics of the Model

The theoretical equation for insertion loss may not account for all acoustical effects present in the enclosure. Because this was the case, it was necessary to fit the predicted insertion loss to the measured insertion loss. Using Eq. 2.8, the equation for insertion loss was assumed to be in the form of a 2nd degree polynomial.

$$IL = a1 + a2 * \log_{10}\frac{\Pi}{\Pi_{rad}} + a3 * \left(\log_{10}\frac{\Pi}{\Pi_{rad}}\right)^2 \quad (4.3)$$

The constants in Eq. 4.3 are found by fitting the model to the experimental data using a least squares method (LS). If the model worked perfectly, the constants would be expected to be a1=0, a2=10, and a3=0. Figure 4.2 shows the initial LS fit to the experimental data for the initial 24 enclosure configurations. The coefficients for this fit were a1= 4.91, a2=-5.28, and a3=15.   Figure 4.3 is the distribution of overall errors for these 24 configurations with the least squares fit.

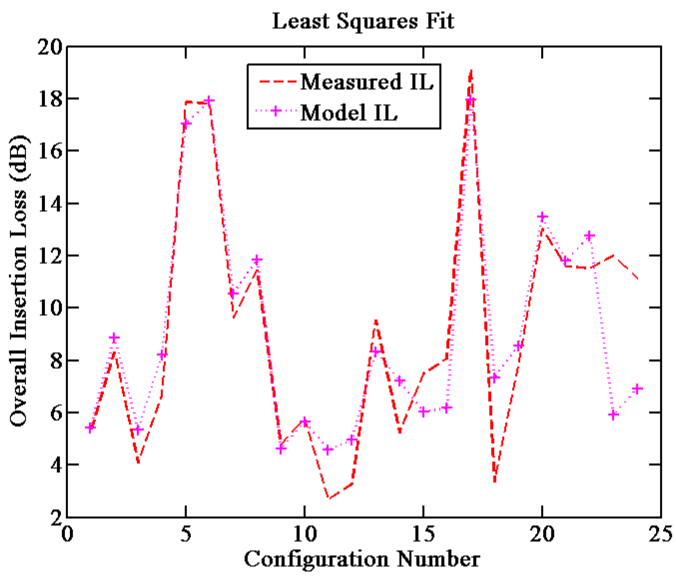


Figure 4.2 – Least Squares Fit of the initial 24 configurations

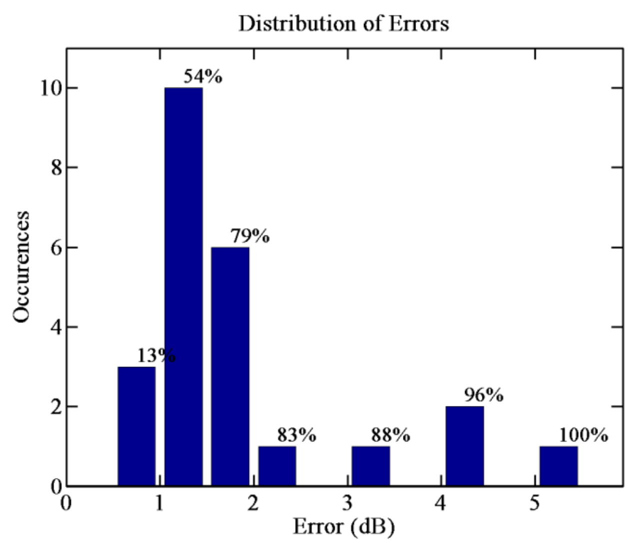


Figure 4.3 – Distribution of Errors for the initial 24 configurations

Several parameters of the enclosure were analyzed to see if there were any characteristics of an enclosure that were common in the configurations with larger errors. One parameter that seemed to contribute to a large error was a low solid angle from the source to apertures (Ω_{SAp}).

Figure 4.4 is the overall error of each configuration with respect to the configuration's Ω_{SAp}. Notice that the errors are largest where the Ω_{SAp} is low. More specifically, configurations 23 and 24 (see Sec. 5.4) had very low Ω_{SAp} and underestimated insertion loss considerably. Both of these configurations had the solid angle of the source to apertures blocked by obstructions, which contributed to the low Ω_{SAp}. Other configurations, such as 9 and 10, were similar to configurations 23 and 24 except those had a higher Ω_{SAp}. A hypothesis was created that this source configuration created an effect where the sound energy is localized to this "cavity" where the source is located and was not distributed uniformly throughout the enclosure the way the model predicts. The energy was more likely to reflect off of the surfaces in the cavity and eventually be absorbed than to reach an aperture by or shortly after the first reflection. This being the case, the model would overestimate the sound power radiated from the source, thereby underestimating insertion loss.

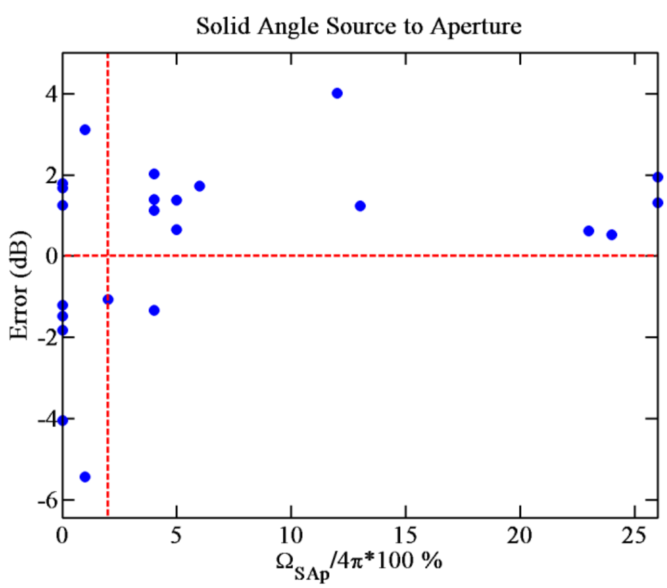


Figure 4.4 – Error with respect to Ω_{SAp} for the initial 24 configurations.

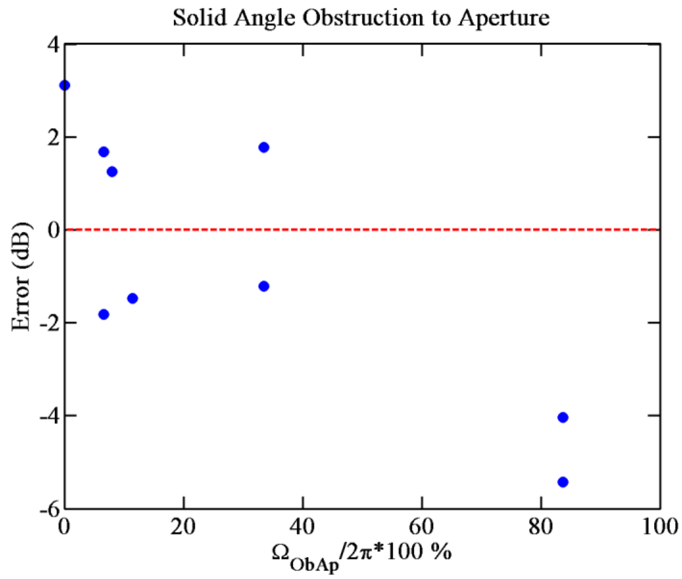


Figure 4.5 – Error with respect to Ω_{ObAp} for the configurations with a $\Omega_{Sap}\% \leq 2\%$.

An attempt was made to characterize this localized energy effect. Configurations with a $\Omega_{SAp}\%$ equal to or less than 2% were considered to have a low Ω_{Sap} (see the vertical dashed line in Fig. 4.4). The solid angle of the obstruction side to apertures (Ω_{ObAp}) for these configurations was also determined, and used as a measurement of the localized energy effect. If the Ω_{ObAp} was high (meaning the obstruction was close to and largely blocked the aperture), then the localized energy effect was deemed to be present in the configuration. This can be seen from Figure 4.5.

In order to test this hypothesis, 12 configurations were developed in addition to the 24 initial configurations. Six of the twelve configurations have a low Ω_{Sap} and varying Ω_{ObAp}, while the other six are similar configurations with a higher Ω_{Sap} (see Sec. 5.4, configurations 25 and 30). The entire set was measured and the LS fit was applied to all 36 measurements (see Fig. 4.6). In order to accommodate all 36 configurations, the coefficients changed to a1= -0.361, a2=14.3, and a3=1.71. These coefficients are much closer to the desired 0, 10, 0 coefficients of a perfect model.

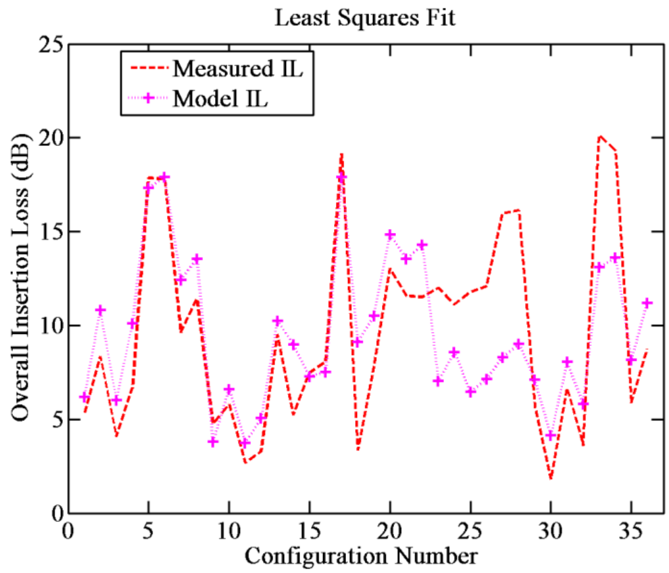


Figure 4.6 – LS fit for all 36 configurations. The coefficients for this fit were a1=-0.361, a2=14.3, and a3=1.71. These coefficients are much closure to the insertion loss definition, which would have coefficients of a1=0, a2=10, and a3=0.

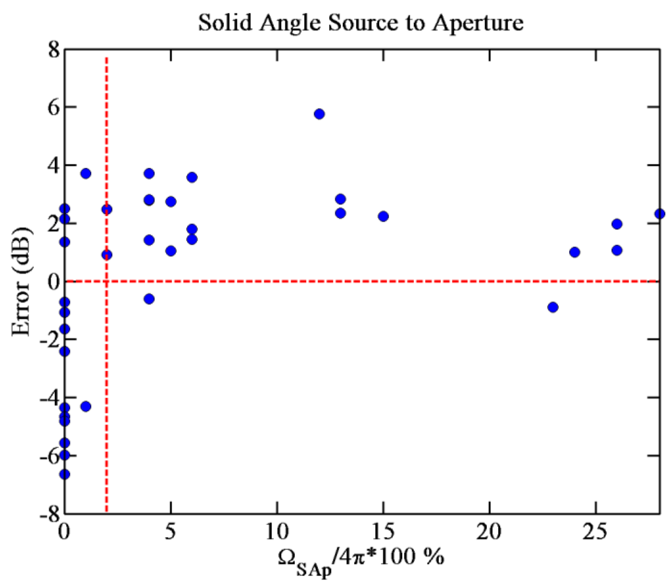


Figure 4.7 – Error with respect to Ω_{SAp} for all 36 configurations. Notice the significant underestimation at Ω_{Sap}% less than 2%.

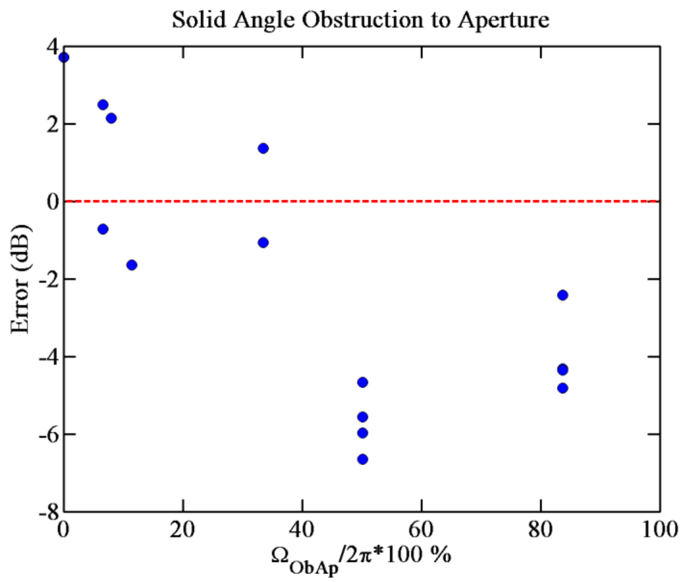


Figure 4.8 – Error with respect to Ω_{ObAp} for all configurations with a Ω_{SAp}% less than 2%. The majority of these configurations underestimate insertion loss.

Figure 4.7 shows the error of each configuration with respect to the Ω_{Sap}% for all 36 runs, and Figure 4.8 is similar to Figure 4.5 for all 36 configurations. Figure 4.9 is the distribution of errors for the LS fit of all 36 configurations. The impact of configurations 25 to 32, which were meant to be extreme cases, is seen in this figure. The maximum error is much larger for all 36 cases than it was for just the initial 24 cases.

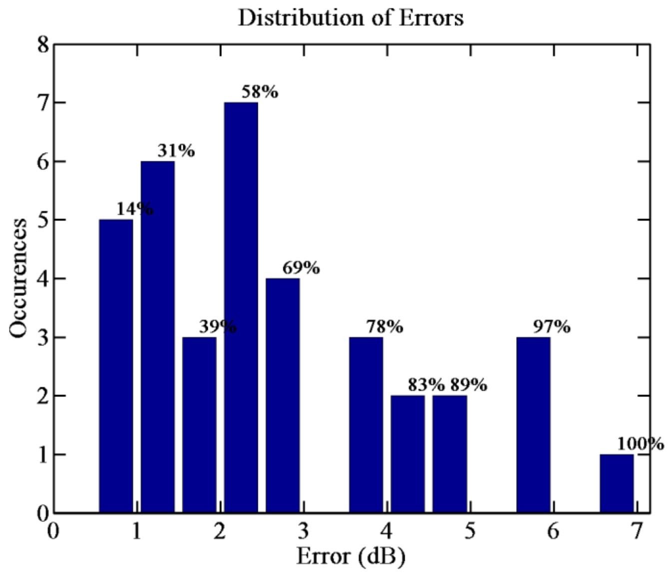


Figure 4.9 – Distribution of errors for all 36 configurations using a LS fit.

4.3 Solid Angle Fit

The model calculates the power radiated from the enclosure, and from that power the insertion loss is calculated. The LS fit is done in the insertion loss domain (i.e. dB scale). Because the model actually gives sound power information, a fit in the sound power domain was investigated and proved to be more useful. Since the LS fit of all 36 runs gave coefficients that were near to a perfect theoretical model, a fit was developed by going back to the original equation for insertion loss (Eq. 2.8), and including a fit to the $\frac{\Pi}{\Pi_{rad}}$ term. Figures 4.10 and 4.11 show the data when the coefficients were changed to a1=0, a2=10, and a3=0.

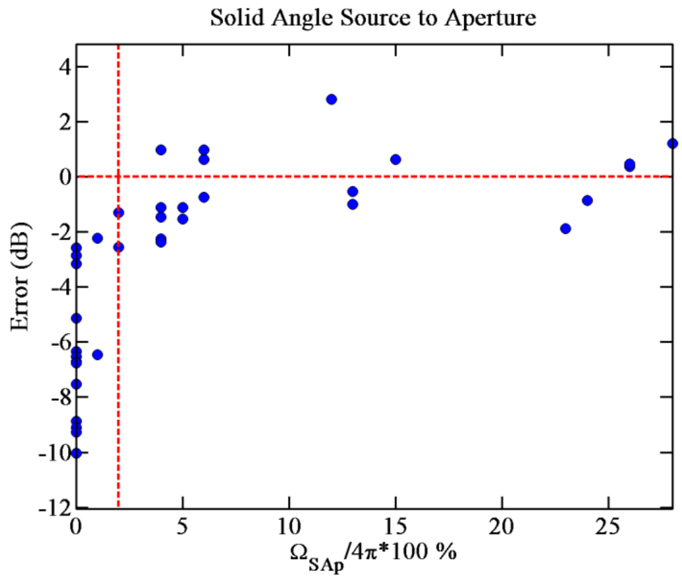


Figure 4.10 – Error with respect to Ω_{SAp}% for all 36 configurations with the traditional insertion loss coefficients: a1=0, a2=10, a3=0.

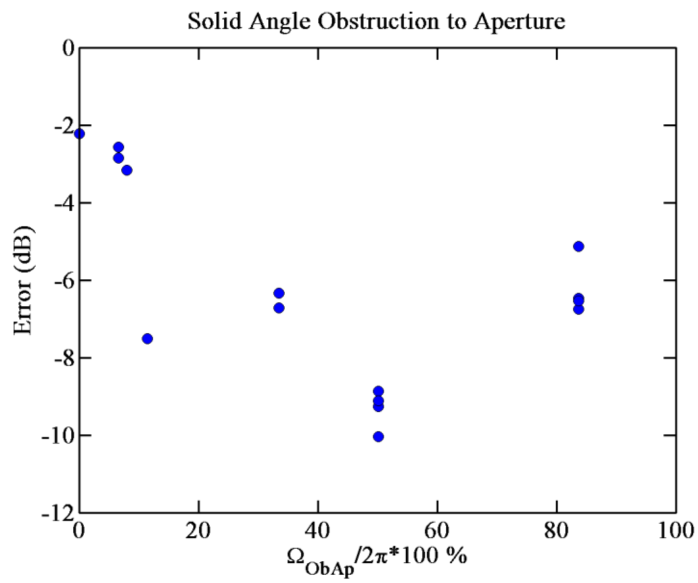


Figure 4.11 – Error with respect to Ω_{ObAp} for all configurations with a Ω_{SAp}% less than 2% with the traditional insertion loss coefficients: a1=0, a2=10, a3=0.

The fit was modified based on the solid angle of the source to apertures (Ω_{SAp}) and the solid angle of the obstructions to apertures (Ω_{ObAp}). It is basically an attempt to correct for the underestimation of the model for subspaces of the data and bring the overall error of all the configurations within the desired range. The errors were found to be largest when the Ω_{SAp} term was low. The idea behind the solid angle fit is to multiply the sound power radiated by the aperture by a constant in order to decrease the sound power radiated from the enclosure. This would increase the predicted insertion loss to better match the measurements. The sound power radiated by the apertures is given in Eq. 2.7. Two constants were used to optimize the fit. The first constant (C1) is used when a configuration has a Ω_{SAp}% less than 2% and a Ω_{ObAp}% less than 10%.

$$\Pi_{Ap} = \Pi\left[\left(\frac{\Omega_{SAp}\tau_{Ap}}{4\pi}\right) + \left(1 - \frac{\sum_i^N \Omega_{Si}(\alpha_i+\tau_i)}{4\pi}\right)\sum_{k\neq Ap}^N \frac{(1-\alpha_k-\tau_k)\Omega_{kj}S_k\tau_{Ap}}{2\pi S(\bar{\alpha}+\bar{\tau})}\right] * C1 \quad (4.4)$$

The second constant (C2) is applied to configurations with Ω_{SAp}% less than 2% and Ω_{ObAp}% greater than 10%, (i.e. a configuration exhibiting the localized energy effect).

$$\Pi_{Ap} = \Pi\left[\left(\frac{\Omega_{SAp}\tau_{Ap}}{4\pi}\right) + \left(1 - \frac{\sum_i^N \Omega_{Si}(\alpha_i+\tau_i)}{4\pi}\right)\sum_{k\neq Ap}^N \frac{(1-\alpha_k-\tau_k)\Omega_{kAp}S_k\tau_{Ap}}{2\pi S(\bar{\alpha}+\bar{\tau})}\right] * C2 \quad (4.5)$$

After the correct constant has been applied to the aperture term, the sound power radiated from the enclosure is given by the following:

$$\Pi_{rad} = \Pi_{\mathrm{Ap}} + \sum_{j\neq Ap}^N \Pi_{rad} \quad (4.6)$$

Insertion loss is then calculated as normal using the sound power term in Eq. 4.6. The best results were obtained when C1 = 0.2 and C2 = 0.1. Figure 4.12 is the same as Fig. 4.10 but with the solid angle fit applied to the data. The measurements with low solid angle are now basically centered on zero, with a much smaller margin of error. Figure 4.13 shows the distribution of errors, which is much more acceptable as a rough estimate than the previous LS fit (Fig. 4.9).

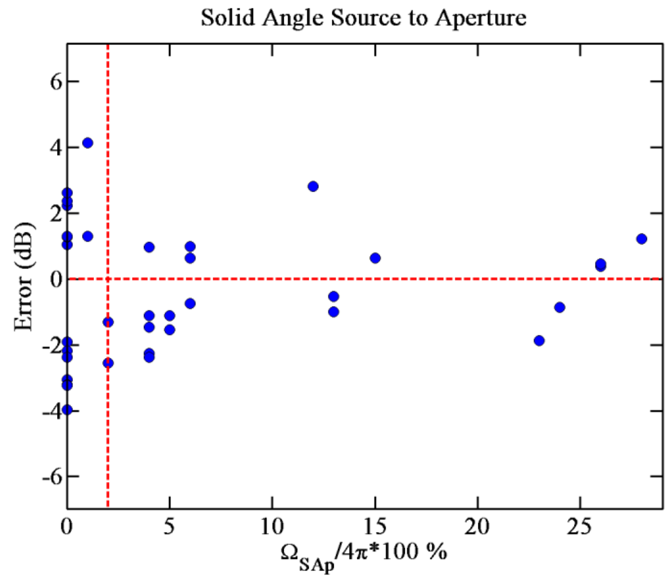


Figure 4.12 – Error with respect to Ω_{SAp}% for all 36 configurations with the solid angle fit applied (some of the points overlap).

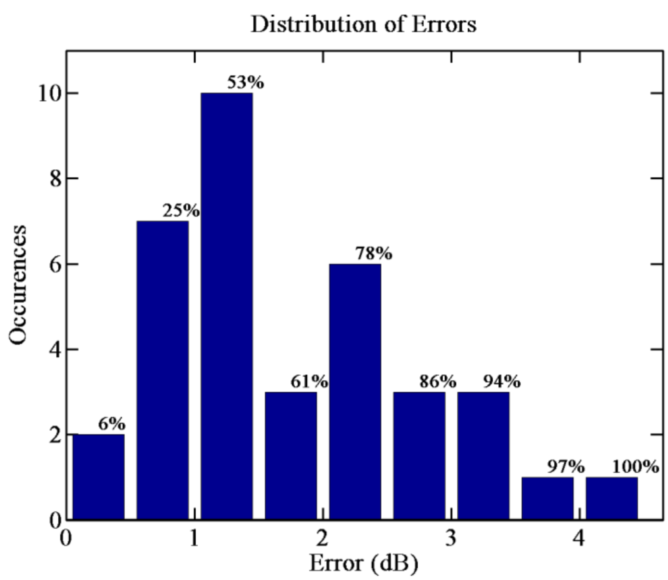


Figure 4.13 – Distribution of errors with the solid angle fit applied to all 36 configurations.

Figure 4.14 is a visual comparison of the LS fit to the solid angle fit. The solid angle fit does a much better job of fitting configurations 23 through 28 and 33 and 34, the extreme Ω_{SAp} cases.

This model has been tested for an enclosure with a single source and the basic enclosure elements discussed. With nearly 80% of the overall insertion loss errors less than 2.5 dB, and 100% less than 4.5 dB, the model is acceptable as a quick and efficient way to estimate insertion loss for an enclosure. Should an enclosure design be tested with parameters outside of the ranges in Table 3.1, the outcome of the insertion loss prediction is less certain.

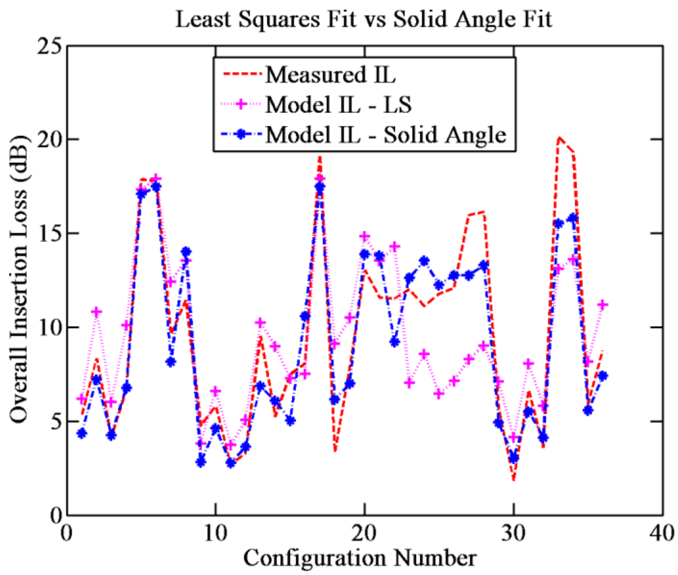


Figure 4.14 – Visual comparison of LS fit to solid angle fit. The solid angle fit does a much better job of fitting the extreme cases of configurations 23 through 28.

Section 5

Appendix

5.1 References

[1] - Carter , Amy Elizabeth, "Design of Partial Enclosures for Acoustical Applications." MS Thesis, University of Kentucky, Paper 356. http://uknowledge.uky.edu/gradschool_theses/356 (2006).

[2] - Ming, R. and J. Pan, "Insertion loss of an acoustic enclosure." J. Acoust. Soc. Am., **116**(6), 3453-3459 (2004).

[3] - Sgard, F., H. Nelisse, et al., "Prediction of the acoustical performance of enclosures using a hybrid statistical energy analysis: Image source model." J. Acoust. Soc. Am., **127**(2), 784-795 (2010).

[4] - Taylor, J. R., *Classical Mechanics*, (University Science Books, www.uscibooks.com, 2005).

[5] - Khadjavi, A., "Calculation of Solid Angle Subtended by Rectangular Apertures." J. Opt. Soc. Am., **58,** 1417 (1968).

[6] - Weisstein, Eric W. "Least Squares Fitting." From MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/LeastSquaresFitting.html

[7] - ISO 3741:2010, *Acoustics - Determination of sound power levels and sound energy levels of noise sources using sound pressure - Precision methods for reverberation test rooms.*

5.2 Definitions

Π_{rad} - The source power radiated from the enclosure

Π - Total sound power of the source when outside the enclosure

Π_{direct} – The direct power that escapes the enclosure without any reflections

Π_{reverb} – The total reverberant power that escapes the enclosure

Ω_{Si} – The solid angle of the source to the i-th element. (See Figure B.1)

$\Omega_{Si}/4\pi$ – The fraction of the total solid angle that the source "sees" the i-th element

Ω_{SAp} – Total solid angle of the source to all aperture elements

Ω_{kj} – The solid angle of the k-th element to the j-th element. (See Figure B.2)

$\Omega_{kj}/2\pi$ – The fraction of the total available solid angle that the k-th element "sees" the j-th element

Ω_{ObAp} – Total solid angle of the obstructions to all aperture elements

α_i – Absorption coefficient of the i-th element (α=0 for aperture elements)

τ_i - Transmission coefficient of the i-th element (τ=1 for aperture elements)

S – Total surface area of the enclosure (including apertures)

S_i – Surface area of the i-th element

$\bar{\alpha} = \frac{\sum S_i \alpha_i}{S}$ – Spatially averaged absorption coefficient

$\bar{\tau} = \frac{\sum S_i \tau_i}{S}$ – Spatially averaged transmission coefficient

N – Total number of elements in the enclosure configuration

T_{60} – Reverberation time

5.3 Diffraction and Directivity

5.3.1 Diffraction

Diffraction is the bending of sound around objects that are smaller than its wavelength which, in the case of enclosure configurations, is most prominent below the frequency range of 1-8 kHz. It was hypothesized that the inclusion of diffraction in Eq. 2.6 may be necessary to predict the power radiated by the enclosure at lower frequencies. A simple example of diffraction is shown in Figure 5.1, which shows a source and a wall element, with an obstruction between them. The gray color represents the sound pressure from the source on the element, and the brightness represents the strength. The shadow zone (low sound pressure) is more similar to the unobstructed solid angle of the source to the element for high frequencies. For low frequencies the shadow zone is much smaller.

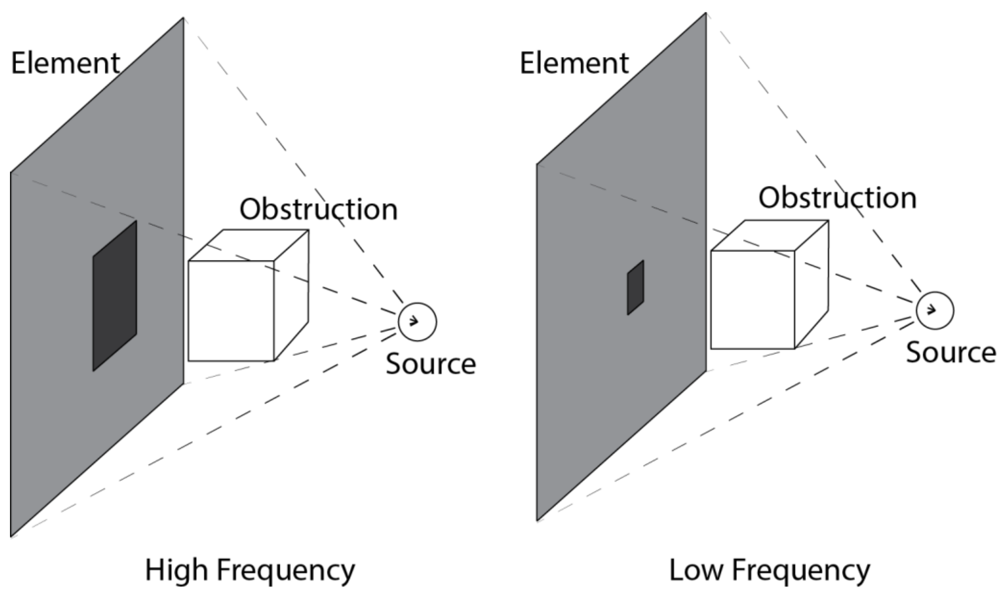


Figure 5.1 – The sound pressure incident on an element with an obstruction between the element and the source. The gray color represents the sound pressure, brighter being a higher sound pressure. The dark gray area represents the shadow zone, or areas that the obstruction affects the sound pressure incident on the element.

An attempt was made to determine a diffraction correction (κ) that would account for this effect. The value of κ was determined by a computational experiment, and is given by the following equations:

$$\kappa = \begin{cases} 1, & y > 1 \\ y, & 0 \le y \le 1 \end{cases} \quad (5.1)$$

$$y = 15.791e^{-0.4073x} \quad (5.2)$$

$$x = 10\log_{10}\left(\frac{f}{100}\right) + 1 \quad (5.3)$$

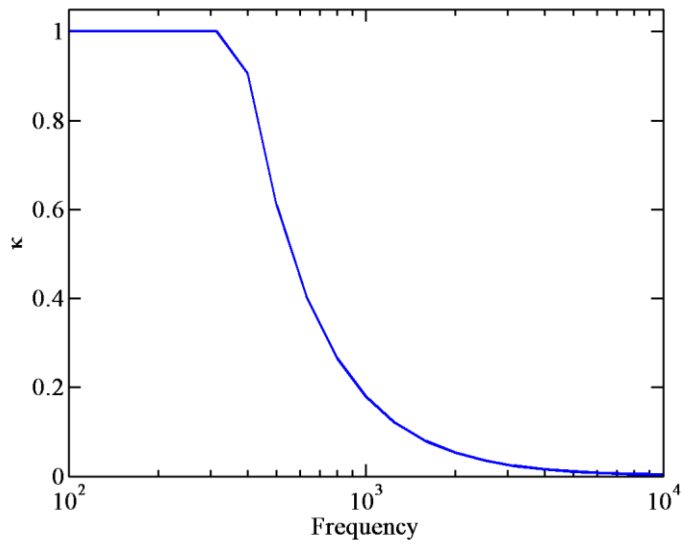


Figure 5.2 – Diffraction correction (κ) as a function of frequency.

In Eq. 5.3, f is frequency, which means that x is effectively a 1/3 octave band index where 100 Hz = 1. Eq. 5.2 gives the value of κ, and Eq. 5.1 shows that κ never goes above 1. Figure 5.2 shows κ across all frequencies. The total solid angle (Ω_T) was then redefined as the sum of the unobstructed solid angle (Ω_U) and the obstructed solid angle (Ω_O) multiplied by κ.

$$\Omega_T = \Omega_U + \kappa * \Omega_O \quad (5.4)$$

At high frequencies, κ is nearly 0, and thus $\Omega_T = \Omega_U$. This is consistent with the simple model. For low frequencies, $\kappa > 0$, which means Ω_T becomes larger, or the obstruction effectively becomes smaller. This concept is shown in Figure 5.3. Eq. 5.5 is the equation for radiated power with the redefined total solid angle.

$$\Pi_{rad} = \Pi \left[\sum_j \frac{\Omega_{Sj}\tau_j}{4\pi} + \left(1 - \frac{\sum_i \Omega_{Si}(\alpha_i + \tau_i)}{4\pi}\right) \sum_j \sum_{k \neq j} \frac{(1-\alpha_k-\tau_k)\Omega_{kj} S_k \tau_j}{2\pi S(\bar{\alpha}+\bar{\tau})} \right] \quad (5.5)$$

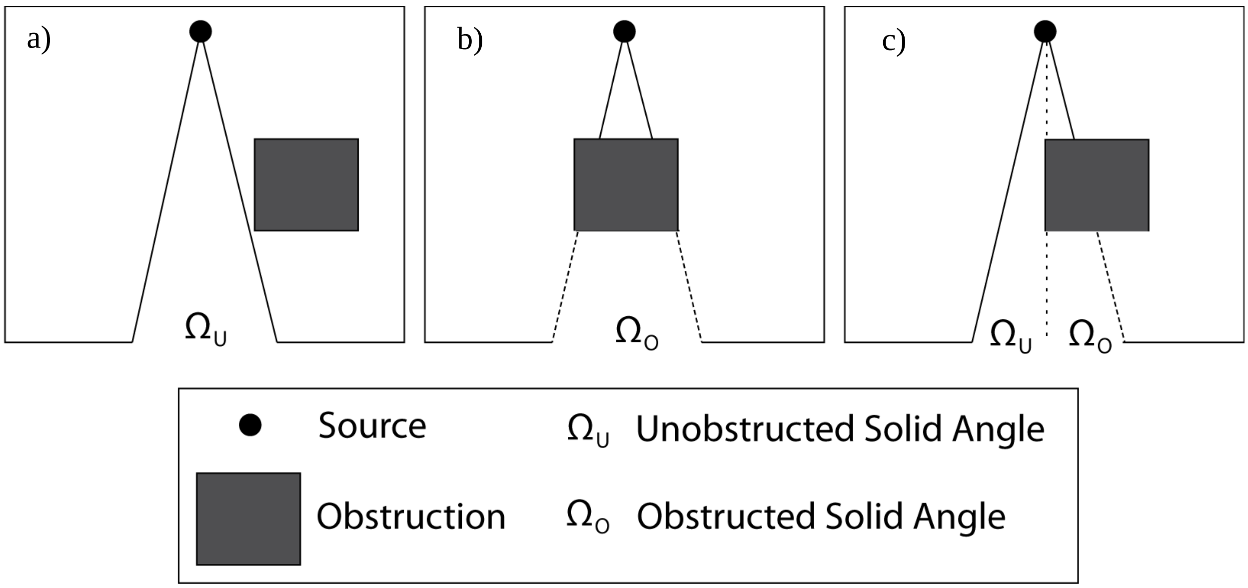


Figure 5.3 – a) Completely unobstructed source, b) completely obstructed source, c) partially obstructed source.

The diffraction correction was found to give larger errors in both the high and low frequency regions. Because the research was primarily to analyze how well the model works in the 1-8 kHz region, the diffraction correction was not further investigated.

5.3.2 Directivity

The equation used for radiated sound power here assumes an omni-directional source. A quick experiment was done to see the effects of directivity on insertion loss for different enclosure configurations. A compression driver without a horn was used as a more directional source. The driver was placed in the enclosure on its back, or on its side facing each of the cardinal directions to see the effect of the directionality on insertion loss. The configurations effected most by the directivity were the cases with considerable absorption near the source. At frequencies above 2 kHz, differences of up to 10 dB or more are seen between the directions. This was for a very directional source compared to a dodecahedron; however, more investigation may be warranted.

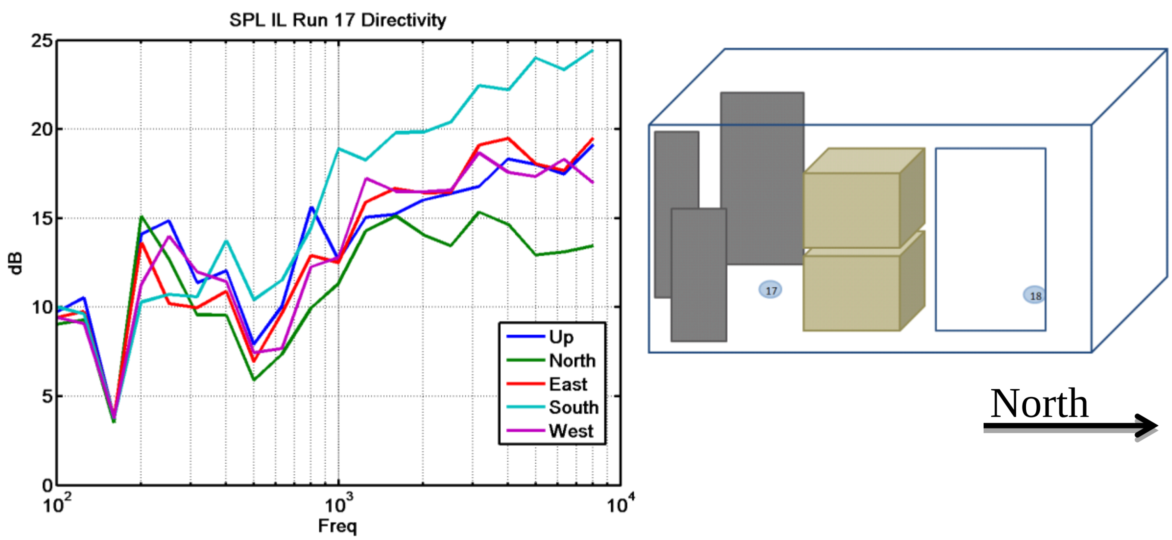
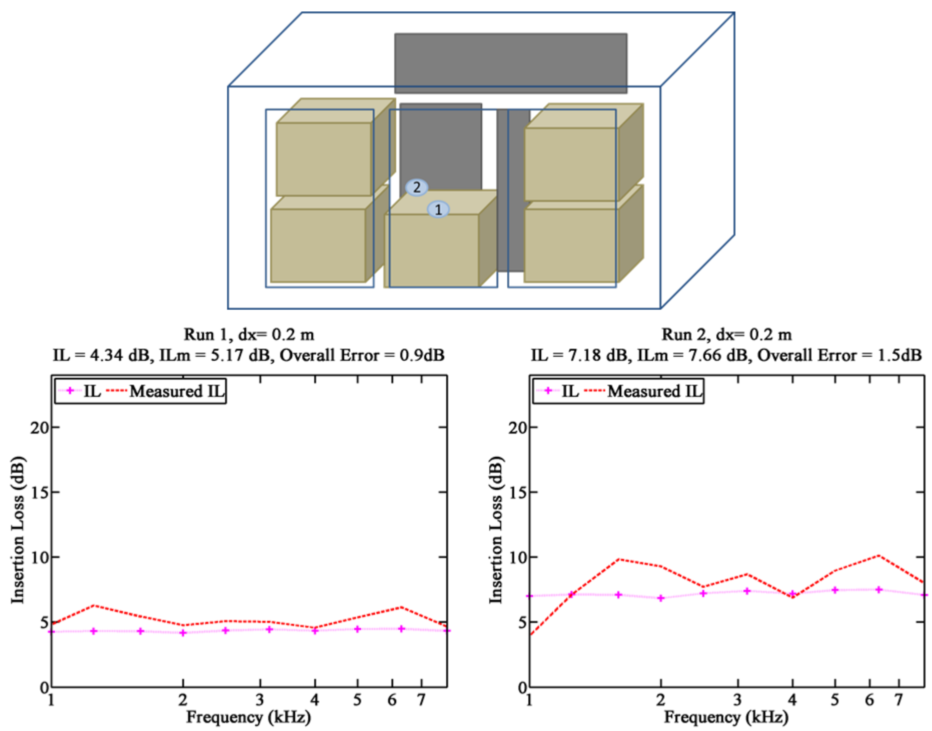


Figure 5.4 – Insertion loss of the compression driver in different orientations for configuration 17.

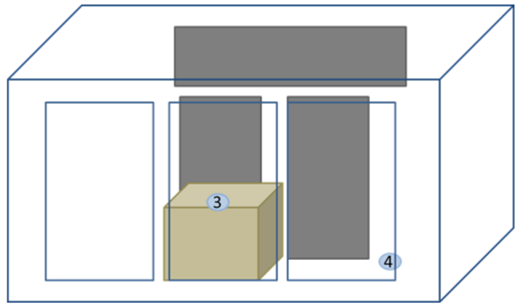
5.4 DOE and Results

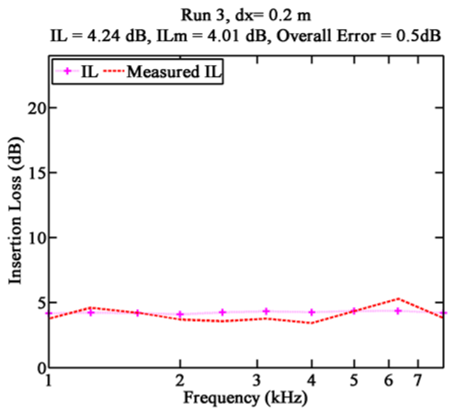
The following figures show the enclosure configurations, along with the results from the measurements and the model. There were 18 enclosure setups (absorption, apertures and obstructions), each with 2 source positions, totaling 36 different enclosure configurations. Above the graphs, the total IL levels are noted, IL meaning total insertion loss predicted by the model (magenta), ILm is the total insertion loss of the measurement (red), and the overall error is the error discussed in Sec. 4.1.

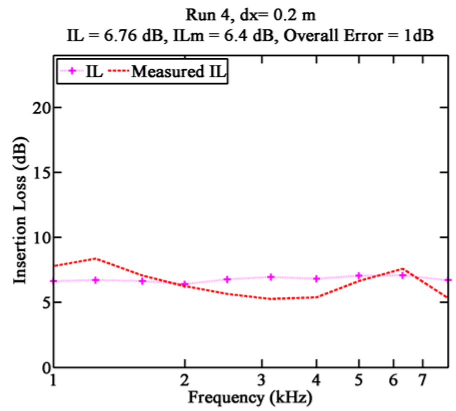
Configurations 1 and 2



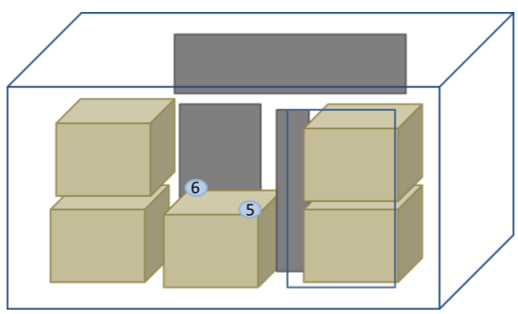
Configurations 3 and 4

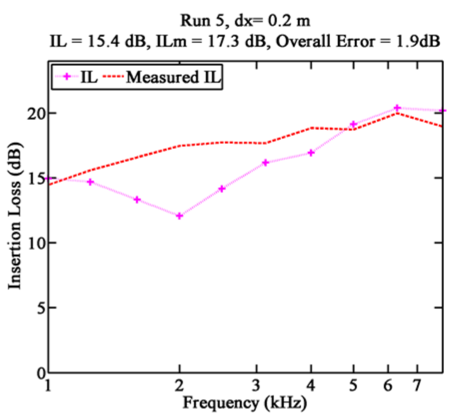


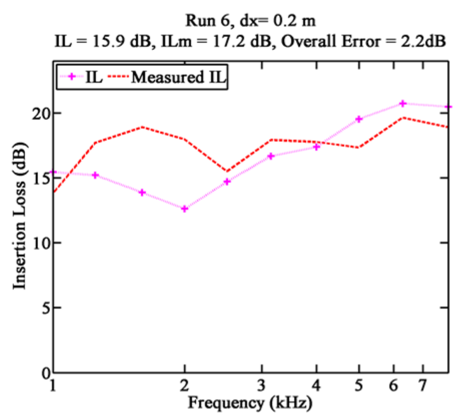




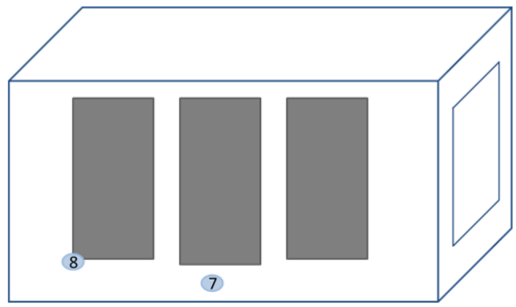
Configurations 5 and 6

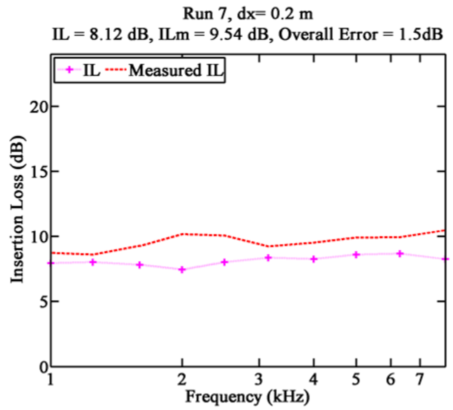


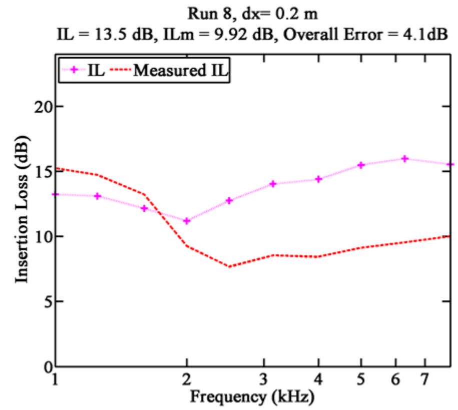




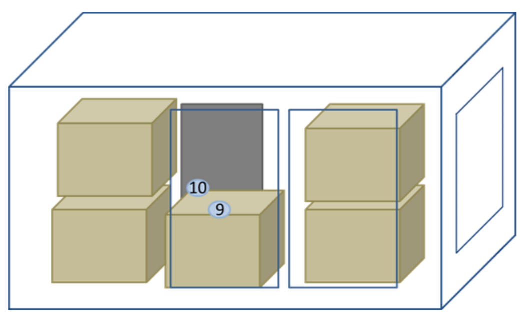
Configurations 7 and 8

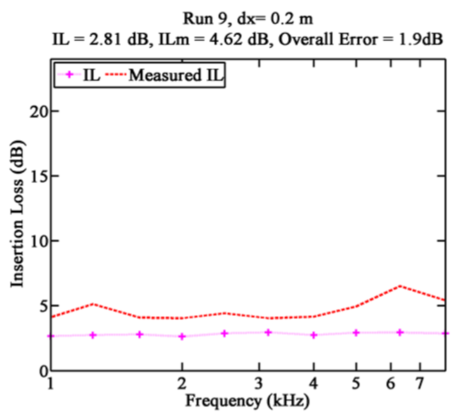


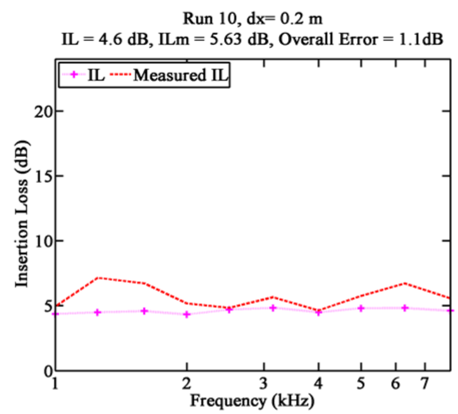




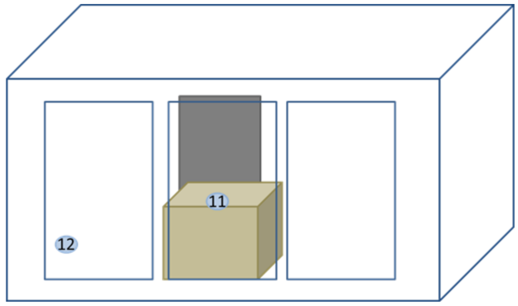
Configurations 9 and 10

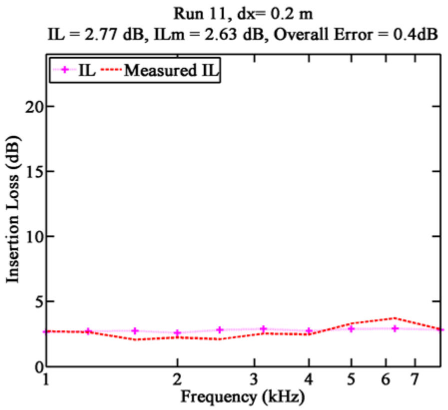


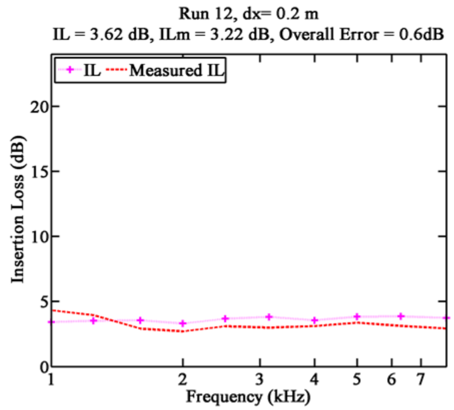




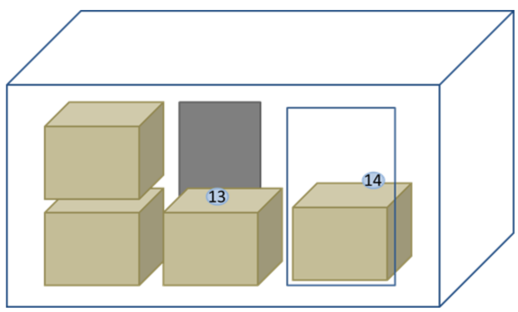
Configurations 11 and 12

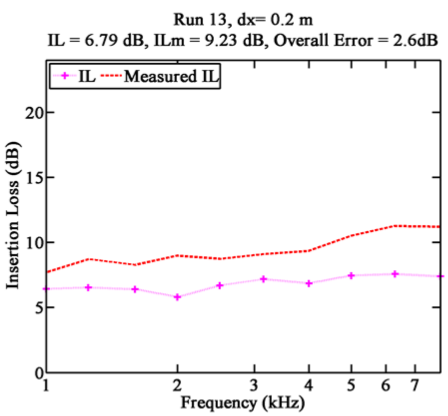


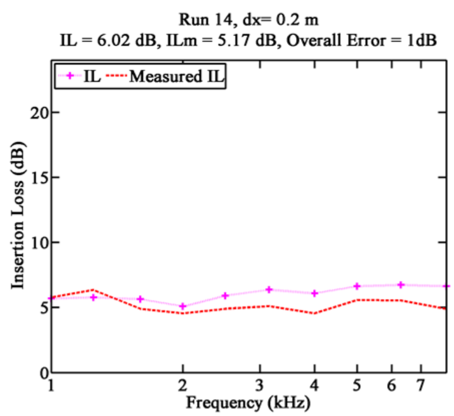




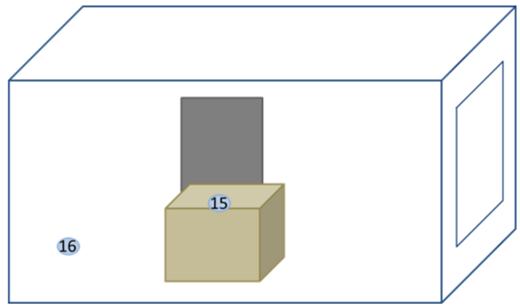
Configurations 13 and 14

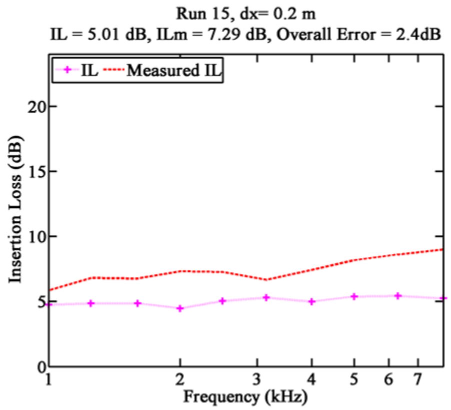


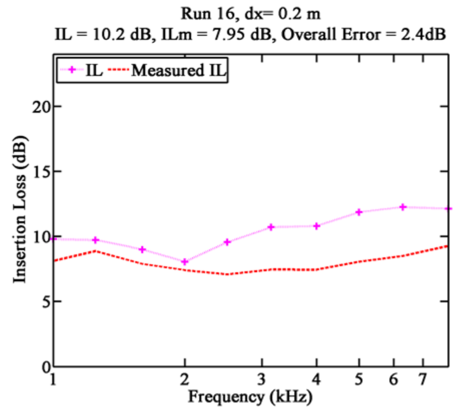




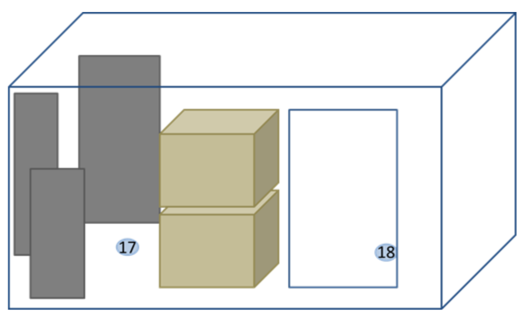
Configurations 15 and 16

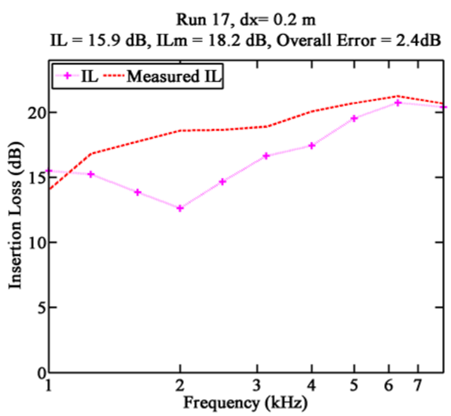


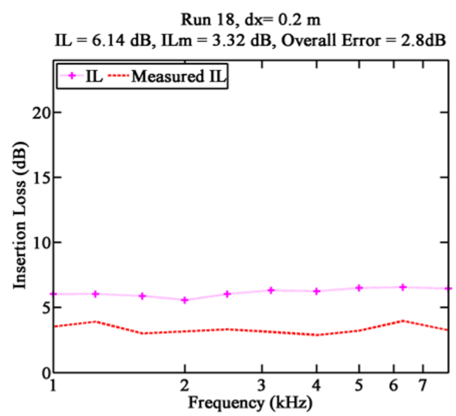




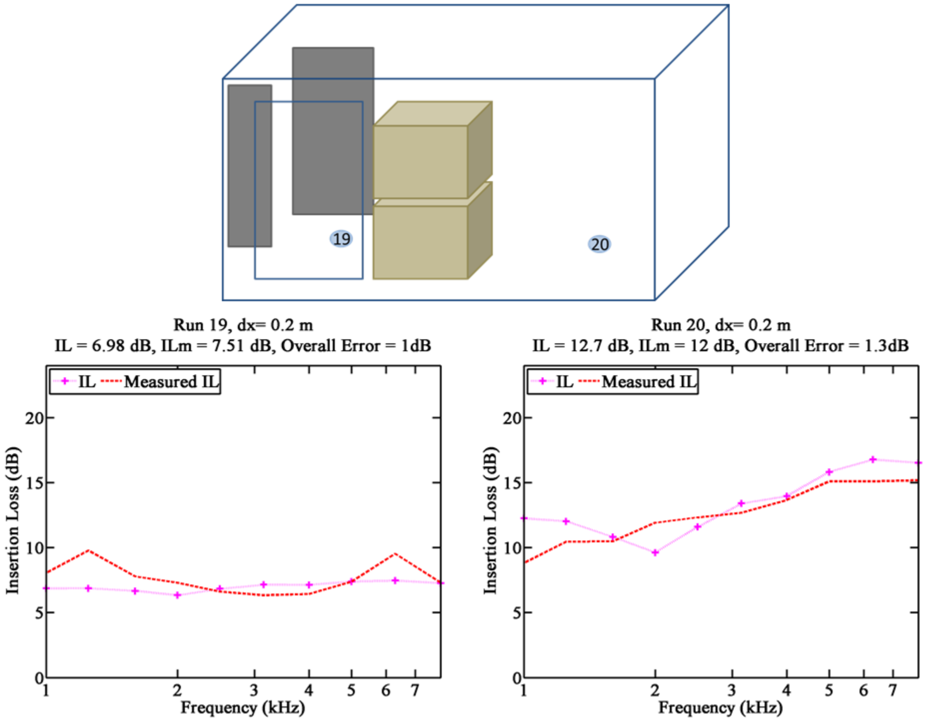
Configurations 17 and 18



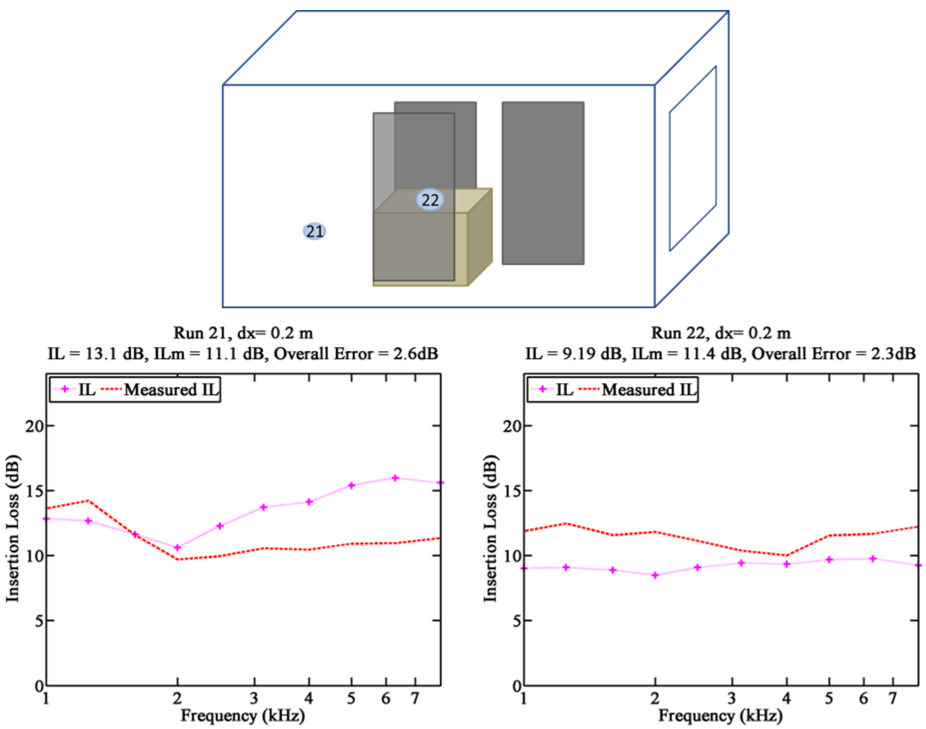




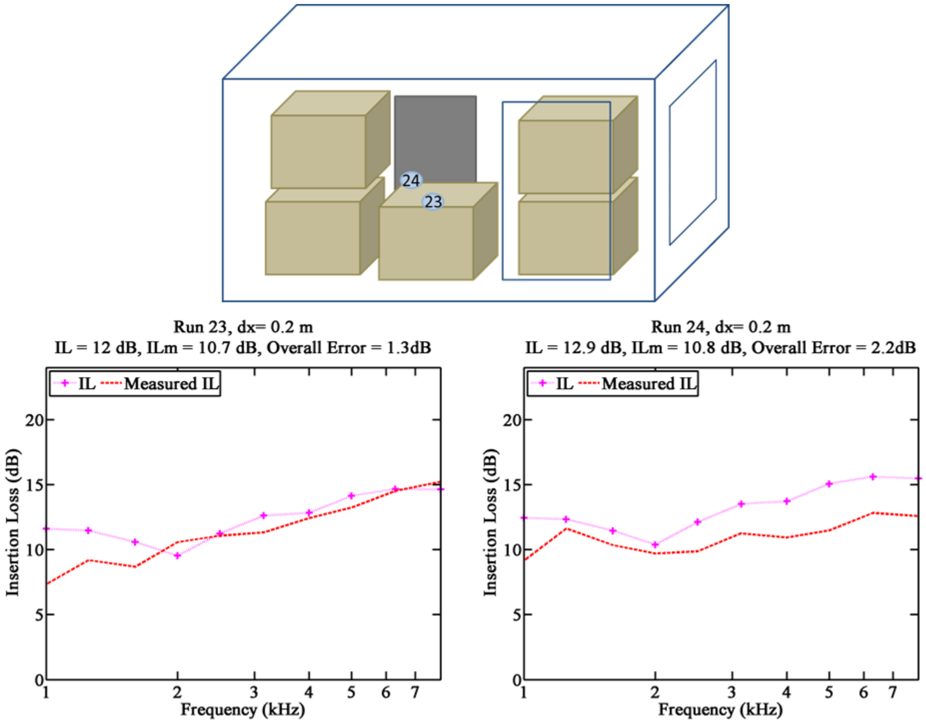
Configurations 19 and 20



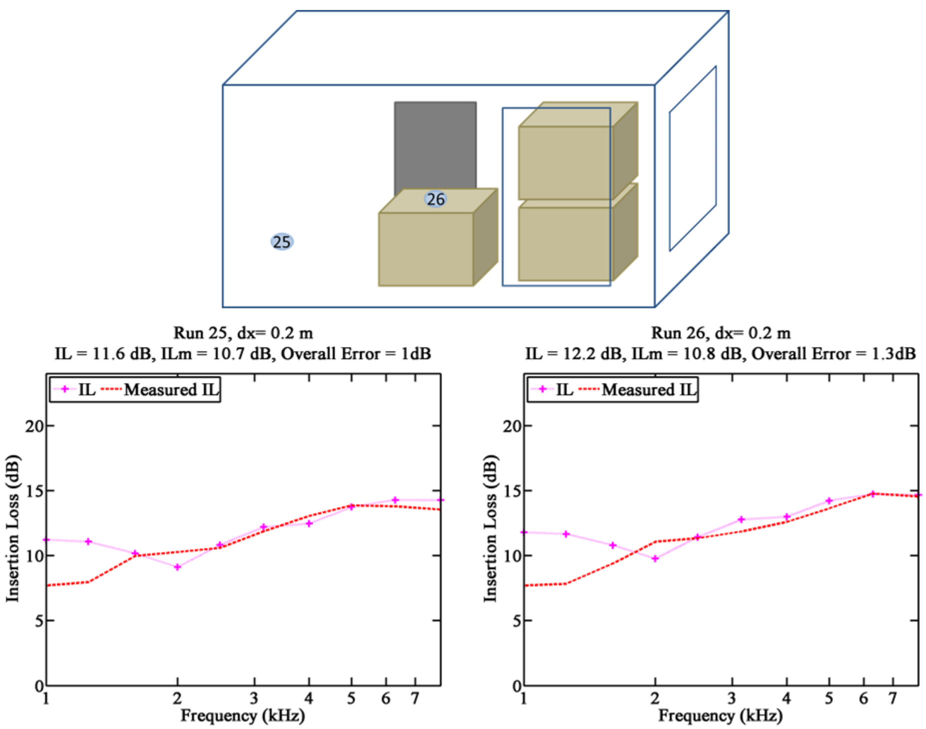
Configurations 21 and 22



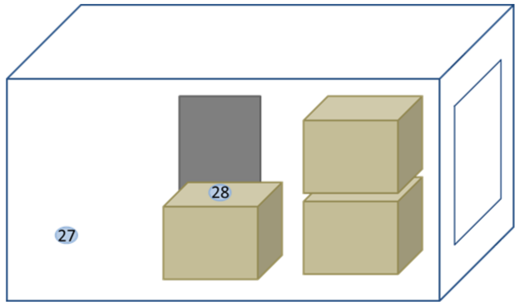
Configurations 23 and 24

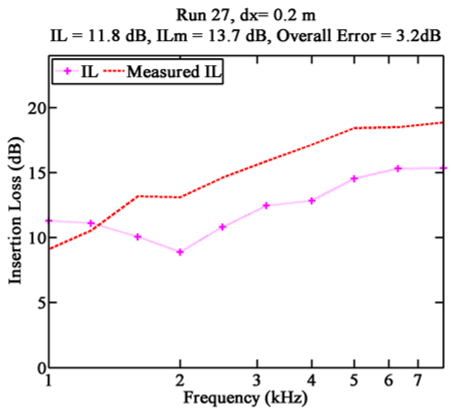


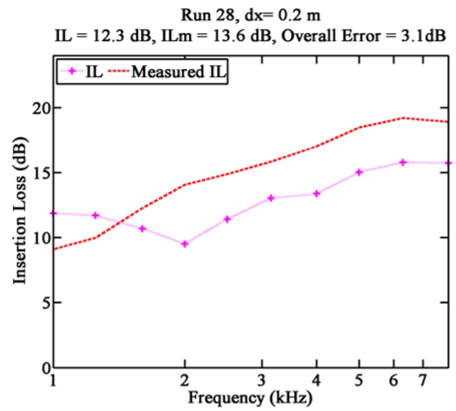
Configurations 25 and 26



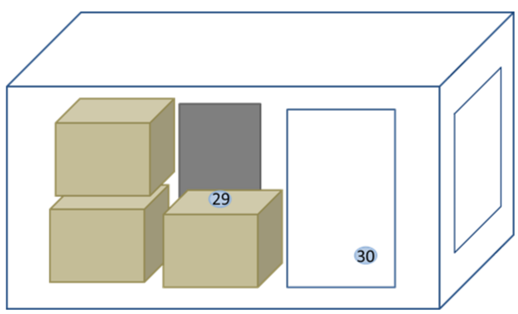
Configurations 27 and 28

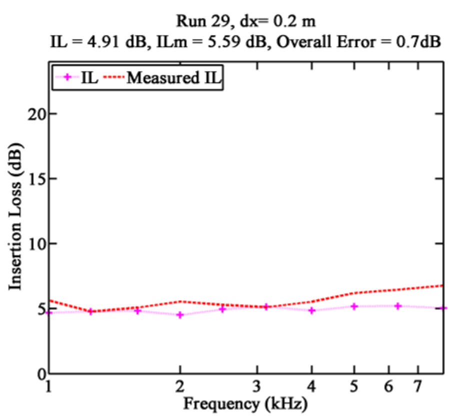


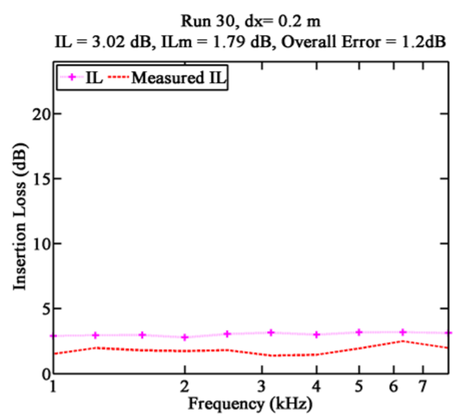




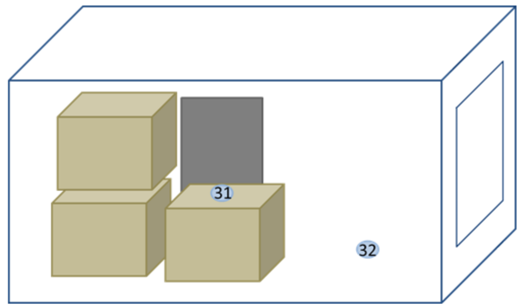
Configurations 29 and 30

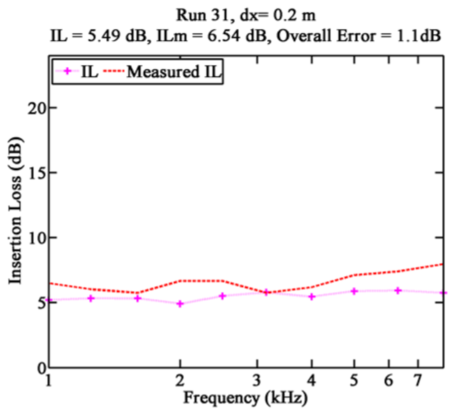


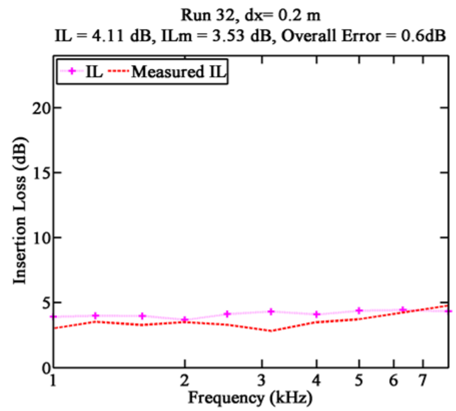




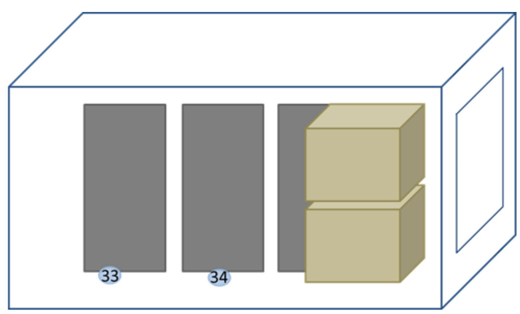
Configurations 31 and 32

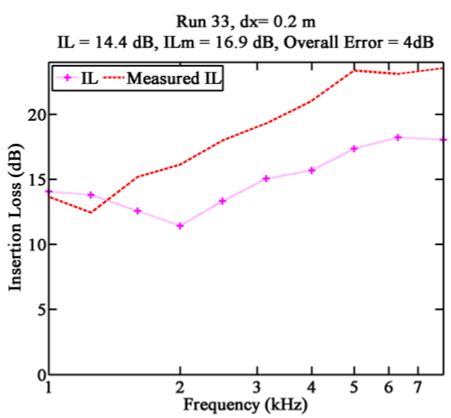


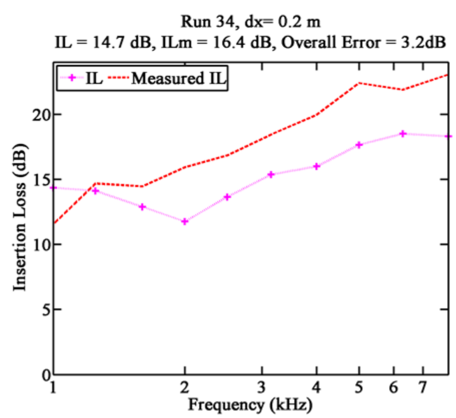




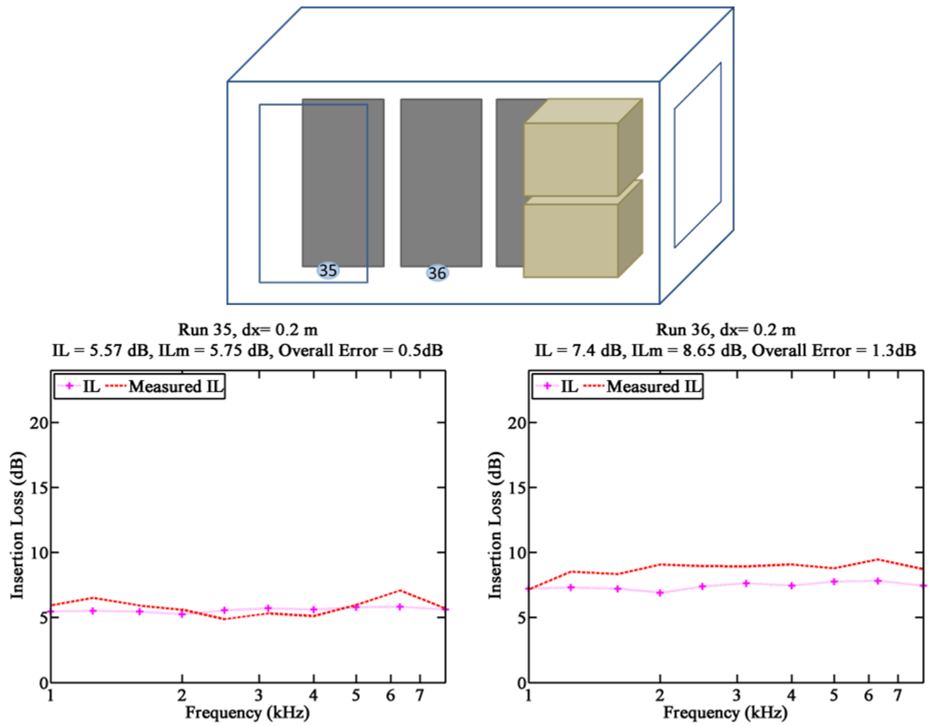
Configurations 33 and 34







Configurations 35 and 36



5.4 Psuedo Code and MatLab Scripts

CATSimpleModel.m Pseudo-Code

The **CATSimpleModel.m** script is the main script used to determine the sound power radiated from an enclosure. The following is the pseudo-code for the MatLab program. Sonic + has a method of designing an enclosure, and so for Sonic + users the relevant code will be from the SolidAngleFunction.m and afterward.

- Define constants used in code: d, dx, frequency (1/3 octave) and frequency band width (1-8kHz)

- **Materials.m** Function: Defines absorption and transmission coefficients of materials used in enclosure

- **DefineEnclosureDOE.m:** Creates enclosure geometry and determines total volume and surface area percentages for enclosure. Defines each element and its absorption and transmission.
  - **getBlockPanels.m** – Breaks a block into 6 panels. Each panel is an element with dimensions and absorption and transmission coefficients.
  - **getAbsorberPanels.m** – Creates absorption and aperture elements with appropriate constants and dimensions. Uses **MakeCoord.m** for the coordinates and **getabsSA.m** for surface area.

- **SolidAngleFunction.m:** Finds solid angles of enclosure geometry and defines important variables used in the sound power calculation, i.e. ReverbPercent and Theta (see below).
  - **getSolidAngleObstruction.m** – Breaks each element into a mesh, redefines coordinates so that the source is at the source and determines if there is another element obstructing the source. Then uses getSolidAng to calculate the solid angle if there is no obstruction.
    - **getSolidAng.m –** Uses rotaCoord.m to rotate the coordinates of the element into the x-z plane, then calculates the solid angle of that element using the solid angle approximations mentioned in section C [6].
  - **getA2AangObstruction.m** – Breaks an element into mesh elements. Then uses getSolidAngleObstruction.m to find the solid angle from the mesh element center to an element, then multiplies that solid angle by dx^2 and moves to the next mesh element.
  - Defines OmegaSO, and OmegaWO and calculates the ReverbPercent and Theta variables.

- **Parameterize.m:** Finds the parameters of the configuration, obstruction, aperture, and absorption percentages, along with solid angle information. Side2Ap keeps track of the obstruction to aperture solid angle, which is used to determine which coefficient is applied to the PI_Ap term.

- Determine the sound power radiated from each element for each frequency. The code is:

  PI_rad(j,f)=OmegaSO(j)*Elements(j).tau(f)/(4*pi)+(1-ReverbPercent(f)).*(Theta(j,f)*Elements(j).tau(f)/(2*pi*(Salphabar1(f)+Staubar1(f))));

  - This corresponds to Eq. C.6, with ReverbPercent = Eq. C.4 (with the source power (Π) cancelled) and Theta =$(1-\alpha_k-\tau_k)\Omega_{kj}S_k$. These quantities are calculated in SolidAngleFunction.m.
  - In the code above, PI_rad(j,f) is a matrix of the j-th element, and f-th one-third octave band. The power radiated from any individual element at any given one-third octave band is given by picking the appropriate j-th and f-th index.

- Once the sound power radiated from the enclosure has been calculated, insertion loss can be calculated and plotted using

  IL=10*log(PI/(PI*PI_rad)=10*log(1/PI_rad)

- Plot the insertion loss and write it to a file

CatSimpleModel.m

```
%% Simple Model - Function for each of the different properties
clear all; close all;
tic
%% Set Plotting Defaults
set(0,'DefaultAxesFontName','Arial');
set(0,'DefaultAxesFontSize',14);
set(0,'DefaultAxesFontWeight','demi')
set(0,'DefaultAxesLineWidth',1);
set(0,'DefaultLineLineWidth',2.5);
set(0,'DefaultLineMarkerSize',6);
%% Define constants
  Side2ApMat=[];
  Source2AbMat=[];

  d=.001; %Small amount of distance between absorption/obstructions and enclosure walls.
  dx=.2; %Size of grids used in the model.

  % Frequency in 1/3 octave bands

Freq=[31.5;40;50;63;80;100;125;160;200;250;315;400;500;630;800;1000;1250;1600;2000;2500;3150;4000;5000;6300;8000;10000;12500;16000
];
  FreqLow=1000;
  FreqHigh=8000;

  % Creates an index for the frequency band
  for i = 1:length(FreqLow);
    FreqInd = find(Freq>=FreqLow(i) & Freq<=FreqHigh(i));
  end

%% Define Material Constants
  Material='MDF'; %'MDF'
  [aMDF,tauMDF,tauObs,aAbs,tauAbs,aAp,tauAp] = Materials(Material);

%% Simple Model
```

```
runs=1:36; % Runs through each of the configurations
for l=runs
%% Define Enclosure
   run=l;
   name=['Run ',num2str(run)];
% Standard DOE enclosure configurations

[Elements,apInd,abInd,blockInd,SBlock,VBlock,Sopening,SabSize,Swalls,Xs,X,Y,Z]=DefineEnclosureDOE(d,run,aMDF,tauMDF,tauObs,aAbs,
tauAbs,aAp,tauAp);

      Sopening1 = sum(Sopening);          %Surface area of all openings (Matrix of SA in every frequency Bin)
      Sabsize1 = sum(SabSize);            %Surface area of all absorbers
      SBlock1 = sum(SBlock);              %Surface Area of 4 sides of blocks
      Salphabar1 = ((Swalls -Sopening1-Sabsize1+SBlock1).*aMDF(:,1) + Sopening1.*aAp(:,1) + Sabsize1.*aAbs(:,1)); %S*alphabar
      Staubar1 = ((Swalls - Sopening1-Sabsize1+SBlock1).*tauMDF(:,1)+ Sopening1.*tauAp(:,1)+Sabsize1.*tauAbs(:,1)); %S*taubar
%% Calculate Solid Angles
   [OmegaSO,OmegaWO,ReverbPercent,Theta]=SolidAngleFunction(Xs,Elements,dx,apInd,Freq);

%% Parameters - Calculates parameters of enclosure configuration
   [Side2ApMat1,Source2AbMat1,Side2ApSum]=parameterize(blockInd,apInd, abInd, Elements, Xs,run,VBlock, SabSize, Swalls,
Sopening,OmegaSO,OmegaWO,X,Y,Z);
   Side2ApMat=[Side2ApMat;Side2ApMat1]; % Matrix used to keep track of the obstruction to aperture information
   Source2AbMat=[Source2AbMat;Source2AbMat1]; % Matrix keeps track of source to absorption information

EnclVol= (X*Y*Z); % Enclosure volume
SolidAngleAp(l)=sum(OmegaSO(apInd)); % Solid Angle Source to Aperture
SolidAngleAb(l)=sum(OmegaSO(abInd)); % Solid Angle Source to Absorption
SolidAngleOb(l)=sum(OmegaSO(blockInd)); % Solid Angle Source to Obstruction
SolidAngleEncl(l)=sum(OmegaSO(1:6)); % Solid Angle Source to Enclosure
%% The Equation - (CAT IL Document Section C)
   %%IL= 10*log10(Pi/Pi_rad) , where Pi_Rad_j == Pi*(direct_field_j
   %%+reverberant_field_term) (see CAT Insertion Loss Model Document,
   %%section C)

   PI_rad=zeros(length(Elements),length(Freq));
   for j=1:length(Elements)
      for f=1:length(Freq)
   PI_rad(j,f)=OmegaSO(j)*Elements(j).tau(f)/(4*pi)+(1-ReverbPercent(f)).*(Theta(j,f)*Elements(j).tau(f)/(2*pi*(Salphabar1(f)+Staubar1(f))));
      end
   end

   % If statement for Aperture terms - multiply by fudge factor
   C1=.2; % Offset for low solid angle source to Ap
   C2=.1; % Offset for low solid angle source to Ap and high obstruction to Ap solid angle
   C3=1;

   if SolidAngleAp(run)/(4*pi)*100<2 % OmegaSAp percentage < 2%
      if Side2ApSum/(2*pi)*100 > 10 % OmegaOAp percentage > 10%
      PI_aper=sum(PI_rad(apInd,:),1).*C2; % The sum of aperture terms multiplied by C2
      display(['Run ',num2str(run),'OmegaSAp<2%,OmegaOAp>10%'])
      else
      PI_aper=sum(PI_rad(apInd,:),1).*C1; % The sum of aperture terms multiplied by C1
      display(['Run ',num2str(run),'OmegaSAp<2%'])
      end
   else % If the above conditions are not met, the aperture term isn't changed
   PI_aper=sum(PI_rad(apInd,:),1);
   end

   % Zeros out the aperture terms in the PI_rad matrix
   for i=1:length(apInd)
      PI_rad(apInd(i),:)=zeros(1,length(Freq));
   end

   % Redifines PI_rad matrix with new aperture terms
   PI_rad=[PI_rad;PI_aper];

   % Calculate the ratio of sound power divided by the sound power
   % radiated from the enclosure X1=PI/(PI*PI_rad).  The PI terms cancel
   % out of the equation.
   X1=1./(sum(PI_rad,1));
```

```
    % Coefficients for the insertion loss calculation
    a1=0; a2=10; a3=0;

    % Calculate insertion loss using a polynomial fit
    IL(run,:) = a1 + a2.*log10(sum(X1,1).*C3) + a3.*(log10(sum(X1,1))).^2;

    % Read in the experimental data
    Exp = dlmread('Measurements\Extra12Test-IL-1-5-13.txt');
    Measured(run,:)=Exp(run+1,:);

    % Calculate overall insertion loss and errors
    OAILm(run)=10*log10(length(Measured(run,FreqInd-5))./(sum(10.^(Measured(run,FreqInd-5)./-10))));
    OAIL(run)=10*log10(length(IL(run,FreqInd))./(sum(10.^(IL(run,FreqInd)./-10))));
    resIL(run,:)=IL(run,FreqInd)-Measured(run,FreqInd-5); % Frequency dependent error
    ResErr(run)=mean(abs(resIL(run,:))); % Overall error

    display(['Run ',num2str(run),' -> ',num2str(round(OAIL(run)*10)/10),' dB'])

%% Plotting
    figure1 = figure;
    set(figure1,'Visible','off');
    figuresize(10,8,'inches');
    modelStr = (['Run ',num2str(run)]);
    semilogx(Freq(FreqInd),IL(run,FreqInd),':m+')
    hold on
    semilogx(Freq(FreqInd),Measured(run,FreqInd-5),'--r')
    hold off
    ylabel('Insertion Loss (dB)')
    xlabel('Frequency (Hz)')
    title({[modelStr,', dx= ',num2str(dx),' m'];['IL = ',num2str(OAIL(run),3),' dB, ILm = ',...
        num2str(OAILm(run),3),' dB, Res Error = ',num2str(ResErr(run),3),'dB']});
    legend('IL','Measured IL','Location','SouthEast','Orientation','horizontal')
    saveas(gcf,['Figures\',modelStr,'.png'],'png');

%% Print to file
dlmwrite(['Results\',name,'_SM.txt'],[X1;Freq';IL(run,:)],'newline','pc','delimiter','\t');
end

toc
```

Materials.m Pseudo-Code

The function uses a switch/case function to define the materials used in the enclosure. At this point, only one enclosure type was tested, only MDF was used.

Materials.m

```
function [aMDF,tauMDF,tauObs,aAbs,tauAbs,aAp,tauAp] = Materials(Mat)
%Absorption and Transmission Constants
    switch Mat
        case 'MDF'
    % MDF values
    aMDF = [0.01;0.02;0.025;0.03;0.035;0.04;0.04;0.05;0.05;0.045;...
        0.04;0.05;0.05;0.05;0.05;0.05;0.0543;0.0607;0.0517;0.0642;...
        0.0671;0.0529;0.0632;0.0635;0.0638;0.06;0.055;0.05]; % alpha of MDF
    tauMDF=load('Measurements\taudodec.txt');
        case 'Other' % Use to define another material
    end

tauObs=zeros(28);
```

```
%Absorption values
   aAbs = [0.01 0.02 0.025 0.03 0.04 0.05 0.055 0.1 0.12 0.19 0.3 0.44 0.58 0.7 ...
      0.8 0.98 1 1 1 1 1 0.99 1 1 0.9 0.9 0.9 0.85]'; %alpha of Absorbers
   tauAbs = tauMDF;  %%%%%%%%%%%%%%

%Aperture has an absorption of 0, transmission of 1
   aAp = zeros(28,1);
   tauAp= ones(28,1);

return
```

DefineEnclosure.m Pseudo-Code

This code is not necessary for Sonic + users, since the enclosure geometry is already determined by Sonic +.

- Size of the enclosure

- Define enclosure sides (Defined as a block facing inward using getBlockPanels)
    - Dimensions of the block
    - Front Left Corner
    - getBlockPanels.m – creates 6 elements for the block, each side of the block is created as an element.
        - MakeCoord.m – creates the coordinates of each panel based on the direction it is facing.
    - Determine surface area of the enclosure.

- Define aperture elements (uses cases for each enclosure configuration)
    - apInd creates an index to keep track of the aperture elements
    - Determine where the aperture is located [bottom left corner; top right corner]
    - getAbsorberPanels.m – creates an aperture element based on geometry. This function is also used to create absorption elements as they are both assumed to be two-dimensional.
    - Sopening sums the total aperture surface area.

- Define absorption elements
    - SabSize is the total absorption surface area.
    - abInd creates an index for the elements that are absorption panels.
    - Determine where the absorption panel goes.
    - getAbsorberPanels.m – creates the absorption panel element

- Define Obstructions
    - Size of the obstruction.
    - Obstruction placement.
    - blockInd keeps track of which elements are obstructions.
    - getBlockPanels.m – again creates six elements (every side) for each obstruction.
    - Determine the total surface area and volume of the obstructions.

- Source Positions
  - Each configuration has a different source position, defined with a switch/case function.

DefineEnclosure.m

```
function [Elements,apInd,abInd,blockInd,SBlock,VBlock,Sopening,SabSize,Swalls,Xs,X,Y,Z]=...
  DefineEnclosureDOE(d,run,aMDF,tauMDF,tauObs,aAbs,tauAbs,aAp,tauAp)
%Defines a blank matrix from the Aperture and Absorbtion index numbers.
  apInd=[];
  abInd=[];
  blockInd=[];
  Elements=[];
  %Constants
  %Size of Enclosure, looking towards the front of the box (the side with
  %three openings), the origin is the bottom left corner.
  X = 1.95;
  Y = .745;
  Z = 1.2;

% Define Enclosure
  blockDim = [X,Y,Z];
  corner = [0 0 0]; %Front left corner
  %getBlockPanels assumes that the object is parrallel to xy,yz,and zx
  %planes.  Breaks enclosure into 6 panels, each a structure with the
  %following fields: corners, center, facing, alpha, SA of panel.
  blockElement = getBlockPanels(corner,blockDim,aMDF,tauMDF,-1);%Last parameter is -1 because the enclosure wall is facing inward.
  %Calls function called getBlockPanels(which takes the information given
  %above - Enclosure dimensions, absorption, and creates a stucture
  Elements=[Elements, blockElement];
  Swalls = (blockDim(1)*blockDim(2)*2 + blockDim(2)*blockDim(3)*2 + blockDim(1)*blockDim(3)*2); %Total surface area of walls

% Define Apertures
  Sopening=0;
  %Left Aperture
    switch run
      case {1,2,3,4,11,12,19,20,35,36}
        apInd=[apInd,length(Elements)+1]; %apInd keeps track of which elements are apertures
        corners = [0.12, d, 0.08; 0.58, d, 1.012];
        %[Bottom Left; Top Right],aperture is treated the same as an absorber in previous codes,
        %should that be the case when using transmission loss? Need to
        %write another similar function getApPanels
        facing = [0 1 0]; %faces from surface to inside of the enclosure
        Elements = [Elements,getAbsorberPanels(corners, aAp, tauAp, facing, d)];
        Sopening = Sopening + (corners(2,1)-corners(1,1))*(corners(2,3)-corners(1,3));% Total SA of openings for each run
    end
  %Middle Aperture
    switch run
      case {1,2,3,4,9,10,11,12}
        apInd = [apInd,length(Elements)+1];
        corners = [0.725, d, 0.08; 1.20, d, 1.012];
        facing = [0 1 0];
        Elements = [Elements,getAbsorberPanels(corners, aAp, tauAp, facing, d)];
        Sopening = Sopening + (corners(2,1)-corners(1,1))*(corners(2,3)-corners(1,3));
    end
  %Right Aperture
    switch run
      case {1,2,3,4,5,6,9,10,11,12,13,14,17,18,23,24,25,26,29,30}
    apInd = [apInd,length(Elements)+1];
    corners = [1.373, d, 0.08; 1.811, d, 1.012];
    facing = [0 1 0];
    Elements = [Elements,getAbsorberPanels(corners, aAp, tauAp, facing, d)];
    Sopening = Sopening + (corners(2,1)-corners(1,1))*(corners(2,3)-corners(1,3));
    end
  %Right Side Aperture
    switch run
```

```
        case {7,8,9,10,15,16,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36}
      apInd = [apInd,length(Elements)+1];
      corners = [X-d, .095, 0.085; X-d, .685, 1.08];
      facing = [-1 0 0];
      Elements = [Elements,getAbsorberPanels(corners, aAp, tauAp, facing, d)];
      Sopening = Sopening + (corners(2,2)-corners(1,2))*(corners(2,3)-corners(1,3));
      end

% Define Absorption

  thk = .025;
  SabSize = 0;
  % A
    switch run
      case {1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36}
      SabSize = SabSize + 0.6*1.18; %Total surface area for absorption
      abInd = [abInd, length(Elements)+1]; %Index for absorption elements
      corners = [0.7, Y-d, 0.0; 1.3, Y-d, Z]; %[Bottom left; top right]
      facing = [0 -1 0]; %Unit vector facing in direction from surface to inside of enclosure
      Elements = [Elements,getAbsorberPanels(corners, aAbs, tauAbs, facing,d)]; %Same as aperture element structures
    end
  % B
    switch run
      case {1 2 5 6}
      SabSize = SabSize + 0.6*1.18;
      abInd = [abInd, length(Elements)+1];
      corners = [1.325-d, Y-.6,0.0; 1.325-d, Y, Z];
      facing = [-1 0 0];
      Elements = [Elements,getAbsorberPanels(corners, aAbs, tauAbs, facing,d)];
    end
  % C
    switch run
      case {1 2 3 4 5 6}
      SabSize = SabSize + 0.6*1.18;
      abInd = [abInd, length(Elements)+1];
      corners = [0.7, Y-.6, Z-d; 1.88, Y, Z-d];
      facing = [0 0 -1];
      Elements = [Elements,getAbsorberPanels(corners, aAbs, tauAbs, facing,d)];
    end
  % D
    switch run
      case {3 4 7 8 21 22 33 34 35 36}
      SabSize = SabSize + 0.6*1.18;
      abInd = [abInd, length(Elements)+1];
      corners = [1.3, Y-d, 0; 1.9, Y-d, Z];
      facing = [0 -1 0];
      Elements = [Elements,getAbsorberPanels(corners, aAbs, tauAbs, facing,d)];
    end
  % E
    switch run
      case {7 8 17 18 19 20 33 34 35 36}
      SabSize = SabSize + 0.6*1.18;
      abInd = [abInd, length(Elements)+1];
      corners = [.1, Y-d, 0; 0.7, Y-d, Z];
      facing = [0 -1 0];
      Elements = [Elements,getAbsorberPanels(corners, aAbs, tauAbs, facing,d)];
      end
  % F
    switch run
      case {17 18}
      SabSize = SabSize + 0.6*1.18;
      abInd = [abInd, length(Elements)+1];
      corners = [.1, d, 0; 0.7, d, Z];
      facing = [0 1 0];
      Elements = [Elements,getAbsorberPanels(corners, aAbs, tauAbs, facing,d)];
      end
  % G
    switch run
      case {17 18 19 20}
      SabSize = SabSize + 0.6*1.18;
```

```
      abInd = [abInd, length(Elements)+1];
      corners = [d, .145, 0; d, .745, Z];
      facing = [1 0 0];
      Elements = [Elements,getAbsorberPanels(corners, aAbs, tauAbs, facing,d)];
      end
  %H
    switch run
      case {21 22}
      SabSize = SabSize + 0.6*1.18;
      abInd = [abInd, length(Elements)+1];
      corners = [0.7, d, 0.0; 1.3, d, Z];
      facing = [0 1 0];
      Elements = [Elements,getAbsorberPanels(corners, aAbs, tauAbs, facing,d)];
      end

% Define Obstructions

  SBlock=0;
  VBlock=0;
  % Block I
  switch run
    case {1 2 5 6 9 10 13 14 23 24 29 30 31 32}
    blockDim = [0.57,0.57,0.57*2];  % x, y, z ; two blocks on top of each other.
    corner = [0.105 0.08 d];
    blockInd=[blockInd,length(Elements)+1:length(Elements)+6];
    blockElement = getBlockPanels(corner, blockDim, aMDF, tauObs, 1);
    Elements = [Elements,blockElement];
    SBlock = SBlock + (blockDim(2)*blockDim(3)*2 + blockDim(1)*blockDim(3)*2); %Total surface area of blocks
    VBlock = VBlock + blockDim(1)*blockDim(2)*blockDim(3);
  end
  % Block II
  switch run
    case {1 2 3 4 5 6 9 10 11 12 13 14 15 16 21 22 23 24 25 26 27 28 29 30 31 32}
    blockDim = [0.57,0.57,0.57];
    corner = [.715 0.08 d];
    blockInd=[blockInd,length(Elements)+1:length(Elements)+6];
    blockElement = getBlockPanels(corner, blockDim, aMDF, tauObs, 1);
    Elements = [Elements,blockElement];
    SBlock = SBlock + (blockDim(2)*blockDim(3)*2 + blockDim(1)*blockDim(3)*2); %Total surface area of blocks
    VBlock = VBlock + blockDim(1)*blockDim(2)*blockDim(3);
  end
  % Block III
  switch run
    case {1 2 5 6 9 10 23 24 25 26 27 28 33 34 35 36}
    blockDim = [0.57,0.57,0.57*2];  % x, y, z ; two blocks on top of each other.
    corner = [1.325 0.08 d];
    blockInd=[blockInd,length(Elements)+1:length(Elements)+6];
    blockElement = getBlockPanels(corner, blockDim, aMDF, tauObs, 1);
    Elements = [Elements,blockElement];
    SBlock = SBlock + (blockDim(2)*blockDim(3)*2 + blockDim(1)*blockDim(3)*2); %Total surface area of blocks
    VBlock = VBlock + blockDim(1)*blockDim(2)*blockDim(3);
  end
  % Block IV
  switch run
    case {13 14}
    blockDim = [0.57,0.57,0.57];
    corner = [1.325 0.08 d];
    blockInd=[blockInd,length(Elements)+1:length(Elements)+6];
    blockElement = getBlockPanels(corner, blockDim, aMDF, tauObs, 1);
    Elements = [Elements,blockElement];
    SBlock = SBlock + (blockDim(2)*blockDim(3)*2 + blockDim(1)*blockDim(3)*2); %Total surface area of blocks
    VBlock = VBlock + blockDim(1)*blockDim(2)*blockDim(3);
  end
  %Block V
  switch run
    case {17 18 19 20}
    blockDim = [0.57,0.57,0.57*2];  % x, y, z ; two blocks on top of each other (middle).
    corner = [.715 0.08 d];
    blockInd=[blockInd,length(Elements)+1:length(Elements)+6];
    blockElement = getBlockPanels(corner, blockDim, aMDF, tauObs, 1);
```

```
        Elements = [Elements,blockElement];
        SBlock = SBlock + (blockDim(2)*blockDim(3)*2 + blockDim(1)*blockDim(3)*2); %Total surface area of blocks
        VBlock = VBlock + blockDim(1)*blockDim(2)*blockDim(3);
    end
    %Block XXX Used for when no obstruction is present
    switch run
        case {7 8}
        blockDim = [0.01,0.01,0.01];
        corner = [.05 0.08 d];
        blockInd=[blockInd,length(Elements)+1:length(Elements)+6];
        blockElement = getBlockPanels(corner, blockDim, aMDF, tauObs, 1);
        Elements = [Elements,blockElement];
        SBlock = SBlock + (blockDim(2)*blockDim(3)*2 + blockDim(1)*blockDim(3)*2); %Total surface area of blocks
        VBlock = VBlock + blockDim(1)*blockDim(2)*blockDim(3);
    end

% Source Positions - all in meters
    switch run
        case 1
    % 1
    Xs = [0.984 0.186 0.775];
        case 2
    % 2
    Xs = [0.8 0.567 0.775];
        case 3
    % 3
    Xs = [0.984 0.186 0.775];
        case 4
    % 4
    Xs=[1.778 0.559 0.203];
        case 5
    % 5
    Xs = [1.177 0.186 0.775];
        case 6
    % 6
    Xs = [0.8 0.562 0.775];
        case 7
    % 7
    Xs = [0.997 0.152 0.203];
        case 8
    % 8
    Xs = [0.121 0.537 0.203];
        case 9
    % 9
    Xs = [0.984 0.186 0.775];
        case 10
    % 10
    Xs = [0.8 0.567 0.775];
        case 11
    % 11
    Xs = [0.984 0.186 0.775];
        case 12
    % 12
    Xs = [0.121 0.537 .203];
        case 13
    % 13
    Xs = [0.991 0.356 0.775];
        case 14
    % 14
    Xs = [1.778 0.559 0.775];
        case 15
    % 15
    Xs = [0.984 0.186 0.775];
        case 16
    % 16
    Xs = [.121 0.537 .203];
        case 17
    % 17
    Xs = [0.305 0.305 .203];
        case 18
```

```
  %18
  Xs = [1.651 0.305 .203];
    case 19
  %19
  Xs = [0.305 0.305 .203];
    case 20
  % 20
  Xs = [1.651 0.305 .203];
    case 21
  % 21
  Xs = [.121 0.537 .203];
    case 22
  %22
  Xs = [0.984 0.186 0.775];
    case 23
  %23
  Xs = [0.984 0.186 0.775];
    case 24
  %24
  Xs = [0.8 0.567 .775];
    case 25
  %25
  Xs = [0.305 0.305 .203];
    case 26
  %26
  Xs = [0.991 0.356 0.775];
    case 27
  %27
  Xs = [0.305 0.305 .203];
    case 28
  %28
  Xs =[0.991 0.356 0.775];
    case 29
  %29
  Xs = [0.991 0.356 0.775];
    case 30
  %30
  Xs = [1.651 0.305 0.203];
    case 31
  %31
  Xs = [0.991 0.356 0.775];
    case 32
  %32
  Xs = [1.651 0.305 .203];
    case 33
  %33
  Xs = [0.305 0.305 .203];
    case 34
  %34
  Xs = [0.991 0.356 0.203];
    case 35
  %35
  Xs = [0.305 0.305 .203];
    case 36
  %36
  Xs = [0.991 0.356 0.203];
  end

return
```

getBlockPanels.m

```
function Elements = getBlockPanels(corner, Dim, alpha, tau, facing)
% Break a rectangular block into six panels.
% It is assumed that the block is parallel to xy, yz and xz planes.
%   Input:
%       Corner:  Looking directly at the box, this is the front left
```

```
%          corner. The coordinates with the smallest values.
%       Dim: The X,Y,Z dimensions of the block
%       alpha: The absorption coefficient of the block material, assuming
%          uniform absorption.
%       tau: The transmission coefficient of block material, assuming
%          uniform transmission.
%       facing: 1 if outside of block considered (obstruction), -1 if
%          inside of block is considered (enclosure).
%   Output:
%       Elements: Vector of six element structures.

%% Front Panel
i=1;
Elements(i).corners = MakeCoord([corner; corner+[Dim(1) 0 Dim(3)]]); %Front Left Bottom Corner to Front Right Top Corner
Elements(i).center = mean(Elements(i).corners,1);
Elements(i).facing = [0 -1 0]*facing;
Elements(i).alpha = alpha;
Elements(i).tau = tau;
Elements(i).SA = Dim(1)*Dim(3);
Elements(i).name = 'Front';
%% Left Panel
i=2;
Elements(i).corners = MakeCoord([corner;corner+[0 Dim(2) Dim(3)]]); %Front left bottom corner to rear left top corner
Elements(i).center = mean(Elements(i).corners,1);
Elements(i).facing = [-1 0 0]*facing;
Elements(i).alpha = alpha;
Elements(i).tau = tau;
Elements(i).SA = Dim(2)*Dim(3);
Elements(i).name = 'Left';
%% Rear Panel
i=3;
Elements(i).corners = MakeCoord([corner+[0 Dim(2) 0];corner+[Dim(1) Dim(2) Dim(3)]]); %Rear left bottom corner to rear right top corner
Elements(i).center = mean(Elements(i).corners,1);
Elements(i).facing = [0 1 0]*facing;
Elements(i).alpha = alpha;
Elements(i).tau = tau;
Elements(i).SA = Dim(1)*Dim(3);
Elements(i).name = 'Rear';
%% Right Panel
i=4;
Elements(i).corners = MakeCoord([corner+[Dim(1) 0 0];corner+[Dim(1) Dim(2) Dim(3)]]); %Front right bottom corner to rear right top corner
Elements(i).center = mean(Elements(i).corners,1);
Elements(i).facing = [1 0 0]*facing;
Elements(i).alpha = alpha;
Elements(i).tau = tau;
Elements(i).SA = Dim(2)*Dim(3);
Elements(i).name = 'Right';
%% Top Panel
i=5;
Elements(i).corners = MakeCoord([corner+[0 0 Dim(3)];corner+[Dim(1) Dim(2) Dim(3)]]); %Front left top corner to rear right top corner
Elements(i).center = mean(Elements(i).corners,1);
Elements(i).facing = [0 0 1]*facing;
Elements(i).alpha = alpha;
Elements(i).tau = tau;
Elements(i).SA = Dim(1)*Dim(2);
Elements(i).name = 'Top';
%% Bottom Panel
i=6;
Elements(i).corners = MakeCoord([corner;corner+[Dim(1) Dim(2) 0]]); %Front left bottom corner to rear left bottom corner
Elements(i).center = mean(Elements(i).corners,1);
Elements(i).facing = [0 0 -1]*facing;
Elements(i).alpha = alpha;
Elements(i).tau = tau;
Elements(i).SA = Dim(1)*Dim(2);
Elements(i).name = 'Bottom';
```

MakeCoord.m

```
function H=MakeCoord(X)
```

```
if X(1,1)==X(2,1)
   H=[X(1,1) X(1,2) X(1,3)
      X(1,1) X(2,2) X(1,3)
      X(1,1) X(2,2) X(2,3)
      X(1,1) X(1,2) X(2,3)];

elseif X(1,2)==X(2,2)
   H=[X(1,1) X(1,2) X(1,3)
     X(2,1) X(1,2) X(1,3)
     X(2,1) X(1,2) X(2,3)
     X(1,1) X(1,2) X(2,3)];

elseif X(1,3)==X(2,3)
   H=[X(1,1) X(1,2) X(1,3)
      X(2,1) X(1,2) X(1,3)
      X(2,1) X(2,2) X(1,3)
      X(1,1) X(2,2) X(1,3)];
end
end
```

SolidAngleFunction.m Pseudo-Code

- Define source position, element information, dx, and frequency

- Calculate the solid angle of the source to each element (OmegaSO)
    - getSolidAngleObstructions.m - determines if the path is obstructed
        - getSolidAngle.m - calculates the solid angle

- Calculate the solid angle of the k-th element to the j-th element (OmegaWO)
    - getA2AangObstruction.m – creates a mesh on the k-th element, and redefines the center of the mesh element as the origin. Then follows the same procedure as source to element with the center of the k-th element's mesh element as the source. Once the solid angle has been calculated, it is multiplied by the area of the mesh element, and moves on to the next mesh element.

- Calculate ReverbPercent. In Eq. C.4, the variable ReverbPercent in the code is defined as $\frac{\sum_i \Omega_{Si}(\alpha_i+\tau_i)}{4\pi}$.

- Calculate Theta. Theta is a term that has been used in some of the developments of "The Equation". It is defined as $(1-\alpha_k-\tau_k)\Omega_{kj}S_k$ – the k terms in the numerator of Eq. C.5

SolidAngleFunction.m

```
function [OmegaSO,OmegaWO,ReverbPercent,Theta]=SolidAngleFunction(Xs,Elements,dx,apInd,Freq)
      clear ('OmegaSO','OmegaSOalpha','OmegaSOtheta')

% Define zero matrices
OmegaSOtau=zeros(length(Elements),length(Elements(1,1).alpha));
OmegaSOnum=zeros(length(Elements),length(Elements(1,1).alpha));

% Calculates the solid angle of the source to the j-th element
    for j=1:length(Elements)
       OmegaSO(j)=getSolidAngleObstruction(Xs,j,Elements,dx);
    end
% Calculates the solid angle of the k-th element to the j-th element
% OmegaWO is also multiplied by the surface area of the k-th element
```

```
% OmegaWO = Omega_kj*S_k
    OmegaWO=zeros(length(Elements),length(Elements));
    for k=1:length(Elements)
       for j=1:length(Elements)
          OmegaWO(k,j)=getA2AangObstruction(Elements, k, j, dx, apInd);
       end
    end
% Calculates the percentage of reverberant power left in the enclosure
% after the first reflection
    ReverbPercent=zeros(1,length(Freq));
    for f=1:length(Freq)
      for i=1:length(Elements)
         ReverbPercent(f)=ReverbPercent(f)+(OmegaSO(i)*(Elements(i).alpha(f)+Elements(i).tau(f))/(4*pi));
      end
    end
% Calculates a part of the power that leaves the enclosure.  This term has
% been called theta in some of the developments.
% Theta=(1-alpha_k-tau_k*Omega_kj*S_k)
    Theta=zeros(length(Elements),length(Freq));
    for j=1:length(Elements)
       for f=1:length(Freq)
          for k=1:length(Elements)
             Theta(j,f)=Theta(j,f)+(1-Elements(k).alpha(f)-Elements(k).tau(f))*OmegaWO(k,j);
          end
       end
    end
```

getSolidAngleObstruction.m – Pseudo-Code

- Read in variables:
    - x0 = source location
    - targetInd = ith Element
    - Element = Elements structure
    - dx = mesh size

- eN = counter for Element size

- XX0 = 4x3 matrix with x0 values

- tol = dx/1000 for boolean operation

- Generate variables for orientation, center, and corners (center and corners are referenced to x0 position)

- Generate mesh in X,Y,&Z directions. (For single-valued direction, add a repeated value)

- Initialize ang = 0 for adding mesh element solid angles

- IF statement determines if Element faces source (negative value faces inward)

- Loop through x,y,z mesh elements

- Generate corners for ith mesh element

- Find center of mesh element

- Generate trigonometric variables for mesh element (more information available in m-file and illustrations

- Loop through every Element to determine if mesh element is obstructed

- Find center and orientation for Element

- Determine if source sees Element

- Determine if visible Element obstructs mesh element

- If no obstruction for mesh element, calculate solid angle, else return 0 for solid angle
    - Use getSolidAng.m to calculate solid angles

- Sum all mesh element solid angles and return value for solid angle of ith Element

getSolidAngleObstruction.m

```
function ang = getSolidAngleObstruction(x0, targetInd, Elements, dx)

% The function getSolidAngleObstruction recieves four inputs:
% - x0 is the center point of the source in question
% - targetInd is the ith Element in the Elements structure
% - Elements is the structure containing all of the information for each
%   element (center, corners,SA,band,etc.)
% - dx is the block size to be used to create the mesh
% The function outputs an array of the solid angle for each element that
% the source can see

eN = length(Elements);  %Creates a variable indicating the number of elements in structure Elements
XX0 = ones(4,1)*x0;     %Creates a 4x3 matrix with the X,Y,and Z coordinates of source in each respective column


tol = dx/1000;          %small offset for boolean in line 99

tFacing = Elements(targetInd).facing;       %temporary variable storing the orientation coordinates
tCenter = Elements(targetInd).center-x0;    %Redfines Element center with origin defined at x0
tCorners = Elements(targetInd).corners-XX0; %Calculates distance of corners from x0

tX = [min(tCorners(:,1)):dx:max(tCorners(:,1))];    %Creates mesh along x-direction
xN = length(tX);
if (xN == 1 || max(tX)<max(tCorners(:,1)))
   tX = [tX, max(tCorners(:,1))];
   xN = xN + 1;
end

tY = [min(tCorners(:,2)):dx:max(tCorners(:,2))];    %Creates mesh along y-direction
yN = length(tY);
if (yN == 1 || max(tY)<max(tCorners(:,2)))
   tY = [tY, max(tCorners(:,2))];
   yN = yN + 1;
end

tZ = [min(tCorners(:,3)):dx:max(tCorners(:,3))];    %Creates mesh along z-direction
zN = length(tZ);
if (zN == 1 || max(tZ)<max(tCorners(:,3)))
   tZ = [tZ, max(tCorners(:,3))];
```

```
    zN = zN +1;
end

ang = 0;    %Initializes solid angle variable
if (tCenter)*tFacing'<0     %Determines if source can see the element
    for l = 1:xN-1
        for m = 1: yN-1;
            for n = 1:zN-1
                if(tFacing(1)~=0)       %Builds corners for ith mesh element
                    tCorneri = [ tX(l),tY(m),tZ(n);
                                tX(l),tY(m+1),tZ(n);
                                tX(l),tY(m+1),tZ(n+1);
                                tX(l),tY(m), tZ(n+1)];
                elseif(tFacing(2)~=0)
                    tCorneri = [ tX(l),tY(m),tZ(n);
                                tX(l+1),tY(m),tZ(n);
                                tX(l+1),tY(m),tZ(n+1);
                                tX(l),tY(m), tZ(n+1)];
                elseif(tFacing(3)~=0)
                    tCorneri = [ tX(l),tY(m),tZ(n);
                                tX(l+1),tY(m),tZ(n);
                                tX(l+1),tY(m+1),tZ(n);
                                tX(l),tY(m+1), tZ(n)];
                end
                tCenteri = mean(tCorneri,1); %Finds center of mesh element
                r = sqrt(tCenteri*tCenteri.');          %Distance of center from origin
                rr = sqrt(tCenteri(1)^2+tCenteri(2)^2); %Distance of X&Y center from origin
                costheta = tCenteri(3)/r;               %Z center over r
                sintheta = rr/r;                        %X&Y center over r
                cosphi = tCenteri(1)/rr;                %X center over rr
                sinphi = tCenteri(2)/rr;                %Y center over rr
                blockflag = 0;
                for i = 1:eN     %Compares ith mesh element to all other elements...
                    % and determines if the ith mesh element is obstructed
                    facing = Elements(i).facing; %Reads in Element orientation
                    center = Elements(i).center-x0; %Finds center of element referenced to source location
                    isFacing = center*facing'; % only for panels normal to xy, xz or yz plane.
                    if isFacing == 0    %If source doesn't see element, breaks loop
                        if (tCenteri*facing'<0)
                            blockflag = 1;
                            break;
                        end
%                   elseif isFacing < 0
                    else
                        corners = Elements(i).corners-XX0; %Finds corners of Element referenced to source location
                        plane = center.*abs(facing);  %Zeroes all center values except facing direction
                        if abs(plane(1))>0
                            rr1 = plane(1)/cosphi;  %(Xi/X)*rr
                            r1 = rr1/sintheta;      %(Xi/X)*r
                            x1 = [plane(1), rr1*sinphi, r1*costheta];
                        elseif abs(plane(2))>0
                            rr1 = plane(2)/sinphi;  %(Yi/Y)*rr
                            r1 = rr1/sintheta;      %(Yi/Y)*r
                            x1 = [rr1*cosphi, plane(2), r1*costheta];
                        else
                            r1 = plane(3)/costheta; %(Zi/Z)*r
                            rr1 = r1*sintheta;      %(Zi/Z)*rr
                            x1 = [rr1*cosphi, rr1*sinphi, plane(3)];
                        end

                        if( r1 >= 0 && r1 < r-tol)  %Determines if any Element obstructs ith mesh element
                            upper = max(corners);
                            lower = min(corners);
                            tempflag = 1;
                            for j = 1: 3
                                if (x1(j) < lower(j) || x1(j) > upper(j))
                                    tempflag = 0;
                                end
                            end
                            if tempflag
```

```
                    blockflag = 1;
                    break;
                end
            end
        end
    end
if ~blockflag
    ang = ang + getSolidAng([0 0 0],tCorneri); %Calculates solid angle of ith mesh element...
    % and sums for total solid angle of ith Element
end
end
end
end
end

end
```

getA2AangObstruction.m Pseudo-Code

- Read in variables:
    - Elements = Elements structure
    - sInd = ith Element
    - tInd = jth aperture
    - dx = mesh size
- Check if ith Element is also an aperture
- Create variables for ith Element orientation and corners
- Generate mesh in X,Y,&Z directions. (For single-valued direction, add a repeated value)
- Initialize ang variable for solid angle summation
- Loop through x,y,z mesh elements
- Generate mesh element corners and center
- Send variables to getSolidAngleObstruction.m:
    - sCenteri = mesh element center
    - tInd = jth aperture
    - Elements = Elements structure
    - dx = mesh size
- getSolidAngleObstruction returns:
    - ang = solid angle of unobstucted mesh element to aperture
- multiply ang by surface area of mesh element and sum all solid angles
- Return summation of ang to main program

getA2AangObstruction.m

```
function ang = getA2AangObstruction(Elements, sInd, tInd, dx,apInd)
%Function getA2AangObstruction finds the solid angle of absorbers,
%obstructions, and enclosure walls that can see an aperture. May also be
%useful for finding solid angles of planar sources.
%The inputs are: Elements structure, ith Element, Aperture index, and dx
%The output is the solid angle for each ith Element

if sInd == tInd     %Determines if the ith Element is same as jth Element
   ang = 0;
   return
end
for i=1:length(apInd)
if sInd==apInd(i)
   ang=0;
   return
end
end

sFacing = Elements(sInd).facing;
sCorners = Elements(sInd).corners;

tX = [min(sCorners(:,1)):dx:max(sCorners(:,1))]; %Generates mesh in X-direction
xN = length(tX);
if (xN == 1 || max(tX)<max(sCorners(:,1)))
   tX = [tX, max(sCorners(:,1))];
   xN = xN + 1;
end

tY = [min(sCorners(:,2)):dx:max(sCorners(:,2))]; %Generates mesh in Y-direction
yN = length(tY);
if (yN == 1 || max(tY)<max(sCorners(:,2)))
   tY = [tY, max(sCorners(:,2))];
   yN = yN + 1;
end

tZ = [min(sCorners(:,3)):dx:max(sCorners(:,3))]; %Generates mesh in Z-direction
zN = length(tZ);
if (zN == 1 || max(tZ)<max(sCorners(:,3)))
   tZ = [tZ, max(sCorners(:,3))];
   zN = zN +1;
end

ang = 0;    %Initializes solid angle variable

for l = 1:xN-1
   for m = 1: yN-1;
      for n = 1:zN-1
         if(sFacing(1)~=0)
            sCorneri = [ tX(l),tY(m),tZ(n);
                         tX(l),tY(m+1),tZ(n);
                         tX(l),tY(m+1),tZ(n+1);
                         tX(l),tY(m), tZ(n+1)];
         elseif(sFacing(2)~=0)
            sCorneri = [ tX(l),tY(m),tZ(n);
                         tX(l+1),tY(m),tZ(n);
                         tX(l+1),tY(m),tZ(n+1);
                         tX(l),tY(m), tZ(n+1)];
         elseif(sFacing(3)~=0)
            sCorneri = [ tX(l),tY(m),tZ(n);
                         tX(l+1),tY(m),tZ(n);
                         tX(l+1),tY(m+1),tZ(n);
                         tX(l),tY(m+1), tZ(n)];
         end
            sCenteri = mean(sCorneri,1);
            ang = ang + getSolidAngleObstruction(sCenteri, tInd, Elements, dx)*dx^2;
      end
   end
end
end
```

getSolidAngle.m – Pseudo-Code

This is the function that performs the actual solid angle calculations. The function redefines the element into the x-z plane.

- Read in variables
  - Center = location of source
  - Corners = 4x3 matrix of corners of viewed mesh element

- Call function rotaCoord
  - Generate x1, x2, y1, z1, z2

- Find solid angle and return value

getSolidAngle.m

```
function solidAng = getSolidAng(center,corners)
% Find the solid angle from a point at center to a rectangular whose corners are defined by X, Y and Z.
% center: coordinates of the point, (x,y,z);
% X,Y,Z: the boundaries of the rectangular; corners: (x_1,y_1,z_1;x_2,y_2,z_2;...).

[x1,x2,y,z1,z2] = rotaCoord(center, corners);
solidAng = abs((asin((x1*z1)/(sqrt(x1^2+y^2)*sqrt(y^2+z1^2)))-asin((x2*z1)/(sqrt(x2^2+y^2)*sqrt(y^2+z1^2))))...
   -(asin((x1*z2)/(sqrt(x1^2+y^2)*sqrt(y^2+z2^2)))-asin((x2*z2)/(sqrt(x2^2+y^2)*sqrt(y^2+z2^2)))));
return

function [x1,x2,y1,z1,z2] = rotaCoord(center, corners)
% put center at the origin, rotate the rectangular to a x-z plane.
% Right now only working for rectangulars originally in xy yz,xz planes.
ind1 = 1:4;
ind2 = [4,1:3];
a = sum((corners(ind1,:) - corners(ind2,:)).^2,1);
ind = find(a == 0);
y1 = mean(corners(:,ind)) - center(ind);
ind1 = mod(ind+1,3);
if ind1 == 0
   ind1 = 3;
   ind2 = 1;
else
   ind2 = ind1+1;
end

if corners(1,ind1)== corners(2,ind1)
   b = ind1;
   ind1 = ind2;
   ind2 = b;
end
x1 = corners(1,ind1)-center(ind1);
x2 = corners(2,ind1)-center(ind1);
z1 = corners(1,ind2)-center(ind2);
z2 = corners(4,ind2)-center(ind2);
return
```