An Empirical and Computational Investigation into the Acoustical Environment at the Launch of a Space Vehicle

by

Wesley Oliver Smith III

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
December 14, 2013

Keywords: Eldred, Sound Directivity, Sound Diffraction, Detached Eddy Simulation, Ffowcs Williams-Hawkings, and Statistical Energy Analysis

Approved by

Malcolm J. Crocker, Chair, Emeritus Professor of Mechanical Engineering
Andrew B. Shelton, Co-chair, Assistant Professor of Aerospace Engineering
Jay M. Khodadadi, Professor of Mechanical Engineering
Roy J. Hartfield, Professor of Aerospace Engineering

Abstract

During the launch of a space vehicle, the intense sound pressure levels created around the vehicle can excite the vehicle itself, payload, launch tower, and ground support systems into hazardous and potentially damaging vibration. The hostile acoustical environment can represent a severe dynamic load caused by the random pressure fluctuations generated by the turbulent rocket exhaust. This research has enhanced the existing noise prediction models for rocket exhaust during the initial launch phase of rocket powered space vehicles. The endeavor had three major components: 1) extend an empirically-based model to account for different sound directivities and surface diffractions, 2) connect conventional computational fluid dynamics models to analytical acoustic models to provide an improvement in the empirically-based model, and 3) compare sound transmission loss predictions based on statistical energy analysis with experimentally-determined sound transmission loss. This research has led to substantial improvements in methods to predict the sound generated by the space vehicle exhausts and has enhanced the understanding of the aero-acoustic rocket launch pad environments.

Acknowledgments

> But Jesus said unto him, "Follow me; and let the dead bury their dead." And when he was entered into a ship, his disciples followed him. And, behold, there arose a great tempest in the sea, insomuch that the ship was covered with the waves: but he was asleep. And his disciples came to him, and awoke him, saying, "Lord, save us: we perish." And he saith unto them, "Why are ye fearful, O ye of little faith?" Then he arose … (KJV Mat 8:22-26)

I am standing! I am standing not by my own strength for I wanted to stay curled up in the arms of my savior Jesus Christ. My life over this past year has gone from the epitome of health and prosperity to just struggling to make it through each day. In the fall semester of 2012, I had just completed my proposed research to my doctoral committee. I had fellowship funding for another year, and my research was on track. In December, 2012, my heath started to diminish. On January 9th 2013, I was diagnosed was type II diabetics, high blood pressure, and hyperthyroidism. The thyroid problem was the major issue. After several weeks, the doctors found that I had a nodule embedded in my thyroid that was cancerous. I was scheduled for surgery to remove my thyroid on March 21st, 2013. My family was planning to come for the surgery, but on March 11th, 2013, my mother passed away of a heart attack. Just over a week before my surgery, I was spreading the ashes of my mother. My family was unable to come for the surgery. I completed the

surgery and radiation therapy with my wife at my side. The different medications that I must take for my conditions conflict against each other. Between the medical issues and medications, my ability to complete my research had been severely hampered. It is now almost September, 2013. My fellowship funding is gone, and my dissertation is yet to be finished. Additional funding was very graciously offered to allow me to complete this dissertation.

I am so very thankful that my Lord had allowed me the opportunity to stand and complete the work He has set for me. I want to thank my wife, Cheryl! She has been the reason I persevere each day. She has helped me in every aspect of my life, and I would not be where I am without her. My mother and father were so instrumental in me starting my academic career, so thank you. I would also like to thank Dr. M. Crocker, Dr. A. Shelton, Alabama Space Grant Consortium (NASA Training Grant #NNX10AJ80H), and Dr. G. Flowers for their generous financial and advisory support. Also, I would like to thank the rest of my doctorial committee, Dr. J. Khodadadi, Dr. R. Hartfield, Dr. B Foster for their advisory support. Lastly, I would like to thank all the professors that I have had through these many years of classes. A few of them were very instrumental in me getting to this point in my academic career, so thank you Dr. G. Nail (UTM), Dr. D. Sterrett (UTM), Mr. J. Hadley (JSCC), and Mr. W. Dixon (JSCC).

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AUSM+	advection upstream splitting method
CAA	computational aero-acoustics
CFD	computational fluid dynamics
CPWL	cells per wavelength
deg	degrees
DES	detached eddy simulation
DI	directivity index
FFT	fast Fourier transform
FW-H	Ffowcs Williams – Hawkings
Hz	cycles per second
LES	large eddy simulation
Ma	Mach number
m/s	metres per second
NASA	National Aeronautics and Space Administration
NSE	Navier-Stokes equations
OASPL	overall sound pressure level
rad	radians
RANS	Reynolds averaged Navier-Stokes

Re	Reynolds number
RMS	root mean squared
SEA	statistical energy analysis
SGS	sub-grid scale
SPL	sound pressure level
SRB	solid rocket booster
SSDL	sound spectrum density level
SSME	Space Shuttle main engine
SST	shear stress transport
St	Strouhal number
SWL	sound power level
TL	transmission loss
TSPWP	time steps per wave period
TWCS	thin-walled cylindrical shells

1 Introduction

At the beginning of a launch sequence, a space vehicle's engines fire and a high pressure wave emanates from the nozzle region. This initial wave envelops the vehicle and is often referred to as the ignition overpressure wave.[1] Once the engines are completely operating, the acoustic field is primarily caused by the fluctuating turbulence in the mixing region of the exhaust flow.[2] After launch, the sound pressure driven acoustic load on the vehicle decreases as the vehicle accelerates. When the vehicle's velocity exceeds the speed of sound, the propulsion induced acoustic loading over most of the space vehicle is reduced to zero. This is a result of the sound generated by the rocket's exhaust propagating forward at a velocity less than that of the vehicle. The loads on the vehicle are a result of the dynamic instability in the supersonic turbulent boundary layers as it flows over the vehicle surface.[3] These induced aerodynamic loads produce a self-excited oscillation in the vehicle surface known as panel flutter.[4] Canabal's[5] research about ignition overpressure and Hartfield and Crocker's[6] research about turbulent fluctuations are important topics to space vehicle safety. These topics have been analyzed in detail; therefore, the primary focus of this work is on predicting the acoustic loads on the vehicle once the engines are fully operating.

During launch, the intense sound pressure levels around the space vehicle can excite the vehicle itself, payload, launch tower, and ground support systems into hazardous and

potentially damaging vibration. Large magnitude vibrations in the range of 100 Hz to 10,000 Hz (cycles per second) can result in tens or even hundreds of thousands of stress reversals and the possibility of space vehicle fatigue in the first 10 to 20 seconds of flight during lift off and initial climb-out. Launch and ground support structures are also forced to vibrate undesirably at low frequency, which has the potential to lead to catastrophic failure. In the 1970s, National Aeronautics and Space Administration (NASA) reports[7,8] revealed that 30-60% of the satellite failures that occur on the first day could be attributed to launch vibration.

The sound pressure level (SPL) and frequency content of the intense noise experienced by the space vehicle can be higher than 160 dB in the immediate vicinity of the exhaust nozzles, see Figure 1. With this intense acoustical loading, a robust launch vehicle is required which generally adds both complexity and weight to the vehicle. Thomas, Fadick, and Fram[9] have stated that up to 70% of the space vehicle structural

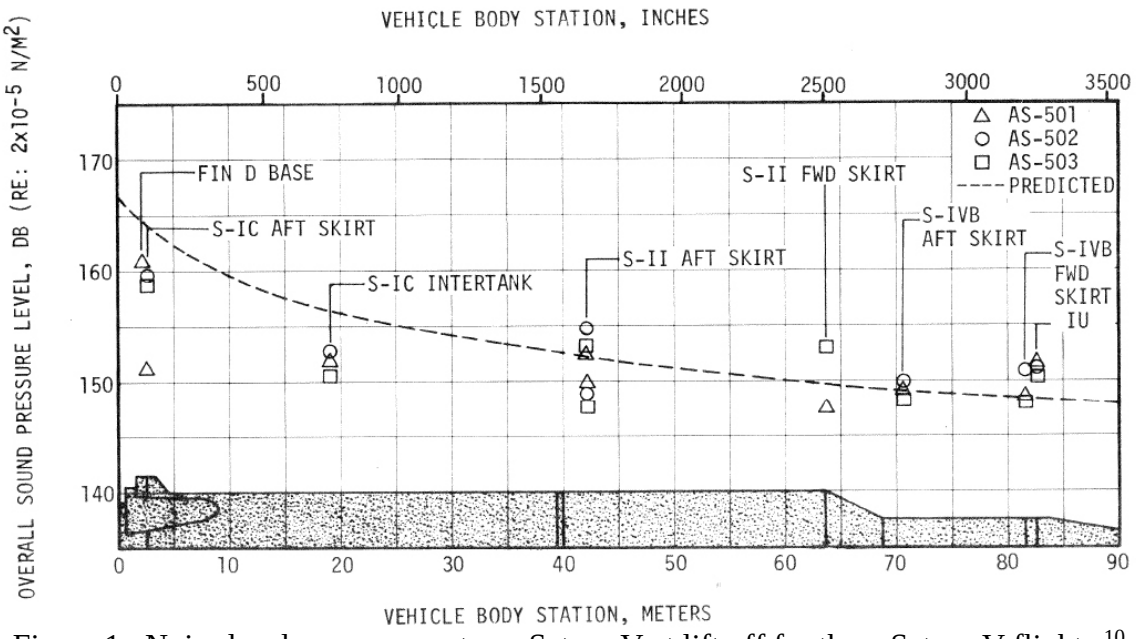

Figure 1: Noise level measurements on Saturn V at lift off for three Saturn V flights.[10]

mass is dedicated solely to ensure that the payload survives the acoustical loads at launch. A reduction in the pressure fluctuations will in turn reduce the required noise shielding thereby reducing the weight and complexity of the vehicle. Success in reducing launch noise will result in substantial improvement in vehicle performance, reduced risk within the launch environment, and reduced risk for the crew aboard manned vehicles.

Efforts to control the intense launch noise have involved the use of water injection since the inception of the US Space program. The injection of unaccelerated solid particles or liquid droplets into a jet exhaust is known, from momentum principles, to reduce the jet velocity.[11] Since the jet exhaust velocity is known to be the key driver of jet noise, the assumption has been that injecting water jets directly into the exhaust stream will substantially reduce rocket-plume-induced pressure fluctuation levels including the radiated noise. If water is injected into the rocket exhaust flow, then there are at least two major effects: First, there are changes in momentum caused by the injection of water droplets which are converted into steam and the consequent reduction in exhaust gas velocity, see Figure 2 below. Second, there is a reduction in the absolute exhaust gas temperature, which results in changes in the sound speed distribution and consequently changes in sound radiation, refraction and scattering. The reduction of the extreme aero-acoustic conditions characteristic of rocket nozzle exhausts at launch has, however, been substantially less successful than the largely empirical models predict. If extreme rates of momentum exchange and atomization are assumed, enormous reductions in rocket exhaust flow speed results and large associated drops in rocket exhaust flow noise should be expected. However, experimental evidence has shown that, in practice, reductions of only a few decibels are realized in real full scale rocket situations.[12] This is

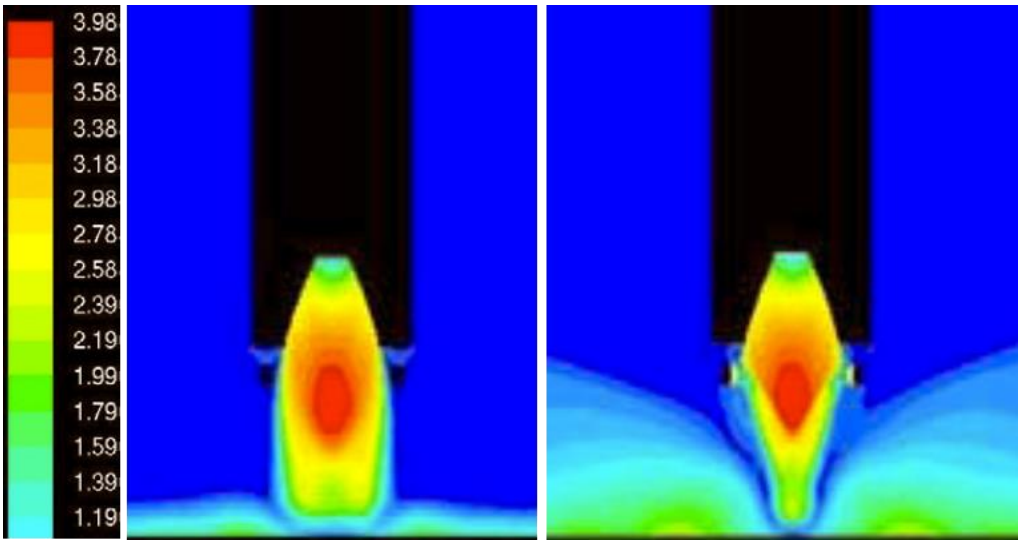

Figure 2: Rocket exhaust Mach number without water injection (left) with water (right).

due at least in part to an incomplete understanding of the relationship between the water injection placement and the sound producing flow structures. With an incomplete understanding of the physical phenomena involved in the conditions necessary to produce sound pressure fluctuations, the design of systems has focused on delivering large quantities of water over a short time frame during launch rather than on the precise effects that the water jets have on the gas dynamic environment. A better understanding is necessary to predict the changes in the sound pressure field caused by the waters injection.

A range of mostly empirical methods has been employed in an effort to develop a physical relationship between the bulk exhaust flow parameters and the jet noise. One empirical method, by Eldred,[2] is still very effective in the prediction of the acoustic loads in the launch environment. While Eldred's model is somewhat straightforward to conceptualize, it relies on empirical assumptions about the location of the noise sources within the exhaust flow. During a discussion with M. J. Crocker about noise source locations, he implied that if you know something about the sound pressure levels, you know nothing about the sound sources; however, if you know something about the sound sources,

you know everything about the sound pressure levels. Eldred's noise source locations are, of course, based on experimental data, but the reality is that there are an infinite number of possible combinations of noise sources which can result in experimental sound pressure data. Other tools should be utilized to predict the size and strength of the noise sources.

The experimental work of Krothapalli[13] and others has suggested that there is a strong correlation between the turbulence intensities and the fluctuating in the flow and the sound pressure level created in the field external to the flow. This correlation is to be expected since the source at the turbulent energy is the shearing action of the fluid. The connection between the energy input into the turbulence in the form of shearing action and the resulting pressure fluctuation levels in the sound field has already been developed.

Early work by Lighthill[14,15] demonstrated the principle that aerodynamic sound radiation is a consequence of turbulence which allows for the modeling of the sound field generation using a quadrupole source distribution. For low speed flows, a physical interpretation of this concept leads to the inhomogeneous wave equation, which Lighthill was able to solve using Green's Functions. The resulting acoustic power was then shown to be proportional to the jet velocity to the eighth power. This jet velocity raised to the eighth power is Lighthill's acclaimed acoustic scaling law for subsonic jets; however, for higher speed flows the inhomogeneous wave equation results are invalid. Kandula and Vu[16] have provided an excellent review of the scaling laws which have been developed and some of their shortcomings. Simple energy arguments can be used to show that for very high speed flows, the sound power predicted using this scaling law would become more than 100% of the propulsive power. Efforts to address this non-physical result have involved reducing the power to which the jet velocity is raised for scaling the sound power

radiation. Indeed, several theories[17,18] have been advanced to support scaling sound power radiated with jet velocity cubed in high Mach number exhaust flows. Additional empirical scaling laws to predict jet noise levels have been proposed, and the general trend is that, for noise predictions, the exponents of the jet velocity should be diminished for supersonic jet exhaust velocities.

Based on Lighthill's theory, Goldstein[19] assumed that the locally turbulent flows were homogeneous and isotropic. Similar to the Goldstein model, the Proudman[20] noise source model evaluates acoustic power per unit volume of the sound that is generated by quadrupoles normally found in supersonic turbulent flows. Both Goldstein and Proudman acoustical analogies utilize the Reynolds averaged Navier-Stokes (RANS) computational fluid dynamics (CFD) model of the rocket exhaust plume. There are a number of drawbacks to both the Goldstein and Proudman models. First, the models only make predictions inside the computational environment. Second, the overall noise level is estimated directly from the turbulent shear layers but does not allow for propagation of the noise outside the turbulence. Finally, no information is provided about the frequency content of the predicted noise. The RANS / Proudman model required a minimal amount of computational power when compared to a transient Detached Eddy Simulation (DES) coupled with a Ffowcs Williams-Hawkings[21] (FW-H) acoustical analogy.

The DES / FW-H model can be used to provide accurate acoustical noise predictions. This method has the ability to predict the sound propagated into the far field outside the computational grid. The propagation of sound into the far field is not explicitly calculated, but it is computed using an analytical integral solution to the generalized wave equation. Therefore, it is possible to use a transient form of computational fluid dynamics

to solve for turbulence intensity levels and connect them with a highly simplified analytically-based model to predict the aero-acoustic field strengths for a large class of flows.

The main consideration in the previous discussion centered on the prediction and evaluation of high speed rocket exhaust noise. The primary interest is the prediction of acoustic loads generated at the launch of a space vehicle. These loads can have a detrimental effect on the vehicle structure itself and the space vehicle structure and equipment in the interior. There is method to predict the internal noise based on the statistical energy analysis (SEA). The analytical SEA model provides an avenue to compare exterior levels with interior noise level predictions.

The remainder of the dissertation is organized as follows: Chapter 2 presents the empirical model proposed by Eldred; Chapter 3 evaluates Eldred's model; Chapter 4 details the CFD and acoustical schemes; Chapter 5 evaluates the computational models; Chapter 6 presents Wang's statistical energy analysis for the sound transmission loss of cylindrical structures; Chapter 7 evaluates use of the SEA model with cylindrical shells; and Chapter 8 provides some conclusions and observations.

2 Empirical Launch Noise Prediction Model

Chapter 2 is a discussion covering the steps required to complete the Eldred[2] empirical model and the modifications that are recommended to enhance the method. The three sections are: Section 2.1 Eldred Empirical Model Formulation, Section 2.2 Sound Source Directivity Index, and Section 2.3 Sound Wave Surface Diffraction.

2.1 Eldred Empirical Model Formulation

Section 2.1 is intended to detail the steps Eldred outlined for predicting the sound pressure level generated at the launch of a space vehicle by the rocket exhaust propulsion system. It is based upon the assumption that a minimal amount of information about the rocket exhaust is sufficient to describe a series of acoustical point sources. The point sources are then used to predict the broadband noise that the exhaust generates. The contribution of all the sources is combined to predict the sound pressure level (SPL) anywhere in the pertinent area. This Eldred method does not include methods to predict the internal noise within the vehicle nor does it explicitly account for external sound reflections. This method was originally programed into Matlab by Bechet, who was a visiting scholar at Auburn University. His original program was extensively modified to improve the computational efficiency and enhance the program features. Even though the program size was more than doubled, the overall speed of computation was reduced by

more than an order of magnitude as a result of preallocating memory and limiting hard drive access. Discussions of pertinent program changes are addressed in the appropriate steps in the description provided for the Eldred method.

Eldred produced two methods for the prediction of far-field sound pressure levels based on assumed locations of noise sources along the exhaust stream. The first method uses the technique of assigning each frequency band to a specific source location along the flow axis. The frequency band widths are arbitrarily assigned in octave, one-third octave, or narrow band within the audible range from 20 Hz to 20,000 Hz. The second method realizes that the noise in each frequency band within the audible range is generated throughout the flow, instead of at a distinct location. This second method is more difficult to apply than the first method, but it is more realistic when considering the complex nature of the rocket exhaust flow. For this reason, only the second method was considered here and will be discussed. The steps to complete the second method are as follows:

1. Calculate the flow axis relative to the vehicle, where x represents the distance along the exhaust flow, see Figure 3.

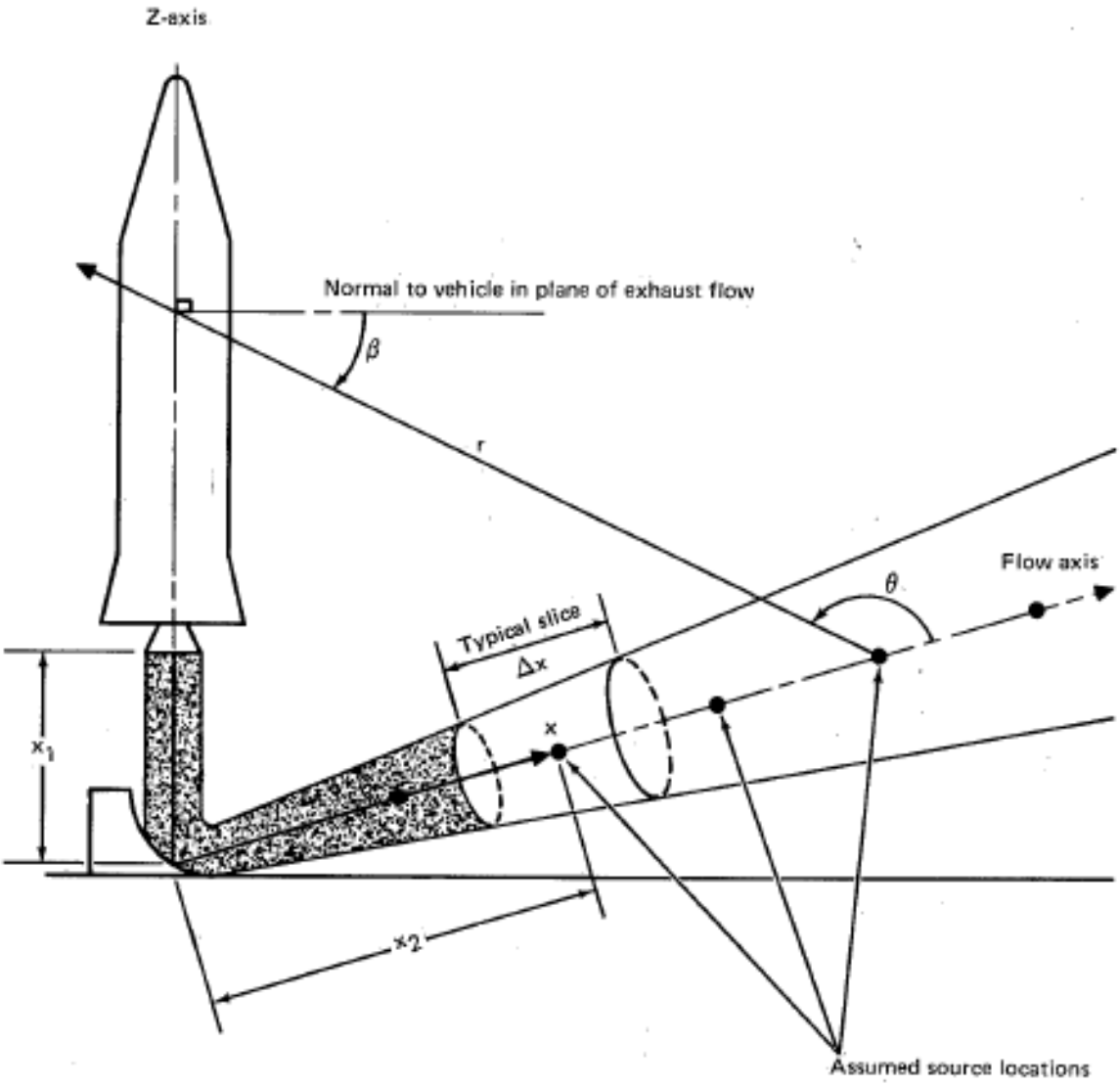

Figure 3: Source locations and geometry.[2]

2. Determine the overall sound power based on the engines' mechanical properties from

$$W_{OA} = \frac{\eta}{2} EFU_e \tag{1}$$

where W_{OA} is the overall sound power in watts (W), E is the number of nozzles, F is the thrust of each engine in newtons (N), U_e is the nozzle exit

velocity in metres per second (m/s), and η is the percentage of the overall acoustical efficiency taken from Figure 4 below.

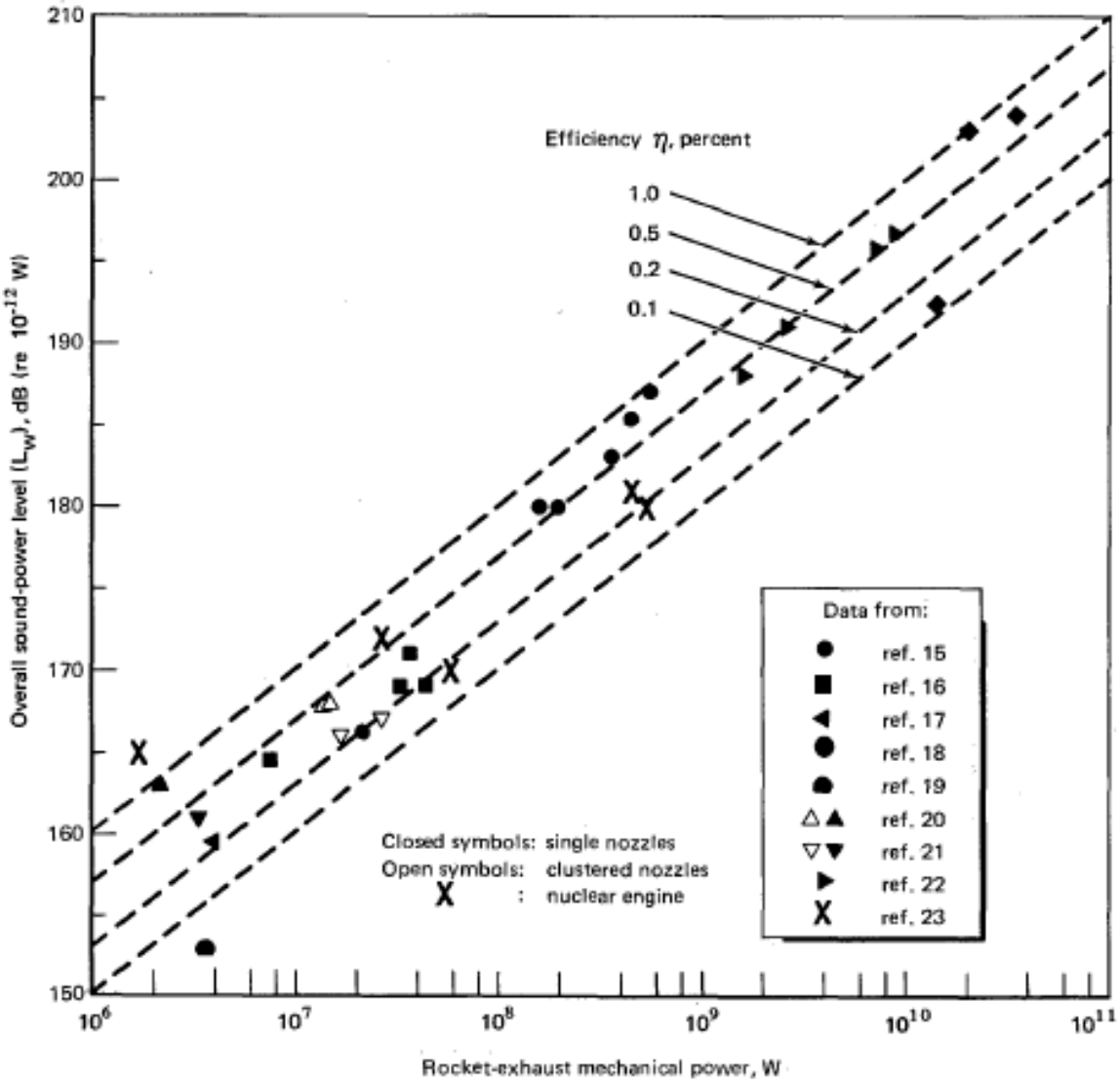

Figure 4: Overall acoustical efficiency of undeflected rockets engines.[2]

3. Compute the overall sound power level, L_W,

$$L_W = 10 log_{10}\left[\frac{W_{OA}}{10^{-12}}\right], \tag{2}$$

where the reference power is 10^{-12} W.

4. For space vehicles with more than one engine, determine the equivalent nozzle exit diameter, d_e in metres (m), using the equation below,

$$d_e = \sqrt{E} d_{ei}, \tag{3}$$

where d_{ei} (m) is the exit diameter of each nozzle.

5. Calculate the length of the core, x_t (m),

$$x_t = 3.45 d_e (1 + 0.38 M_e)^2, \tag{4}$$

where M_e is the fully expanded exit Mach number (Ma).

6. Divide the exhaust flow into the slices as shown in Figure 3 above, where x (m) is the distance along the length of the flow. Each slice is assumed to be concentrated at its center in order to form an assumed source location.

7. Obtain the normalized sound power distribution, $10 log_{10}\left[\frac{x_t W(x)}{W_{OA}}\right]$, along the length of the rocket exhaust, where the function, $W(x)$, is the sound power per axial length along the flow. Eldred's method requires that the distribution should be based upon Figure 5 shown below. Bechet used a linear curve fit to estimate the power distribution equation; however, a smoother fit to the original data is possible with a series of higher order equations. The proposed piecewise defined function is shown in Equation (5) with the corresponding comparison to Eldred's and Bechet's curves shown in Figure 6 below.

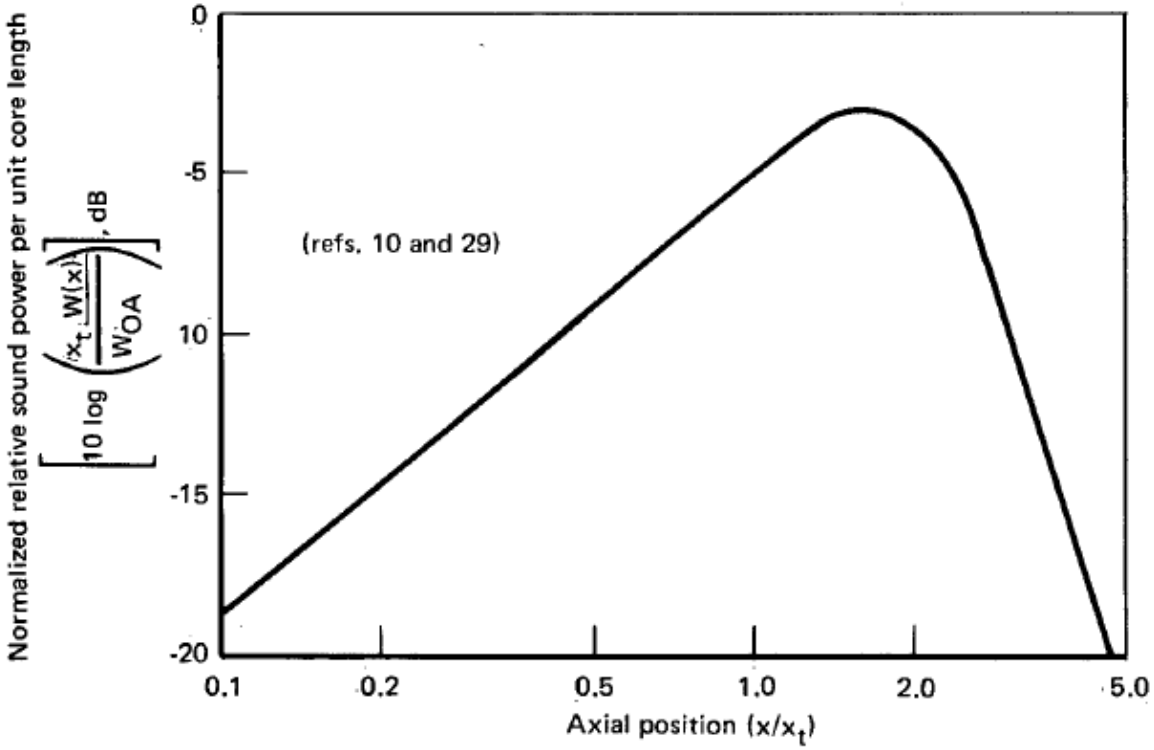

Figure 5: Source power distribution for standard chemical rockets after Eldred.[2]

$$10\log_{10}\left[\frac{x_t W(x)}{W_{OA}}\right] = \begin{cases} = 13.8\log_{10}\left[\dfrac{x}{x_t}\right] - 5, \text{ for } \dfrac{x}{x_t} \text{ less than } 1.1, \\ = -75\left(\log_{10}\left[\dfrac{x}{1.6x_t}\right]\right)^3 - 65\left(\log_{10}\left[\dfrac{x}{1.6x_t}\right]\right)^2 - 3, \\ \quad \text{for } \dfrac{x}{x_t} \text{ between } 1.1 \text{ and } 2.6, \\ = -53\log_{10}\left[\dfrac{x}{x_t}\right] - 15.5, \text{ for } \dfrac{x}{x_t} \text{ greater than } 2.6, \end{cases} \quad (5)$$

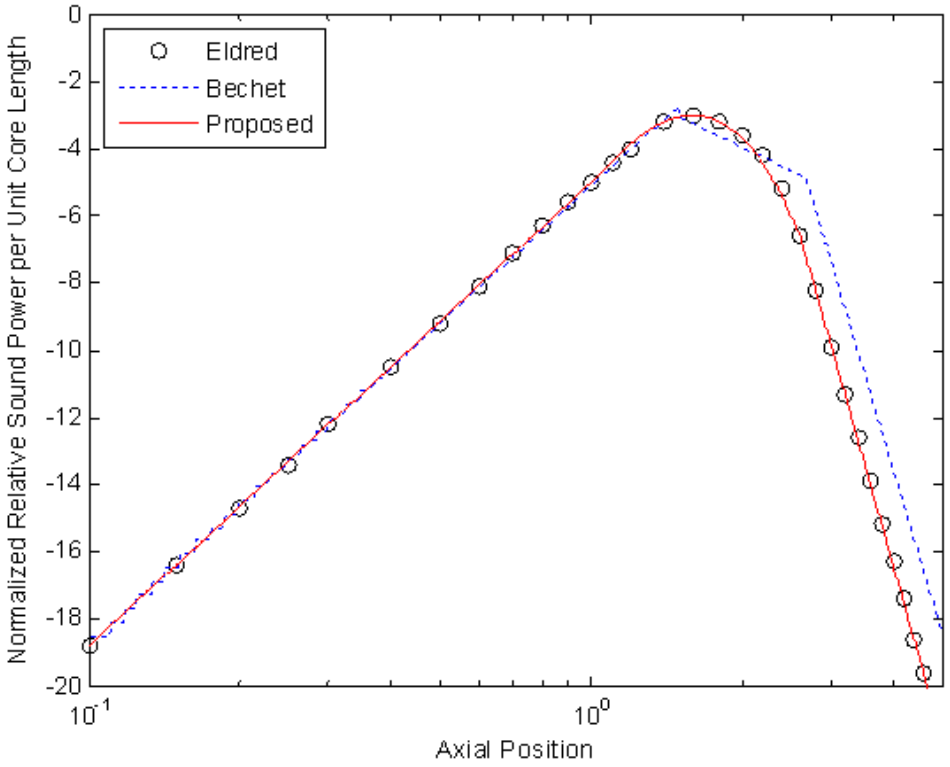

Figure 6: Proposed source power distribution for standard chemical rockets.

8. Find the sound power for each slice, $L_{w,s}$, using,

$$L_{w,s} = 10log_{10}\left[\frac{x_t W(x)}{W_{OA}}\right] + L_w + 10log_{10}\left[\frac{\Delta x}{x_t}\right], \tag{6}$$

where Δx (m) is the length of each slice.

9. Transform the normalized spectrum to a conventional bandwidth for each slice, $L_{w,s,b}$, using $10log_{10}\left[\frac{W(f,x)}{W(x)}\frac{U_e a_o}{x a_e}\right]$,

$$L_{w,s,b} = 10log_{10}\left[\frac{W(f,x)}{W(x)}\frac{U_e a_o}{x a_e}\right] + L_{w,s} - 10log_{10}\left[\frac{U_e a_o}{x a_e}\right] + 10log_{10}[\Delta f_b], \tag{7}$$

where $W(f,x)$ is the sound power per center band frequency per unit axial length along the flow (W/Hz/m), a_o (m/s) is the speed of sound in the

atmosphere, a_e (m/s) is the speed of sound based on the nozzle exit temperature, and Δf_b (Hz) is the bandwidth of each frequency band. Eldred's method requires that the spectrum distribution term, $10\log_{10}\left[\frac{W(f,x)}{W(x)}\frac{U_e a_o}{x a_e}\right]$, should be based upon Figure 7 shown below. Bechet used a linear curve fit to estimate the power spectrum distribution equation. However, a smoother fit to the original data is possible with a series of higher order equations. The proposed piecewise defined function is shown below with the corresponding comparison to Eldred's and Bechet's curves shown in Figure 8 below.

$$10\log_{10}\left[\frac{W(f,x)}{W(x)}\frac{U_e a_o}{x a_e}\right] \begin{cases} = 10\log_{10}\left[\frac{f x a_e}{U_e a_o}\right] - 7, \text{ for } \frac{f x a_e}{U_e a_o} \text{ less than } 0.8, \\ = -12\left(\log_{10}\left[\frac{f x a_e}{1.5 U_e a_o}\right]\right)^3 - 20\left(\log_{10}\left[\frac{f x a_e}{1.5 U_e a_o}\right]\right)^2 - 6.8, \\ \quad \text{for } \frac{f x a_e}{U_e a_o} \text{ between } 0.8 \text{ and } 3.5, \\ = -23\log_{10}\left[\frac{f x a_e}{U_e a_o}\right] - 2.5, \text{ for } \frac{f x a_e}{U_e a_o} \text{ greater than } 3.5, \end{cases} \quad (8)$$

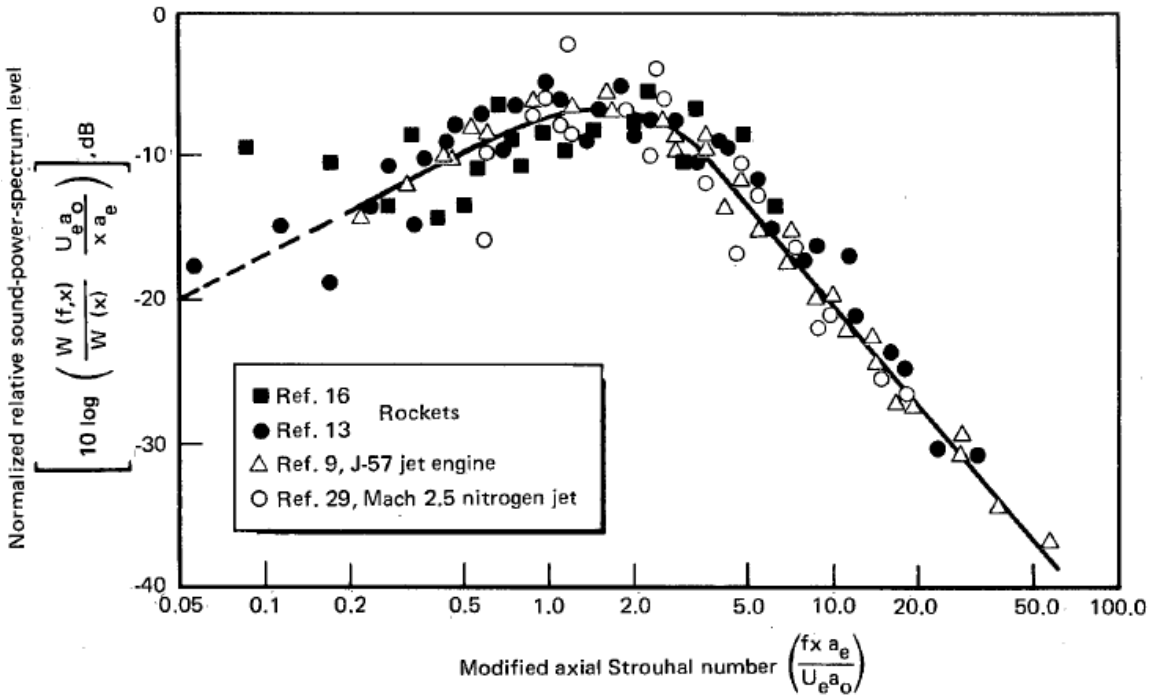

Figure 7: Eldred's normalized sound power spectrum distribution.[2]

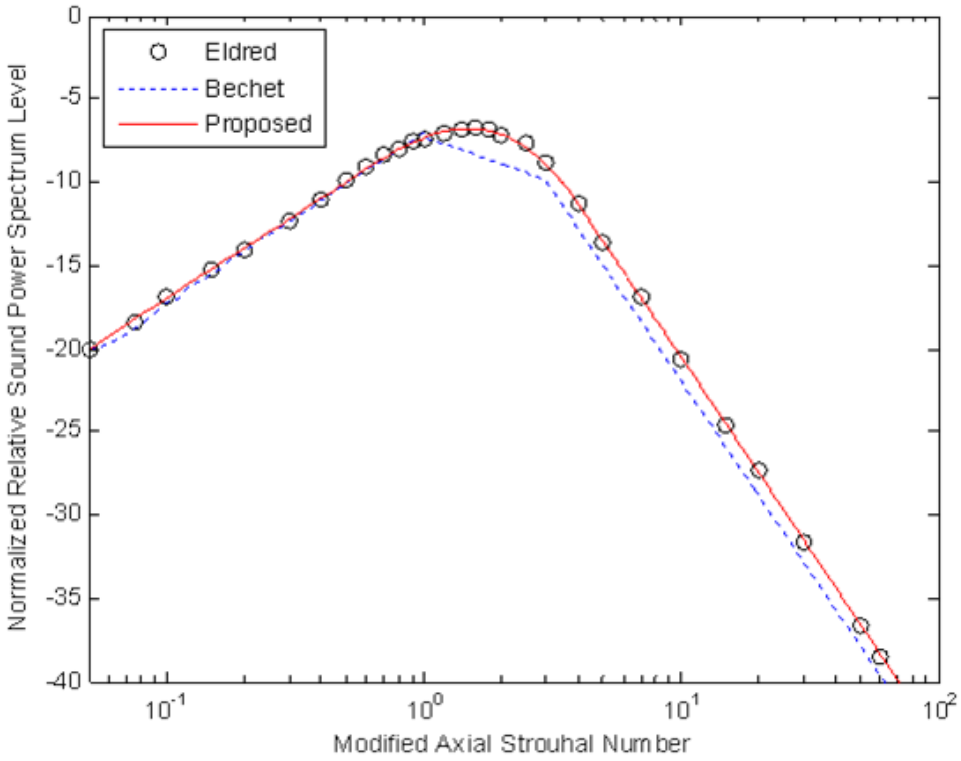

Figure 8: Proposed normalized sound power spectrum distribution.

10. Calculate the sound pressure level (SPL) in each frequency band from each assumed source within the flow, $SPL_{s,b,p}$,

$$SPL_{s,b,p} = L_{w,s,b} - 10\log_{10}\left[r^2\right] - 11 + DI(b,\theta), \tag{9}$$

where r (m) is the length from the assumed source to a point of interest on the vehicle, θ is the angle between the flow centerline and r, see Figure 3, and $DI(b,\theta)$ is the directivity index (*DI*) of the radiating sound pattern discussed in Section 2.2.

11. Find the sound pressure level in each band at each point of interest on the vehicle using the logarithmic summation of each assumed source within the flow, $SPL_{b,p}$,

$$SPL_{b,p} = 10\log_{10}\left[\sum_{s}\left(10\wedge\left(\frac{SPL_{s,b,p}}{10}\right)\right)\right] \tag{10}$$

12. Lastly, the overall sound pressure, SPL_p, is computed at each point of interest using a logarithmic summation,

$$SPL_p = 10\log_{10}\left[\sum_{All\ b}\left(10\wedge\left(\frac{SPL_{b,p}}{10}\right)\right)\right] \tag{11}$$

In practice, the application of Equation (9) requires modifications to account for several factors including deflector geometry and reflections from the ground and other structures. Experience is an important role in properly accounting for such aspects. Although Eldred's report[2] was published in 1971, it is considered to remain as the most widely used[22] empirical procedure predicting the sound pressure level generated by rocket engines during liftoff. Discussion on determining the directivity index in Equation (9) is

given in the next section. The Matlab code based on Eldred's model can be found in Appendix 1.

2.2 Sound Source Directivity Index

The directivity index (*DI*) for turbulent rocket exhaust flows is extremely launch engine specific. Figure 9 provides a diagram of the directional characteristics of four rockets and jets with regards to the maximum overall sound pressure levels involved. It can be seen that the angle of maximum radiation relative to the exhaust axis is directly related to the increase in the speed of sound of the exhaust flow. Eldred believed that this was the result of the refraction of the sound conveyed through the shear layers of the exhaust gas flow.

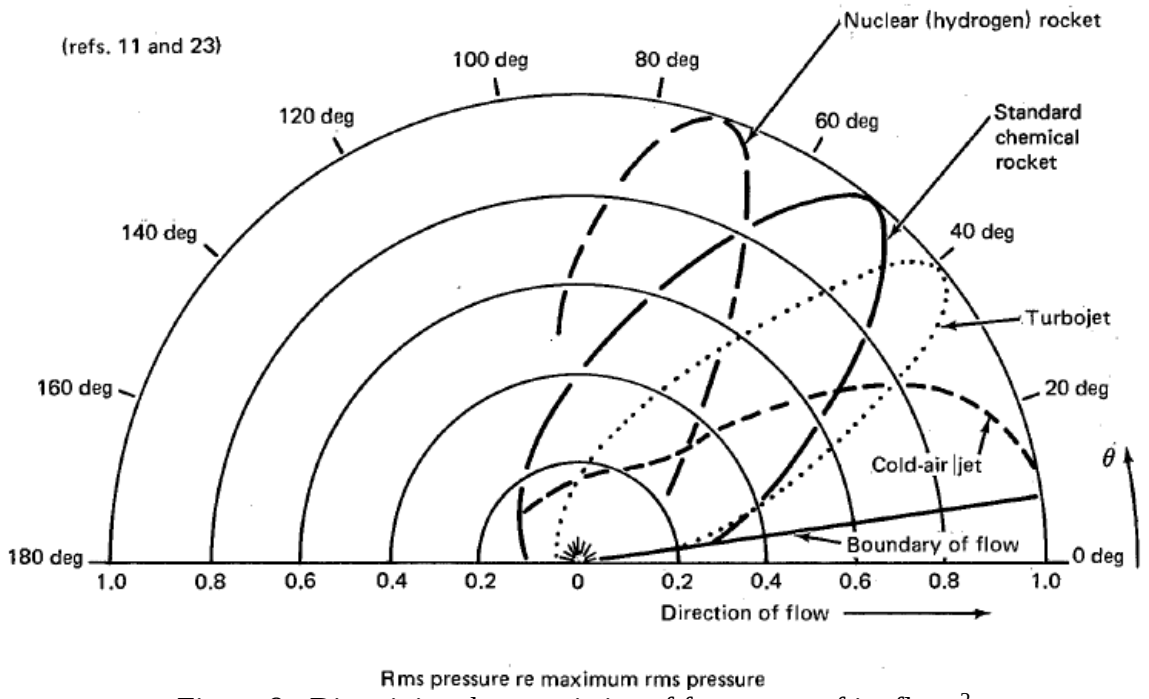

Figure 9: Directivity characteristics of four types of jet flow.[2]

Figure 10 provides the directional characteristics of sound with regards to its functions of frequency for chemical rockets. It is generally understood that the directional features for a particular Strouhal number $\left(St = \frac{f\ d_e}{U_e} \right)$ will reach maximum sound pressure level at an angle predicted from simple refraction theory[23] for high frequencies. For lower frequencies, sound reaches its maximum sound pressure level at more acute angles to the flow axis. Eldred believed that both instances were a direct result of the low frequencies not being refracted at the same angle as the higher frequencies because of differences in the sound wavelengths when compared to the width of the shear layer.

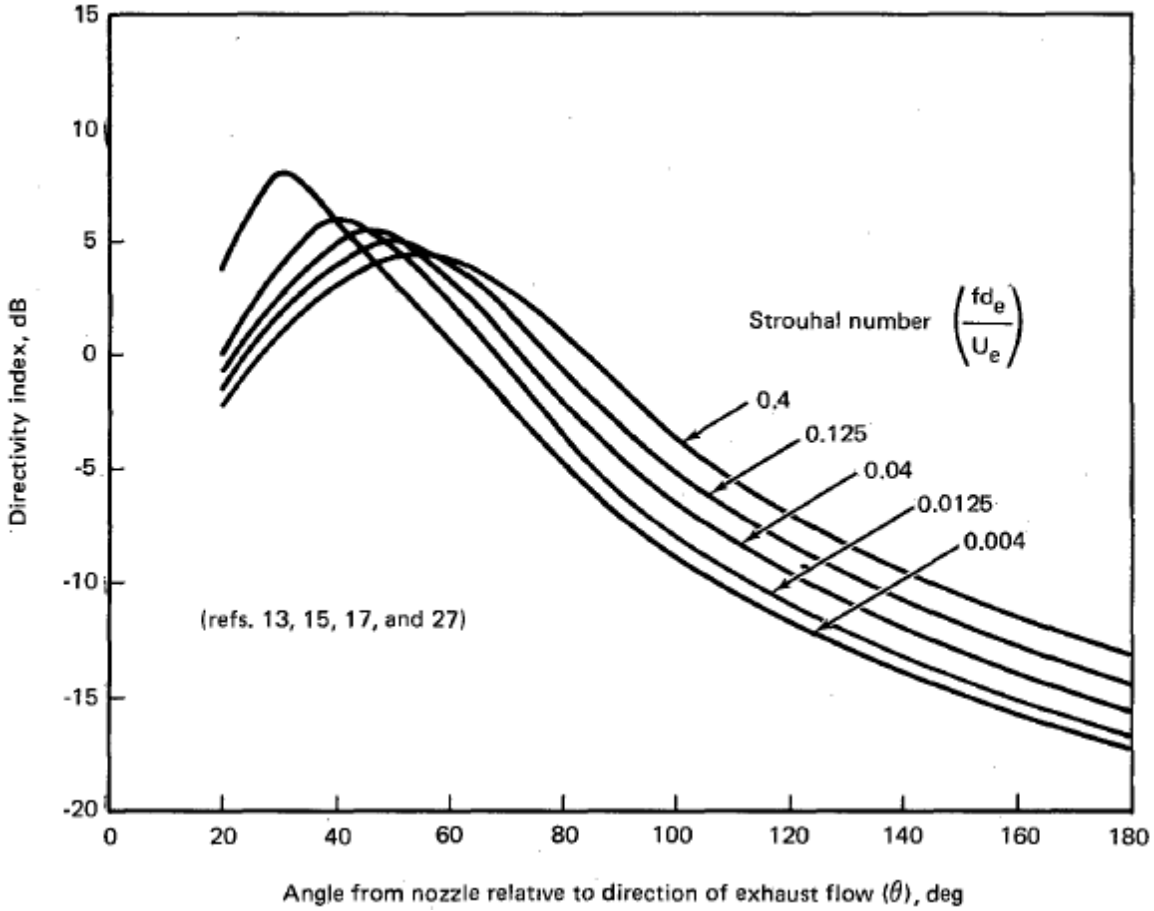

Figure 10: Directivity characteristic for standard chemical rockets after Eldred.[2]

The directivity characteristics given in Figure 10 are only plots and are not sufficient for programing into Matlab. Bechet developed an equation to estimate the original plot in Figure 10; however, a smoother fit to Bechet's original equation is possible. The proposed piecewise fit is given in Equation (12) as shown below,

$$DI(b,\theta) \begin{cases} = \left(0.4 - 0.07St^*\right)\theta - 27.5 - 3.75\,St^* + 0.56\left(St^*\right)^2, \\ \quad \text{for } \theta \text{ less than } (23 + 12\,St^*), \\ = \left(0.035\,St^* - 0.28\right)\theta - 16.3 + 0.575\,St^* - 0.35\left(St^*\right)^2, \\ \quad \text{for } \theta \text{ between } (23 + 12\,St^*) \text{ and } (90 + 12\,St^*), \\ = -0.1\theta - 0.3 - 2\,St^*, \\ \quad \text{for } \theta \text{ greater than } (90 + 12\,St^*), \end{cases} \tag{12}$$

where St^* is defined as $log_{10}\left(\frac{f\,d_e}{U_e}\right) + 3$. While Equation (12) is still piecewise linear, it is a better fit to Eldred's data. The proposed equation is compared with Eldred's and Bechet's curves as shown in Figure 11 below. The purpose of curve fitting the data was to reproduce Eldred's original work before making any modifications to the directivity plots.

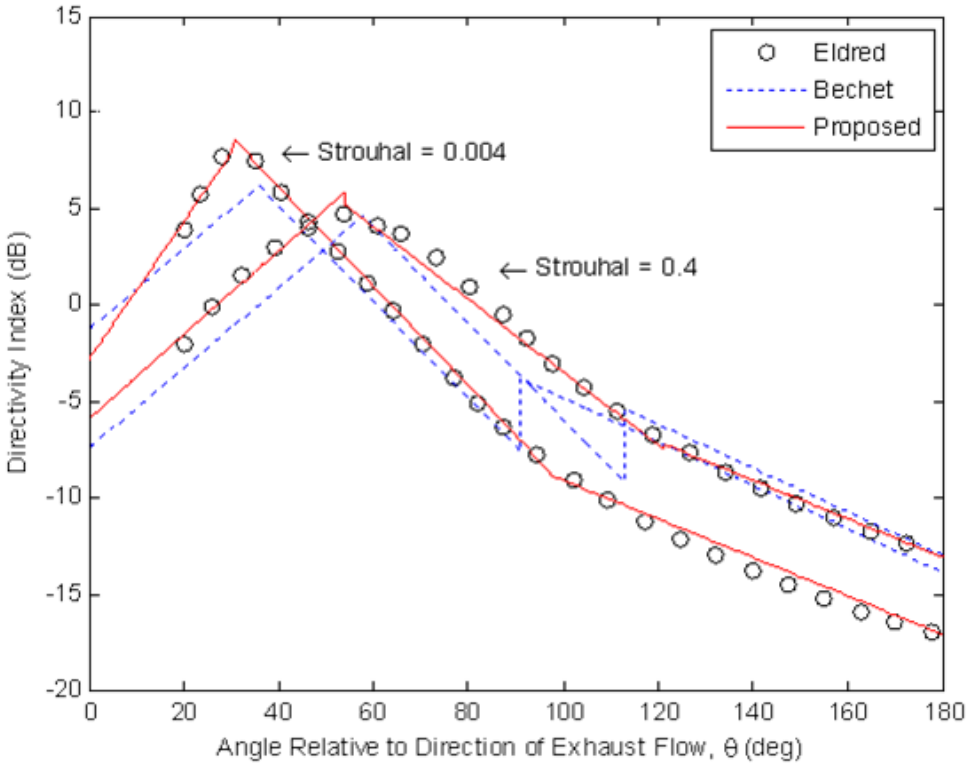

Figure 11: Proposed directivity characteristic for standard chemical rockets.

In 2005, Wilby[24] presented three new directivity equations. The first equation was developed for undeflected exhausts,

$$DI(St,\theta) = 34.61059 + 3.32662\,log_{10}[St] + 0.326131(log_{10}[St])^2 - 6.215516\left(\frac{\theta}{10}\right) + 0.3097785\left(\frac{\theta}{10}\right)^2 - 0.00596348\left(\frac{\theta}{10}\right)^3, \quad (13)$$

Equation (13) is effective for a range of angles between 90° and 180°, which is the range of angles in which Wilby was interested. He believed that the sound impinging on the launch vehicle would likely only be between these values. This first relationship was calculated based on the chemical rocket data in Figure 10 from which Eldred developed his plot; therefore, it is not surprising that Wilby's equation follows Eldred's plot within the applicable range. The comparison between Wilby's first equation and Eldred's is

shown in Figure 12 below. After reviewing Equation (13), Wilby's first equation actually followed Eldred's plot data below 90° to approximately 60° as shown in Figure 13 below.

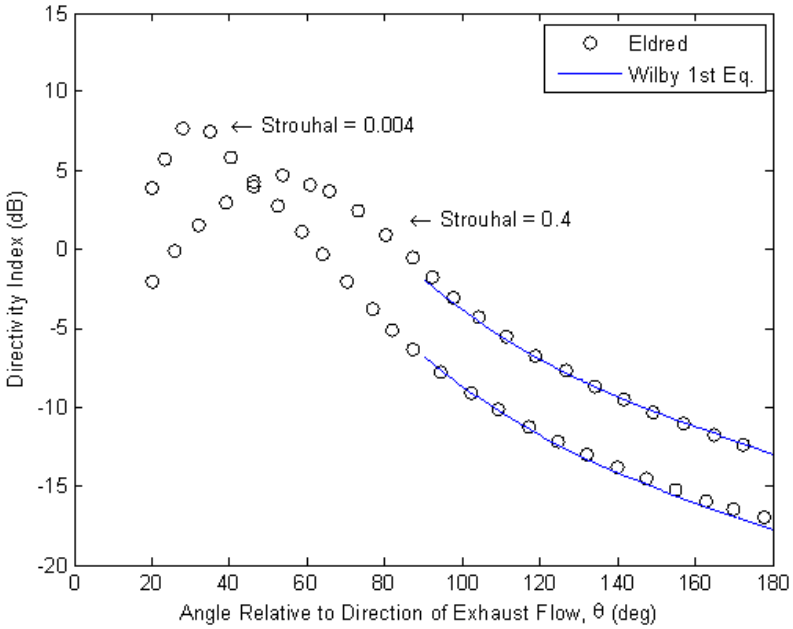

Figure 12: Comparison of Wilby's 1st directivity plot with Eldred's.

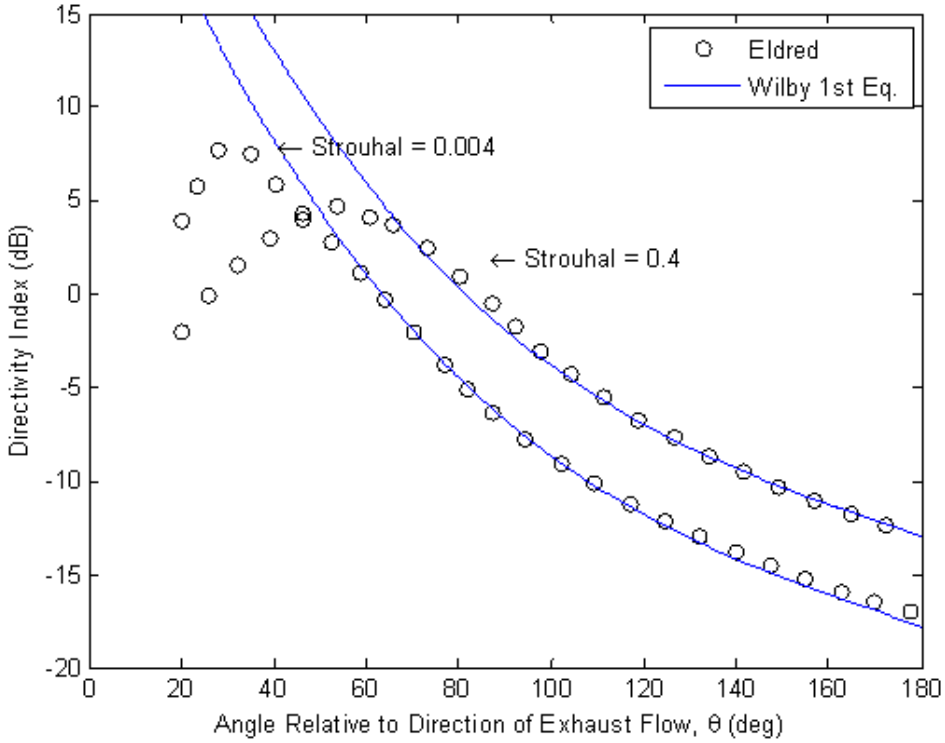

Figure 13: Extension of Wilby's 1st directivity plot compared with Eldred's.

Wilby believed that the directivity index decreased much more slowly than Eldred did as the angle θ increased. Consequently, Wilby developed a second directivity equation,

$$DI(St,\theta) = 18.89991 + 4.0511 log_{10}[St] + 0.3898576(log_{10}[St])^2 - 3.24416\left(\frac{\theta}{10}\right) - 0.1596216\left(\frac{\theta}{10}\right)^2 - 0.00305944\left(\frac{\theta}{10}\right)^3, \quad (14)$$

where again the equation is valid for θ between 90° and 180°. A comparison between Eldred's and Wilby's second equation is shown in Figure 14 below. The plot indicates the dramatic decrease in the directivity index that Wilby's studies showed.

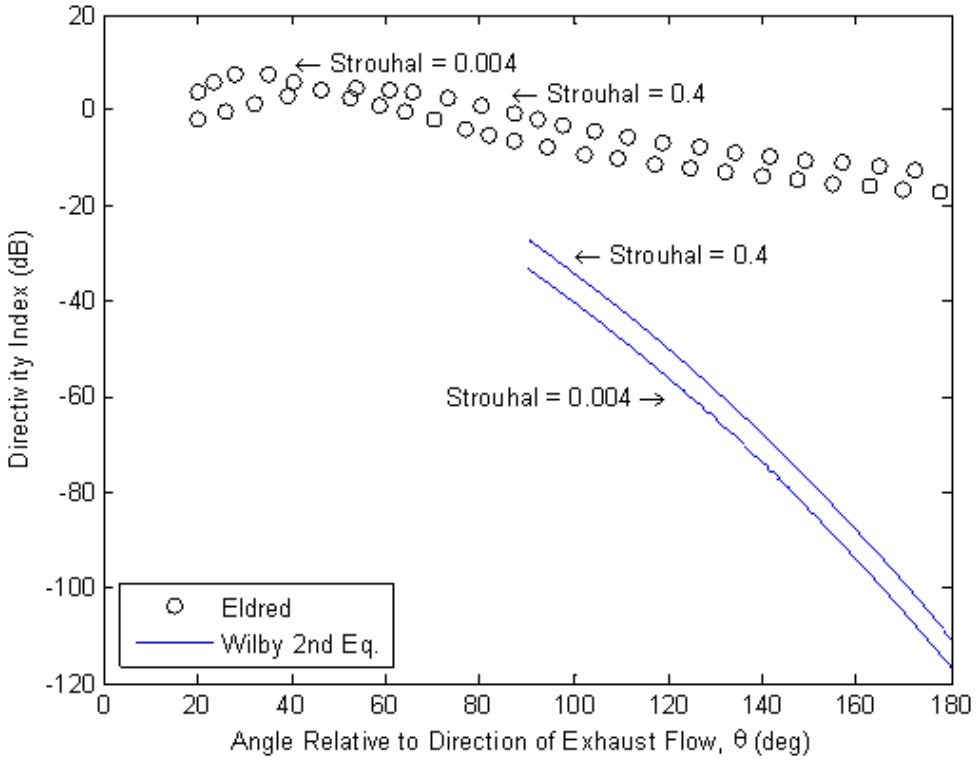

Figure 14: Comparison of Wilby's 2nd directivity with Eldred's.

Subsequent analysis identified a better fit with noise data for a deflected rocket could be achieved with an even slower fall off of directivity index as the angle θ increased. Wilby's third equation given below,

$$
\begin{aligned}
DI(St,\theta) &= 0.088828 + 7.411814\,log_{10}[St] + 1.607279(log_{10}[St])^2 + \Delta \\
\Delta &= 0,\ 90° < \theta < 170°, \\
&= 0,\ 170° \le \theta < 180°,\ St > 0.04345, \\
&= 61.7691 + 45.3515\,log_{10}[St],\ 170° \le \theta < 180°,\ 0.01 \le St \le 0.04345, \\
&= -28.9339,\ 170° \le \theta < 180°,\ St < 0.01,
\end{aligned}
\tag{15}
$$

where the value of the valid directivity index is nearly independent for angles between 90° and 180°. Although Wilby states that the third equation is "almost independent"[24] of the angle, Figure 15 shows that in reality his third directivity equation is constant across the range of angles. The dramatic drop in the directivity index for low frequencies at angles above 170° is attributable to the negative delta function for Strouhal numbers below 0.01.

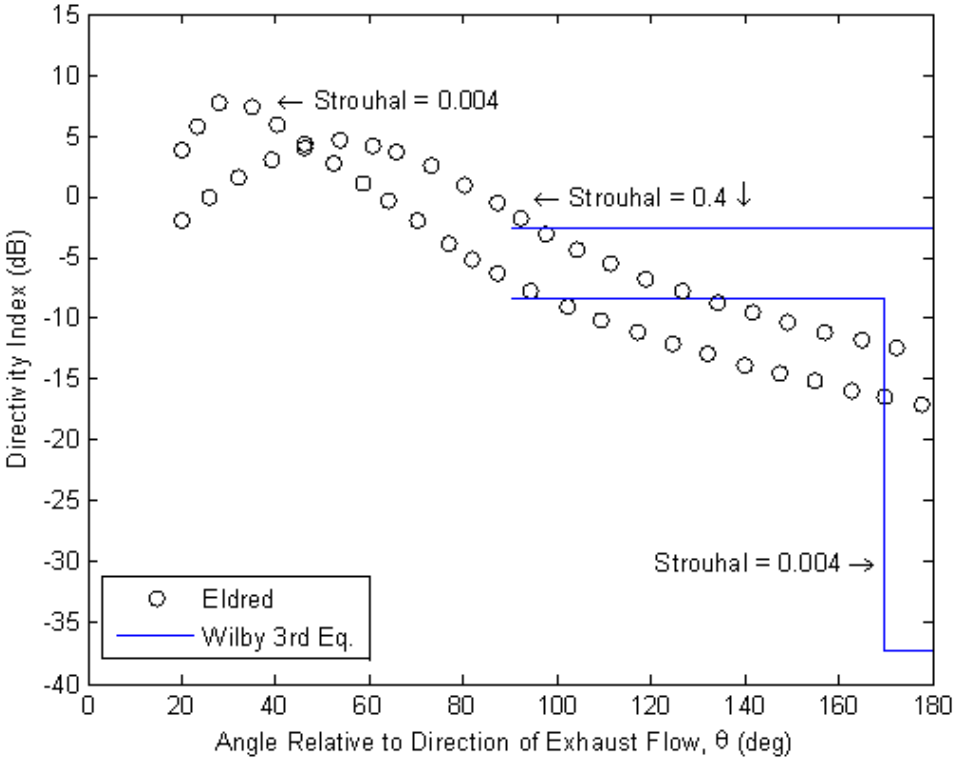

Figure 15: Comparison of Wilby's 3rd directivity with Eldred's.

Wilby's assertion that the sound impinging on the launch vehicle would only be between 90° and 180° is not completely accurate. Figure 16, below, shows the angle

between the exhaust flow direction and a prediction location 85.3 m up on the Saturn V launch vehicle. When considering the locations of the point sources along the exhaust flow axis, the angle theta, θ, changes radically at an assumed flat plate deflector location. After this point, theta starts out at approximately 65° instead of the 90°. The directivity index must account for theta angles below 90°. This researcher will assume that the Wilby equations are valid between 0° and 180°.

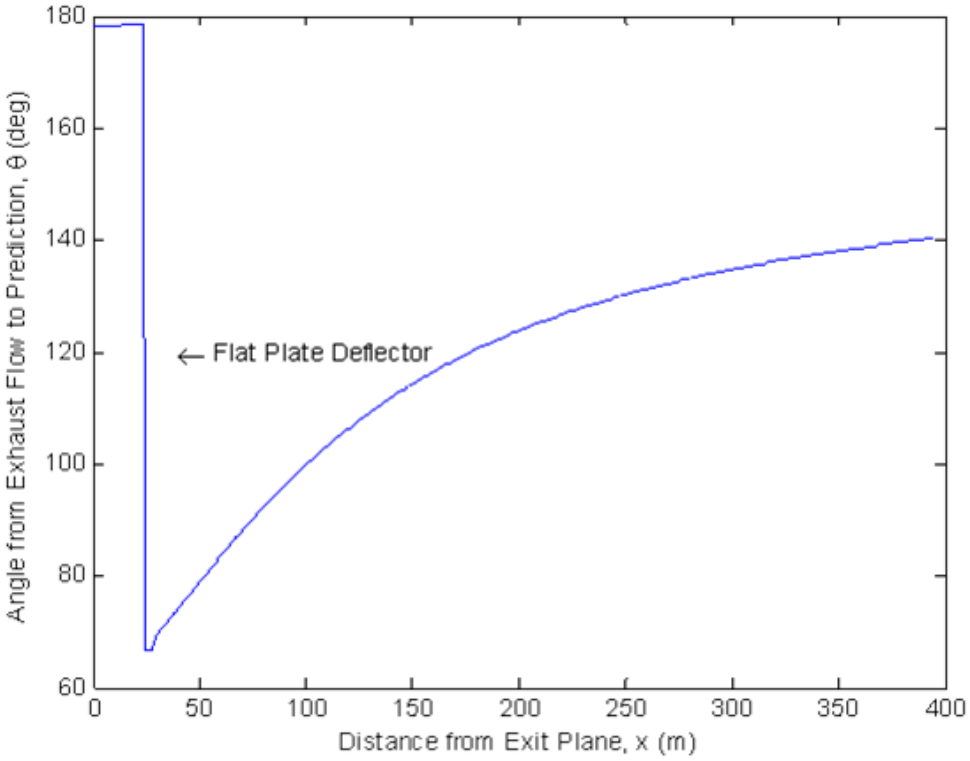

Figure 16: Values of theta, θ, for Saturn V at 85.3 m prediction location.

In 2007, Plotkin and Sutherland[25] produced a new directivity equation for rocket noise sources in undeflected flow. This new model more accurately interprets the jet engine noise directivity variation with regard to the Strouhal number. The model combines some theoretical work by Ribner[26], and experimental work by Plumblee[27], and four experiments studied by Potter discussed in the report[28] by Sutherland. The new expression

starts with a calculation of an effective Strouhal dependent angle, θ_e in radians (rad). This angle is based on the true angle, θ, relative to the rocket exhaust direction, shown in Figure 3. The effective angle, θ_e (rad), varies with the base 10 logarithm of the ratio of the Strouhal Number, St, and the maximum directivity index, DI_{max},

$$\theta_e = \theta - 9.61 log_{10}\left[\frac{St}{DI_{max}}\right], \tag{16}$$

where DI_{max} is 0.0515. Consequently, by using this effective angle, θ_e, in the Strouhal independent directivity model by Ribner and Plumblee, a Strouhal dependent equation was developed. The resulting expression is:

$$DI(St,\theta) = 10 log_{10}\left[\frac{C_1\left[1+(cos(\theta_e))^4\right]}{\left\{\left[(1-M_{ec}cos(\theta_e))^2+0.3{M_{ec}}^2\right]^{2.5}\right\}\{1+C_2 exp(-C_3\theta_e)\}}\right]+\Delta$$
$$\Delta = -C_4 log_{10}[St] - C_5, \tag{17}$$

where M_{ec} is the typical eddy convection Mach number for a heated jet (0.75) and $C_1 - C_5$ are all positive constants: $C_1 = 0.37$, $C_2 = 310$, $C_3 = 9$, $C_4 = 0.698$, and $C_5 = 1.67$. A plot of the directivity Equation (17) is shown in Figure 17 below. A comparison is made with Eldred's directivity index in Figure 18 below. The general trend of the Plotkin and Sutherland equation follows Eldred's data; however, within the predominant angles of interest (90^o to 140^o), the directivity index given by the Plotkin and Sutherland equation is lower than the directivity index suggested by Eldred.

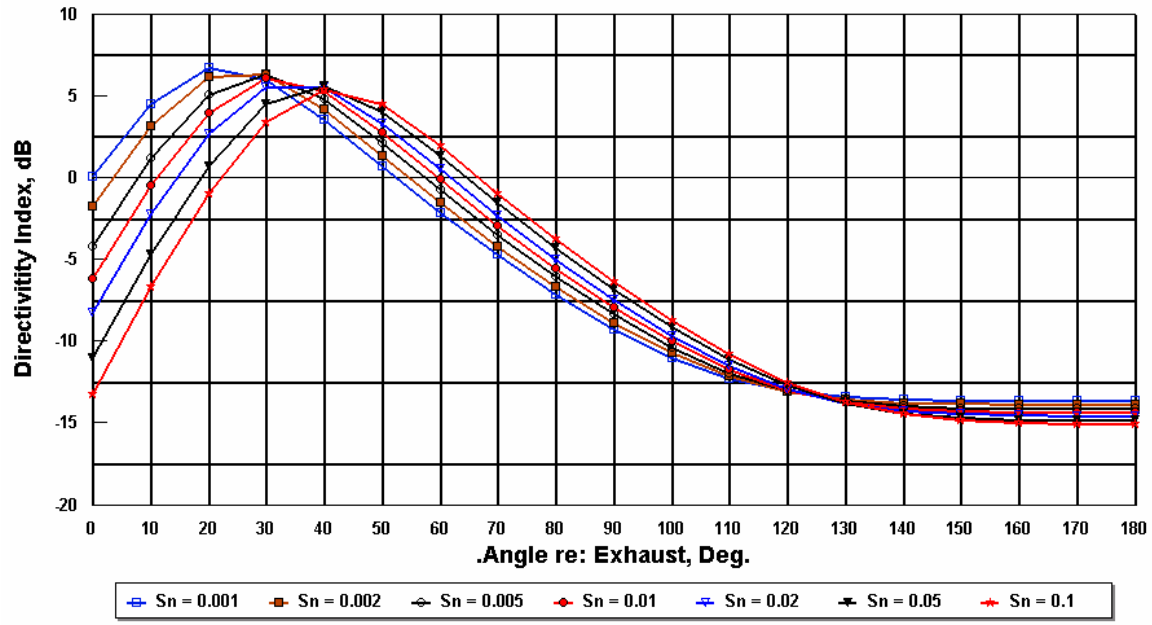

Figure 17: Plotkin and Sutherland's directivity index of rocket noise at launch.[25]

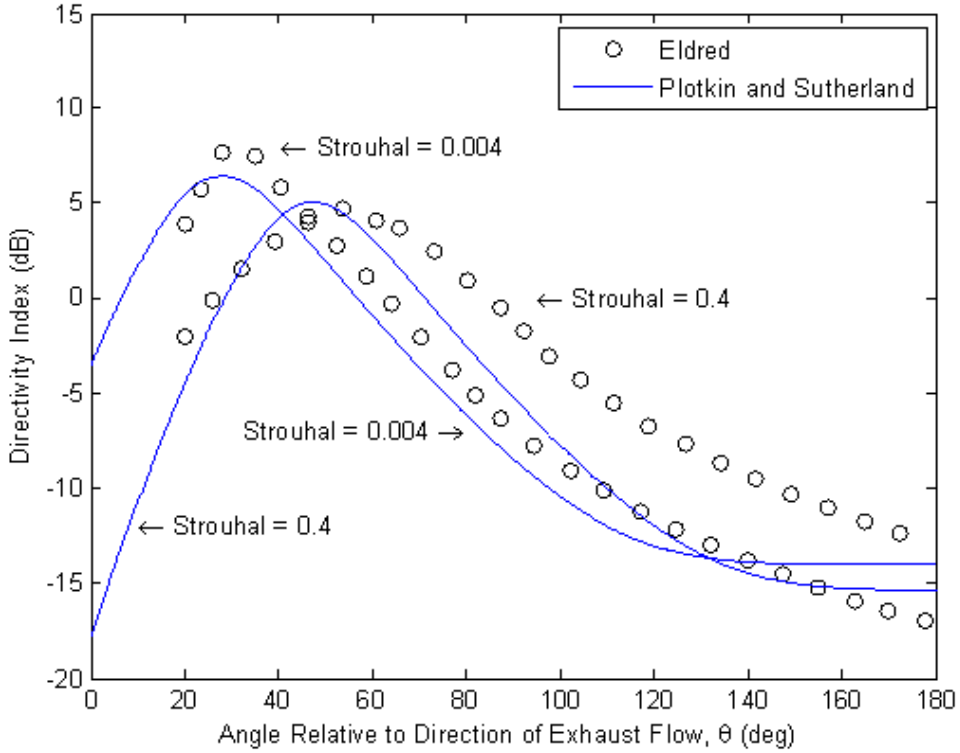

Figure 18: Comparison of directivity indexes of Plotkin and Sutherland with Eldred.

The directivity indexes presented in this section includes a curve fit of Eldred's data; Wilby's first, second, and third equations; and Plotkin and Sutherland's equation. They each modify the radiating sound patterns of each sound source. Similar to the directivity index term, $DI(b,\theta)$ in Equation (9), the sound source patterns can be modified to account for sound wave diffraction on the surface of the launch vehicle.

2.3 Sound Wave Surface Diffraction

As a sound wave strikes the surface of a space vehicle, there will be an increase in the sound pressure compared to that predicted in a free field. This acoustical effect was not included in Eldred's original model; although, he suggested that the effect could be included in an analysis. The reflection of the acoustical wave by the vehicle surface will cause a localized increase in the sound pressure level up to a maximum of 6 dB. The functional relationship for the pressure increase is highly nonlinear and is dependent upon the two angles of incidence, the frequency of the sound wave, and the diameter of the cylindrical surface it impacts. When sound waves strike a hard surface such as the external wall of a space vehicle, the sound pressure is increased at the vehicle surface due to the interaction between the incident and reflected sound waves.

In 1947, Wiener[29] formulated an analytical solution for part of the surface effect. In Figure 19 below, a plane sound wave, with wave number *k*, is striking a cylindrical surface, of radius *a*, at a combination of two angles, β and ϕ, relative to the cylinder's axis. A wave number, $k = \omega / a_o$, is the angular frequency, ω, of the wave divided by the wave speed, a_o. Wiener assumed that the plane wave struck a rigid cylinder of infinite length at an angle of 0^o for β. His analytical solution is complicated and requires several

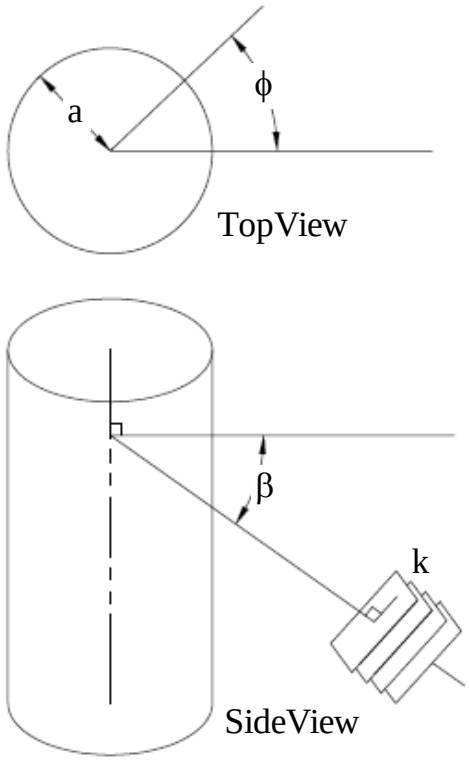

Figure 19: Plane wave approaching a cylindrical surface.

equations. When combining all the relations together, the following equation is formed,

$$\left|\frac{p}{p_o}\right| = \frac{4}{\pi ka}\sum_{m=0}^{\infty}\varepsilon_m \frac{\cos\left(m(\pi-\phi)\right)}{\frac{d}{d(ka)}\left[H_m^{(1)}(ka)\right]}, \tag{18}$$

where $\left|\frac{p}{p_o}\right|$ is the magnitude of ratio of the sound pressure, p, at the cylinder surface relative to the sound pressure, p_o, in an undisturbed wave. This ratio is the adjustment that must be added to Equation (9) to account for the surface diffraction effect. The coefficient ε_m is equal to 0.5 when m is 0, and the coefficient ε_m is equal to 1 for all other values of m. The denominator, $\frac{d}{d(ka)}\left[H_m^{(1)}(ka)\right]$, in Equation (18) is the derivative of a type one Hankel function, which is a complex sum of two Bessel functions one in a real

plane and one in an imaginary plane. There is no analytical solution to the infinite summation of this equation, and it is very slow to converge.

Although the wave front approaching the surface of the cylinder in Figure 19 is assumed to be a flat plane sound wave, it is in reality some sort of elliptical wave front similar to that shown in Figure 9. The plane wave assumption is believed to be reasonable for source distances greater than 5 to 10 cylinder radii from the cylinder surface (approximately 20 m to 30 m for a full size launch vehicle). An additional drawback to the solution that Wiener poses is that he limits the angle β to only 0^o. Figure 20 below, shows the angle between a horizontal line at a prediction location 85.3 m up the Saturn V launch vehicle and the assumed point source locations. When considering the locations of the point sources along the exhaust flow axis, the angle beta, β, changes at an assumed flat

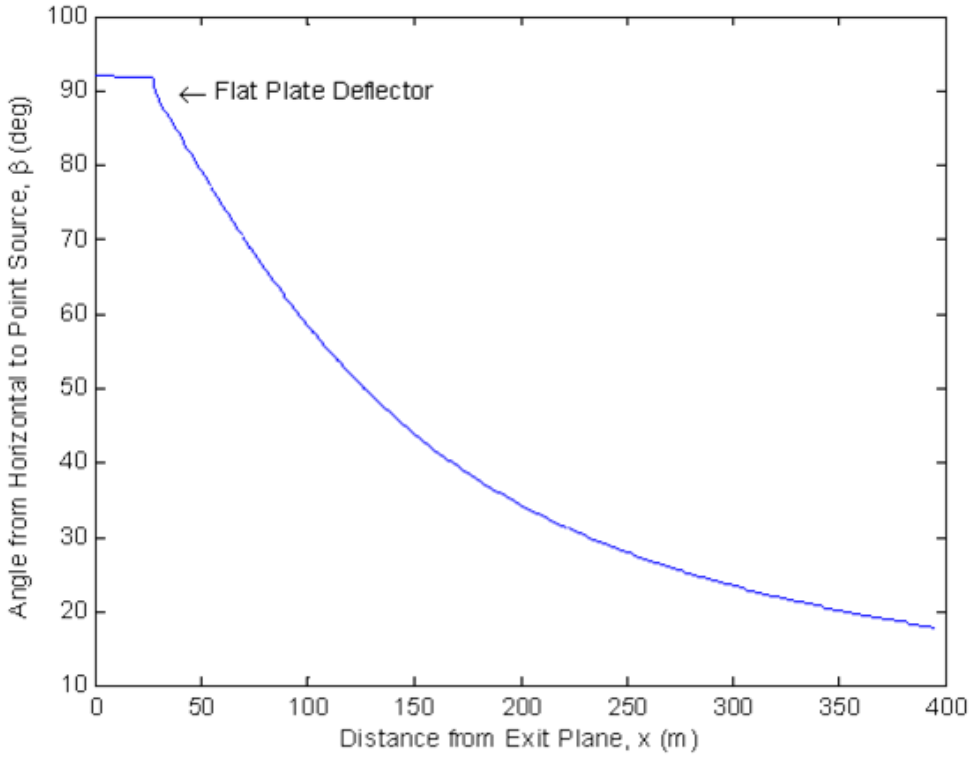

Figure 20: Values of beta, β, for Saturn V at 85.3 m prediction location.

plate deflector location. The problem is that the angle β is only near zero at the end of the assumed exhaust flow. Czyz and Gudra[30] were able to resolve the limit imposed by Wiener's equation.

In 1992, Czyz and Gudra presented an analytical solution of the approximate solution for sound wave's force on a small cylinder. Within their derivation, they formulated an equation based on the similar assumptions of Wiener except that the ka terms were modified. Equation(19) below gives the magnitude of the sound pressure ratio, $\left|\frac{p}{p_o}\right|$,

$$\left|\frac{p}{p_o}\right| = \frac{4}{\pi ka\sin(\beta)} \sum_{m=0}^{\infty} \varepsilon_m \frac{\cos(m\phi)}{\frac{d}{d(ka\sin(\beta))}\left[H_m^{(2)}(ka\sin(\beta))\right]}, \tag{19}$$

where $\frac{d}{d(ka\sin(\beta))}\left[H_m^{(2)}(ka\sin(\beta))\right]$ is a derivative of a type two Hankel function. Figure 21 below shows the results of how the pressure ratio changes at various values of ka when β is 90°. Note that the product of k and a is a non-dimensional number, which is equivalent to a ratio of the surface radius and the frequency wave length. Holding a fixed at a value of 1, Equation (19) leads to an increase in the mid to high frequency between 2 to 6 dB. The low frequency waves generally have little effect on the cylindrical surface because of the size difference between the wavelength and the cylinder diameter, i.e. for small values of ka. The long wavelength of the low frequency waves travel around the cylindrical surface as if it were not there. The necessary changes to allow for the sound diffraction effect to be incorporated into the Eldred prediction method is discussed in the next chapter.

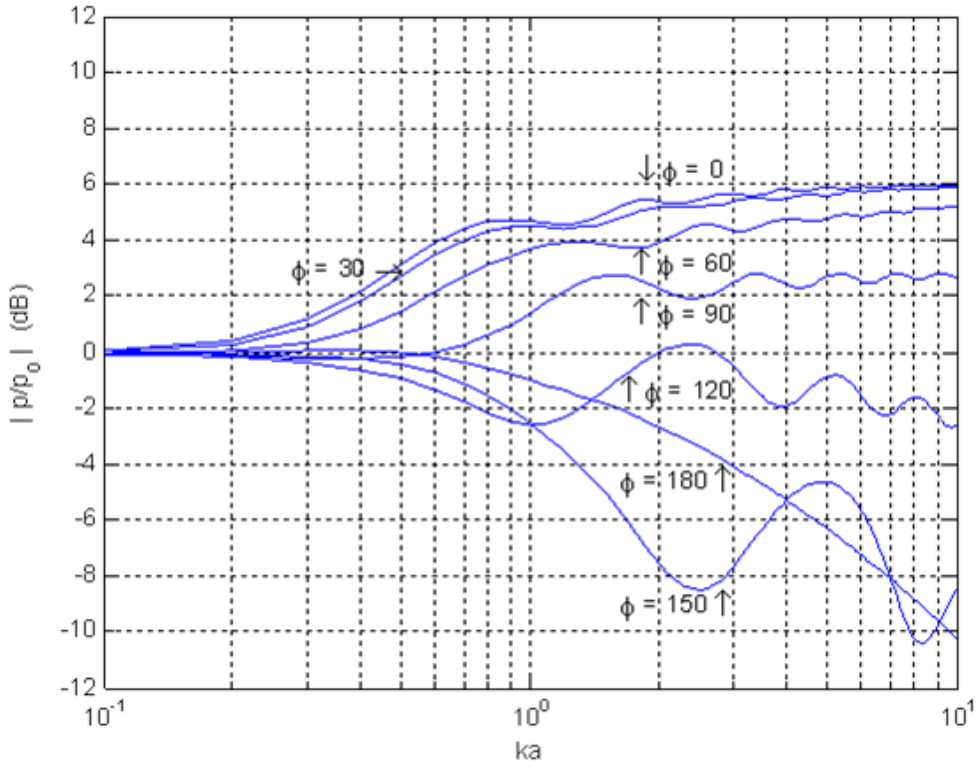

Figure 21: Sound pressure distribution ratio, $|p/p_o|$, around a rigid infinite cylinder.

3 Empirical Launch Noise Prediction Model Evaluation

Chapter 3 contains comparisons of the Eldred[2] empirical model with actual launch vehicles. The modifications that were recommended in Section 2.2 and 2.3 to enhance the Eldred method are discussed in Section 3.1. There are four sections Section 3.1 Saturn V Launch Noise Predictions, Section 3.2 Space Shuttle Launch Noise Predictions, Section 3.3 Ares I-X Launch Noise Predictions, and Section 3.4 Supersonic Jet Wind Tunnel Noise Predictions.

3.1 Saturn V Launch Noise Predictions

The Eldred empirical model requires knowledge of only a small amount of exhaust flow parameters. These parameters include the number of engines, location of each rocket engine exit plane, nozzle exit diameter, engine thrust, the exit velocity, exit temperature, and the position of the deflector ramp. Starting with the number and location of each engine, the Saturn V Aerothermodynamics Flight Evaluation Summary, compiled by Krausse[10], shows that the five engines are located at (Cartesian [X, Y, Z] m): engine 1 [-3.27, -3.27, 0], engine 2 [-3.27, 3.27, 0], engine 3 [3.27, 3.27, 0], engine 4 [3.27, -3.27, 0], and engine 5 [0, 0, 0]. The engine exit plane was assumed to be 24 m above a flat plate ramp, which will cause an apparent exhaust flow to turn from vertical to 22^{o} with the horizontal in the Y-Z plane. Because of the nature of the NASA programs, the launch pads

were reused for three launch vehicles (Saturn V, Space Shuttle, and Ares I); and the apparent 22° exhaust flow in the Y-Z plane is assumed to occur in all three cases. The remaining engine parameters can be ascertained from the F-1 Engine fact sheet[31]: the engine thrust is 6,672,333 N; the engine exit temperature is 3572°K; the engine nozzle exit diameter is 3.53 m; and the engine exit velocity is 2585 m/s.

Before considering the prediction locations, there are four remaining parameters that are used for each of the three test cases. The ambient temperature is assumed to be 300°K. The position of each point source is assumed to be 0.5 m apart. Due to the close proximity of the sources, there was no perceivable improvement in the prediction results using finer resolution. The distance along the exhaust flow must be truncated at a certain point. Using Figure 22 below, Eldred suggested that for rockets with multiple nozzles

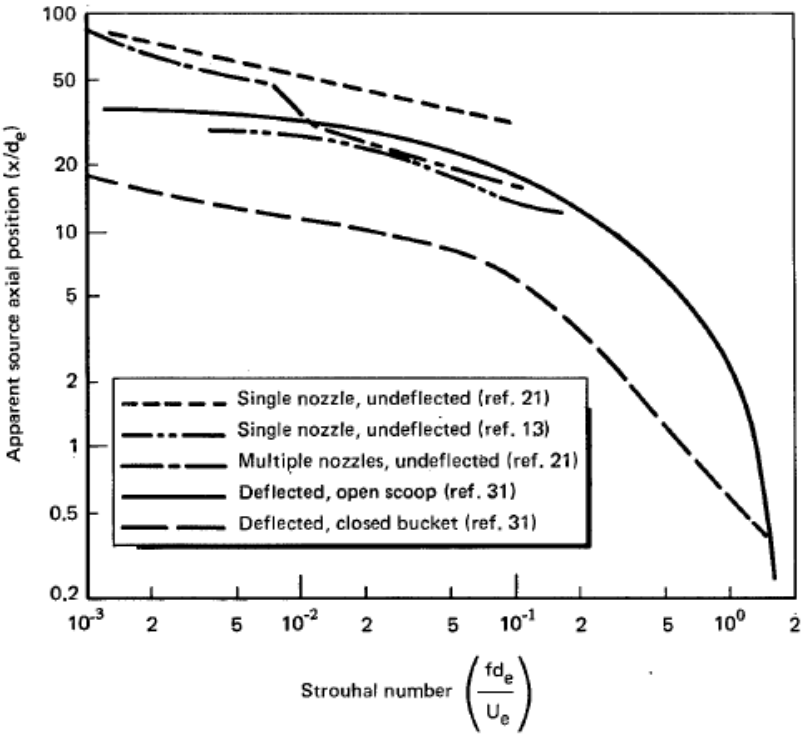

Figure 22: Axial location of apparent sources for chemical rockets.[2]

that are deflected, a normalized apparent source distance, x/d_e, of 50 is sufficient. The last parameter needed is the engine's acoustic efficiency; however, Eldred believed that the efficiency was related more with the exhaust deflector than the engine itself. In Figure 23 below, it was assumed that for all three test cases the acoustical efficiency was 0.1 %, which is for impingement onto a 45° flat plate.

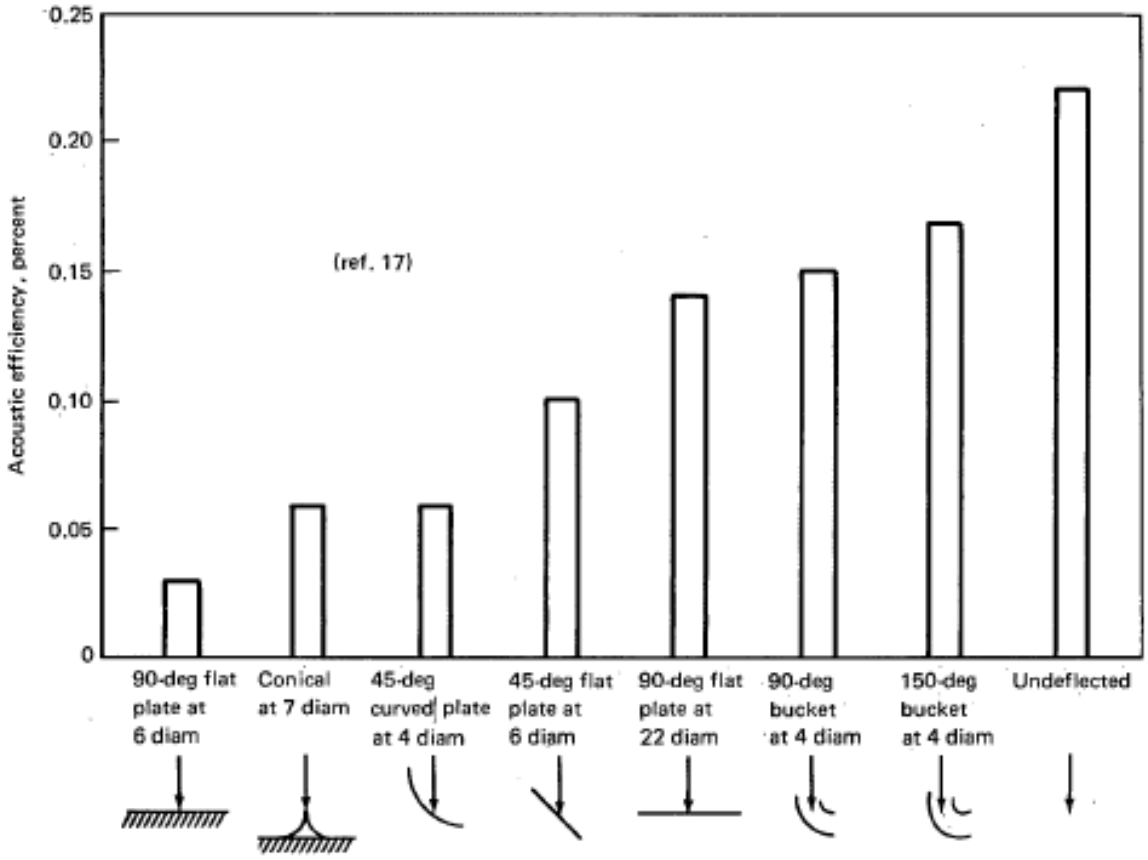

Figure 23: Acoustical efficiency for deflected chemical rockets.[2]

The prediction locations were chosen to coincide with existing launch noise data locations from a comprehensive report by Krausse.[10] All of the sensor locations for the different launches are presented in Figure 24 below. Of these locations, six were chosen

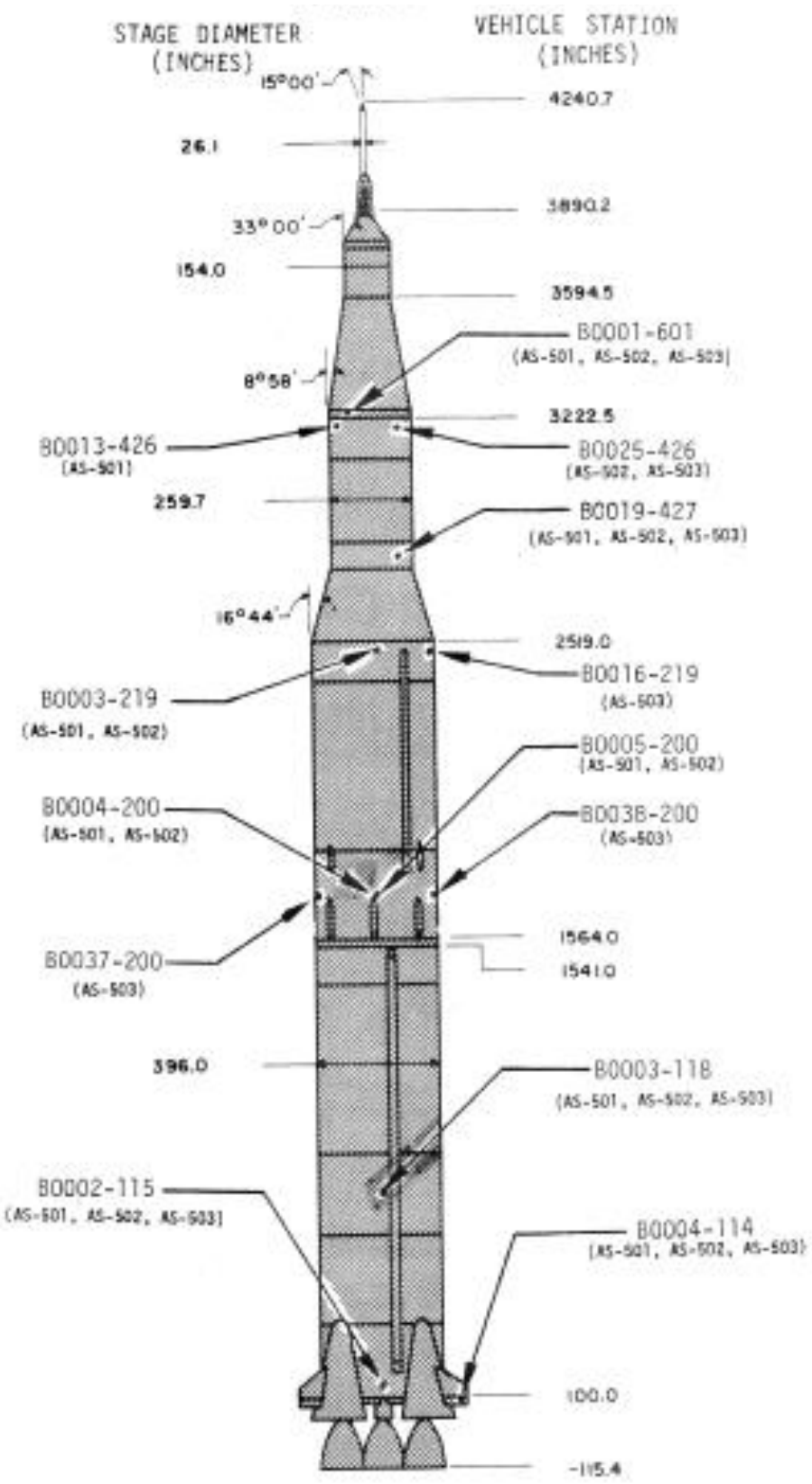

Figure 24: External microphone locations for Saturn V launches.[10]

as examples based on varying heights and orientations to the exhaust flow. Table 1 below summarizes the microphone numbers and locations. From this point forward, discussion about these microphone locations will be referred to as: prediction location 85.3 m (B0001-601), prediction location 84.5 m (B0013-426), prediction location 66.3 m (B0003-219), prediction location 45.2 m (B0005-200), prediction location 22.0 m (B0003-118), and prediction location 6.0 m (B0002-115). These prediction locations are measured from the exhaust exit plane up the launch vehicle (Z-axis).

Table 1: Saturn V chosen microphone locations for prediction comparison.

Microphone #	Launch #	Cartesian (m)		
		X	Y	Z
B0001-601	AS-501	1.3	3.0	85.3
B0013-426	AS-501	-3.2	-0.7	84.5
B0003-219	AS-501	0.0	5.0	66.3
B0005-200	AS-501	0.0	5.0	45.2
B0003-118	AS-501	0.0	-5.0	22.0
B0002-115	AS-502	0.0	-5.0	6.0

All of the launch data from the Saturn V report[10], including the data shown in Figure 25 below, are sound spectrum density level (SSDL). SSDL is computed by subtracting the frequency bandwidth (Δf) from the sound pressure level (SPL) as shown in Kinsler's[23] equation below,

$$\mathrm{SSDL} = \mathrm{SPL} - 10\log_{10}[\Delta f], \tag{20}$$

While it is theoretically possible to convert the SSDL values to SPL, the reality is, without the center frequencies or the bandwidths, accurate representation of the Saturn V SPLs are not possible based on information from the Krausse's report. For this reason, both SPL predictions, the solid red line in Figure 25, and SSDL predictions, the broken blue line in Figure 25, will be presented in this format. The Saturn V launch data, the black circles in

Figure 25, are the only one in SSDL; the launch data for both the Space Shuttle and Ares I-X are in SPL. All predictions and launch data presented in each of the figures for this chapter will follow this format. This dissertation explicitly uses standard 1/3 octave center band frequencies (Hz): 19.7, 24.8, 31.3, 39.4, 49.6, 62.5, 78.7, 99.2, 125, 157.5, 198.4, 250, 315, 396.9, 500, 630, 793.7, 1000, 1259.9, 1587.4, 2000, 2519.8, 3174.8, 4000, 5039.7, 6349.6, 8000, 10079.4, 12699.2, 16000, 20158.7.

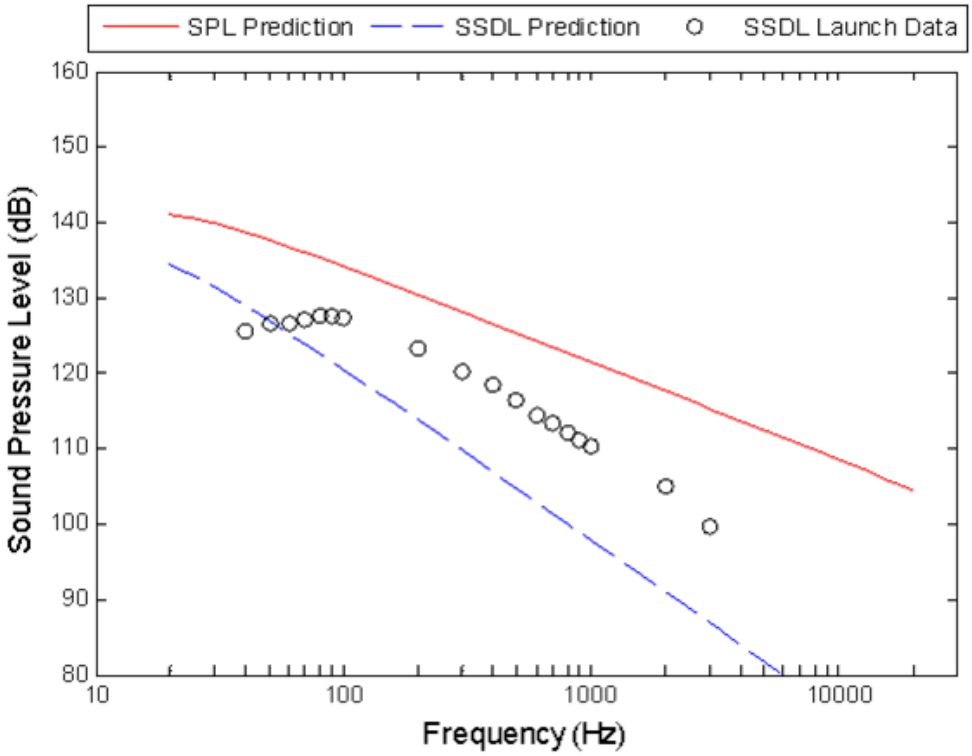

Figure 25: Saturn V noise levels at 85.3 m using no directivity index and no surface diffraction.

The prediction location 85.3 m was used to test the effects on the predictions with modifications to the Eldred method for different directivity indexes and the inclusion of the surface diffraction effect. Starting with the completely unmodified prediction method, in Figure 25, different directivity indexes were applied with no surface diffraction effects, see Equation (19). The results were then compared. The directivity functions and plots

used were: (1) the proposed Eldred's directivity[2] from Equation (12), see Figure 26; (2) Wilby's 1st directivity[24] from Equation (13), see Figure 27; (3) Wilby's 2nd directivity from Equation (14), see Figure 28; (4) Wilby's 3rd directivity from Equation (15), see Figure 29; and (5) Plotkin's directivity[25] from Equation (17), see Figure 30. Except for the plots obtained using Wilby's 2nd directivity function shown in Figure 28, all the SSDL predictions are nearly the same. They under predict by approximately 10 dB across most of the frequency spectrum. This may be a result of using an un-deflected directivity equation with predictions for a deflected exhaust flow. A real deflected exhaust flow would have sources reflected from the ground plane, which would cause the launch noise level data to be elevated by as much as 6 dB. The predictions made using Wilby's 2nd directivity function drastically underpredicts the SSDL launch noise levels by nearly 30 dB. The profile of the prediction curve is very interesting because it nearly matches the profile of the launch noise data.

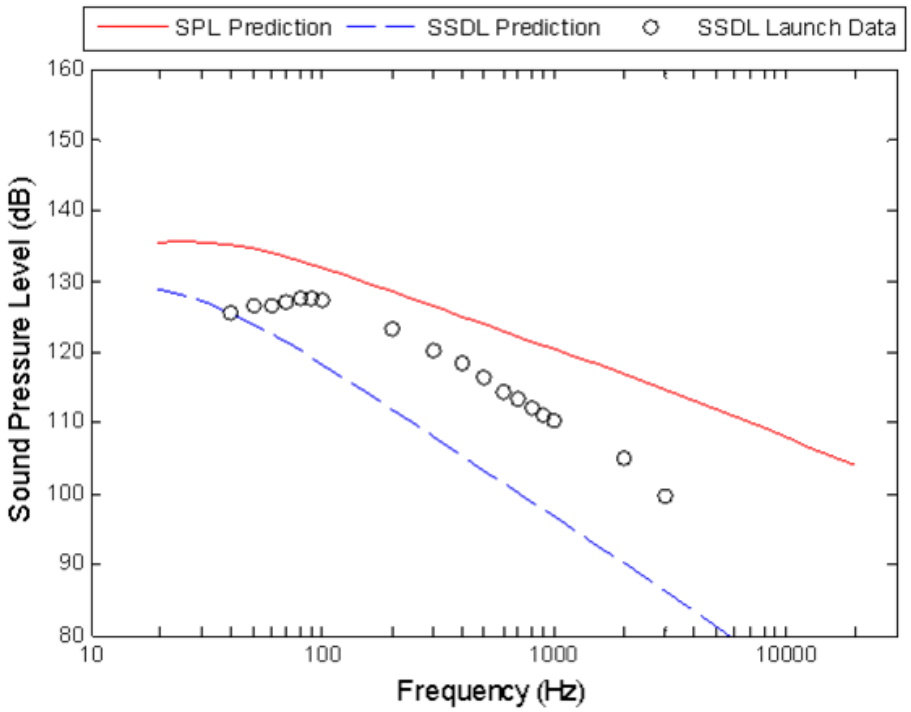

Figure 26: Saturn V noise levels at 85.3 m using Eldred's directivity and no surface diffraction.

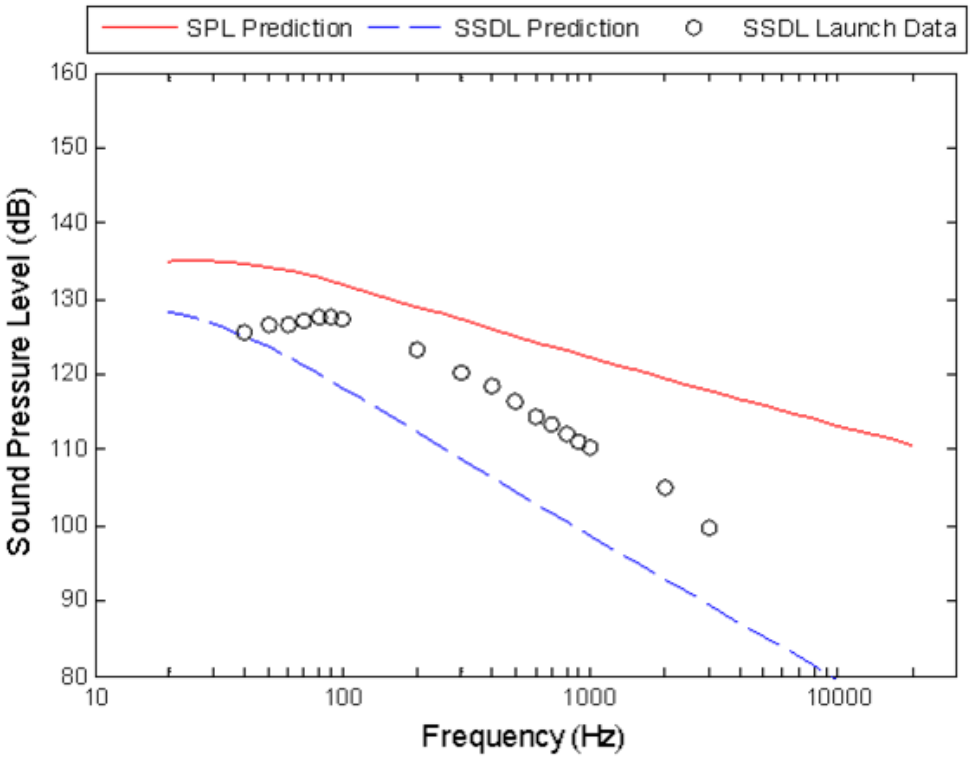

Figure 27: Saturn V noise levels at 85.3 m using Wilby's 1st directivity and no surface diffraction.

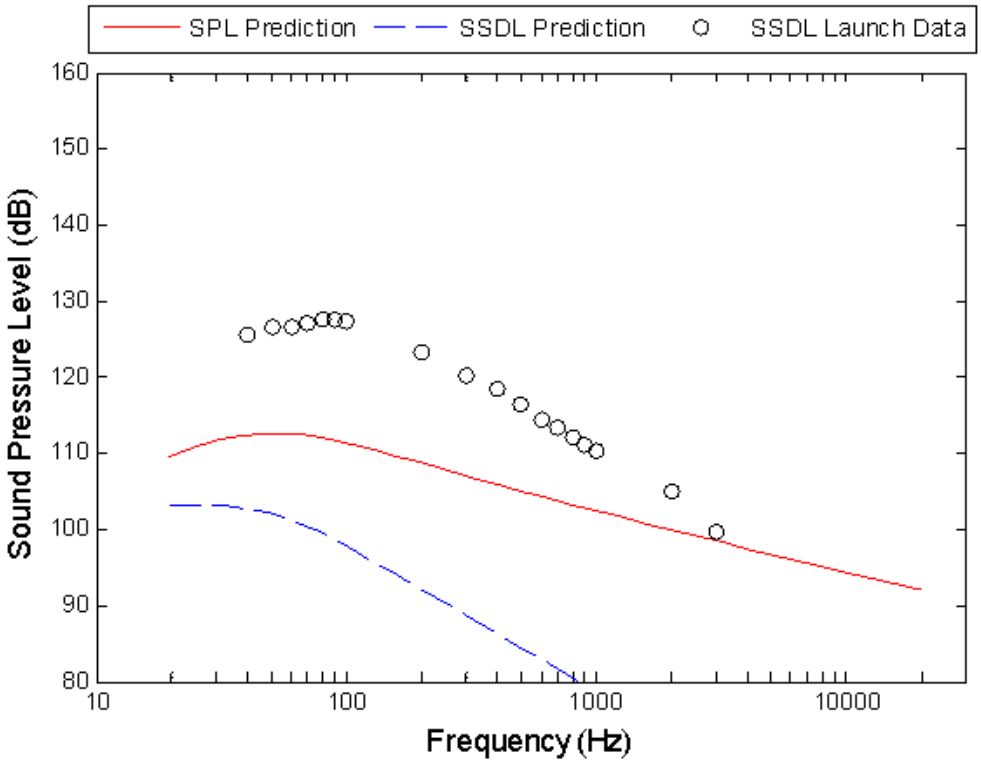

Figure 28: Saturn V noise levels at 85.3 m using Wilby's 2nd directivity and no surface diffraction.

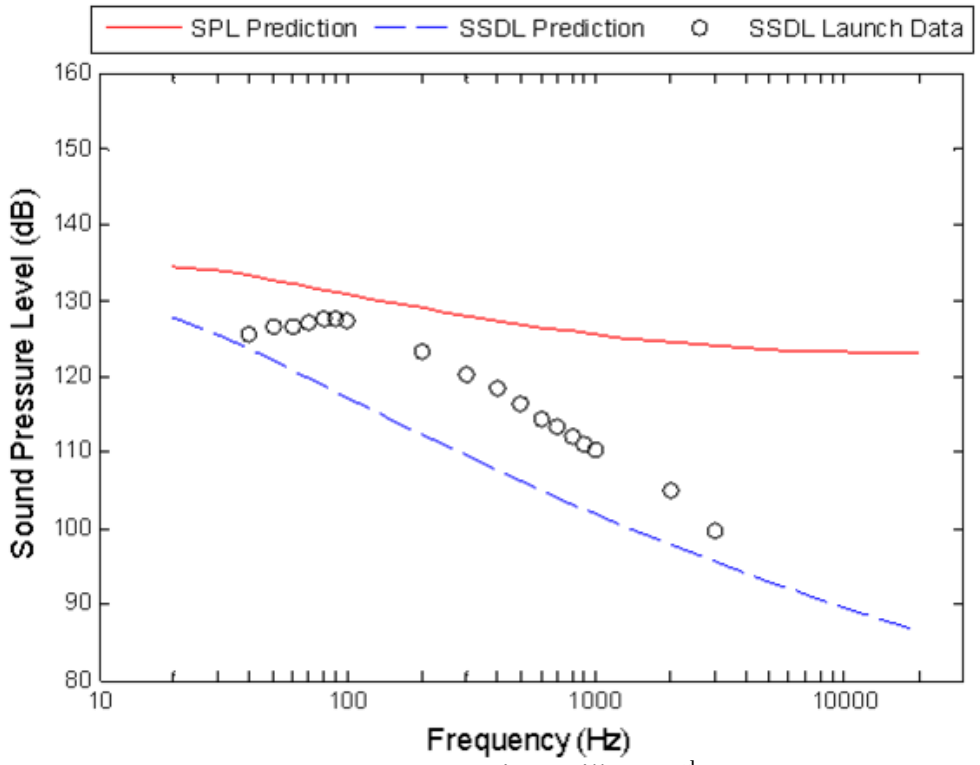

Figure 29: Saturn V noise levels at 85.3 using Wilby's 3rd directivity and no surface diffraction.

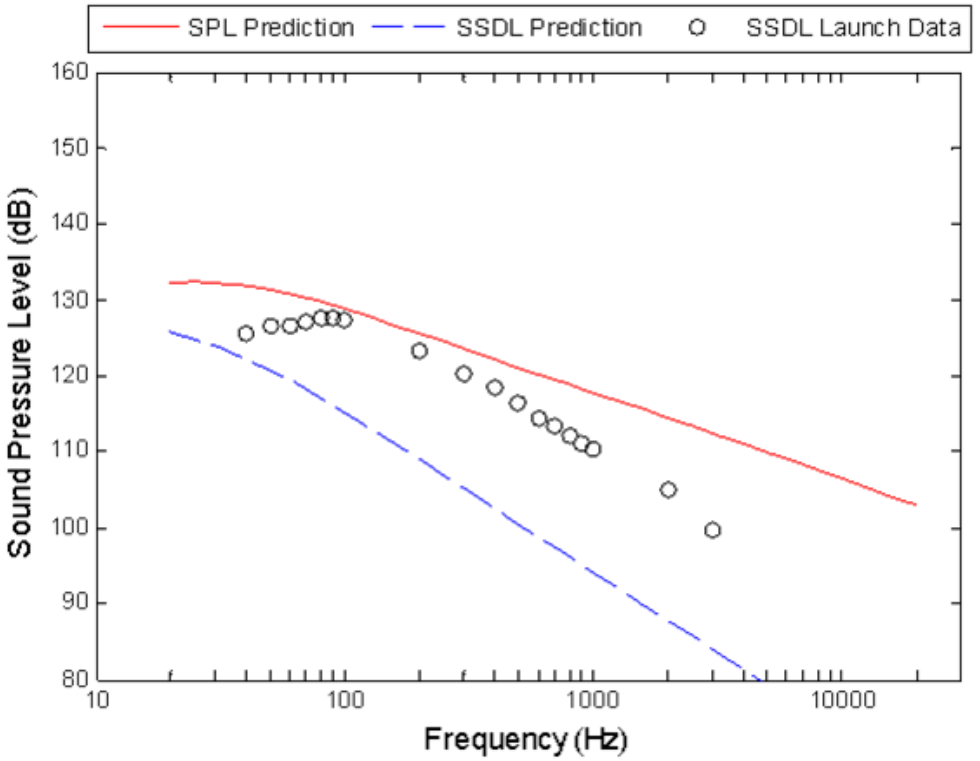

Figure 30: Saturn V noise levels at 85.3 m using Plotkin's directivity and no surface diffraction.

The same directivity functions were used with surface diffraction effects included in the predictions and the plots given include: (1) the proposed Eldred's directivity, see Figure 31; (2) Wilby's 1st directivity, see Figure 32; (3) Wilby's 2nd directivity, see Figure 33; (4) Wilby's 3rd directivity, see Figure 34; and (5) Plotkin's directivity, see Figure 35. Again, all the SSDL predictions are nearly the same except for Wilby's 2nd directivity prediction. The surface diffraction predictions are approximately 5 dB higher than the non-surface diffraction predictions. The general slight increase in the predicted levels has caused the lower frequencies to be over-predicted.

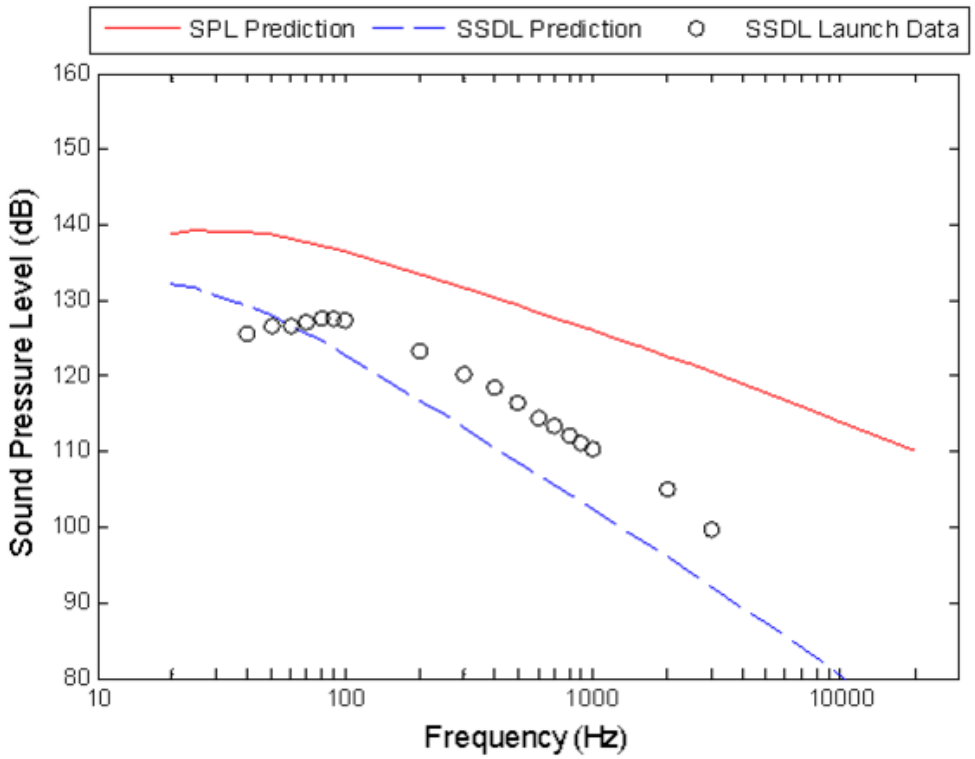

Figure 31: Saturn V noise levels at 85.3 m using Eldred's directivity and surface diffraction effects.

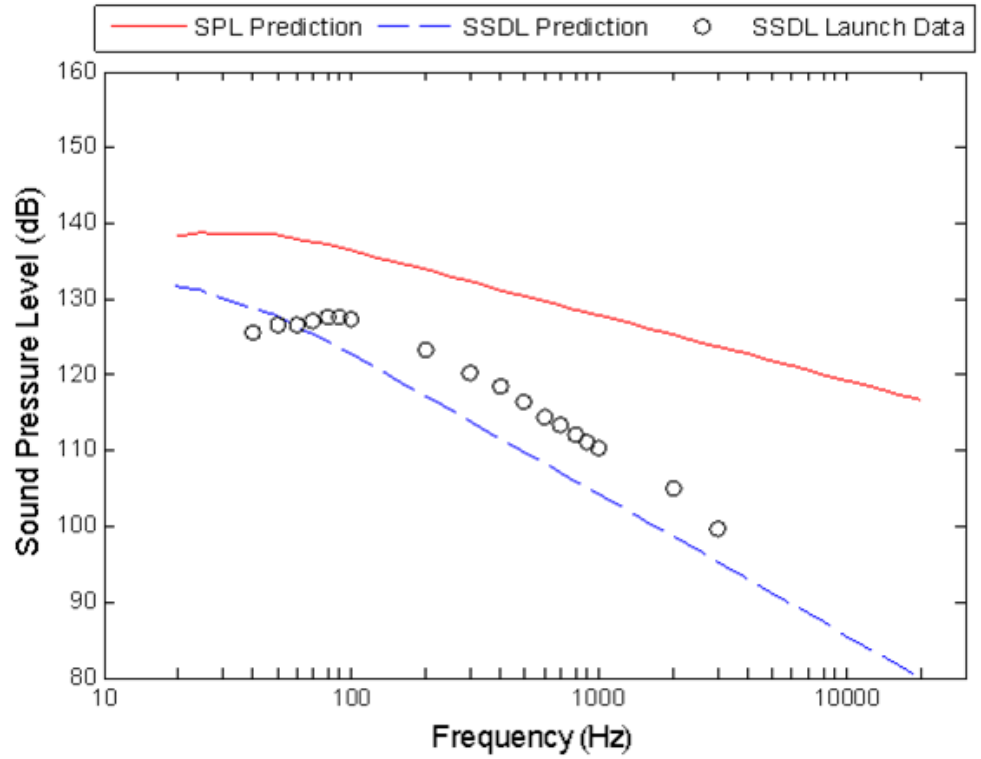

Figure 32: Saturn V noise levels at 85.3 m using Wilby's 1st directivity and surface diffraction effects.

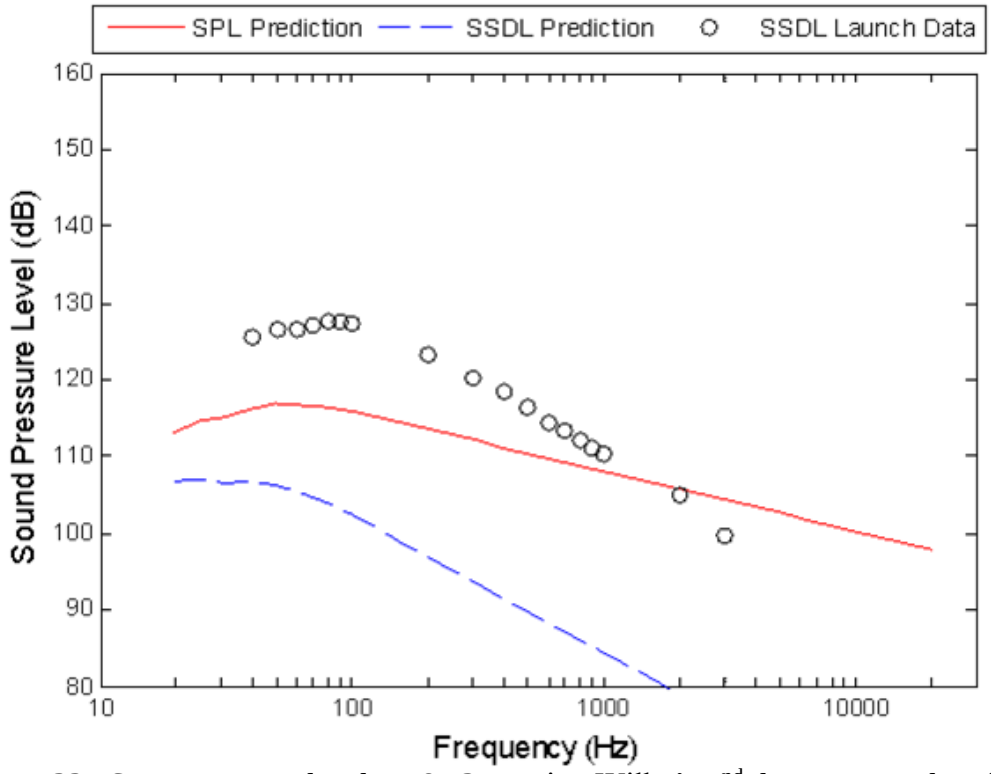

Figure 33: Saturn V noise levels at 85.3 m using Wilby's 2nd directivity and surface diffraction effects.

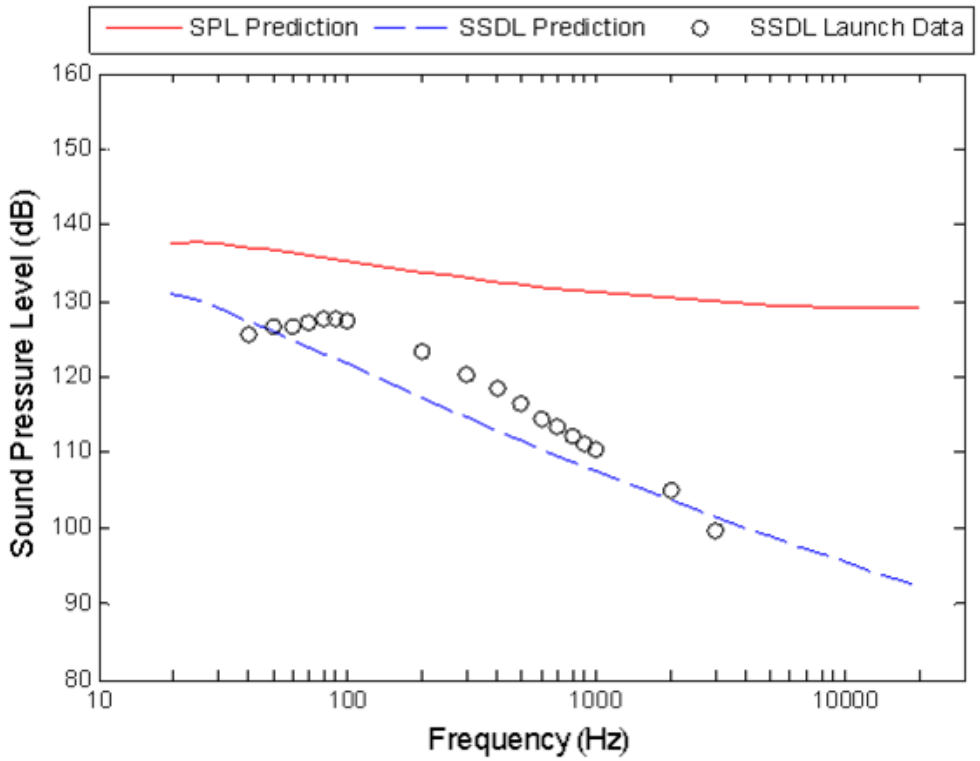

Figure 34: Saturn V noise levels at 85.3 m using Wilby's 3rd directivity and surface diffraction effects.

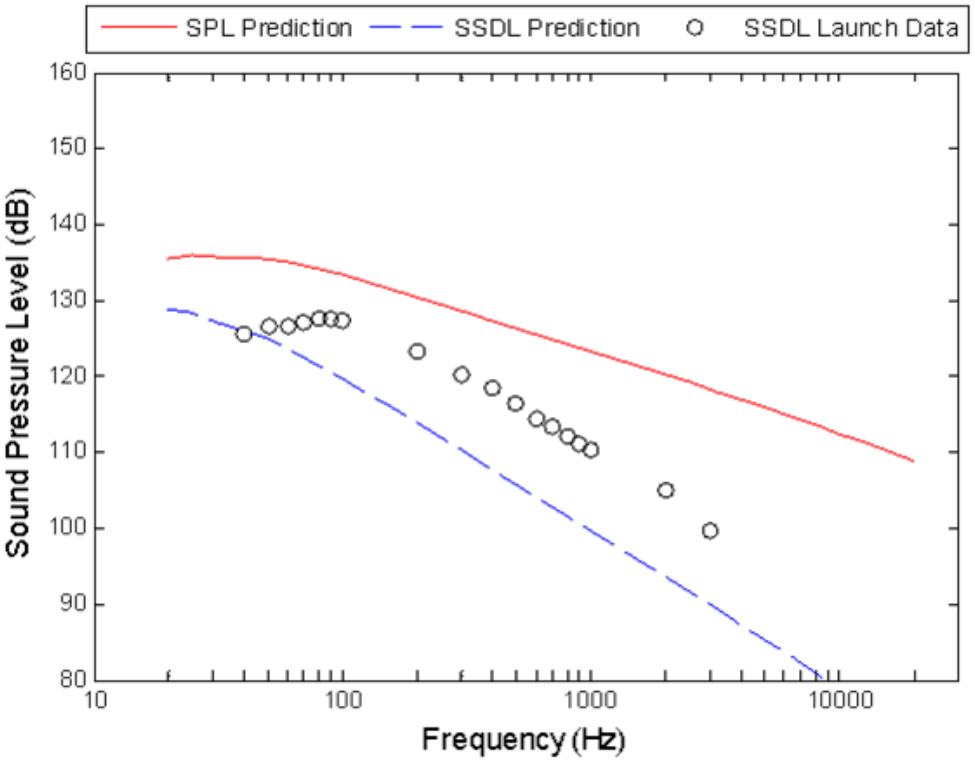

Figure 35: Saturn V noise levels at 85.3 m using Plotkin's directivity and surface diffraction effects.

The frequency spectra shown in the previous ten plots reasonably follow the launch data, but there is no observable conclusion as to which directivity function should be used. Instead of relying on only visual observations, the sound pressure levels can be summed to achieve a single number for comparison. The overall sound pressure level (OASPL) is calculated by[23],

$$\text{OASPL} = 10\log_{10}\left[\sum\left(10\text{^}\left(\text{SPL}/10\right)\right)\right]. \tag{21}$$

The OASPL was computed for each of the different cases and tabulated in Table 2 below. The table includes the reference prediction shown in Figure 25 without any (None) directivity index and no surface diffraction effect. Three cases, Eldred's directivity, Wilby's 1st directivity, and Wilby's 3rd directivity, seem to provide the closest OASPL to the launch data. All three cases were obtained using the surface diffraction effect. Using these three cases as a basis to determine which directivity index to use in the predictions, Wilby's 3rd directivity approximates the launch data at both ends of the launch data frequency spectrum using the surface diffraction equation.

Table 2: OASPL for various directivity indexes and surface diffraction effects for Saturn V at 85.3 m.

OASPL (dB)	Surface Diffraction	
Directivity Index	No	Yes
None	148.1	
Eldred	144.6	148.6
Wilby 1st	144.4	148.5
Wilby 2nd	122.6	127.0
Wilby 3rd	144.1	148.5
Plotkin	141.4	145.5
OASPL Launch Data (dB)		150.8

Using Wilby's 3rd directivity with the surface diffraction equation, the noise levels at the remaining chosen microphone locations were predicted. The predictions are presented in Figures 36 through 40 with the OASPL shown in Table 3 below. Most of the SSDL predictions approximate the launch data except below 80 Hz, at which point the predictions tend to overpredict the launch noise data.

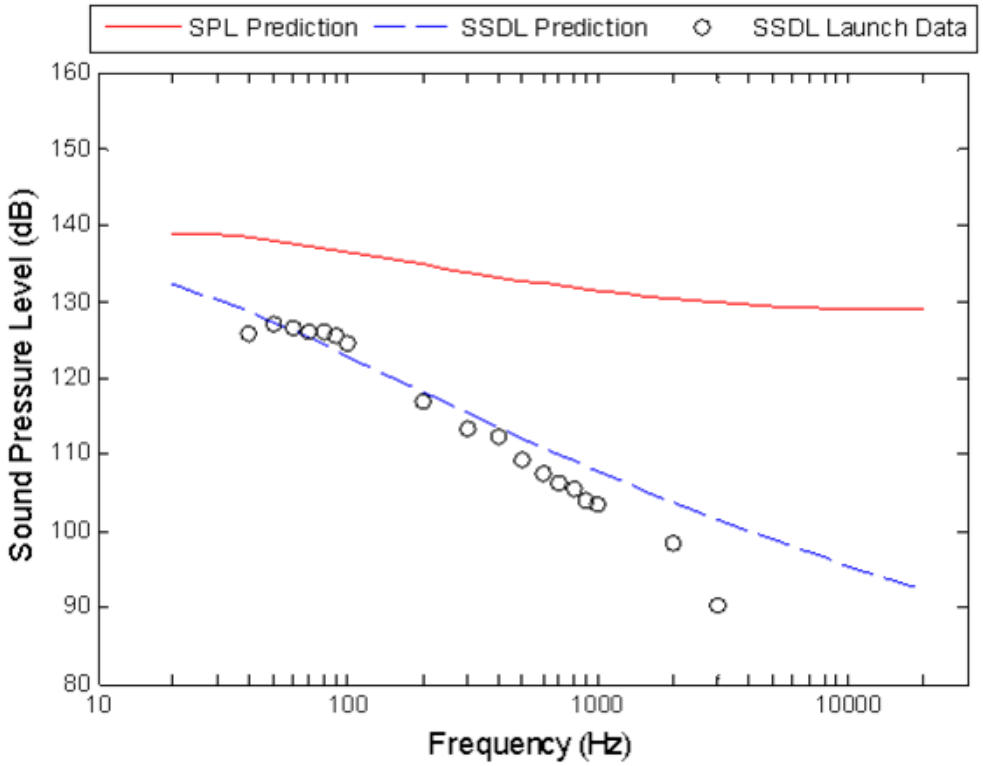

Figure 36: Saturn V noise level predictions at 84.5 m.

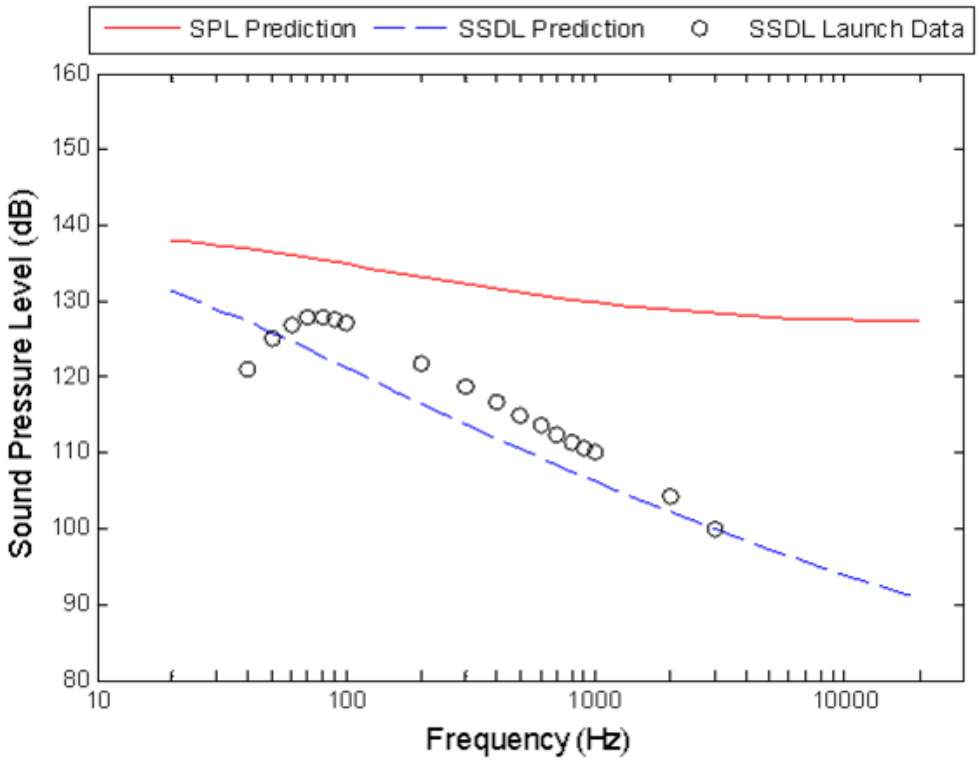

Figure 37: Saturn V noise level predictions at 66.3 m.

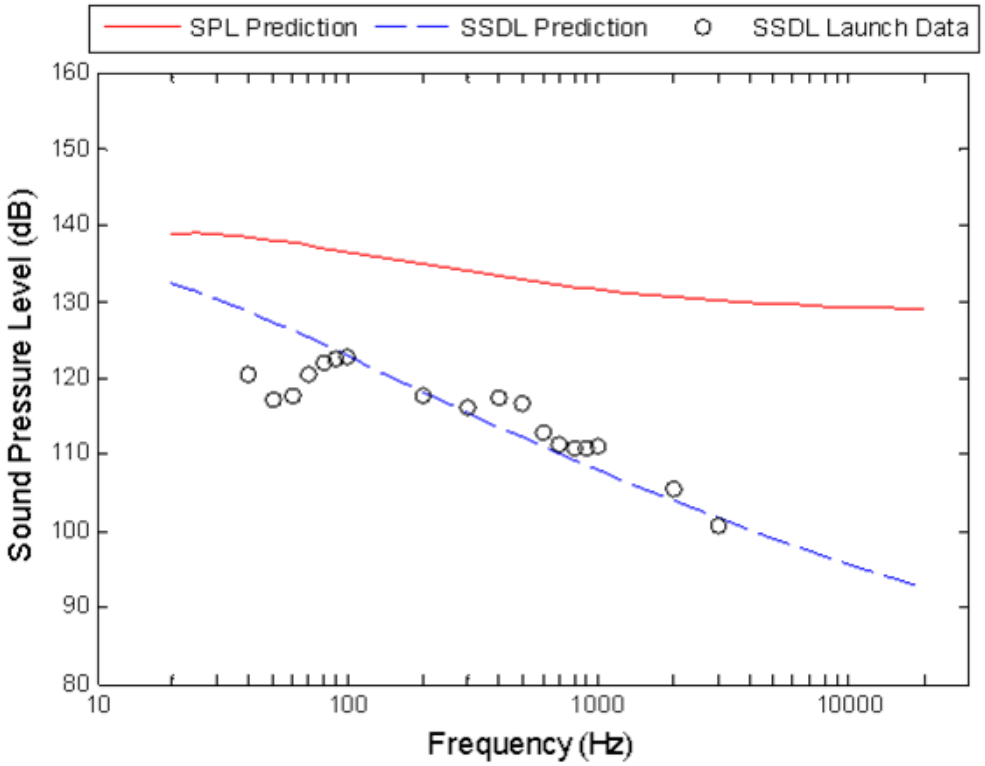

Figure 38: Saturn V noise level predictions at 45.2 m.

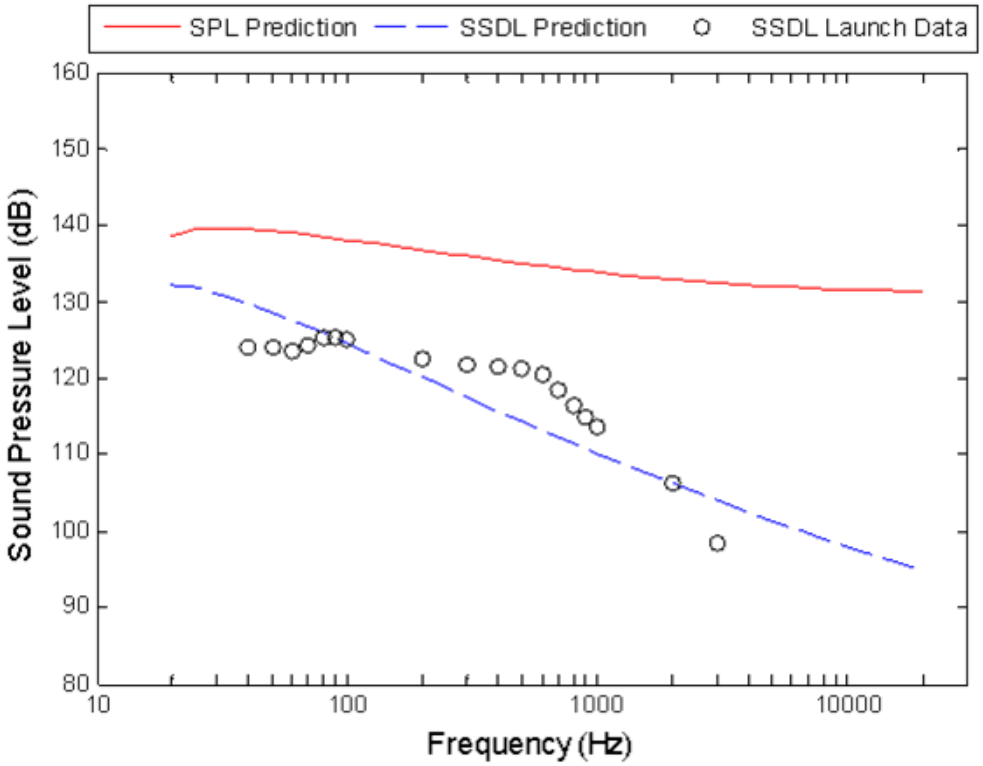

Figure 39: Saturn V noise level predictions at 22.0 m.

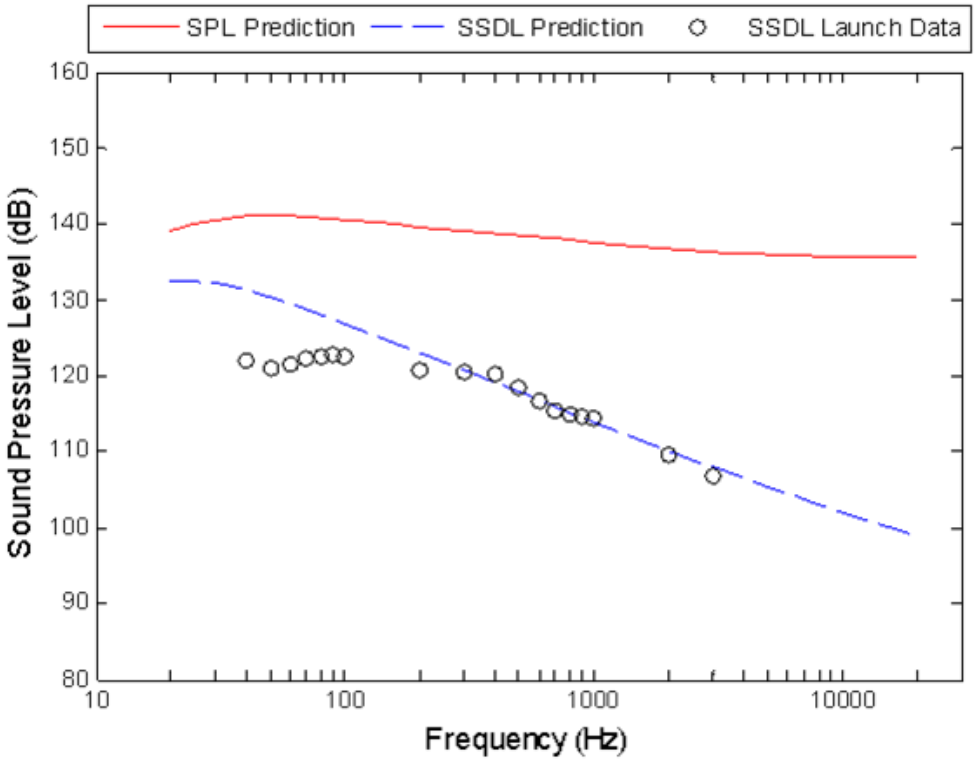

Figure 40: Saturn V noise level predictions at 6.0 m.

Table 3: OASPL prediction and launch data for Saturn V.

Distance from exit plane (m)	Prediction Wilby 3rd with Diffraction (dB)	Launch Data (dB)
85.3	148.5	150.8
84.5	149.5	147.4
66.3	148.0	150.1
45.2	149.6	147.4
22.0	150.0	151.4
6.0	153.5	150.2

3.2 Space Shuttle Launch Noise Predictions

With the predictions for the Saturn V launch vehicle completed in Section 3.1, the Space Shuttle sound pressure levels were predicted. Some modifications to the exhaust flow parameters for the Saturn V were required so that the Eldred empirical model could be used for the Space Shuttle predictions. Starting with the number and location of each engine, the Space Shuttle test specifications report, compiled by Murphy[32], shows that there were three Space Shuttle main engines (SSME) and two solid rocket boosters (SRB) used for each shuttle flight. The five engines were assumed to be located at (Cartesian [X, Y, Z] m): SSME engine 1 [1.56, 7.73, 4.54], SSME engine 2 [0, 10.63, 4.54], SSME engine 3 [-1.56, 7.73, 4.54], SRB engine 1 [6.36, 0, 0], and SRB engine 2 [-6.36, 0, 0]. The SSMEs exit plane is 4.43 m higher than the SRB exit plane. A flat plate ramp was assumed to lie 27.2 m below the SRB exit plane, which will cause an apparent exhaust flow to turn from vertical to 22^{o} with the horizontal in the Y-Z plane. The remaining engine parameters[32] include: the engine thrust for the SSME 1665859 N and for the SRB 11787790 N; the engine exit temperature for the SSME 1673^{o}K and for the SRB 3477^{o}K; the engine nozzle

exit diameter for the SSME 2.30 m and for the SRB 3.70 m; and the engine exit velocity for the SSME 4420 m/sec and for the SRB 2332 m/sec.

Sound pressure level predictions were made at five locations on the Space Shuttle. The five positions shown in Figure 41 include the microphone numbers as follows: ① left SRB forward skirt, microphone 7988; ② right SRB attachment ring, microphone 8981; ③ left SRB above aft skirt edge, microphone 7987; ④ right SRB above aft skirt edge, microphone 8982; and ⑤ left SRB aft skirt edge, microphone 7986. Table 4 below summarizes the microphone numbers and locations. The microphone data recordings[33] started approximately at the same instant of time after engine ignition. From this point forward discussion about these microphone locations will be referred to as: prediction location 38.5 m (microphone 7988), prediction location 12.2 m (microphone 8981), prediction location 3.2 m (microphone 7987), prediction location 3.1 m (microphone 8982), and prediction location 1.6 m (microphone 7986).

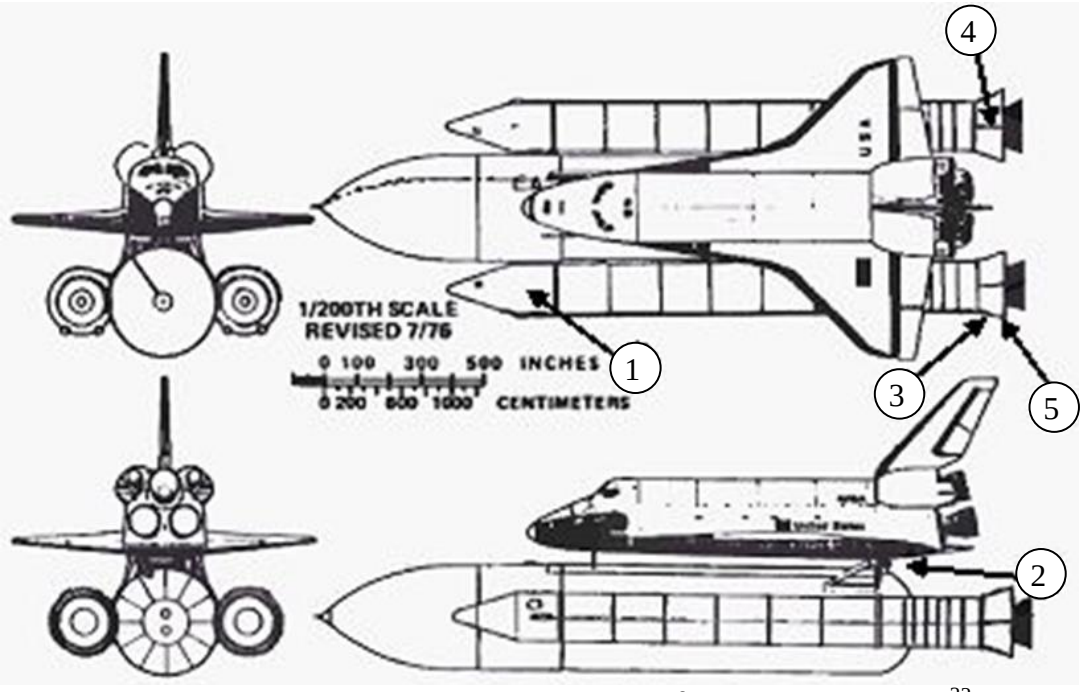

Figure 41: External microphone locations for the Space Shuttle.[32]

Table 4: Space Shuttle chosen microphone locations for noise prediction comparisons.

Mic. #	Launch STS-5, Microphone #	Launch Time (sec)	Cartesian (m)		
			X	Y	Z
1	7988	1.2-1.7	-7.7	1.3	38.5
2	8981	0.9-1.9	6.4	1.9	12.2
3	7987	0.9-1.9	-8.5	0.0	3.2
4	8982	0.9-1.9	6.4	2.2	3.1
5	7986	0.9-1.4	-8.9	0.0	1.6

When both the SSME and SRB engines are operating, the Space Shuttle noise level spectra predictions are shown in Figures 42 through 46 at the different locations. The figures include the SPL predictions using Wilby's 3rd directivity and including surface diffraction effects, the SPL launch data, and the SSDL predictions. The launch data for the Space Shuttle generally have less low frequency and more high frequency levels than the predictions indicate. The further away from the exhaust exit plane, the better the predictions were, specifically referring to Figure 42 below. At prediction locations closer to the exhaust exit plane, more non-linear effects become apparent, as shown in Figures 43 through 46 below. While the frequency spectrum predictions have limited agreement with the launch data, the OASPL, shown in Table 5 below, have improved disparity.

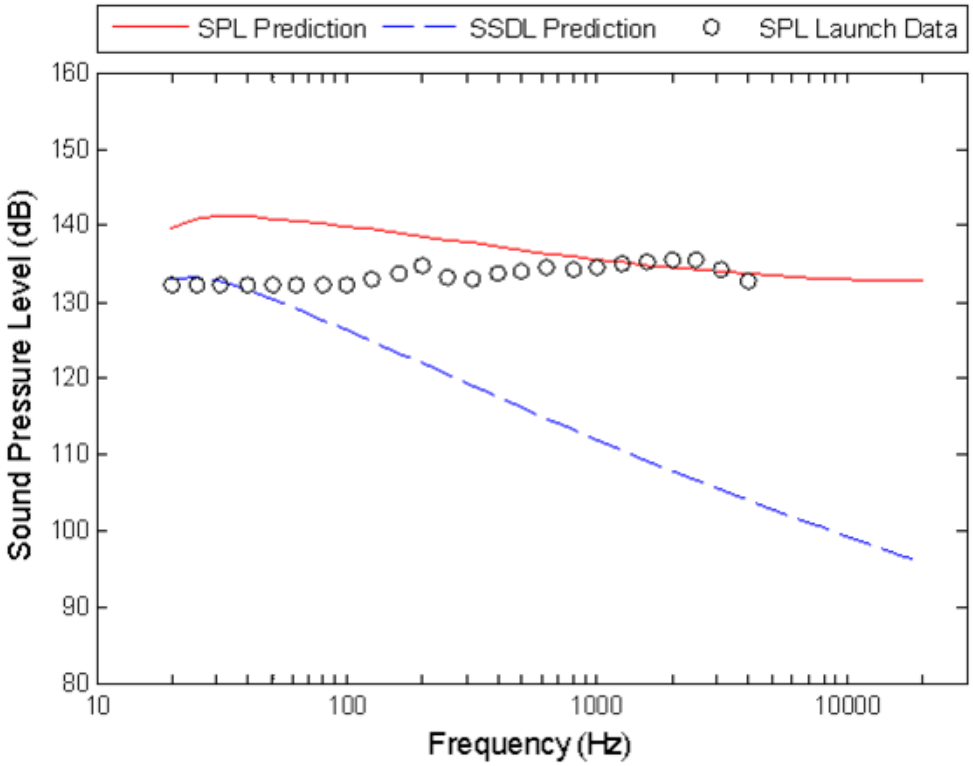

Figure 42: Space Shuttle noise level predictions at 38.5 m.

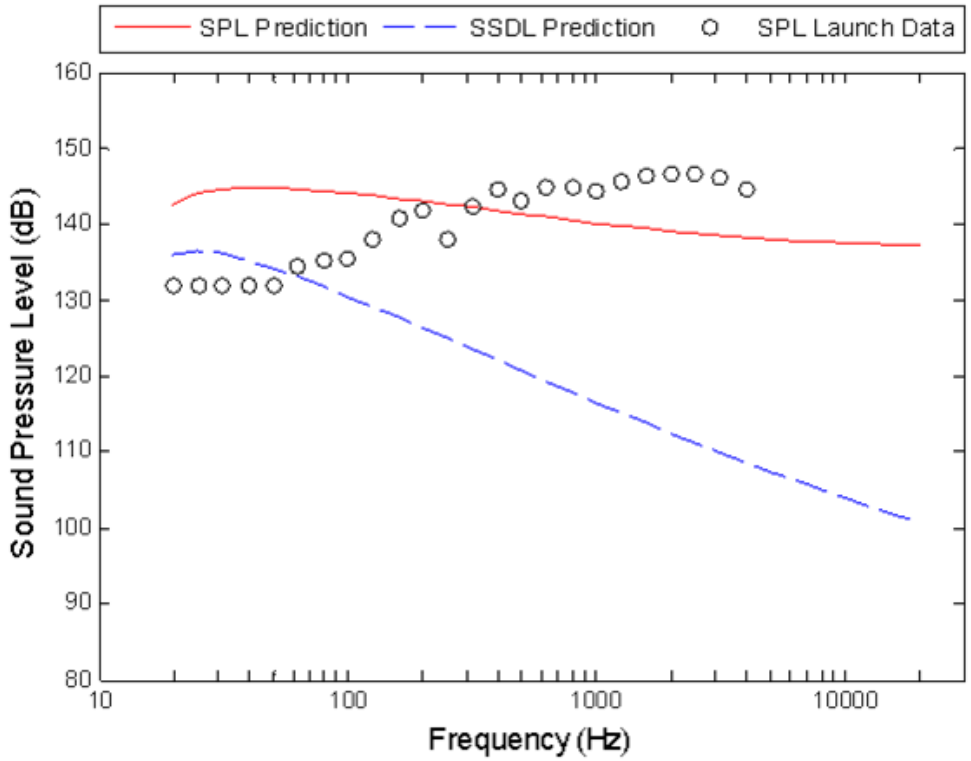

Figure 43: Space Shuttle noise level predictions at 12.2 m.

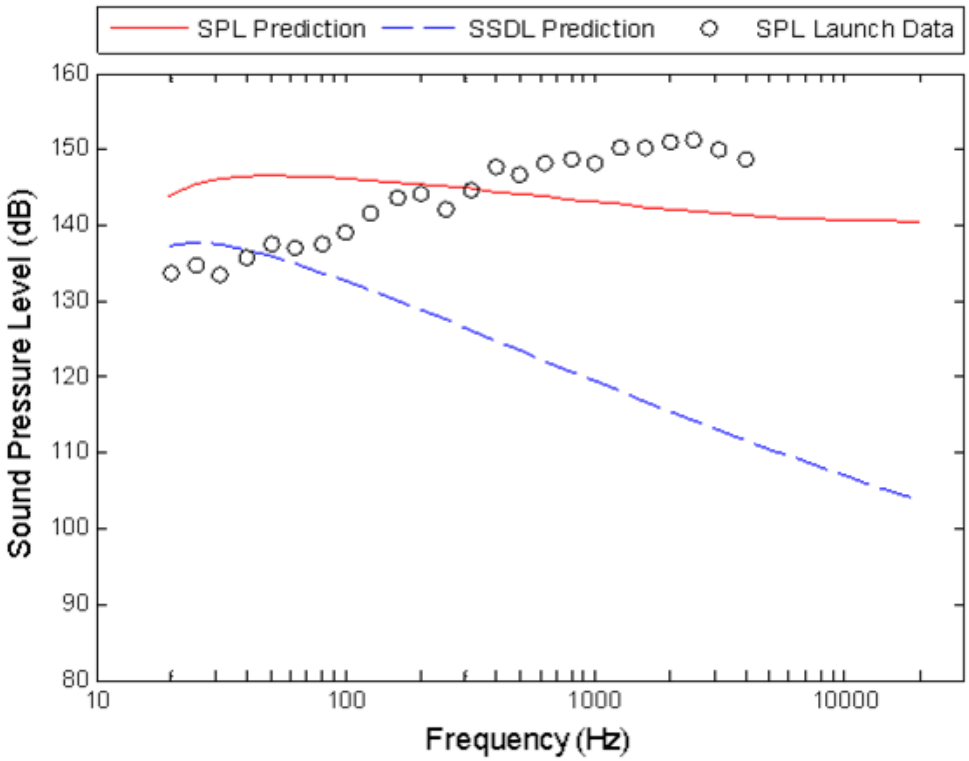

Figure 44: Space Shuttle noise level predictions at 3.2 m.

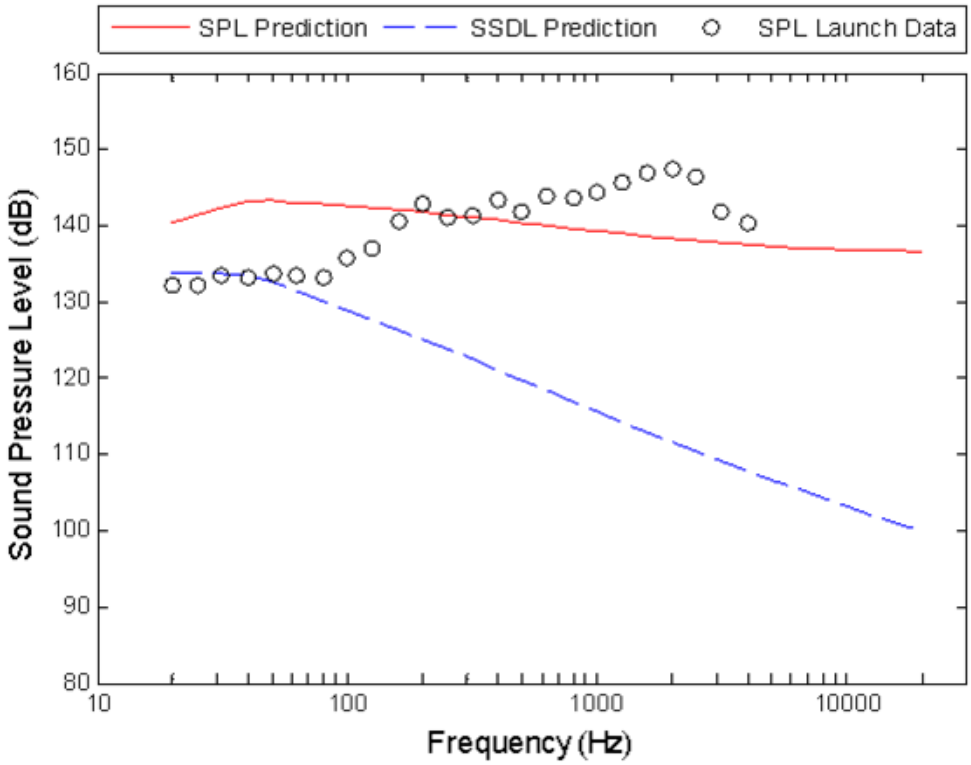

Figure 45: Space Shuttle noise level predictions at 3.1 m.

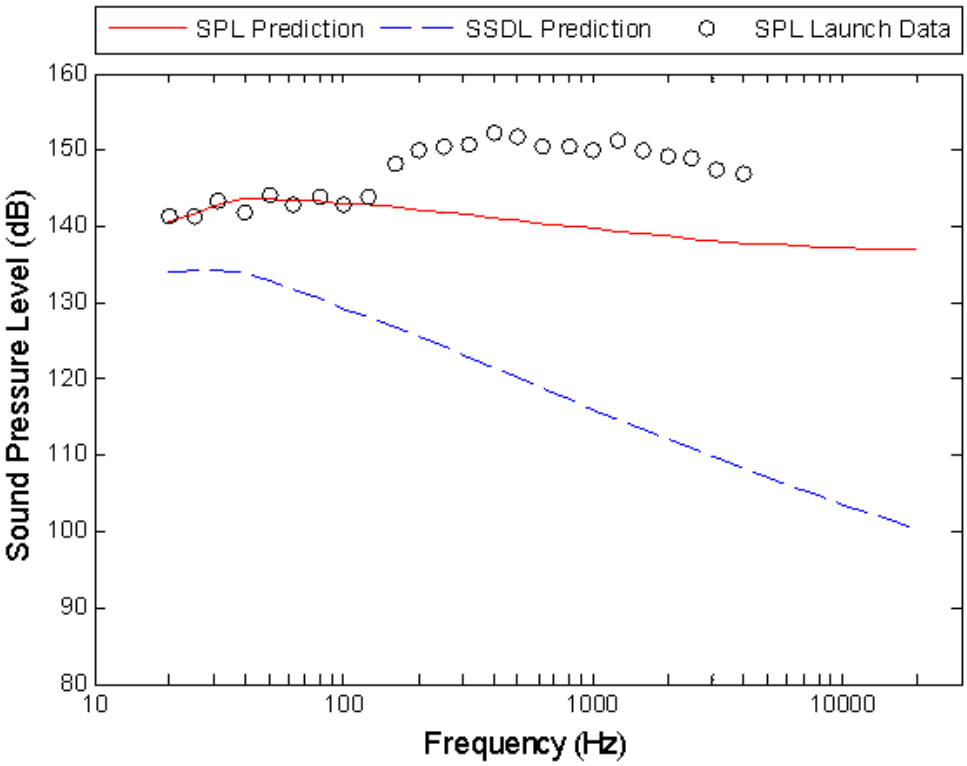

Figure 46: Space Shuttle noise level predictions at 1.6 m.

Table 5: OASPL predictions and launch data for the Space Shuttle.

Distance from exit plane (m)	Prediction Wilby 3rd with Diffraction (dB)	Launch Data (dB)
38.5	153.1	147.6
12.2	157.1	156.9
3.2	155.7	156.1
3.1	159.4	160.3
1.6	156.1	162.3

3.3 Ares I-X Launch Noise Predictions

The NASA Constellation Program developed the Ares launch vehicles as a direct replacement for the then retiring Space Shuttle. The new designs, shown in Figure 47 below, replaced the reusable Space Shuttle glider type to a multi-stage Saturn V type of space vehicle. The Ares I was the smallest of the new vehicle designs, which used a two stage configuration. The first stage of the Ares I incorporated a reusable solid rocket motor, which was based on the design of the Space Shuttle's solid rocket motors. In order to test the flight worthiness of the Ares I design, the Ares I-X was developed as a proof of concept vehicle.[34]

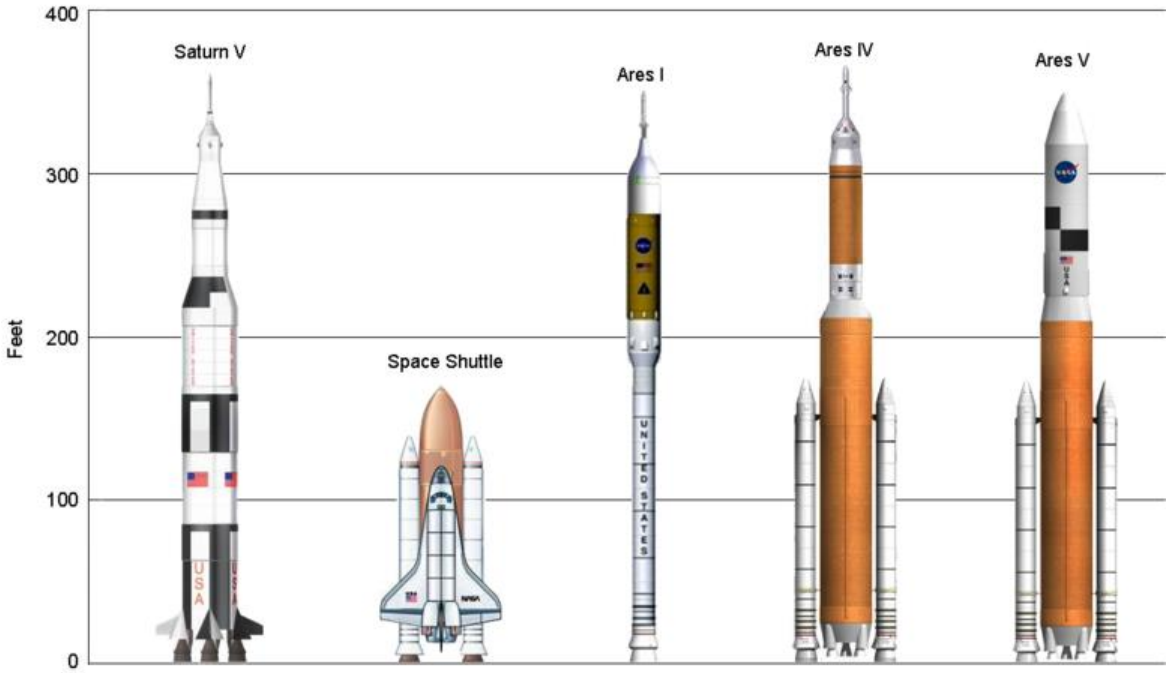

Figure 47: Size comparison between space vehicles.[35]

The Ares I-X exhaust flow parameters required for use in the Eldred empirical noise prediction method were taken from information provided in an email from J. M. Haynes[36] on January 16, 2009. Haynes was an acoustical engineer at the Marshall Space Flight

Center, and his input was invaluable to complete the Ares I-X predictions. He provided: the engine thrust of 14634649.1 N, the nozzle exit diameter of 4.401 m, the engine exit velocity of 2529.84 m/sec, and the deflector ramp offset of 9.296 m. The engine exit temperature of 3477^{o}K was assumed to be the same as used on the Shuttle's booster. The assumption was based on unavailable data and the similarities in design between the two vehicles. The single engine exit was assumed to be positioned at Cartesian coordinates [0, 0, 0] m.

Two noise predictions were made for the Ares I-X vehicle based on available launch data from D. D. Counter's report.[34] He compared the sound pressure levels (SPL) at the two locations shown in Table 6 below. These microphone locations will be referred to as: prediction location 83.8 m (IAD915P) and prediction location 56.4 m (IAD098P). Figures 48 and 49 compare the SPL predictions with the Ares I-X SPL launch data. These figures show that the predictions using Eldred reasonably predict the SPL using the Wilby's 3rd directivity and surface diffraction effect. The Ares I-X noise levels at the lower frequencies were over-predicted, in the same way as the Saturn V noise levels. The OASPL, shown in Table 7 below, is over-predicted as well.

Table 6: Ares I-X chosen microphone location used for prediction comparisons.

Microphone #	Launch #	Cartesian (m)		
		X	Y	Z
IAD915P	Ares I-X	0.0	2.8	83.8
IAD098P	Ares I-X	0.0	2.8	56.4

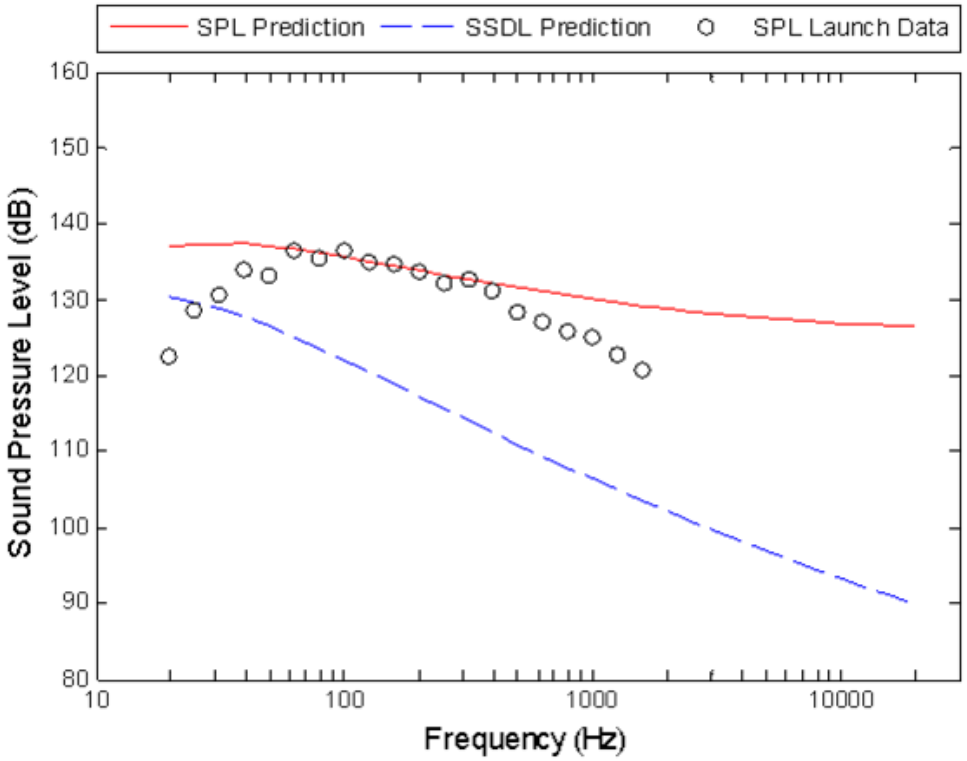

Figure 48: Ares I-X noise level predictions at 83.8 m.

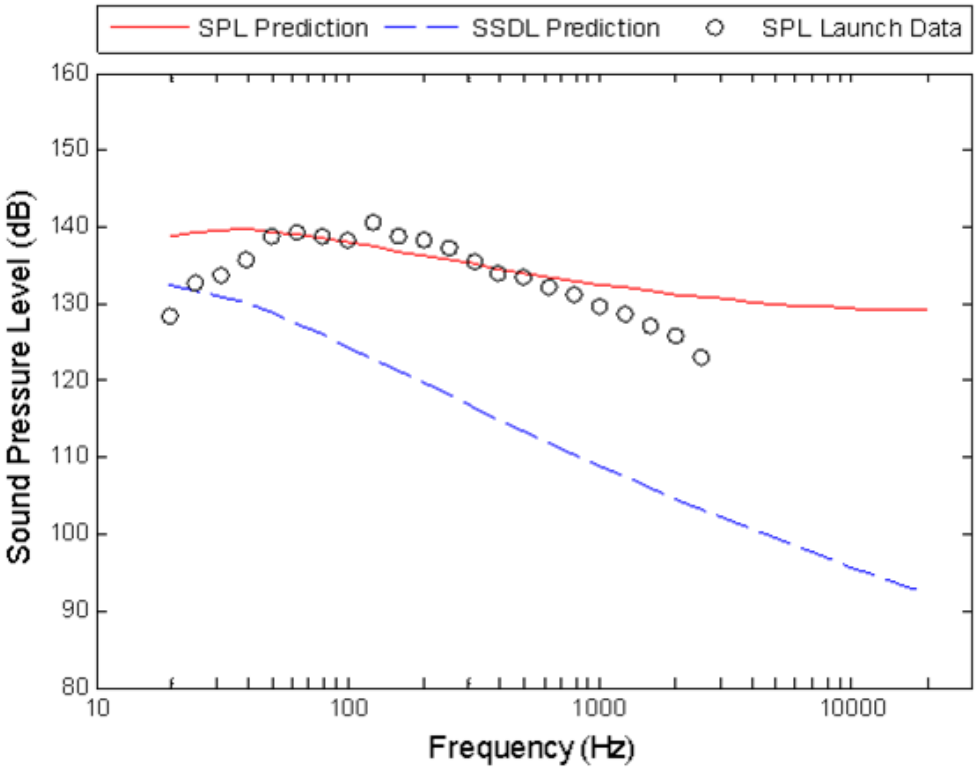

Figure 49: Ares I-X noise level predictions at 56.4 m.

Table 7: OASPL prediction and launch data for Ares I-X.

Distance from exit plane (m)	Prediction Wilby 3rd with Diffraction (dB)	Launch Data (dB)
83.8	148.2	145.4
56.4	150.5	149.2

3.4 Supersonic Jet Wind Tunnel Noise Predictions

Norum[37] has performed some experimental research on jet exhaust noise at the NASA Langley Low Speed Aero-acoustics Wind Tunnel. His wind tunnel investigations, shown in Figure 50 below, included other aspects of jet noise not explicitly related to this dissertation research. Norum was primarily interested in the noise reduction that could be obtained using water injection. The injector parameters such as injection locations,

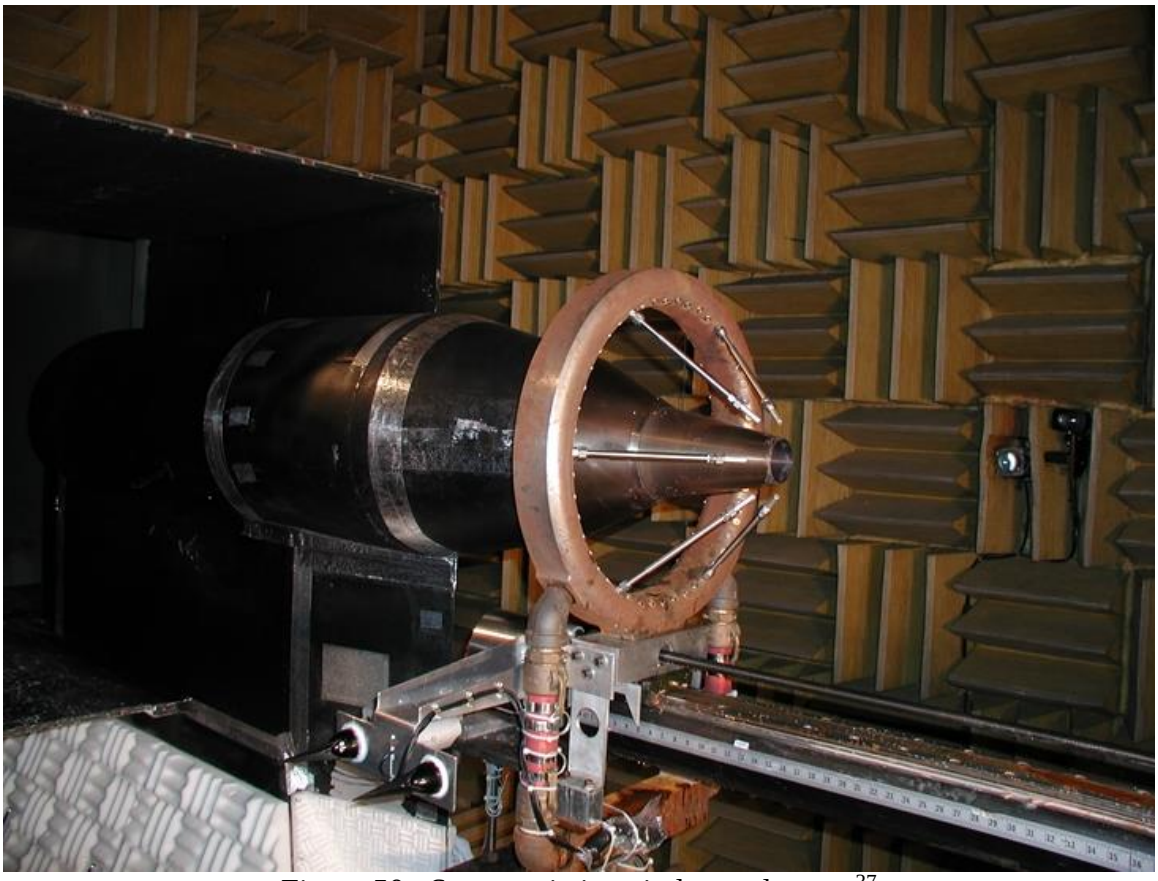

Figure 50: Supersonic jet wind tunnel setup.[37]

spray patterns, and mass flow rates were adjusted to suppress the dominant noise sources from the jet nozzles. Norum injected water into the shear layers of both cold and hot jets operating at subsonic and supersonic conditions. His results showed that the water injected disrupted the shock noise sources within the jet plume, and he was able to show large reductions in the radiated shock noise. However, only small reductions in jet mixing noise were made. An interesting fact was that the measured noise reduction in the upstream direction was consistently larger than in the noise in the downstream direction. Norum's results showed that water injection at the nozzle exit rather than downstream were required to achieve large noise reductions. The extent of the noise reduction was a direct result of the increase in the water pressure and mass flow rates.[37]

Concerning the various jet flow conditions investigated by Norum, only one case was selected to compare with Eldred's empirical model. The case used a convergent-divergent nozzle designed for shock-free flow at exit Mach (Ma) number of 1.5; however, the nozzle was run at 1.15 Ma for this particular investigation. The 1.15 Mach air flow rate would give an exit velocity of 406.6 m/s in ambient air of 311ºK. Norum's "cold" jet investigation was for a nozzle exit air temperature using the aforementioned ambient air temperature. This nozzle had an exit diameter of 0.073 m. In order to estimate the thrust, F, of 794.3 N, the following equation was provided by Street[38],

$$F = \rho_e (U_e)^2 A_e, \tag{22}$$

where ρ_e is the exit flow density of 1.1354 kg/m^3, U_e (m/s) is the nozzle exit velocity, and A_e is the cross sectional area at the nozzle exit. The nozzle discharged into a wind tunnel that was able to provide a sound reflection free environment above 100 Hz. Due to

the style of experiment, the center of the exit plane was chosen for the Cartesian coordinates [0, 0, 0] m with no deflector ramp nor offset.

Although Norum took sound pressure measurements using 28 quarter inch free-field microphones at a distance of 3.5 m from the nozzle flow center line, three noise predictions were completed. The microphones were directed towards the nozzle exit and provided results for θ angles (Figure 3) between 135° and 30° to the flow axis. Three microphones, shown in "bold" in Table 8 below, will be referred to as: prediction location 3.5 m (θ=135°), prediction location 0.0 m (θ=90°), and prediction location -3.5 m (θ =45°).

Table 8: Supersonic jet chosen microphone location used for prediction comparisons.

Microphone,	Cartesian (m)				Microphone,	Cartesian (m)		
θ Angle (deg)	X	Y	Z		θ Angle (deg)	X	Y	Z
135	**0.0**	**3.5**	**3.5**		67	0.0	3.5	-1.5
132	0.0	3.5	3.2		63	0.0	3.5	-1.8
129	0.0	3.5	2.8		58	0.0	3.5	-2.2
125	0.0	3.5	2.4		54	0.0	3.5	-2.5
121	0.0	3.5	2.1		51	0.0	3.5	-2.9
116	0.0	3.5	1.7		47	0.0	3.5	-3.2
111	0.0	3.5	1.4		**45**	**0.0**	**3.5**	**-3.5**
106	0.0	3.5	1.0		42	0.0	3.5	-3.9
101	0.0	3.5	0.7		39	0.0	3.5	-4.3
95	0.0	3.5	0.3		37	0.0	3.5	-4.7
90	**0.0**	**3.5**	**0.0**		35	0.0	3.5	-5.0
84	0.0	3.5	-0.4		33	0.0	3.5	-5.4
78	0.0	3.5	-0.8		32	0.0	3.5	-5.7
72	0.0	3.5	-1.1		30	0.0	3.5	-6.1

The Eldred method has shown reasonable predictions for Saturn V, Space Shuttle, and Ares I-X launch cases using the Wilby's 3rd directivity and surface diffraction effect; however, the supersonic jet has a much lower sound power level and a different diffraction effect. Using Eldred's original overall acoustical power Equation (1), led to a gross over-

prediction of the sound pressure levels (SPL) for the supersonic jet. A much lower sound power level was calculated using Lush's[39] equation for sub-sonic jets,

$$W_{OA} = K_1 \rho_e (d_{ei})^2 (U_e)^4 / a_e \tag{23}$$

where W_{OA} (W) is the overall acoustical power, K_1 is a constant altered to 3E-5, d_{ei} (m) is the exit diameter, and a_e (m/s) is the speed of sound based on the nozzle exit temperature. A comparison of the sound power predicted by Eldred, Lush, and Norum is shown in Table 9 below. Norum's sound power was estimated based on a spatial average of the sound pressure along with an assumption that a uniform line source produced a cylindrical wave. Lastly, the surface diffraction effects used in the past launch cases were valid for predictions on the surface of the launch vehicles; however, the wind tunnel conditions approximated free space, i.e. not a launch vehicle surface.

Table 9: Sound power prediction of the supersonic jet.

	Sound Power (W)
Eldred	1614.7
Lush	14.0
Norum	23.5

The Wilby's 3rd directivity used in the past launch cases provided reasonable noise predictions compared with those of the supersonic jet. Figures 51 through 53 show the empirical model noise level predictions for the three microphones from Table 8. Generally, the model over-predicts the noise levels for the 3.5 m and the 0.0 m locations. Figure 53 is particularly interesting because the predication indicates a similar trend to the wind tunnel data for a directivity pattern that was discussed in Chapter 2, see Figure 9. Reasonable agreement between the OASPL predictions and the noise levels measured for supersonic jet occurs, as shown in Table 10 below.

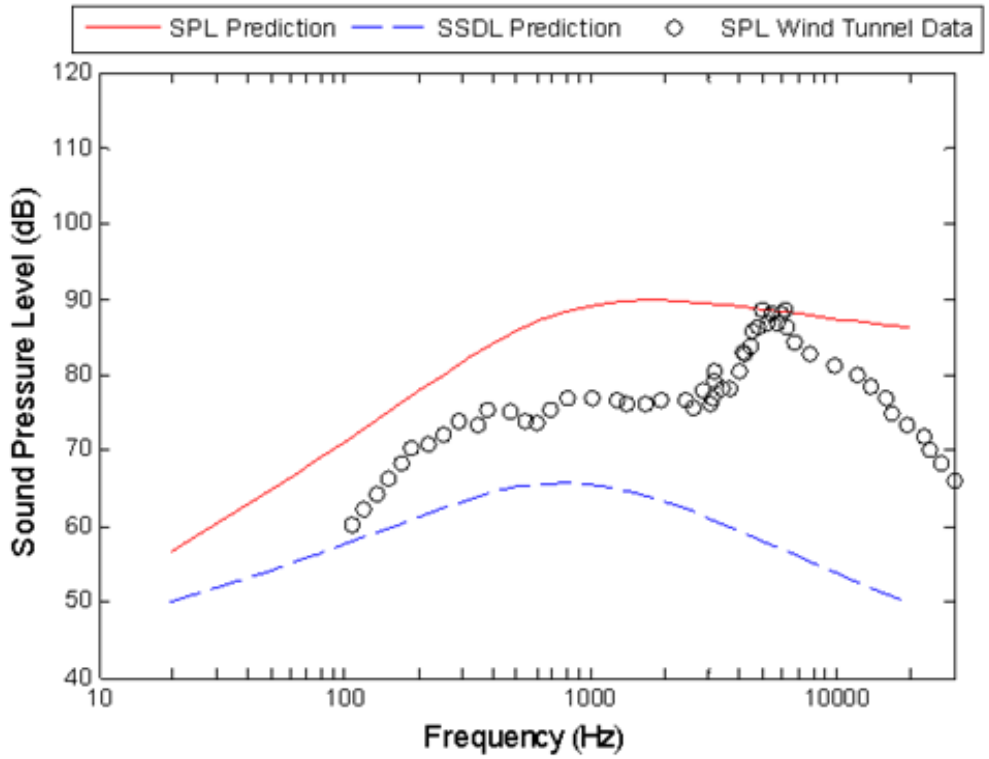

Figure 51: Supersonic jet noise level predictions at 3.5 m.

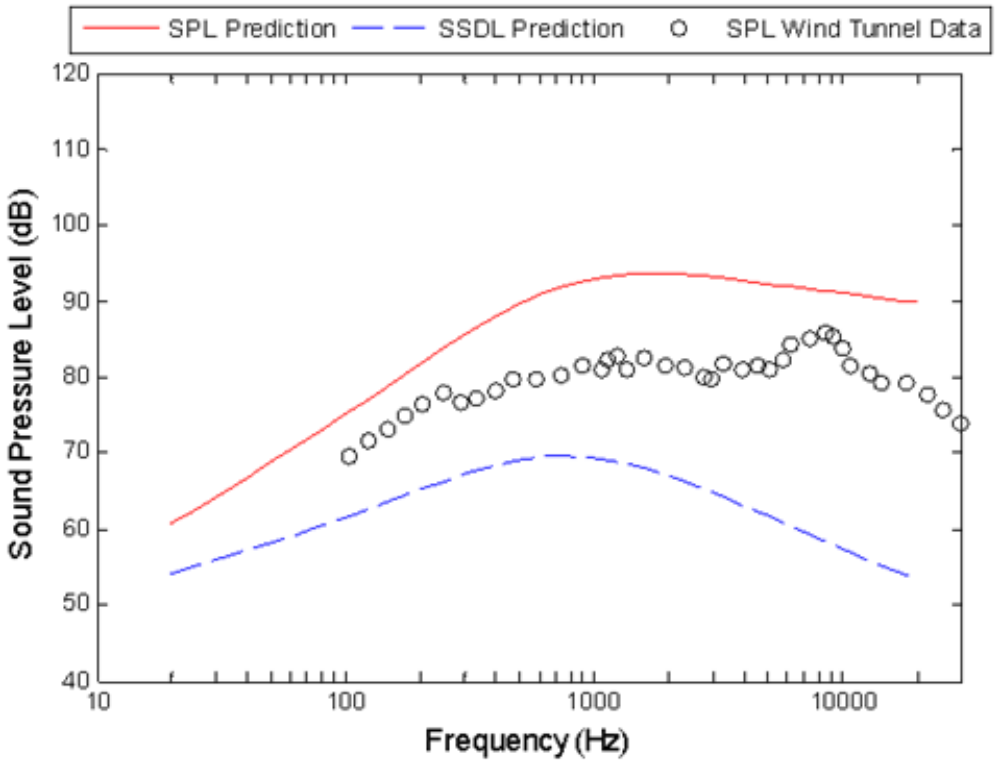

Figure 52: Supersonic jet noise level predictions at 0.0 m.

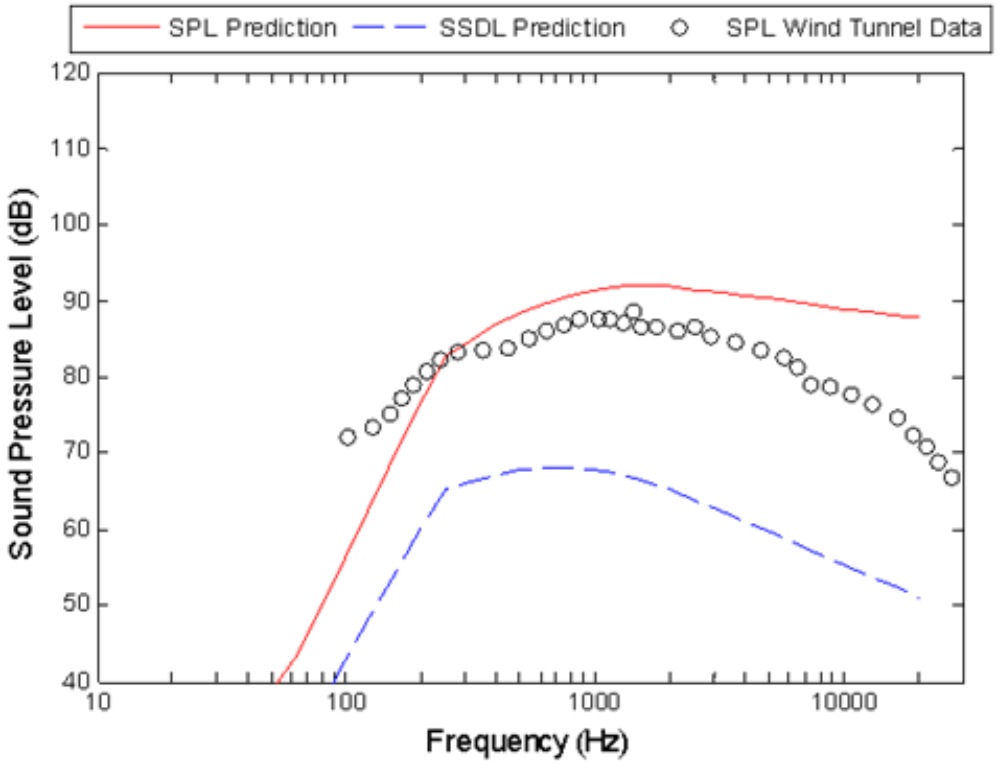

Figure 53: Supersonic jet noise level predictions at -3.5 m.

Table 10: OASPL prediction and data for supersonic jet.

Distance along Z axis (m)	Prediction Wilby 3rd w/o Diffraction (dB)	Wind Tunnel Data (dB)
3.5	101.0	98.9
0.0	104.7	96.9
-3.5	102.9	99.4

4 Computational Fluid Dynamic Model

In this chapter, the governing equations of fluid dynamics and heat transfer are described. It is assumed that the reader has some knowledge in this field of study; therefore, a complete derivation of the fundamental equations is not included. The fundamental equations are presented first. Then, equations of increasing complexity representing the simplest flow to the most complex are presented next.[40] The simplest inviscid Euler flow model provides the ability to describe the effects of a sound wave but not the generation of sound from turbulent flows. A detached eddy simulation (DES) of turbulent flow provides an abundance of information but requires prolonged initialization of the domain before the prediction of sound can be made. A shorter route to this initialization problem is to run a steady state Reynolds-averaged Navier-Stokes model first, and then transition to the unsteady DES. The order of materials presented in Section 4.1 Fluid Model Formulation are: Section 4.1.1, Inviscid Euler Fluid Model; Section 4.1.2, Reynolds-Averaged Navier-Stokes Fluid Model; and Section 4.1.3, Large Eddy Simulation and Detached Eddy Simulation. The last Section 4.2, Acoustic Model Formulation, discusses the methodology for the acoustic predictions in the following order: Section 4.2.1 Proudman Acoustic Model and Section 4.2.2 Ffowcs Williams – Hawkings Acoustic Model.

4.1 Fluid Model Formulation

The fundamental equations of fluid dynamics are founded on the general laws of conservation. The three fundamental laws are: Conservation of Mass, Conservation of Momentum, and Conservation of Energy. The first equation applies the Conservation of Mass law to a fluid flow resulting in what is known as the continuity equation. The second set of equations applies the Conservation of Momentum law based entirely on Newton's Second Law. It produces a vector set of equations known as the momentum equations. The last equation applies the Conservation of Energy law from the First Law of Thermodynamics. The final resulting fluid motion equation is referred to as the energy equation.[40]

There are two general approaches for considering fluid elements: the Lagrangian and Eulerian approaches. In the Lagrangian approach, all fluid element properties are documented as if an observer were moving with the fluid elements. Using the Eulerian approach, a fixed control volume of fluid is defined. As fluid passes through the control volume, as shown in Figure 54 below, changes in the control volume fluid properties are documented. The Eulerian viewpoint is frequently used in fluid dynamics studies, and this approach will be used exclusively for this derivation.[40]

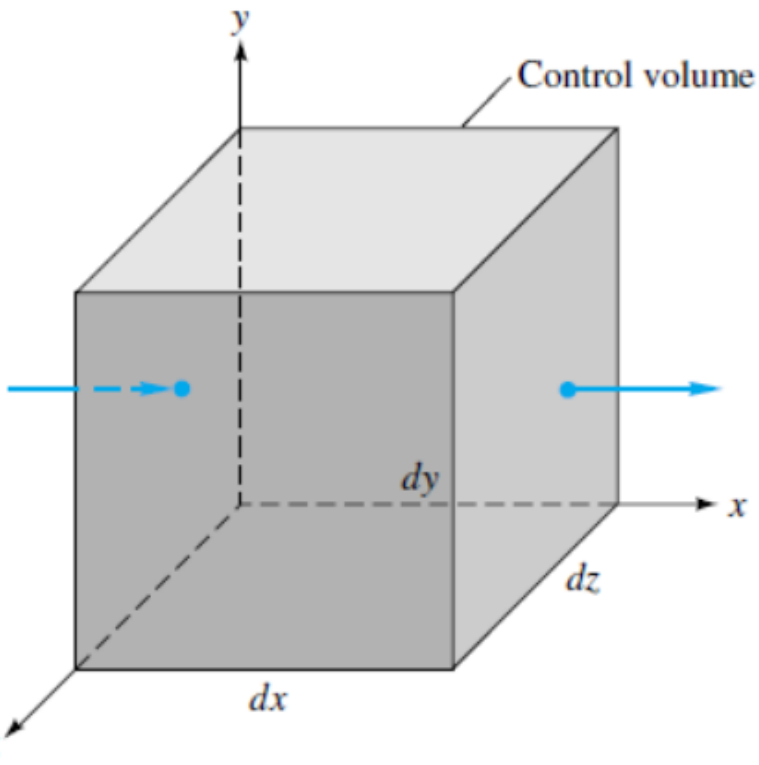

Figure 54: Elemental Cartesian fixed control volume for Eulerian approach.[41]

Using the Eulerian viewpoint with the Conservation of Mass law on an infinitesimal fixed control volume, the following continuity equation can be developed,

$$\frac{\partial \rho}{\partial t} + \vec{\nabla} \bullet \left(\rho \vec{V} \right) = 0, \tag{24}$$

where ρ is the fluid density and $\vec{V}$ is the vector form of the fluid velocity. The velocities in each of the Cartesian directions *x*, *y*, and *z* are *u*, *v*, and *w*, respectively. The first term in Equation (24) describes the rate of density increase in the control volume. The second term symbolizes the rate of mass flux moving through the control volume surface.[40]

Applying the Conservation of Momentum law to the fluid that passes through the same fixed control volume, the following momentum equation can be developed,

$$\frac{\partial(\rho\vec{V})}{\partial t}+\vec{\nabla}\bullet\left(\rho\vec{V}\vec{V}\right)=\rho\vec{f}+\vec{\nabla}\bullet\prod_{ij}, \tag{25}$$

The first term in the above equation symbolizes the rate of increase of momentum in the control volume. The second term symbolizes the rate of momentum lost by convection through the control volume walls. The first term on the right side of Equation (25) represents the body forces, $\vec{f}$, acting on the control volume. The last term on the right side is the external stresses, $\prod_{ij}$, acting on the control volume walls. The stress tensor can be expanded if the fluid is assumed to be Newtonian;

$$\begin{aligned}&\prod_{ij}=-P\delta_{ij}+\tau_{ij},\\&\tau_{ij}=\mu\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right)+\mu'\frac{\partial u_k}{\partial x_k};\ \text{for}\ i,j,k=1,2,3,\end{aligned} \tag{26}$$

where P is the surface pressure; δ_{ij} is the Kronecker delta function; μ is first coefficient of viscosity; $u_1, u_2, \text{and}\, u_3$ are the three vector components of velocity, $\vec{V}$; x_1, x_2, and x_3 are the three Cartesian components; and μ' is second coefficient of viscosity. The two viscosities are related to the bulk viscosity, κ, by the following equation,

$$\kappa=\frac{2}{3}\mu+\mu'. \tag{27}$$

Bulk viscosity is generally thought to be insignificant and is equaled to zero.[40]

Applying the Conservation of Energy law to the fluid that passes through the same fixed control volume, the following energy equation can be developed,

$$\frac{\partial e_t}{\partial t}+\vec{\nabla}\bullet e_t\vec{V}=\frac{\partial\bar{q}}{\partial t}-\vec{\nabla}\bullet\vec{q}+\rho\vec{f}\bullet\vec{V}+\vec{\nabla}\bullet\left(\prod_{ij}\bullet\vec{V}\right), \tag{28}$$

where e_t is the total energy of the control volume as shown in Equation (29),

$$e_t = \rho\left(e + \frac{\left(\overline{V}\right)^2}{2} + PE + \ldots \right), \tag{29}$$

and e is the internal energy of the control volume, $\overline{V}$ is the average internal velocity, PE is the potential energy. The first term of Equation (28) symbolizes the rate of total energy in the control volume; the second term symbolizes the rate of total energy lost by convection through the control volume walls. The first term on the right side of Equation (28) symbolizes the average rate of internal heat, $\overline{q}$, produced by external effects; the second term on the right side symbolizes the rate of heat, $\vec{q}$, lost by conduction through the control volume walls. The third term on the right side symbolizes the work done by the body forces, $\vec{f}$, on the control volume. The last term on the right side symbolizes the work done on the control volume by the surface forces. Using Fourier's law[42] for heat transfer by conduction, the heat transfer was expressed by,

$$\vec{q} = -K\vec{\nabla}T, \tag{30}$$

where K is the thermal conductivity and T is the temperature.[40]

The previous equations of continuity, momentum, and energy can be combined into a condensed vector equation. If the fluid is assumed to be Newtonian and bulk viscosity is neglected, the conservative form of the Navier-Stokes equations (NSE) without external heat addition, body forces, finite rate chemical reactions, or mass diffusion is presented in Cartesian coordinates as,

$$\frac{\partial \vec{U}}{\partial t} + \frac{\partial \vec{F}}{\partial x} + \frac{\partial \vec{G}}{\partial y} + \frac{\partial \vec{H}}{\partial z} = 0, \tag{31}$$

where the vectors $\vec{U}, \vec{F}, \vec{G}, \text{and } \vec{H}$ from Equation (31) are expressed as

$$\vec{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e_t \end{bmatrix}, \vec{F} = \begin{bmatrix} \rho u \\ \rho u^2 + P - \tau_{xx} \\ \rho uv - \tau_{xy} \\ \rho uw - \tau_{xz} \\ ue_t + uP - u\tau_{xx} - v\tau_{xy} - w\tau_{xz} + q_x \end{bmatrix},$$

$$\vec{G} = \begin{bmatrix} \rho v \\ \rho uv - \tau_{xy} \\ \rho v^2 + P - \tau_{yy} \\ \rho vw - \tau_{yz} \\ ve_t + vP - u\tau_{xy} - v\tau_{yy} - w\tau_{yz} + q_y \end{bmatrix}, \text{and} \tag{32}$$

$$\vec{H} = \begin{bmatrix} \rho w \\ \rho uw - \tau_{xw} \\ \rho vw - \tau_{yz} \\ \rho w^2 + P - \tau_{zz} \\ we_t + wP - u\tau_{xz} - v\tau_{yz} - w\tau_{zz} + q_z \end{bmatrix}.$$

The first row of Equation (32) represents the continuity equation; the second through fourth rows represent the momentum equation; and the fifth row represents the energy equation. The foundation of the whole viscous flow theory is based on the NSE. However, the description "Navier-Stokes equations" applies only to the viscous momentum equations, but it traditionally includes both the continuity and energy equations.[40]

4.1.1 Inviscid Euler Fluid Model

The Navier-Stokes equations govern most the fluid flow generally encountered. However, the computation of the solution of these equations is often difficult and time consuming. An alternative to the NSE would be to neglect the viscous effects resulting in the formulation of the inviscid Euler equations. The solution of the Euler equations is

useful early in the design phase of a project. The Euler equations describe the movement of inviscid non-heat conducting gas in most flow fields, including flows where high velocity gradients may produce shock waves. There are shock-capturing techniques used for calculating flow fields that contain shock waves, such as those present in launch vehicle exhaust flows.[40]

In order to use a shock-capturing method, the Euler equations are altered so that shock waves and other discontinuities are calculated as part of the solution using a central differences approach. Shocks captured using this method typically spread the shock wave over multiple control volume intervals causing the solution to appear smeared. The method fails when dealing with very large shock waves. This is likely due to oscillations that build in the flow field. The concept of flux-vector-splitting was developed to resolve the problems created by the central differences approach.[40]

The idea of flux-vector-splitting refers to splitting the fluxes flowing into and out of each control volume face into positive and negative components. Each component then has a single-sided upwind numerical method that is applied in order to capture the shock waves.[40] The following two-dimensional Steger-Warming[43] flux-vector-splitting derivation was originally developed for a computational exercise given by Shelton[44] in his graduate level computational fluid dynamics class. The conservative form[40] of the governing equations, shown in Equation (32), is integrated over the control volume surface, shown in Figure 54. Vectors $\vec{U}, \vec{F},$ and $\vec{G}$ are modified from Equation (32) to form a new set of equations known as the inviscid Euler equations expressed as,

$$\vec{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho e_t \end{bmatrix}, \vec{F} = \begin{bmatrix} \rho u \\ \rho u^2 + P \\ \rho uv \\ \rho u h_t \end{bmatrix}, \text{ and } \vec{G} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + P \\ \rho v h_t \end{bmatrix}, \tag{33}$$

where h_t is the total enthalpy $\left(e_t + P/\rho\right)$. The continuity equation can be integrated and reduced to a discrete form by applying Green's Theorem[45] resulting in the following form,

$$\frac{\partial \vec{U}}{\partial t} \Delta x \Delta y + \sum_{\text{cell faces}} \left(\vec{F}\, \Delta y \Delta z + \vec{G}\, \Delta x \Delta z \right) = 0, \tag{34}$$

where Δx and Δy are the cell wall lengths in the *x* and *y* directions, respectively. Recasting Equation (34) in a slightly different form utilizes Figure 55 as the basis for the indices. The new discretized equation is,

$$\left(\vec{U}_{i,j}^{\,n+1} - \vec{U}_{i,j}^{\,n} \right) \frac{\Delta x \Delta y}{\Delta t} + \left(\vec{F}_{i+1/2,j}^{\,n} - \vec{F}_{i-1/2,j}^{\,n} \right) \Delta y + \left(\vec{G}_{i,j+1/2}^{\,n} - \vec{G}_{i,j-1/2}^{\,n} \right) \Delta x = 0, \tag{35}$$

In order to deal with the flow field oscillations, the four cell faces are each divided into two separate boundaries, an inner face and an outer face. This is the heart of the Steger-Warming[43] flux vector splitting method. Consider the right most control volume face (east surface in Figure 55) located at $x = i + 1/2$. This face is split into a right outward surface and a left inward surface where the flux (fluid flow) across the surface is based on an "upwind" numerical scheme. The simplest, first order approximation, uses flux data from a control volume element centered at $x = i$ for the left face and data from a control volume element centered at $x = i + 1$ for the right face. Higher order approximations are possible utilizing more data points relying on a single-sided upwind methodology, i.e. using flux data from both the control volume elements centered at $x = i$ and centered at $x = i - 1$ for only the left face.

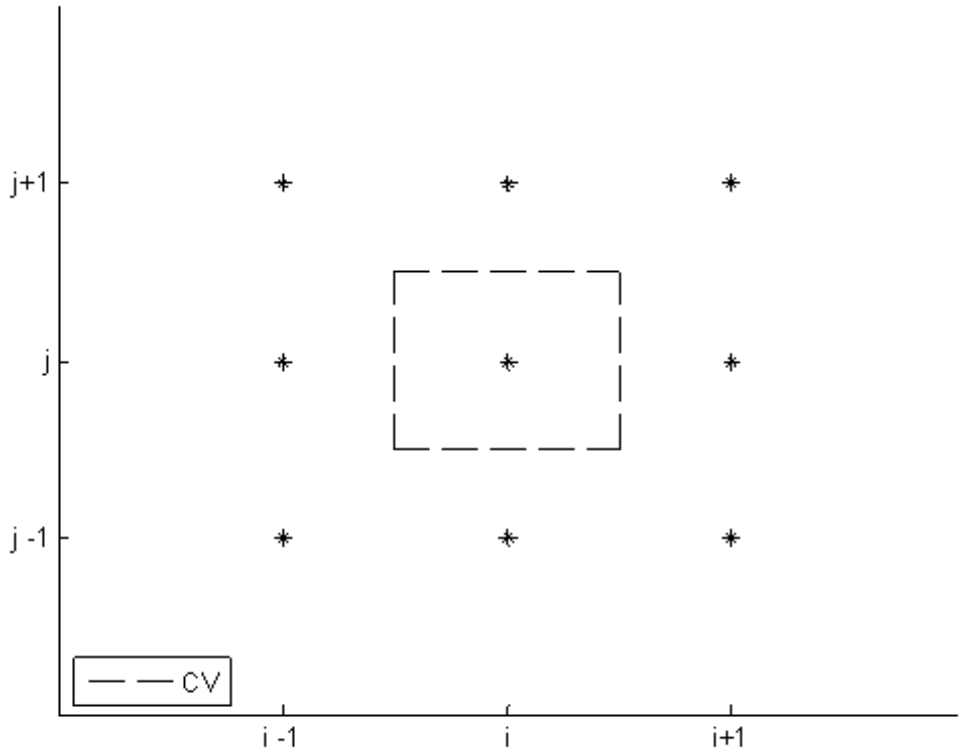

Figure 55: Two dimensional control volume (CV) discretization.

One main issue must be resolved in order to proceed with solving the numerical problem: the determination of how to decompose the flux contributions on each of the control volume surfaces into positive and negative flow directions. When the flow directions are known, the appropriate upwind method can be applied. In order to proceed from here, the two flux vectors, $\vec{F}$ and $\vec{G}$, should be reformed into a single generic vector using a linear combination of the vectors and the flux Jacobian. Starting with the linear combination of the flux vectors,

$$\vec{F}' = k_1 \vec{F} + k_2 \vec{G}, \tag{36}$$

the generic vector, $\vec{F}'$, can be rewritten using a flux Jacobian of the $\vec{U}$ vector,

$$\vec{F'} = A\vec{F}, \tag{37}$$

where $A = \partial \vec{F'} \big/ \partial U$. Using the fact that the flux Jacobian can be diagonalized by the eigenvalues and eigenvectors, the matrix *A* represents the wave speeds flowing into or out of the control volume. Under the assumption that the eigenvalues are known,

$$\lambda_1 = \lambda_2 = k_1 u + k_2 v, \lambda_3 = \lambda_1 + a\sqrt{k_1^2 + k_2^2}, \lambda_4 = \lambda_1 - a\sqrt{k_1^2 + k_2^2}, \tag{38}$$

the generic vector, $\vec{F'}$, is calculated by the following vector,

$$\vec{F'} = \frac{\rho}{2\gamma}\begin{bmatrix} 2(\gamma-1)\lambda_1^{\pm} + \lambda_3^{\pm} + \lambda_4^{\pm} \\ 2(\gamma-1)u\lambda_1^{\pm} + (u + \bar{k}_1 a)\lambda_3^{\pm} + (u - \bar{k}_1 a)\lambda_4^{\pm} \\ 2(\gamma-1)v\lambda_1^{\pm} + (v + \bar{k}_2 a)\lambda_3^{\pm} + (v - \bar{k}_2 a)\lambda_4^{\pm} \\ (\gamma-1)(u^2+v^2)\lambda_1^{\pm} + [h_t + a(\bar{k}_1 u + \bar{k}_2 v)]\lambda_3^{\pm} + [h_t + a(\bar{k}_1 u + \bar{k}_2 v)]\lambda_4^{\pm} \end{bmatrix}, \tag{39}$$

where

$$\begin{aligned} &\vec{F'} = \vec{F'^{+}} + \vec{F'^{-}}, \lambda^{\pm} = \frac{\lambda \pm |\lambda|}{2}, \bar{k}_1 = \frac{k_1}{\sqrt{k_1^2 + k_2^2}}, \\ &\bar{k}_2 = \frac{k_2}{\sqrt{k_1^2 + k_2^2}}, a = \sqrt{\frac{\gamma P}{\rho}}, \text{and } h_t = \frac{\rho e_t + P}{\rho}. \end{aligned} \tag{40}$$

One additional point about the flux across the inward or outward control volume surface is that the higher order upwind schemes are advantageous for their accuracy but can cause oscillations near discontinuities. A simple method to mitigate the oscillatory nature of the second order scheme is to combine it with the first order scheme. Equation (41) demonstrates this method regarding the east control volume surface for left (L) inner surface and right (R) outer surface,

$$\vec{U}^{L}_{i+1/2,j} = \vec{U}_{i,j} + 0.5\left(\vec{U}_{i,j} - \vec{U}_{i-1,j}\right)\phi_{i,j} \text{ and}$$
$$\vec{U}^{R}_{i+1/2,j} = \vec{U}_{i+1,j} - 0.5\left(\vec{U}_{i+2,j} - \vec{U}_{i+1,j}\right)\psi_{i+1,j} \quad (41)$$

where ϕ and ψ are 0 for first order scheme and 1 for second order scheme. More sophisticated formulations of the function $\vec{U}$ were developed in the Roe,[46] van Leer,[47] and Minmod,[48] methods.

An alternative to the Steger-Warming approach is a modified version of the advection upstream splitting method (AUSM+) by Liou.[49] Instead of including the pressure term in the flux vectors, as shown in Equation (33), the fluxes are split into velocity and pressure vectors,

$$\vec{F} = \vec{F}_V + \vec{F}_P$$
$$\vec{F} = \begin{bmatrix} \rho u \\ \rho u^2 \\ \rho uv \\ \rho u h_t \end{bmatrix} + \begin{bmatrix} 0 \\ P \\ 0 \\ 0 \end{bmatrix}. \quad (42)$$

Each new vector is then sub-divided into separate inside and outside contributions from the control volume wall. The contributions are dependent upon the direction of the velocity in the form of a polynomial equation based on the Mach number at the wall interface. Liou's formulation of AUSM+ has demonstrated improved accuracy and reliability and will be used exclusively in this computational investigation.[49]

Because the viscous effects are neglected, the Inviscid Euler formulation cannot be used to solve the turbulent behaviour in the rocket exhaust flows. However, it can be used to represent the effects of sound wave movement through the modeled environment constructed by the Navier-Stokes equations. More complex models are necessary to

describe the turbulent flows which Eldred[2] believed were the sources of the acoustic loads generated at the launch of a space vehicle.

4.1.2 Reynolds-Averaged Navier-Stokes Fluid Model

From an engineering viewpoint, interest often focuses on average or mean quantities while analyzing turbulent flows through the time averaging of the Navier-Stokes equations. These equations are also known as the Reynolds-averaged Navier-Stokes (RANS) equations. The RANS equations are formulated by splitting the primitive variables into time-averaged and fluctuating components, and then time-averaging the entire Navier-Stokes equations. The classical time-weighted averaging, by Reynolds,[50] and the mass-weighted averaging, by Favre,[51] are the two types of averaging often employed.[40]

When encountering compressible flows, Tannehill[40] states that it is more convenient to use Favre's mass-weighted averaging. The Favre-averaged velocity, $\tilde{V}_i$, takes on the following form,

$$\tilde{V}_i = \frac{1}{\bar{\rho}} \int_{t_0}^{t_0+\Delta t} \rho V_i dt, \tag{43}$$

where $\bar{\rho}$ is the mean density and V_i is the instantaneous velocity component. The mass-averaged variable, $\tilde{A}$, takes on the form,

$$\tilde{A} = \overline{\rho A} \Big/ \bar{\rho}, \tag{44}$$

where A is a dummy variable used as a place holder. When substituting $\tilde{A}$ into the fundamental equations, the primitive variable (A) takes on the form,

$$A = \tilde{A} + A'', \tag{45}$$

where A'' is a fluctuating quantity. Substituting the concept presented in Equation (45) into Equation (31) and time averaging it, the continuity equation becomes as follows,

$$\frac{\partial \bar{\rho}}{\partial t} + \frac{\partial}{\partial x_i}\left(\bar{\rho}\tilde{V}_i\right) = 0. \tag{46}$$

Next, the same concept in Equation (45) is substituted into Equation (31), time averaged, and reduced. The momentum equation becomes,

$$\frac{\partial}{\partial t}\left(\bar{\rho}\tilde{V}_i\right) + \frac{\partial}{\partial x_j}\left(\bar{\rho}\tilde{V}_i\tilde{V}_j\right) = -\frac{\partial \bar{P}}{\partial x_j} + \frac{\partial}{\partial x_j}\left(\bar{\tau}_{ij} - \overline{\rho V_i'' V_j''}\right), \tag{47}$$

where

$$\bar{\tau}_{ij} = \mu\left[\left(\frac{\partial \tilde{V}_i}{\partial x_j} + \frac{\partial \tilde{V}_j}{\partial x_i}\right) - \frac{2}{3}\delta_{ij}\frac{\partial \tilde{V}_k}{\partial x_k}\right] + \mu\left[\left(\frac{\partial \overline{V_i''}}{\partial x_j} + \frac{\partial \overline{V_j''}}{\partial x_i}\right) - \frac{2}{3}\delta_{ij}\frac{\partial \overline{V_k''}}{\partial x_k}\right]. \tag{48}$$

Lastly, the concept in Equation (45) is again substituted into Equation (31), time averaged, and reduced. The energy equation becomes,

$$\begin{aligned}\frac{\partial}{\partial t}\left(\bar{\rho}c_p\tilde{T}\right) + \frac{\partial}{\partial x_j}\left(\bar{\rho}c_p\tilde{T}\tilde{V}_j\right) &= \frac{\partial \bar{P}}{\partial t} + \tilde{V}_j\frac{\partial \bar{P}}{\partial x_j} + \overline{V_j''\frac{\partial P}{\partial x_j}} \\ &+ \frac{\partial}{\partial x_j}\left(K\frac{\partial \tilde{T}}{\partial x_j} + K\frac{\partial \overline{T''}}{\partial x_j} - c_p\overline{\rho T'' V_j''}\right) + \bar{\Phi},\end{aligned} \tag{49}$$

where

$$\bar{\Phi} = \overline{\tau_{ij}\frac{\partial V_i}{\partial x_j}} = \bar{\tau}_{ij}\frac{\partial \tilde{V}_i}{\partial x_j} + \overline{\tau_{ij}\frac{\partial V_i''}{\partial x_j}}. \tag{50}$$

The combination of the Reynolds equations are complex, and it is possible to question whether or not any progress can be made towards solving turbulent flow problems. The Favre mass-averaged Navier-Stokes equations cannot be solved because the exact

correlation between two Favre fluctuating variables is not known. To proceed, additional equations or assumptions must be formulated to address the relation between the new turbulent quantities and the time mean variables.[40]

Based upon the number of equations that are used to estimate the unknown terms, the Reynolds-averaged Navier-Stokes closure models[40] are divided into four groups. These four categories include algebraic models, one equation models, two equation models, and second order closure models. The most popular of all the RANS closure models are the two equation models, $k-\varepsilon$ and $k-\omega$.[52] Both models rely on $-\overline{\rho V_i'' V_j''}$ from the RANS equations,

$$-\overline{\rho V_i'' V_j''} = 2\mu_T S_{ij} - \frac{2}{3}\delta_{ij}\left(\mu_T \frac{\partial V_k}{\partial x_k} + \rho\bar{k}\right) \text{ and}$$
$$S_{ij} = \frac{1}{2}\left(\frac{\partial V_i}{\partial x_j} + \frac{\partial V_j}{\partial x_i}\right), \tag{51}$$

where μ_T is the turbulent viscosity and $\bar{k}$ is the kinetic energy of turbulence. The turbulent viscosity can be modeled as,

$$\mu_T = \rho V_T L_T, \tag{52}$$

where V_T and L_T are characteristic velocity and length scales of turbulence. The turbulence length scale can be defined by either,

$$L_T = C_D \frac{\bar{k}^{2/3}}{\varepsilon} \text{ or } L_T = C_D \frac{\bar{k}^{1/2}}{\omega}, \tag{53}$$

where C_D is approximately 0.164. The solution to these equations requires additional partial differential equations each adding more coefficients.[40] Both the $\bar{k}-\varepsilon$ and $\bar{k}-\omega$ equations have their advantages and disadvantages. Jones and Launder[53] originally

developed the $\bar{k}-\varepsilon$ model, but the model does not account for buoyancy and compressibility effects without modification. Wilcox[54] assembled a comprehensive book on an alternative model to address the limitations of the $\bar{k}-\varepsilon$ model. The alternative model was the $\bar{k}-\omega$, which improved computational performance without the need for a wall distance calculation. However, the $\bar{k}-\omega$ model is sensitive to the free stream and inlet conditions, an issue that is not seen in the $\bar{k}-\varepsilon$ model. Menter[55] recognized that the two models, $\bar{k}-\varepsilon$ and $\bar{k}-\omega$, were similar. He showed that ε from the $\bar{k}-\varepsilon$ model could be altered with some variable substitutions to result in a similar form of ω in the $\bar{k}-\omega$ model. The approach essentially combined the two models $\bar{k}-\varepsilon$ for the far-field and $\bar{k}-\omega$ for the near wall region. Menter's new method is known as the shear stress transport (SST) $\bar{k}-\omega$ model.[56]

The Reynolds-averaged Navier-Stokes equations are similar to the inviscid Euler equations in that they can model the effects of a concept but not the actual source of that concept. RANS does not resolve the turbulent eddies but only models the effect the turbulent eddies have on the flow; on the other hand, large eddy simulation (LES) is able to resolve the larger eddies and model the smaller ones. The resolved eddies are solved without any kind of approximations in the Navier-Stokes equations. The LES is computationally expensive but is more accurate than the RANS method. The biggest problem concerning the LES is that if it contains walls in the domain, then a very fine mesh is needed in the boundary layer. A way to overcome the LES disadvantages is to combine it with a RANS model near the wall. The new hybrid RANS/LES model is the detached eddy simulation (DES).[52]

4.1.3 Large Eddy Simulation and Detached Eddy Simulation

In the large eddy simulation, larger turbulent eddies within the flow are computed directly and the effects of the smaller turbulent eddies are modeled. The two major steps in LES are filtering and sub-grid modeling. The so called filtering step is a type of space averaging of the flow variables similar to Equation (45) in that,

$$V_i = \hat{V}_i + V_i' . \tag{54}$$

The filtered variable, $\hat{V}_i$, is defined by the convolution integral,

$$\hat{V}_i(x_1, x_2, x_3) = \iiint_D \left[\prod_{j=1}^{3} G_j(x_j, x_j') \right] V_i(x_1', x_2', x_3') dx_1' dx_2' dx_3' . \tag{55}$$

Although several functions have been proposed by Aldama[57] for the filter, *G*, the volume-averaging box filter is the most frequently used.[40] If LES is applied to a compressible flow, the Favre averaging method must be applied together with the filtering; otherwise, the new Navier-Stokes equations would include new products of the primitive variables. After filtering of the Navier-Stokes equations, a new set of equations provide a resolved flow. The continuity equation becomes,

$$\frac{\partial \overline{\rho}}{\partial t} + \frac{\partial}{\partial x_i}\left(\overline{\rho}\hat{V}_i\right) = 0 . \tag{56}$$

The momentum equation becomes,

$$\frac{\partial}{\partial t}\left(\overline{\rho}\hat{V}_i\right) + \frac{\partial}{\partial x_j}\left(\overline{\rho}\hat{V}_i\hat{V}_j\right) + \frac{\partial \overline{P}}{\partial x_j} - \frac{\partial \hat{\sigma}_{ij}}{\partial x_j} = -\frac{\partial \tau_{ij}^{SF}}{\partial x_j} + \frac{\partial}{\partial x_j}\left(\overline{\sigma}_{ij} - \hat{\sigma}_{ij}\right). \tag{57}$$

The energy equation becomes,

$$\frac{\partial}{\partial t}\left(\overline{\rho}\hat{e}_t\right) + \frac{\partial}{\partial x_j}\left(\overline{\rho}\hat{V}_j\hat{e}_t\right) + \frac{\partial \hat{q}}{\partial x_j} + \overline{P}\hat{S}_{kk} - \hat{\sigma}_{ij}\hat{S}_{ij} = -A - B - C + D, \tag{58}$$

where the sub-grid scale (SGS) terms are,

$$
\begin{aligned}
A &= \frac{\partial}{\partial x_j}\left(\overline{\rho}\left(\overline{V_j e_t} - \hat{V}_j \hat{e}_t\right)\right) \text{ is the divergence of SGS heat flux,} \\
B &= \frac{\partial}{\partial x_j}\left(\overline{q}_j - \hat{q}_j\right) \text{ is the divergence of SGS heat diffusion,} \\
C &= \left(\overline{PS_{kk}} - \overline{P}\hat{S}_{kk}\right) \text{ is the SGS pressure dilatation, and} \\
D &= \left(\overline{\sigma_{ij}S_{ij}} - \hat{\sigma}_{ij}\hat{S}_{ij}\right) \text{ is the SGS viscous dissipation,}
\end{aligned}
\tag{59}
$$

which also includes,

$$
\begin{aligned}
\overline{\sigma}_{ij} &= \overline{2\mu S_{ij}} + \overline{\left(\mu_B - \frac{2\mu}{3}\right)\delta_{ij}S_{kk}}, \\
\hat{\sigma}_{ij} &= 2\hat{\mu}\hat{S}_{ij} + \left(\hat{\mu}_B - \frac{2\hat{\mu}}{3}\right)\delta_{ij}\hat{S}_{kk}, \\
S_{ij} &= \frac{1}{2}\overline{\left(\frac{\partial \hat{V}_i}{\partial x_j} + \frac{\partial \hat{V}_j}{\partial x_i}\right)}, \text{ and} \\
\overline{q}_j &= -K\frac{\partial \overline{T}}{\partial x_j}, -\hat{K}\frac{\partial \hat{T}}{\partial x_j}.
\end{aligned}
\tag{60}
$$

The introduction of the sub-grid scale terms in the previous equations brings about the second major step in a LES, which is sub-grid modeling.[58]

The purpose of a sub-grid scale model is to describe the energy transfer between the large eddies and the sub-grid scales. The energy, on the average, is transported from the large turbulent and cascaded small ones. Therefore, a SGS model has to provide a method for sufficient energy to be dissipated. There are some instances where the energy may flow from the small eddies to large ones. The sub-grid scale model should account for this energy dissipation. Various SGS models have been proposed over the years, and research still continues on the validity of the models.[58]

The detached eddy simulation is a hybrid approach that combines the characteristics of the Reynolds-averaged Navier-Stokes model in some parts of the flow and the large eddy simulation in others parts of the flow. The DES turbulence model is calculated so that the shear layers are resolved using a RANS model. The turbulence model is fundamentally modified so that, if the computational grid is fine enough, the need for a sub-grid scale model will be mitigated. By using this combination, the simulation includes the advantages of both the RANS in the steady shear layers and the LES in the unsteady turbulent domain. Even though the methodology has potential for certain simulations, it must be understood that a DES does not answer all turbulent modeling issues. The building of an acceptable computational grid is somewhat of an art form.[56] This issue will be discussed in Chapter 5.

4.2 Acoustic Model Formulation

This research is primarily interested in the study of the frequency spectra from the sound generated aerodynamically. Lighthill[14] stated that previous experiments showed that the frequencies within the flow dynamics are identical to those of the radiating sound field. The investigators, whose previous experiments Lighthill examined, were primarily interested in trying to relate the flow frequencies with basis flow constants. The prevailing theory during Lighthill's studies was that the generated frequencies were caused by flow instability. Experiments conducted showed that the flow instability was amplified by the acoustic excitation at the exit of a nozzle. The fluid flow was shown to be sensitive to external sources of sound. However, the general set of equations to estimate the sound power had not been defined at that time even by the physicists, who developed the

foundation for the science of acoustics. Lighthill's fundamental problem was to determine the mechanism that converted the kinetic energy in the fluctuating shearing flow into the acoustic energy in the sound waves.[14]

Lighthill's theoretical approach was to divide a fluid region into two defined parts. The first region was a small area of fluctuating fluid; the second region was the remainder of the domain, which remained stationary and followed ordinary laws of acoustics. At this point, the principal equations for the density fluctuations in the smaller region could be compared to those of the larger region, which convey sound. The Reynolds momentum equation presented in Section 4.1, Equation (25), states that the momentum in a smaller fixed region of space changes at a rate precisely the same as if the gas were at rest by the combined action of the constant stresses, *P*, and the fluctuating stresses, $\rho\vec{V}\vec{V}$. The larger homogeneous acoustic medium experiences stress from a simple hydrostatic pressure, which varies in proportion to the density. The proportionality constant is the square of the speed of sound, a_o^2. Therefore, the equations that transmit sound in the fluid at rest from the applied fluctuating stresses are,

$$\begin{aligned} &\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i}\left(\rho V_i\right) = 0, \\ &\frac{\partial}{\partial t}\left(\rho V_i\right) + a_o^2 \frac{\partial \rho}{\partial x_i} = -\frac{\partial \Theta_{ij}}{\partial x_i}, \text{ and} \\ &\frac{\partial^2 \rho}{\partial t^2} - a_o^2 \nabla^2 \rho = \frac{\partial^2 \Theta_{ij}}{\partial x_i \partial x_j}, \end{aligned} \tag{61}$$

where the applied instantaneous stress is,

$$\Theta_{ij} = \rho V_i V_j + P_{ij} - a_o^2 \rho \delta_{ij}\,. \tag{62}$$

The above equations are the basic equations for sound created aerodynamically.[14]

4.2.1 Proudman Acoustic Model

Lighthill was working closely with several experimentalists to formulate his theory on aerodynamic sound generation. Proudman[20] studied a special case where the turbulent region had isotropic turbulence. His underlying assumption was for a turbulent flow with a high Reynolds number but a low Mach number. Proudman believed that the noise generated by the turbulent eddies contributed to the viscous dissipation rate of kinetic energy, but he stated that the contribution was negligible. At great distances from the turbulence, Proudman was able to show that sound intensity is due to a volume distribution of simple acoustic sources. In his formulation, the entire fluid region was assumed to be uniform and stationary similar to Lighthill's theory. Proudman presented the acoustic power output, W_m, per turbulent fluid mass. The approximate formula,

$$W_m = -\frac{3}{2}\alpha \frac{d\overline{V^2}}{dt}\left(\frac{\overline{V^2}}{a_o^2}\right)^{5/2}, \tag{63}$$

where α is a numerical constant, $\overline{V^2}$ is the average square fluctuating velocity, *t* is the time, and a_o is the velocity of sound in the fluid. The constant α is expressed in terms of a velocity correlation function. The numerical value of α is approximately 38. The overall noise level that can be estimated from the turbulent shear layers does not propagate outside the turbulence. A method that has the ability to provide sound propagation into the far field outside the computational grid is known as the Ffowcs Williams-Hawkings method.[20]

4.2.2 Ffowcs Williams – Hawkings Acoustic Model

The theory for sound generated by aerodynamic flows is a comprehensive theory developed by Lighthill. It provides a firm foundation for understanding sound generated by fluctuating airflows when vibrating solids are not present. Lighthill's acoustic theory can be used to relate the non-linear turbulent motion to induced acoustic radiation.[17] Ffowcs Williams and Hawkings[21] used Lighthill's theory and reformulated the continuity and momentum equations for a moving turbulent region bounded by a defined surface,

$$\frac{\partial \overline{\rho}}{\partial t} + \frac{\partial}{\partial x_i}\left(\overline{\rho V_i}\right) = \rho_o V_i \delta(f)\frac{\partial f}{\partial x_i} \text{ and}$$
$$\frac{\partial}{\partial t}\left(\overline{\rho V_i}\right) + \frac{\partial}{\partial x_i}\left(\overline{\rho V_i V_j} + \overline{P_{ij}}\right) = P_{ij}\delta(f)\frac{\partial f}{\partial x_i}, \tag{64}$$

where $f = 0$ defines a surface between the two Lighthill regions, $\delta(f)$ is a one-dimensional delta function which is zero everywhere except at f equals zero. The term ρ_o is the density of the larger unbounded non-moving fluid region. By eliminating $\overline{\rho V_i}$ from Equation (64), the wave equation was formulated,

$$\frac{\partial^2}{\partial t^2}\left(\overline{\rho - \rho_o}\right) - a_o^2 \frac{\partial^2}{\partial x_i^2}\left(\overline{\rho - \rho_o}\right) = \frac{\partial^2 \overline{\Theta_{ij}}}{\partial x_i \partial x_j}$$
$$- \frac{\partial}{\partial x_i}\left(P_{ij}\delta(f)\frac{\partial f}{\partial x_j}\right) + \frac{\partial}{\partial t}\left(\rho_o V_i \delta(f)\frac{\partial f}{\partial x_i}\right). \tag{65}$$

The dependent variable is the density function, $\overline{\rho - \rho_o}$, which is a measure of the sound amplitude.[21] When comparing the theoretical approach with experimental data, the techniques developed by Lighthill are successful at low speeds and provides a theoretical basis for the study of rocket noise.[17]

5 Computational Fluid Dynamic Model Evaluation

Chapter 5 contains the basis for quantifying the broadband noise that is generated by turbulent flows. In general, obtaining the necessary pressure field fluctuations from computational fluid dynamics (CFD) in order to make the noise predictions is an extremely difficult task.[59] In order to begin the computational approach, a simple pure-tone one-dimensional sound wave traveling through a three-dimensional free space was modeled using the inviscid Euler equations. The spatial and temporal resolutions were adjusted until the computational errors were minimized. This allowed for the building of a simple turbulent flow over a cylinder to predict the noise generated using the detached eddy simulation equations. Because of the computational size of the model for the flow over a cylinder and the abundance of the research results generated, the model provided a basis for building more complex supersonic jet flow fields. The flow characteristics found in a supersonic jet flow field are similar to those in the exhaust flow of a rocket during the launch of a space vehicle. A basic overview of the experiments performed is shown in Table 11 below and will be discussed in the appropriate sections of this chapter.

Two primary computational systems were utilized during this research project. The first system was equipped with dual Intel Xeon X5690 processors. Each processor had six physical cores (12 logical cores) running at a maximum speed of 3.7 GHz. The system

Table 11: Overview of computational sound field insight for the fluid / acoustic model discussed in Chapter 5.

Simulation Time Assessment	Steady State	Transient			
Fluid Model	RANS	Inviscid Euler	Unsteady RANS	DES	DES
Acoustic Model	Proudman	None	FW-H	FW-H	CAA
CFD Experiment # (Exp. Location)	25, 114 (Ch. 5.2); 134, 137 (Ch. 5.3)	68-103 (Ch. 5.1)	137 (Ch. 5.3)	25, 114, 115 (Ch. 5.2); 134, 137 (Ch. 5.3)	25 (Ch. 5.2)
Sound Field Insight	Local sound power levels with no frequency content	Sound wave temporal and spatial resolution requirement for wave propagation	Absolutely no insight into the sound field!	Reasonable local and far field sound pressure levels with frequency content	Local sound pressure levels with frequency content
Simulation Notes	Time to complete simulation is reasonable with limited computational storage and resource requirements	Time to complete simulation is fast with limited computational storage and resource requirements	Time to complete simulation is long with limited computational storage and resource requirements	Time to complete simulation is extremely long with limited computational storage and elevated resource requirements	Time to complete simulation is extremely long with large computational storage and elevated resource requirements

also had 96 GB of DDR3-1333 computer memory installed. The operating system was Microsoft Windows 7 Pro. The second system was a cluster equipped with 512 Intel Xeon X5560 processors. Each processor had four physical cores running at a maximum speed of 2.8 GHz. The system could utilize 1536 GB (24 GB per CPU) of DDR3-1333 distributed computer memory. The cluster was based on a Linux operating system. The estimates for the time to complete several computational experiments will be discussed further in the appropriate sections. The CFD software used was STAR-CCM+ (Version 7.02.008).

The task of evaluating the computational models was broken down into three sections: Section 5.1 Sound Wave in Free Space Noise Predictions, Section 5.2 Flow Over a Cylinder Noise Predictions, and Section 5.3 Supersonic Jet Wind Tunnel Noise Predictions.

5.1 Sound Wave in Free Space Noise Predictions

The initial effort to predict the noise generated in turbulent flows was fraught with a host of computational issues, such as dissipation (amplitude error), dispersion (phase error), boundary reflection, etc. It was extremely difficult to quantify the noise generated by turbulent flows because the energy in the sound field is only a small fraction of the energy in the primary flow. For example, a 50 decibel (dB) pure-tone sound wave has fluctuating amplitude of only 0.009 Pa in an environment with a standard atmospheric pressure of 101,325 Pa! The noise generated by turbulent flows is inherently unsteady and requires temporal and spatial resolutions for the acoustics that are an order of magnitude smaller than the time and space resolutions required for the turbulent eddies.

Temporal and spatial resolutions must be extremely fine in order to resolve the fluctuating waves that are generated by a turbulent flow; but computational resources have finite limitations. Generally, the energy in a sound wave spans across multiple discreet grid cells within a computational environment. As a sound wave propagates, the energy in the discreet cells will also fluctuate. A simple question must be asked then, "What are the required temporal and spatial resolutions?" It was assumed in this study that 10 discreet points would be sufficient to define one period of a sine wave. The computational software used exclusively in this research, STAR-CCM+, explicitly states in the User Manual[56] to

use 10 points per wave period for the temporal resolution. The User Manual also advises using approximately 58 to 145 points per wavelength for the spatial resolution. An example of a discreet 10-point sine wave is shown in Figure 56 along with a corresponding discreet 100-point sine wave. Obviously, a discreet 10-point sine wave is not sufficient to represent a continuous sine wave accurately. Shur, Spalart, Strelets, and Travin[60] suggested that a spatial resolution of four cells per wavelength (CPWL) and a temporal resolution of 50 time steps per wave period (TSPWP) should be used. Fukuda et al.[61] suggested that a spatial resolution of 60 CPWL and a temporal resolution of 60 TSPWP should be used. Reinelt[62] states that a spatial resolution of 10-15 CPWL and a temporal resolution of one TSPWP should be used. Wagner, Huttl, and Sagaut[63] states that for a second-order, centeral, finite difference scheme that a spatial resolution of 8-100 CPWL should be used. They state that an approximate order of accurate within 10% by using 8 CPWL, approximate accurate within 1% by using 25 CPWL, approximate accurate within 0.1% by using 100 CPWL. Further investigation is warranted into the sufficient temporal and spatial resolutions that are needed to model the propagation of sound waves.

One period of a pure-tone sound wave was placed within an inviscid Euler computational environment and allowed to propagate. A sound wave in this ideal gas inviscid Euler computational environment should not lose any energy, and it should propagate at the standard speed of sound (a_o=347.3 m/s). The constructed wave was similar to a one period sine wave, except that the ends were smoother by utilizing the following exponential equation,

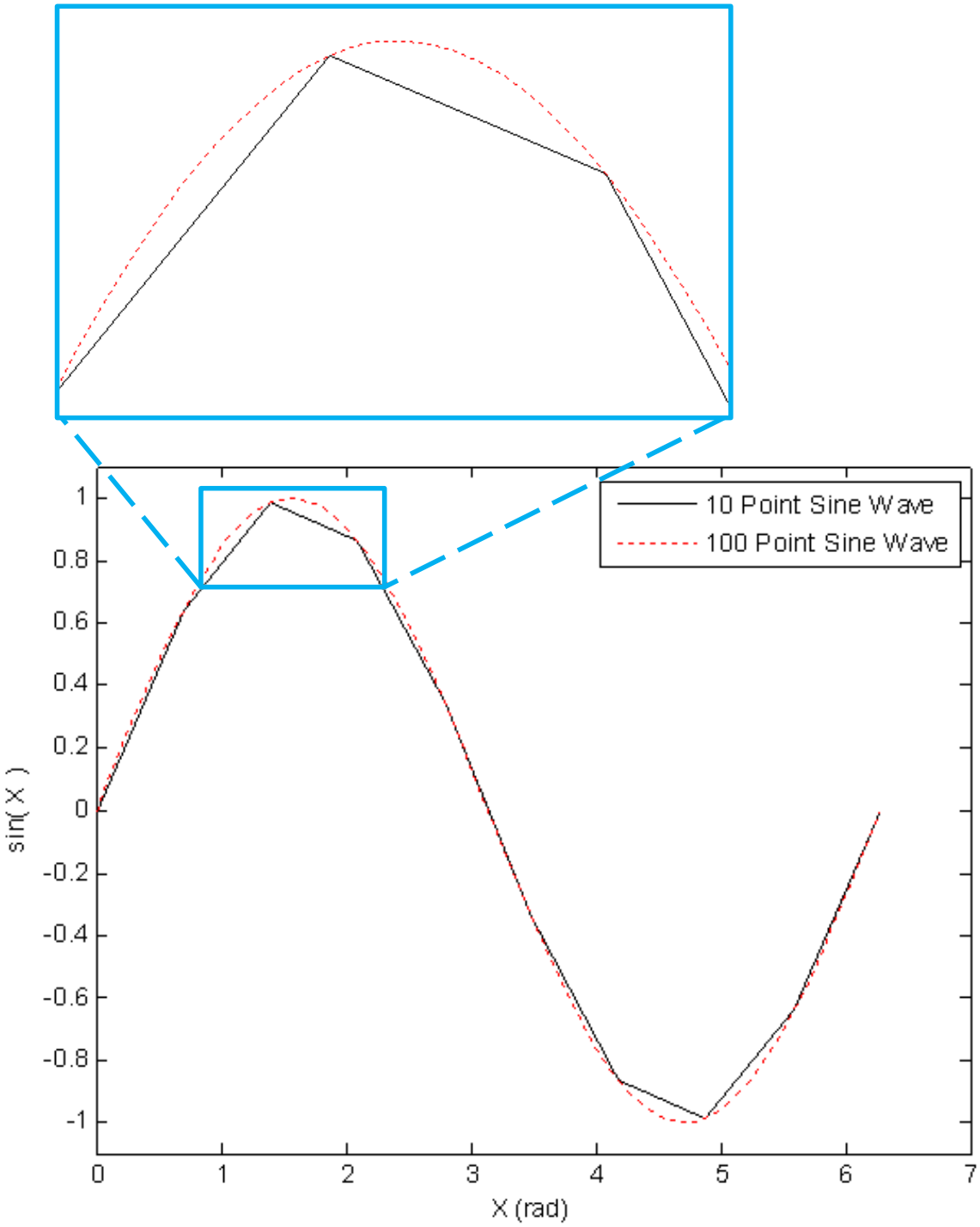

Figure 56: Comparison of a discreet 10-point sine wave with a discreet 100-point sine wave.

$$P = -C_1 x \exp\left(-\left(\frac{x}{C_2}\right)^2\right)\rho\, a_o\,, \tag{66}$$

where P is the fluctuating pressure, x is the coordinate in the direction of the propagation axis, and ρ is the density of the fluid. The reason behind using the exponential wave lies in the desire to avoid the sharp sine wave end gradients that would introduce possible errors. The two constants, C_1 and C_2, were adjusted so that the exponential wave approximates three separate 50 dB pure-tone waves, 10,000 Hz, 2,000 Hz, and 500 Hz, as shown in Table 12. A comparison of a 50 dB, 10,000 Hz, sound wave and an exponential wave is shown in Figure 57. It is also apparent that the smooth head and tail of the exponential wave is clearly visible. The decision to utilize only one wave period provided a method to track the propagation of the wave front through the computational environment. The computational environment is a computational grid with finite resolution, which represents a free-field. One computational grid, which has a grid

Table 12: Exponential equation constants for a 50 dB sound wave approximation.

Wave (Hz)	C_1	C_2
10,000	4.1050E-03	0.01243
2,000	8.3650E-04	0.061
500	2.0830E-04	0.245

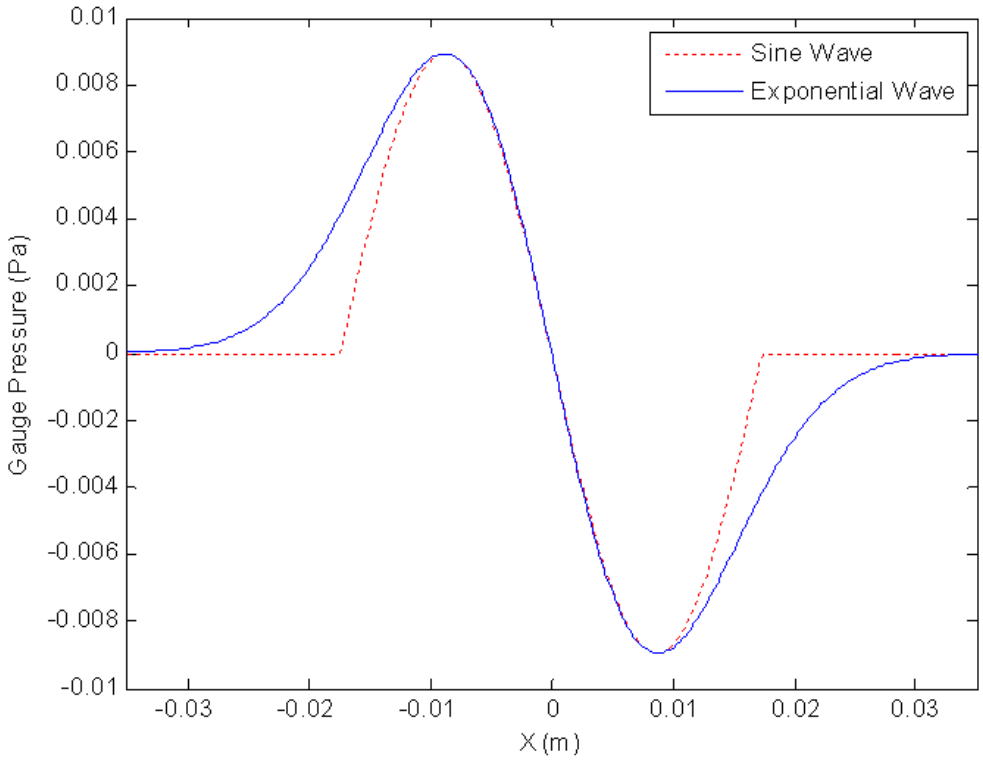

Figure 57: Comparison of a sine wave with an exponential wave.

spacing of 0.25 mm, is shown in Figure 58. Within the environment, monitoring points were positioned along the grid to study the wave movement. The seven monitoring points were positioned at -99.9 mm, -50 mm, -25 mm, 0 mm, 25 mm, 50 mm, and 99.9 mm for the 0.25 mm grid spacing. The wave was made to propagate out of one boundary and to enter into the opposite boundary by utilizing periodic computational boundaries. The wave could travel an infinite number of loops in order to monitor the slow numerical decay in pressure amplitude as the wave progressed in the environment. Figure 59 shows an example of approximately one and a half loops with the 0.25 mm grid mesh. There is a small measurable drop in pressure amplitude between the time histories at the monitoring point.

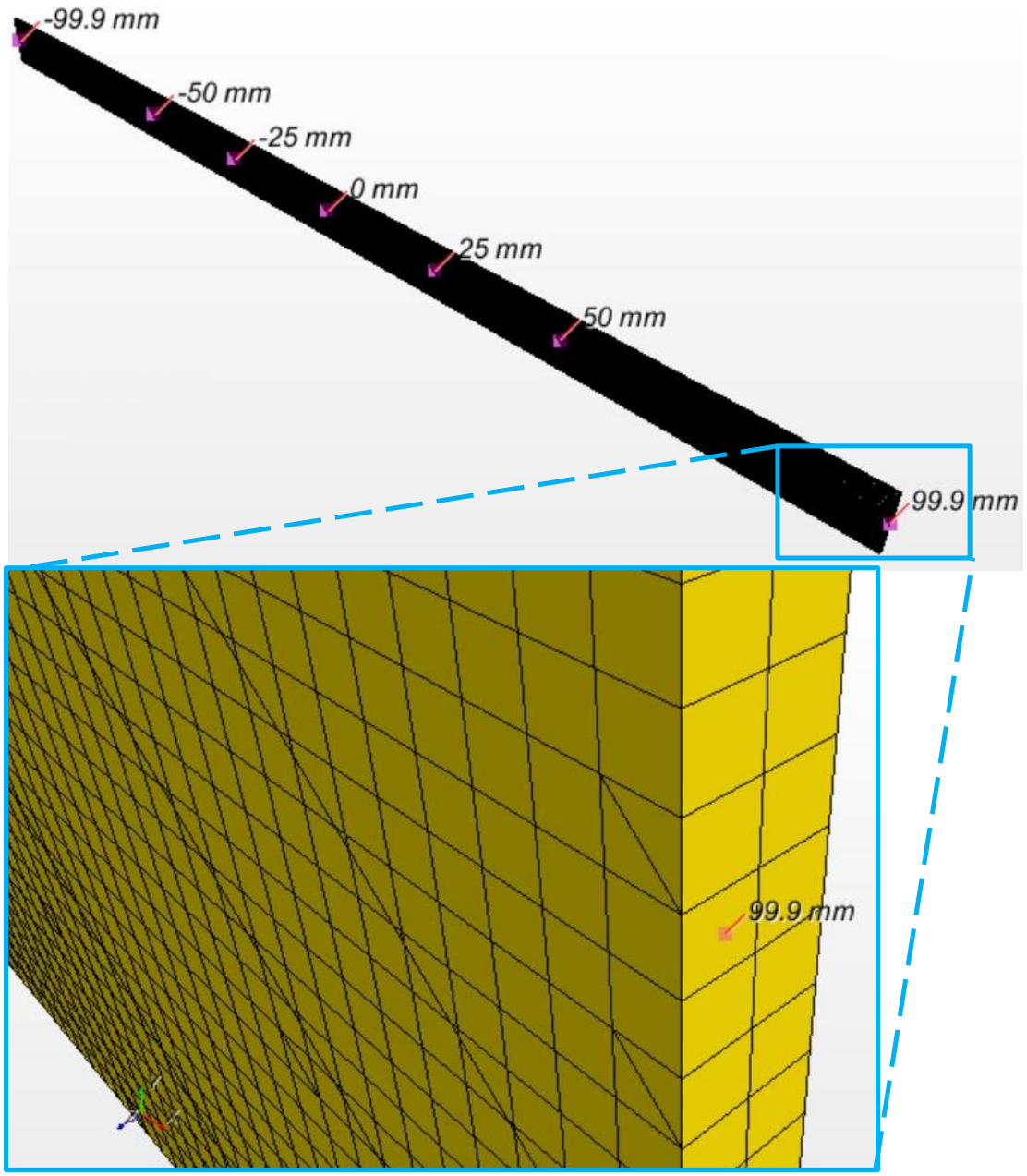

Figure 58: Computational grid for sound wave in free space CFD investigation with mesh spacing of 0.25 mm and monitoring points.

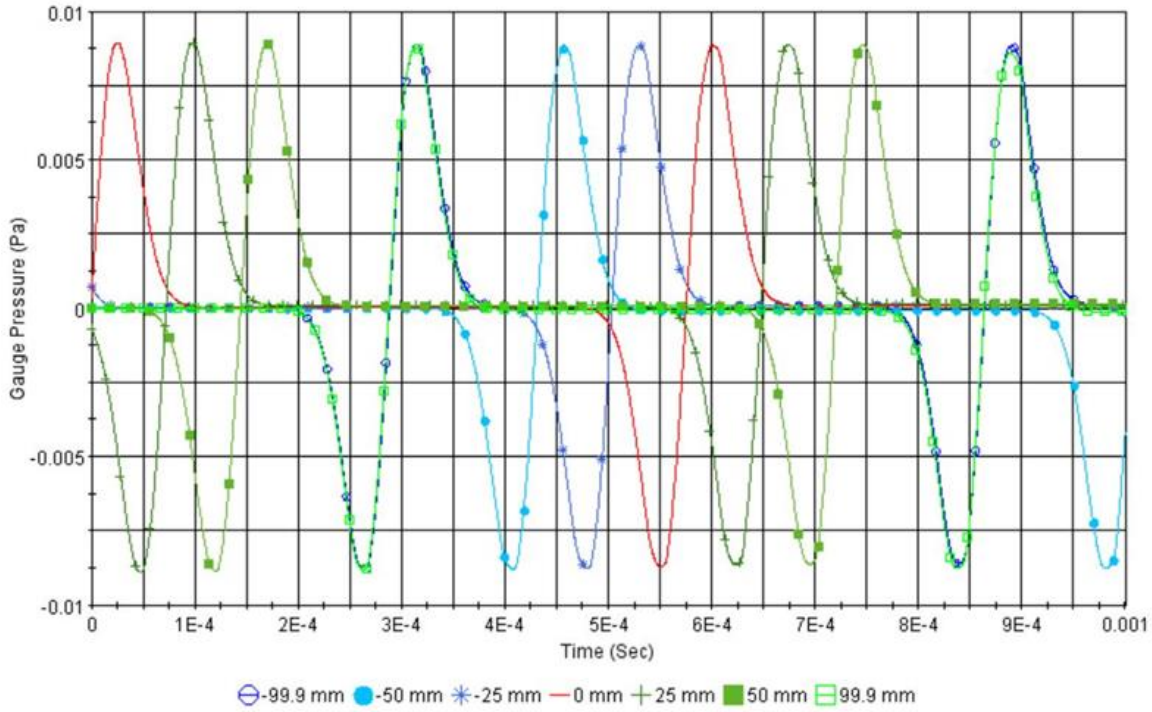

Figure 59: Gauge pressure recorded by monitor points as the sound wave progressed in the computational environment.

The non-marker solid line, in Figure 59, is the primary monitoring point under consideration, which is stationed at the 0 mm position. The exponential wave amplitude only decreased by 0.94% after traveling an arbitrary distance of 1.009 m. The small amplitude decrease was achieved with a temporal resolution of 7.13×10^{-7} second (s), which was approximately 140 time steps per sine wave's period. The spatial resolution was 0.25 mm, which used approximately 139 grid cells during the wavelength of the sine wave. The information presented in Figure 59 was for a sine wave at 10,000 Hz and a level of 50 dB as in the CFD Experiment 69. Additional CFD experiments were performed with varying temporal and spatial resolutions that minimized amplitude (dissipation) and propagation (dispersion) errors. The data are compiled in Tables 13 through 15. Table 13 is for a 10,000 Hz, 50 dB sound wave; Table 14 is for a 2,000 Hz, 50 dB sound wave; and Table 15 is for a 500 Hz, 50 dB sound wave. Several patterns emerged from the tabulated

data. First, there is little variation in either amplitude or propagation error as the temporal resolution is varied for any given spatial resolution. Second, spatial resolution is the predominant factor in minimizing the amplitude and propagation errors. Lastly, a spatial resolution of at least 100 cells per wavelength (up to the frequency of interest) is found to be necessary to reduce amplitude error to less than two percent. A corresponding temporal resolution of 100 time steps per wave period is suggested. The high-lighted rows in Tables 13 through 15 are for a spatial resolution of 100 CPWL and temporal resolution of 100 TSPWP.

One additional table was constructed for the CFD experiment in order to allow an exponential wave to propagate further than one meter. CFD Experiment 68 was for a fine spatial resolution of 139 CPWL and a fine temporal resolution of 140 TSPWP. Although the resolution had a fine grid, the overall computational grid was only 64,000 cells. Each time step only required approximately 0.38 seconds for an estimated total time of 28 hours for the experiment. Table 16 shows the nonlinear decrease in wave amplitude for various propagation distances. The sound pressure level error of the wave steadily increased, but the speed of propagation error steadily decreased. After the wave traveled 19.6 m, the wave amplitude was reduced by 3.58%, and the speed of propagation had a discrepancy of only 0.07%.

Table 13: SPL predictions of a 10,000 Hz sound wave in free space.

Experiment Dimensions (mm)		200 x 10 x 0.5 (Length x Height x Width)			**10,000 Hz**, ~50 dB		Wave Length (m)	3.473E-02	Wave Period (sec)	1.000E-04			
	Exp. #	Grid (mm)	CFL	Cells per wave length	Time Step (sec)	Steps per wave period	Exp. #	Travel (m)	Time (sec)	Amplitude (m/sec)	SPL (dB)	SPL (% err)	Speed (m/s)	Speed (% err)
Fine Grid	69	0.25	0.99	138.93	7.13E-07	140.33	69	1.009	2.90E-03	2.07E-05	49.525	0.94%	347.48	-0.05%
	99	**0.35**	**0.9923318**	**99.23**	**1.00E-06**	**100.00**	**99**	**1.009**	**2.90E-03**	**1.99E-05**	**49.154**	**1.68%**	**347.44**	**-0.04%**
	70	0.5	0.99	69.46	1.43E-06	70.16	70	1.009	2.90E-03	1.87E-05	48.642	2.71%	347.33	-0.01%
	71	0.5	0.495	69.46	7.13E-07	140.33	71	1.009	2.90E-03	1.88E-05	48.683	2.63%	347.33	-0.01%
Medium Grid	72	1	0.99	34.73	2.85E-06	35.08	72	1.009	2.91E-03	1.46E-05	46.490	7.01%	346.65	0.19%
	73	1	0.495	34.73	1.43E-06	70.16	73	1.009	2.91E-03	1.44E-05	46.349	7.30%	346.68	0.18%
	74	1	0.2475	34.73	7.13E-07	140.33	74	1.009	2.91E-03	1.43E-05	46.290	7.41%	346.65	0.19%
Coarse Grid	75	3.5	0.99	9.92	9.98E-06	10.02	75	1.009	2.94E-03	4.86E-06	36.923	26.15%	342.83	1.29%
	76	3.5	0.28285714	9.92	2.85E-06	35.08	76	1.009	2.94E-03	4.84E-06	36.897	26.20%	342.90	1.27%
	77	3.5	0.14142857	9.92	1.43E-06	70.16	77	1.009	2.94E-03	4.84E-06	36.893	26.21%	342.90	1.27%
	78	3.5	0.07071429	9.92	7.13E-07	140.33	78	1.009	2.94E-03	4.84E-06	36.890	26.21%	342.91	1.27%

Table 14: SPL predictions of a 2,000 Hz sound wave in free space.

Experiment Dimensions (mm)		1000 x 70 x 1 (Length x Height x Width)			**2000 Hz**, ~50 dB		Wave Length (m)	1.737E-01	Wave Period (sec)	5.000E-04			
	Exp. #	Grid (mm)	CFL	Cells per wave length	Time Step (sec)	Steps per wave period	Exp. #	Travel (m)	Time (sec)	Amplitude (m/sec)	SPL (dB)	SPL (% err)	Speed (m/s)	Speed (% err)
Fine Grid	100	0.35	0.9923318	496.17	1.00E-06	500.00	100	1.043	3.00E-03	2.18E-05	49.963	0.07%	347.57	-0.07%
	79	1.25	0.99	138.93	3.56E-06	140.33	79	1.043	3.01E-03	2.14E-05	49.802	0.39%	346.64	0.20%
	102	**1.75**	**0.9923318**	**99.23**	**5.00E-06**	**100.00**	**102**	**1.043**	**3.01E-03**	**2.11E-05**	**49.679**	**0.64%**	**346.98**	**0.10%**
	80	2.5	0.9	69.46	6.48E-06	77.18	80	1.043	3.02E-03	2.06E-05	49.459	1.08%	345.99	0.38%
	81	2.5	0.495	69.46	3.56E-06	140.33	81	1.043	3.01E-03	2.06E-05	49.457	1.08%	346.68	0.18%
Medium Grid	82	5	0.9	34.73	1.30E-05	38.59	82	1.043	3.02E-03	2.06E-05	49.454	1.09%	345.78	0.44%
	83	5	0.495	34.73	7.13E-06	70.16	83	1.043	3.03E-03	1.93E-05	48.896	2.20%	344.78	0.73%
	84	5	0.2475	34.73	3.56E-06	140.33	84	1.043	3.03E-03	1.93E-05	48.897	2.20%	344.78	0.73%
Coarse Grid	85	17.5	0.99	9.92	4.99E-05	10.02	85	1.043	3.10E-03	1.14E-05	44.332	11.33%	336.71	3.05%
	86	17.5	0.28285714	9.92	1.43E-05	35.08	86	1.043	3.10E-03	1.14E-05	44.329	11.34%	336.85	3.01%
	87	17.5	0.14142857	9.92	7.13E-06	70.16	87	1.043	3.10E-03	1.12E-05	44.180	11.64%	336.45	3.13%
	88	17.5	0.07071429	9.92	3.56E-06	140.33	88	1.043	3.10E-03	1.12E-05	44.179	11.64%	336.45	3.13%

Table 15: SPL predictions of a 500 Hz sound wave in free space.

Experiment Dimensions (mm)			1500 x 70 x 1 (Length x Height x Width)			**500 Hz**, ~50 dB			Wave Length (m)	6.946E-01	Wave Period (sec)	2.000E-03			
	Exp. #	Grid (mm)	CFL	Cells per wave length	Time Step (sec)	Steps per wave period			Exp. #	Travel (m)	Time (sec)	Amplitude (m/sec)	SPL (dB)	SPL (% err)	Speed (m/s)	Speed (% err)
Fine Grid	101	0.35	0.9923318	1984.66	1.00E-06	2000.00			101	1.173	3.38E-03	2.19E-05	49.997	0.00%	347.44	-0.04%
	89	5	0.99	138.93	1.43E-05	140.33			89	1.173	3.39E-03	2.16E-05	49.885	0.23%	345.59	0.50%
	103	**7**	**0.9923318**	**99.23**	**2.00E-05**	**100.00**			**103**	**1.173**	**3.40E-03**	**2.17E-05**	**49.934**	**0.13%**	**345.10**	**0.64%**
	90	10	0.99	69.46	2.85E-05	70.16			90	1.173	3.40E-03	2.16E-05	49.884	0.23%	344.84	0.71%
	91	10	0.495	69.46	1.43E-05	140.33			91	1.173	3.40E-03	2.16E-05	49.885	0.23%	344.84	0.71%
Medium Grid	92	20	0.99	34.73	5.70E-05	35.08			92	1.173	3.45E-03	2.10E-05	49.635	0.73%	340.29	2.02%
	93	20	0.495	34.73	2.85E-05	70.16			93	1.173	3.45E-03	2.10E-05	49.635	0.73%	340.29	2.02%
	94	20	0.2475	34.73	1.43E-05	140.33			94	1.173	3.45E-03	2.10E-05	49.635	0.73%	340.29	2.02%
Coarse Grid	95	50	0.99	13.89	1.43E-04	14.03			95	1.173	3.29E-03	1.03E-05	43.445	13.11%	356.06	-2.52%
	96	50	0.28285714	13.89	4.07E-05	49.12			96	1.173	3.29E-03	1.03E-05	43.441	13.12%	356.16	-2.55%
	97	50	0.14142857	13.89	2.04E-05	98.23			97	1.173	3.29E-03	1.03E-05	43.441	13.12%	356.18	-2.55%
	98	50	0.07071429	13.89	1.02E-05	196.46			98	1.173	3.29E-03	1.03E-05	43.440	13.12%	356.17	-2.55%

Table 16: Additional SPL predictions of a 10,000 Hz sound wave in free space.

Experiment Dimensions (mm)			200 x 10 x 0.5 (Length x Height x Width)			**10,000 Hz**, ~50 dB			Wave Length (m)	3.473E-02	Wave Period (sec)	1.000E-04			
	Exp. #	Grid (mm)	CFL	Cells per wave length	Time Step (sec)	Steps per wave period			Exp. #	Travel (m)	Time (sec)	Amplitude (m/sec)	SPL (dB)	SPL (% err)	Speed (m/s)	Speed (% err)
	68	0.25	0.99	138.93	7.13E-07	140.33			68	0.209	6.37E-04	2.14E-05	49.812	0.37%	327.89	5.59%
	68	0.25	0.99	138.93	7.13E-07	140.33			68	0.409	1.21E-03	2.11E-05	49.696	0.60%	337.09	2.94%
	68	0.25	0.99	138.93	7.13E-07	140.33			68	0.609	1.79E-03	2.09E-05	49.614	0.76%	340.54	1.95%
	68	0.25	0.99	138.93	7.13E-07	140.33			68	0.809	2.36E-03	2.08E-05	49.547	0.90%	342.18	1.48%
	68	0.25	0.99	138.93	7.13E-07	140.33			68	1.009	2.94E-03	2.06E-05	49.492	1.01%	343.25	1.17%
	68	0.25	0.99	138.93	7.13E-07	140.33			68	2.009	5.82E-03	2.01E-05	49.249	1.50%	345.28	0.59%
	68	0.25	0.99	138.93	7.13E-07	140.33			68	4.009	1.16E-02	1.94E-05	48.960	2.07%	346.32	0.29%
	68	0.25	0.99	138.93	7.13E-07	140.33			68	5.009	1.45E-02	1.94E-05	48.935	2.12%	346.54	0.22%
	68	0.25	0.99	138.93	7.13E-07	140.33			68	10.009	2.89E-02	1.90E-05	48.767	2.46%	346.90	0.12%
	68	0.25	0.99	138.93	7.13E-07	140.33			68	15.009	4.33E-02	1.83E-05	48.447	3.10%	347.02	0.09%
	68	0.25	0.99	138.93	7.13E-07	140.33			68	19.609	5.65E-02	1.78E-05	48.205	3.58%	347.07	0.07%

5.2 Flow Over a Cylinder Noise Predictions

Building upon the studies with a spatial resolution of 100 cells per wavelength and a temporal resolution of 100 time steps per wave period used for the sound wave in the free space experiment, a series of CFD investigations of a simple flow over a cylinder were performed. The investigations were performed on a uniform ideal gas in a cross-flow with a velocity of 50 m/s over a 20 mm diameter cylinder. The Reynolds number (Re) for this flow was 70,000. Oscillatory vortices were formed in the wake region behind the 20 mm diameter cylinder. This simple type of fluid flow generated a 550 Hz Aeolian tone directly from the vortex shedding, shown in Figure 60 below. The 550 Hz prediction was based on a Strouhal number (St) of 0.22.

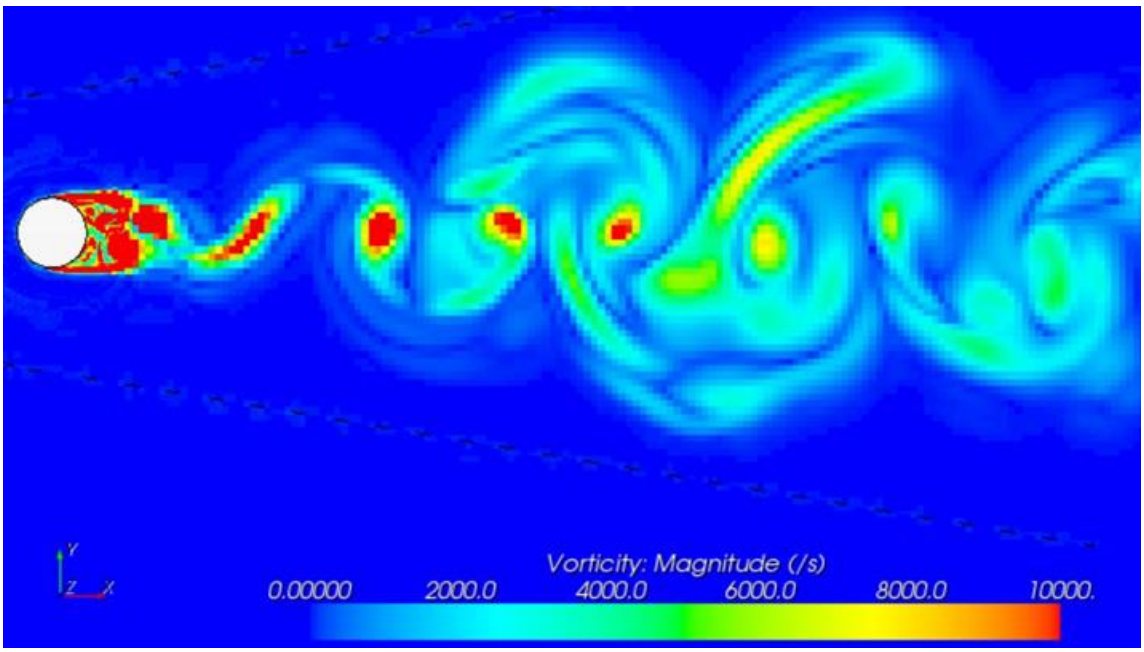

Figure 60: Vortex shedding from a 20 mm diameter cylinder in a 50 m/s uniform ideal gas cross-flow.

Although a 550 Hz tone should be generated by the turbulent flow over a cylinder, turbulence in general does not produce mono-tone sound but broad-band noise; therefore, a spatial resolution of 100 cells per wavelength up to the frequency of interest was

suggested by the free space experiment. Three CFD investigations were performed to study the nature of broad-band noise generated from flow over a cylinder. The first was CFD investigation #25, in which the cylinder was placed inside a cylindrical domain with a 0.6 m diameter that was 0.04 m wide. The time step size was 2.5×10^{-5} s. The maximum grid size was 2 mm within the turbulent region. The overall computational grid size was 750,000 cells. The second CFD investigation was #114, which had a bell shape domain shown in the upper image of Figure 61. This domain had a diameter of 0.15 m upstream of the cylinder and expanded to a height of 0.6 m at 0.6 m downstream. A maximum grid size of 2 mm was considered to be sufficient within the turbulent region. The overall computational grid size was 925,000 cells, which is shown in the upper part of Figure 61. The last CFD investigation of the flow over a cylinder was #115, which had a domain identical to #114. The time step size for CFD investigations #114 and #115 was 1.0×10^{-6} s.

In order to extract the acoustic information from the transient detached eddy simulation (DES), a choice had to be made between the two types of surfaces described by Ffowcs Williams – Hawkings (FW-H), impermeable or permeable. An impermeable FW-H surface is generally more practical when the surface is near or within the predominant noise producing region. These surfaces usually have a boundary layer, which requires an extremely fine grid. The grid height near the wall should be small enough to provide Wall $Y+$[56] values less than 1.0. The CFD investigations #25 and #114 of the flow over the cylinder resulted in a maximum Wall Y+ value of 0.35. In order to control the grid

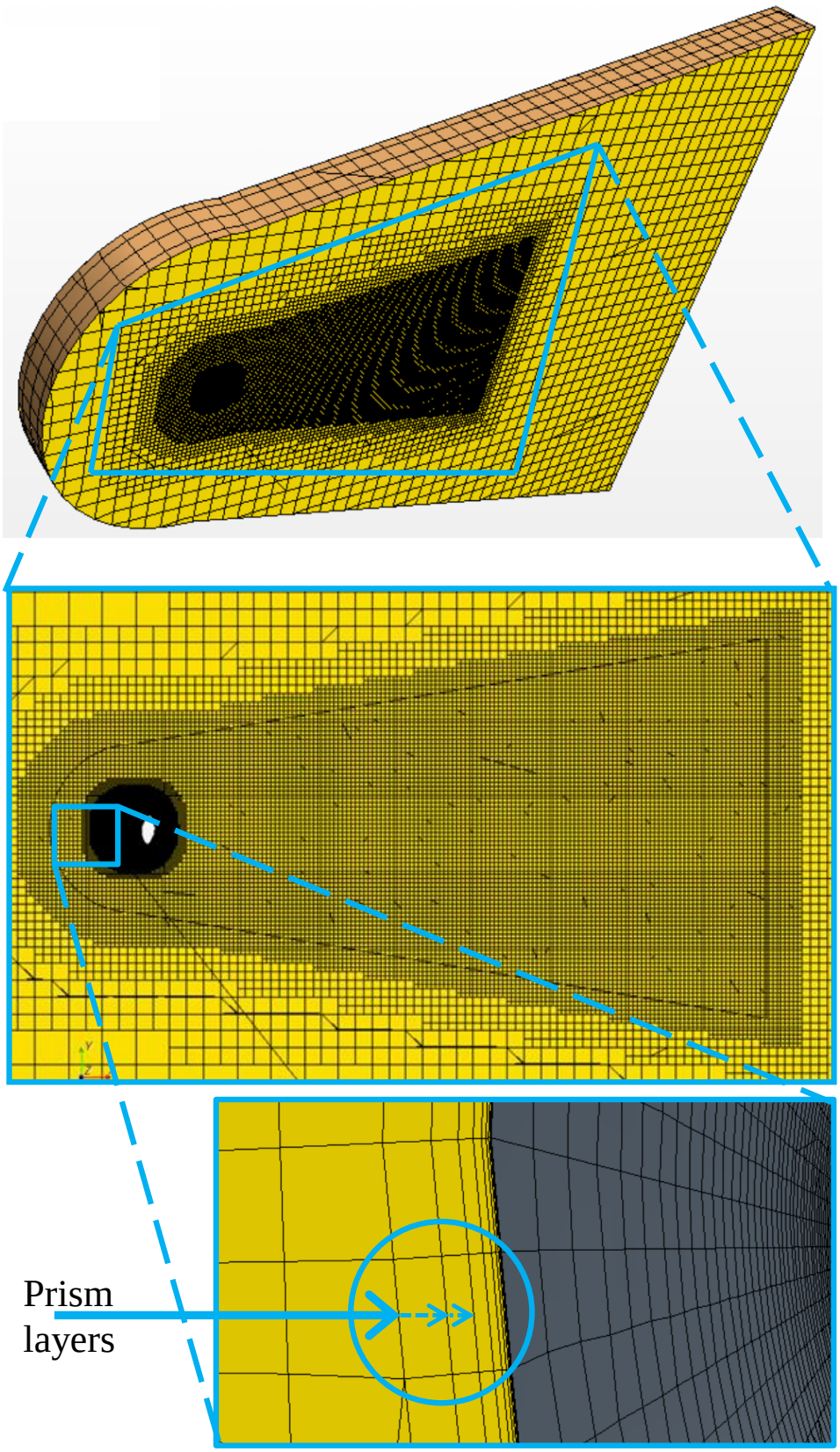

Figure 61: Computational grid for the flow over a cylinder in free space used in CFD investigation 114 and 115.

resolution within the boundary layer region, a specific type of computational grid was used called prism layers. A prism layer was uniformly built radially around the surface of the cylinder. Several of the prism layers can be seen in the bottom image of Figure 61. A total of 10 prism layers were used with a starting cell height of 0.001 mm. The lateral (cross-flow direction) had grid surface size of 0.8 mm in CDF investigations #25 and #114, which provided accurate resolution up to 4,400 Hz. One of the main problems with using impermeable surfaces is that there is an underlying assumption that the fluid properties are uniform between the FW-H surface and the prediction locations. The fluid properties near an impermeable surface within a boundary layer have significantly different fluid properties from those that exist in the far field. While an impermeable FW-H surface was used for the cylinder wall in CFD investigations #25 and #114, a permeable FW-H surface was used in CFD investigation #115. A permeable FW-H surface should be used to surround the turbulent noise producing region, and a maximum grid size of 100 cell cells per wavelength must strictly be enforced. A maximum 2 mm grid size was maintained within the permeable FW-H surface and provided accurate resolution up to 1,750 Hz. The permeable FW-H surface can be seen in the middle image of Figure 61. The maximum 2 mm grid size was maintained within the FW-H surface in order to limit the degradation of the propagating sound waves. This requirement significantly increased the required computational resources to complete the DES. For example, the steady state calculations of CFD investigation #114 were completed in 42 hours, but the transient DES calculations required considerably more time, an additional 633 hours.

The steady state case is more advantageous from a computational efficiency point of view but provides a limited amount of acoustical information. Using the flow field

quantities from the Reynolds-averaged Navier-Stokes equation, a CFD model will provide turbulent time and length scales that can be coupled to acoustical correlations. These correlations or "noise source analogies" can be used to compute the location and strength of the main sources of sound generated aerodynamically. These models can also be used to estimate the local sound power generated per unit volume in the turbulent flow field. One particular model, by Goldstein,[19] is based on Lighthill's[14] theory for predicting aerodynamic noise from a turbulent shear flow and assumes locally homogeneous and isotropic turbulence. Based on Goldstein's theory, this model is a generalization of the model developed by Ribner[26] for axisymmetric turbulent flows. Similar to the Goldstein model, the Proudman[20] noise source model evaluates the sound power per unit volume of the noise that is generated by quadrupoles normally found in supersonic turbulent flows. Along with the Goldstein and Proudman models, there are additional models that can be used to provide insight into the sound generated by turbulent flow fields; however, all of these analogies only predict localized noise generation and do not allow for propagation of the sound outside the turbulent flow, as shown in Figure 62 below.

Another approach was to simply calculate the pressure fluctuations for all the cells within the computational domain. This method is called direct computational aero-acoustics (CAA). A very large computational effort must be performed so that the pertinent sound pressure locations of interest are within the computational domain. The significant computational effort is not an issue directly from the CDF model but from the data

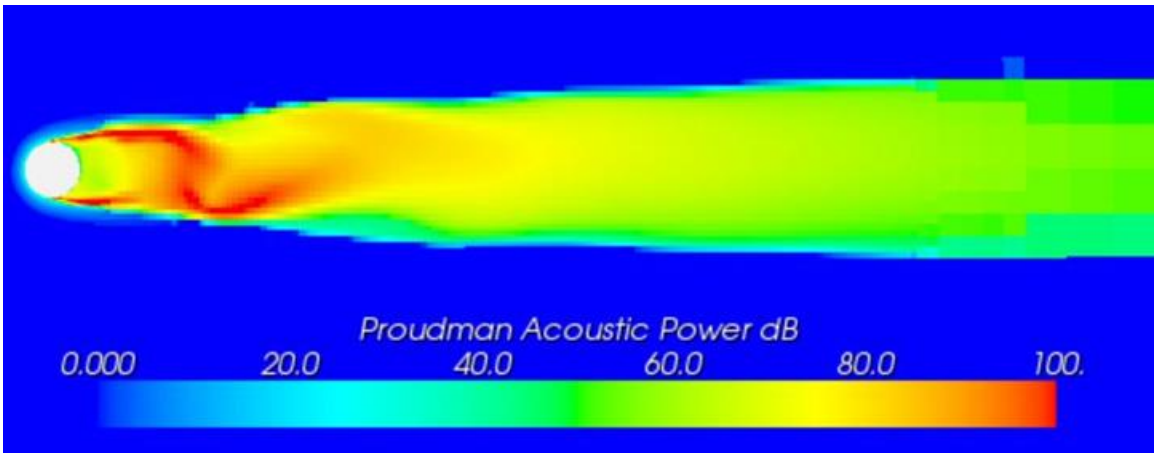

Figure 62: Noise prediction of a flow over a cylinder using Proudman's model.

storage size requirement and speed of access to that storage medium. During CFD investigation #25, CAA was performed. The model file required only 192 MB of hard drive storage space; the CAA file of the data generated from a plane in the center of the computational domain and the cylinder surface required 16.2 GB. An extremely fine grid of 100 cells per wave length should be built from the noise source locations to the prediction locations. Noise predictions at any frequency were possible within the computational domain using CAA. The OASPL predictions from the direct CAA of CFD investigation #25 are shown in Figure 63. One interesting phenomenon was the plume of noise at the exit computational domain on the far right side of Figure 63. The larger than normal distribution of noise at the domain exit is due to the domain is too close to the turbulent field. The noise plume issue required modification of the computational domain, which was one of the reasons for undertaking CFD investigations #114 and #115. While insight was obtained from direct CAA, no other CAA investigations were performed because of the data storage requirement.

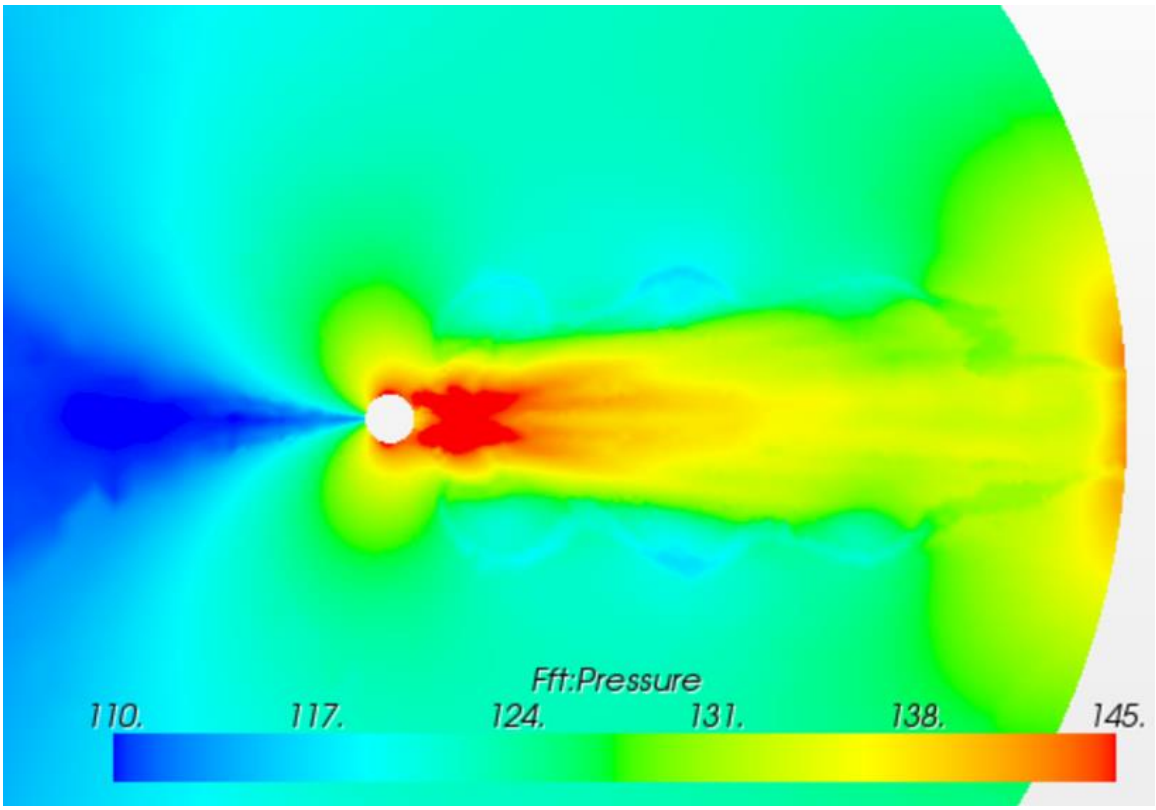

Figure 63: OASPL noise prediction of a flow over a cylinder using direct CAA

The FW-H method can be used to provide sound propagation into the far field outside the computational grid. The model can also be used to calculate mid- to far-field sound radiation from the near-field pressure fluctuations on the FW-H surface. The propagation of sound cannot be explicitly calculated as with the CAA, but it can be computed using an analytical integral solution to the generalized wave equation. FW-H requires significantly less computational resources than direct CAA. An example of the noise distribution from CFD investigation #115 is shown in Figure 64. The location for the prediction was 0.2 m from the cylinder center-line and at 90 degrees relative to the downstream direction. The broad-band noise prediction showed a significant ~550 Hz contribution. The ~550 Hz peak was actually at 564.6 Hz with a SPL of 97.7 dB; a second peak was seen at 1709 Hz with a SPL of 82.2 dB. Lienhard[64] suggested that there

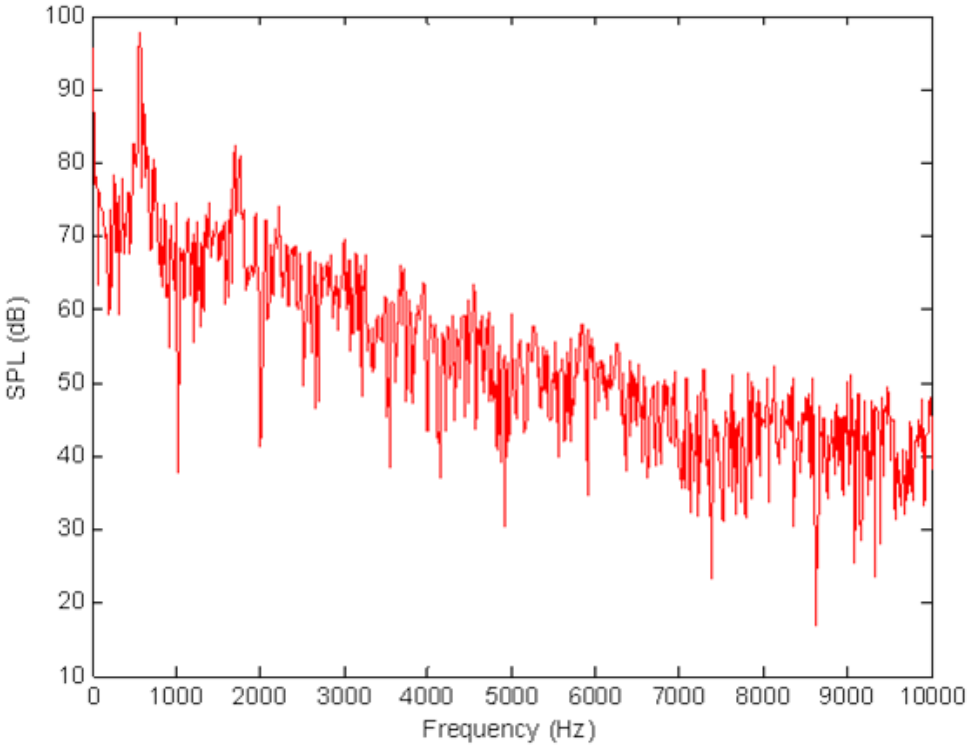

Figure 64: Noise prediction from CFD investigation #115 at 0.2 m and 90 degrees.

is a three-dimensional nature to the flow over a cylinder at turbulent Reynolds number (Re) between 300 and 3 million. Within this turbulent flow, Lienhard suggested that a second non-dominant frequency could be present. A second tone based on Strouhal number (St) of between 0.267 and 0.47 was suggested by Lienhard; however, a prediction made using a St of 0.47 results in a tone of 1,175 Hz. Although Lienhard's prediction of 1,175 Hz is much lower than that seen in the data, the frequency of 1,709 Hz nearly matches three times the dominant frequency, i.e. 1,694 Hz (3x564.6). The second peak frequency seen in Figure 64 at the 0.2 m location is not present in the data for the 2.0 m location, which is shown in Figure 65. The only perceivable peak is at 564.6 Hz with a SPL of 77.5 dB.

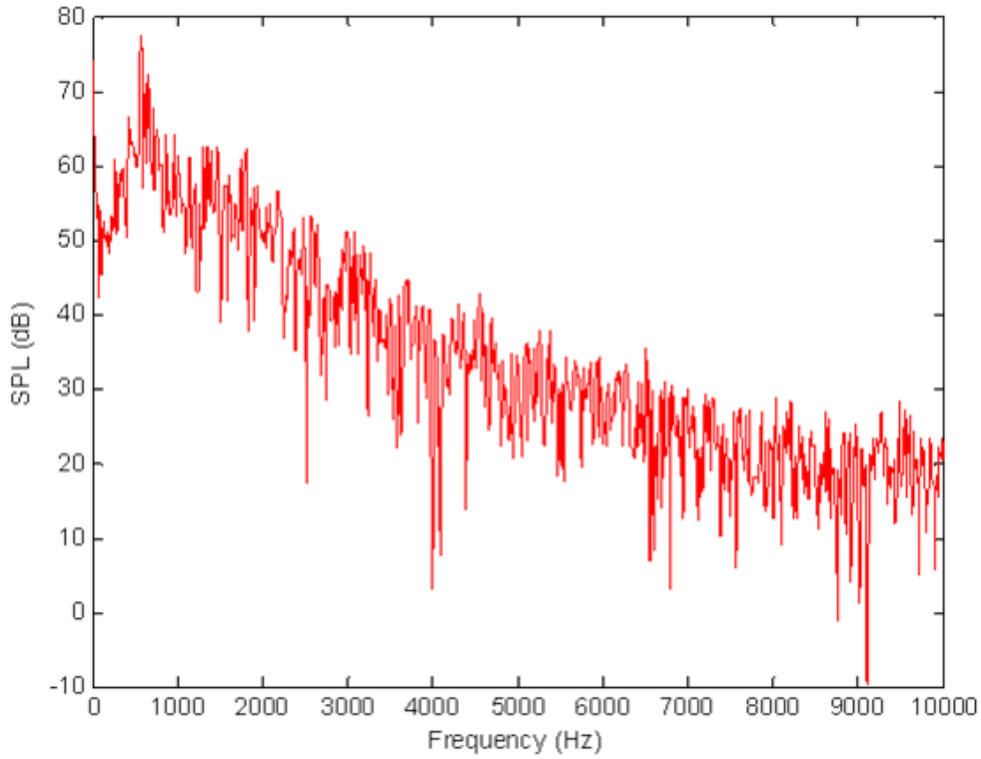

Figure 65: Noise prediction from CFD investigation #115 at 2.0 m and 90 degrees.

Combining the sound pressures at all the prediction locations for each of the different CFD investigations, a table of the OASPL was constructed and is presented as Table 17. Several entries in Table 17 are grayed because the data from that CFD investigation was not collected but could have been collected. Also, the data for crossed-out entries in Table 17 could not have been collected do to the model constraints. The FW-H predictions along the vertical direction using the FW-H permeable surface in CFD investigation #115 were quite accurate. The vertical directions, which are 90 and 270 degrees relative to the downstream direction, are the directions in which the predominant noise is produced. The accuracy was based on the sound pressure, P, calculated from a modified version of Blevins'[65] empirical equation,

		Location: [Radius (m), Angle (deg)]											
Experiment #	CFD Model / Acoustic Model	[0.2, 0]	[2.0, 0]	[4.0, 0]	[6.0, 0]	[0.2, 45]	[2.0, 45]	[4.0, 45]	[6.0, 45]	[0.2, 90]	[2.0, 90]	[4.0, 90]	[6.0, 90]
25	RANS / Proudman	72.0				0.0				0.0			
25	DES / FW-H	95.7	57.8	48.8	44.3					91.3	68.0		
25	DES / CAA	133.7				123.0				120.3			
114	RANS / Proudman	72.4				0.0				0.0			
114	DES / FW-H	93.2	60.9	54.3	50.7	94.6	69.4	63.3	50.7	95.0	72.2	66.2	62.7
115	DES / FW-H	107.1	77.2	66.0	62.0	122.6	86.8	79.9	62.0	101.6	83.5	77.8	74.4
Empirical	Blevins	-∞	-∞	-∞	-∞	101.1	81.1	75.1	71.6	104.1	84.1	78.1	74.6
Experiment #	CFD Model / Acoustic Model	[0.2,135]	[2.0,135]	[4.0,135]	[6.0,135]	[0.2,180]	[2.0,180]	[4.0,180]	[6.0,180]	[0.2,225]	[2.0,225]	[4.0,225]	[6.0,225]
25	RANS / Proudman	0.0				0.0				0.0			
25	DES / FW-H					96.1	57.6						
25	DES / CAA	115.8				110.2				115.4			
114	RANS / Proudman	0.0				0.0				0.0			
114	DES / FW-H	93.8	69.7	63.6	60.1	93.6	60.8	54.2	50.5	94.1	69.7	63.6	60.1
115	DES / FW-H	115.9	79.7	74.0	70.6	118.6	74.3	67.3	63.7	115.9	78.9	73.1	69.8
Empirical	Blevins	101.1	81.1	75.1	71.6	-∞	-∞	-∞	-∞	101.1	81.1	75.1	71.6
Experiment #	Model Type	[0.2,270]	[2.0,270]	[4.0,270]	[6.0,270]	[0.2,315]	[2.0,315]	[4.0,315]	[6.0,315]				
25	RANS / Proudman	0.0				0.0							
25	DES / FW-H	91.1	68.0	61.9	58.4								
25	DES / CAA	120.6				123.5							
114	RANS / Proudman	0.0				0.0							
114	DES / FW-H	94.9	72.2	66.2	62.6	94.3	69.3	63.2	59.7				
115	DES / FW-H	101.3	83.2	77.5	74.1	122.7	86.7	79.9	76.1				
Empirical	Blevins	104.1	84.1	78.1	74.6	101.1	81.1	75.1	71.6				

$$P^2 = \frac{\sin^2\theta}{32R^2a_o^2}\left(\rho^2U^6L^2C_L^2St^2\right) \tag{67}$$

where θ is the angle relative to the down-stream direction, R is the radial distance to the cylinder center-line, a_o is the speed of sound in air (347.3 m/s), ρ is the density of air (1.177 kg/m^3), U is the cross-flow air velocity (50 m/s), L is the length of the cylinder, (0.050 m), C_L is the lift coefficient, and St is the Strouhal number (0.22). Equation (67) is nearly the same as the equations in Gerrard's first[66] and second[67] empirical models for predicting the overall sound pressure level for a flow over a cylinder. From Gerrard's second model, the lift coefficient, C_L, was calculated to be 0.8571. In the CFD investigation #25, a peak value of the lift coefficient was approximately 0.89, which was within the expected values from the data Lienhard[64] presented. The FW-H impermeable surface used in the predictions for the CFD investigation #114 generally provided noise predictions that were 10 dB or more lower than the noise predictions using FW-H permeable surface in CFD investigation #115. The under-prediction of noise in CFD investigation #114 was likely due to the fluid properties not being uniform as the sound wave passed through the boundary layer. The FW-H permeable surface in the CFD investigation #115 was able to provide accurate noise predictions. The CAA noise predictions in CFD investigation #25 were excessively high when compared to the Blevins empirical model. The high noise prediction obtained by the direct CAA method was uncharacteristic and not expected. Near field noise predictions are difficult to get accurate, better agreement between CAA and Blevins was expected.

The data obtained from the noise generated from the turbulent flow were discrete samples of a real world continuous system. The data shown in Figures 64 and 65 was

generated by running the time varying discrete pressure samples through a fast Fourier transform (FFT). A FFT requires that the sample size be a multiple of 2 regardless of the sampling rate. A question remains, "What sample size is needed to reproduce a broad-band noise signal?" Using a combination of five sound waves and random noise, a time varying pressure signal was constructed. The five sinusoidal signals were all of different frequencies and amplitudes: ~106.7 Hz at ~70.4 dB, ~628.8 Hz at ~66.3 dB, ~1,271.4 Hz at ~61.4 dB, ~5,761.8 Hz at ~52.0 dB, and ~10,234 Hz at ~50.9 dB. The noise signal had random amplitude across the frequency spectrum with maximum noise amplitude of no more than 10 dB. The summation of this signal is shown in Figure 66. The sample rate was 1×10^{-6} s, which was a temporal resolution of 100 time steps of a 10,000 Hz wave period. A random beginning time of 2.7×10^{-3} s was selected for the start of the data collection. Next, several different sample sizes were selected and processed through a FFT. Examples of the FFTs from a $\sim 4 \times 10^{-3}$ s and $\sim 3.2 \times 10^{-2}$ s sample size are shown in Figure 67 below. The resulting peak amplitudes from each FFT were compared with the amplitudes from the original sound wave signal. The root mean squared (RMS) error of the difference between the two amplitudes was tabulated and shown in Figure 68 below. The data showed that a minimum of 1.0×10^{-2} s of data is required for the RMS error to be below 1% with a discrete sample rate of 1.0×10^{-6} s. The RMS error at $\sim 3.2 \times 10^{-2}$ s sample size did not drop significantly; but as the FFT clearly shows in Figure 67, the frequency resolution was much finer. $\sim 3.2 \times 10^{-2}$ s (2^{15} samples), the frequency resolution was 30.5 Hz $\left(1/\left[\left(1.0 \times 10^{-6}\right) 2^{15}\right]\right)$. If a smaller frequency resolution is required, more data points must be used in the FFT. For example, a ~1 Hz resolution would require

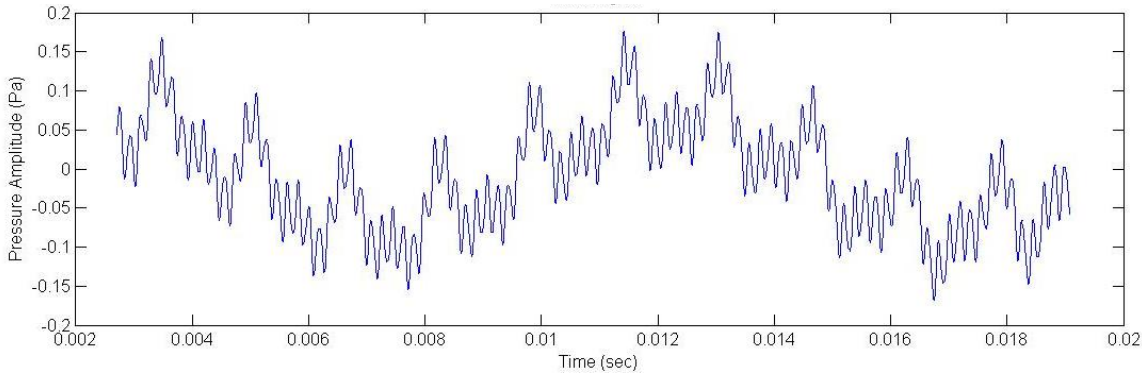

Figure 66: Time varying pressure of a summation of five sound waves and a 10 dB random broad-band noise source.

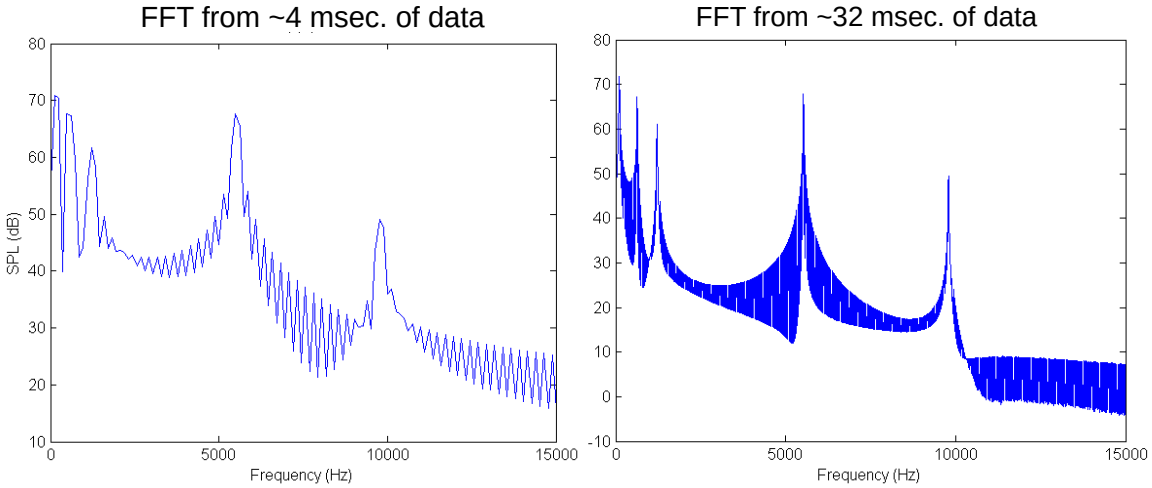

Figure 67: Fast Fourier transform of time varying pressure with two sample sizes.

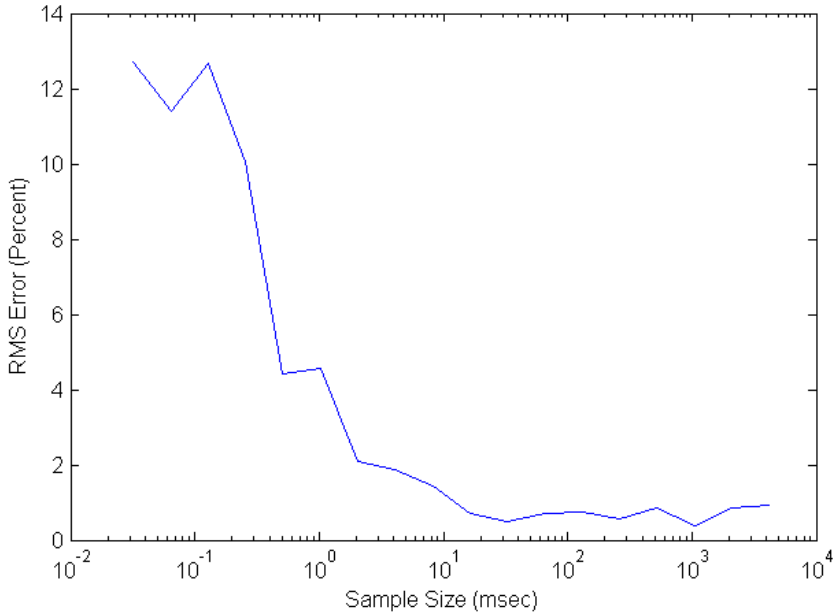

Figure 68: The RMS error percentage compared with the sample size.

~1.05 seconds $\left(1/\left[\left(1.0x10^{-6}\right)2^{20}\right]\right)$ of data at a 1.0 x 10^{-6} s sample rate. Reinelt[62] suggested that a sample should include at least five wave periods for the lowest frequency of interest. Using five wave periods within ~3.2 x 10^{-2} s, the lowest frequency that could be resolved would be 156.25 Hz, which is higher than the ~106.7 Hz signal. Based on the RMS error plot, a smaller set of wave periods is needed in order to resolve the lower frequencies. Within an ~3.2 x 10^{-2} s data sample, there are 3.5 wave periods of an ~106.7 Hz signal. This suggested that only three wave periods are needed to resolve the lower end of the frequencies spectra.

Although analysis of a simple flow over a cylinder has been studied for more than one hundred years[68] by some well-known scientists in fluid dynamics, the ability to predict lift and drag forces and the sound generated by much more complex shapes are possible within the CFD environment. Ayers[69] designed several experiments to test the forces on IsoTruss® that had potential application in offshore oil rigs. The IsoTruss® structure can be seen in the support legs of the oil rig, shown in Figure 69 below. Branscomb[70] built several of these complex structures for his doctoral work at Auburn University. He was interested in the noise generated from the cross-flow through open architecture composite tubes, shown in Figure 70 below. An example of the velocity field and noise generated are shown in Figures 71 and 72, respectively. As can be seen in Figure 71, the normal oscillator vortices will not form in the wake region behind the outside diameter of the tube because of the open design.

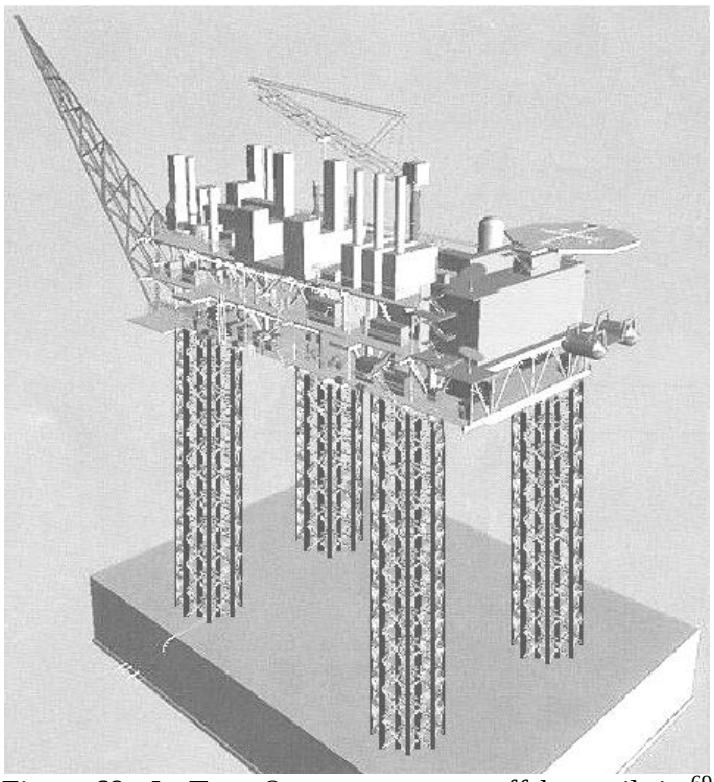

Figure 69: IsoTruss® structure on an offshore oil rig.[69]

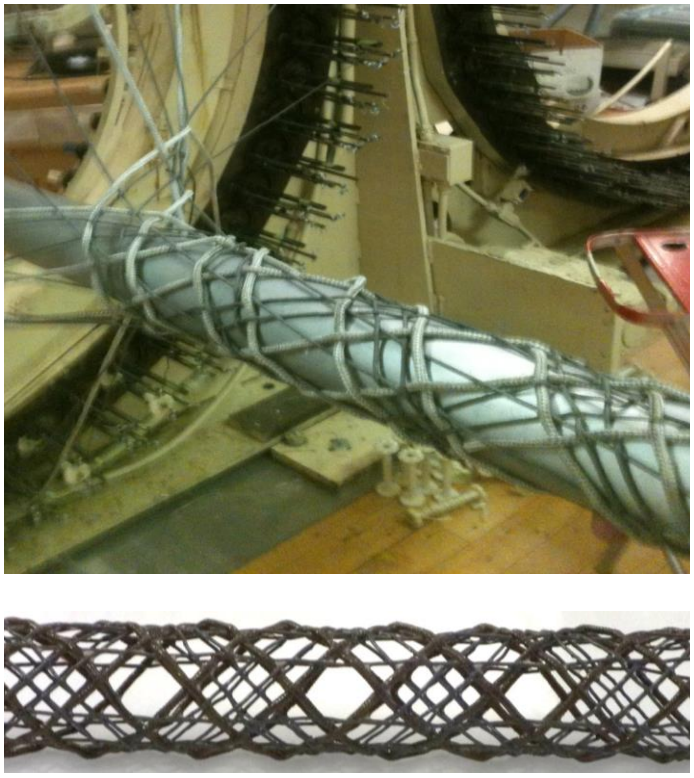

Figure 70: Manufacturing of an open architecture composite tube.[70]

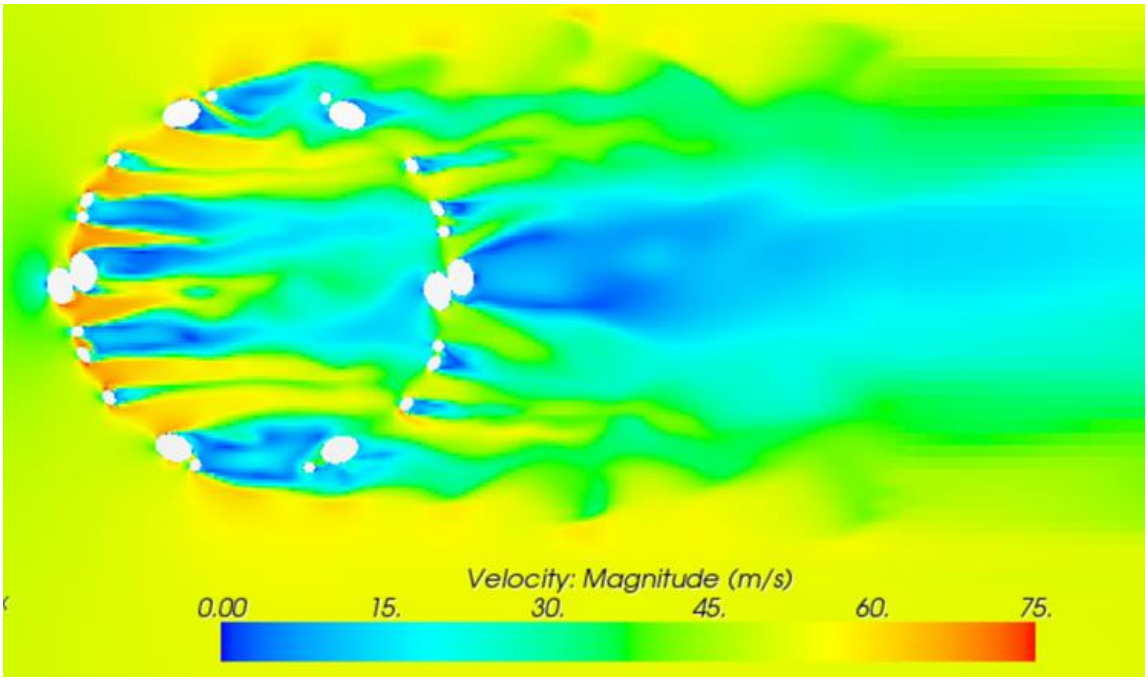

Figure 71: Velocity field of an ideal gas in cross-flow through a composite tube.

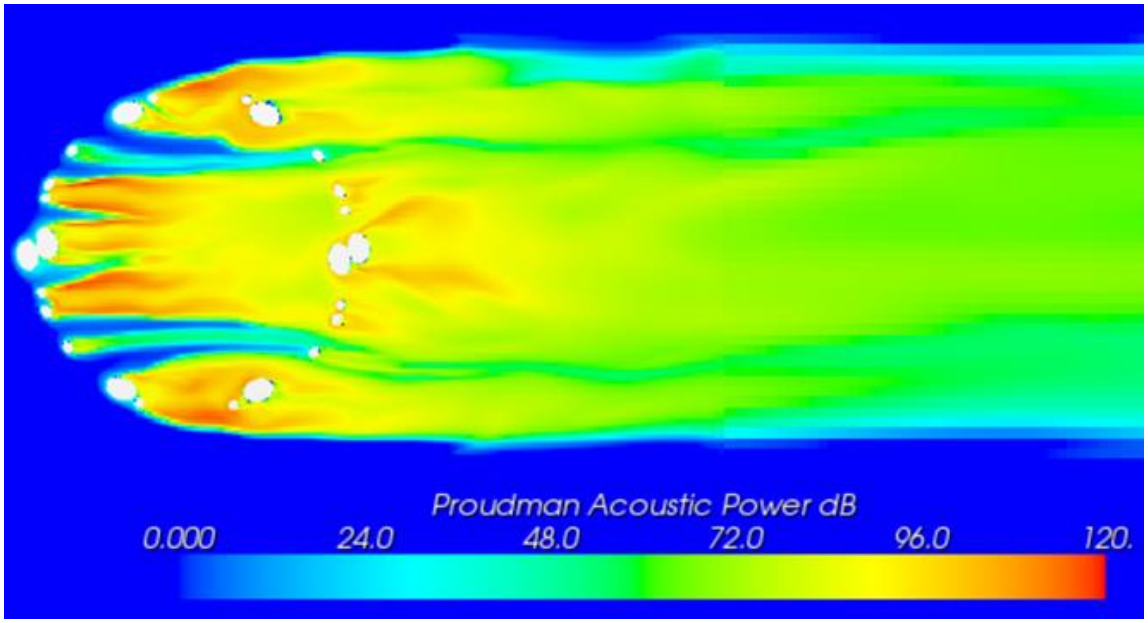

Figure 72: Noise generated with an ideal gas by cross-flow through a composite tube.

5.3 Supersonic Jet Wind Tunnel Noise Predictions

The sound wave in free space investigation provided data that defined the spatial and temporal resolution requirements for noise predictions using CFD. The flow over a cylinder investigation provided data that revealed that broad-band noise predictions from a turbulent flow were possible. The RANS / Proudman and DES / FW-W methods were used in the supersonic jet wind tunnel investigation in order to predict the broad-band noise for a simplified version of a space vehicle. The simplified version of a space vehicle was based on Norum's[37] supersonic jet research discussed in Section 3.4. The supersonic jet had an air flow rate at the nozzle exit of 406.6 m/s (1.15 M), which is shown in Figure 73 below. Also, the formations of mild shock diamonds were clearly visible in the lower image shown in Figure 73. The core region of the jet was also easily defined as the steady-state images shown in Figure 73; but the unsteady velocity profile, shown in Figure 74, indicated significant break down of the core region of the jet. Both the ambient air and the jet had a nominal temperature of 311^{o}K. The exit diameter of the nozzle was 0.073 m. The supersonic jet flowed into an anechoic wind tunnel with an air speed of 35.35 m/s (0.1 Ma). The broad-band noise produced by the jet does not have a dominant tone like the flow over the cylinder. An example of the broad-band nature of the noise is produced shown in Figure 75.

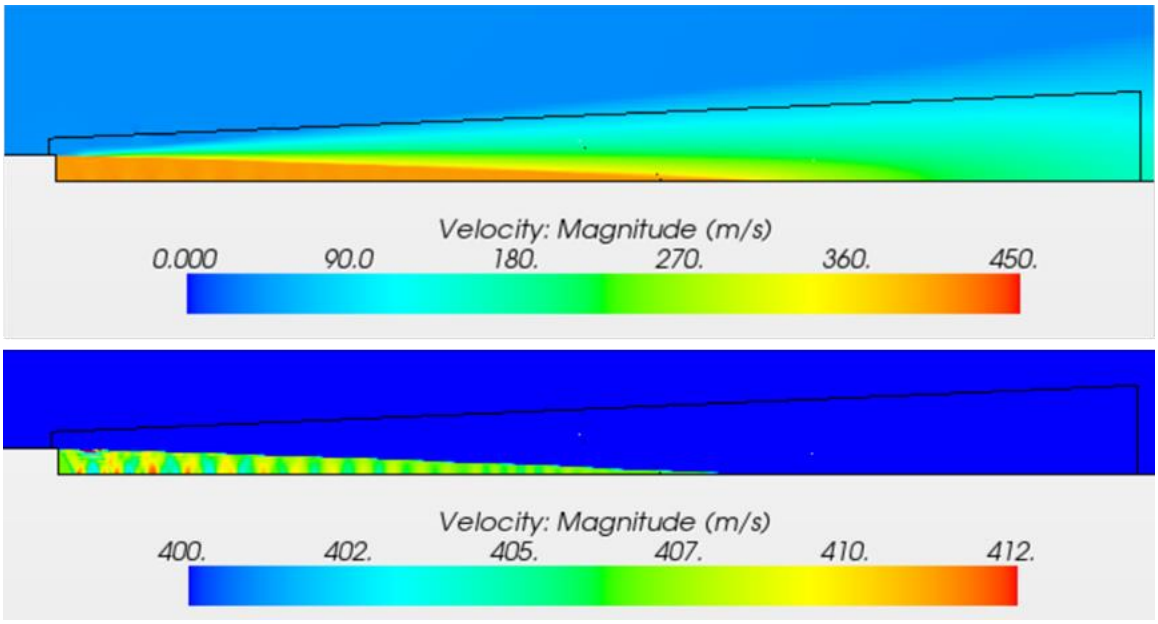

Figure 73: Steady-state velocity profile of supersonic jet.

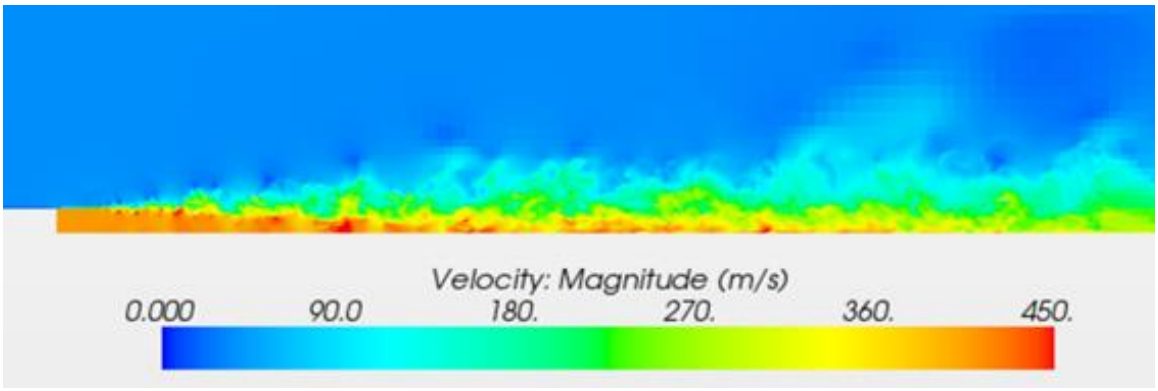

Figure 74: Unsteady velocity profile of supersonic jet.

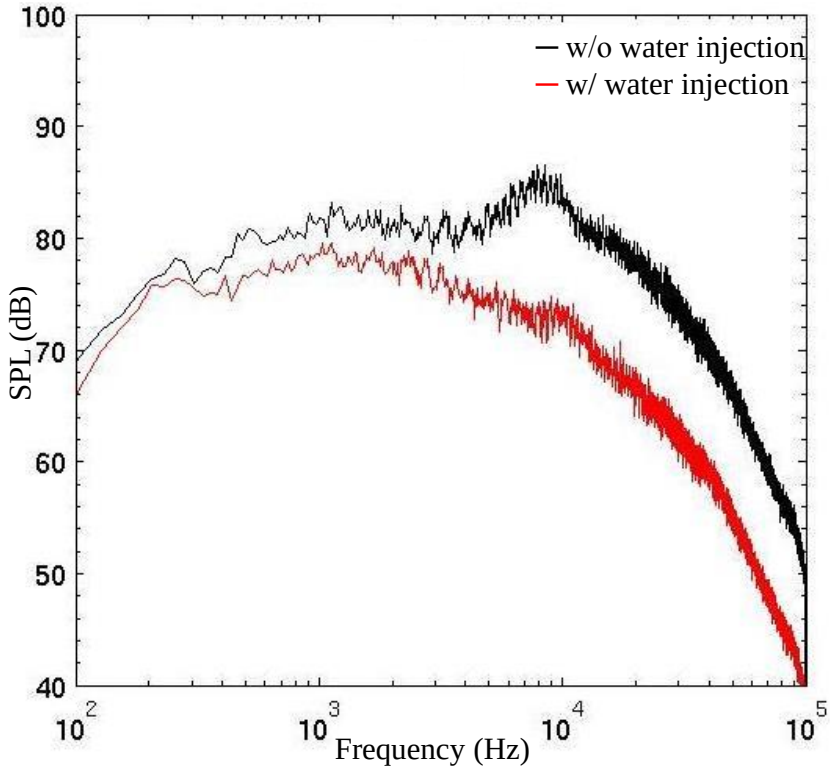

Figure 75: Example of broad-band noise produced by the supersonic jet in Norum's research.[37]

The noise produced by the turbulence was broad-band; therefore, a spatial resolution of 100 cells per wavelength was a crucial factor in determining the maximum frequency that could be resolved during the supersonic jet CFD investigation. Based on Figure 75, a maximum frequency of 30,000 Hz would provide a solid foundation to compare with the results from Norum's research; however, the FW-H analysis would require a maximum grid size of 0.12 mm within a FW-H permeable surface. A much smaller grid size was chosen for the supersonic jet CFD investigations due to computational memory constraints. The maximum grid size was 0.625 mm used within the FW-H permeable surface, which provided a frequency resolution up to 5500 Hz.

Two CFD investigations will be discussed for their broad-band noise generation nature by the supersonic jet. Although more CFD investigations were performed, only the

two investigations provided relevant information. The first was CFD investigation #134, which was a 90^{o} wedge region shown in the upper image of Figure 76. The 90^{o} wedge represented one quarter of a full 360^{o} cylindrical domain. The majority of the domain represented an ideal gas wind tunnel flow field with the exception of the nozzle horizontal wall and nozzle vertical exit plane both are shown in the middle image of Figure 76. The extent of the computational domain was 0.15 m upstream and 2.2 m downstream of the nozzle exit plane; the domain had a general cylindrical shape with a radius of 0.350 m. The FW-H permeable surface was positioned 0.012 m upstream at a radius of 0.06 m and 1.5 m downstream at a radius of 0.125 m. The FW-H permeable surface for the CFD investigations was a wedge shaped surface shown in Figure 77. The cross-sectional outline of this FW-H surface is shown in Figure 73 as a black line in the middle of the flow field. The prism layers are shown in the lower image of Figure 76, which shows the step transition between the nozzle wall and nozzle exit. The overall computational grid size was 50.5 million cells. At the time of building this grid, it was the largest grid that could be built within the 48 GB memory constraint of the computer workstation. The steady state calculations were completed in 840 hours, but the complete transient DES calculations would have required more than 4000 hours. The second CFD investigation was #137, which was a two-dimensional slice of a three-dimensional wedge from CFD investigation #134. The extent of the two-dimensional computational domain and FW-H surface were the same except that the nature of the domain was 2D axis-symmetric. The overall computational grid size was 289,000 cells. The steady state calculations were completed in 12 hours, and the transient RANS calculations required an additional 264 hours. The time step size for both CFD investigations #134 and #137 was 1.0×10^{-6} s.

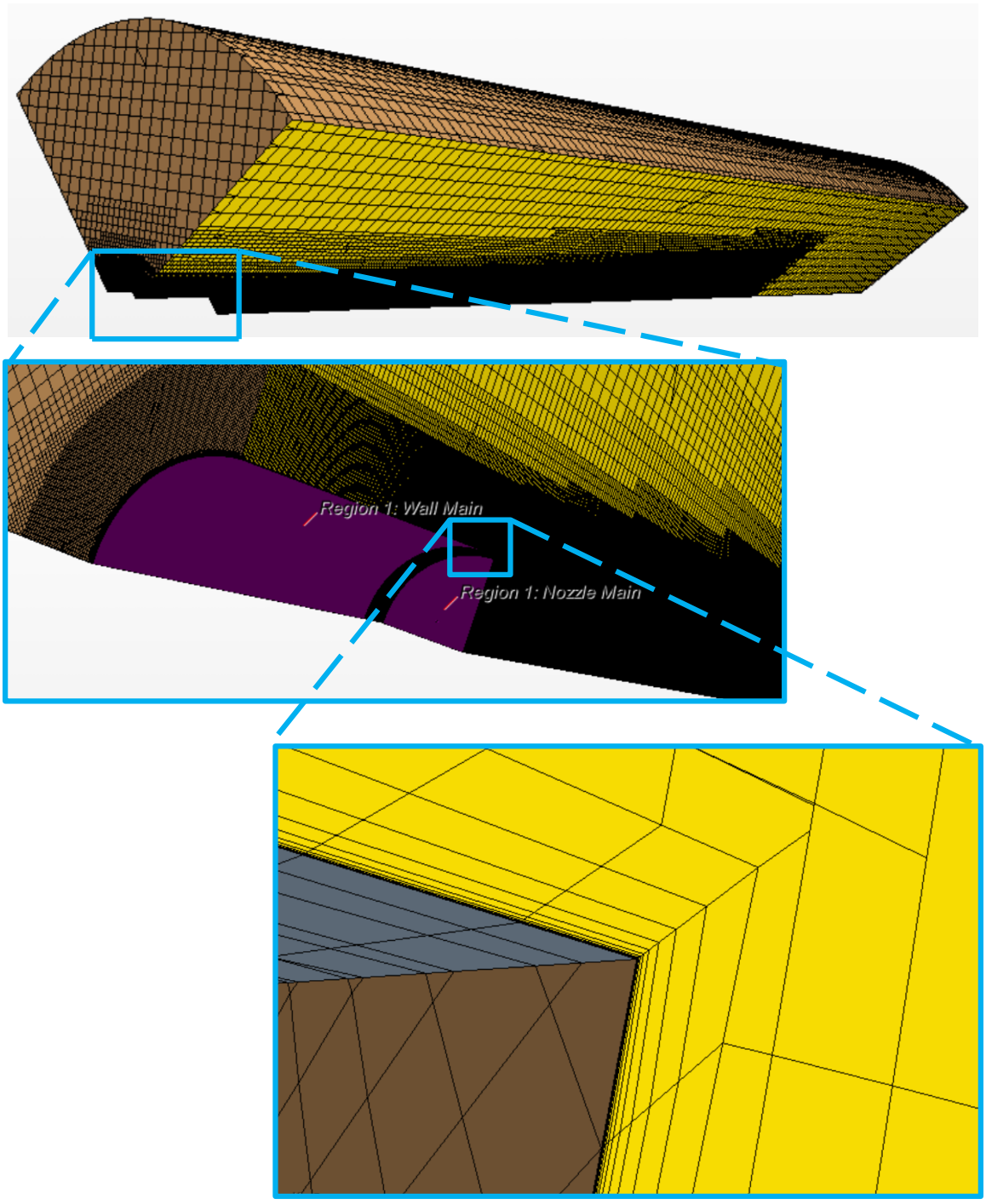

Figure 76: Computational grid for supersonic jet CFD investigation.

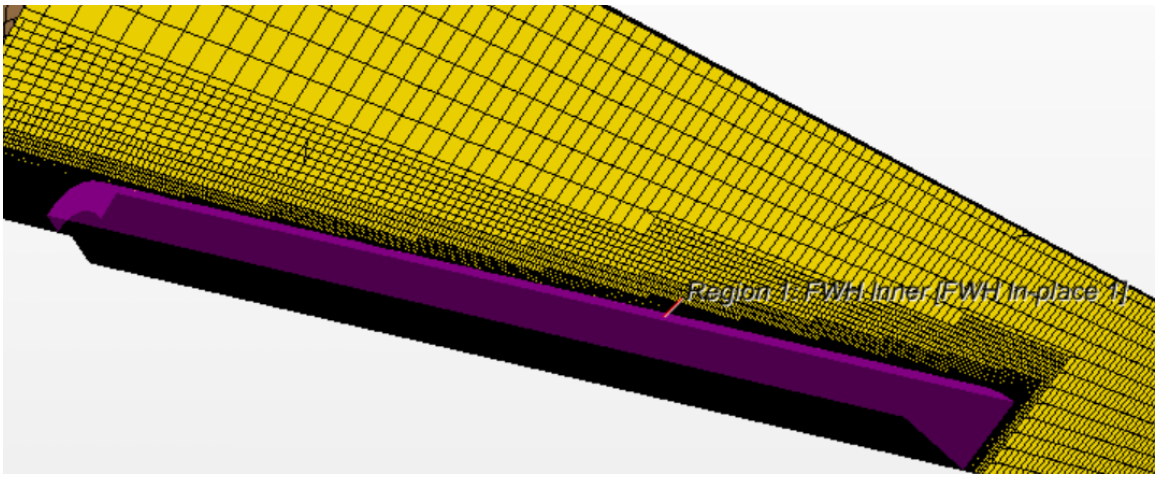

Figure 77: FW-H permeable surface used in the supersonic jet CFD investigations.

The size and position of the FW-H permeable surface was an integral part of the CFD investigations used to provide accurate noise predictions for the supersonic jet. The research of Shur et al.[60] demonstrated that an "open" FW-H surface should not be used. Shur et al. defined that an "open" FW-H surface was one that does not include the vertical disk surface at the downstream end of the FW-H permeable surface. If an "open" surface was used, Shur et al. showed that significant under-prediction of the sound pressure would occur at angles less than 40 degrees relative to the downstream direction from the nozzle exit plane. The prevailing theory of Shur et al. was that the FH-W surface should completely surround the predominant noise producing region. This region was more than 17 diameters downstream. Shur et al. showed that the FH-W surface should extend 21 to 25 diameters downstream in order to provide better predictions. The FH-W surface used in CFD investigations #134 and #137 was 20.4 diameters downstream. Generally, the FH-W permeable surface should be positioned within the uniform flow field with the exception of the truncation at the downstream position.

As found in the flow over the cylinder CDF investigations, the starting grid height near any wall surface should be small enough to provide Wall Y+ values less than 1.0. A maximum Wall Y+ value of 0.8 across nearly the entire wall surface was achieved in the CFD investigations #134 and #137. The Wall Y+ values for the CFD investigation #134 are shown in Figure 78 below. The prism layers were controlled separately in four

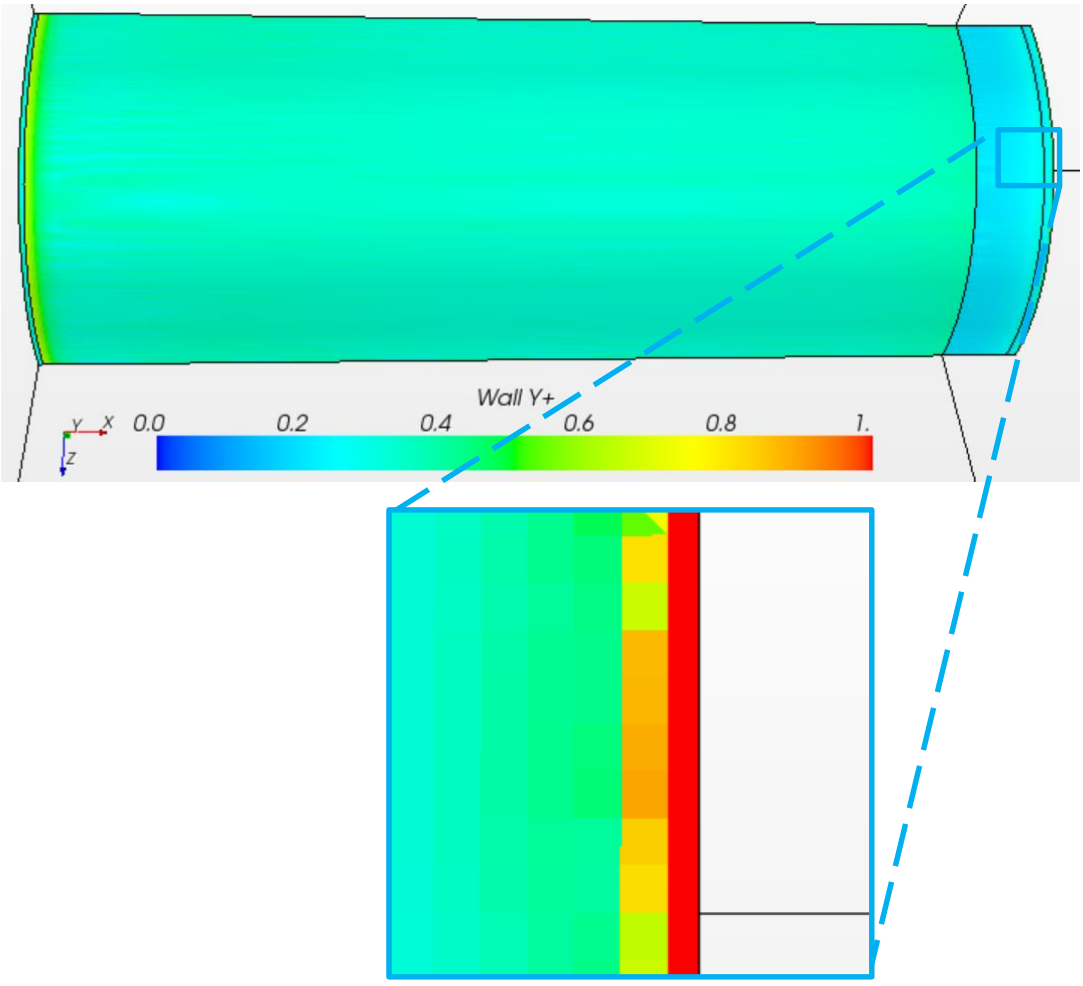

Figure 78: Wall Y+ value for supersonic jet CFD investigations #134.

different sections of the wall surface, which represented the nozzle outer wall surface. The majority of the wall had a starting cell height of 6.0×10^{-3} mm but shrunk to 1.0×10^{-6} mm at the domain inlet and nozzle step transition. The lateral surface grid size was of 0.31

mm for the majority of the wall and shrunk to 0.078 mm at the domain inlet and nozzle step transition. While a Wall Y+ value of less than 1 was maintained for the domain inlet with these grid sizes, the nozzle step transition could not as easily be controlled. The Wall Y+ value had an average value of 1.6 with a maximum value of 2.2 for the very last cell before the start of the nozzle domain boundary. A starting cell height of less than 1.0 x 10^{-6} mm was chosen initially for the nozzle step transition region but computational issues developed. The nozzle wall boundary layer and nozzle step transition were extremely critical because of the turbulent instabilities, which started within this region.

As stated in Section 5.2, the RANS CFD model coupled with the Proudman model provided noise predictions of limited use. The Proudman acoustical model can be used to predict the sound power within the CFD domain. Utilizing this acoustical model, the sound power levels (SWL) were predicted along the primary flow axis of the jet. The SWL distribution predicted by the RANS / Proudman model were compared with the distributed SWL from Equation (6) of the Lush[39] / Eldred[2] / Wilby[24] empirical noise model used in Section 3.4. Two supersonic jet flow axes were selected for the comparison of the distributed SWL. These axes started at the nozzle exit plane and extended to the end of the computational domain. The first axis was located at the jet center-line; the second axis was located at the jet core-edge. Both axes are shown in Figure 81. The distributed SWL predicted by Proudman was scaled down by 30 dB and is shown in Figures 82 and 83. The distributed SWL predicted by the two-dimensional (2D) steady-state RANS / Proudman model of the CFD investigation #137 is nearly the same as the 3D steady-state RANS / Proudman of CFD investigation #134.

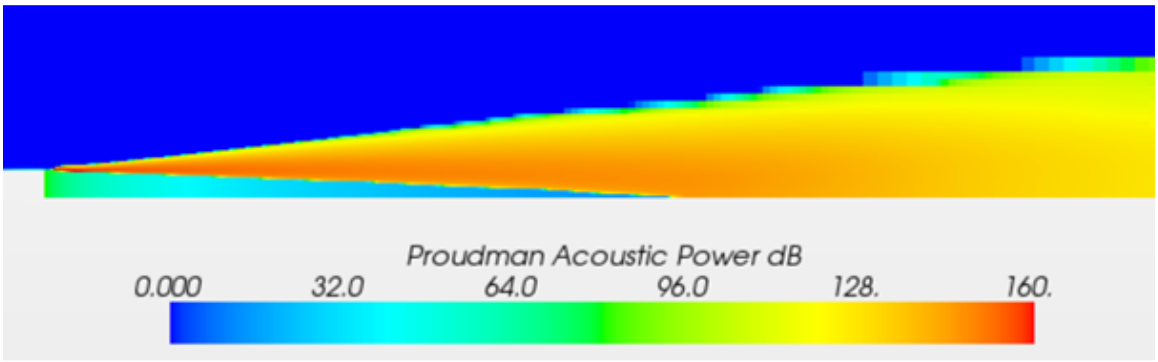

Figure 79: Proudman acoustic power level predictions for 3D supersonic jet CFD investigation #134.

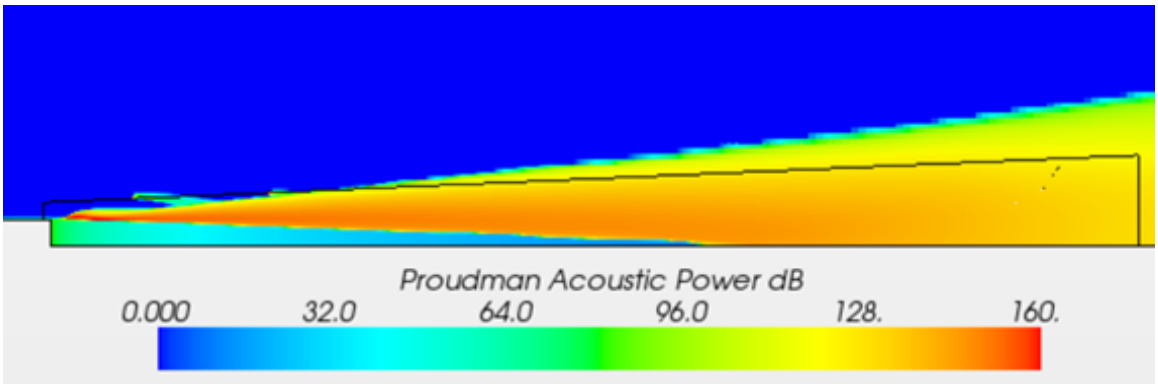

Figure 80: Proudman acoustic power level predictions for 2D supersonic jet CFD investigation #137.

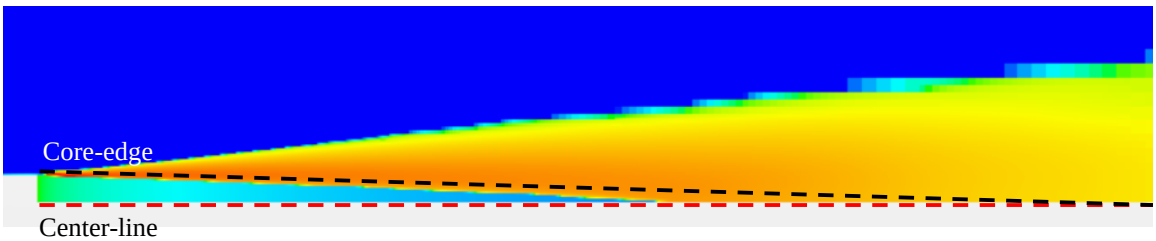

Figure 81: Supersonic jet flow axes used for distributed Proudman sound power level predictions.

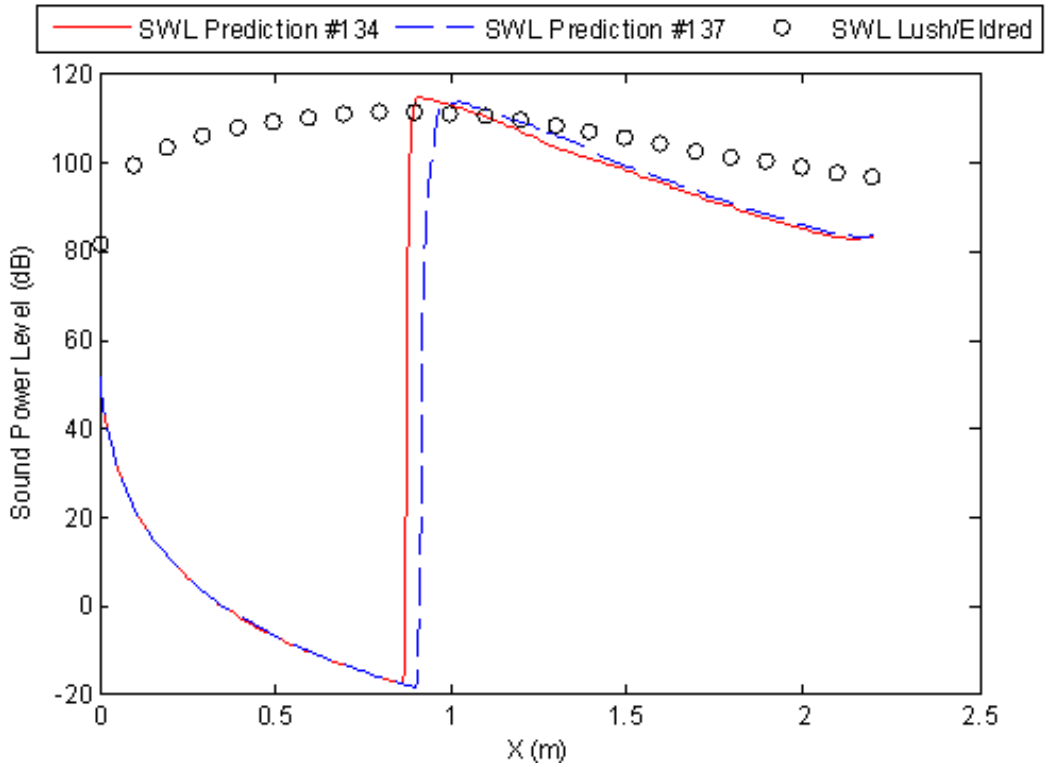

Figure 82: Adjusted Proudman sound power level predictions at the center-line axis.

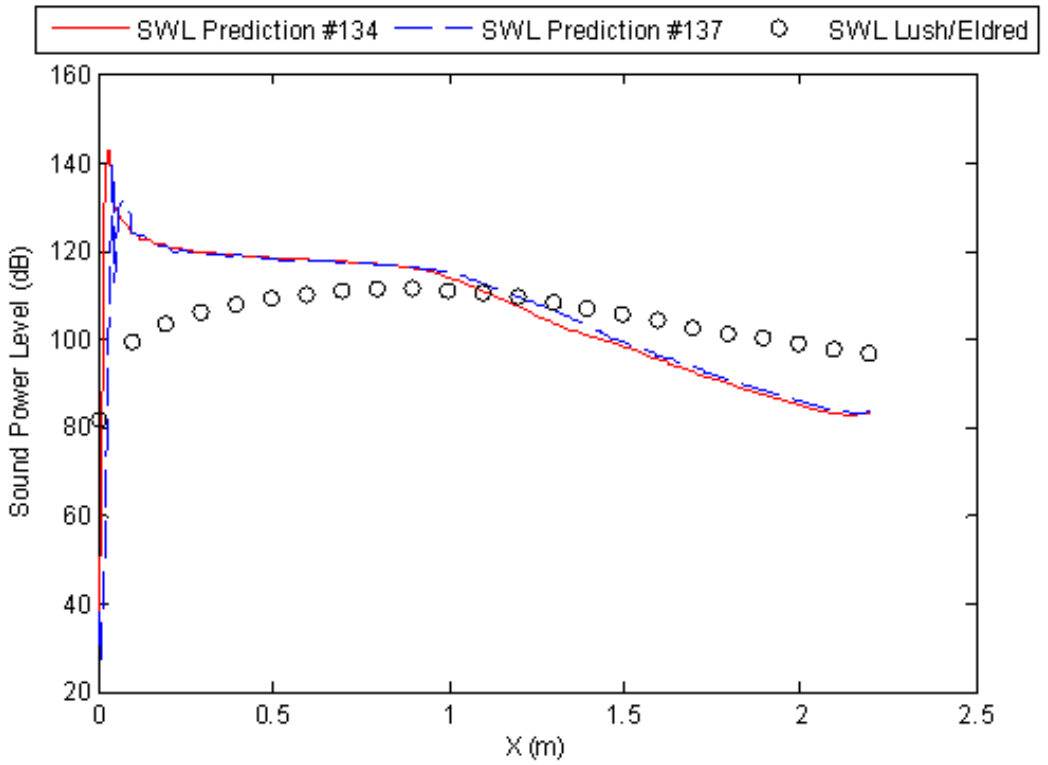

Figure 83: Adjusted Proudman sound power level predictions at the core-edge axis

The time required to complete the 2D steady-state computation was 1.4% of that needed for the 3D model! The adjusted SWL predictions from the 2D CFD investigation #137 along the center-line axis and the 3D CFD investigation #134 along the core-edge axis were selected and replaced for the distributed SWL predicted by Equation (6). At this juncture, the Eldred empirical noise prediction model proceeded using the same methodology outlined in Section 2.1 with the exception of utilizing Wilby's 3rd directivity, Equation (15), from Section 2.2. Figures 84 through 86 shows the noise level predictions based on using the 2D center-line axis RANS / Proudman model for the three primary microphone locations listed in Table 8. As stated in Section 3.4, the three microphone locations are referred to as: prediction location 3.5 m (3.5 m upstream of nozzle exit plane, 3.5 m from the nozzle center-line), prediction location 0.0 m (at nozzle exit plane, 3.5 m from the nozzle center-line), and prediction location -3.5 m (3.5 m downstream of nozzle exit plane, 3.5 m from the nozzle center-line). Figures 87 through 89 shows the noise level predictions for the three microphone locations based on the 3D core-edge RANS / Proudman model. When the noise predictions based on the center-line axis SWL were compared with the noise predictions based on the core-edge axis SWL, the high frequency noise predictions were significantly reduced for the model based on the center-line axis SWL. The reduction in high frequency SPL was due to the reduction in the SWL between the nozzle exit plane and approximately 0.8 m downstream. Between the two RANS / Proudman SWL based noise prediction models, the center-line axis-based model was superior to the core-edge axis-based model.

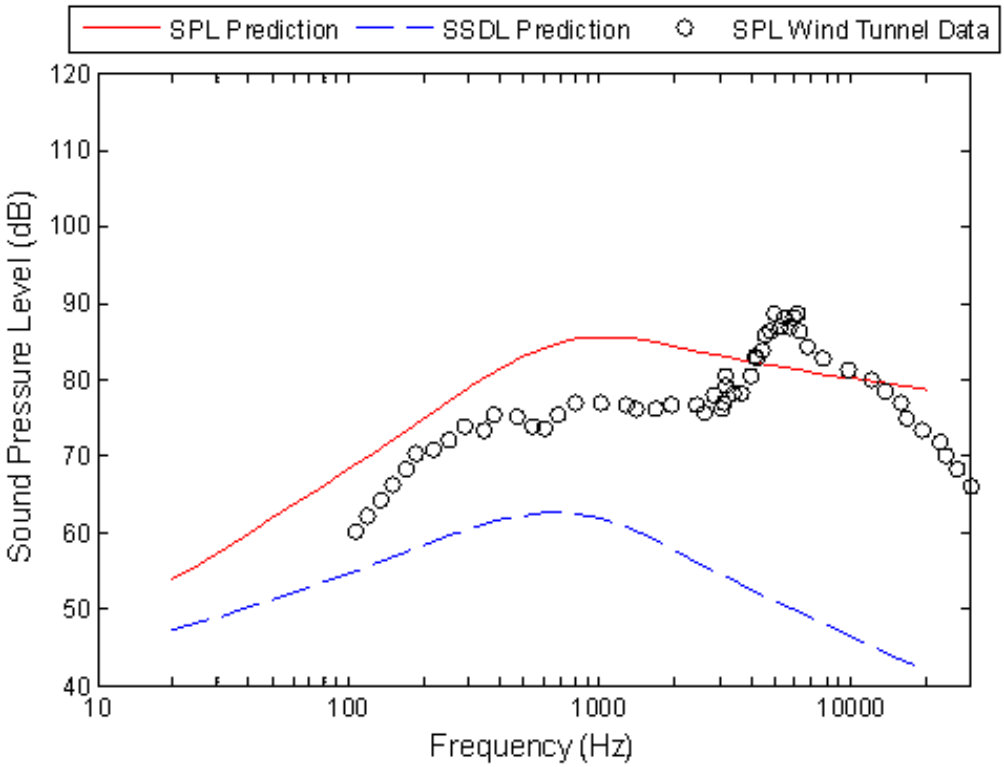

Figure 84: Supersonic jet noise level predictions at 3.5 m using center-line axis SWL from CDF investigation #137.

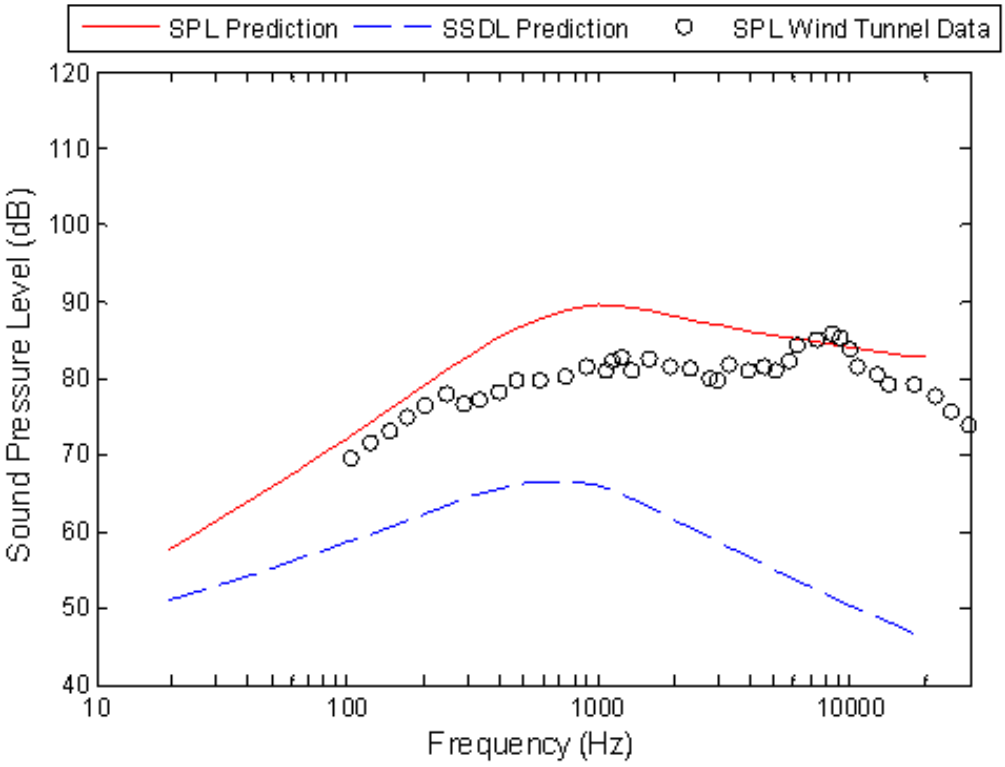

Figure 85: Supersonic jet noise level predictions at 0.0 m using center-line axis SWL from CDF investigation #137.

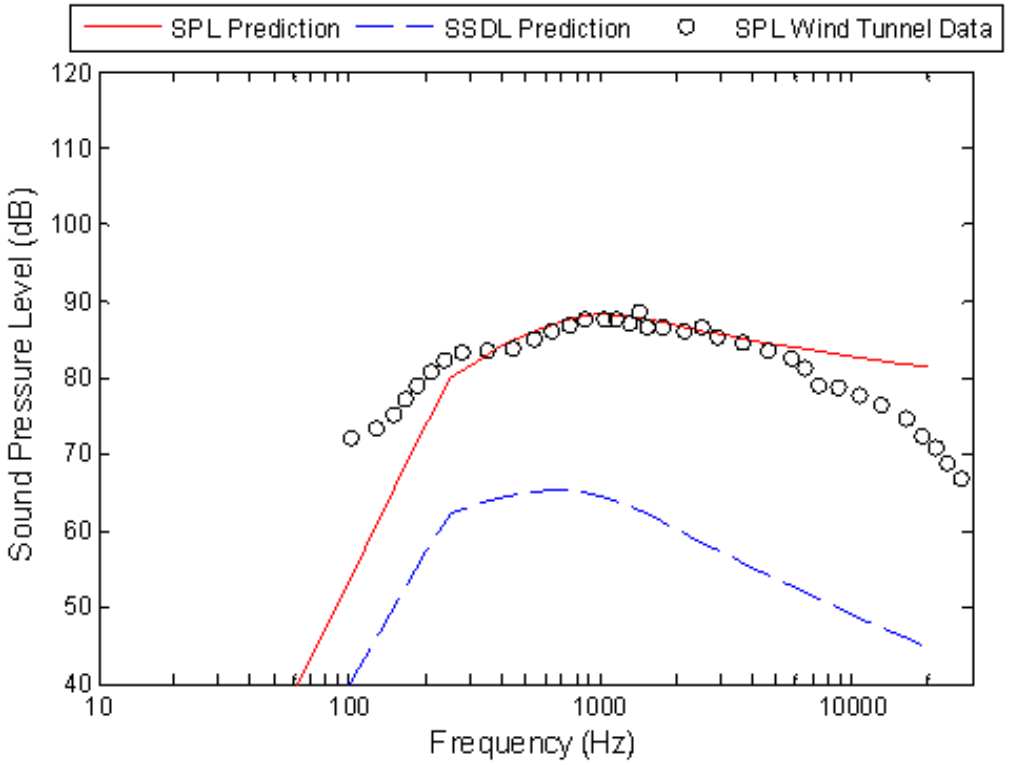

Figure 86: Supersonic jet noise level predictions at -3.5 m using center-line axis SWL from CDF investigation #137.

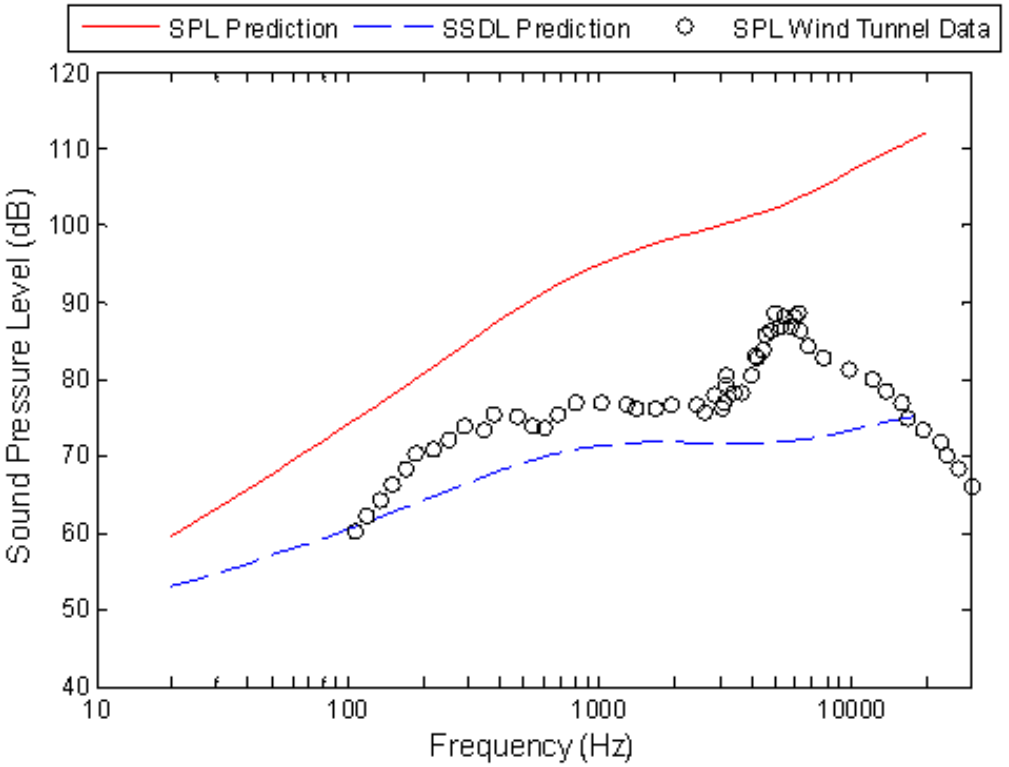

Figure 87: Supersonic jet noise level predictions at 3.5 m using core-edge axis SWL from CDF investigation #134.

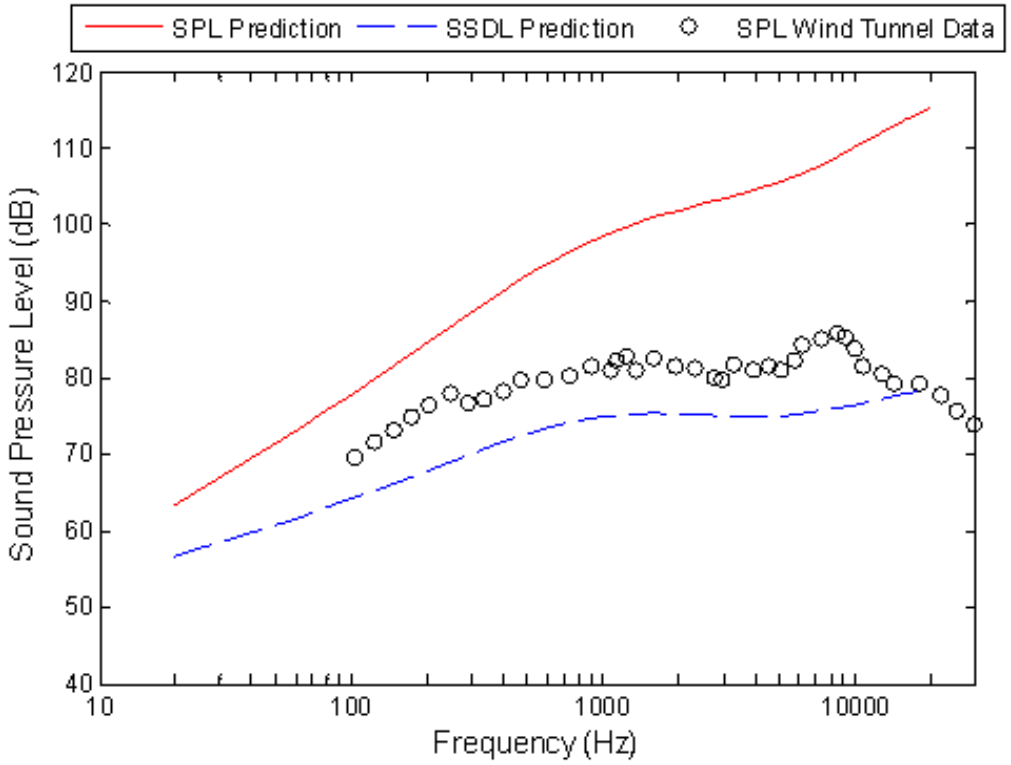

Figure 88: Supersonic jet noise level predictions at 0.0 m using core-edge axis SWL from CDF investigation #134.

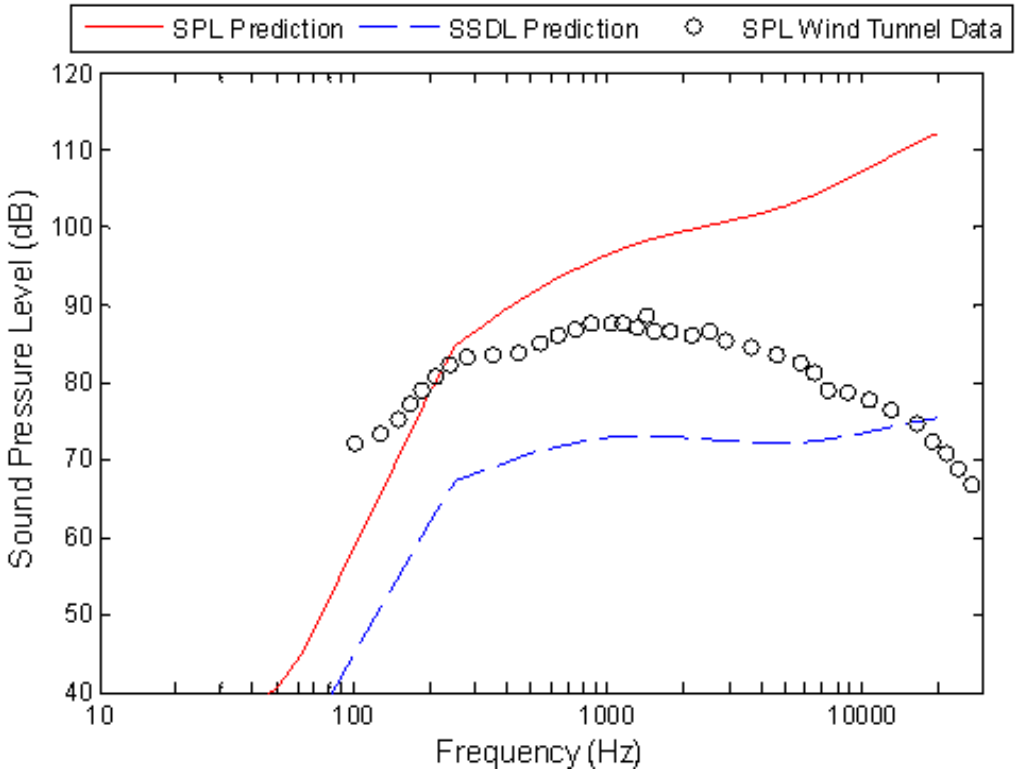

Figure 89: Supersonic jet noise level predictions at -3.5 m using core-edge axis SWL from CDF investigation #134.

The transient CFD investigations utilized the FW-H model to provide SPL predictions. The software used in these CFD investigations, StarCCM+, allows computations for unsteady 2D investigations utilizing only unsteady RANS and not DES. An example of the unsteady RANS / FW-H noise prediction is shown in Figure 90 below, which was for the 0.0 m prediction location. The unsteady RANS / FW-H noise prediction, shown in Figure 90, provided no meaningful information and cannot be used for noise predictions. This research suggests that the mass-averaging of the unsteady Reynolds-averaged Navier-Stokes equations eliminates the small amplitude sound waves. The DES/ FW-H noise predictions, shown in Figures 91 through 93, provided reasonable noise predictions. The OASPL for each of the different supersonic jet noise prediction cases is shown in Table 18. The noise prediction model based on the center-line axis RANS / Proudman produced the most accurate results.

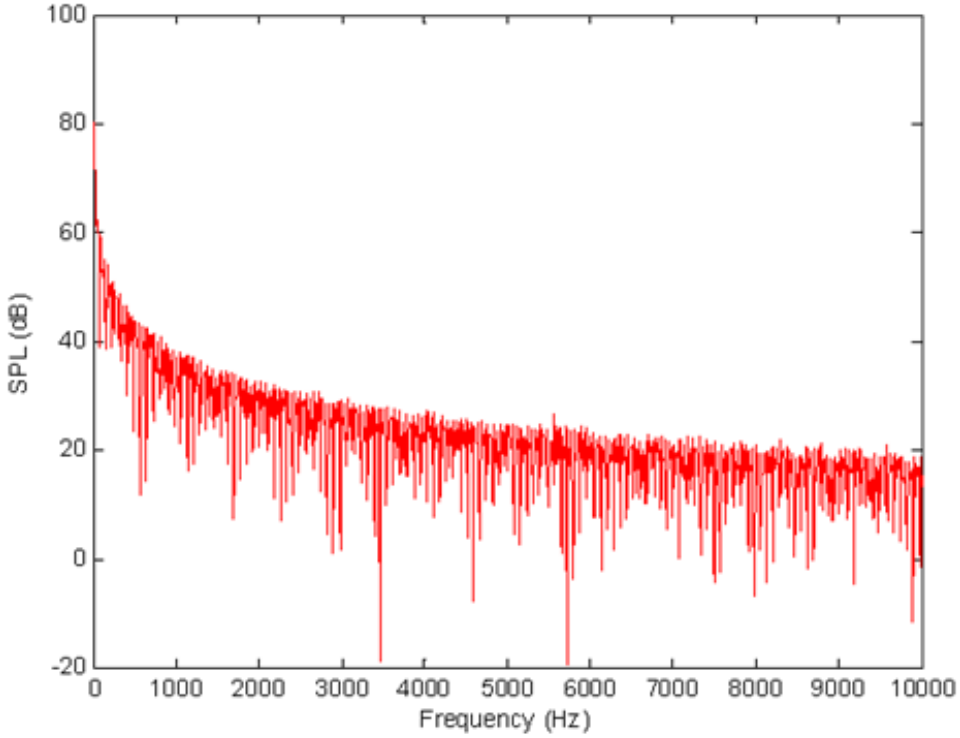

Figure 90: Supersonic jet noise level predictions at 0.0 m using 2D unsteady RANS / FW-H CFD investigation #137.

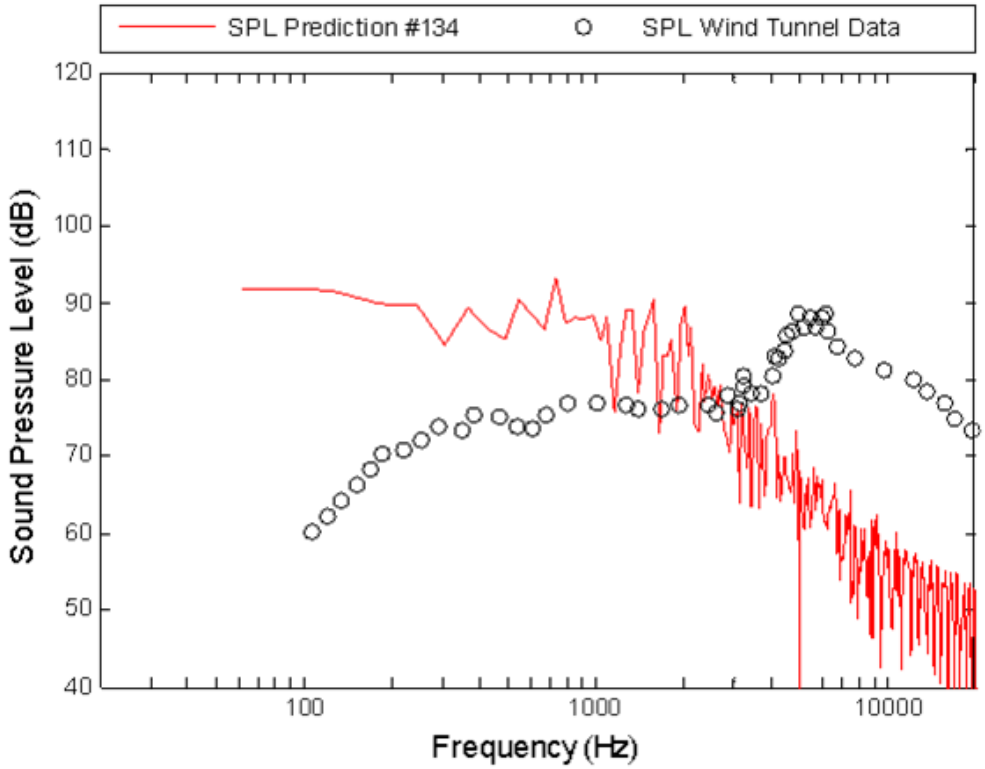

Figure 91: Supersonic jet noise level predictions at 3.5 m using 3D DES / FW-H CFD investigation #134.

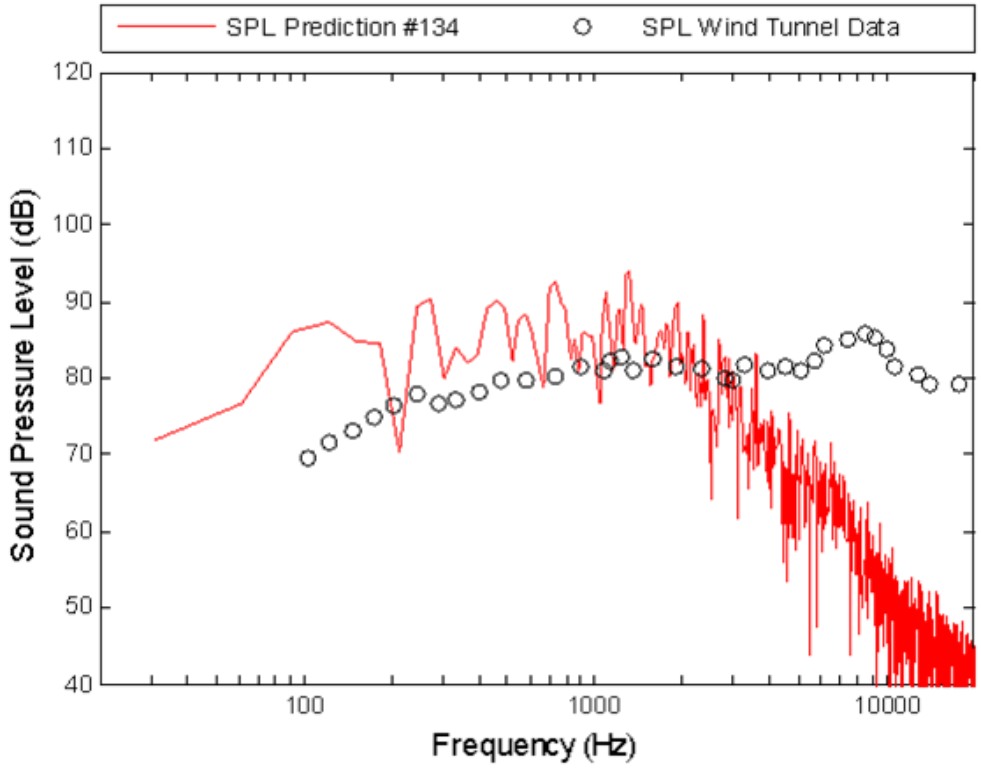

Figure 92: Supersonic jet noise level predictions at 0.0 m using 3D DES / FW-H CFD investigation #134.

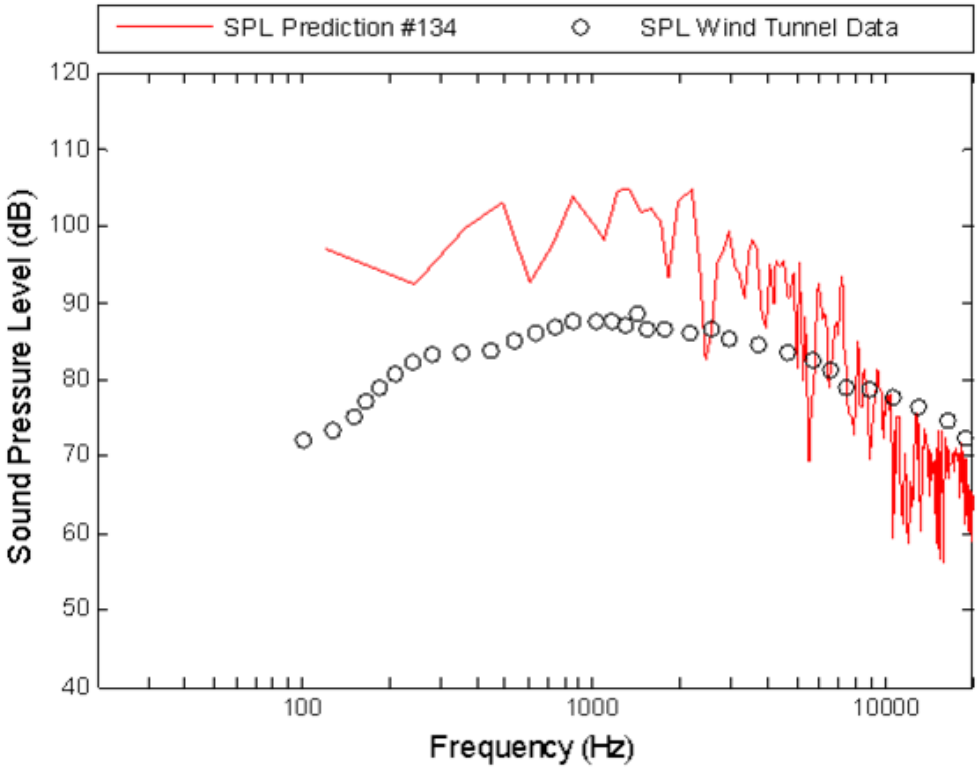

Figure 93: Supersonic jet noise level predictions at -3.5 m using 3D DES / FW-H CFD investigation #134.

Table 18: OASPL noise predictions for a supersonic jet.

		Location: [Z-azis (m)]		
Experiment #	CFD Model / Acoustic Model	[3.5]	[0.0]	[-3.5]
137	2D RANS / Proudman with center-line axis	95.8	99.7	98.3
134	3D RANS / Proudman with core-edge axis	117.3	120.5	117.5
134	DES / FW-H	103.7	106.2	115.5
Empirical	Lush / Eldred	101.0	104.7	102.9
Wind Tunnel	Norum	98.9	96.9	99.4

6 Statistical Energy Analysis Model

The main consideration in the previous four chapters was the prediction and evaluation of high speed rocket exhaust noise. The primary concern was to predict the acoustic loads generated during the launch of a space vehicle. These loads can have a detrimental effect on the vehicle structure itself and the equipment carried inside. This chapter will discuss a method to predict the internal noise based on the statistical energy analysis (SEA) approach. This model could be adapted and combined with the empirical model proposed by Eldred[2] for the prediction of noise transmitted into a space vehicle's interior. The Li and Vipperman[71] model could also be integrated into the Eldred model in a similar way to the Czyz and Gudra[30] model presented in Section 2.3. Available space vehicle launch data inside the surface of space vehicles are unknown at this time to compare with the Czyz and Gudra model. The SEA analytical model provides an avenue to compare exterior and interior noise levels in a laboratory setting.

In the early 1960s, SEA was developed because of the need by the aerospace industry to predict the vibrational response to rocket noise of launch vehicles and their satellite payloads. Although computational methods for predicting the structural vibrational modes were available, the number of degrees of freedom the models could handle with limited computer capabilities at that time were restricted to only a few of the lowest order modes. This posed a serious difficulty because the frequency range of the

significant system response would encompass the natural frequencies of a multitude of higher order modes. An estimate of the number of natural frequencies of the Saturn launch vehicle was approximately 500,000 natural frequencies in the range 0 Hz to 2,000 Hz. The prediction of the precise broadband vibrational response of a complex system such as the Saturn V vehicle was impossible. However, the space, time, and frequency averaged response of the spacecraft vehicle system based on its general physical properties such as geometric shape, dimensions, and material properties was possible.[72]

The general shape of space vehicles consists of finite cylindrical stacked segments. The individual segments can be considered as cylindrical shells that are neither simply-supported nor clamped at each boundary segment. Wang[73] explained that there is little difference in the predicted natural frequencies above the fifth mode between a simply-supported and clamped typical cylindrical shell structure. The vibrational mode shapes that are generated in a cylindrical shell are important due to the coupling between the structural response and the sound transmitted into the cylindrical shell from the exterior broad-band noise. A block diagram of the energy flow for three coupled systems presented by Crocker[74] is shown in Figure 94 below. System (1) has sound energy input, Π_{in1}. System (1) represents an air space where a sound field is generated. There are three possible avenues for the energy to flow from the system (1): the energy can be dissipated in the air space (1), Π_{diss1}; the energy can flow into system (2), Π_{12}; or the energy could flow into system (3), Π_{13}. System (2) represents the cylindrical shell. System (3) represents the air space in the interior of the cylindrical shell. System (2) has two sound energy inputs, Π_{12} and Π_{in2}, where Π_{in2} is zero in this experimental system. There are two possible avenues for the energy to flow from system (2): the energy can be dissipated

in system (2), $\Pi_{diss\,2}$; or the energy can flow into system (3), Π_{23}. System (3) has three sound energy inputs, Π_{13}, Π_{23}, and $\Pi_{in\,3}$, where $\Pi_{in\,3}$ is zero in this model. There is one possible avenue for the energy to flow from system (3), where the energy can be dissipated system, $\Pi_{diss\,3}$.[74] Crocker and Price[75] present a complete derivation for a three coupled system culminating in the derivation of the sound transmission loss (TL) equation for sound transmission through flat plates.

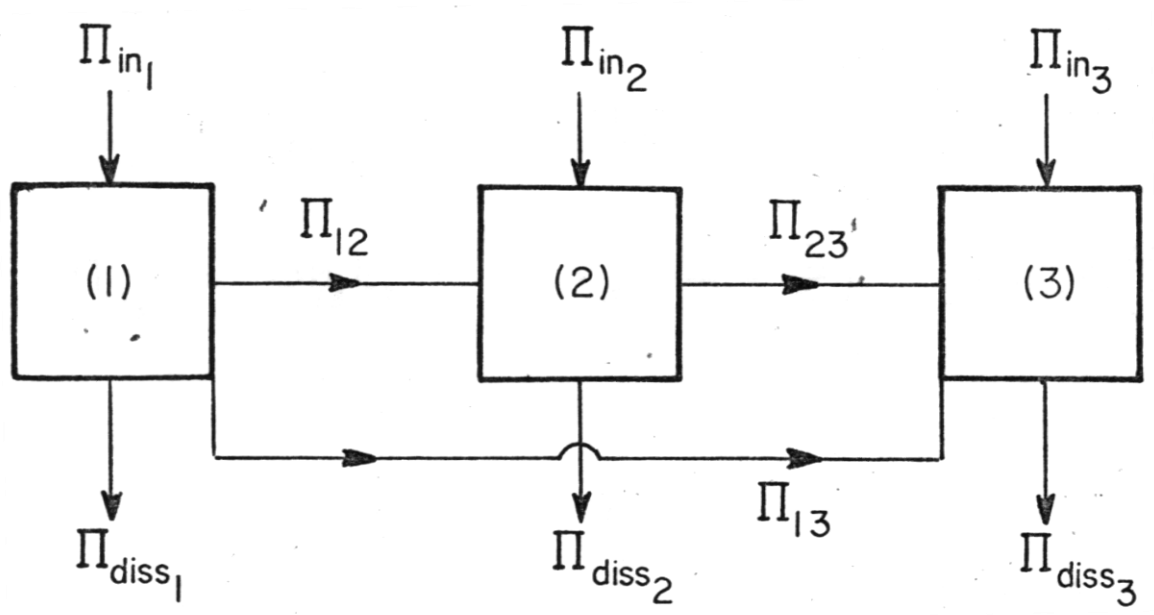

Figure 94: Block diagram of the energy flow for three coupled systems.[74]

For the simplest case of a simple-supported cylindrical shell, the sound transmission through a finite length cylinder was presented by Wang[73] as,

$$TL = 10\log_{10}\left[\frac{\text{Incident Sound Power}}{\text{Transmitted Sound Power}}\right]. \tag{68}$$

Wang states that the transmitted sound power through the cylinder can be considered to be comprised of the sum of two parts: the resonant transmitted power and

the non-resonant transmitted power. The resonant transmitted power was derived by Wang as,

$$TL_{res} = -10\log_{10}\left[\frac{8\pi^2 a_o^2 n(\omega_o) R_{rad}^2}{\omega_o^2 m_s S^2 (2R_{rad} + R_{mech})}\right], \tag{69}$$

where a_o is the speed of sound in air, $n(\omega_o)$ is the modal density in modes per Hz, R_{rad} is the radiation resistance of all the modes that are resonating in a frequency band under consideration, ω_o is the center frequency of each band, m_s is the mass of the cylinder per unit area, S is the radiating surface area, and R_{mech} is the mechanical resistance. The radiation resistance, R_{rad}, was provided by Szechenyi[76] as,

$$R_{rad} = \rho_o a_o \sigma S, \tag{70}$$

where σ is the radiation efficiency. The mechanical resistance, R_{rad}, was provided by Szechenyi as,

$$R_{mech} = S m_s \eta \omega_o, \tag{71}$$

where η is the mechanical energy dissipation loss factor efficiency. The non-resonant transmitted power is presented by Wang as a piecewise defined function as,

$$\begin{aligned} TL_{nr} &= 8.33\log_{10}\left[\left[\nu_o^2 (h/R)^2 E\rho_m / 4\rho_o^2 a_o^2\right]\left[1-(\nu_o\, f_r / f_c)^2\right]^2 + 2.3\right] \\ &\quad -3 + 20\log_{10}\left[\pi/2\sin^{-1}\left[\nu_o\left[1-(\nu_o\, f_r / f_c)^2\right]^{1/2}\right]^{1/2}\right], \\ &\quad \text{for} \nu_o < 1 \text{(below the ring frequency) and} \end{aligned} \tag{72}$$

$$\begin{aligned} TL_{nr} &= 8.33\log_{10}\left[\left[\nu_o^2 (h/R)^2 E\rho_m / 4\rho_o^2 a_o^2\right]\left[1-(\nu_o\, f_r / f_c)^2\right]^2 + 2.3\right] - 3 \\ &\quad \text{for} \nu_o > 1 \text{(above the ring frequency)}, \end{aligned}$$

where v_o is the normalized natural frequency (ω/ω_r), h is the cylinder shell thickness, R is the radius of the cylinder, E is the cylinder's Young's modulus, and ρ_m is the density of the cylinder. The ring frequency, f_r, is given by Wang as,

$$f_r = \frac{1}{2\pi R}\sqrt{\frac{E}{\rho_m}}. \tag{73}$$

The critical frequency, f_c, is given by Fahy[77] as,

$$f_c = \frac{a_o^2\sqrt{12}}{4\pi^2 f_r R h}, \tag{74}$$

where μ is Poisson's ratio. The normalized natural frequencies, v_o, are given by Wang as,

$$v_o = v_{mn} = \frac{\omega_{mn}}{2\pi f_r}\left[\left(k_a^2 + k_c^2\right)^2 + k_a^4/\left(k_a^2 + k_c^2\right)^2\right]^{1/2}, \tag{75}$$

where k_a and k_c are the longitudinal and circumferential wave numbers defined by Wang as,

$$\begin{aligned} k_a &= \frac{m\pi}{L}\left[h^2R^2/12\left(1-\mu^2\right)\right]^{1/4} \text{and} \\ k_c &= \frac{n}{R}\left[h^2R^2/12\left(1-\mu^2\right)\right]^{1/4}. \end{aligned} \tag{76}$$

The variable m is the longitudinal mode number, and the variable n is the circumferential mode number. A select representation of both longitudinal and circumferential mode shapes is shown in Figure 95 below. Wang[73] developed Equation (76) by utilizing Szechenyi's[78] work, which was derived from the Love[79] theory. As stated before, the total transmission power is the sum of the resonant transmitted power and non-resonant transmitted power presented by Szechenyi[76] as,

$$TL = -10\log_{10}\left[\frac{1}{10\text{^}\left(TL_{res}/10\right)} + \frac{1}{10\text{^}\left(TL_{nr}/10\right)}\right], \tag{77}$$

Wang showed that the theoretical transmission loss predicted by his approach compared well with his experimental results. The Matlab code based on Wang's model can be found in Appendix 2.

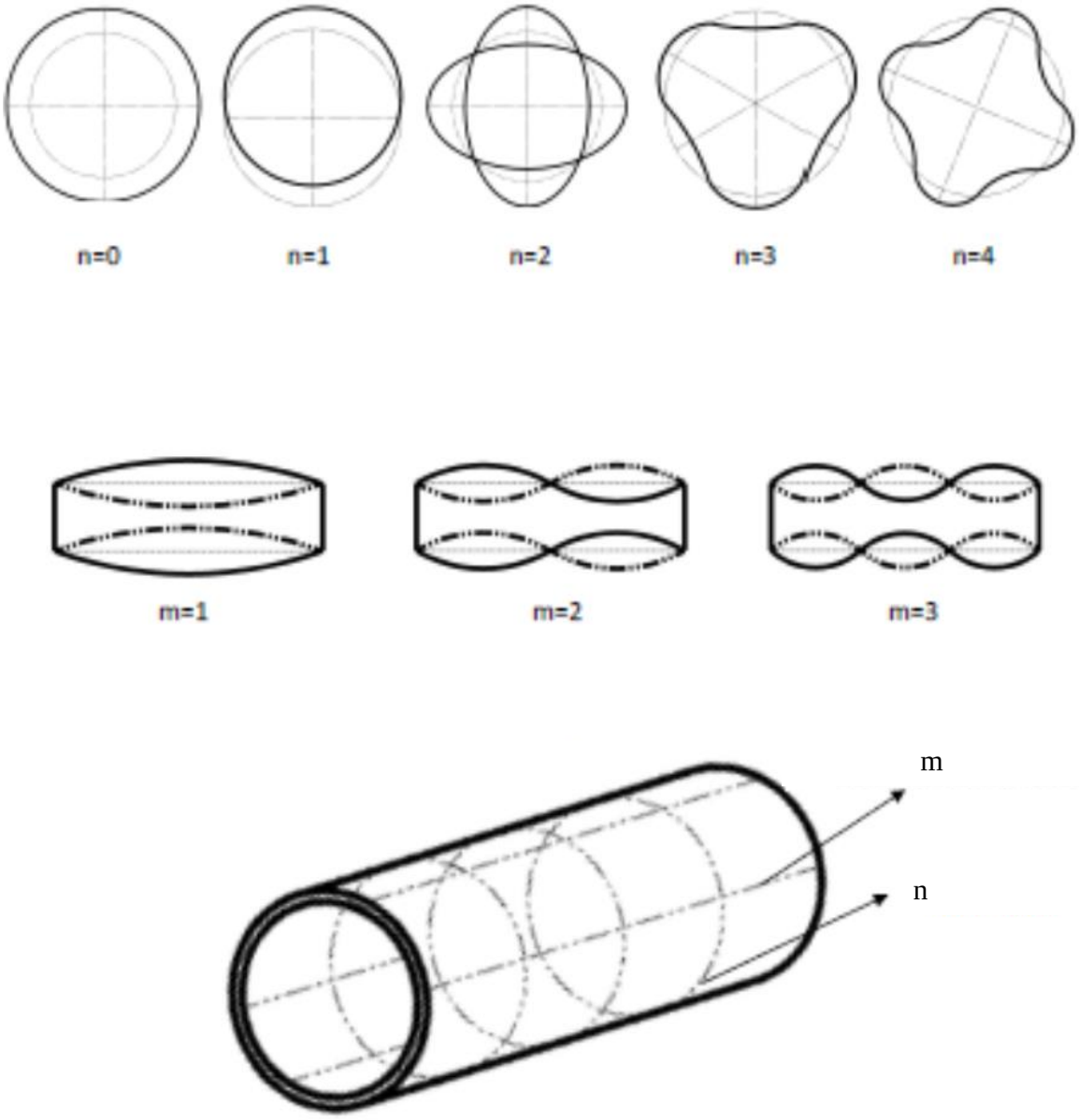

Figure 95: Longitudinal (m) and circumferential (n) mode shapes.[80]

7 Statistical Energy Analysis Model Compared with Measurements on a Cylindrical Structure

Chapter 7 contains the basis for evaluating Wang's statistical energy analysis (SEA) analytical model with two laboratory experiments on a cylindrical structure. Both experiments were conducted with a cylindrical structure in a broad-band noise reverberant field. There are two sections to this chapter: Section 7.1 Vibration Response and Mode Count for a 6 Inch Diameter Aluminium Cylinder and Section 7.2 Noise Transmission Loss Prediction for a 24 Inch Diameter Steel Cylinder and Comparison with Experimental Results.

7.1 Vibration Response and Mode Count for a 6 Inch Diameter Aluminium Cylinder

There are numerous structures that have the basic form of thin-walled cylindrical shells (TWCS). Several examples of TWCS are air conditioning ducts, aircraft fuselages, and liquid storage tanks. However, a long slender cylinder with a relatively thick wall will only have transverse bending (beam like) modes where the distortion of the cross section is negligible. This is especially true for low-frequency vibration of industrial pipes such as those used in petroleum and chemical plants.[80]

When the ratio of the cylinder radius to wall thickness is large, the wave propagation involving the distortion of the cross section at relatively low frequencies is important. An analysis of the vibration characteristics of the TWCS is much more complex

than an analysis of a one-dimensional beam or a two-dimensional plate. The complexity is mainly due to the combination of the equations of motion and the boundary conditions for a cylindrical shell. There have been extensive studies focusing on TWCS. Leissa[81] and Amabili and Paidoussis[82] provide comprehensive reviews of models concerning the vibration of shells.[80]

The thin-walled cylindrical shell used during this experimental investigation is shown in Figure 96 below. The TWCS had an outside diameter of 0.1524 m (six inch) with a wall thickness of 1.4732 mm (0.058 inch). The cylindrical shell was made of type 3004 aluminium alloy with an estimated 2720 kg/m^3 density, 69 GPa Young's modulus, and 0.33 Poisson's ratio. The overall length of the TWCS was 1.8288 m (72 inches) with a 50.8 mm (two inch) wide solid steel cylinder placed inside both ends of the aluminium TWCS. The solid steel cylinder is shown in Figure 97 below. The TWCS was clamped around the steel cylinder using four RK 530 heavy duty motorcycle chains (two on each end) in order to simulate clamped boundary conditions. The inside distance between the two steel clamps was 1.7272 m (68 inches). The TWCS was suspended inside a 53.0 m^3 diffuse (reverberant) chamber using nylon-braided rope.

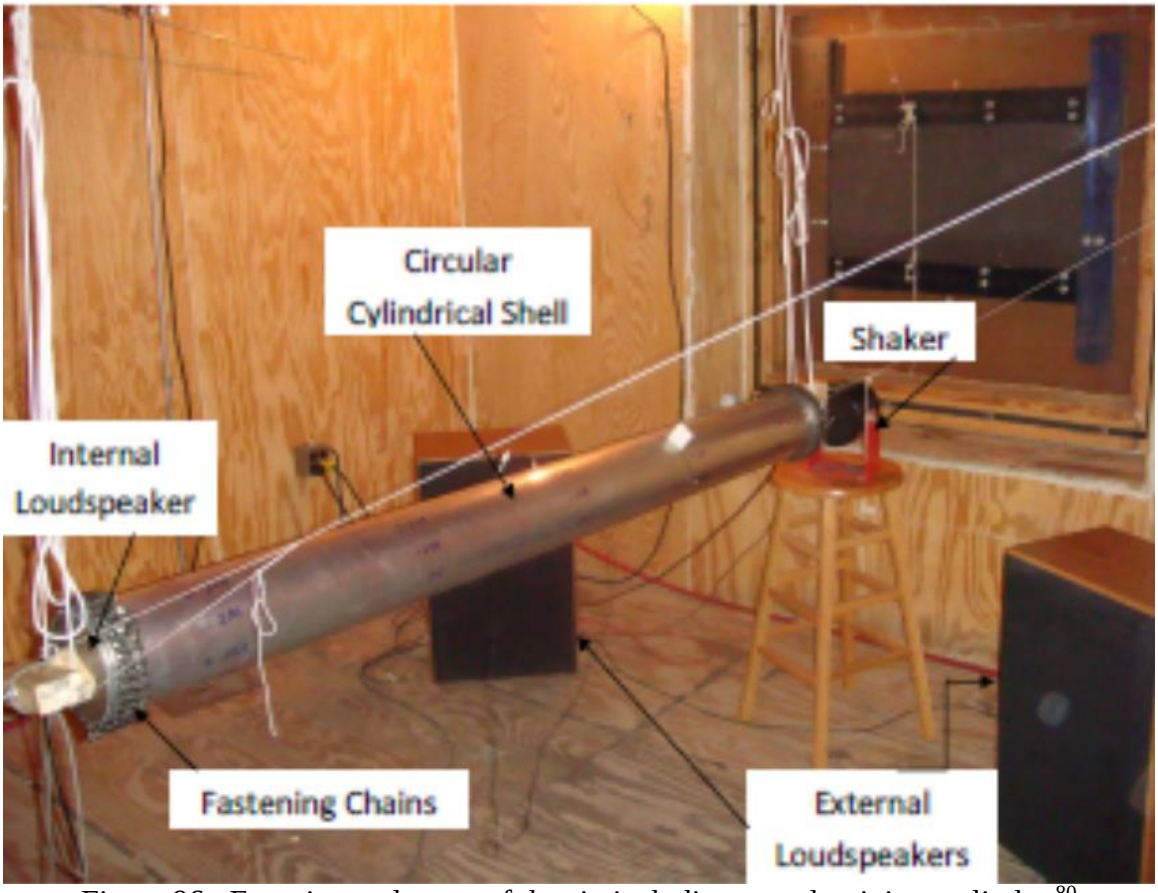

Figure 96: Experimental setup of the six inch diameter aluminium cylinder.[80]

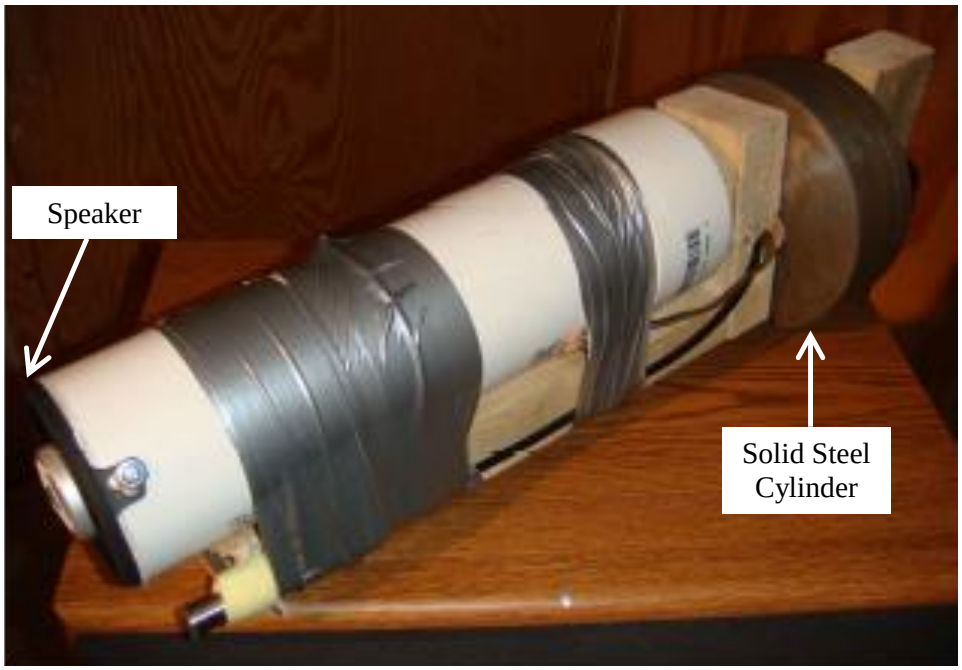

Figure 97: Internal speaker and solid steel cylinder inside the six inch diameter aluminium cylinder.[80]

One objective of this experimental investigation was to study the free vibration of the TWCS. The TWCS was stimulated by the sound generated from the internal loudspeaker, which is shown in Figure 97. The speaker was an Infinity REF3022CF, which is a 3-1/2 inch two-way loudspeaker. The generated surface acceleration was used to determine the individual fundamental mode shapes and natural frequencies on the TWCS. In order to compare mode shapes and frequencies with analytical approximations, five different analytical methods were selected for comparison, which are shown in Table 19 below. These methods include: (1) the Flugge[83] theory; (2) the Love[84] theory; (3) the wave propagation (WP) approach by Zhang[85], which is based on the Flugge theory; (4) the simplified neglect of tangential inertia (NTI) approach by Forsberg[86], which is based on the Flugge theory; and (5) Yu's[87] simplified assumption, which is based on the Flugge theory.[80] Both Flugge's and Love's theories, shown in Table 19, provide reasonable predictions of the individual natural frequencies. Wang[73] effectively utilized Love's theory in his SEA derivations that were discussed previously in Chapter 6.

The SEA of the six inch aluminium cylinder began with a preliminary determination of the ring and critical frequencies. The ring frequency, shown in Equation (73) of Chapter 6, provided a calculated frequency of 10,520 Hz. The critical frequency, shown in Equation (74) of Chapter 6, provided a calculated frequency of 8,742 Hz. Due to the close proximity of the ring and critical frequencies, study of a different size cylinder was necessary to provide perceivable differences from gathered data.

Table 19: TWCS mode frequency experimental comparison with predictions based on five analytical methods.[80]

Longitudinal Wave Parameter	Circumferential Wave Parameter	Natural Frequency (Hz)					
m	n	Experiment	Flugge	Love	WP[1] (Flugge)	Simplified NTI[2] (Flugge)	Simplified Yu (Flugge)
1	1	138.40	138.88	138.93	198.86	197.85	142.63
1	2	190.30	172.83	172.94	194.88	193.40	172.16
2	2	310.50	244.8	244.78	283.27	274.16	245.25
3	2	496.60	423.76	424.15	517.61	476.60	438.75
1	3	502.20	471.38	471.46	498.99	497.03	470.42
2	3	477.00	481.24	481.44	516.22	507.67	477.58
3	3	558.90	514.06	514.47	565.69	542.72	507.41
2	1	464.70	517.95	518.02	778.31	749.05	572.99
4	3	638.30	586.02	586.64	673.61	619.29	580.19
4	2	679.80	687.78	688.23	922.83	775.88	741.75
5	3	782.00	705.25	706.05	871.43	746.05	709.69
6	3	833.80	869.68	870.62	1215.20	920.90	898.84
1	0	842.50	892.07	892.08	892.17	10497.00	-
1	4	884.40	902.20	902.29	932.32	930.14	901.15
2	4	887.00	906.59	906.75	944.07	934.82	902.39
3	4	981.60	917.05	917.34	969.10	945.87	907.77
4	4	945.80	937.82	938.28	1015.90	967.64	922.09
5	4	964.70	973.81	974.47	1097.90	1005.20	951.58

[1] Wave Propagation

[2] Neglect of Tangential Inertia

7.2 Noise Transmission Loss Prediction for a 24 Inch Diameter Steel Cylinder and Comparison with Experimental Results

The thin-walled cylindrical shell used during this experimental investigation is shown in Figure 98 below. The TWCS had an outside diameter of 0.6096 m (24 inch) with a wall thickness of 1.27 mm (0.050 inch). The cylindrical shell was made of galvanized steel metal with an estimated 7850 kg/m^3 density, 200 GPa Young's modulus, and 0.28 Poisson's ratio. The overall length of the TWCS was 2.0574 m (81 inches) with a 19.05 mm (3/4 inch) wide plywood disk placed inside both ends of the steel TWCS. The inside distance between the two plywood disks was 2.0066 m (79 inches). The TWCS was suspended inside a 53.0 m^3 reverberation chamber using nylon-braided rope.

Figure 98: Experimental setup of the 24 inch diameter steel cylinder.

In order to measure the sound transmission loss of the 24 inch diameter steel cylinder, the sound must primarily pass through the cylinder shell. One method used to obtain the data was to place the cylinder in a reverberant sound field. The cylinder ends

were capped with plywood disks to limit the sound transmission through this part of the cylinder. In principle, the sound that travels to the inside of the cylinder will continue to build up if it is not absorbed by sound absorbing material placed inside. A 16" diameter layer of ordinary fiberglass insulation was placed around the central axis of the cylinder. The layer of insulation approximates anechoic conditions inside the cylinder. Sound pressure levels were measured (1) outside and (2) inside of the cylinder near the surface of the cylinder.

The ring frequency and critical frequency were both determined from Equations (73) and (74) in Chapter 6 and from the sound transmission loss plot. The ring frequency, from Equation (73), provided a calculated frequency of 2,636 Hz; the ring frequency determined from the experimental data in Figure 99 provided a frequency of 2,625 Hz. The critical frequency, from Equation (74), provided a calculated frequency of 10,118 Hz; the critical frequency determined from the experimental data in Figure 99 provided a frequency of 10,541 Hz.

The data presented in Figure 99 includes the experimental sound transmission loss (TL) and statistical energy analysis (SEA) prediction of TL. Below the ring frequency, SEA becomes unreliable when few or no resonant modes are present within a frequency band. This unreliability occurred at approximately 700 Hz for the 24 inch steel cylinder. Above 700 Hz, the data is in reasonable agreement between the experimental TL, solid line in Figure 99, and the SEA, dashed line.

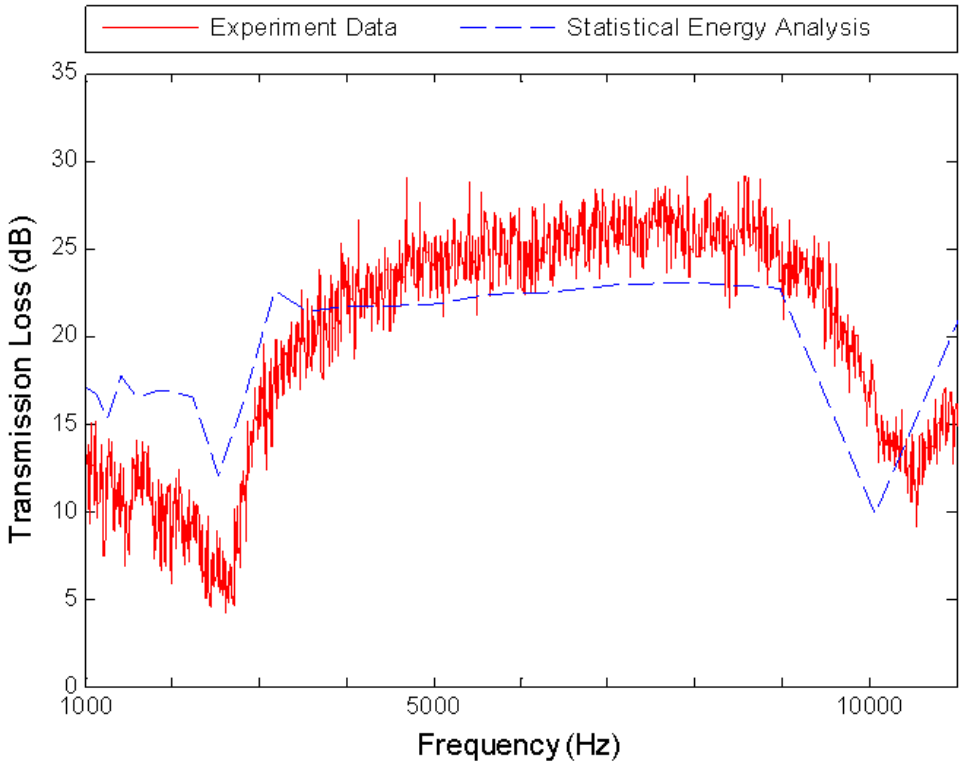

Figure 99: Comparison of the experimental sound transmission loss with SEA for the 24-inch diameter steel cylinder.

8 Summary and Conclusion

The main focus of this dissertation is on the sound pressure levels (SPL) emitted by rocket exhaust flow during the launch of space vehicles. The SPL around the space vehicle can excite the vehicle itself, payload, launch tower, and ground support systems into hazardous and potentially damaging vibration. The large magnitude vibrations in the range of 100 Hz to 10,000 Hz can result in tens or even hundreds of thousands of stress reversals and the possibility of space vehicle fatigue in the first 10 to 20 seconds of flight during lift off and the initial climb off the launch pad. In 1971, Eldred developed two empirical noise predictions models based on the space vehicle engine exhaust flow parameters.

Eldred's two methods for the prediction of far-field SPLs are based on assumed locations of noise sources along the exhaust stream. The first method uses the technique of assigning each frequency band to a specific source location along the flow axis. The frequency bandwidths are arbitrarily assigned in octave, one-third octave, or narrow bands within the audible range from 20 Hz to 20,000 Hz. The second method assumes that the noise in each frequency band within the audible range is generated throughout the flow, instead of at a distinct location. This second method is more difficult to apply than the first method, but it is more realistic when considering the complex nature of the rocket exhaust

flow. Eldred's second method has been shown to be a highly effective launch noise prediction tool.

One important component of the second method is the sound source directivity index. The directivity index for rocket exhaust flows is extremely engine specific. The angle of maximum radiation relative to the exhaust axis is directly related to the increase in the speed of sound of the exhaust flow. Eldred believed that this was the result of the refraction of the sound conveyed through the shear layers of the exhaust gas flow. Eldred included a sound source directivity index for chemical rockets within his original work. Many researchers have proposed new sound source directivity index equations: Bechet; Wilby 1st; Wilby 2nd; Wilby 3rd; and Plotkin and Sutherland. The present research has determined that the third directivity equation proposed by Wilby provides the best correlations with the launch data.

An acoustical effect that was not included in Eldred's original model was sound wave surface diffraction. Sound wave surface diffraction occurs when a sound wave strikes the surface of a space vehicle. When sound waves strike a hard surface such as the external wall of a space vehicle, the sound pressure is increased at the vehicle surface due to the interaction between the incident and reflected sound waves. The reflection of the sound wave by the vehicle surface causes a localized increase in the sound pressure level. The functional relationship for the pressure increase is nonlinear and is dependent upon the two angles of incidence, the frequency of the sound wave, and the diameter of the cylindrical surface it impacts. Czyz and Gudra have provided an equation for these nonlinear effects. Their equation contains an infinite summation of a type two Hankel function. The present research provides a way to include the Czyz and Gudra equation in

Eldred's second model, which leads to an increase in mid-to high-frequency noise prediction levels of between 2 to 6 dB. The low frequency waves generally are not affected by the cylindrical surface because of the difference in dimension between the wavelength and the cylinder diameter. Incorporating both Wilby's third directivity equation and the Czyz and Gudra surface diffraction equation into Eldred's second method provides improved launch noise predictions for the Saturn V, Space Shuttle and Ares I-X space vehicles launch noise data.

The characteristics found in a supersonic jet flow field are similar to those found in a rocket's exhaust flow during launch. Norum performed some experimental wind tunnel research on jet exhaust noise. His primary interest was the noise reduction that could be obtained using water injection. Using Eldred's original second method on a supersonic jet leads to a gross over-prediction of the SPL for the supersonic jet due to the excessively high assumed sound power level. A much lower sound power level (SWL) is estimated using the Lush equation for subsonic jets. This research provides reasonable noise predictions when Eldred's second method is used in conjunction with Lush's sound power equation and Wilby's third directivity equation.

The Eldred second method relies fundamentally on empirical assumptions about the size and strength of the locations of sound sources. Another approach studied utilized computational fluid dynamics (CFD) to predict the sound sources. In general, obtaining the necessary pressure field fluctuations from CFD for the noise predictions is an extremely difficult task. In order to carry out the task, a simple pure-tone sound wave traveling through free space was modeled using the inviscid Euler equations. During the research, spatial and temporal resolutions were adjusted until the computational errors were

minimized. The data show that 100 cells per wavelength and 100 time steps per wave period up to the frequency of interest are able to reduce the loss in sound pressure level to less than 0.8 decibel (dB) over a distance of 1 m. The loss in energy in the sound wave due to computational errors was neither linear nor asymptotic. In the prediction, by the time a 10,000 Hz sound wave travelled 19.6 m in an inviscid Euler environment, the wave had decreased by 1.8 dB by using 140 cells per wavelength and 140 time steps per wave period.

The simplest inviscid Euler flow model described the propagation of a sound wave but not the generation of sound from turbulent flows. A detached eddy simulation (DES) of turbulent flow provides an abundance of information but requires prolonged initialization of the domain before the prediction of sound propagation can be made. A shorter route to this initialization issue is to run a steady state Reynolds-averaged Navier-Stokes (RANS) model first, and then transition the method to the unsteady DES. This allows for faster building of a CFD model for the simple turbulent flow over a cylinder to allow prediction of the noise generated using the DES equations.

In order to extract the acoustical information from the transient DES, the choice between the two types of Ffowcs Williams – Hawkings (FW-H) surfaces, impermeable or permeable, was shown to be an important factor for accurate noise prediction. The FW-H predictions along the vertical direction using the permeable surface were quite accurate when compared with the modified version of Blevins' empirical equation. The model studied assumed a permeable FW-H surface completely surrounding the dominant noise producing turbulence. The FW-H prediction method demonstrated that the discrete data collected from the turbulent flow required at least ~1.0×10^{-2} seconds (s) of data in order

to minimize the error in the data. With ~3.2 x 10^{-2} s of data (2^{15} samples), the fast Fourier transform showed an improvement in the frequency resolution. The research has determined that at least 3 wave length periods were needed in order to resolve the lowest frequency of interest. The flow over a cylinder model provided a foundation for building a more complex supersonic jet flow field.

The supersonic jet wind tunnel CFD investigation applied the knowledge gained from the previous studies towards a simplified version of a space vehicle in order to predict launch noise. The simplified version was based on Norum's supersonic jet research. The RANS CFD model coupled with the Proudman acoustical model provided limited noise predictions. The Proudman model can be used to predict the sound power level (SWL) within the CFD domain. Utilizing this acoustical model, the SWL were predicted along the jet's primary flow axis and compared with the distributed SWL for the Lush / Eldred empirical noise model. This research compared the distributed SWL predicted by the two-dimensional (2D) steady-state RANS / Proudman model with the 3D steady-state RANS / Proudman model and found that the SWL were nearly the same. The time required to complete the 2D steady-state RANS / Proudman computation was 1.4% of the time needed for the 3D RANS / Proudman model! Although the SWL were scaled down by 30 dB, the RANS / Proudman approach provided a method for comparison with the Norum's wind tunnel data without the need to run a 3D DES / FW-H model. A 2D RANS / Proudman calculation required 0.3% of the time required that was needed for the 3D DES / FW-H approach.

With the success of noise predictions based on the RANS / Proudman and the DES / FW-H calculations, this research is at a juncture where the development of a more

complete understanding of the effect of water injection on rocket launch noise could be possible. An initial effort could be focused on comparing Norum's supersonic jet wind tunnel data with data from a 2D RANS / Proudman model. Norum has published acoustical data from subsonic and supersonic jets without water injection and with water injection using various methods of water injection. After the computation models with water injection have been evaluated using a 2D RANS / Proudman model, a genetic optimizer algorithm could be used to predict the reduction in the noise levels. While the Proudman model approach will not provide the frequency content of the noise, it has the capacity to provide a single parameter to be minimized by optimizer algorithm. The parameter could use a simple volume average of the SWL in the predominant noise producing region, which would provide an estimate of the overall sound pressure level. Several water injection methods may be important to consider in reducing the noise levels. The main focus of the optimization could include: 1) water / exhaust mass flow rate ratio, 2) axial injection location(s), 3) water injection angle, 4) number of injectors, method of injection (jet type or spray type), 5) water droplet size, 6) water injection velocity, and 7) functional relation between water injection rate and the exhaust frequency content. Optimal water injection methods are needed to more effectively control the aero-acoustic environment at the launch of a space vehicle and thereby reduce hazardous and potentially damaging structural vibration. This research can also be extended to further study the noise predictions of a rocket exhaust flow during the launch of a space vehicle.

The prediction of an average broadband vibrational response of the system such as the Saturn V vehicle is possible based on its general physical properties. For the first five modes, Love's theory was in good agreement with experimentally predicted mode

frequencies. Wang explained that there is little difference in the predicted natural frequencies above the fifth mode between a simply-supported and clamped cylindrical shell structures. Utilizing Wang's statistical energy analysis (SEA) approach, the sound transmission loss (TL) for the cylindrical shell is seen to be in reasonable agreement with the experimental sound transmission loss.

References

[1] Canabal, F. & Frendi A. (2006). Study of the Ignition Overpressure Suppression Technique by Water Addition. Journal of Spacecraft and Rockets, Vol. 43, No. 4, pp. 853-865.

[2] Eldred, K. M. (1971). Acoustic Loads Generated by the Propulsion System (NASA Technical Note SP-8072).

[3] Robertson, J. E. (1971). Prediction of In-flight Fluctuating Pressure Environments Including Protuberance Induced Flow (Report WR 71-10). Wyle Laboratories.

[4] Onawola, O. O. (2008). A Feedback Linearization Approach for Panel Flutter Suppression with Piezoelectric Actuation (Dissertation). Auburn University, Auburn, AL.

[5] Canabal, F. (2004). Suppression of the Ignition Overpressure Generated by Launch Vehicles (Dissertation). University of Alabama at Huntsville, Huntsville, AL.

[6] Hartfield, R. & Crocker M. (2009). Assessment of Launch Vehicle Ascent Pressure Fluctuation Levels. National Aeronautics and Space Administration NASA Engineering Safety Center.

[7] Timmins, A. R. (1974). A Study of First-Month Space Malfunctions (NASA Technical Note D-7750).

[8] Timmins, A. R. & Heuser, R. E. (1971). A Study of First-Day Space Malfunctions (NASA Technical Note D-6474).

[9] Thomas, G. R., Fadick, C. M., & Fram B. J. (2005). Launch Vehicle Payload Adapter Design with Vibration Isolation Features. 12th Annual SPIE International Symposium on Smart Structures and Materials.

[10] Krausse, S. C. (1969). Saturn V Aerothermodynamics Flight Evaluation Summary – AS-501 Through AS -503 (Document number D5-15796-1). Boeing Company Space Division Launch Systems Branch, Aerophysics Group.

[11] Hill, P. & Peterson, C. (1992). Mechanics and Thermodynamics of Propulsion (Second Edition). Prentice Hall, Upper Saddle River, NJ, pp. 594-598.

[12] Kandula, M. (2009). Broadband shock noise reduction in turbulent jets by water injection. Applied Acoustics, Vol. 70, pp.1009–1014.

[13] Krothapalli, A., Venkatakrishnan, L., Lourenco, L., Greska, B., & Elavarasan, R. (2003). Turbulence and Noise Suppression of a High-speed Jet by Water Injection. Journal of Fluid Mechanics, Vol. 491, pp. 131-159.

[14] Lighthill, M. J. (1952). On Sound Generated Aerodynamically, I. General Theory. Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences, Vol. 211, pp. 564–587.

[15] Lighthill, M. J. (1954). On Sound Generated Aerodynamically, II. Turbulence as a Source of Sound. Proceedings of the Royal Society of London, Series A, Mathematical and Physical Sciences, Vol. 222, pp. 1–32.

[16] Kandula, M. & Vu, B. (2003). On the Scaling Laws for Jet Noise in Subsonic and Supersonic Flow. American Institute of Aeronautics and Astronautics.

[17] Ffowcs Williams, J. E. (1963). The Noise from Turbulence Convected at High Speed. Philosophical Transactions for the Royal Society of London, Series A, Mathematical and Physical Sciences, Vol. 255, Issue 1061, pp. 469-503.

[18] Tam, C. K. W. (1972). On the Noise of a Nearly Ideally Expanded Supersonic Jet. Journal of Fluid Mechanics, Vol. 51, Part 1, pp. 69-95.

[19] Goldstein, M.E. & B. Rosenbaum. (1973). Effect of Anisotropic Turbulence on Aerodynamic Noise. Journal of Acoustical Society of America, Vol. 54, pp. 630-645.

[20] Proudman, I. (1952). The Generation of Noise by Isotropic Turbulence. Proceedings of the Royal Society of London, Series A, Mathematical and Physical Sciences, Vol. 214, Issue 1116, pp. 119-132.

[21] Ffowcs Williams, J. & Hawkings, D. (1969). Sound Generation by Turbulence and Surfaces in Arbitrary Motion. Philosophical Transactions of Royal Society of London, Series A, Mathematical and Physical Sciences, Vol. 264, Issue 1151, pp. 321-342.

[22] Keegan, W. B. (2001). Dynamic Environmental Criteria (NASA report # HDBK-7005). NASA Technical Handbook.

[23] Kinsler, L. E. & Frey A. R. (1962). Fundamentals of Acoustics (Second Edition). John Wiley & Sons, New York, NY.

[24] Wilby, J. F. (2005). Liftoff Noise (Technical Memorandum 179-04).

[25] Plotkin, K. J. & Sutherland, L. C. (2007). Prediction of the Acoustics Environment Induced by the Launch of Ares I Vehicle (Report WR 07-27). Wyle Laboratories.

[26] Ribner, H.S. (1964). The Generation of Sound by Turbulent Jets. Advances in Applied Mechanics, Vol. 8, pp. 104-178.

[27] Plumblee, H. E., Ballentine, J. R., & Passinos B. (1967). Near field noise analysis of aircraft propulsion systems with emphasis on prediction techniques (Technical Report AFFDL TR 67-43).

[28] Sutherland, L.C. (1968). Sonic and Vibration Environments for Ground Facilities - A Design Manual (Report WR 68-2). Wyle Laboratories.

[29] Wiener, F. M. (1947). Sound Diffraction by Rigid Spheres and Circular Cylinders. Journal of the Acoustical Society of America, Vol. 19, Num. 3.

[30] Czyz, H. & Gudra T. (1992). Force Due to Diffraction of Sound Wave on Small Diameter Cylindrical Particles. Journal De Physique III, Vol. 2, pp. 741-744

[31] Boeing Company. (1967). F-1 Engine. Saturn V News Reference, pp. 3-1 – 3-10.

[32] Murphy, W. (1991). Environment and Test Specification Levels Ground Support Equipment for Space Shuttle System Launch Complex 39, SSME/ASRM Volume III, Acoustics (NASA report # GP-1059 Volume III).

[33] Griffin, L. (2007). Space Shuttle Solid Rocket Booster Lift-off Acoustic Data, (Memorandum for Record, ER42(07-040)).

[34] Counter, D. D. (2012). Verification of Ares I Liftoff Acoustic Environments via the Ares I Scale Model Acoustic Test. 27th Aerospace Testing Seminar.

[35] Matsos, H. (2012). Gallery_Image_6433. Retrieved from http://www.astrobio.net/images/galleryimages_images/Gallery_Image_6433.jpg

[36] Haynes, J. M. (2009, January 16). Shuttle and Ares I Parameters / Geometry, (electronic personal communication).

[37] Norum, T. D. (2004). Reductions in Multi Component Jet Noise by Water Injection. 10th AIAA/CEAS Aeroacoustics Conference.

[38] Street R. L., Watters, G. Z., & Vennard, J. K. (1996). Elementary Fluid Mechanics (Seventh Edition). John Wiley & Sons, New York, NY.

[39] Lush, P. A. (1971). Measurement of Subsonic Jet Noise and Comparison with Theory. Journal of Fluid Mechanics, Vol. 46, pp. 477-500.

[40] Tannehill, J. C., Anderson, D. A., & Pletcher, R. H. (1997). Computational Fluid Mechanics and Heat Transfer (Second Edition). Taylor & Francis, Philadelphia, PA.

[41] White, F.M. (2001). Fluid Mechanics (Fourth Edition). McGraw Hill, Boston, MA.

[42] Fourier, J. B. J. (1822). Théorie analytique de la chaleur (The Analytic Theory of Heat). Libraires Pour Les Mathématiques, Paris, France.

[43] Steger, J. L. & Warming, R. F. (1979). Flux Vector Splitting of the Inviscid Gasdynamic Equations with Application to Finite-Difference Methods (NASA TM D-78605).

[44] Shelton, A. B. (Spring, 2011). AERO 7140: Computational Fluid Dynamics, Computer Project 2, Rudimentary Euler Solver. Auburn University, Auburn, AL.

[45] Green, G. (1828). An Essay on the Application of Mathematical Analysis to the Theories of Electricity and Magnetism. T. Wheelhouse, Nottingham, England.

[46] Roe, P. L. (1981). Approximate Riemann Solver, Parameter Vectors and Difference Schemes. Journal of Computational Physics, Vol. 43, pp. 357-372.

[47] van Leer, B. (1982). Flux Vector Splitting for the Euler Equations. Proceedings of the 8th International Conference on Numerical Methods in Fluid Dynamics, Lect. Notes Phys., Vol. 170 Springer-Verlag, New York, NY, pp. 507-512.

[48] Roe, P.L. (1986). Characteristic-based schemes for the Euler equations. Annual Review of Fluid Mechanics, Vol. 18, pp. 337–365.

[49] Liou, M-S. (1996). A Sequel to AUSM: AUSM+. Journal of Computational Physics, Vol. 129, pp. 364-382.

[50] Reynolds, O. (1895). On the Dynamical Theory of Incompressible Viscous Fluids and the Determination of the Criterion. Philosophical Transactions of the Royal Society of London. Vol. 186, pp. 123-164.

[51] Favre, A (1965). Equation des Gaz Turbulents Compressibles: 1. Formes Générales, Jour. de Mecanique, Vol. 4, pp. 361-390.

[52] Ghuge, H. (2007). Detached Eddy Simulations of a Simplified Tractor Trailer Geometry (Thesis). Auburn University, Auburn, AL.

[53] Jones, W.P. & Launder, B.E. (1972). The Prediction of Laminarization with a Two-Equation Model of Turbulence, International Journal of Heat and Mass Transfer, Vol. 15, pp. 301-314.

[54] Wilcox, D.C. (1998). Turbulence Modeling for CFD (Second Edition), DCW Industries Inc., La Canada, CA.

[55] Menter, F.R. (1994). Two-equation eddy-viscosity turbulence modeling for engineering applications. American Institute of Aeronautics and Astronautics Journal, Vol. 32, Num. 8, pp. 1598-1605.

[56] CD-adapco. (2012). STAR-CCM+ User Guide (Version 7.02.008). CD-adapco, Orlando, FL.

[57] Aldama, A. A. (1990). Filtering Techniques for Turbulent Flow Simulation, Lecture Notes Engineering, Vol. 56. Springer-Verlag, New York, NY.

[58] Blazek, J. (2007). Computational Fluid Dynamics: Principles and Applications (Second Edition). Elsevier Ltd., Amsterdam, Netherlands.

[59] Brentner, K. S. & Farassat F. (2003). Modeling Aerodynamically Generated Sound of Helicopter Rotors. Progress in Aerospace Sciences, Vol. 39, pp. 83-120.

[60] Shur, M. L., Spalart, P. R., Strelets, M. Kh., & Travin, A. K. (2003). Towards the Prediction of Noise from Jet Engines. International Journal of Heat and Fluid Flow, Vol. 24, pp. 551-561.

[61] Fukuda, K., Tsutsumi, S., Fujii, K., Ui, K., Ishii, T., Oinuma, H., Kazawa, J., and Minesugi, K. (2009). Acoustic Measurement and Prediction of Solid Rockets in Static Firing Tests. 15th AIAA/CEAS Aeroacoustics Conference.

[62] Reinelt, R. (2004). Acoustics with FLUENT (presentation). CFD-Engineer, Automotive & Aerospace Team, ANSYS Inc, Darmstadt, Germany.

[63] Wagner, C.A., Huttl, T. and Sagaut, P. (2007). Large Eddy Simulation for Acoustics. Cambridge University Press, New York, NY.

[64] Lienhard, J.H. (1966). Synopsis of Lift, Drag, and Vortex Frequency Data for Rigid Circular Cylinders. Washington State University, Pullman, WA.

[65] Blevins, R.D. (2001). Flow Induced Vibration (Second Edition). Krieger Publishing Company, Malabar, FL.

[66] Gerrard, J.H. (1955). Measurements of the Sound from Circular Cylinders in an Air Stream. Proceedings of the Physical Society, Section B, Volume 68, Number 7, pp. 453-461.

[67] Gerrard, J.H. (1961). An Experimental Investigation of the Oscillating lift and Drag of a Circular Cylinder Shedding Turbulent Vortices. Journal of Fluid Mechanics, Vol. 11, part 2, pp. 244-256.

[68] Mallock, A. (1907). On the Resistance of Air. Proceedings of the Royal Society of London, Series A, Vol. 79, pp. 262–265.

[69] Ayers III, J.T. (2005). Hydrodynamic Drag and Flow Visualization of IsoTruss® Lattice Structures (Thesis). Brigham Young University, Provo, UT.

[70] Branscomb, D., Gurley, A., Beale, D., and Broughton, R. (2012). Open-Architecture Composite Tube Design and Manufacture. 2012 ASME Early Career Technical Conference, Atlanta, Georgia.

[71] Li, D. and Vipperman, J.S. (2005). Mathematical Model for Characterizing Noise Transmission into Finite Cylindrical Structures. Journal of the Acoustical Society of America, Vol. 117, Num. 2.

[72] Fahy, F. (1994). Statistical Energy Analysis: A Critical Overview. Philosophical Transactions of the Royal Society of London: Physical Sciences and Engineering, Vol. 346, No. 1681, pp. 431-447.

[73] Wang, Y.S. (1982). Transmission of Sound through a Cylindrical Shell and a Light Aircraft Fuselage (Dissertation). Purdue University, West Lafayette, IN.

[74] Crocker, M.J. (1969). The Response of Structures to Acoustic Excitation and the Transmission of Sound and Vibration (Dissertation). University of Liverpool, Liverpool, UK.

[75] Crocker, M.J. and Price, A.J. (1969). Sound Transmission Using Statistical Energy Analysis. Journal of Sound and Vibration, Vol. 9, No. 3, pp. 469-486.

[76] Szechenyi, E. (1971). Sound Transmissions through Cylinder Walls Using Statistical Energy Considerations. Journal of Sound and Vibration, Vol. 19, No. 1, pp. 83-94.

[77] Fahy, F. (1985). Sound and Structural Vibration: Radiation, Transmission, and Response. Academic Press, Orlando, FL.

[78] Szechenyi, E. (1971). Approximate Methods for the Determination of the Natural Frequencies of Stiffened and Curved Plates. Journal of Sound and Vibration, Vol. 14, No. 1, pp. 401-418.

[79] Love, A.E.H. (1954). A Treatise on the Mathematical Theory of Elasticity (Forth Edition). Dover Publication, New York, NY.

[80] Farshidianfar, A., Farshidianfar, M.H., Crocker, M.J., and Smith, W.O. (2011).Vibration Analysis of Long Cylindrical Shells Using Acoustical Excitation. Journal of Sound and Vibration, Vol. 330, No. 14, pp. 3381-3399.

[81] Leissa, W. (1973). Vibration of Shells (NASA SP-288). US Government Printing Office, Washington, DC.

[82] Amabili, M. and Paidoussis, M.P. (2003) Review of studies on geometrically nonlinear vibrations and dynamics of circular cylindrical shells and panels, with and without fluid structure interaction. Applied Mechanics Reviews, Vol. 56, No. 4, pp. 349-381.

[83] Flugge, W, (1973). Stresses in Shells. Springer, New York, NY.

[84] Love, A.E.H. (1888) The Small Free Vibrations and Deformation of a Thin Elastic Shell. Philosophical Transactions of the Royal Society of London, Series A, Vol. 179, pp. 491–546.

[85] Zhang, X.M., Liu, G.R., and Lam, K.Y. (2001) Vibration Analysis of Thin Cylindrical Shells Using Wave Propagation Approach, Journal of Sound and Vibration, Vol. 239, No. 3, pp. 397-403.

[86] Forsberg, K. (1965). A Review of Analytical Methods Used to Determine the Modal Characteristics of Cylindrical Shells (NASA CR-613). National Aeronautics and Space Administration, Washington, DC.

[87] Yu, Y.Y. (1955). Free Vibrations of Thin Cylindrical Shells Having Finite Lengths with Freely Supported and Clamped Edges. Journal of Applied Mechanics, Vol. 22, pp. 547-552.

Appendix 1a Eldred Empirical Model Matlab Code (part a):

Acoustic_Loads_2_13.m

```
function Acoustic_Loads_2_13
% Acoustic Loads 2.13:
% Main rewrite Programmer (Aug 08): Wesley O. Smith III
      (wesley.o.smith.iii@outlook.com)
% Ares I        Parameter (Feb 09): Karen Kirk (kirk.karen.d@gmail.com)
% Directivity  Programmer (Nov 08): Wilson Everett
      (weverettwilson@yahoo.com)
% Original     Programmer (Feb 08): Cedric Bechet
      (cedric.bechet@hotmail.fr)
%
% Version Updates:
% 2.13 (Summer 2013)
% 1: New curve fit in function "fill_DI_r_meth_2" and "fill_DI_meth_2"
%    (Eldred, NASA SP-8072, p. 12, Fig. #10)

% 2.09: (Spring 2009)
% 1: Shuttle programed into GUI
% 2: Found mistake in "xt" eq. in func. "approach2" (Eldred, NASA SP-
      8072,
%    p. 13, Fig. 11)
% 3: New curve fit in function "fill_relative_sound_power" (Eldred,
      NASA
%    SP-8072, p. 13, Fig. #12)
% 4: New curve fit in function "fill_relative_sound_power_spectrum"
%    (Eldred, NASA SP-8072, p. 14, Fig. #13)
% 5: Add many comments to the code
% 6: Removed most "Excel" and old "save" commands that had been
      commented out
% 7: Fixed errors in the mic. location data so that more than one mic.
%    location can be run at any one time
% 8: Ares I programed into GUI
% 9: Saturn V and Shuttle launch data included in plots
% 10: Loop added to so both the Shuttle and Boosters predictions are
      added
%     together
%       (Summer 2009)
% 11: Saturn V launch and mic. parameteres updated
% 12: Shuttle mic. parameteres updated
% 13: Added pressure double programing
% Known error: can't use the "user defined mic locations".
%
% 2.08: (Fall 2008)
% 1: Complete rebiuld of the GUI from 2.0
% 2: Commented out all matric save commands
% 3: Removed requirement of running Excel
% 4: Found mistake in "valn" (# of nozzle) it was set to 1 in function
%    "calculbt2_Callback"
% 5: Add Directivity Equations added
% 6: Found error in Directivity programed by Everett, he used "log"
      (base e)
```

```
%    instead of "log10" (base 10)
% 7: Saturn V programed into GUI
% Known error: can't run more than one mic. location at a time.
%
% 2.0:
% Original noise perdiction program built by Cedric Bechet based on
% K. M. Eldred report "Acoustic Loads Generated by the Propulsion
     System"
% (NASA SP-8072) June 1971

close all
clear all
clc

% Creates and then hide the GUI (figure window) as it is being
      constructed.
main_window = figure(...
    'Visible','off',...
    'Units','normalized',...
    'Position',[0.05 0.05 0.9 0.85],...
    'Tag','Acoustic_Load_main_window',...
    'Resize','off',...
    'Color','blue',...
    'NumberTitle','off',...
    'MenuBar','none',...
    'Name','Acoustic Load 2.13');

% Construct the components.
% Creates push button at the bottom of first screen.
% Note that the menu items are built in the the "enter_Callback"
      function.
uicontrol(...
    'Style','pushbutton',...
    'Units','normalized',...
    'Position',[0.375 0.10 0.25 0.05],...
    'Tag','enter_pushbutton',...
    'Callback',{@Enter_Callback},...
    'FontSize',16,...
    'ForegroundColor','black',...
    'String','ENTER');

% Creates main text at the top of first screen.
uicontrol(...
    'Style','text',...
    'Units','normalized',...
    'Position',[0.175 0.80 0.65 0.18],...
    'FontSize',16, ...
    'ForegroundColor','red',...
    'BackgroundColor','blue',...
    'Tag','main_text',...
    'String',{'Ground System Structural Vibration at Launch',...
    'Project Supported by NASA/Glenn Research Center',...
    'Auburn University Sound and Vibration Research Laboratories',...
    'August 2006 - August 2013'});

%Create a plot in the axes.
```

```
% width 256 pixels by height 226 pixels
set(main_window,'Units','pixels');
main_window_size = get(gcf,'Position');
scale_factor = 0.6.*main_window_size(1,4)./226;
AU_Logo_norm_width = 256.*scale_factor./main_window_size(1,3);
AU_Logo_width_start = 0.5-AU_Logo_norm_width./2;
AU_Logo_position = [AU_Logo_width_start 0.2 AU_Logo_norm_width 0.6];
set(main_window,'Units','normalized');
axes(...
    'Units','normalized', ...
    'Position',AU_Logo_position);
% Creates first data (image) to plot.
image(imread('aulogo.jpg'),'Tag','AU_Logo');
axis 'off'

% Make the GUI visible.
set(main_window,'Visible','on');

%.....................................................................
      ....

function Enter_Callback(hObject, eventdata)
% When "ENTER" in pushed the currently function is called.
global test_a test_b
test_a = hObject;
test_b = eventdata;

delete(findobj(gcf,'Tag','enter_pushbutton'));

%Create a plot in the axes.
% width 437 pixels by height 386 pixels
set(main_window,'Units','pixels');
main_window_size = get(gcf,'Position');
scale_factor = 0.6.*main_window_size(1,4)./386;
AU_Logo_norm_width = 437.*scale_factor./main_window_size(1,3);
AU_Logo_width_start = 0.5-AU_Logo_norm_width./2;
AU_Logo_position = [AU_Logo_width_start 0.2 AU_Logo_norm_width 0.6];
set(main_window,'Units','normalized');
axes(...
    'Units','normalized', ...
    'Position',AU_Logo_position);
% Creates second data (image) to plot.
image(imread('launcher.jpg'),'Tag','AU_Logo');
axis off;

uicontrol(...
    'Style','text',...
    'Units','normalized',...
    'Position',[0.42 0.15 0.35 0.05],...
    'FontSize',10, ...
    'ForegroundColor','red',...
    'BackgroundColor','blue',...
    'Tag','main_figure_text',...
    'String','(Eldred, NASA SP-8072, Figure 17, 1971)');
```

```
main_menu = uimenu(...
     'Label','File',...
     'Tag','main_menu');
uimenu(main_menu,...
     'Label','First Method (NOT completed)',...
     'Callback',{@First_Method_Multi_Exhaust_Panel_Callback});
uimenu(main_menu,...
     'Label','Second Method',...
     'Callback',{@Second_Method_Multi_Exhaust_Panel_Callback});
uimenu(main_menu,...
     'Label','Ground reflection properties (NOT completed)',...
     'Callback',{@Ground_Reflection_Panel_Callback});
uimenu(main_menu,...
     'Label','Sum of different exhaust nozzles (NOT completed)',...
     'Callback',{@Sum_Exhaust_Panel_Callback});
uimenu(main_menu,...
     'Label','Exit',...
     'Callback',{@Exit_Callback});

exhaust_menu = uimenu(...
     'Label','File',...
     'Tag','exhaust_menu');
uimenu(exhaust_menu,...
     'Label','Back to Main Menu',...
     'Callback',{@Back_Main_Menu_Callback});
uimenu(exhaust_menu,...
     'Label','Exit',...
     'Callback',{@Exit_Callback});
set(exhaust_menu,'Visible','off')

help_menu = uimenu(...
     'Label','Help',...
     'Tag','help_menu');
uimenu(help_menu,...
     'Label','Acoustic Loads 2.08 Help',...
     'Callback',{@Help_Callback});
uimenu(help_menu,...
     'Label','About Acoustic Loads 2.08 (NOT completed)',...
     'Callback',{@About_Callback});

end

%.....................................................................
      ....

function First_Method_Multi_Exhaust_Panel_Callback(hObject, eventdata)
% Calculate using first method
global test_a test_b
test_a = hObject;
test_b = eventdata;

set(findobj(gcf,'Tag','main_menu'),'Visible','off');
set(findobj(gcf,'Tag','exhaust_menu'),'Visible','on');
set(findobj(gcf,'Tag','AU_Logo'),'Visible','off');
set(findobj(gcf,'Tag','main_figure_text'),'Visible','off');
set(findobj(gcf,'Tag','main_text'),'Visible','off');
```

```
Not_Complete_Text
end

%.....................................................................
      ....

function Second_Method_Multi_Exhaust_Panel_Callback(hObject, eventdata)
% Calculate using second method
global test_a test_b
test_a = hObject;
test_b = eventdata;

global shuttle_and_booster
shuttle_and_booster=1;

set(findobj(gcf,'Tag','main_menu'),'Visible','off');
set(findobj(gcf,'Tag','exhaust_menu'),'Visible','on');
set(findobj(gcf,'Tag','AU_Logo'),'Visible','off');
set(findobj(gcf,'Tag','main_figure_text'),'Visible','off');
set(findobj(gcf,'Tag','main_text'),'Visible','off');

data_panel = uipanel( ...
    'Visible','off',...
    'Title','Vehicle Selection', ...
    'ForegroundColor','red',...
    'FontSize',16,...
    'BackgroundColor','blue',...
    'Tag','data_panel',...
    'Units','normalized',...
    'Position',[0.05,0.05,0.9,0.9]);

%               [ St-W, St-H,    W,    H]
text_left     = [-0.10  0     0.10  0.02];
text_bottom   = [-0.10 -0.03  0.20  0.02];
input_right   = [ 0     0     0.10  0.02];
data_button   = [ 0.03 -0.02  0.10  0   ];

table_title_l = [ 0.02  0.09  0.08  0.06];
table_title_r = [ 0.12  0.09  0.08  0.06];

table_title_1 = [ 0.00  0.04  0.04  0.05];
table_title_2 = [ 0.04  0.04  0.04  0.05];
table_title_3 = [ 0.08  0.04  0.04  0.05];
table_title_4 = [ 0.12  0.04  0.04  0.05];
table_title_5 = [ 0.16  0.04  0.04  0.05];

table_column_1 = [ 0.00 -0.11  0.04  0.17];
table_column_2 = [ 0.04 -0.11  0.04  0.17];
table_column_3 = [ 0.08 -0.11  0.04  0.17];
table_column_4 = [ 0.12 -0.11  0.04  0.17];
table_column_5 = [ 0.16 -0.11  0.04  0.17];

% Which data five button group:
data_button_section = uibuttongroup(...
```

```
    'Parent',data_panel,...
    'BackgroundColor','blue',...
    'Units','normalized',...
    'Position',[0.025 0.92 0.95 0.05],...
    'Tag','data_button_section',...
    'Title','',...
    'SelectionChangeFcn',{@Data_Button_Callback});

% Saturn V:
uicontrol(...
    'Parent',data_button_section,...
    'Style','radiobutton',...
    'Units','normalized',...
    'Position',[0.01 0.05 0.10 0.90],...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'Tag','saturn_v_data',...
    'ForegroundColor','white',...
    'Value',1,...
    'UserData',1,...
    'String','Saturn V');

% Shuttle:
uicontrol(...
    'Parent',data_button_section,...
    'Style','radiobutton',...
    'Units','normalized',...
    'Position',[0.27 0.05 0.08 0.90],...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'Tag','shuttle_data',...
    'ForegroundColor','white',...
    'UserData',2,...
    'String','Shuttle');

% Shuttle Booster:
uicontrol(...
    'Parent',data_button_section,...
    'Style','radiobutton',...
    'Units','normalized',...
    'Position',[0.36 0.05 0.08 0.90],...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'Tag','booster_data',...
    'ForegroundColor','white',...
    'UserData',3,...
    'String','Booster');

% Shuttle with Booster:
uicontrol(...
    'Parent',data_button_section,...
    'Style','radiobutton',...
    'Units','normalized',...
    'Position',[0.45 0.05 0.08 0.90],...
    'FontSize',10, ...
    'BackgroundColor','blue',...
```

```
    'Tag','booster_data',...
    'ForegroundColor','white',...
    'UserData',4,...
    'String','Both');

% Ares I:
uicontrol(...
    'Parent',data_button_section,...
    'Style','radiobutton',...
    'Units','normalized',...
    'Position',[0.58 0.05 0.10 0.90],...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'Tag','ares_data',...
    'ForegroundColor','white',...
    'Value',0,...
    'UserData',5,...
    'String','Ares I');

% Super Sonic Jet:
uicontrol(...
    'Parent',data_button_section,...
    'Style','radiobutton',...
    'Units','normalized',...
    'Position',[0.80 0.05 0.10 0.90],...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'Tag','ares_data',...
    'ForegroundColor','white',...
    'Value',0,...
    'UserData',6,...
    'String','Sonic Jet');

% User defined data:
uicontrol(...
    'Parent',data_button_section,...
    'Style','radiobutton',...
    'Units','normalized',...
    'Position',[0.90 0.05 0.10 0.90],...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'Tag','user_defined_data',...
    'ForegroundColor','white',...
    'UserData',9,...
    'String','User Defined');

%.............................................
% Saturn V Data:
%.............................................

% Microphone data:
position_list = [0.02 0.89 0.20 0.02];
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list,...
```

```
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Microphone Location Z-axis (m)');

position_list = [0.04 0.865 0 0];
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','saturn_mic_1',...
    'Value',0,...
    'String','06.0',...
    'Callback',@Mic_Data_Button_Callback);
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right+data_button,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','saturn_mic_1_data',...
    'Value',0,...
    'Enable','off',...
    'String','Data (mic. #B0002-115)');

position_list = [0.04 0.82 0 0];
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','saturn_mic_2',...
    'Value',0,...
    'String','22.0',...
    'Callback',@Mic_Data_Button_Callback);
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right+data_button,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','saturn_mic_2_data',...
    'Value',0,...
    'Enable','off',...
    'String','Data (mic. #B0003-118)');
```

```
position_list = [0.04 0.775 0 0];
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','saturn_mic_3',...
    'Value',0,...
    'String','45.2',...
    'Callback',@Mic_Data_Button_Callback);
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right+data_button,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','saturn_mic_3_data',...
    'Value',0,...
    'Enable','off',...
    'String','Data (mic. #B0005-200)');

position_list = [0.04 0.73 0 0];
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','saturn_mic_4',...
    'Value',0,...
    'String','66.3',...
    'Callback',@Mic_Data_Button_Callback);
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right+data_button,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','saturn_mic_4_data',...
    'Value',0,...
    'Enable','off',...
    'String','Data (mic. #B0003-219)');

position_list = [0.04 0.685 0 0];
uicontrol(...
    'Parent',data_panel,...
```

```
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','saturn_mic_5',...
    'Value',0,...
    'String','84.5',...
    'Callback',@Mic_Data_Button_Callback);
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right+data_button,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','saturn_mic_5_data',...
    'Value',0,...
    'Enable','off',...
    'String','Data (mic. #B0013-426)');

position_list = [0.04 0.64 0 0];
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','saturn_mic_6',...
    'Value',0,...
    'String','85.3',...
    'Callback',@Mic_Data_Button_Callback);
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right+data_button,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','saturn_mic_6_data',...
    'Value',0,...
    'Enable','off',...
    'String','Data (mic. #B0001-601)');

position_list = [0.04 0.595 0 0];
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
```

```
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','saturn_mic_9',...
    'Value',0,...
    'String','User Defined',...
    'Callback',@Data_Button_Callback);

% F: Thrust of each Engine
position_list = [0.11 0.57 0 0];
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_left,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','F (N)');
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_bottom,...
    'FontSize',10,...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','(thrust of each engine)');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','6672333');

% Te: Engine Exit Temperature
position_list = [0.11 0.50 0 0];
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_left,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Te (K)');
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_bottom,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','(exit temperature)');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
```

```
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','3572');

% de: Engine Exit Diamemter
position_list = [0.11 0.43 0 0];
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_left,...
    'FontSize',10,...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','de (m)');
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_bottom,...
    'FontSize',10,...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','(exit nozzle diamemter)');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','3.53');

% Ue: Engine Exit Velosity
position_list = [0.11 0.36 0 0];
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_left,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Ue (m/s)');
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+[-0.10 -0.03 0.2 0.03],...
    'FontSize',10,...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','(fully expanded exit velosity)');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
```

```
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','2585');

engine_data_x = {'-3.27','-3.27','3.27','3.27','0'}; % (m)
engine_data_y = {'-3.27','3.27','3.27','-3.27','0'};
engine_data_z = {'0','0','0','0','0'};
engine_number_string = 5;

ramp_data_y_z={'22','22','22','22','22'};
ramp_data_x_y={'0','0','0','0','0'};

position_list = [0.01 0.17 0 0];
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_title_l,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String',{'Engine','Position'});

uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_title_1,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','X');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_column_1,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'Max',engine_number_string,'Min',0,...
    'String',engine_data_x);

uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_title_2,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Y');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
```

```
    'Position',position_list + table_column_2,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'Max',engine_number_string,'Min',0,...
    'String',engine_data_y);

uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_title_3,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Z');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_column_3,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'Max',engine_number_string,'Min',0,...
    'String',engine_data_z);

uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_title_r,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String',{'Apparent','Flow Angle'});

uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_title_4,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','X-Y');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_column_4,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'Max',engine_number_string,'Min',0,...
    'String',ramp_data_x_y);

uicontrol('Parent',data_panel,...
    'Style','text',...
```

```
    'Units','normalized',...
    'Position',position_list + table_title_5,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Y-Z');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_column_5,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'Max',engine_number_string,'Min',0,...
    'String',ramp_data_y_z);

%.........................................
% Shuttle Data:
%.........................................

% Microphone data:
position_list = [0.25 0.89 0.20 0.02];
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Microphone Location');

position_list = [0.28 0.865 0.10 0];
% Left SRB Nozzle Shirt Edge (mic. # 7986)
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','shuttle_mic_1',...
    'Value',0,...
    'String','Left SRB Aft Skirt Edge',...
    'Callback',@Mic_Data_Button_Callback);
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right+data_button,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','shuttle_mic_1_data',...
    'Value',0,...
```

```
    'Enable','off',...
    'String','Data (mic. # 7986)');

position_list = [0.28 0.82 0.10 0];
% Left SRB Above Nozzle Shirt Edge (mic. # 7987)
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','shuttle_mic_2',...
    'Value',0,...
    'String','Left SRB Above Aft Skirt Edge',...
    'Callback',@Mic_Data_Button_Callback);
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right+data_button,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','shuttle_mic_2_data',...
    'Value',0,...
    'Enable','off',...
    'String','Data (mic. # 7987)');

position_list = [0.28 0.775 0.10 0];
% Left SRB Below Nose Shoulder (mic. # 7988)
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','shuttle_mic_3',...
    'Value',0,...
    'String','Left SRB Forward Skirt',...
    'Callback',@Mic_Data_Button_Callback);
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right+data_button,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','shuttle_mic_3_data',...
    'Value',0,...
    'Enable','off',...
    'String','Data (mic. # 7988)');
```

```
position_list = [0.28 0.73 0.10 0];
% Right SRB between nozzle and SRB Center (mic. # 8981)
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','shuttle_mic_4',...
    'Value',0,...
    'String','Right SRB/ET Attach Ring',...
    'Callback',@Mic_Data_Button_Callback);
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right+data_button,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','shuttle_mic_4_data',...
    'Value',0,...
    'Enable','off',...
    'String','Data (mic. # 8981)');

position_list = [0.28 0.685 0.10 0];
% Right SRB Above Nozzle Shirt Edge (mic. # 8988)
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','shuttle_mic_5',...
    'Value',0,...
    'String','Right SRB Above Aft Skirt Edge',...
    'Callback',@Mic_Data_Button_Callback);
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right+data_button,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','shuttle_mic_5_data',...
    'Value',0,...
    'Enable','off',...
    'String','Data (mic. # 8988)');

position_list = [0.28 0.64 0.10 0];
```

```
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','shuttle_mic_9',...
    'Value',0,...
    'String','User Defined',...
    'Callback',@Data_Button_Callback);

% F: Thrust of each Engine
position_list = [0.35 0.57 0 0];
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_left,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','F (N)');
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_bottom,...
    'FontSize',10,...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','(thrust main engine [booter])');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+[0 0 0.15 0.03],...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','1665859 [11787790]');

% Te: Engine Exit Temperature
position_list = [0.35 0.50 0 0];
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_left,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Te (K)');
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_bottom,...
    'FontSize',10, ...
```

```
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','(exit temperature [booster])');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','1673 [3477]');

% de: Engine Exit Diamemter
position_list = [0.35 0.43 0 0];
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_left,...
    'FontSize',10,...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','de (m)');
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+[-0.10 -0.03 0.25 0.03],...
    'FontSize',10,...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','(exit nozzle diamemter [booster])');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','2.304 [3.700]');

% Ue: Engine Exit Velocity
position_list = [0.35 0.36 0 0];
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_left,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Ue (m/s)');
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+[-0.13 -0.03 0.3 0.03],...
    'FontSize',10,...
```

```
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','(fully expanded exit velosity [booster])');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','4419.6 [2331.72]');

engine_data_x={'1.6','0','-1.6','6.4','-6.4'};
engine_data_y={'7.7','10.6','7.7','0','0'};
engine_data_z={'4.5','4.5','4.5','0','0'};
engine_number_string = 5;

ramp_data_y_z={'22','22','22','22','22'};
ramp_data_x_y={'0','0','0','0 ','0 '};

position_list = [0.25 0.17 0 0];
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_title_l,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String',{'Engine','Position'});

uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_title_1,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','X');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_column_1,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'Max',engine_number_string,'Min',0,...
    'String',engine_data_x);

uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_title_2,...
    'FontSize',10, ...
    'ForegroundColor','white',...
```

```
    'BackgroundColor','blue',...
    'String','Y');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_column_2,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'Max',engine_number_string,'Min',0,...
    'String',engine_data_y);

uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_title_3,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Z');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_column_3,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'Max',engine_number_string,'Min',0,...
    'String',engine_data_z);

uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_title_r,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String',{'Apparent','Flow Angle'});

uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_title_4,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','X-Y');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_column_4,...
    'FontSize',10, ...
    'ForegroundColor','white',...
```

```
    'BackgroundColor','blue',...
    'Max',engine_number_string,'Min',0,...
    'String',ramp_data_x_y);

uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_title_5,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Y-Z');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_column_5,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'Max',engine_number_string,'Min',0,...
    'String',ramp_data_y_z);

%.........................................
% Ares I Data:
%.........................................

% Microphone data:
position_list = [0.56 0.89 0.20 0.02];
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Microphone Location Z-axis (m)');

position_list = [0.58 0.865 0.10 0];
% Ares I-X Sensor IAD098P
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','ares_mic_1',...
    'Value',0,...
    'String','56.4',...
    'Callback',@Mic_Data_Button_Callback);
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
```

```
    'Position',position_list+input_right+data_button,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','ares_mic_1_data',...
    'Value',0,...
    'Enable','off',...
    'String','Data (mic. # IAD098P)');

position_list = [0.58 0.82 0.10 0];
% Ares I-X Sensor IAD915P
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','ares_mic_2',...
    'Value',0,...
    'String','83.8',...
    'Callback',@Mic_Data_Button_Callback);
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right+data_button,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','ares_mic_2_data',...
    'Value',0,...
    'Enable','off',...
    'String','Data (mic. # IAD915P)');

position_list = [0.58 0.775 0.10 0];
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','ares_mic_9',...
    'Value',0,...
    'String','User Defined',...
    'Callback',@Data_Button_Callback);

% F: Thrust of each Engine
position_list = [0.65 0.57 0 0];
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_left,...
```

```
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','F (N)');
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_bottom,...
    'FontSize',10,...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','(thrust of each engine)');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','14634649.1');

% Te: Engine Exit Temperature
position_list = [0.65 0.50 0 0];
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_left,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Te (K)');
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_bottom,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','(exit temperature)');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','3477'); % used shuttle

% de: Engine Exit Diamemter
position_list = [0.65 0.43 0 0];
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_left,...
```

```
    'FontSize',10,...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','de (m)');
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_bottom,...
    'FontSize',10,...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','(exit nozzle diamemter)');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','4.401');

% Ue: Engine Exit Velosity
position_list = [0.65 0.36 0 0];
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_left,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Ue (m/s)');
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+[-0.10 -0.03 0.2 0.03],...
    'FontSize',10,...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','(fully expanded exit velosity)');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','2529.84');

engine_data_x={'0'};
engine_data_y={'0'};
engine_data_z={'0'};
engine_number_string = 1;

ramp_data_y_z={'22'};
```

```
ramp_data_x_y={'0'};

position_list = [0.55 0.17 0 0];
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_title_1,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String',{'Engine','Position'});

uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_title_1,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','X');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_column_1,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'Max',engine_number_string,'Min',0,...
    'String',engine_data_x);

uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_title_2,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Y');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_column_2,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'Max',engine_number_string,'Min',0,...
    'String',engine_data_y);

uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_title_3,...
    'FontSize',10, ...
    'ForegroundColor','white',...
```

```
    'BackgroundColor','blue',...
    'String','Z');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_column_3,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'Max',engine_number_string,'Min',0,...
    'String',engine_data_z);

uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_title_r,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String',{'Apparent','Flow Angle'});

uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_title_4,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','X-Y');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_column_4,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'Max',engine_number_string,'Min',0,...
    'String',ramp_data_x_y);

uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_title_5,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Y-Z');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_column_5,...
    'FontSize',10, ...
    'ForegroundColor','white',...
```

```
    'BackgroundColor','blue',...
    'Max',engine_number_string,'Min',0,...
    'String',ramp_data_y_z);

%.........................................
% Super Sonic Jet Data:
%.........................................

% Microphone data:
position_list = [0.78 0.89 0.20 0.02];
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Microphone Location Z-axis (m)');

position_list = [0.80 0.865 0.10 0];
% Sensor 1
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','jet_mic_1',...
    'Value',0,...
    'String','theta = 135',...
    'Callback',@Mic_Data_Button_Callback);
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right+data_button,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','jet_mic_1_data',...
    'Value',0,...
    'Enable','off',...
    'String','Data (mic. # 1)');

position_list = [0.80 0.82 0.10 0];
% Sensor 2
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
```

```
    'Tag','jet_mic_2',...
    'Value',0,...
    'String','theta = 90',...
    'Callback',@Mic_Data_Button_Callback);
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right+data_button,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','jet_mic_2_data',...
    'Value',0,...
    'Enable','off',...
    'String','Data (mic. # 2)');

position_list = [0.80 0.775 0 0];
% Sensor 3
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','jet_mic_3',...
    'Value',0,...
    'String',' theta = 45',...
    'Callback',@Mic_Data_Button_Callback);
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right+data_button,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','jet_mic_3_data',...
    'Value',0,...
    'Enable','off',...
    'String','Data (mic. # 3)');

position_list = [0.80 0.73 0.10 0];
uicontrol(...
    'Parent',data_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','jet_mic_9',...
    'Value',0,...
    'String','User Defined',...
```

```
    'Callback',@Data_Button_Callback);

% F: Thrust of each Engine
position_list = [0.85 0.57 0 0];
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_left,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','F (N)');
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_bottom,...
    'FontSize',10,...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','(thrust of each engine)');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','1351');

% Te: Engine Exit Temperature
position_list = [0.85 0.50 0 0];
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_left,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Te (K)');
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_bottom,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','(exit temperature)');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
```

```
    'String','311');

% de: Engine Exit Diamemter
position_list = [0.85 0.43 0 0];
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_left,...
    'FontSize',10,...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','de (m)');
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_bottom,...
    'FontSize',10,...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','(exit nozzle diamemter)');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','0.073');

% Ue: Engine Exit Velosity
position_list = [0.85 0.36 0 0];
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_left,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Ue (m/s)');
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+[-0.10 -0.03 0.2 0.03],...
    'FontSize',10,...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','(fully expanded exit velosity)');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
```

```
    'String','530');

engine_data_x={'0'};
engine_data_y={'0'};
engine_data_z={'0'};
engine_number_string = 1;

ramp_data_y_z={'-90'};
ramp_data_x_y={'0'};

position_list = [0.75 0.17 0 0];
uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_title_1,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String',{'Engine','Position'});

uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_title_1,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','X');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_column_1,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'Max',engine_number_string,'Min',0,...
    'String',engine_data_x);

uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_title_2,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Y');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_column_2,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'Max',engine_number_string,'Min',0,...
```

```
    'String',engine_data_y);

uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_title_3,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Z');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_column_3,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'Max',engine_number_string,'Min',0,...
    'String',engine_data_z);

uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_title_r,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String',{'Apparent','Flow Angle'});

uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_title_4,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','X-Y');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_column_4,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'Max',engine_number_string,'Min',0,...
    'String',ramp_data_x_y);

uicontrol('Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_title_5,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
```

```
    'String','Y-Z');
uicontrol(...
    'Parent',data_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list + table_column_5,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'Max',engine_number_string,'Min',0,...
    'String',ramp_data_y_z);

%..........................................
% End of screen info:
%..........................................

uicontrol(...
    'Parent',data_panel,...
    'Style','pushbutton',...
    'Units','normalized',...
    'Position',[0.325 0.01 0.15 0.05],...
    'Tag','back_pushbutton',...
    'Callback',{@Back_Main_Menu_Callback},...
    'FontSize',16,...
    'ForegroundColor','black',...
    'String','Back');

uicontrol(...
    'Parent',data_panel,...
    'Style','pushbutton',...
    'Units','normalized',...
    'Position',[0.525 0.01 0.15 0.05],...
    'Tag','next_pushbutton',...
    'Callback',{@Exhaust_Panel_Callback},...
    'FontSize',16,...
    'ForegroundColor','black',...
    'String','Next');

Data_Button_Callback
set(data_panel,'Visible','on');

end

%.....................................................................
      ....

function Data_Button_Callback(hObject, eventdata, handles)

global test_a test_b test_c
if exist('hObject','var')
    test_a = hObject;
end
if exist('eventdata','var')
    test_b = eventdata;
end
```

```
if exist('handles','var')
    test_c = handles;
end

global microphone_number_string_section ...
       microphone_offset_section ...
       microphone_data_x_section ...
       microphone_data_y_section ...
       microphone_data_z_section ...
       microphone_data_r_section ...
       microphone_data_h_section ...
       microphone_data_v_section ...
       microphone_launch_data_exist ...
       microphone_launch_data_frec ...
       microphone_launch_data_frec_width ...
       microphone_launch_data ...
       engine_thrust_data_section ...
       engine_exit_temperature_data_section ...
       engine_exit_diamemter_data_section ...
       engine_exit_velosity_data_section ...
       engine_data_x_section ...
       engine_data_y_section ...
       engine_data_z_section ...
       engine_number_string_section ...
       ramp_data_y_z_section ...
       ramp_data_x_y_section ...
       data_button_section

data_button_section =
      get(get(findobj(gcf,'Tag','data_button_section'),'SelectedObject'
      ),'UserData');

microphone_launch_data_exist = [];
microphone_launch_data_frec  = [];
microphone_launch_data       = [];

if data_button_section==1 % Saturn V Data:
    % Coordinate Reference in Inches (Xs,Ys,Zs):
    % Engine A: (-128.69, -128.69, 0)
    % Engine B: (-128.69,  128.69, 0)
    % Engine C: ( 128.69,  128.69, 0)
    % Engine D: ( 128.69, -128.69, 0)
    % Engine E: (0,0,0) center
    % Mic B0002-115, AS-502,  120"+115.4" at PI:   (       0,   -198,
      235.4)
    % Mic B0003-118, AS-501,  750"+115.4" at PI:   (       0,   -198,
      865.4)
    % Mic B0005-200, AS-501, 1664"+115.4" at PIII: (       0,    198,
      1779.4)
    % Mic B0003-219, AS-501, 2494"+115.4" at PIII: (       0,    198,
      2609.4)
    % Mic B0013-426, AS-501, 3213"+115.4" at PI:   ( -126.67, -26.92,
      3328.4)
    % Mic B0001-601, AS-501, 3244"+115.4" at PI:   (   50.60, 119.21,
      3359.4)
    % Ramp base at Xs = ~?
```

```
% +Xs = program +X-axis = P-IV direction, note -Xs is LUT side or
  P-II direction
% +Ys = program +Y-axis = P-III direction
% +Zs = program +Z-axis = structure vertical
% Please see F-1 Fact Sheet, NASA DS-15796-1 report, and
% http://www.apollosaturn.com/ for addition information.

set(findobj(gcf,'Tag','saturn_mic_1'),'Enable','on');
set(findobj(gcf,'Tag','saturn_mic_2'),'Enable','on');
set(findobj(gcf,'Tag','saturn_mic_3'),'Enable','on');
set(findobj(gcf,'Tag','saturn_mic_4'),'Enable','on');
set(findobj(gcf,'Tag','saturn_mic_5'),'Enable','on');
set(findobj(gcf,'Tag','saturn_mic_6'),'Enable','on');
set(findobj(gcf,'Tag','saturn_mic_9'),'Enable','on');

set(findobj(gcf,'Tag','shuttle_mic_1'),'Enable','off','value',0);
set(findobj(gcf,'Tag','shuttle_mic_2'),'Enable','off','value',0);
set(findobj(gcf,'Tag','shuttle_mic_3'),'Enable','off','value',0);
set(findobj(gcf,'Tag','shuttle_mic_4'),'Enable','off','value',0);
set(findobj(gcf,'Tag','shuttle_mic_5'),'Enable','off','value',0);
set(findobj(gcf,'Tag','shuttle_mic_9'),'Enable','off','value',0);


  set(findobj(gcf,'Tag','shuttle_mic_1_data'),'Enable','off','value
  ',0);

  set(findobj(gcf,'Tag','shuttle_mic_2_data'),'Enable','off','value
  ',0);

  set(findobj(gcf,'Tag','shuttle_mic_3_data'),'Enable','off','value
  ',0);

  set(findobj(gcf,'Tag','shuttle_mic_4_data'),'Enable','off','value
  ',0);

  set(findobj(gcf,'Tag','shuttle_mic_5_data'),'Enable','off','value
  ',0);

set(findobj(gcf,'Tag','ares_mic_1'),'Enable','off','value',0);
set(findobj(gcf,'Tag','ares_mic_2'),'Enable','off','value',0);
set(findobj(gcf,'Tag','ares_mic_9'),'Enable','off','value',0);

set(findobj(gcf,'Tag','ares_mic_1_data'),'Enable','off','value',0);
set(findobj(gcf,'Tag','ares_mic_2_data'),'Enable','off','value',0);

set(findobj(gcf,'Tag','jet_mic_1'),'Enable','off','value',0);
set(findobj(gcf,'Tag','jet_mic_2'),'Enable','off','value',0);
set(findobj(gcf,'Tag','jet_mic_3'),'Enable','off','value',0);
set(findobj(gcf,'Tag','jet_mic_9'),'Enable','off','value',0);

set(findobj(gcf,'Tag','jet_mic_1_data'),'Enable','off','value',0);
set(findobj(gcf,'Tag','jet_mic_2_data'),'Enable','off','value',0);
set(findobj(gcf,'Tag','jet_mic_3_data'),'Enable','off','value',0);

saturn_mic_1 = get(findobj(gcf,'Tag','saturn_mic_1'),'value');
```

```
saturn_mic_2 = get(findobj(gcf,'Tag','saturn_mic_2'),'value');
saturn_mic_3 = get(findobj(gcf,'Tag','saturn_mic_3'),'value');
saturn_mic_4 = get(findobj(gcf,'Tag','saturn_mic_4'),'value');
saturn_mic_5 = get(findobj(gcf,'Tag','saturn_mic_5'),'value');
saturn_mic_6 = get(findobj(gcf,'Tag','saturn_mic_6'),'value');
saturn_mic_9 = get(findobj(gcf,'Tag','saturn_mic_9'),'value');

saturn_mic_1_data =
  get(findobj(gcf,'Tag','saturn_mic_1_data'),'value');
saturn_mic_2_data =
  get(findobj(gcf,'Tag','saturn_mic_2_data'),'value');
saturn_mic_3_data =
  get(findobj(gcf,'Tag','saturn_mic_3_data'),'value');
saturn_mic_4_data =
  get(findobj(gcf,'Tag','saturn_mic_4_data'),'value');
saturn_mic_5_data =
  get(findobj(gcf,'Tag','saturn_mic_5_data'),'value');
saturn_mic_6_data =
  get(findobj(gcf,'Tag','saturn_mic_6_data'),'value');

microphone_offset_section = '24';

if saturn_mic_9==1
    microphone_data_x_section = {'0'};
    microphone_data_y_section = {'0'};
    microphone_data_z_section = {'0'};
    microphone_data_r_section = {'0'};
    microphone_data_h_section = {'0'};
    microphone_data_v_section = {'0'};
    microphone_number_string_section = num2str(1);

  set(findobj(gcf,'Tag','saturn_mic_1'),'Enable','off','value',0);

  set(findobj(gcf,'Tag','saturn_mic_2'),'Enable','off','value',0);

  set(findobj(gcf,'Tag','saturn_mic_3'),'Enable','off','value',0);

  set(findobj(gcf,'Tag','saturn_mic_4'),'Enable','off','value',0);

  set(findobj(gcf,'Tag','saturn_mic_5'),'Enable','off','value',0);

  set(findobj(gcf,'Tag','saturn_mic_6'),'Enable','off','value',0);

  set(findobj(gcf,'Tag','saturn_mic_1_data'),'Enable','off','value'
  ,0);

  set(findobj(gcf,'Tag','saturn_mic_2_data'),'Enable','off','value'
  ,0);

  set(findobj(gcf,'Tag','saturn_mic_3_data'),'Enable','off','value'
  ,0);

  set(findobj(gcf,'Tag','saturn_mic_4_data'),'Enable','off','value'
  ,0);
```

```
  set(findobj(gcf,'Tag','saturn_mic_5_data'),'Enable','off','value'
  ,0);

  set(findobj(gcf,'Tag','saturn_mic_6_data'),'Enable','off','value'
  ,0);
    microphone_launch_data_exist(2,1) = 0;
else
    saturn_mic_sum =
  saturn_mic_1+saturn_mic_2+saturn_mic_3+saturn_mic_4+saturn_mic_5+
  saturn_mic_6;
    microphone_number_string_section = num2str(saturn_mic_sum);
    microphone_data_x_section = []; %#ok<NASGU>
    microphone_data_y_section = []; %#ok<NASGU>
    microphone_data_z_section = []; %#ok<NASGU>
    temp_x =
  {'00.000','00.000','00.000','00.000','00.000','00.000'};
    temp_y =
  {'00.000','00.000','00.000','00.000','00.000','00.000'};
    temp_z =
  {'00.000','00.000','00.000','00.000','00.000','00.000'};
    microphone_data_x_section = temp_x(1,1:saturn_mic_sum);
    microphone_data_y_section = temp_y(1,1:saturn_mic_sum);
    microphone_data_z_section = temp_z(1,1:saturn_mic_sum);

    microphone_data_r_section = []; %#ok<NASGU>
    microphone_data_h_section = []; %#ok<NASGU>
    microphone_data_v_section = []; %#ok<NASGU>
    temp_r =
  {'00.000','00.000','00.000','00.000','00.000','00.000'};
    temp_h =
  {'00.000','00.000','00.000','00.000','00.000','00.000'};
    temp_v =
  {'00.000','00.000','00.000','00.000','00.000','00.000'};
    microphone_data_r_section = temp_r(1,1:saturn_mic_sum);
    microphone_data_h_section = temp_h(1,1:saturn_mic_sum);
    microphone_data_v_section = temp_v(1,1:saturn_mic_sum);

    i=0;

    while i<=saturn_mic_sum
        i=i+1;
        if i==1
            microphone_launch_data_frec = [40 50 60 70 80 90 100
  200 300 400 500 600 700 800 900 1000 2000 3000]';
            microphone_launch_data_frec_width = [10 10 10 10 10 10
  55 100 100 100 100 100 100 100 100 550 1000 1000]';
        end
        if saturn_mic_1==1 % Mic B0002-115, AS-502
            microphone_data_x_section(1,i) = {'00.000'};
            microphone_data_y_section(1,i) = {'-5.029'};
            microphone_data_z_section(1,i) = {' 5.979'};
            microphone_data_r_section(1,i) = {'05.029'};
            microphone_data_h_section(1,i) = {'90.000'};
            microphone_data_v_section(1,i) = {'90.000'};
            microphone_launch_data_exist(1,i) = 1; %#ok<AGROW>
```

```
            microphone_launch_data_exist(2,i) = 0; %#ok<AGROW>
            if saturn_mic_1_data==1
                microphone_launch_data_exist(2,i) = 1; %#ok<AGROW>
                % AS-502
                % microphone_launch_data(:,i) = [126.4 130.2 130.5
130.3 130.6 131.2 130.5 131.5 130.7 130.4 128.1 127.9 127.2 126.4
125.5 125.3 120.2 113.7]; %#ok<AGROW>
                % AS-501
                microphone_launch_data(:,i) = [122.1 121.0 121.4
122.3 122.6 122.7 122.6 120.8 120.5 120.3 118.6 116.6 115.4 114.8
114.7 114.5 109.7 106.8]; %#ok<AGROW>
            end
            i=i+1;
        end
        if saturn_mic_2==1 % Mic B0003-118, AS-501
            microphone_data_x_section(1,i) = {'00.000'};
            microphone_data_y_section(1,i) = {'-5.029'};
            microphone_data_z_section(1,i) = {'21.981'};
            microphone_data_r_section(1,i) = {'05.029'};
            microphone_data_h_section(1,i) = {'90.000'};
            microphone_data_v_section(1,i) = {'90.000'};
            microphone_launch_data_exist(1,i) = 1; %#ok<AGROW>
            microphone_launch_data_exist(2,i) = 0; %#ok<AGROW>
            if saturn_mic_2_data==1
                microphone_launch_data_exist(2,i) = 1; %#ok<AGROW>
                microphone_launch_data(:,i) = [124.1 124.1 123.5
124.3 125.3 125.4 125.0 122.5 121.7 121.5 121.3 120.4 118.5 116.5
115.0 113.6 106.3 98.3]; %#ok<AGROW>
            end
            i=i+1;
        end
        if saturn_mic_3==1 % Mic B0005-200, AS-501
            microphone_data_x_section(1,i) = {'00.000'};
            microphone_data_y_section(1,i) = {' 5.029'};
            microphone_data_z_section(1,i) = {'45.196'};
            microphone_data_r_section(1,i) = {'05.029'};
            microphone_data_h_section(1,i) = {'-90.00'};
            microphone_data_v_section(1,i) = {'90.000'};
            microphone_launch_data_exist(1,i) = 1; %#ok<AGROW>
            microphone_launch_data_exist(2,i) = 0; %#ok<AGROW>
            if saturn_mic_3_data==1
                microphone_launch_data_exist(2,i) = 1; %#ok<AGROW>
                microphone_launch_data(:,i) = [120.5 117.2 117.8
120.5 122.0 122.6 122.7 117.8 116.1 117.4 116.6 112.9 111.3 110.8
110.9 111.0 105.5 100.7]; %#ok<AGROW>
            end
            i=i+1;
        end
        if saturn_mic_4==1 % Mic B0003-219, AS-501
            microphone_data_x_section(1,i) = {'00.000'};
            microphone_data_y_section(1,i) = {' 5.029'};
            microphone_data_z_section(1,i) = {'66.279'};
            microphone_data_r_section(1,i) = {'05.029'};
            microphone_data_h_section(1,i) = {'-90.00'};
            microphone_data_v_section(1,i) = {'90.000'};
            microphone_launch_data_exist(1,i) = 1; %#ok<AGROW>
            microphone_launch_data_exist(2,i) = 0; %#ok<AGROW>
```

```
            if saturn_mic_4_data==1
                microphone_launch_data_exist(2,i) = 1; %#ok<AGROW>
                microphone_launch_data(:,i) = [121.1 125.0 126.8
  127.8 128.0 127.7 127.2 121.7 118.7 116.7 115.0 113.7 112.5 111.4
  110.6 110.0 104.2 100.0]; %#ok<AGROW>
            end
            i=i+1;
        end
        if saturn_mic_5==1 % Mic B0013-426, AS-501
            microphone_data_x_section(1,i) = {'-3.217'};
            microphone_data_y_section(1,i) = {'-0.683'};
            microphone_data_z_section(1,i) = {'84.541'};
            microphone_data_r_section(1,i) = {'03.298'};
            microphone_data_h_section(1,i) = {'12.000'};
            microphone_data_v_section(1,i) = {'90.000'};
            microphone_launch_data_exist(1,i) = 1; %#ok<AGROW>
            microphone_launch_data_exist(2,i) = 0; %#ok<AGROW>
            if saturn_mic_5_data==1
                microphone_launch_data_exist(2,i) = 1; %#ok<AGROW>
                microphone_launch_data(:,i) = [125.8 127.2 126.5
  126.1 126.1 125.5 124.6 116.9 113.5 112.3 109.3 107.6 106.4 105.4
  104.1 103.5 98.4 90.3]; %#ok<AGROW>
            end
            i=i+1;
        end
        if saturn_mic_6==1 % Mic B0001-601, AS-501
            microphone_data_x_section(1,i) = {' 1.285'};
            microphone_data_y_section(1,i) = {' 3.027'};
            microphone_data_z_section(1,i) = {'85.329'};
            microphone_data_r_section(1,i) = {'03.298'};
            microphone_data_h_section(1,i) = {'67.000'};
            microphone_data_v_section(1,i) = {'90.000'};
            microphone_launch_data_exist(1,i) = 1; %#ok<AGROW>
            microphone_launch_data_exist(2,i) = 0; %#ok<AGROW>
            if saturn_mic_6_data==1
                microphone_launch_data_exist(2,i) = 1; %#ok<AGROW>
                microphone_launch_data(:,i) = [125.6 126.7 126.5
  127.1 127.5 127.6 127.4 123.4 120.2 118.6 116.4 114.5 113.3 112.2
  111.1 110.4 105.1 99.6]; %#ok<AGROW>
            end
            i=i+1;
        end
    end
end

engine_thrust_data_section           = '6672333'; % (N, 1.5 mil lb)
engine_exit_temperature_data_section = '3572';    % (K, 5970 F)
engine_exit_diamemter_data_section   = '3.53';    % (m, 139 in)
engine_exit_velosity_data_section    = '2585';    % (m/s, 8485
  ft/s)

engine_data_x_section = {'-3.27','-3.27','3.27','3.27','0'}; % (m)
engine_data_y_section = {'-3.27','3.27','3.27','-3.27','0'};
engine_data_z_section = {'0','0','0','0','0'};
engine_number_string_section  = '5';
```

```
    ramp_data_y_z_section = {'22','22','22','22','22'};
    ramp_data_x_y_section = {'0','0','0','0','0'};

elseif data_button_section==2 % Shuttle Data:
    % Space Shuttle "System" Coordinate Reference in Inches (Xs,Ys,Zs):
    % Right  Engine: (2354, 61.25,304.2)
    % Middle Engine: (2354,   0  ,418.3)
    % Left   Engine: (2354,-61.25,304.2)
    % Mic 7986: (2468.0,-348.7,   0)
    % Mic 7987: (2409.0,-336.1,   0)
    % Mic 7988: (1015.0,-302.1,51.6)
    % Mic 8981: (2054.4, 250.5,73.0)
    % Mic 8982: (2408.0, 250.5,85.6)
    % Ramp base at Xs = ~3602
    % +Xs (opposite) = program +Z-axis = structure vertical
    % +Ys            = program +X-axis = structure west
    % +Zs            = program +Y-axis = structure south
    % Please see web site
      http://spaceflight.nasa.gov/shuttle/reference/
    % and Excel file "Space Shuttle and Solid Rocket Booster
      referances.xls"
    % for addition information.

    set(findobj(gcf,'Tag','saturn_mic_1'),'Enable','off','value',0);
    set(findobj(gcf,'Tag','saturn_mic_2'),'Enable','off','value',0);
    set(findobj(gcf,'Tag','saturn_mic_3'),'Enable','off','value',0);
    set(findobj(gcf,'Tag','saturn_mic_4'),'Enable','off','value',0);
    set(findobj(gcf,'Tag','saturn_mic_5'),'Enable','off','value',0);
    set(findobj(gcf,'Tag','saturn_mic_6'),'Enable','off','value',0);
    set(findobj(gcf,'Tag','saturn_mic_9'),'Enable','off','value',0);


      set(findobj(gcf,'Tag','saturn_mic_1_data'),'Enable','off','value'
      ,0);

      set(findobj(gcf,'Tag','saturn_mic_2_data'),'Enable','off','value'
      ,0);

      set(findobj(gcf,'Tag','saturn_mic_3_data'),'Enable','off','value'
      ,0);

      set(findobj(gcf,'Tag','saturn_mic_4_data'),'Enable','off','value'
      ,0);

      set(findobj(gcf,'Tag','saturn_mic_5_data'),'Enable','off','value'
      ,0);

      set(findobj(gcf,'Tag','saturn_mic_6_data'),'Enable','off','value'
      ,0);

    set(findobj(gcf,'Tag','shuttle_mic_1'),'Enable','on');
    set(findobj(gcf,'Tag','shuttle_mic_2'),'Enable','on');
    set(findobj(gcf,'Tag','shuttle_mic_3'),'Enable','on');
    set(findobj(gcf,'Tag','shuttle_mic_4'),'Enable','on');
    set(findobj(gcf,'Tag','shuttle_mic_5'),'Enable','on');
    set(findobj(gcf,'Tag','shuttle_mic_9'),'Enable','on');
```

```
set(findobj(gcf,'Tag','ares_mic_1'),'Enable','off','value',0);
set(findobj(gcf,'Tag','ares_mic_2'),'Enable','off','value',0);
set(findobj(gcf,'Tag','ares_mic_9'),'Enable','off','value',0);

set(findobj(gcf,'Tag','ares_mic_1_data'),'Enable','off','value',0);
set(findobj(gcf,'Tag','ares_mic_2_data'),'Enable','off','value',0);

set(findobj(gcf,'Tag','jet_mic_1'),'Enable','off','value',0);
set(findobj(gcf,'Tag','jet_mic_2'),'Enable','off','value',0);
set(findobj(gcf,'Tag','jet_mic_3'),'Enable','off','value',0);
set(findobj(gcf,'Tag','jet_mic_9'),'Enable','off','value',0);

set(findobj(gcf,'Tag','jet_mic_1_data'),'Enable','off','value',0);
set(findobj(gcf,'Tag','jet_mic_2_data'),'Enable','off','value',0);
set(findobj(gcf,'Tag','jet_mic_3_data'),'Enable','off','value',0);

shuttle_mic_1 = get(findobj(gcf,'Tag','shuttle_mic_1'),'value');
shuttle_mic_2 = get(findobj(gcf,'Tag','shuttle_mic_2'),'value');
shuttle_mic_3 = get(findobj(gcf,'Tag','shuttle_mic_3'),'value');
shuttle_mic_4 = get(findobj(gcf,'Tag','shuttle_mic_4'),'value');
shuttle_mic_5 = get(findobj(gcf,'Tag','shuttle_mic_5'),'value');
shuttle_mic_9 = get(findobj(gcf,'Tag','shuttle_mic_9'),'value');

shuttle_mic_1_data =
  get(findobj(gcf,'Tag','shuttle_mic_1_data'),'value');
shuttle_mic_2_data =
  get(findobj(gcf,'Tag','shuttle_mic_2_data'),'value');
shuttle_mic_3_data =
  get(findobj(gcf,'Tag','shuttle_mic_3_data'),'value');
shuttle_mic_4_data =
  get(findobj(gcf,'Tag','shuttle_mic_4_data'),'value');
shuttle_mic_5_data =
  get(findobj(gcf,'Tag','shuttle_mic_5_data'),'value');

microphone_offset_section = '27.163';

if shuttle_mic_9==1
    microphone_data_x_section = {'0'};
    microphone_data_y_section = {'0'};
    microphone_data_z_section = {'0'};
    microphone_data_r_section = {'0'};
    microphone_data_h_section = {'0'};
    microphone_data_v_section = {'0'};
    microphone_number_string_section = num2str(1);

  set(findobj(gcf,'Tag','shuttle_mic_1'),'Enable','off','value',0);

  set(findobj(gcf,'Tag','shuttle_mic_2'),'Enable','off','value',0);

  set(findobj(gcf,'Tag','shuttle_mic_3'),'Enable','off','value',0);

  set(findobj(gcf,'Tag','shuttle_mic_4'),'Enable','off','value',0);

  set(findobj(gcf,'Tag','shuttle_mic_5'),'Enable','off','value',0);
```

```
  set(findobj(gcf,'Tag','shuttle_mic_1_data'),'Enable','off','value
  ',0);

  set(findobj(gcf,'Tag','shuttle_mic_2_data'),'Enable','off','value
  ',0);

  set(findobj(gcf,'Tag','shuttle_mic_3_data'),'Enable','off','value
  ',0);

  set(findobj(gcf,'Tag','shuttle_mic_4_data'),'Enable','off','value
  ',0);

  set(findobj(gcf,'Tag','shuttle_mic_5_data'),'Enable','off','value
  ',0);
    microphone_launch_data_exist(2,1) = 0;
else
    shuttle_mic_sum =
  shuttle_mic_1+shuttle_mic_2+shuttle_mic_3+shuttle_mic_4+shuttle_m
  ic_5;
    microphone_number_string_section = num2str(shuttle_mic_sum);
    microphone_data_x_section = []; %#ok<NASGU>
    microphone_data_y_section = []; %#ok<NASGU>
    microphone_data_z_section = []; %#ok<NASGU>
    temp_x = {'00.000','00.000','00.000','00.000','00.000'};
    temp_y = {'00.000','00.000','00.000','00.000','00.000'};
    temp_z = {'00.000','00.000','00.000','00.000','00.000'};
    microphone_data_x_section = temp_x(1,1:shuttle_mic_sum);
    microphone_data_y_section = temp_y(1,1:shuttle_mic_sum);
    microphone_data_z_section = temp_z(1,1:shuttle_mic_sum);

    microphone_data_r_section = []; %#ok<NASGU>
    microphone_data_h_section = []; %#ok<NASGU>
    microphone_data_v_section = []; %#ok<NASGU>
    temp_r =
  {'00.000','00.000','00.000','00.000','00.000','00.000'};
    temp_h =
  {'00.000','00.000','00.000','00.000','00.000','00.000'};
    temp_v =
  {'00.000','00.000','00.000','00.000','00.000','00.000'};
    microphone_data_r_section = temp_r(1,1:shuttle_mic_sum);
    microphone_data_h_section = temp_h(1,1:shuttle_mic_sum);
    microphone_data_v_section = temp_v(1,1:shuttle_mic_sum);

    i=0;
    while i<=shuttle_mic_sum
        i=i+1;
        if i==1
            microphone_launch_data_frec = [20 25 31 40 50 63 80 100
  125 160 200 250 315 400 500 630 800 1000 1250 1600 2000 2500 3150
  4000]';
        end
        if shuttle_mic_1==1 % Mic 7986: STS-5 corrected (1.0 sec)
            microphone_data_x_section(1,i) = {'-8.856'};
            microphone_data_y_section(1,i) = {'00.001'}; % ~0
            microphone_data_z_section(1,i) = {'01.640'};
```

```
            microphone_data_r_section(1,i) = {'02.494'};
            microphone_data_h_section(1,i) = {'90.000'};
            microphone_data_v_section(1,i) = {'90.000'};
            microphone_launch_data_exist(1,i) = 1; %#ok<AGROW>
            microphone_launch_data_exist(2,i) = 0; %#ok<AGROW>
            if shuttle_mic_1_data==1
                microphone_launch_data_exist(2,i) = 1; %#ok<AGROW>
                microphone_launch_data(:,i) = [141.25   141.25
143.35  141.95  144.15  142.85  143.85  142.85  143.75  148.25
149.85  150.35  150.85  152.25  151.65  150.55  150.45  150.05
151.15  149.85  149.25  148.85  147.35  146.85]'; %#ok<AGROW>
            end
            i=i+1;
        end
        if shuttle_mic_2==1 % Mic 7987: STS-5 corrected (1.0 sec)
            microphone_data_x_section(1,i) = {'-8.537'};
            microphone_data_y_section(1,i) = {'00.001'};
            microphone_data_z_section(1,i) = {'03.139'};
            microphone_data_r_section(1,i) = {'02.174'};
            microphone_data_h_section(1,i) = {'90.000'};
            microphone_data_v_section(1,i) = {'90.000'};
            microphone_launch_data_exist(1,i) = 1; %#ok<AGROW>
            microphone_launch_data_exist(2,i) = 0; %#ok<AGROW>
            if shuttle_mic_2_data==1
                microphone_launch_data_exist(2,i) = 1; %#ok<AGROW>
                microphone_launch_data(:,i) = [132.14   132.14
133.34  133.14  133.64  133.54  133.24  135.64  136.94  140.64
142.84  141.04  141.24  143.24  141.84  143.94  143.74  144.44
145.74  146.94  147.34  146.44  141.74  140.44]'; %#ok<AGROW>
            end
            i=i+1;
        end
        if shuttle_mic_3==1 % Mic 7988: STS-5 corrected (1.7 sec)
            microphone_data_x_section(1,i) = {'-7.674'};
            microphone_data_y_section(1,i) = {'01.311'};
            microphone_data_z_section(1,i) = {'38.548'};
            microphone_data_r_section(1,i) = {'01.855'};
            microphone_data_h_section(1,i) = {'45.000'};
            microphone_data_v_section(1,i) = {'90.000'};
            microphone_launch_data_exist(1,i) = 1; %#ok<AGROW>
            microphone_launch_data_exist(2,i) = 0; %#ok<AGROW>
            if shuttle_mic_3_data==1
                microphone_launch_data_exist(2,i) = 1; %#ok<AGROW>
                microphone_launch_data(:,i) = [132.12   132.12
132.12  132.12  132.12  132.12  132.12  132.12  133.02  133.62
134.82  133.32  132.92  133.82  133.92  134.42  134.32  134.42
134.92  135.32  135.52  135.42  134.32  132.62]'; %#ok<AGROW>
            end
            i=i+1;
        end
        if shuttle_mic_4==1 % Mic 8981: STS-5 corrected (1.0 sec)
            microphone_data_x_section(1,i) = {'06.363'};
            microphone_data_y_section(1,i) = {'01.855'};
            microphone_data_z_section(1,i) = {'12.155'};
            microphone_data_r_section(1,i) = {'01.855'};
            microphone_data_h_section(1,i) = {'00.000'};
            microphone_data_v_section(1,i) = {'90.000'};
```

```
                microphone_launch_data_exist(1,i) = 1; %#ok<AGROW>
                microphone_launch_data_exist(2,i) = 0; %#ok<AGROW>
                if shuttle_mic_4_data==1
                    microphone_launch_data_exist(2,i) = 1; %#ok<AGROW>
                    microphone_launch_data(:,i) = [131.84   131.84
      131.84  131.84  131.84  134.43  135.14  135.54  137.94  140.84
      141.74  137.94  142.34  144.74  143.14  144.84  144.84  144.44
      145.74  146.34  146.64  146.74  146.04  144.54]'; %#ok<AGROW>
                end
                i=i+1;
            end
            if shuttle_mic_5==1 % Mic 8982:
                microphone_data_x_section(1,i) = {'06.363'};
                microphone_data_y_section(1,i) = {'02.174'};
                microphone_data_z_section(1,i) = {'03.164'};
                microphone_data_r_section(1,i) = {'02.174'};
                microphone_data_h_section(1,i) = {'00.000'};
                microphone_data_v_section(1,i) = {'90.000'};
                microphone_launch_data_exist(1,i) = 1; %#ok<AGROW>
                microphone_launch_data_exist(2,i) = 0; %#ok<AGROW>
                if shuttle_mic_5_data==1
                    microphone_launch_data_exist(2,i) = 1; %#ok<AGROW>
                    microphone_launch_data(:,i) = [133.73   134.63
      133.43  135.63  137.63  137.13  137.63  139.13  141.63  143.63
      144.13  142.13  144.63  147.63  146.63  148.13  148.63  148.13
      150.13  150.13  150.93  151.13  149.93  148.63]'; %#ok<AGROW>
                end
                i=i+1;
            end
        end
    end

    engine_thrust_data_section = {'1665859'};
    engine_exit_temperature_data_section = {'1673'};
    engine_exit_diamemter_data_section = {'2.304'};
    engine_exit_velosity_data_section = {'4419.6'};

    engine_data_x_section = {'1.556','0','-1.556'};
    engine_data_y_section = {'7.727','10.625','7.727'};
    engine_data_z_section = {'4.536','4.536','4.536'};
    engine_number_string_section  = '3';

    ramp_data_y_z_section = {'22','22','22'};
    ramp_data_x_y_section = {'0','0','0'};

elseif data_button_section==3 % Shuttle Booster Data:
    % Space Shuttle "System" Coordinate Reference in Inches (Xs,Ys,Zs):
    % Right  Engine: (2532.6, 250.5,0)
    % Left   Engine: (2532.6,-250.5,0)
    % Mic 7986: (2468.0,-348.7,   0)
    % Mic 7987: (2409.0,-336.1,   0)
    % Mic 7988: (1015.0,-302.1,51.6)
    % Mic 8981: (2054.4, 250.5,73.0)
    % Mic 8982: (2408.0, 250.5,85.6)
    % Ramp base at Xs = ~3602
    % +Xs (opposite) = program +Z-axis = structure vertical
```

```
% +Ys            = program +X-axis = structure west
% +Zs            = program +Y-axis = structure south
% Please see web site
  http://spaceflight.nasa.gov/shuttle/reference/
% and Excel file "Space Shuttle and Solid Rocket Booster
  referances.xls"
% for addition information.

set(findobj(gcf,'Tag','saturn_mic_1'),'Enable','off','value',0);
set(findobj(gcf,'Tag','saturn_mic_2'),'Enable','off','value',0);
set(findobj(gcf,'Tag','saturn_mic_3'),'Enable','off','value',0);
set(findobj(gcf,'Tag','saturn_mic_4'),'Enable','off','value',0);
set(findobj(gcf,'Tag','saturn_mic_5'),'Enable','off','value',0);
set(findobj(gcf,'Tag','saturn_mic_6'),'Enable','off','value',0);
set(findobj(gcf,'Tag','saturn_mic_9'),'Enable','off','value',0);


  set(findobj(gcf,'Tag','saturn_mic_1_data'),'Enable','off','value'
  ,0);

  set(findobj(gcf,'Tag','saturn_mic_2_data'),'Enable','off','value'
  ,0);

  set(findobj(gcf,'Tag','saturn_mic_3_data'),'Enable','off','value'
  ,0);

  set(findobj(gcf,'Tag','saturn_mic_4_data'),'Enable','off','value'
  ,0);

  set(findobj(gcf,'Tag','saturn_mic_5_data'),'Enable','off','value'
  ,0);

  set(findobj(gcf,'Tag','saturn_mic_6_data'),'Enable','off','value'
  ,0);

set(findobj(gcf,'Tag','shuttle_mic_1'),'Enable','on');
set(findobj(gcf,'Tag','shuttle_mic_2'),'Enable','on');
set(findobj(gcf,'Tag','shuttle_mic_3'),'Enable','on');
set(findobj(gcf,'Tag','shuttle_mic_4'),'Enable','on');
set(findobj(gcf,'Tag','shuttle_mic_5'),'Enable','on');
set(findobj(gcf,'Tag','shuttle_mic_9'),'Enable','on');

set(findobj(gcf,'Tag','ares_mic_1'),'Enable','off','value',0);
set(findobj(gcf,'Tag','ares_mic_2'),'Enable','off','value',0);
set(findobj(gcf,'Tag','ares_mic_9'),'Enable','off','value',0);

set(findobj(gcf,'Tag','ares_mic_1_data'),'Enable','off','value',0);
set(findobj(gcf,'Tag','ares_mic_2_data'),'Enable','off','value',0);

set(findobj(gcf,'Tag','jet_mic_1'),'Enable','off','value',0);
set(findobj(gcf,'Tag','jet_mic_2'),'Enable','off','value',0);
set(findobj(gcf,'Tag','jet_mic_3'),'Enable','off','value',0);
set(findobj(gcf,'Tag','jet_mic_9'),'Enable','off','value',0);

set(findobj(gcf,'Tag','jet_mic_1_data'),'Enable','off','value',0);
```

```
set(findobj(gcf,'Tag','jet_mic_2_data'),'Enable','off','value',0);
set(findobj(gcf,'Tag','jet_mic_3_data'),'Enable','off','value',0);

shuttle_mic_1 = get(findobj(gcf,'Tag','shuttle_mic_1'),'value');
shuttle_mic_2 = get(findobj(gcf,'Tag','shuttle_mic_2'),'value');
shuttle_mic_3 = get(findobj(gcf,'Tag','shuttle_mic_3'),'value');
shuttle_mic_4 = get(findobj(gcf,'Tag','shuttle_mic_4'),'value');
shuttle_mic_5 = get(findobj(gcf,'Tag','shuttle_mic_5'),'value');
shuttle_mic_9 = get(findobj(gcf,'Tag','shuttle_mic_9'),'value');

shuttle_mic_1_data =
  get(findobj(gcf,'Tag','shuttle_mic_1_data'),'value');
shuttle_mic_2_data =
  get(findobj(gcf,'Tag','shuttle_mic_2_data'),'value');
shuttle_mic_3_data =
  get(findobj(gcf,'Tag','shuttle_mic_3_data'),'value');
shuttle_mic_4_data =
  get(findobj(gcf,'Tag','shuttle_mic_4_data'),'value');
shuttle_mic_5_data =
  get(findobj(gcf,'Tag','shuttle_mic_5_data'),'value');

microphone_offset_section = '27.163';

if shuttle_mic_9==1
    microphone_data_x_section = {'0'};
    microphone_data_y_section = {'0'};
    microphone_data_z_section = {'0'};
    microphone_data_r_section = {'0'};
    microphone_data_h_section = {'0'};
    microphone_data_v_section = {'0'};
    microphone_number_string_section = num2str(1);

  set(findobj(gcf,'Tag','shuttle_mic_1'),'Enable','off','value',0);

  set(findobj(gcf,'Tag','shuttle_mic_2'),'Enable','off','value',0);

  set(findobj(gcf,'Tag','shuttle_mic_3'),'Enable','off','value',0);

  set(findobj(gcf,'Tag','shuttle_mic_4'),'Enable','off','value',0);

  set(findobj(gcf,'Tag','shuttle_mic_5'),'Enable','off','value',0);

  set(findobj(gcf,'Tag','shuttle_mic_1_data'),'Enable','off','value
  ',0);

  set(findobj(gcf,'Tag','shuttle_mic_2_data'),'Enable','off','value
  ',0);

  set(findobj(gcf,'Tag','shuttle_mic_3_data'),'Enable','off','value
  ',0);

  set(findobj(gcf,'Tag','shuttle_mic_4_data'),'Enable','off','value
  ',0);

  set(findobj(gcf,'Tag','shuttle_mic_5_data'),'Enable','off','value
  ',0);
```

```
    microphone_launch_data_exist(2,1) = 0;
else
    shuttle_mic_sum =
  shuttle_mic_1+shuttle_mic_2+shuttle_mic_3+shuttle_mic_4+shuttle_m
  ic_5;
    microphone_number_string_section = num2str(shuttle_mic_sum);
    microphone_data_x_section = []; %#ok<NASGU>
    microphone_data_y_section = []; %#ok<NASGU>
    microphone_data_z_section = []; %#ok<NASGU>
    temp_x = {'00.000','00.000','00.000','00.000','00.000'};
    temp_y = {'00.000','00.000','00.000','00.000','00.000'};
    temp_z = {'00.000','00.000','00.000','00.000','00.000'};
    microphone_data_x_section = temp_x(1,1:shuttle_mic_sum);
    microphone_data_y_section = temp_y(1,1:shuttle_mic_sum);
    microphone_data_z_section = temp_z(1,1:shuttle_mic_sum);

    microphone_data_r_section = []; %#ok<NASGU>
    microphone_data_h_section = []; %#ok<NASGU>
    microphone_data_v_section = []; %#ok<NASGU>
    temp_r =
  {'00.000','00.000','00.000','00.000','00.000','00.000'};
    temp_h =
  {'00.000','00.000','00.000','00.000','00.000','00.000'};
    temp_v =
  {'00.000','00.000','00.000','00.000','00.000','00.000'};
    microphone_data_r_section = temp_r(1,1:shuttle_mic_sum);
    microphone_data_h_section = temp_h(1,1:shuttle_mic_sum);
    microphone_data_v_section = temp_v(1,1:shuttle_mic_sum);

    i=0;
    while i<=shuttle_mic_sum
        i=i+1;
        if i==1
            microphone_launch_data_frec = [20 25 31 40 50 63 80 100
  125 160 200 250 315 400 500 630 800 1000 1250 1600 2000 2500 3150
  4000]';
        end
        if shuttle_mic_1==1 % Mic 7986:
            microphone_data_x_section(1,i) = {'-8.856'};
            microphone_data_y_section(1,i) = {'00.001'}; % ~0
            microphone_data_z_section(1,i) = {'01.640'};
            microphone_data_r_section(1,i) = {'02.494'};
            microphone_data_h_section(1,i) = {'90.000'};
            microphone_data_v_section(1,i) = {'90.000'};
            microphone_launch_data_exist(1,i) = 1; %#ok<AGROW>
            microphone_launch_data_exist(2,i) = 0; %#ok<AGROW>
            if shuttle_mic_1_data==1
                microphone_launch_data_exist(2,i) = 1; %#ok<AGROW>
                microphone_launch_data(:,i) = [141.25   141.25
  143.35  141.95  144.15  142.85  143.85  142.85  143.75  148.25
  149.85  150.35  150.85  152.25  151.65  150.55  150.45  150.05
  151.15  149.85  149.25  148.85  147.35  146.85]'; %#ok<AGROW>
            end
            i=i+1;
        end
        if shuttle_mic_2==1 % Mic 7987:
```

```
            microphone_data_x_section(1,i) = {'-8.537'};
            microphone_data_y_section(1,i) = {'00.001'};
            microphone_data_z_section(1,i) = {'03.139'};
            microphone_data_r_section(1,i) = {'02.174'};
            microphone_data_h_section(1,i) = {'90.000'};
            microphone_data_v_section(1,i) = {'90.000'};
            microphone_launch_data_exist(1,i) = 1; %#ok<AGROW>
            microphone_launch_data_exist(2,i) = 0; %#ok<AGROW>
            if shuttle_mic_2_data==1
                microphone_launch_data_exist(2,i) = 1; %#ok<AGROW>
                microphone_launch_data(:,i) = [132.14   132.14
133.34  133.14  133.64  133.54  133.24  135.64  136.94  140.64
142.84  141.04  141.24  143.24  141.84  143.94  143.74  144.44
145.74  146.94  147.34  146.44  141.74  140.44]'; %#ok<AGROW>
            end
            i=i+1;
        end
        if shuttle_mic_3==1 % Mic 7988:
            microphone_data_x_section(1,i) = {'-7.674'};
            microphone_data_y_section(1,i) = {'01.311'};
            microphone_data_z_section(1,i) = {'38.548'};
            microphone_data_r_section(1,i) = {'01.855'};
            microphone_data_h_section(1,i) = {'45.000'};
            microphone_data_v_section(1,i) = {'90.000'};
            microphone_launch_data_exist(1,i) = 1; %#ok<AGROW>
            microphone_launch_data_exist(2,i) = 0; %#ok<AGROW>
            if shuttle_mic_3_data==1
                microphone_launch_data_exist(2,i) = 1; %#ok<AGROW>
                microphone_launch_data(:,i) = [132.12   132.12
132.12  132.12  132.12  132.12  132.12  132.12  133.02  133.62
134.82  133.32  132.92  133.82  133.92  134.42  134.32  134.42
134.92  135.32  135.52  135.42  134.32  132.62]'; %#ok<AGROW>
            end
            i=i+1;
        end
        if shuttle_mic_4==1 % Mic 8981:
            microphone_data_x_section(1,i) = {'06.363'};
            microphone_data_y_section(1,i) = {'01.855'};
            microphone_data_z_section(1,i) = {'12.155'};
            microphone_data_r_section(1,i) = {'01.855'};
            microphone_data_h_section(1,i) = {'00.000'};
            microphone_data_v_section(1,i) = {'90.000'};
            microphone_launch_data_exist(1,i) = 1; %#ok<AGROW>
            microphone_launch_data_exist(2,i) = 0; %#ok<AGROW>
            if shuttle_mic_4_data==1
                microphone_launch_data_exist(2,i) = 1; %#ok<AGROW>
                microphone_launch_data(:,i) = [131.84   131.84
131.84  131.84  131.84  134.43  135.14  135.54  137.94  140.84
141.74  137.94  142.34  144.74  143.14  144.84  144.84  144.44
145.74  146.34  146.64  146.74  146.04  144.54]'; %#ok<AGROW>
            end
            i=i+1;
        end
        if shuttle_mic_5==1 % Mic 8982:
            microphone_data_x_section(1,i) = {'06.363'};
            microphone_data_y_section(1,i) = {'02.174'};
            microphone_data_z_section(1,i) = {'03.164'};
```

```
                microphone_data_r_section(1,i) = {'02.174'};
                microphone_data_h_section(1,i) = {'00.000'};
                microphone_data_v_section(1,i) = {'90.000'};
                microphone_launch_data_exist(1,i) = 1; %#ok<AGROW>
                microphone_launch_data_exist(2,i) = 0; %#ok<AGROW>
                if shuttle_mic_5_data==1
                    microphone_launch_data_exist(2,i) = 1; %#ok<AGROW>
                    microphone_launch_data(:,i) = [133.73   134.63
      133.43  135.63  137.63  137.13  137.63  139.13  141.63  143.63
      144.13  142.13  144.63  147.63  146.63  148.13  148.63  148.13
      150.13  150.13  150.93  151.13  149.93  148.63]'; %#ok<AGROW>
                end
                i=i+1;
            end
        end
    end

    engine_thrust_data_section = {'11787790'};
    engine_exit_temperature_data_section = {'3477'};
    engine_exit_diamemter_data_section = {'3.700'};
    engine_exit_velosity_data_section = {'2331.72'};

    engine_data_x_section = {'6.363','-6.363'};
    engine_data_y_section = {'0','0'};
    engine_data_z_section = {'0','0'};
    engine_number_string_section  = '2';

    ramp_data_y_z_section = {'22','22'};
    ramp_data_x_y_section = {'0','0'};

elseif data_button_section==4 % Both Shuttle and Booster: (but starting
      with Shuttle then Booster see "Exhaust_Panel_Check_Callback"
      function for Booster data)

    set(findobj(gcf,'Tag','saturn_mic_1'),'Enable','off','value',0);
    set(findobj(gcf,'Tag','saturn_mic_2'),'Enable','off','value',0);
    set(findobj(gcf,'Tag','saturn_mic_3'),'Enable','off','value',0);
    set(findobj(gcf,'Tag','saturn_mic_4'),'Enable','off','value',0);
    set(findobj(gcf,'Tag','saturn_mic_5'),'Enable','off','value',0);
    set(findobj(gcf,'Tag','saturn_mic_6'),'Enable','off','value',0);
    set(findobj(gcf,'Tag','saturn_mic_9'),'Enable','off','value',0);


      set(findobj(gcf,'Tag','saturn_mic_1_data'),'Enable','off','value'
      ,0);

      set(findobj(gcf,'Tag','saturn_mic_2_data'),'Enable','off','value'
      ,0);

      set(findobj(gcf,'Tag','saturn_mic_3_data'),'Enable','off','value'
      ,0);

      set(findobj(gcf,'Tag','saturn_mic_4_data'),'Enable','off','value'
      ,0);
```

```
  set(findobj(gcf,'Tag','saturn_mic_5_data'),'Enable','off','value'
  ,0);

  set(findobj(gcf,'Tag','saturn_mic_6_data'),'Enable','off','value'
  ,0);

set(findobj(gcf,'Tag','shuttle_mic_1'),'Enable','on');
set(findobj(gcf,'Tag','shuttle_mic_2'),'Enable','on');
set(findobj(gcf,'Tag','shuttle_mic_3'),'Enable','on');
set(findobj(gcf,'Tag','shuttle_mic_4'),'Enable','on');
set(findobj(gcf,'Tag','shuttle_mic_5'),'Enable','on');
set(findobj(gcf,'Tag','shuttle_mic_9'),'Enable','on');

set(findobj(gcf,'Tag','ares_mic_1'),'Enable','off','value',0);
set(findobj(gcf,'Tag','ares_mic_2'),'Enable','off','value',0);
set(findobj(gcf,'Tag','ares_mic_9'),'Enable','off','value',0);

set(findobj(gcf,'Tag','ares_mic_1_data'),'Enable','off','value',0);
set(findobj(gcf,'Tag','ares_mic_2_data'),'Enable','off','value',0);

set(findobj(gcf,'Tag','jet_mic_1'),'Enable','off','value',0);
set(findobj(gcf,'Tag','jet_mic_2'),'Enable','off','value',0);
set(findobj(gcf,'Tag','jet_mic_3'),'Enable','off','value',0);
set(findobj(gcf,'Tag','jet_mic_9'),'Enable','off','value',0);

set(findobj(gcf,'Tag','jet_mic_1_data'),'Enable','off','value',0);
set(findobj(gcf,'Tag','jet_mic_2_data'),'Enable','off','value',0);
set(findobj(gcf,'Tag','jet_mic_3_data'),'Enable','off','value',0);

shuttle_mic_1 = get(findobj(gcf,'Tag','shuttle_mic_1'),'value');
shuttle_mic_2 = get(findobj(gcf,'Tag','shuttle_mic_2'),'value');
shuttle_mic_3 = get(findobj(gcf,'Tag','shuttle_mic_3'),'value');
shuttle_mic_4 = get(findobj(gcf,'Tag','shuttle_mic_4'),'value');
shuttle_mic_5 = get(findobj(gcf,'Tag','shuttle_mic_5'),'value');
shuttle_mic_9 = get(findobj(gcf,'Tag','shuttle_mic_9'),'value');

shuttle_mic_1_data =
  get(findobj(gcf,'Tag','shuttle_mic_1_data'),'value');
shuttle_mic_2_data =
  get(findobj(gcf,'Tag','shuttle_mic_2_data'),'value');
shuttle_mic_3_data =
  get(findobj(gcf,'Tag','shuttle_mic_3_data'),'value');
shuttle_mic_4_data =
  get(findobj(gcf,'Tag','shuttle_mic_4_data'),'value');
shuttle_mic_5_data =
  get(findobj(gcf,'Tag','shuttle_mic_5_data'),'value');

microphone_offset_section = '27.163';

if shuttle_mic_9==1
    microphone_data_x_section = {'0'};
    microphone_data_y_section = {'0'};
    microphone_data_z_section = {'0'};
    microphone_data_r_section = {'0'};
```

```
      microphone_data_h_section = {'0'};
      microphone_data_v_section = {'0'};
      microphone_number_string_section = num2str(1);

    set(findobj(gcf,'Tag','shuttle_mic_1'),'Enable','off','value',0);

    set(findobj(gcf,'Tag','shuttle_mic_2'),'Enable','off','value',0);

    set(findobj(gcf,'Tag','shuttle_mic_3'),'Enable','off','value',0);

    set(findobj(gcf,'Tag','shuttle_mic_4'),'Enable','off','value',0);

    set(findobj(gcf,'Tag','shuttle_mic_5'),'Enable','off','value',0);

    set(findobj(gcf,'Tag','shuttle_mic_1_data'),'Enable','off','value
    ',0);

    set(findobj(gcf,'Tag','shuttle_mic_2_data'),'Enable','off','value
    ',0);

    set(findobj(gcf,'Tag','shuttle_mic_3_data'),'Enable','off','value
    ',0);

    set(findobj(gcf,'Tag','shuttle_mic_4_data'),'Enable','off','value
    ',0);

    set(findobj(gcf,'Tag','shuttle_mic_5_data'),'Enable','off','value
    ',0);
      microphone_launch_data_exist(2,1) = 0;
  else
      shuttle_mic_sum =
    shuttle_mic_1+shuttle_mic_2+shuttle_mic_3+shuttle_mic_4+shuttle_m
    ic_5;
      microphone_number_string_section = num2str(shuttle_mic_sum);
      microphone_data_x_section = []; %#ok<NASGU>
      microphone_data_y_section = []; %#ok<NASGU>
      microphone_data_z_section = []; %#ok<NASGU>
      temp_x = {'00.000','00.000','00.000','00.000','00.000'};
      temp_y = {'00.000','00.000','00.000','00.000','00.000'};
      temp_z = {'00.000','00.000','00.000','00.000','00.000'};
      microphone_data_x_section = temp_x(1,1:shuttle_mic_sum);
      microphone_data_y_section = temp_y(1,1:shuttle_mic_sum);
      microphone_data_z_section = temp_z(1,1:shuttle_mic_sum);

      microphone_data_r_section = []; %#ok<NASGU>
      microphone_data_h_section = []; %#ok<NASGU>
      microphone_data_v_section = []; %#ok<NASGU>
      temp_r =
    {'00.000','00.000','00.000','00.000','00.000','00.000'};
      temp_h =
    {'00.000','00.000','00.000','00.000','00.000','00.000'};
      temp_v =
    {'00.000','00.000','00.000','00.000','00.000','00.000'};
      microphone_data_r_section = temp_r(1,1:shuttle_mic_sum);
      microphone_data_h_section = temp_h(1,1:shuttle_mic_sum);
      microphone_data_v_section = temp_v(1,1:shuttle_mic_sum);
```

```
  i=0;
  while i<=shuttle_mic_sum
      i=i+1;
      if i==1
          microphone_launch_data_frec = [20 25 31 40 50 63 80 100
125 160 200 250 315 400 500 630 800 1000 1250 1600 2000 2500 3150
4000]';
      end
      if shuttle_mic_1==1 % Mic 7986:
          microphone_data_x_section(1,i) = {'-8.856'};
          microphone_data_y_section(1,i) = {'00.001'}; % ~0
          microphone_data_z_section(1,i) = {'01.640'};
          microphone_data_r_section(1,i) = {'02.494'};
          microphone_data_h_section(1,i) = {'90.000'};
          microphone_data_v_section(1,i) = {'90.000'};
          microphone_launch_data_exist(1,i) = 1; %#ok<AGROW>
          microphone_launch_data_exist(2,i) = 0; %#ok<AGROW>
          if shuttle_mic_1_data==1
              microphone_launch_data_exist(2,i) = 1; %#ok<AGROW>
              microphone_launch_data(:,i) = [141.25   141.25
143.35  141.95  144.15  142.85  143.85  142.85  143.75  148.25
149.85  150.35  150.85  152.25  151.65  150.55  150.45  150.05
151.15  149.85  149.25  148.85  147.35  146.85]'; %#ok<AGROW>
          end
          i=i+1;
      end
      if shuttle_mic_2==1 % Mic 7987:
          microphone_data_x_section(1,i) = {'-8.537'};
          microphone_data_y_section(1,i) = {'00.001'};
          microphone_data_z_section(1,i) = {'03.139'};
          microphone_data_r_section(1,i) = {'02.174'};
          microphone_data_h_section(1,i) = {'90.000'};
          microphone_data_v_section(1,i) = {'90.000'};
          microphone_launch_data_exist(1,i) = 1; %#ok<AGROW>
          microphone_launch_data_exist(2,i) = 0; %#ok<AGROW>
          if shuttle_mic_2_data==1
              microphone_launch_data_exist(2,i) = 1; %#ok<AGROW>
              microphone_launch_data(:,i) = [132.14   132.14
133.34  133.14  133.64  133.54  133.24  135.64  136.94  140.64
142.84  141.04  141.24  143.24  141.84  143.94  143.74  144.44
145.74  146.94  147.34  146.44  141.74  140.44]'; %#ok<AGROW>
          end
          i=i+1;
      end
      if shuttle_mic_3==1 % Mic 7988:
          microphone_data_x_section(1,i) = {'-7.674'};
          microphone_data_y_section(1,i) = {'01.311'};
          microphone_data_z_section(1,i) = {'38.548'};
          microphone_data_r_section(1,i) = {'01.855'};
          microphone_data_h_section(1,i) = {'45.000'};
          microphone_data_v_section(1,i) = {'90.000'};
          microphone_launch_data_exist(1,i) = 1; %#ok<AGROW>
          microphone_launch_data_exist(2,i) = 0; %#ok<AGROW>
          if shuttle_mic_3_data==1
              microphone_launch_data_exist(2,i) = 1; %#ok<AGROW>
```

```
                microphone_launch_data(:,i) = [132.12   132.12
  132.12  132.12  132.12  132.12  132.12  132.12  133.02  133.62
  134.82  133.32  132.92  133.82  133.92  134.42  134.32  134.42
  134.92  135.32  135.52  135.42  134.32  132.62]'; %#ok<AGROW>
              end
              i=i+1;
          end
          if shuttle_mic_4==1 % Mic 8981:
              microphone_data_x_section(1,i) = {'06.363'};
              microphone_data_y_section(1,i) = {'01.855'};
              microphone_data_z_section(1,i) = {'12.155'};
              microphone_data_r_section(1,i) = {'01.855'};
              microphone_data_h_section(1,i) = {'00.000'};
              microphone_data_v_section(1,i) = {'90.000'};
              microphone_launch_data_exist(1,i) = 1; %#ok<AGROW>
              microphone_launch_data_exist(2,i) = 0; %#ok<AGROW>
              if shuttle_mic_4_data==1
                  microphone_launch_data_exist(2,i) = 1; %#ok<AGROW>
                  microphone_launch_data(:,i) = [131.84   131.84
  131.84  131.84  131.84  134.43  135.14  135.54  137.94  140.84
  141.74  137.94  142.34  144.74  143.14  144.84  144.84  144.44
  145.74  146.34  146.64  146.74  146.04  144.54]'; %#ok<AGROW>
              end
              i=i+1;
          end
          if shuttle_mic_5==1 % Mic 8982:
              microphone_data_x_section(1,i) = {'06.363'};
              microphone_data_y_section(1,i) = {'02.174'};
              microphone_data_z_section(1,i) = {'03.164'};
              microphone_data_r_section(1,i) = {'02.174'};
              microphone_data_h_section(1,i) = {'00.000'};
              microphone_data_v_section(1,i) = {'90.000'};
              microphone_launch_data_exist(1,i) = 1; %#ok<AGROW>
              microphone_launch_data_exist(2,i) = 0; %#ok<AGROW>
              if shuttle_mic_5_data==1
                  microphone_launch_data_exist(2,i) = 1; %#ok<AGROW>
                  microphone_launch_data(:,i) = [133.73   134.63
  133.43  135.63  137.63  137.13  137.63  139.13  141.63  143.63
  144.13  142.13  144.63  147.63  146.63  148.13  148.63  148.13
  150.13  150.13  150.93  151.13  149.93  148.63]'; %#ok<AGROW>
              end
              i=i+1;
          end
      end
  end

  engine_thrust_data_section = {'1665859'};
  engine_exit_temperature_data_section = {'1673'};
  engine_exit_diamemter_data_section = {'2.304'};
  engine_exit_velosity_data_section = {'4419.6'};

  engine_data_x_section = {'1.556','0','-1.556'};
  engine_data_y_section = {'7.727','10.625','7.727'};
  engine_data_z_section = {'4.536','4.536','4.536'};
  engine_number_string_section  = '3';
```

```
    ramp_data_y_z_section = {'22','22','22'};
    ramp_data_x_y_section = {'0','0','0'};

elseif data_button_section==5 % Ares I Data:
    % Launch Data, see Verification of Ares I Liftoff Acoustic
      Environments via the Ares I Scale Model Acoustic Test - D.D.
      Counter and J. Houston (2012)

    set(findobj(gcf,'Tag','saturn_mic_1'),'Enable','off','value',0);
    set(findobj(gcf,'Tag','saturn_mic_2'),'Enable','off','value',0);
    set(findobj(gcf,'Tag','saturn_mic_3'),'Enable','off','value',0);
    set(findobj(gcf,'Tag','saturn_mic_4'),'Enable','off','value',0);
    set(findobj(gcf,'Tag','saturn_mic_5'),'Enable','off','value',0);
    set(findobj(gcf,'Tag','saturn_mic_6'),'Enable','off','value',0);
    set(findobj(gcf,'Tag','saturn_mic_9'),'Enable','off','value',0);


      set(findobj(gcf,'Tag','saturn_mic_1_data'),'Enable','off','value'
      ,0);

      set(findobj(gcf,'Tag','saturn_mic_2_data'),'Enable','off','value'
      ,0);

      set(findobj(gcf,'Tag','saturn_mic_3_data'),'Enable','off','value'
      ,0);

      set(findobj(gcf,'Tag','saturn_mic_4_data'),'Enable','off','value'
      ,0);

      set(findobj(gcf,'Tag','saturn_mic_5_data'),'Enable','off','value'
      ,0);

      set(findobj(gcf,'Tag','saturn_mic_6_data'),'Enable','off','value'
      ,0);

    set(findobj(gcf,'Tag','shuttle_mic_1'),'Enable','off','value',0);
    set(findobj(gcf,'Tag','shuttle_mic_2'),'Enable','off','value',0);
    set(findobj(gcf,'Tag','shuttle_mic_3'),'Enable','off','value',0);
    set(findobj(gcf,'Tag','shuttle_mic_4'),'Enable','off','value',0);
    set(findobj(gcf,'Tag','shuttle_mic_5'),'Enable','off','value',0);
    set(findobj(gcf,'Tag','shuttle_mic_9'),'Enable','off','value',0);


      set(findobj(gcf,'Tag','shuttle_mic_1_data'),'Enable','off','value
      ',0);

      set(findobj(gcf,'Tag','shuttle_mic_2_data'),'Enable','off','value
      ',0);

      set(findobj(gcf,'Tag','shuttle_mic_3_data'),'Enable','off','value
      ',0);

      set(findobj(gcf,'Tag','shuttle_mic_4_data'),'Enable','off','value
      ',0);
```

```
  set(findobj(gcf,'Tag','shuttle_mic_5_data'),'Enable','off','value
  ',0);

set(findobj(gcf,'Tag','ares_mic_1'),'Enable','on');
set(findobj(gcf,'Tag','ares_mic_2'),'Enable','on');
set(findobj(gcf,'Tag','ares_mic_9'),'Enable','on');

set(findobj(gcf,'Tag','jet_mic_1'),'Enable','off','value',0);
set(findobj(gcf,'Tag','jet_mic_2'),'Enable','off','value',0);
set(findobj(gcf,'Tag','jet_mic_3'),'Enable','off','value',0);
set(findobj(gcf,'Tag','jet_mic_9'),'Enable','off','value',0);

set(findobj(gcf,'Tag','jet_mic_1_data'),'Enable','off','value',0);
set(findobj(gcf,'Tag','jet_mic_2_data'),'Enable','off','value',0);
set(findobj(gcf,'Tag','jet_mic_3_data'),'Enable','off','value',0);

ares_mic_1 = get(findobj(gcf,'Tag','ares_mic_1'),'value');
ares_mic_2 = get(findobj(gcf,'Tag','ares_mic_2'),'value');
ares_mic_9 = get(findobj(gcf,'Tag','ares_mic_9'),'value');

ares_mic_1_data =
  get(findobj(gcf,'Tag','ares_mic_1_data'),'value');
ares_mic_2_data =
  get(findobj(gcf,'Tag','ares_mic_2_data'),'value');

microphone_offset_section = '24';

if ares_mic_9==1
    microphone_data_x_section = {'0'};
    microphone_data_y_section = {'0'};
    microphone_data_z_section = {'0'};
    microphone_data_r_section = {'0'};
    microphone_data_h_section = {'0'};
    microphone_data_v_section = {'0'};
    microphone_number_string_section = num2str(1);
    set(findobj(gcf,'Tag','ares_mic_1'),'Enable','off','value',0);
    set(findobj(gcf,'Tag','ares_mic_2'),'Enable','off','value',0);

  set(findobj(gcf,'Tag','ares_mic_1_data'),'Enable','off','value',0
  );

  set(findobj(gcf,'Tag','ares_mic_2_data'),'Enable','off','value',0
  );
    microphone_launch_data_exist(2,1) = 0;
else
    ares_mic_sum = ares_mic_1+ares_mic_2;
    microphone_number_string_section = num2str(ares_mic_sum);
    microphone_data_x_section = []; %#ok<NASGU>
    microphone_data_y_section = []; %#ok<NASGU>
    microphone_data_z_section = []; %#ok<NASGU>
    temp_x = {'00.000','00.000','00.000','00.000','00.000'};
    temp_y = {'00.000','00.000','00.000','00.000','00.000'};
    temp_z = {'00.000','00.000','00.000','00.000','00.000'};
    microphone_data_x_section = temp_x(1,1:ares_mic_sum);
```

```
  microphone_data_y_section = temp_y(1,1:ares_mic_sum);
  microphone_data_z_section = temp_z(1,1:ares_mic_sum);

  microphone_data_r_section = []; %#ok<NASGU>
  microphone_data_h_section = []; %#ok<NASGU>
  microphone_data_v_section = []; %#ok<NASGU>
  temp_r = {'00.000','00.000','00.000','00.000','00.000'};
  temp_h = {'00.000','00.000','00.000','00.000','00.000'};
  temp_v = {'00.000','00.000','00.000','00.000','00.000'};
  microphone_data_r_section = temp_r(1,1:ares_mic_sum);
  microphone_data_h_section = temp_h(1,1:ares_mic_sum);
  microphone_data_v_section = temp_v(1,1:ares_mic_sum);

  i=0;
  while i<=ares_mic_sum
      i=i+1;
      if i==1
          microphone_launch_data_frec =
[19.7,24.8,31.3,39.4,49.6,62.5,78.7,99.2,125,157.5,198.4,250,315,
396.9,500,630,793.7,1000,1259.9,1587.4,2000,2519.8]';
      end
      if ares_mic_1==1 % Ares I-X Sensor IAD098P
          microphone_data_x_section(1,i) = {'00.000'};
          microphone_data_y_section(1,i) = {'02.750'};
          microphone_data_z_section(1,i) = {'56.388'};
          microphone_data_r_section(1,i) = {'02.750'};
          microphone_data_h_section(1,i) = {'00.000'};
          microphone_data_v_section(1,i) = {'90.000'};
          microphone_launch_data_exist(1,i) = 1; %#ok<AGROW>
          microphone_launch_data_exist(2,i) = 0; %#ok<AGROW>
          if ares_mic_1_data==1
              microphone_launch_data_exist(2,i) = 1; %#ok<AGROW>
              microphone_launch_data(:,i) =
[128.354400000000,132.616000000000,133.797400000000,135.843900000
000,138.755300000000,139.409300000000,138.713000000000,138.312200
000000,140.600000000000,138.670900000000,138.248900000000,137.151
900000000,135.464100000000,134.071700000000,133.523200000000,132.
088600000000,131.118100000000,129.746800000000,128.523200000000,1
27.025300000000,125.822800000000,123.101300000000]';%#ok<AGROW>
          end
          i=i+1;
      end
      if ares_mic_2==1 % Ares I-X Sensor IAD915P
          microphone_data_x_section(1,i) = {'00.000'};
          microphone_data_y_section(1,i) = {'02.750'};
          microphone_data_z_section(1,i) = {'83.820'};
          microphone_data_r_section(1,i) = {'02.750'};
          microphone_data_h_section(1,i) = {'00.000'};
          microphone_data_v_section(1,i) = {'90.000'};
          microphone_launch_data_exist(1,i) = 1; %#ok<AGROW>
          microphone_launch_data_exist(2,i) = 0; %#ok<AGROW>
          if ares_mic_2_data==1
              microphone_launch_data_exist(2,i) = 1; %#ok<AGROW>
              microphone_launch_data(:,i) =
[122.418100000000,128.523200000000,130.637800000000,133.912000000
000,133.127600000000,136.606400000000,135.429700000000,136.452900
```

```
000000,135.020500000000,134.747600000000,133.588000000000,132.274
900000000,132.752400000000,131.081200000000,128.455000000000,127.
090700000000,125.760600000000,124.976100000000,122.708000000000,1
20.798100000000,0,0]'; %#ok<AGROW>
                end
                i=i+1;
            end
        end
    end


    engine_thrust_data_section = '14634649.1';
    engine_exit_temperature_data_section = '3477'; % used shuttle
    engine_exit_diamemter_data_section = '4.401';
    engine_exit_velosity_data_section = '2529.84';

    engine_number_string_section  = '1';
    engine_data_x_section = '0';
    engine_data_y_section = '0';
    engine_data_z_section = '0';

    ramp_data_y_z_section = '22';
    ramp_data_x_y_section = '0';

elseif data_button_section==6 % Sonic Jet Data:
    % see ?

    set(findobj(gcf,'Tag','saturn_mic_1'),'Enable','off','value',0);
    set(findobj(gcf,'Tag','saturn_mic_2'),'Enable','off','value',0);
    set(findobj(gcf,'Tag','saturn_mic_3'),'Enable','off','value',0);
    set(findobj(gcf,'Tag','saturn_mic_4'),'Enable','off','value',0);
    set(findobj(gcf,'Tag','saturn_mic_5'),'Enable','off','value',0);
    set(findobj(gcf,'Tag','saturn_mic_6'),'Enable','off','value',0);
    set(findobj(gcf,'Tag','saturn_mic_9'),'Enable','off','value',0);


      set(findobj(gcf,'Tag','saturn_mic_1_data'),'Enable','off','value'
      ,0);

      set(findobj(gcf,'Tag','saturn_mic_2_data'),'Enable','off','value'
      ,0);

      set(findobj(gcf,'Tag','saturn_mic_3_data'),'Enable','off','value'
      ,0);

      set(findobj(gcf,'Tag','saturn_mic_4_data'),'Enable','off','value'
      ,0);

      set(findobj(gcf,'Tag','saturn_mic_5_data'),'Enable','off','value'
      ,0);

      set(findobj(gcf,'Tag','saturn_mic_6_data'),'Enable','off','value'
      ,0);

    set(findobj(gcf,'Tag','shuttle_mic_1'),'Enable','off','value',0);
```

```
set(findobj(gcf,'Tag','shuttle_mic_2'),'Enable','off','value',0);
set(findobj(gcf,'Tag','shuttle_mic_3'),'Enable','off','value',0);
set(findobj(gcf,'Tag','shuttle_mic_4'),'Enable','off','value',0);
set(findobj(gcf,'Tag','shuttle_mic_5'),'Enable','off','value',0);
set(findobj(gcf,'Tag','shuttle_mic_9'),'Enable','off','value',0);


  set(findobj(gcf,'Tag','shuttle_mic_1_data'),'Enable','off','value
  ',0);

  set(findobj(gcf,'Tag','shuttle_mic_2_data'),'Enable','off','value
  ',0);

  set(findobj(gcf,'Tag','shuttle_mic_3_data'),'Enable','off','value
  ',0);

  set(findobj(gcf,'Tag','shuttle_mic_4_data'),'Enable','off','value
  ',0);

  set(findobj(gcf,'Tag','shuttle_mic_5_data'),'Enable','off','value
  ',0);

set(findobj(gcf,'Tag','ares_mic_1'),'Enable','off','value',0);
set(findobj(gcf,'Tag','ares_mic_2'),'Enable','off','value',0);
set(findobj(gcf,'Tag','ares_mic_9'),'Enable','off','value',0);

set(findobj(gcf,'Tag','ares_mic_1_data'),'Enable','off','value',0);
set(findobj(gcf,'Tag','ares_mic_2_data'),'Enable','off','value',0);

set(findobj(gcf,'Tag','jet_mic_1'),'Enable','on');
set(findobj(gcf,'Tag','jet_mic_2'),'Enable','on');
set(findobj(gcf,'Tag','jet_mic_3'),'Enable','on');
set(findobj(gcf,'Tag','jet_mic_9'),'Enable','on');

jet_mic_1 = get(findobj(gcf,'Tag','jet_mic_1'),'value');
jet_mic_2 = get(findobj(gcf,'Tag','jet_mic_2'),'value');
jet_mic_3 = get(findobj(gcf,'Tag','jet_mic_3'),'value');
jet_mic_9 = get(findobj(gcf,'Tag','jet_mic_9'),'value');

jet_mic_1_data = get(findobj(gcf,'Tag','jet_mic_1_data'),'value');
jet_mic_2_data = get(findobj(gcf,'Tag','jet_mic_2_data'),'value');
jet_mic_3_data = get(findobj(gcf,'Tag','jet_mic_3_data'),'value');

microphone_offset_section = '00';

if jet_mic_9==1
    microphone_data_x_section = {'0'};
    microphone_data_y_section = {'0'};
    microphone_data_z_section = {'0'};
    microphone_data_r_section = {'0'};
    microphone_data_h_section = {'0'};
    microphone_data_v_section = {'0'};
    microphone_number_string_section = num2str(1);
    set(findobj(gcf,'Tag','jet_mic_1'),'Enable','off','value',0);
    set(findobj(gcf,'Tag','jet_mic_2'),'Enable','off','value',0);
```

```
    set(findobj(gcf,'Tag','jet_mic_3'),'Enable','off','value',0);

  set(findobj(gcf,'Tag','jet_mic_1_data'),'Enable','off','value',0)
  ;

  set(findobj(gcf,'Tag','jet_mic_2_data'),'Enable','off','value',0)
  ;

  set(findobj(gcf,'Tag','jet_mic_3_data'),'Enable','off','value',0)
  ;
    microphone_launch_data_exist(2,1) = 0;
else
    jet_mic_sum = jet_mic_1+jet_mic_2+jet_mic_3;
    microphone_number_string_section = num2str(jet_mic_sum);
    microphone_data_x_section = []; %#ok<NASGU>
    microphone_data_y_section = []; %#ok<NASGU>
    microphone_data_z_section = []; %#ok<NASGU>
    temp_x = {'00.000','00.000','00.000','00.000','00.000'};
    temp_y = {'00.000','00.000','00.000','00.000','00.000'};
    temp_z = {'00.000','00.000','00.000','00.000','00.000'};
    microphone_data_x_section = temp_x(1,1:jet_mic_sum);
    microphone_data_y_section = temp_y(1,1:jet_mic_sum);
    microphone_data_z_section = temp_z(1,1:jet_mic_sum);

    microphone_data_r_section = []; %#ok<NASGU>
    microphone_data_h_section = []; %#ok<NASGU>
    microphone_data_v_section = []; %#ok<NASGU>
    temp_r = {'00.000','00.000','00.000','00.000','00.000'};
    temp_h = {'00.000','00.000','00.000','00.000','00.000'};
    temp_v = {'00.000','00.000','00.000','00.000','00.000'};
    microphone_data_r_section = temp_r(1,1:jet_mic_sum);
    microphone_data_h_section = temp_h(1,1:jet_mic_sum);
    microphone_data_v_section = temp_v(1,1:jet_mic_sum);

    i=0;
    while i<=jet_mic_sum
        i=i+1;
        if i==1
            microphone_launch_data_frec = [0]';
        end
        if jet_mic_1==1 % Theta = 135 deg
            microphone_data_x_section(1,i) = {'00.000'};
            microphone_data_y_section(1,i) = {'03.505'};
            microphone_data_z_section(1,i) = {'03.505'};
            microphone_data_r_section(1,i) = {'00.000'};
            microphone_data_h_section(1,i) = {'00.000'};
            microphone_data_v_section(1,i) = {'00.000'};
            microphone_launch_data_exist(1,i) = 1; %#ok<AGROW>
            microphone_launch_data_exist(2,i) = 0; %#ok<AGROW>
            if jet_mic_1_data==1
                microphone_launch_data_exist(2,i) = 1; %#ok<AGROW>
                microphone_launch_data(:,i) = [0]';%#ok<AGROW>
            end
            i=i+1;
        end
        if jet_mic_2==1 % Theta = 90 deg
```

```
                microphone_data_x_section(1,i) = {'00.000'};
                microphone_data_y_section(1,i) = {'03.505'};
                microphone_data_z_section(1,i) = {'00.042'};
                microphone_data_r_section(1,i) = {'00.000'};
                microphone_data_h_section(1,i) = {'00.000'};
                microphone_data_v_section(1,i) = {'00.000'};
                microphone_launch_data_exist(1,i) = 1; %#ok<AGROW>
                microphone_launch_data_exist(2,i) = 0; %#ok<AGROW>
                if jet_mic_2_data==1
                    microphone_launch_data_exist(2,i) = 1; %#ok<AGROW>
                    microphone_launch_data(:,i) = [0]'; %#ok<AGROW>
                end
                i=i+1;
            end
            if jet_mic_3==1 % Theta = 45 deg
                microphone_data_x_section(1,i) = {'00.000'};
                microphone_data_y_section(1,i) = {'03.505'};
                microphone_data_z_section(1,i) = {'-3.588'};
                microphone_data_r_section(1,i) = {'00.000'};
                microphone_data_h_section(1,i) = {'00.000'};
                microphone_data_v_section(1,i) = {'00.000'};
                microphone_launch_data_exist(1,i) = 1; %#ok<AGROW>
                microphone_launch_data_exist(2,i) = 0; %#ok<AGROW>
                if jet_mic_3_data==1
                    microphone_launch_data_exist(2,i) = 1; %#ok<AGROW>
                    microphone_launch_data(:,i) = [0]'; %#ok<AGROW>
                end
                i=i+1;
            end
        end
    end


    engine_thrust_data_section = '794.28';
    engine_exit_temperature_data_section = '311';
    engine_exit_diamemter_data_section = '0.073';
    engine_exit_velosity_data_section = '406.57';

    engine_number_string_section  = '1';
    engine_data_x_section = '0';
    engine_data_y_section = '0';
    engine_data_z_section = '0';

    ramp_data_y_z_section = '-90';
    ramp_data_x_y_section = '0';

elseif data_button_section==9 % User Defined Data:
    set(findobj(gcf,'Tag','saturn_mic_1'),'Enable','off');
    set(findobj(gcf,'Tag','saturn_mic_2'),'Enable','off');
    set(findobj(gcf,'Tag','saturn_mic_3'),'Enable','off');
    set(findobj(gcf,'Tag','saturn_mic_4'),'Enable','off');
    set(findobj(gcf,'Tag','saturn_mic_5'),'Enable','off');
    set(findobj(gcf,'Tag','saturn_mic_6'),'Enable','off');
    set(findobj(gcf,'Tag','saturn_mic_9'),'Enable','off');

    set(findobj(gcf,'Tag','shuttle_mic_1'),'Enable','off');
```

```
    set(findobj(gcf,'Tag','shuttle_mic_2'),'Enable','off');
    set(findobj(gcf,'Tag','shuttle_mic_3'),'Enable','off');
    set(findobj(gcf,'Tag','shuttle_mic_4'),'Enable','off');
    set(findobj(gcf,'Tag','shuttle_mic_5'),'Enable','off');
    set(findobj(gcf,'Tag','shuttle_mic_9'),'Enable','off');

    set(findobj(gcf,'Tag','ares_mic_1'),'Enable','off');
    set(findobj(gcf,'Tag','ares_mic_9'),'Enable','off');

    microphone_number_string_section = '1';
    microphone_offset_section = '0';
    microphone_data_x_section = '0';
    microphone_data_y_section = '0';
    microphone_data_z_section = '0';

    microphone_data_r_section = '0';
    microphone_data_h_section = '0';
    microphone_data_v_section = '0';

    engine_thrust_data_section = '0';
    engine_exit_temperature_data_section = '0';
    engine_exit_diamemter_data_section = '0';
    engine_exit_velosity_data_section = '0';

    engine_number_string_section  = '1';
    engine_data_x_section = '0';
    engine_data_y_section = '0';
    engine_data_z_section = '0';

    ramp_data_y_z_section = '0';
    ramp_data_x_y_section = '0';

else
    error('"Data Button Callback" funtion in "data button section"
      logic at line ~2258')
end

end

%.....................................................................
      ....

function Mic_Data_Button_Callback(hObject, eventdata, handles)

global test_b test_c
if exist('eventdata','var')
    test_b = eventdata;
end
if exist('handles','var')
    test_c = handles;
end

tag_name = get(hObject,'Tag');
data_name = '_data';
```

```
tag_data_name = str2mat(strcat(tag_name,data_name));

data_button_section =
      get(get(findobj(gcf,'Tag','data_button_section'),'SelectedObject'
      ),'UserData');

if (get(hObject,'Value') == get(hObject,'Max')) && (data_button_section
      == 1 || data_button_section == 2 || data_button_section == 3 ||
      data_button_section == 4 || data_button_section == 5 ||
      data_button_section == 6)
    % if checkbox is checked
    set(findobj(gcf,'Tag',tag_data_name),'Enable','on');
else % if checkbox is not checked
    set(findobj(gcf,'Tag',tag_data_name),'Enable','off','value',0);
end

end

%.....................................................................
      ....

function Pres_Doub_Button_Callback(hObject, eventdata, handles)

global test_b test_c
if exist('eventdata','var')
    test_b = eventdata;
end
if exist('handles','var')
    test_c = handles;
end

if get(hObject,'Value') == 1
    % if checkbox is checked

      set(findobj(gcf,'Tag','pressure_doubling_radius_data'),'Enable','
      on');

      set(findobj(gcf,'Tag','pressure_doubling_horizontal_data'),'Enabl
      e','on');

      set(findobj(gcf,'Tag','pressure_doubling_vertical_data'),'Enable'
      ,'on');
    set(findobj(gcf,'Tag','pressure_doubling_display'),'Enable','on');
else % if checkbox is not checked

      set(findobj(gcf,'Tag','pressure_doubling_radius_data'),'Enable','
      off');

      set(findobj(gcf,'Tag','pressure_doubling_horizontal_data'),'Enabl
      e','off');

      set(findobj(gcf,'Tag','pressure_doubling_vertical_data'),'Enable'
      ,'off');
```

```
      set(findobj(gcf,'Tag','pressure_doubling_display'),'Enable','off'
      ,'value',0);
end

end

%.....................................................................
      ....

function Exhaust_Panel_Callback(hObject, eventdata)

global test_a test_b
if exist('hObject','var')
    test_a = hObject;
end
if exist('eventdata','var')
    test_b = eventdata;
end

global engine_thrust_data_section ...
       engine_exit_temperature_data_section ...
       engine_exit_diamemter_data_section ...
       engine_exit_velosity_data_section ...
       engine_number_string_section ...
       shuttle_and_booster ...
       microphone_offset_section

if
      get(get(findobj(gcf,'Tag','data_button_section'),'SelectedObject'
      ),'UserData')==4 && shuttle_and_booster==2
   % nothing
else
    Data_Button_Callback
end
set(findobj(gcf,'Tag','data_panel'),'Visible','off');

exhaust_panel = uipanel( ...
    'Visible','off',...
    'Title','Exhaust Information', ...
    'ForegroundColor','red',...
    'FontSize',16,...
    'BackgroundColor','blue',...
    'Tag','exhaust_panel',...
    'Units','normalized',...
    'Position',[0.05,0.05,0.9,0.9]);

text_left     = [-0.10  0     0.10 0.03];
text_bottom   = [-0.10 -0.04  0.20 0.03];
input_right   = [ 0     0     0.10 0.04];

% F: Thrust of each Engine
position_list = [0.11 0.90 0 0];
uicontrol('Parent',exhaust_panel,...
    'Style','text',...
```

```
    'Units','normalized',...
    'Position',position_list+text_left,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','F (N)');
uicontrol('Parent',exhaust_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_bottom,...
    'FontSize',10,...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','(thrust of each engine)');
uicontrol(...
    'Parent',exhaust_panel,...
    'Style','edit',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','white',...
    'Tag','engine_thrust',...
    'ForegroundColor','black',...
    'String',engine_thrust_data_section);

% Te: Engine Exit Temperature
position_list = [0.11 0.80 0 0];
uicontrol('Parent',exhaust_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_left,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Te (K)');
uicontrol('Parent',exhaust_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_bottom,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','(exit temperature)');
uicontrol(...
    'Parent',exhaust_panel,...
    'Style','edit',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','white',...
    'Tag','engine_exit_temperature',...
    'ForegroundColor','black',...
    'String',engine_exit_temperature_data_section);

% de: Engine Exit Diamemter
position_list = [0.11 0.70 0 0];
```

```
uicontrol('Parent',exhaust_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_left,...
    'FontSize',10,...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','de (m)');
uicontrol('Parent',exhaust_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_bottom,...
    'FontSize',10,...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','(exit nozzle diamemter)');
uicontrol(...
    'Parent',exhaust_panel,...
    'Style','edit',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','white',...
    'Tag','engine_exit_diamemter',...
    'ForegroundColor','black',...
    'String',engine_exit_diamemter_data_section);

% Ue: Engine Exit Velosity
position_list = [0.11 0.60 0 0];
uicontrol('Parent',exhaust_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_left,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Ue (m/s)');
uicontrol('Parent',exhaust_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+[-0.10 -0.04 0.2 0.03],...
    'FontSize',10,...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','(fully expanded exit velosity)');
uicontrol(...
    'Parent',exhaust_panel,...
    'Style','edit',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','white',...
    'Tag','engine_exit_velosity',...
    'ForegroundColor','black',...
    'String',engine_exit_velosity_data_section);
```

```
% n: Number of Engines
position_list = [0.35 0.90 0 0];
uicontrol('Parent',exhaust_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_left,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','n');
uicontrol('Parent',exhaust_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_bottom,...
    'FontSize',10,...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','(number of engines)');
uicontrol(...
    'Parent',exhaust_panel,...
    'Style','edit',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','white',...
    'Tag','engine_number',...
    'ForegroundColor','black',...
    'Callback',{@Location_Each_Engine_Callback},...
    'CreateFcn',{@Location_Each_Engine_Callback},...
    'String',engine_number_string_section);

% Engine and Ramp panels are located in function
% "Location_Each_Engine_Callback"

% Microphones offset
position_list = [0.65 0.90 0 0];
uicontrol('Parent',exhaust_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_left,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Z Offset:');
uicontrol('Parent',exhaust_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_bottom,...
    'FontSize',10,...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','(relative height of vehical)');
uicontrol(...
    'Parent',exhaust_panel,...
    'Style','edit',...
    'Units','normalized',...
```

```
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','white',...
    'Tag','microphone_offset',...
    'ForegroundColor','black',...
    'String',microphone_offset_section);

uicontrol(...
    'Parent',exhaust_panel,...
    'Style','pushbutton',...
    'Units','normalized',...
    'Position',[0.325 0.01 0.15 0.05],...
    'Tag','back_data_pushbutton',...
    'Callback',{@Back_Data_Panel_Callback},...
    'FontSize',16,...
    'ForegroundColor','black',...
    'String','Back');

uicontrol(...
    'Parent',exhaust_panel,...
    'Style','pushbutton',...
    'Units','normalized',...
    'Position',[0.525 0.01 0.15 0.05],...
    'Tag','calculate_pushbutton',...
    'Callback',{@Microphone_Panel_Callback},...
    'FontSize',16,...
    'ForegroundColor','black',...
    'String','Next');

set(exhaust_panel,'Visible','on');

end

%.....................................................................
      ....

function Microphone_Panel_Callback(source,eventdata)
% When "Bandwidth Choice" in pushed the currently function is called.
global test_a test_b
test_a = source;
test_b = eventdata;

global microphone_number_string_section


set(findobj(gcf,'Tag','exhaust_panel'),'Visible','off');

text_left     = [-0.10  0     0.10 0.03];
text_bottom   = [-0.10 -0.04  0.20 0.03];
input_right   = [ 0     0     0.10 0.04];

microphone_panel = uipanel( ...
    'Visible','off',...
    'Title','Microphone Information', ...
    'ForegroundColor','red',...
```

```
    'FontSize',16,...
    'BackgroundColor','blue',...
    'Tag','microphone_panel',...
    'Units','normalized',...
    'Position',[0.05,0.05,0.9,0.9]);

% Directivity five button group:
directivity_button_section = uibuttongroup(...
    'Parent',microphone_panel,...
    'BackgroundColor','blue',...
    'Units','normalized',...
    'Position',[0.05 0.69 0.33 0.25],...
    'FontSize',10,...
    'ForegroundColor','white',...
    'Tag','directivity_button_section',...
    'Title','Directivity Section');

% Eldred Directivity (Wes's curve fit):
uicontrol(...
    'Parent',directivity_button_section,...
    'Style','radiobutton',...
    'Units','normalized',...
    'Position',[0.05 0.78 0.90 0.16],...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'Tag','directivity_cedrics_curve_fit',...
    'ForegroundColor','white',...
    'Value',0,...
    'UserData',1,...
    'String','Eldred NASA SP-8072 (1971)');

% Wilby Memo #4 Eq. 4.12 Directivity:
uicontrol(...
    'Parent',directivity_button_section,...
    'Style','radiobutton',...
    'Units','normalized',...
    'Position',[0.05 0.60 0.90 0.16],...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'Tag','directivity_wilby_4_12',...
    'ForegroundColor','white',...
    'Value',0,...
    'UserData',2,...
    'String','Wilby Memo 179-14 #4 Eq 4.12 (2006)');

% Wilby Memo #4 Eq. 4.13 Directivity:
uicontrol(...
    'Parent',directivity_button_section,...
    'Style','radiobutton',...
    'Units','normalized',...
    'Position',[0.05 0.42 0.90 0.16],...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'Tag','directivity_wilby_4_13',...
    'ForegroundColor','white',...
    'Value',0,...
```

```
    'UserData',3,...
    'String','Wilby Memo 179-14 #4 Eq 4.13 (2006)');

% Wilby Memo #4 Eq. 4.14 Directivity:
uicontrol(...
    'Parent',directivity_button_section,...
    'Style','radiobutton',...
    'Units','normalized',...
    'Position',[0.05 0.24 0.90 0.16],...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'Tag','directivity_wilby_4_14',...
    'ForegroundColor','white',...
    'Value',1,...
    'UserData',4,...
    'String','Wilby Memo 179-14 #4 Eq 4.14 (2006)');

% Plotkin and Sutherland Directivity:
uicontrol(...
    'Parent',directivity_button_section,...
    'Style','radiobutton',...
    'Units','normalized',...
    'Position',[0.05 0.06 0.90 0.16],...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'Tag','directivity_plotkin_sutherland',...
    'ForegroundColor','white',...
    'Value',0,...
    'UserData',5,...
    'String','Plotkin-Sutherland Power-Point (2008)');

% Number of microphones
position_list = [0.11 0.50 0 0];
uicontrol('Parent',microphone_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_left,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Microphones:');
uicontrol('Parent',microphone_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_bottom,...
    'FontSize',10,...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','(number of microphones)');
uicontrol(...
    'Parent',microphone_panel,...
    'Style','edit',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','white',...
```

```
    'Tag','microphone_number',...
    'ForegroundColor','black',...
    'Callback',{@Location_Each_Microphone_Callback},...
    'CreateFcn',{@Location_Each_Microphone_Callback},...
    'String',microphone_number_string_section);

% Microphone Location panel is located in function
% "Location_Each_Microphone_Callback"

% Surface Diffraction:
uicontrol(...
    'Parent',microphone_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',[0.45 0.50 0.25 0.05],...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','pressure_doubling_checkbox',...
    'Value',0,...
    'String','Surface Diffraction',...
    'Callback',@Pres_Doub_Button_Callback);

% Pressure Doubling Displayed to user:
uicontrol(...
    'Parent',microphone_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',[0.60 0.50 0.25 0.05],...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','pressure_doubling_display',...
    'Value',0,...
    'String','Display dB',...
    'Callback',@Pres_Doub_Button_Callback);

% Pressure Doubling panel is located in function
% "Location_Each_Microphone_Callback"

uicontrol(...
    'Parent',microphone_panel,...
    'Style','pushbutton',...
    'Units','normalized',...
    'Position',[0.325 0.01 0.15 0.05],...
    'Tag','back_data_pushbutton',...
    'Callback',{@Back_Exhaust_Panel_Callback},...
    'FontSize',16,...
    'ForegroundColor','black',...
    'String','Back');

uicontrol(...
    'Parent',microphone_panel,...
    'Style','pushbutton',...
    'Units','normalized',...
    'Position',[0.525 0.01 0.15 0.05],...
```

```
    'Tag','calculate_pushbutton',...
    'Callback',{@Calculation_Panel_Callback},...
    'FontSize',16,...
    'ForegroundColor','black',...
    'String','Next');

set(microphone_panel,'Visible','on');
end

%.....................................................................
      ....

function Calculation_Panel_Callback(source,eventdata)
global test_a test_b
test_a = source;
test_b = eventdata;

set(findobj(gcf,'Tag','microphone_panel'),'Visible','off');

text_left     = [-0.10  0     0.10 0.03];
text_bottom   = [-0.10 -0.04  0.20 0.03];
input_right   = [ 0     0     0.10 0.04];

radio_right   = [ 0     0     0.15 0.04];

calculation_panel = uipanel( ...
    'Visible','off',...
    'Title','Calculation Information', ...
    'ForegroundColor','red',...
    'FontSize',16,...
    'BackgroundColor','blue',...
    'Tag','calculation_panel',...
    'Units','normalized',...
    'Position',[0.05,0.05,0.9,0.9]);

% Number Exhanst Splits
position_list = [0.11 0.60 0 0];
uicontrol('Parent',calculation_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_left,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Flow Split');
uicontrol('Parent',calculation_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_bottom,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','(Number Exhanst Splits)');
uicontrol(...
    'Parent',calculation_panel,...
```

```
    'Style','edit',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','white',...
    'Tag','split_flow',...
    'ForegroundColor','black',...
    'String','1');

% Ta: Ambient Temperature
position_list = [0.11 0.50 0 0];
uicontrol('Parent',calculation_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_left,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Ta (K)');
uicontrol('Parent',calculation_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_bottom,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','(ambient temperature)');
uicontrol(...
    'Parent',calculation_panel,...
    'Style','edit',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','white',...
    'Tag','ambient_temperature',...
    'ForegroundColor','black',...
    'String','300');

% Sound Power Efficiency
position_list = [0.11 0.40 0 0];
uicontrol('Parent',calculation_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_left,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','eta (dec)');
uicontrol('Parent',calculation_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_bottom,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','(rocket acoustic efficiency)');
```

```
uicontrol(...
    'Parent',calculation_panel,...
    'Style','edit',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','white',...
    'Tag','sound_power_efficiency',...
    'ForegroundColor','black',...
    'String','0.01');

% delta X: Slice of the Flow
position_list = [0.11 0.30 0 0];
uicontrol('Parent',calculation_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_left,...
    'FontSize',10,...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','delta x (m)');
uicontrol('Parent',calculation_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_bottom,...
    'FontSize',10,...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','(slice of the flow)');
uicontrol(...
    'Parent',calculation_panel,...
    'Style','edit',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','white',...
    'Tag','slice_flow',...
    'ForegroundColor','black',...
    'String','0.5');

% X1: Starting Position
position_list = [0.11 0.20 0 0];
uicontrol('Parent',calculation_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_left,...
    'FontSize',10,...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','x1 (m)');
uicontrol('Parent',calculation_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+[-0.11 -0.04 0.23 0.03],...
    'FontSize',10,...
    'ForegroundColor','white',...
```

```
    'BackgroundColor','blue',...
    'String','(starting position for calculation)');
uicontrol(...
    'Parent',calculation_panel,...
    'Style','edit',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','white',...
    'Tag','starting_position',...
    'ForegroundColor','black',...
    'String',get(findobj(gcf,'Tag','microphone_offset'),'String'));

% X / de: Maximum apparent source axial Position
position_list = [0.11 0.10 0 0];
uicontrol('Parent',calculation_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_left,...
    'FontSize',10,...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','x / de');
uicontrol('Parent',calculation_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+[-0.10 -0.04 0.3 0.03],...
    'FontSize',10,...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','(maximum apparent source axial position)');
uicontrol(...
    'Parent',calculation_panel,...
    'Style','edit',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','white',...
    'Tag','apparent_source',...
    'ForegroundColor','black',...
    'String','50');

% Plot data:
position_list = [0.40 0.90 0.20 0.03];
uicontrol('Parent',calculation_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Plot Data');

position_list = [0.50 0.86 0 0];
uicontrol(...
    'Parent',calculation_panel,...
```

```
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+radio_right,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','plot_position_noise_sources',...
    'Value',0,...
    'String','Noise Position');

position_list = [0.50 0.82 0 0];
uicontrol(...
    'Parent',calculation_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+radio_right,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','plot_each_engine',...
    'Value',0,...
    'String','Each Engine');

position_list = [0.50 0.78 0 0];
uicontrol(...
    'Parent',calculation_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+radio_right,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','plot_OASPL',...
    'Value',1,...
    'String','OASPL');

position_list = [0.50 0.74 0 0];
uicontrol(...
    'Parent',calculation_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',position_list+radio_right,...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','plot_sum_engines',...
    'Value',1,...
    'String','Sum All Engines');

% fmax: Maximun Frequency
position_list = [0.89 0.90 0 0];
uicontrol('Parent',calculation_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_left,...
    'FontSize',10, ...
```

```
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','fmax (Hz)');
uicontrol('Parent',calculation_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_bottom,...
    'FontSize',10,...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','(maximun frequency)');
uicontrol(...
    'Parent',calculation_panel,...
    'Style','edit',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','white',...
    'Tag','maximun_frequency',...
    'ForegroundColor','black',...
    'String','20000');%20000

% fc: Center Frequency
position_list = [0.89 0.80 0 0];
uicontrol('Parent',calculation_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_left,...
    'FontSize',10,...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','fc (Hz)');
uicontrol('Parent',calculation_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_bottom,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','(center frequency)');
uicontrol(...
    'Parent',calculation_panel,...
    'Style','edit',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','white',...
    'Tag','center_frequency',...
    'ForegroundColor','black',...
    'String','1000');

% fmin: Minimum Frequency
position_list = [0.89 0.70 0 0];
uicontrol('Parent',calculation_panel,...
    'Style','text',...
    'Units','normalized',...
```

```
    'Position',position_list+text_left,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','fmin (Hz)');
uicontrol('Parent',calculation_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',position_list+text_bottom,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','(minimum frequency)');
uicontrol(...
    'Parent',calculation_panel,...
    'Style','edit',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','white',...
    'Tag','minimum_frequency',...
    'ForegroundColor','black',...
    'String','20');%20

% Octave three button group:
octave_button_group = uibuttongroup(...
    'Parent',calculation_panel,...
    'BackgroundColor','blue',...
    'Units','normalized',...
    'Position',[0.76 0.49 0.13 0.15],...
    'Tag','octave_button_section',...
    'Title','',...
    'SelectionChangeFcn',{@N_Octave_Callback});

% 1/N Octave selection:
uicontrol(...
    'Parent',octave_button_group,...
    'Style','radiobutton',...
    'Units','normalized',...
    'Position',[0.05 0.65 0.90 0.30],...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'UserData',1,...
    'Value',0,...
    'String','1/N');

% 1/3 Octave selection:
uicontrol(...
    'Parent',octave_button_group,...
    'Style','radiobutton',...
    'Units','normalized',...
    'Position',[0.05 0.35 0.90 0.30],...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
```

```
    'UserData',2,...
    'Value',1,...
    'String','Third Octave');

% Full Octave selection:
uicontrol(...
    'Parent',octave_button_group,...
    'Style','radiobutton',...
    'Units','normalized',...
    'Position',[0.05 0.05 0.90 0.30],...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'UserData',3,...
    'Value',0,...
    'String','Full Octave');

% 1/N Octave:
position_list = [0.89 0.60 0 0];
uicontrol(...
    'Parent',calculation_panel,...
    'Style','edit',...
    'Units','normalized',...
    'Position',position_list+input_right,...
    'FontSize',10, ...
    'BackgroundColor','white',...
    'ForegroundColor','black',...
    'Tag','N_octave_string',...
    'Enable','off',...
    'String','6');

% Summing power three button group:
power_button_section = uibuttongroup(...
    'Parent',calculation_panel,...
    'BackgroundColor','blue',...
    'Units','normalized',...
    'Position',[0.76 0.31 0.23 0.15],...
    'Tag','power_button_section',...
    'Title','');

% Summing total power:
uicontrol(...
    'Parent',power_button_section,...
    'Style','radiobutton',...
    'Units','normalized',...
    'Position',[0.05 0.65 0.90 0.30],...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'Tag','total_power_selection',...
    'ForegroundColor','white',...
    'Value',1,...
    'UserData',1,...
    'String','Total Power');

% Summing power before ramp:
uicontrol(...
```

```
    'Parent',power_button_section,...
    'Style','radiobutton',...
    'Units','normalized',...
    'Position',[0.05 0.35 0.90 0.30],...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'Tag','before_power_selection',...
    'ForegroundColor','white',...
    'UserData',2,...
    'String','Power Before Ramp');

% Summing power after ramp:
uicontrol(...
    'Parent',power_button_section,...
    'Style','radiobutton',...
    'Units','normalized',...
    'Position',[0.05 0.05 0.90 0.30],...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'Tag','after_power_selection',...
    'ForegroundColor','white',...
    'UserData',3,...
    'String','Power After Ramp');

% Saving data:
uicontrol(...
    'Parent',calculation_panel,...
    'Style','checkbox',...
    'Units','normalized',...
    'Position',[0.74 0.24 0.25 0.05],...
    'FontSize',10, ...
    'BackgroundColor','blue',...
    'ForegroundColor','white',...
    'Tag','saving_data_file',...
    'Value',0,...
    'String','Save Data');

% Automatic Method: ????
% uicontrol(...
%     'Parent',calculation_panel,...
%     'Style','checkbox',...
%     'Units','normalized',...
%     'Position',[0.74 0.18 0.25 0.05],...
%     'FontSize',10, ...
%     'BackgroundColor','blue',...
%     'ForegroundColor','white',...
%     'Tag','automatic_method',...
%     'Value',1,...
%     'String','Automatic Method');

% Reflection Effect: ????
% uicontrol(...
%     'Parent',calculation_panel,...
%     'Style','checkbox',...
%     'Units','normalized',...
%     'Position',[0.74 0.12 0.25 0.05],...
```

```
%     'FontSize',10, ...
%     'BackgroundColor','blue',...
%     'ForegroundColor','white',...
%     'Tag','reflection_effect',...
%     'Value',0,...
%     'String','The effect of reflection off the');
% uicontrol('Parent',calculation_panel,...
%     'Style','text',...
%     'Units','normalized',...
%     'Position',[0.74 0.07 0.25 0.05],...
%     'FontSize',10, ...
%     'BackgroundColor','blue',...
%     'ForegroundColor','white',...
%     'String','ground is included in the prediction.');

uicontrol(...
    'Parent',calculation_panel,...
    'Style','pushbutton',...
    'Units','normalized',...
    'Position',[0.325 0.01 0.15 0.05],...
    'Tag','back_data_pushbutton',...
    'Callback',{@Back_Microphone_Panel_Callback},...
    'FontSize',16,...
    'ForegroundColor','black',...
    'String','Back');

uicontrol(...
    'Parent',calculation_panel,...
    'Style','pushbutton',...
    'Units','normalized',...
    'Position',[0.525 0.01 0.15 0.05],...
    'Tag','calculate_pushbutton',...
    'Callback',{@Exhaust_Panel_Check_Callback},...
    'FontSize',16,...
    'ForegroundColor','black',...
    'String','Calculate');

set(calculation_panel,'Visible','on');
end

%.....................................................................
      ....

function Exhaust_Panel_Check_Callback(source,eventdata)
% When "Bandwidth Choice" in pushed the currently function is called.
global test_a test_b
test_a = source;
test_b = eventdata;

global engine_thrust_data_section ...
       engine_exit_temperature_data_section ...
       engine_exit_diamemter_data_section ...
       engine_exit_velosity_data_section ...
       engine_data_x_section ...
       engine_data_y_section ...
       engine_data_z_section ...
```

```
        engine_number_string_section ...
        ramp_data_y_z_section ...
        ramp_data_x_y_section ...
        data_button_section ...
        shuttle_and_booster

set(findobj(gcf,'Tag','calculation_panel'),'Visible','off');
Checking_Data
data_check=1;
% must convert using str2double
ambient_temperature =
      str2double(get(findobj(gcf,'Tag','ambient_temperature'),'String')
      );
[m,n] = size(ambient_temperature);
if isnan(ambient_temperature) || ambient_temperature<=0 || m>1 || n>1
    ambient_temperature
    Something_Wrong
    set(findobj(gcf,'Tag','calculation_panel'),'Visible','on');
    data_check=0;
end
sound_power_efficiency =
      str2double(get(findobj(gcf,'Tag','sound_power_efficiency'),'Strin
      g'));
[m,n] = size(sound_power_efficiency);
if isnan(sound_power_efficiency) || sound_power_efficiency<=0 || m>1 ||
      n>1
    sound_power_efficiency
    Something_Wrong
    set(findobj(gcf,'Tag','calculation_panel'),'Visible','on');
    data_check=0;
end
engine_number =
      str2double(get(findobj(gcf,'Tag','engine_number'),'String'));
[m,n] = size(engine_number);
if isnan(engine_number) || engine_number<=0 || m>1 || n>1
    engine_number
    Something_Wrong
    set(findobj(gcf,'Tag','calculation_panel'),'Visible','on');
    data_check=0;
end
engine_thrust =
      str2double(get(findobj(gcf,'Tag','engine_thrust'),'String'));
[m,n] = size(engine_thrust);
if isnan(engine_thrust) || engine_thrust<=0 || m>1 || n>1
    engine_thrust
    Something_Wrong
    set(findobj(gcf,'Tag','calculation_panel'),'Visible','on');
    data_check=0;
end
engine_exit_temperature =
      str2double(get(findobj(gcf,'Tag','engine_exit_temperature'),'Stri
      ng'));
[m,n] = size(engine_exit_temperature);
if isnan(engine_exit_temperature) || engine_exit_temperature<=0 || m>1
      || n>1
    engine_exit_temperature
    Something_Wrong
```

```
    set(findobj(gcf,'Tag','calculation_panel'),'Visible','on');
    data_check=0;
end
engine_exit_diamemter =
      str2double(get(findobj(gcf,'Tag','engine_exit_diamemter'),'String
      '));
[m,n] = size(engine_exit_diamemter);
if isnan(engine_exit_diamemter) || engine_exit_diamemter<=0 || m>1 ||
      n>1
    engine_exit_diamemter
    Something_Wrong
    set(findobj(gcf,'Tag','calculation_panel'),'Visible','on');
    data_check=0;
end
engine_exit_velosity =
      str2double(get(findobj(gcf,'Tag','engine_exit_velosity'),'String'
      ));
[m,n] = size(engine_exit_velosity);
if isnan(engine_exit_velosity) || engine_exit_velosity<=0 || m>1 || n>1
    engine_exit_velosity
    Something_Wrong
    set(findobj(gcf,'Tag','calculation_panel'),'Visible','on');
    data_check=0;
end
slice_flow = str2double(get(findobj(gcf,'Tag','slice_flow'),'String'));
[m,n] = size(slice_flow);
if isnan(slice_flow) || slice_flow<=0 || m>1 || n>1
    slice_flow
    Something_Wrong
    set(findobj(gcf,'Tag','calculation_panel'),'Visible','on');
    data_check=0;
end
apparent_source =
      str2double(get(findobj(gcf,'Tag','apparent_source'),'String'));
[m,n] = size(apparent_source);
if isnan(apparent_source) || apparent_source<=0 || m>1 || n>1
    apparent_source
    Something_Wrong
    set(findobj(gcf,'Tag','calculation_panel'),'Visible','on');
    data_check=0;
end
starting_position =
      str2double(get(findobj(gcf,'Tag','starting_position'),'String'));
[m,n] = size(starting_position);
if isnan(starting_position) || starting_position<0 || m>1 || n>1
    starting_position
    Something_Wrong
    set(findobj(gcf,'Tag','calculation_panel'),'Visible','on');
    data_check=0;
end
microphone_offset =
      str2double(get(findobj(gcf,'Tag','microphone_offset'),'String'));
[m,n] = size(microphone_offset);
if isnan(microphone_offset) || microphone_offset<0 || m>1 || n>1
    microphone_offset
    Something_Wrong
    set(findobj(gcf,'Tag','calculation_panel'),'Visible','on');
```

```
    data_check=0;
end
if microphone_offset~=starting_position
    microphone_offset
    starting_position
    Something_Wrong
    set(findobj(gcf,'Tag','calculation_panel'),'Visible','on');
    data_check=0;
end

if data_check==1
    Ok_Data
    Acoustic_Loads_subroutines_2_13
    if data_button_section==4 && shuttle_and_booster==2
        delete(findobj(gcf,'Tag','exhaust_panel'));
        delete(findobj('Tag','microphone_panel'));
        delete(findobj(gcf,'Tag','calculation_panel'));

        engine_thrust_data_section = {'11787790'};
        engine_exit_temperature_data_section = {'3477'};
        engine_exit_diamemter_data_section = {'3.700'};
        engine_exit_velosity_data_section = {'2331.72'};
        engine_data_x_section = {'6.363','-6.363'};
        engine_data_y_section = {'0','0'};
        engine_data_z_section = {'0','0'};
        engine_number_string_section  = '2';
        ramp_data_y_z_section = {'22','22'};
        ramp_data_x_y_section = {'0','0'};

        Exhaust_Panel_Callback
    else
        Run_Complete
    end
end

end

%.....................................................................
      ....

function Ground_Reflection_Panel_Callback(hObject, eventdata)
% Calculate ????
global test_a test_b
test_a = hObject;
test_b = eventdata;

set(findobj(gcf,'Tag','main_menu'),'Visible','off');
set(findobj(gcf,'Tag','exhaust_menu'),'Visible','on');
set(findobj(gcf,'Tag','AU_Logo'),'Visible','off');
set(findobj(gcf,'Tag','main_text'),'Visible','off');

Not_Complete_Text
end
```

```
%.....................................................................
      ....

function Sum_Exhaust_Panel_Callback(hObject, eventdata)
% Calculate ????
global test_a test_b
test_a = hObject;
test_b = eventdata;

set(findobj(gcf,'Tag','main_menu'),'Visible','off');
set(findobj(gcf,'Tag','exhaust_menu'),'Visible','on');
set(findobj(gcf,'Tag','AU_Logo'),'Visible','off');
set(findobj(gcf,'Tag','main_text'),'Visible','off');

Not_Complete_Text
end

%.....................................................................
      ....

function Location_Each_Engine_Callback(hObject, eventdata)
%Location of each Engine:
global test_a test_b
test_a = hObject;
test_b = eventdata;

global engine_data_x_section ...
       engine_data_y_section ...
       engine_data_z_section ...
       engine_number_string_section ...
       ramp_data_y_z_section ...
       ramp_data_x_y_section

delete(findobj(gcf,'Tag','engine_number_panel'));
delete(findobj(gcf,'Tag','ramp_number_panel'));
exhaust_panel = findobj(gcf,'Tag','exhaust_panel');
engine_number_string =
      str2double(get(findobj(gcf,'Tag','engine_number'),'String'));
data_button_section =
      get(get(findobj(gcf,'Tag','data_button_section'),'SelectedObject'
      ),'UserData');

if data_button_section==9 || engine_number_string ~=
      str2double(engine_number_string_section)
    engine_data_x(1:engine_number_string,1)='0';
    engine_data_y(1:engine_number_string,1)='0';
    engine_data_z(1:engine_number_string,1)='0';
    ramp_data_y_z(1:engine_number_string,1)='0';
    ramp_data_x_y(1:engine_number_string,1)='0';
else
    engine_data_x = engine_data_x_section;
    engine_data_y = engine_data_y_section;
    engine_data_z = engine_data_z_section;
    ramp_data_y_z = ramp_data_y_z_section;
    ramp_data_x_y = ramp_data_x_y_section;
```

```
end

engine_number_panel = uipanel(...
    'Parent',exhaust_panel,...
    'Visible','off',...
    'ForegroundColor','red',...
    'FontSize',12,...
    'BackgroundColor','blue',...
    'Tag','engine_number_panel',...
    'Units','normalized',...
    'Position',[0.25 0.56 0.30 0.30],...
    'Title','Engine Location (m)');

first_column = [0.05 0.89 0.25 0.10];
middle_column = [0.375 0.89 0.25 0.10];
last_column = [0.70 0.89 0.25 0.10];

uicontrol('Parent',engine_number_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',first_column,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','X-location');
uicontrol(...
    'Parent',engine_number_panel,...
    'Style','edit',...
    'Units','normalized',...
    'Position',first_column+[0 -0.87 0 0.75],...
    'FontSize',10, ...
    'BackgroundColor','white',...
    'ForegroundColor','black',...
    'Tag','engine_x_data',...
    'Max',engine_number_string,'Min',0,...
    'String',engine_data_x);

uicontrol('Parent',engine_number_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',middle_column,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Y-location');
uicontrol(...
    'Parent',engine_number_panel,...
    'Style','edit',...
    'Units','normalized',...
    'Position',middle_column+[0 -0.87 0 0.75],...
    'FontSize',10, ...
    'BackgroundColor','white',...
    'ForegroundColor','black',...
    'Tag','engine_y_data',...
    'Max',engine_number_string,'Min',0,...
    'String',engine_data_y);
```

```
uicontrol('Parent',engine_number_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',last_column,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Z-location');
uicontrol(...
    'Parent',engine_number_panel,...
    'Style','edit',...
    'Units','normalized',...
    'Position',last_column+[0 -0.87 0 0.75],...
    'FontSize',10, ...
    'BackgroundColor','white',...
    'ForegroundColor','black',...
    'Tag','engine_z_data',...
    'Max',engine_number_string,'Min',0,...
    'String',engine_data_z);

right_column = [0.10 0.89 0.35 0.10];
left_column  = [0.55 0.89 0.35 0.10];

ramp_number_panel = uipanel(...
    'Parent',exhaust_panel,...
    'Visible','off',...
    'ForegroundColor','red',...
    'FontSize',12,...
    'BackgroundColor','blue',...
    'Tag','ramp_number_panel',...
    'Units','normalized',...
    'Position',[0.55 0.56 0.20 0.30],...
    'Title','Apparent Flow Angle (deg)');

uicontrol('Parent',ramp_number_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',right_column,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','X-Y');
uicontrol(...
    'Parent',ramp_number_panel,...
    'Style','edit',...
    'Units','normalized',...
    'Position',right_column+[0 -0.87 0 0.75],...
    'FontSize',10, ...
    'BackgroundColor','white',...
    'ForegroundColor','black',...
    'Tag','ramp_x_y_data',...
    'Max',engine_number_string,'Min',0,...
    'String',ramp_data_x_y);

uicontrol('Parent',ramp_number_panel,...
```

```
    'Style','text',...
    'Units','normalized',...
    'Position',left_column,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Y-Z');
uicontrol(...
    'Parent',ramp_number_panel,...
    'Style','edit',...
    'Units','normalized',...
    'Position',left_column+[0 -0.87 0 0.75],...
    'FontSize',10, ...
    'BackgroundColor','white',...
    'ForegroundColor','black',...
    'Tag','ramp_y_z_data',...
    'Max',engine_number_string,'Min',0,...
    'String',ramp_data_y_z);

set(engine_number_panel,'Visible','on');
set(ramp_number_panel,'Visible','on');
end

%.....................................................................
      ....

function Location_Each_Microphone_Callback(hObject, eventdata)
%Location of each Microphone:
global test_a test_b
test_a = hObject;
test_b = eventdata;

global microphone_number_string_section ...
       microphone_data_x_section ...
       microphone_data_y_section ...
       microphone_data_z_section ...
       microphone_data_r_section ...
       microphone_data_h_section ...
       microphone_data_v_section ...
       microphone_launch_data_exist

delete(findobj(gcf,'Tag','microphone_number_panel'));
microphone_panel = findobj(gcf,'Tag','microphone_panel');
microphone_number_string =
      str2double(get(findobj(gcf,'Tag','microphone_number'),'String'));
data_button_section =
      get(get(findobj(gcf,'Tag','data_button_section'),'SelectedObject'
      ),'UserData');

if data_button_section==9 || (microphone_number_string ~=
      str2double(char(microphone_number_string_section)))
    microphone_data_x(1:microphone_number_string,1) = '0';
    microphone_data_y(1:microphone_number_string,1) = '0';
    microphone_data_z(1:microphone_number_string,1) = '0';
    microphone_data_r(1:microphone_number_string,1) = '0';
    microphone_data_h(1:microphone_number_string,1) = '0';
```

```
    microphone_data_v(1:microphone_number_string,1) = '0';
    microphone_launch_data_exist(1,1:microphone_number_string) = 1;
    microphone_launch_data_exist(2,1:microphone_number_string) = 0;
else
    microphone_data_x = microphone_data_x_section;
    microphone_data_y = microphone_data_y_section;
    microphone_data_z = microphone_data_z_section;
    microphone_data_r = microphone_data_r_section;
    microphone_data_h = microphone_data_h_section;
    microphone_data_v = microphone_data_v_section;
end

first_column  = [0.05 0.89 0.25 0.10];
middle_column = [0.375 0.89 0.25 0.10];
last_column   = [0.70 0.89 0.25 0.10];

microphone_number_panel = uipanel(...
    'Parent',microphone_panel,...
    'Visible','off',...
    'ForegroundColor','red',...
    'FontSize',12,...
    'BackgroundColor','blue',...
    'Tag','microphone_number_panel',...
    'Units','normalized',...
    'Position',[0.06 0.16 0.30 0.30],...
    'Title','Microphone Location (m)');

uicontrol('Parent',microphone_number_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',first_column,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','X-location');
uicontrol(...
    'Parent',microphone_number_panel,...
    'Style','edit',...
    'Units','normalized',...
    'Position',first_column+[0 -0.87 0 0.75],...
    'FontSize',10, ...
    'BackgroundColor','white',...
    'ForegroundColor','black',...
    'Tag','microphone_x_data',...
    'Max',microphone_number_string,'Min',0,...
    'String',microphone_data_x);

uicontrol('Parent',microphone_number_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',middle_column,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Y-location');
uicontrol(...
```

```
    'Parent',microphone_number_panel,...
    'Style','edit',...
    'Units','normalized',...
    'Position',middle_column+[0 -0.87 0 0.75],...
    'FontSize',10, ...
    'BackgroundColor','white',...
    'ForegroundColor','black',...
    'Tag','microphone_y_data',...
    'Max',microphone_number_string,'Min',0,...
    'String',microphone_data_y);

uicontrol('Parent',microphone_number_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',last_column,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Z-location');
uicontrol(...
    'Parent',microphone_number_panel,...
    'Style','edit',...
    'Units','normalized',...
    'Position',last_column+[0 -0.87 0 0.75],...
    'FontSize',10, ...
    'BackgroundColor','white',...
    'ForegroundColor','black',...
    'Tag','microphone_z_data',...
    'Max',microphone_number_string,'Min',0,...
    'String',microphone_data_z);

pressure_doubling_panel = uipanel(...
    'Parent',microphone_panel,...
    'Visible','off',...
    'ForegroundColor','red',...
    'FontSize',12,...
    'BackgroundColor','blue',...
    'Tag','pressure_doubling_panel',...
    'Units','normalized',...
    'Position',[0.40 0.16 0.30 0.30],...
    'Title','Surface Diffraction Information');

uicontrol('Parent',pressure_doubling_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',first_column,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Radius (m)');
uicontrol(...
    'Parent',pressure_doubling_panel,...
    'Style','edit',...
    'Units','normalized',...
    'Position',first_column+[0 -0.87 0 0.75],...
    'FontSize',10, ...
```

```
    'BackgroundColor','white',...
    'ForegroundColor','black',...
    'Tag','pressure_doubling_radius_data',...
    'Max',microphone_number_string,'Min',0,...
    'String',microphone_data_r);

uicontrol('Parent',pressure_doubling_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',middle_column,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Horiz. (deg)');
uicontrol(...
    'Parent',pressure_doubling_panel,...
    'Style','edit',...
    'Units','normalized',...
    'Position',middle_column+[0 -0.87 0 0.75],...
    'FontSize',10, ...
    'BackgroundColor','white',...
    'ForegroundColor','black',...
    'Tag','pressure_doubling_horizontal_data',...
    'Max',microphone_number_string,'Min',0,...
    'String',microphone_data_h);

uicontrol('Parent',pressure_doubling_panel,...
    'Style','text',...
    'Units','normalized',...
    'Position',last_column,...
    'FontSize',10, ...
    'ForegroundColor','white',...
    'BackgroundColor','blue',...
    'String','Vert. (deg)');
uicontrol(...
    'Parent',pressure_doubling_panel,...
    'Style','edit',...
    'Units','normalized',...
    'Position',last_column+[0 -0.87 0 0.75],...
    'FontSize',10, ...
    'BackgroundColor','white',...
    'ForegroundColor','black',...
    'Tag','pressure_doubling_vertical_data',...
    'Max',microphone_number_string,'Min',0,...
    'String',microphone_data_v);

set(microphone_number_panel,'Visible','on');
set(pressure_doubling_panel,'Visible','on');

end

%.....................................................................
      ....

function N_Octave_Callback(hObject, eventdata)
% alpha number: ????
```

```
global test_a test_b
test_a = hObject;
test_b = eventdata;

octave_section =
      get(get(findobj(gcf,'Tag','octave_button_section'),'SelectedObjec
      t'),'UserData');

if octave_section==1
    set(findobj(gcf,'Tag','N_octave_string'),'Enable','on');
else
    set(findobj(gcf,'Tag','N_octave_string'),'Enable','off');
end

end

%.....................................................................
      ....

function Back_Main_Menu_Callback(source,eventdata)
global test_a test_b
if exist('source','var')
    test_a = source;
end
if exist('eventdata','var')
    test_b = eventdata;
end

delete(findobj('Tag','data_panel'));
delete(findobj('Tag','exhaust_panel'));
delete(findobj('Tag','microphone_panel'));
delete(findobj('Tag','calculation_panel'));

set(findobj('Tag','main_menu'),'Visible','on');
set(findobj('Tag','exhaust_menu'),'Visible','off');
set(findobj('Tag','AU_Logo'),'Visible','on');
set(findobj('Tag','main_figure_text'),'Visible','on');
set(findobj('Tag','main_text'),'Visible','on');

end

%.....................................................................
      ....

function Back_Data_Panel_Callback(source,eventdata)
global test_a test_b
test_a = source;
test_b = eventdata;

delete(findobj(gcf,'Tag','exhaust_panel'));
set(findobj(gcf,'Tag','data_panel'),'Visible','on');

end
```

```
%.....................................................................
      ....

function Back_Exhaust_Panel_Callback(source,eventdata)
global test_a test_b
test_a = source;
test_b = eventdata;

delete(findobj(gcf,'Tag','microphone_panel'));
set(findobj(gcf,'Tag','exhaust_panel'),'Visible','on');

end

%.....................................................................
      ....

function Back_Microphone_Panel_Callback(source,eventdata)
global test_a test_b
test_a = source;
test_b = eventdata;

delete(findobj(gcf,'Tag','calculation_panel'));
set(findobj(gcf,'Tag','microphone_panel'),'Visible','on');

end

%.....................................................................
      ....

function Not_Complete_Text
not_complete_text = uicontrol(...
    'Style','text',...
    'Units','normalized',...
    'Position',[0.375 0.475 0.25 0.05],...
    'FontSize',16,...
    'ForegroundColor','red',...
    'BackgroundColor','white',...
    'String','NOT COMPLETE');
pause(2)
delete(not_complete_text);
Back_Main_Menu_Callback
end

%.....................................................................
      ....

function Something_Wrong
set(findobj(gcf,'Tag','checking_data'),'Visible','off');
something_wrong = uicontrol(...
    'Style','text',...
    'Units','normalized',...
    'Position',[0.25 0.475 0.50 0.05],...
    'FontSize',16,...
    'ForegroundColor','red',...
    'BackgroundColor','white',...
```

```
    'String','Some entered data is wrong or missing!');
pause(2)
delete(something_wrong);
end

%.....................................................................
      ....

function Checking_Data
uicontrol(...
    'Style','text',...
    'Tag','checking_data',...
    'Units','normalized',...
    'Position',[0.375 0.475 0.25 0.05],...
    'FontSize',16,...
    'ForegroundColor','red',...
    'BackgroundColor','white',...
    'String','Checking entered data');
pause(0.1)
end

%.....................................................................
      ....

function Ok_Data
delete(findobj(gcf,'Tag','checking_data'));
ok_data = uicontrol(...
    'Style','text',...
    'Tag','ok_data',...
    'Units','normalized',...
    'Position',[0.325 0.475 0.35 0.05],...
    'FontSize',16,...
    'ForegroundColor','red',...
    'BackgroundColor','white',...
    'String','All data entered is satisfactory');
pause(0.1)
delete(ok_data);
end

%.....................................................................
      ....

function Run_Complete
global test_a test_b

main_window = findobj('Tag','Acoustic_Load_main_window');
run_complete = uicontrol(...
    'Parent',main_window,...
    'Style','text',...
    'Tag','run_complete',...
    'Units','normalized',...
    'Position',[0.25 0.45 0.50 0.10],...
    'FontSize',16,...
    'ForegroundColor','red',...
    'BackgroundColor','white',...
```

```
    'String',{'All data calculations are complete.',...
    'Please return to the main menu or exit'});
pause(0.1)
delete(run_complete);
% Back_Main_Menu_Callback
% close(gcf,findobj('Tag','Acoustic_Load_main_window'))
Exit_Callback(test_a,test_b)

end

%.....................................................................
      ....

function Exit_Callback(source,eventdata)
% Close only the current figure.
% gcf: get-current-figure
global test_a test_b
test_a = source;
test_b = eventdata;
close(findobj('Tag','Acoustic_Load_main_window'));
end

%.....................................................................
      ....

function Help_Callback(hObject, eventdata)
% Opens the help menu
global test_a test_b
test_a = hObject;
test_b = eventdata;

open('Help for Acoustic Loads generated by the propulsion system.htm');
end

%.....................................................................
      ....

end
```

Appendix 1b Eldred Empirical Model Matlab Code (part b):

Acoustic_Loads_subroutines_2_13.m

```
function Acoustic_Loads_subroutines_2_13

% From function "Acoustic_Loads_subroutines_2_13"
global ambient_temperature ...
       sound_power_efficiency ...
       engine_thrust ...
       engine_exit_temperature ...
       engine_exit_diamemter ...
       engine_exit_velosity ...
       flow_slice_width ...
       flow_apparent_source_length ...
       engine_number ...
       engine_ramp_data ...
       microphone_number ...
       microphone_data ...
       microphone_offset ...
       microphone_launch_data_exist ...
       microphone_launch_data_frec ...
       microphone_launch_data_frec_width ...
       microphone_launch_data ...
       directivity_section ...
       pressure_doubling_effect ...
       pressure_doubling_display ...
       microphone_r_data ...
       microphone_h_data ...
       microphone_v_data ...
       plot_position_noise_sources ...
       plot_each_engine ...
       plot_OASPL ...
       plot_sum_engines ...
       frequency_maximun ...
       frequency_center ...
       frequency_minimum ...
       frequency_octave_size ...
       defection_point_power ...
       save_data ...
       reflection_effect

ambient_temperature =
      str2double(get(findobj(gcf,'Tag','ambient_temperature'),'String')
      );
sound_power_efficiency =
      str2double(get(findobj(gcf,'Tag','sound_power_efficiency'),'Strin
      g'));
engine_thrust =
      str2double(get(findobj(gcf,'Tag','engine_thrust'),'String'));
engine_exit_temperature =
      str2double(get(findobj(gcf,'Tag','engine_exit_temperature'),'Stri
      ng'));
```

```
engine_exit_diamemter =
      str2double(get(findobj(gcf,'Tag','engine_exit_diamemter'),'String
      '));
engine_exit_velosity =
      str2double(get(findobj(gcf,'Tag','engine_exit_velosity'),'String'
      ));
flow_slice_width =
      str2double(get(findobj(gcf,'Tag','slice_flow'),'String'));
% split_flow =
      str2double(get(findobj(gcf,'Tag','split_flow'),'String'));
% starting_position =
      str2double(get(findobj(gcf,'Tag','starting_position'),'String'));
flow_apparent_source_length =
      str2double(get(findobj(gcf,'Tag','apparent_source'),'String'));

engine_number =
      str2double(get(findobj(gcf,'Tag','engine_number'),'String'));
engine_x_data =
      str2double(get(findobj(gcf,'Tag','engine_x_data'),'String'));
engine_y_data =
      str2double(get(findobj(gcf,'Tag','engine_y_data'),'String'));
engine_z_data =
      str2double(get(findobj(gcf,'Tag','engine_z_data'),'String'));

ramp_y_z_data =
      str2double(get(findobj(gcf,'Tag','ramp_y_z_data'),'String'));
ramp_x_y_data =
      str2double(get(findobj(gcf,'Tag','ramp_x_y_data'),'String'));

microphone_offset =
      str2double(get(findobj(gcf,'Tag','microphone_offset'),'String'));
microphone_number =
      str2double(get(findobj(gcf,'Tag','microphone_number'),'String'));
microphone_x_data =
      str2double(get(findobj(gcf,'Tag','microphone_x_data'),'String'));
microphone_y_data =
      str2double(get(findobj(gcf,'Tag','microphone_y_data'),'String'));
microphone_z_data =
      str2double(get(findobj(gcf,'Tag','microphone_z_data'),'String'));

engine_ramp_data = zeros(engine_number,5);
engine_ramp_data(1:engine_number,1) = engine_x_data;
engine_ramp_data(1:engine_number,2) = engine_y_data;
engine_ramp_data(1:engine_number,3) = engine_z_data +
      microphone_offset;
engine_ramp_data(1:engine_number,4) = ramp_y_z_data; % Y-Z: 22
engine_ramp_data(1:engine_number,5) = ramp_x_y_data; % X-Y: 0

microphone_data = zeros(microphone_number,3);
microphone_data(1:microphone_number,1) = microphone_x_data;
microphone_data(1:microphone_number,2) = microphone_y_data;
microphone_data(1:microphone_number,3) = microphone_z_data +
      microphone_offset;

% These come from the "Data_Button_Callback" function in the
      Acoustic_Loads_2_13 file
```

```
microphone_launch_data_exist; %#ok<VUNUS>
microphone_launch_data_frec;  %#ok<VUNUS>
microphone_launch_data_frec_width; %#ok<VUNUS>
microphone_launch_data;       %#ok<VUNUS>

directivity_section =
      get(get(findobj(gcf,'Tag','directivity_button_section'),'Selected
      Object'),'UserData');

pressure_doubling_effect =
      get(findobj(gcf,'Tag','pressure_doubling_checkbox'),'Value');
pressure_doubling_display =
      get(findobj(gcf,'Tag','pressure_doubling_display'),'Value');
microphone_r_data =
      str2double(get(findobj(gcf,'Tag','pressure_doubling_radius_data')
      ,'String'));
microphone_h_data =
      str2double(get(findobj(gcf,'Tag','pressure_doubling_horizontal_da
      ta'),'String'));
microphone_v_data =
      str2double(get(findobj(gcf,'Tag','pressure_doubling_vertical_data
      '),'String'));

plot_position_noise_sources =
      get(findobj(gcf,'Tag','plot_position_noise_sources'),'Value');
plot_each_engine = get(findobj(gcf,'Tag','plot_each_engine'),'Value');
plot_OASPL = get(findobj(gcf,'Tag','plot_OASPL'),'Value');
plot_sum_engines = get(findobj(gcf,'Tag','plot_sum_engines'),'Value');

frequency_maximun =
      str2double(get(findobj(gcf,'Tag','maximun_frequency'),'String'));
frequency_center  =
      str2double(get(findobj(gcf,'Tag','center_frequency'),'String'));
frequency_minimum =
      str2double(get(findobj(gcf,'Tag','minimum_frequency'),'String'));

octave_section  =
      get(get(findobj(gcf,'Tag','octave_button_section'),'SelectedObjec
      t'),'UserData');
N_octave_string =
      str2double(get(findobj(gcf,'Tag','N_octave_string'),'String'));
if octave_section==1
    frequency_octave_size = N_octave_string; % 1/N_octave_string octave
elseif octave_section==2
    frequency_octave_size = 3;               % 1/3 octave
elseif octave_section==3
    frequency_octave_size = 1;               % 1 octave
else
    error('ERROR in "octave button section" in "Acoustic Loads
      subroutines" logic from the three button group in the
      "Directivity Panel Callback" function  \n')
end

defection_point_power =
      get(get(findobj(gcf,'Tag','power_button_section'),'SelectedObject
      '),'UserData');
```

```
save_data         = get(findobj(gcf,'Tag','saving_data_file'),'Value');
% automatic_method  =
      get(findobj(gcf,'Tag','automatic_method'),'Value');
% reflection_effect =
      get(findobj(gcf,'Tag','reflection_effect'),'Value');
reflection_effect = 0;

% global approach2_loop_counter
% refldi_number valamp_number valphase_number ...
%     multif_number lfen_number file_number

% approach2_loop_counter = 1; % a logic loop counter for approach2
% lfen_number   = 1; % a logic loop counter for the number of loops
      through approach2
% file_number   = 2; % which method: one (1) or two (2))
% refldi_number = 0; % directivity in reflexion

% valamp_number   = 1; % ???????????????
% valphase_number = 0; % ???????????????
% multif_number   = 1; % ???????????????

calculbt2_Callback

end

%.....................................................................
      ....

function calculbt2_Callback
% This second method is used when an acoustical shielding between the
% flow and the vehicle must be accouted for.

% From function "Main_Program" part-2
% global approach2_loop_counter
% multif_number lfen_number

% if approach2_loop_counter==1
    bandwidth;
% end

approach2;

end

%.....................................................................
      ....

function bandwidth

%%%%%%% Bandwidth analysis %%%%%%%

% From function "Acoustic_Loads_subroutines_2_13"
global frequency_maximun ...
       frequency_center ...
```

```
        frequency_minimum ...
        frequency_octave_size

% From function "bandwidth"
global frequency_band_width frequency_band_center

i=1;
ii=0;

bandwidth_ratio = 2.^(1/frequency_octave_size);

Freqc = zeros(1,9);
Freqc(1,1) = frequency_center;
while Freqc(1,1)>=frequency_minimum
    Freqc(1,1) = Freqc(1,1)/bandwidth_ratio;
end
while Freqc(1,i)<=frequency_maximun
    i=i+1;
    Freqc(1,i) = bandwidth_ratio*Freqc(1,i-1);
end

deltafc = zeros(1,(i-1));
while ii<i
    ii=ii+1;
    deltafc(1,ii) = Freqc(1,ii).*((bandwidth_ratio-
      1)./(bandwidth_ratio).^(1/2));
end

frequency_band_width = deltafc'; % The band width of each 1/N octaves
      section based on Freqc.
frequency_band_center = Freqc;   % This is the center frequency of the
      bandwidth divided into 1/N octaves.
% Freq_number or Freq were the lower frequency of a spectrum bandwidth
      divided into 1/N octaves.

end

%.....................................................................
      ....

function approach2
%%%%% Approach 2

% From function "Acoustic_Loads_subroutines_2_13"
global ambient_temperature sound_power_efficiency engine_thrust ...
       engine_exit_temperature engine_exit_diamemter
      engine_exit_velosity ...
       flow_slice_width flow_apparent_source_length engine_number
      engine_ramp_data microphone_number ...
       microphone_data microphone_offset pressure_doubling_effect ...
       plot_position_noise_sources plot_OASPL ...
       plot_sum_engines octave_size frequency_maximun frequency_minimum
      ...
       defection_point_power save_data reflection_effect
```

```
% From function "Acoustic_Loads_subroutines_2_13" part-2
% global approach2_loop_counter
% refldi_number valamp_number valphase_number ...
%        multif_number lfen_number

% From function "bandwidth"
global frequency_band_width frequency_band_center
frequency_band_center_length = length(frequency_band_center);
% size(frequency_band_center) % (1,31)

% From function "Data_Button_Callback" in "Acoustic_Loads_2_13"
global microphone_launch_data_exist microphone_launch_data_frec
      microphone_launch_data_frec_width microphone_launch_data
      data_button_section

% From function "Second_Method_Multi_Exhaust_Panel_Callback" in
      "Acoustic_Loads_2_13"
global shuttle_and_booster

% From function "approach2" and only used if data_button_section=4
global SPLbpint_shuttle_and_booster

% From function "approach2"
% global SPLsbptotcm SPLsbptotsdcm

%%%%% Eldred, NASA SP-8072, p. 28 %%%%% step #4, Eq. 3
de = (engine_number).^(0.5).*engine_exit_diamemter; % equivalent exit
      diameter of a multi-engine vehicle
St = frequency_band_center'.*de./engine_exit_velosity; % Strouhal
      number

%%%%% Eldred, NASA SP-8072, p. 28 %%%%% step #1, refer to p. 20 Fig. 17
%%%% Determine the flow axis reative to the vehicle %%%%

%%%%% Eldred, NASA SP-8072, p. 30 %%%%% step #6, refer to p. 20 Fig. 17
%%%% Allocate the acoustic sources in the middle of each slice %%%%

apparent_engine_ramp_data = zeros(1,5);
apparent_engine_ramp_data(1,1) = 0; % apparent X position of of the
      engine exit
apparent_engine_ramp_data(1,2) = 0; % apparent Y position of of the
      engine exit
apparent_engine_ramp_data(1,3) = engine_ramp_data(1,3); % apparent Z
      position of of the engine exit to the ground plane
apparent_engine_ramp_data(1,4) = engine_ramp_data(1,4); % apparent flow
      angle (deg) in the Y-Z plane
apparent_engine_ramp_data(1,5) = engine_ramp_data(1,5); % apparent flow
      angle (deg) in the X-Y plane NOT USED

% Before defector ramp
X_flow_axis = zeros(1,3);
X_flow_axis_distance = zeros(1,1);
temp_source_location = apparent_engine_ramp_data(1,3) -
      flow_slice_width*0.5;
b=1;
```

```
X_flow_axis(b,1) = apparent_engine_ramp_data(1,1); % assumed position
      in the X-direction of the flow
X_flow_axis(b,2) = apparent_engine_ramp_data(1,2); % assumed position
      in the Y-direction of the flow
X_flow_axis(b,3) = temp_source_location;           % assumed position
      in the Z-direction of the flow
X_flow_axis_distance(b,1) = flow_slice_width*0.5;           % length
      along the flow

temp_source_location = temp_source_location - flow_slice_width;
while temp_source_location>=0
    b=b+1;
    X_flow_axis(b,1) = apparent_engine_ramp_data(1,1); % assumed
      position in the X-direction of the flow
    X_flow_axis(b,2) = apparent_engine_ramp_data(1,2); % assumed
      position in the Y-direction of the flow
    X_flow_axis(b,3) = temp_source_location;           % assumed
      position in the Z-direction of the flow
    X_flow_axis_distance(b,1) = X_flow_axis_distance(b-1,1) +
      flow_slice_width; % length along the flow
    temp_source_location = temp_source_location - flow_slice_width;
end

% After defector ramp
temp_source_location = flow_slice_width*0.5;
b=b+1;
X_flow_axis(b,1) = apparent_engine_ramp_data(1,1);
      % assumed position in the X-direction of the flow
X_flow_axis(b,2) = temp_source_location *
      cosd(apparent_engine_ramp_data(1,4)); % assumed position in the
      Y-direction of the flow
X_flow_axis(b,3) = temp_source_location *
      sind(apparent_engine_ramp_data(1,4)); % assumed position in the
      Z-direction of the flow
X_flow_axis_distance(b,1) = X_flow_axis_distance(b-1,1) +
      X_flow_axis(b-1,3) + flow_slice_width*0.5; % length along the
      flow
X_flow_axis_after_ramp_point = b;

temp_source_location = temp_source_location + flow_slice_width;
while X_flow_axis_distance(b,1)<=(flow_apparent_source_length*de)
    b=b+1;
    X_flow_axis(b,1) = apparent_engine_ramp_data(1,1);
      % assumed position in the X-direction of the flow
    X_flow_axis(b,2) = temp_source_location *
      cosd(apparent_engine_ramp_data(1,4)); % assumed position in the
      Y-direction of the flow
    X_flow_axis(b,3) = temp_source_location *
      sind(apparent_engine_ramp_data(1,4)); % assumed position in the
      Z-direction of the flow
    X_flow_axis_distance(b,1) = X_flow_axis_distance(b-1,1) +
      flow_slice_width; % length along the flow
    temp_source_location = temp_source_location + flow_slice_width;
end
X_flow_axis_length = b;
```

```
if plot_position_noise_sources==1 %&& approach2_loop_counter==1
    figure('Name','Position of the noise sources')
    hold on
    for i=1:microphone_number

      plot3(microphone_data(i,1),microphone_data(i,2),microphone_data(i
      ,3),'ok')
    end
    plot3(X_flow_axis(:,1),X_flow_axis(:,2),X_flow_axis(:,3),'*r')
    view(75,15);
    grid on
    X_plot_max = max(max(max(X_flow_axis)))+10;
    xlabel('X')
    ylabel('Y')
    zlabel('Z')
    axis([-X_plot_max/2 X_plot_max/2 -10 X_plot_max -10 X_plot_max])
end

%%%%% Eldred, NASA SP-8072, p. 28 %%%%% step #2, Eq. 1
if data_button_section==6
    K_1 = 0.3E-4;%1.2E-5;
    rho_1 = 1.1354;
    c_1 = 353.5408;
    Woa =
      K_1*rho_1*(engine_exit_diamemter.^2).*(engine_exit_velosity.^4)./
      (c_1);
%     Woa = 18.6150

      0.5.*sound_power_efficiency.*engine_number.*engine_thrust.*engine
      _exit_velosity;
else
    Woa =
      0.5.*sound_power_efficiency.*engine_number.*engine_thrust.*engine
      _exit_velosity; % overall acoustic power, Watt
end

%%%%% Eldred, NASA SP-8072, p. 28 %%%%% step #3, Eq. 2
Lw = 10.*log10(Woa) + 120; % overall sound power level, dB

%%%%% Eldred, NASA SP-8072, p. 30 %%%%% step #5, refer to p. 13 Fig. 11
% valid for cold-air jets and standard chemical rockets
Me = engine_exit_velosity / (20.05*(engine_exit_temperature)^(0.5));
xt = de.*3.45.*((1+0.38.*Me).^2); % length core (Fig 11)

%%%%% Eldred, NASA SP-8072, p. 30 %%%%% step #7, refer to p. 13 Fig. 12
X_flow_axial_position = X_flow_axis_distance/xt; % Axial position
      (x/xt)
[A] = fill_relative_sound_power(X_flow_axial_position);

% figure
% plot(X_flow_axial_position,A)

%%%%% Eldred, NASA SP-8072, p. 30 %%%%% step #8, refer to p. 30 Eq. 7
% overall acoustic power for each point source based on the assumed
% distance from the engine exit plane
if data_button_section==6
```

```
    load('matlab.mat')
    Lws =
      interp1(Proudman_134_x,Proudman_134_y_fix,X_flow_axis_distance,'l
      inear','extrap'); % (dB)
%     Lws = A + Lw + 10.*log10(flow_slice_width./xt); % (dB)
    % pause
else
    Lws = A + Lw + 10.*log10(flow_slice_width./xt); % (dB)
end
% plot(X_flow_axis_distance,Lws)
% save('temp.mat','X_flow_axis_distance','Lws')
% pause


%%%%% Eldred, NASA SP-8072, p. 14 %%%%% step #9, refer to p. 14 Fig. 13
ae    = (1.4.*(8314./28.97).*engine_exit_temperature)^(1/2); % speed of
      sound in the flow at the nozzle exit (m/sec)
ao    = (1.4.*(8314./28.97).*ambient_temperature)^(1/2);     % speed of
      sound in the atmosphere (m/sec)
St_modified_temp =
      zeros(X_flow_axis_length,frequency_band_center_length);
for i=1:frequency_band_center_length
    St_modified_temp(:,i) =
      frequency_band_center(1,i).*X_flow_axis_distance;
end
St_modified = (St_modified_temp.*ae)./(engine_exit_velosity.*ao);  %
      calculation of the Modified strouhal number
% Changed this curve fit WOS 01-27-09
[AA]  = fill_relative_sound_power_spectrum(St_modified);

%%%%% Eldred, NASA SP-8072, p. 30 %%%%% step #9, refer to p. 30 Eq. 8
% acoustic power for each point source for each frequency band
% based on the assumed distance from the engine exit plane
Lwsb    = zeros(X_flow_axis_length,frequency_band_center_length);
for i=1:X_flow_axis_length % loop for assumed source locations
    for ii=1:frequency_band_center_length % loop for centered band
      frequencies
        Lwsb(i,ii) = AA(i,ii) + Lws(i,1) ...
            - 10*log10((engine_exit_velosity .* ao ) ./
      (X_flow_axis_distance(i,1).*ae)) ...
            + 10.*log10(frequency_band_width(ii,1));
            % shouldn't this be 10.*log10(frequency_band_width) instead
      on log10(frequency_band_width)
    end
end

%%%%% Eldred, NASA SP-8072, p. 20 %%%%% Fig. 17
% "r" is the length of the radius line from the assumed position of
% the frequency source to the point on the vehical.
% length of the radius line, determination of theta and beta in degrees
X_flow_axis_r = zeros(microphone_number,X_flow_axis_length);
X_flow_axis_theta = zeros(microphone_number,X_flow_axis_length);
X_flow_axis_beta = zeros(microphone_number,X_flow_axis_length);
for ii=1:microphone_number % loop for microphones
    for i=1:X_flow_axis_length   % loop for assumed source locations
```

```
        X_flow_axis_r(ii,i) = ((X_flow_axis(i,1)-
      microphone_data(ii,1)).^2 + (X_flow_axis(i,2)-
      microphone_data(ii,2)).^2 + (X_flow_axis(i,3)-
      microphone_data(ii,3)).^2).^(0.5);
        if i<X_flow_axis_after_ramp_point
            temp_angle =
      atand((abs(microphone_data(ii,2))./(microphone_data(ii,3)-
      X_flow_axis(i,3)));
            X_flow_axis_beta(ii,i) = 90 + temp_angle;
            X_flow_axis_theta(ii,i) = 180 - temp_angle;
        elseif i>=X_flow_axis_after_ramp_point
            if (X_flow_axis(i,2))<=(microphone_data(ii,2))
                temp_angle =
      atand((abs(microphone_data(ii,2))./(microphone_data(ii,3)-
      X_flow_axis(i,3)));
                X_flow_axis_beta(ii,i) = 90 + temp_angle;
            elseif (X_flow_axis(i,2))>(microphone_data(ii,2))
                %
      X_flow_axis_beta(microphone_number,X_flow_axis_length)
                X_flow_axis_beta(ii,i) =
      atand((microphone_data(ii,3))./(X_flow_axis(i,2)-
      microphone_data(ii,2)));
            else
                error('ERROR in "length of the radius line,
      determination of theta and beta" logic in function "approach2"
      line ~362')
            end
            % apparent flow angle (deg) in the Y-Z plane,the X-Y plane
      NOT USED
            X_flow_axis_theta(ii,i) = 180 - X_flow_axis_beta(ii,i) -
      apparent_engine_ramp_data(1,4);
        else
            error('ERROR in "length of the radius line, determination
      of theta and beta" logic in function "approach2" line ~367')
        end
    end
end
% X_flow_axis_r(microphone_number,X_flow_axis_length)

% if approach2_loop_counter==1
    if pressure_doubling_effect==1
        [P_Po_mag] =
      Diffraction_off_a_Cylinders(X_flow_axis_beta,X_flow_axis_length,
      ...

      frequency_band_center,frequency_band_center_length, ...
                                                  microphone_number,ao);

      %P_Po_mag(X_flow_axis_beta,frequency_band_center,microphone_numbe
      r);
    elseif pressure_doubling_effect==0
        % nothing
    else
        error('ERROR in "pressure doubling checkbox" logic in
      "approach2" fuction at line ~392')
    end
% end
```

```
% figure
% plot(X_flow_axis_r)
% ylabel('r')
% figure
% plot(X_flow_axis_distance,X_flow_axis_beta)
% xlabel('Distance from Exit Plane, x (m)')
% ylabel('Angle from Horizontal to Point Source, \beta (deg) ')
% text(40,90,'\leftarrow Flat Plate
      Deflector','HorizontalAlignment','left')
% figure
% plot(X_flow_axis_distance,X_flow_axis_theta)
% xlabel('Distance from Exit Plane, x (m)')
% ylabel('Angle from Exhaust Flow to Prediction, \theta (deg) ')
% text(40,120,'\leftarrow Flat Plate
      Deflector','HorizontalAlignment','left')
% error('STOP')

[DI] = fill_DI_meth_2(X_flow_axis_theta,St,microphone_number);
% DI(X_flow_axis_theta,St,microphone_number);

%%%%% Eldred, NASA SP-8072, p. 31 %%%%% step #10, refer to p. 31 Eq. 9
% SPL: Sound Pressure Level
% "s": each slice of exhaust
% "b": each band width of frequencies
% "p": each point on the vehicle (mic locations)
if reflection_effect==0 % no reflection
    SPLsbp    =
      zeros(X_flow_axis_length,frequency_band_center_length,microphone_
      number);
    SPLsbpint =
      zeros(X_flow_axis_length,frequency_band_center_length,microphone_
      number);
    SPLsbp_10 =
      zeros(X_flow_axis_length,frequency_band_center_length,microphone_
      number);
    SPLbp     = zeros(frequency_band_center_length,microphone_number);
    for iii=1:microphone_number % loop for microphones
        for i=1:X_flow_axis_length % loop for assumed source locations
            for ii=1:frequency_band_center_length% loop for centered
      band frequencies
                % SPLsbp: Sound pressure level in each frequency at
      each
                % point contributed by each slice
                if pressure_doubling_effect==0
                    SPLsbp(i,ii,iii) = Lwsb(i,ii) -
      10.*log10(X_flow_axis_r(iii,i).^2) - 11 + DI(i,ii,iii);
                elseif pressure_doubling_effect==1
                    SPLsbp(i,ii,iii) = Lwsb(i,ii) -
      10.*log10(X_flow_axis_r(iii,i).^2) - 11 + DI(i,ii,iii) +
      10.*log10(P_Po_mag(i,ii,iii));
                else
                    error('ERROR in "pressure doubling checkbox" logic
      in "approach2" fuction at line ~429')
                end
                SPLsbp_10(i,ii,iii) = SPLsbp(i,ii,iii)./10;
```

```
            SPLsbpint(i,ii,iii) = 10.^(SPLsbp_10(i,ii,iii)); %
  anti-log of SPLsbp after it was divied by ten
        end
    end

    %%%%% Eldred, NASA SP-8072, p. 31 %%%%% step #11, refer to p.
  31 Eq. 10
    % SPLbp: Sound pressure level in each frequency band for each
  mic
    % by logarithmic summation of contributions from each source.
    % The summation is with respect to each exhaust slice.  The new
    % matric will still be seperated by frequencies and points on
  the
    % vehical (mic locations).
    if defection_point_power==1 % Total Noise Level
        for ii=1:frequency_band_center_length
            SPLbp(ii,iii)    = 10.*log10(sum(SPLsbpint(:,ii,iii)));
        end
    elseif defection_point_power==2 % Noise Level Before Ramp
        for ii=1:frequency_band_center_length
            SPLbp(ii,iii)    =
  10.*log10(sum(SPLsbpint(1:(X_flow_axis_after_ramp_point-
  1),ii,iii)));
        end
    elseif defection_point_power==3 % Noise Level After Ramp
        for ii=1:frequency_band_center_length
            SPLbp(ii,iii)    =
  10.*log10(sum(SPLsbpint(X_flow_axis_after_ramp_point:end,ii,iii))
  );

        end
    else
        error('ERROR in "defection_point_power" logic in function
  "approach2" line ~482')
    end
end
SPLbp_10 = SPLbp./10;
SPLbpint = 10.^(SPLbp_10); % anti-log of SPLbp after it was divied
  by ten
% SPLbpint(frequency_band_center_length,microphone_number)

if frequency_minimum<100
    plot_minimum_frequency = frequency_minimum-10;
elseif frequency_minimum<1000
    plot_minimum_frequency = frequency_minimum-100;
else
    error('ERROR in "minimum_frequency" logic in function
  "approach2" line ~449')
end
if frequency_maximun>10000
    plot_maximun_frequency = frequency_maximun+10000;
elseif frequency_maximun>1000
    plot_maximun_frequency = frequency_maximun+1000;
else
    error('ERROR in "frequency_maximun" logic in function
  "approach2" line ~456')
```

```
    end
    % i = loop for assumed source locations
    % ii = loop for centered band frequencies
    % iii = loop for microphones

    %%%%% Eldred, NASA SP-8072, p. 31 %%%%% step #12, refer to p. 29
      Eq. 6
    % Sound pressure level frequency spectrum %
    SPLbp_sd = zeros(frequency_band_center_length,microphone_number);
    SPLbpint_sd =
      zeros(frequency_band_center_length,microphone_number);
    if data_button_section==4 && shuttle_and_booster==1 % Both Shuttle
      and Booster
        SPLbpint_shuttle_and_booster =
      zeros(frequency_band_center_length,microphone_number,2);
    end
    for iii=1:microphone_number   % loop for microphones
        for ii=1:frequency_band_center_length % loop for centered band
      frequencies
            % spectral density
            SPLbp_sd(ii,iii) = SPLbp(ii,iii) -
      10*log10(frequency_band_width(ii,1));
            SPLbpint_sd(ii,iii) = 10.^((SPLbp_sd(ii,iii))./10);
            if data_button_section==4 % Both Shuttle and Booster
                if shuttle_and_booster==1

      SPLbpint_shuttle_and_booster(ii,iii,shuttle_and_booster) =
      SPLbpint(ii,iii);
                elseif shuttle_and_booster==2

      SPLbpint_shuttle_and_booster(ii,iii,shuttle_and_booster) =
      SPLbpint(ii,iii);
                    SPLbpint(ii,iii) =
      sum(SPLbpint_shuttle_and_booster(ii,iii,:));
                else
                    error('ERROR in logic in approach2 line ~484')
                end
            end
        end
    end

%     if approach2_loop_counter==1
        SPLp = zeros(microphone_number,1);
        SPLp_launch_data = zeros(microphone_number,1);
        for iii=1:microphone_number    % loop for number of micro
            if plot_OASPL==1 && data_button_section~=4 &&
      microphone_launch_data_exist(2,iii)~=1
                SPLp(iii,1)  = 10.*log10(sum(SPLbpint(:,iii))); %
      summation with respect to the centered band frequencies
                fprintf('\nOverall Sound Pressure Level (OASPL) for
      mic. #%g (distance from nozzle %gm)
      \n',iii,(microphone_data(iii,3)-microphone_offset))
                fprintf('OASPL: (dB) %g \n',SPLp(iii,1))% Overall dB
      level
            end
```

```
        if microphone_launch_data_exist(2,iii)==1
            if data_button_section==1 % Saturn V
                temp_microphone_launch_data =
microphone_launch_data(:,iii) +
10.*log10(microphone_launch_data_frec_width);
                SPLp_launch_data(iii,1) =
10.*log10(sum(10.^((temp_microphone_launch_data)./10)));
            elseif data_button_section==6 % Jet
                load Norum_data
                if iii==1
                    data_launch = data_launch_45;
                elseif iii==2
                    data_launch = data_launch_90;
                elseif iii==3
                    data_launch = data_launch_135;
                else
                    error('stop')
                end
                SPLp_launch_data(iii,1) =
10.*log10(sum(10.^((data_launch(1:10:end))./10)));
            else
                SPLp_launch_data(iii,1) =
10.*log10(sum(10.^((microphone_launch_data(:,iii))./10)));
            end
%               [temp_microphone_data_row, temp_microphone_data_min]
=
find(frequency_band_center<=(min(microphone_launch_data_frec)),1,
'last');
%               [temp_microphone_data_row, temp_microphone_data_max]
=
find(frequency_band_center>=(max(microphone_launch_data_frec)),1,
'first');
%               SPLp(iii,1)  =
10.*log10(sum(SPLbpint(temp_microphone_data_min:temp_microphone_d
ata_max,iii))); % summation with respect to the centered band
frequencies
            SPLp(iii,1)  = 10.*log10(sum(SPLbpint(:,iii)));

            if plot_OASPL==1 && data_button_section~=4
                fprintf('\nOverall Sound Pressure Level (OASPL) for
mic. #%g (distance from nozzle %gm)
\n',iii,(microphone_data(iii,3)-microphone_offset))
                fprintf('OASPL: (dB) %g \n',SPLp(iii,1))% Overall
dB level
                fprintf('Launch Data OASPL: (dB) %g
\n',SPLp_launch_data(iii,1))
            elseif plot_OASPL==1 && data_button_section==4 % Both
Shuttle and Booster
                if shuttle_and_booster==1
                    % nothing
                elseif shuttle_and_booster==2
                    if plot_OASPL==1
                        fprintf('\nOverall Sound Pressure Level
(OASPL) for mic. #%g (distance from nozzle %gm)
\n',iii,(microphone_data(iii,3)-microphone_offset))
                        fprintf('OASPL: (dB) %g \n',SPLp(iii,1))%
Overall dB level
```

```
                    fprintf('Launch Data OASPL: (dB) %g
\n',SPLp_launch_data(iii,1))
                end
            else
                error('ERROR in logic in approach2 line ~540')
            end
        else
            error('ERROR in logic in approach2 line ~540')
        end
    end

    if plot_sum_engines==1
        if microphone_launch_data_exist(2,iii)==0
            h   = 'Sum of all the nozzles; Microphone:';
            hh  = strcat(h,48+iii);
            hhh = str2mat(hh);
            hhhhh = figure('Name',hhh);
            axes1 = axes('Parent',hhhhh,...
                'Position',[0.10 0.10 0.80 0.75],...
                'XLim',[plot_minimum_frequency
plot_maximun_frequency],...
                'YLim',[50 160],...
                'XScale','log',...
                'GridLineStyle',':',...
                'Box','on',...

'xtick',[10:10:90,100:100:900,1000:1000:9000,10000:10000:100000],
...

'XTickLabel',{'10';'';'';'';'';'';'';'';'';'100';'';'';'';'';'';'
';'';'';'1000';'';'';'';'';'';'';'';'';'';'10000';'';'';'';'';'';'';
'';'';'100000'});
            grid(axes1,'on')

line(frequency_band_center,SPLbp(:,iii),'Color','b','LineStyle','
-','Parent',axes1,'LineWidth',0.5);

line(frequency_band_center,SPLbp_sd(:,iii),'Color','r','LineStyle
','-','Parent',axes1,'LineWidth',0.5);
            legend(gca,'String',{strcat('Bandwidth:
1/',num2str(octave_size),' octave'),'Spectral
density'},'Units','normalized','Location',[0.1 0.88 0.8
0.05],'Orientation','Horizontal');
            xlabel(axes1,'Frequency,
Hz','Color',[0,0,0],'FontSize',14)
            ylabel(axes1,'Sound Pressure Level,
dB','Color',[0,0,0],'FontSize',14)
            title(axes1,{'Frequency
Spectrum'},'FontSize',14,'Units','normalized','Position',[0.5
1.12])
        elseif microphone_launch_data_exist(2,iii)==1
            h   = 'Sum of all the nozzles; Microphone:';
            hh  = strcat(h,48+iii);
            hhh = str2mat(hh);
            hhhhh = figure('Name',hhh);
            axes1 = axes('Parent',hhhhh,...
                'Position',[0.10 0.10 0.80 0.75],...
```

```
                    'XLim',[plot_minimum_frequency
plot_maximun_frequency],...
                    'XScale','log',...
                    'GridLineStyle',':',...
                    'Box','on',...

'xtick',[10:10:90,100:100:900,1000:1000:9000,10000:10000:100000],
...

'XTickLabel',{'10';'';'';'';'';'';'';'';'';'100';'';'';'';'';'';'
';'';'';'1000';'';'';'';'';'';'';'';'';'10000';'';'';'';'';'';'';
'';'';'100000'});
                % grid(axes1,'on')
                if data_button_section==1

line(frequency_band_center,SPLbp(:,iii),'Color','r','LineStyle','
-','Parent',axes1,'LineWidth',0.5);

line(frequency_band_center,SPLbp_sd(:,iii),'Color','b','LineStyle
','--','Parent',axes1,'LineWidth',0.5);%'Marker','^'
                    ylim([80 160]);

line(microphone_launch_data_frec,microphone_launch_data(:,iii),'C
olor','k','LineStyle','none','Parent',axes1,'LineWidth',0.5,'Mark
er','o');
                    legend(gca,'String',{'SPL Prediction','SSDL
Prediction','SSDL Launch
Data'},'Units','normalized','Location',[0.1 0.88 0.8
0.05],'Orientation','Horizontal');
                    % legend(gca,'String',{'SSDL Prediction','SSDL
Launch Data'},'Units','normalized','Location',[0.1 0.88 0.8
0.05],'Orientation','Horizontal');
                elseif data_button_section==2 ||
data_button_section==3 || data_button_section==4 ||
data_button_section==5

line(frequency_band_center,SPLbp(:,iii),'Color','r','LineStyle','
-','Parent',axes1,'LineWidth',0.5);

line(frequency_band_center,SPLbp_sd(:,iii),'Color','b','LineStyle
','--','Parent',axes1,'LineWidth',0.5);%'Marker','^'
                    ylim([80 160]);

line(microphone_launch_data_frec,microphone_launch_data(:,iii),'C
olor','k','LineStyle','none','Parent',axes1,'LineWidth',0.5,'Mark
er','o');
                    legend(gca,'String',{'SPL Prediction','SSDL
Prediction','SPL Launch
Data'},'Units','normalized','Location',[0.1 0.88 0.8
0.05],'Orientation','Horizontal');
                    % legend(gca,'String',{'SPL Prediction','SPL
Launch Data'},'Units','normalized','Location',[0.1 0.88 0.8
0.05],'Orientation','Horizontal');
                elseif data_button_section==6

line(frequency_band_center,SPLbp(:,iii),'Color','r','LineStyle','
-','Parent',axes1,'LineWidth',0.5);
```

```
line(frequency_band_center,SPLbp_sd(:,iii),'Color','b','LineStyle
','--','Parent',axes1,'LineWidth',0.5);%'Marker','^'
                    load Norum_data
                    if iii==1
                        data_frec = data_frec_45;
                        data_launch = data_launch_45;
                    elseif iii==2
                        data_frec = data_frec_90;
                        data_launch = data_launch_90;
                    elseif iii==3
                        data_frec = data_frec_135;
                        data_launch = data_launch_135;
                    else
                        error('stop')
                    end

line(data_frec(1:10:end),data_launch(1:10:end),'Color','k','LineS
tyle','none','Parent',axes1,'LineWidth',0.5,'Marker','o');
                    ylim([40 120]);
                    legend(gca,'String',{'SPL Prediction','SSDL
Prediction','SPL Wind Tunnel
Data'},'Units','normalized','Location',[0.1 0.88 0.8
0.05],'Orientation','Horizontal');
                    % legend(gca,'String',{'SPL Prediction','SPL
Wind Tunnel Data'},'Units','normalized','Location',[0.1 0.88 0.8
0.05],'Orientation','Horizontal');
                else
                    error('stop')
                end
                % legend(gca,'String',{'Spectral density
prediction','Launch Data'},'Units','normalized','Location',[0.1
0.88 0.8 0.05],'Orientation','Horizontal');
                xlabel(axes1,'Frequency
(Hz)','Color',[0,0,0],'FontSize',12)
                ylabel(axes1,'Sound Pressure Level
(dB)','Color',[0,0,0],'FontSize',12)
                % title(axes1,{'Frequency
Spectrum'},'FontSize',12,'Units','normalized','Position',[0.5
1.12])
            else
                 error('ERROR in "microphone launch data exist"
logic in approach2')
            end
            if data_button_section==4 % Both Shuttle and Booster
                if shuttle_and_booster==1
                    close
                elseif shuttle_and_booster==2
                    % nothing
                else
                    error('ERROR in "microphone launch data exist"
logic in approach2')
                end
            end
        end
```

```
      if (save_data==1 && data_button_section==4 &&
shuttle_and_booster==2) || (save_data==1 &&
data_button_section~=4)
          % if exist('save_file_directory','var')==0
          %     [file_name, save_file_directory, filter_index] =
uiputfile('*.mat', 'Save sound pressure level frequency
spectrum');
          % end
          % if filter_index == 1
          if defection_point_power==1 % Total Noise Level
              data_position = '_T';
          elseif defection_point_power==2 % Noise Level Before
Ramp
              data_position = '_B';
          elseif defection_point_power==3 % Noise Level After
Ramp
              data_position = '_A';
          else
              error('ERROR in "defection point power" logic in
approach2')
          end
          if data_button_section==1 % Saturn V
              if get(findobj('Tag','saturn_mic_1'),'value')==1
                  data_set_name = '_saturn_mic_1';
                  set(findobj('Tag','saturn_mic_1'),'value',0);
              elseif
get(findobj('Tag','saturn_mic_2'),'value')==1
                  data_set_name = '_saturn_mic_2';
                  set(findobj('Tag','saturn_mic_2'),'value',0);
              elseif
get(findobj('Tag','saturn_mic_3'),'value')==1
                  data_set_name = '_saturn_mic_3';
                  set(findobj('Tag','saturn_mic_3'),'value',0);
              elseif
get(findobj('Tag','saturn_mic_4'),'value')==1
                  data_set_name = '_saturn_mic_4';
                  set(findobj('Tag','saturn_mic_4'),'value',0);
              elseif
get(findobj('Tag','saturn_mic_5'),'value')==1
                  data_set_name = '_saturn_mic_5';
                  set(findobj('Tag','saturn_mic_5'),'value',0);
              elseif
get(findobj('Tag','saturn_mic_6'),'value')==1
                  data_set_name = '_saturn_mic_6';
                  set(findobj('Tag','saturn_mic_6'),'value',0);
              elseif
get(findobj('Tag','saturn_mic_9'),'value')==1
                  data_set_name = '_saturn_mic_9';
                  set(findobj('Tag','saturn_mic_9'),'value',0);
              else
                  error('ERROR line in ~590 in summing plot
section of microphone logic in approach2')
              end
          elseif data_button_section==2 % Shuttle
              if get(findobj('Tag','shuttle_mic_1'),'value')==1
                  data_set_name = '_shuttle_mic_1';
                  set(findobj('Tag','shuttle_mic_1'),'value',0);
```

```
                elseif
get(findobj('Tag','shuttle_mic_2'),'value')==1
                    data_set_name = '_shuttle_mic_2';
                    set(findobj('Tag','shuttle_mic_2'),'value',0);
                elseif
get(findobj('Tag','shuttle_mic_3'),'value')==1
                    data_set_name = '_shuttle_mic_3';
                    set(findobj('Tag','shuttle_mic_3'),'value',0);
                elseif
get(findobj('Tag','shuttle_mic_4'),'value')==1
                    data_set_name = '_shuttle_mic_4';
                    set(findobj('Tag','shuttle_mic_4'),'value',0);
                elseif
get(findobj('Tag','shuttle_mic_5'),'value')==1
                    data_set_name = '_shuttle_mic_5';
                    set(findobj('Tag','shuttle_mic_5'),'value',0);
                elseif
get(findobj('Tag','shuttle_mic_9'),'value')==1
                    data_set_name = '_shuttle_mic_9';
                    set(findobj('Tag','shuttle_mic_9'),'value',0);
                else
                    error('ERROR line in ~612 in summing plot
section of microphone logic in approach2')
                end
            elseif data_button_section==3 % Booster
                if get(findobj('Tag','shuttle_mic_1'),'value')==1
                    data_set_name = '_booster_mic_1';
                    set(findobj('Tag','shuttle_mic_1'),'value',0);
                elseif
get(findobj('Tag','shuttle_mic_2'),'value')==1
                    data_set_name = '_booster_mic_2';
                    set(findobj('Tag','shuttle_mic_2'),'value',0);
                elseif
get(findobj('Tag','shuttle_mic_3'),'value')==1
                    data_set_name = '_booster_mic_3';
                    set(findobj('Tag','shuttle_mic_3'),'value',0);
                elseif
get(findobj('Tag','shuttle_mic_4'),'value')==1
                    data_set_name = '_booster_mic_4';
                    set(findobj('Tag','shuttle_mic_4'),'value',0);
                elseif
get(findobj('Tag','shuttle_mic_5'),'value')==1
                    data_set_name = '_booster_mic_5';
                    set(findobj('Tag','shuttle_mic_5'),'value',0);
                elseif
get(findobj('Tag','shuttle_mic_9'),'value')==1
                    data_set_name = '_booster_mic_9';
                    set(findobj('Tag','shuttle_mic_9'),'value',0);
                else
                    error('ERROR line in ~633 in summing plot
section of microphone logic in approach2')
                end
            elseif data_button_section==4 % Both Shuttle and
Booster
                if get(findobj('Tag','shuttle_mic_1'),'value')==1
                    data_set_name = '_shuttle_and_booster_mic_1';
                    set(findobj('Tag','shuttle_mic_1'),'value',0);
```

```
                elseif
get(findobj('Tag','shuttle_mic_2'),'value')==1
                    data_set_name = '_shuttle_and_booster_mic_2';
                    set(findobj('Tag','shuttle_mic_2'),'value',0);
                elseif
get(findobj('Tag','shuttle_mic_3'),'value')==1
                    data_set_name = '_shuttle_and_booster_mic_3';
                    set(findobj('Tag','shuttle_mic_3'),'value',0);
                elseif
get(findobj('Tag','shuttle_mic_4'),'value')==1
                    data_set_name = '_shuttle_and_booster_mic_4';
                    set(findobj('Tag','shuttle_mic_4'),'value',0);
                elseif
get(findobj('Tag','shuttle_mic_5'),'value')==1
                    data_set_name = '_shuttle_and_booster_mic_5';
                    set(findobj('Tag','shuttle_mic_5'),'value',0);
                elseif
get(findobj('Tag','shuttle_mic_9'),'value')==1
                    data_set_name = '_shuttle_and_booster_mic_9';
                    set(findobj('Tag','shuttle_mic_9'),'value',0);
                else
                    error('ERROR line in ~711 in summing plot
section of microphone logic in approach2')
                end
            elseif data_button_section==5 % Ares I
                if get(findobj('Tag','ares_mic_1'),'value')==1
                    data_set_name = '_ares_mic_1';
                    set(findobj('Tag','ares_mic_1'),'value',0);
                elseif get(findobj('Tag','ares_mic_9'),'value')==1
                    data_set_name = '_ares_mic_9';
                    set(findobj('Tag','ares_mic_9'),'value',0);
                else
                    error('ERROR line in ~666 in summing plot
section of microphone logic in approach2')
                end
            elseif data_button_section==9 % User Defined
                data_set_name =
str2mat(strcat('user_defined_mic',num2str(iii)));
            else
                error('ERROR line in ~670 in summing plot section
of microphone logic in approach2')
            end
            tag_name =
str2mat(strcat(data_set_name,data_position));
            tag_name_octave =
str2mat(strcat('SPLtotcmint',tag_name));
            tag_name_spectral =
str2mat(strcat('SPLtotcmintsd',tag_name));
            % current_working_directory = cd;
            % cd(save_file_directory);
            assignin('base','Freqc_number',Freqc_number);
            assignin('base',tag_name_octave,SPLtotcmint);
            assignin('base',tag_name_spectral,SPLtotcmintsd);
            if iii==microphone_number
                if exist('SPLbptot_data.mat','file')==0
                    evalin('base','save SPLbptot_data');
                else
```

```
                        evalin('base','save SPLbptot_data -append');
                    end
                end
                % cd(current_working_directory);
                fprintf('%s Done \n',data_set_name)
            end
        end
        if data_button_section==4 % Both Shuttle and Booster
            if shuttle_and_booster==1
                shuttle_and_booster=2;
            elseif shuttle_and_booster==2
                shuttle_and_booster=3;
            else
                error('ERROR in "microphone launch data exist" logic in
    approach2')
            end
        end
%     end
end

% if reflection_effect==1 && refldi_number==0 % reflexion but no
    directivity in reflexion
%     [deltaLfc] =
    ground_reflection(file_number,valamp_number,valphase_number,r,X,X
    peq_loop,Xint,Freq,Freqc,ao,teta,AA,lfen_number);
%     % Sound pressure level in each frequency band at each point
    contributed
%     % by each each slice
%     SPLsbp =
    zeros(X_flow_axis_length,frequency_band_center_length,microphone_
    number);
%     SPLsbpint =
    zeros(X_flow_axis_length,frequency_band_center_length,microphone_
    number);
%     SPLsbp_10 =
    zeros(X_flow_axis_length,frequency_band_center_length,microphone_
    number);
%     for iii=1:microphone_number   % loop for number of micro
%         for i=1:X_flow_axis_length      % position of sources
%             for ii=1:frequency_band_center_length % Centered
    frequency
%                 SPLsbp(i,ii,iii) = Lwsb(i,ii)-10*log10(r(iii,i)^2)-
    11+DI(i,ii,iii)+deltaLfc(i,ii,iii);
%                 SPLsbp_10(i,ii,iii) = SPLsbp(i,ii,iii)/10;
%                 SPLsbpint(i,ii,iii) = 10^(SPLsbp_10(i,ii,iii));
%             end
%         end
%
%         % Sound pressure level in each frequency band at each point
    by
%         % logarithmic summation of conditions from each of the slices
%
%         for ii=1:frequency_band_center_length
%             SPLbp(1,ii,iii)    = 10*log10(sum(SPLsbpint(:,ii,iii)));
%             SPLbp_10(1,ii,iii) = SPLbp(1,ii,iii)/10;
%         end
%     end
```

```
%
%     % Sound pressure level frequency spectrum
%     SPLbptot     = zeros(frequency_band_center_length,1);
%     SPLbptotsd   = zeros(frequency_band_center_length,1);
%     SPLbptotcm   =
      zeros(frequency_band_center_length,microphone_number);
%     SPLbptotsdcm =
      zeros(frequency_band_center_length,microphone_number);
%     if lfen_number==1
%         SPLsbptotcm   =
      zeros(frequency_band_center_length,microphone_number,1);
%         SPLsbptotsdcm =
      zeros(frequency_band_center_length,microphone_number,1);
%     end
%     for iii=1:microphone_number   % loop for number of micro
%         for i=1:X_flow_axis_length      % position of sources
%             for ii=1:frequency_band_center_length % Centered
      frequency
%                 SPLbptot(ii,1)     = SPLbp(1,ii,iii);
%                 SPLbptotcm(ii,iii) = SPLbp(1,ii,iii);
%                 % spectral density
%                 SPLbptotsd(ii,1)     = SPLbptot(ii,1)-
      10*log10(deltaf_number(ii,1));
%                 SPLbptotsdcm(ii,iii) = SPLbptotcm(ii,iii)-
      10*log10(deltaf_number(ii,1));
%                 if lfen_number>1
%                     SPLsbptotcm(ii,iii,lfen_number)   =
      SPLbptotcm(ii,iii);
%                     SPLsbptotsdcm(ii,iii,lfen_number) =
      SPLbptotsdcm(ii,iii);
%                 end
%             end
%         end
%         h    = 'Microphone:';
%         hh   = ',Exhaust number:';
%         hhh  = strcat(h,48+iii,hh,48+lfen_number);
%         hhhh = str2mat(hhh);
%         figure('Name',hhhh);
%
      semilogx(Freqc_number,SPLbptot,'b',Freqc_number,SPLbptotsd,'r');
%         legend(strcat('Bandwidth: 1/',num2str(octave_size),'
      octave'),'Spectral density');
%         xlabel('Frequency in Hertz','FontSize',16)
%         ylabel('Sound pressure level in dB','FontSize',16)
%         title('\it{Frequency Spectrum}','FontSize',16)
%         xlim([10 20000])
%         ylim([50 160])
%         grid on
%     end
%
%     % Overall sound pressure level at any point p over the entire
      spectrum
%     SPLoap    = zeros(1,1,microphone_number);
%     SPLoapint = zeros(microphone_number,1);
%     SPLbpint  =
      zeros(1,frequency_band_center_length,microphone_number);
%     for iii=1:microphone_number % loop for number of micro
```

```
%         for ii=1:frequency_band_center_length   % Centered frequency
%             SPLbpint(1,ii,iii) = 10.^(SPLbp_10(1,ii,iii));
%         end
%         SPLoap(1,1,iii)  = 10.*log10(sum(SPLbpint(1,:,iii)));
%         SPLoapint(iii,1) = SPLoap(1,1,iii); % solution of the second
      method
%     end
%
%     if lfen_number==1
%         SPLsbptotcm_10 =
      zeros(frequency_band_center_length,microphone_number,1);
%         SPLsbptotcmint =
      zeros(frequency_band_center_length,microphone_number,1);
%         SPLtotoap      = zeros(1,microphone_number);
%         SPLtotcmint    = zeros(frequency_band_center_length,1,1);
%         SPLtotcmint1   =
      zeros(frequency_band_center_length,microphone_number,1);
%         SPLtotcmint2   =
      zeros(frequency_band_center_length,microphone_number,1);
%         SPLtotcmint_10 =
      zeros(frequency_band_center_length,microphone_number,1);
%         SPLtotcmintsd  = zeros(frequency_band_center_length,1,1);
%         if lfen_number==1
%             for iii=1:microphone_number   % loop for number of micro
%                 for ii=1:frequency_band_center_length     % Centered
      frequency
%                     for i=1:1 % exhaust nozzle number
%                         SPLsbptotcm_10(ii,iii,i) =
      SPLsbptotcm(ii,iii,i)/10;
%                         SPLsbptotcmint(ii,iii,i) =
      10.^(SPLsbptotcm_10(ii,iii,i));
%                     end
%                 end
%                 for ii=1:frequency_band_center_length % Centered
      frequency
%                     SPLtotcmint1(ii,iii,1)  =
      10.*log10(sum(SPLsbptotcmint(ii,iii,:)));
%                     SPLtotcmint(ii,1,1)     = SPLtotcmint1(ii,iii,1);
%                     SPLtotcmintsd(ii,1,1)   = SPLtotcmint(ii,1,1)-
      10.*log10(deltaf_number(ii,1));
%                     SPLtotcmint_10(ii,iii,1) =
      SPLtotcmint1(ii,iii,1)./10;
%                     SPLtotcmint2(ii,iii,1)   =
      10.^(SPLtotcmint_10(ii,iii,1));
%                 end
%                 h   = 'Sum of all the nozzles; Microphone:';
%                 hh  = strcat(h,48+iii);
%                 hhh = str2mat(hh);
%                 figure('Name',hhh);
%
      semilogx(Freqc_number,SPLtotcmint,'b',Freqc_number,SPLtotcmintsd,
      'r');
%                 legend(strcat('Bandwidth: 1/',num2str(octave_size),'
      octave'),'Spectral density');
%                 xlabel('Frequency in Hertz','FontSize',16)
%                 ylabel('Sound pressure level in dB','FontSize',16)
%                 title('\it{Frequency Spectrum}','FontSize',16)
```

```
%                 xlim([10 20000])
%                 ylim([0 160])
%                 grid on
%
%                 % sum of all the frequency
%                 SPLtotoap(1,iii) =
    10*log10(sum(SPLtotcmint2(:,iii,1)));
%             end
%         end
%     end
% end

% if reflection_effect==1 && refldi_number==1 % reflexion and
    directivity of the reflexion
%     % Sound pressure level in each frequency band at each point
    contributed
%     % by each slice
%     SPLsbp    =
    zeros(X_flow_axis_length,frequency_band_center_length,microphone_
    number);
%     SPLsbp_10 =
    zeros(X_flow_axis_length,frequency_band_center_length,microphone_
    number);
%     SPLsbpint =
    zeros(X_flow_axis_length,frequency_band_center_length,microphone_
    number);
%     SPLbp     =
    zeros(1,frequency_band_center_length,microphone_number);
%     SPLbp_10  =
    zeros(1,frequency_band_center_length,microphone_number);
%     SPLbpint  =
    zeros(1,frequency_band_center_length,microphone_number);
%     for iii=1:microphone_number   % loop for number of micro
%         for i=1:X_flow_axis_length      % position of sources
%             for ii=1:frequency_band_center_length % Centered
    frequency
%                 SPLsbp(i,ii,iii)    = Lwsb(i,ii)-
    10.*log10(r(iii,i).^2)-11+DI(i,ii,iii);
%                 SPLsbp_10(i,ii,iii) = SPLsbp(i,ii,iii)./10;
%                 SPLsbpint(i,ii,iii) = 10.^(SPLsbp_10(i,ii,iii));
%             end
%         end
%
%         % Sound pressure level in each frequency band at each point
    by
%         % logarithmic summation of conditions from each of the slices
%         for ii=1:frequency_band_center_length
%             SPLbp(1,ii,iii)    = 10.*log10(sum(SPLsbpint(:,ii,iii)));
%             SPLbp_10(1,ii,iii) = SPLbp(1,ii,iii)./10;
%             SPLbpint(1,ii,iii) = 10.^(SPLbp_10(1,ii,iii));
%         end
%     end
%     loop_number = 5;
%     rprim=0;
%     % mapping of mirror sources and loading of directivity index of
    the
%     % new sources
```

```
%     [rprim] =
      ground_reflection2_2(X,microphone_number,1,loop_number,octave_siz
      e,automatic_method,lfen_number,microphone_data,alpha_loop,rprim,L
      wsb,AA,deltaf_number,SPLsbpint,St);
%     loop_number=6;
%     [rprim] =
      ground_reflection2_2(X,microphone_number,1,loop_number,octave_siz
      e,automatic_method,lfen_number,microphone_data,alpha_loop,rprim,L
      wsb,AA,deltaf_number,SPLsbpint,St); %#ok<NASGU>
% end

% approach2_loop_counter = approach2_loop_counter + 1;

end

%.....................................................................
      ....

function [A] = fill_relative_sound_power(Axialp)
% changed curve fit WOS 01-27-09
%%%%%%% Fill in Relative sound power per unit core length for the
      second method
%%%%% Eldred, NASA SP-8072, p. 30 %%%%%% step #7
%%%%% Eldred, NASA SP-8072, p. 13 %%%%%% Fig. #12
% load lfen
% load Axialp
Axialp_length=length(Axialp);
A=zeros(Axialp_length,1);

%%% Approximation of the curve: Original Acoustic Loads 2.0
% for i=1:Axialp_length
%     n=0;
%     test=0;
%     a=1;
%     while test==0
%         y=(2^n)*0.1;
%         z=Axialp(i,1)-y;
%         if z<0.01 && z>0
%             test=1;
%             x=1.6*n;
%         else
%             if z>0
%                 n=n+a;
%             else
%                 if n==0
%                     test=1;
%                     x=0;
%                 else
%                     n=n-a+(a/10);
%                     a=a/10;
%                 end
%             end
%         end
%     end
%
%     if Axialp(i,1)<= 1.5
```

```
%         A(i,1)=(11.87/4.7)*x-18.5;
%     elseif Axialp(i,1)>1.5 && Axialp(i,1)<2.7
%         A(i,1)=(-1.87/1.4)*x+5.285;
%     else
%         A(i,1)=(-15/1.6)*x+66.25;
%     end
% end

% save A A

% a='Rel. SP,Nozzle n';
% b=48+lfen;
% cint=strcat(a,b);
% RSP=str2mat(cint);
% e={'x/xt','Normalized relative sound power per unit core length'};
% xlswrite('data2',e,RSP,'A1')
% xlswrite('data2',Axialp,RSP,'A2')
% xlswrite('data2',A,RSP,'B2')
% Excel = actxserver('Excel.application');
% Excel.Visible = 1;
% winopen data2.xls

%%% Approximation of Figure 12 curve: New Acoustic Loads 2.13 (could be
      improved)
%%%%% Eldred, NASA SP-8072, p. 13 %%%%%%
% Figure data
% x = [0.1    0.15   0.2  0.25   0.3   0.4  0.5  0.6  0.7  0.8  0.9
      1.0  1.1  1.2  1.4  1.6  1.8  2.0  2.2  2.4  2.6  2.8  3.0   3.2
      3.4   3.6   3.8   4.0   4.2   4.4   4.6];
% y = [-18.8 -16.4 -14.7 -13.4 -12.2 -10.5 -9.2 -8.1 -7.1 -6.3 -5.6 -
      5.0 -4.4 -4.0 -3.2 -3.0 -3.2 -3.6 -4.2 -5.2 -6.6 -8.2 -9.9 -11.3
      -12.6 -13.9 -15.2 -16.3 -17.4 -18.6 -19.6];

for i=1:Axialp_length
    if Axialp(i,1)<=1.1
        A(i,1) = 13.8.*log10(Axialp(i,1)) - 5;
    elseif Axialp(i,1)>1.1 && Axialp(i,1)<=2.6
        A(i,1) = -75.*(log10(Axialp(i,1)/1.6)).^3 -
      65.*(log10(Axialp(i,1)/1.6)).^2 - 3;
    elseif Axialp(i,1)>2.6
        A(i,1) = -53.*log10(Axialp(i,1)) + 15.5;
    else
        error('"fill_relative_sound_power" funtion logic at line
      ~1126')
    end
end

end

%.....................................................................
      ....

function [AA] = fill_relative_sound_power_spectrum(Stmod)
% changed curve fit WOS 01-28-09
```

```
%%%%%%%% fill in relative sound power spectrum level for the second
      method
%%%%% Eldred, NASA SP-8072, p. 30 %%%%%% step #9
%%%%% Eldred, NASA SP-8072, p. 14 %%%%%% Fig. #13
% load Stmod
% load Freqc
% load Xflowc
% load lfen
% fb=Freqc_number; %%% center frequency
Stmod_size=size(Stmod);
AA = zeros(Stmod_size(1,1),Stmod_size(1,2));

%%% Approximation of the curve: Original Acoustic Loads 2.0
% for i=1:Stmod_size(1,1) % rows of Stmod
%     for ii=1:Stmod_size(1,2) % comlumns of Stmod
%         n=0;
%         test=0;
%         a=1;
%         while test==0
%             y=(2^n)*0.05;
%             z=Stmod(i,ii)-y;
%             if z<0.01 && z>0
%                 test=1;
%                 x=1.1*n;
%             else
%                 if z>0
%                     n=n+a;
%                 else
%                     if n==0
%                         test=1;
%                         x=0;
%                     else
%                         n=n-a+(a/10);
%                         a=a/10;
%                     end
%                 end
%             end
%         end
%         if Stmod(i,ii)<=1
%             AA(i,ii) = (12.8/4.7)*x - 20;
%         elseif Stmod(i,ii)>1 && Stmod(i,ii)<3
%             AA(i,ii) = (-2.8/1.9)*x - 0.27;
%         else
%             AA(i,ii) = (-30/4.8)*x + 30.62;
%         end
%     end
% end

% save AA AA

% a='Mod. axial strouhal,Nozzle n';
% b=48+lfen;
% cint=strcat(a,b);
% MAS=str2mat(cint);
% ee={'frequency'};
% eee={'Sound power spectrum'};
```

```
% eeee={'Position of the sources'};
% xlswrite('data2',ee,MAS,'B1')
% xlswrite('data2',fb,MAS,'C1')
% xlswrite('data2',eeee,MAS,'A2')
% xlswrite('data2',Xflowc,MAS,'A3')
% xlswrite('data2',Stmod,MAS,'C3')
% a='SP spect. lev.,Nozzle n';
% cint=strcat(a,b);
% SPSL=str2mat(cint);
% xlswrite('data2',ee,SPSL,'B1')
% xlswrite('data2',fb,SPSL,'C1')
% xlswrite('data2',eeee,SPSL,'A2')
% xlswrite('data2',Xflowc,SPSL,'A3')
% xlswrite('data2',AA,SPSL,'C3')
% Excel = actxserver('Excel.application');
% Excel.Visible = 1;
% winopen data2.xls

%%% Approximation of Figure 13 curve: New Acoustic Loads 2.13  (could
      be improved)
%%%%% Eldred, NASA SP-8072, p. 14 %%%%%%
% Figure data
% x = [ 0.05 0.075   0.1  0.15   0.2   0.3   0.4  0.5  0.6  0.7  0.8
      0.9  1.0  1.2   1.4   1.6  1.8  2.0  2.5  3.0   4.0   5.0   7.0
      10.0  15.0  20.0  30.0  50.0  60.0];
% y = [-20.0 -18.4 -16.9 -15.3 -14.1 -12.3 -11.0 -9.9 -9.1 -8.4 -8.0 -
      7.6 -7.4 -7.1 -6.85 -6.75 -6.9 -7.2 -7.7 -8.8 -11.3 -13.6 -16.9 -
      20.6 -24.6 -27.3 -31.6 -36.7 -38.5];

for i=1:Stmod_size(1,1) % rows of Stmod
    for ii=1:Stmod_size(1,2) % comlumns of Stmod
        if Stmod(i,ii)<=0.8
            AA(i,ii) = 10.*log10(Stmod(i,ii)) - 7;
        elseif Stmod(i,ii)>0.8 && Stmod(i,ii)<=3.5
            AA(i,ii) = -12.*(log10(Stmod(i,ii)/1.5)).^3 -
      20.*(log10(Stmod(i,ii)/1.5)).^2 - 6.8;
        elseif Stmod(i,ii)>3.5
            AA(i,ii) = -23.*log10(Stmod(i,ii)) + 2.5;
        else
            error('"fill_relative_sound_power_spectrum" funtion logic
      at line ~1126')
        end
    end
end

end


%.....................................................................
      ....

function [DI] = fill_DI_meth_2(teta,St,microphone_number)
global directivity_section
%%%% fill in DI method 2 and meth 1
```

```
% load teta
% load St
% load s_xp

s_teta = size(teta);
s_St   = size(St);
DI = zeros(s_teta(1,2),s_St(1,1),microphone_number);

% Approximation of the equation of the curve
for i=1:microphone_number   % number of micro
    for ii=1:s_teta(1,2)    % number of sources
        for iii=1:s_St(1,1) % frequency
            if directivity_section==1 %%%% Wesley's curve fitting
      method (summer 2013)
                St_log = log10(St(iii,1)) + 3;
                Tran1 = 23 + 12.*St_log;
                Tran2 = 90 + 12.*St_log;
                if teta(i,ii)<=Tran1
                    DI(ii,iii,i) = ((400-70.*St_log)/1000).*(teta(i,ii)
      - 35 - 8.*St_log) + 13.0 - 2.6.*St_log;
                elseif (teta(i,ii)>=Tran1) && (teta(i,ii)<=Tran2)
                    DI(ii,iii,i) = (-(280-
      35.*St_log)/1000).*(teta(i,ii) - 35 - 10.*St_log) + 6.5 -
      1.0.*St_log;
                elseif teta(i,ii)>Tran2
                    DI(ii,iii,i) = (-(100-0.*St_log)/1000).*teta(i,ii)
      - 0.3 + 2.*St_log;
                else
                    error('ERROR "fill_DI_meth_2" function in "Acoustic
      Loads subroutines 2.13" line ~1260\n')
                end
            elseif directivity_section==2 %%% Wilby Memo # 4: Eq. 4.12
                DI(ii,iii,i) = 34.61059 + 3.32662.*log10(St(iii,1)) ...
                               + 0.326131.*(log10(St(iii))).^2 ...
                               - 6.215516.*(teta(i,ii)./10) ...
                               + 0.3097785.*(teta(i,ii)./10).^2 ...
                               - 0.00596348.*(teta(i,ii)./10).^3;
            elseif directivity_section==3 %%%% Wilby Memo # 4: Eq. 4.13
                DI(ii,iii,i) = 18.89991 + 4.0511.*log10(St(iii,1)) ...
                               + 0.3898576.*(log10(St(iii,1))).^2 ...
                               - 3.24416.*(teta(i,ii)./10) ...
                               - 0.1596216.*(teta(i,ii)./10).^2 ...
                               - 0.00305944.*(teta(i,ii)./10).^3;
            elseif directivity_section==4 %%%% Wilby Memo # 4: Eq. 4.14
                if teta(i,ii)<170
                    delta = 0;
                else
                    if St(iii,1)<0.01
                        delta = -28.9339;
                    elseif St(iii,1)>0.04345
                        delta = 0;
                    else
                        delta = 61.7691 + 45.3515.*log10(St(iii,1));
                    end
                end
```

```
                DI(ii,iii,i) = 0.088828 + 7.411814.*log10(St(iii,1))
      ...
                                + 1.607279.*(log10(St(iii,1))).^2 +
      delta;
            elseif directivity_section==5 %%%% Plotkin and Sutherland
      Directivity
                A = 0.37;
                B = 310;
                C = 9;
                D = 0.698;
                E = 1.67;
                M_ec = 0.75;    %typical eddy convection Mach No. for a
      heated jet
                SnDI = 0.0515;  %value of Strouhal's for which DI is a
      maximum
                % DI_max   = 4.74 - 0.698*log10(St(iii,1)); % maximum
      directivity index
                % Beta_max = 57.4 - 9.61*log10(St(iii,1));  % maximum
      beta
                Beta_eff = (teta(i,ii) -
      9.61.*log10(St(iii,1)./SnDI)).*(pi./180); % radians
                %%%%%%%%%%%%Note:  Check SnDI theory/value
                DI(ii,iii,i) = 10.*log10(A*(1 + (cos(Beta_eff)).^4) ...
                               ./((((1 - M_ec.*cos(Beta_eff)).^2 ...
                               + 0.3.*M_ec.^2).^2.5) ...
                               .*(1 + B.*exp(-C.*Beta_eff)))) ...
                               - D.*log10(St(iii,1)) - E;
            else
                error('ERROR "directivity section" logic in "fill DI
      meth 2" in "Acoustic Loads subroutines 2.09" \n')
            end
        end
    end
end
% save DI DI

end

%
      %.....................................................................
      ..........
%
% function [deltaLfc] =
      ground_reflection(file_number,valamp_number,valphase_number,r,X,X
      peq_loop,microphone_number,Xint,Freq_number,ao,AA)
%
% % load file
% % load valamp
% % load valphase
% % load r
% % load X
% % load Xp %= Xpeq_loop
% % load Xint
% % load Freq = Freq_number
% % load Freqc = Freqc_number
% % load ao
```

```
% % load teta
% % load AA
% % load lfen
%
% % s_teta=size(teta);
% % phi=valphase_number;
% % g=valamp_number;
% % s_xp=size(Xpeq_loop)=microphone_number;
%
% if file_number==1
%     % load lfen
%     s_xint=size(Xint);
%     rprim    = zeros(microphone_number,s_xint(1,1));
%     to       = zeros(microphone_number,s_xint(1,1));
%     deltaLfc = zeros(microphone_number,s_xint(1,1));
%     for ii=1:microphone_number  %% loop for number of micro
%         for i=1:s_xint(1,1)     %% loop for position of the source
%             rprim(ii,i) = ((Xpeq_loop (ii,1))^2+(Xpeq_loop(ii,2)-
      X(i,2))^2+(Xpeq_loop(ii,3)+X(i,3))^2)^(1/2);
%             to(ii,i) = (rprim(ii,i)-r(ii,i))/ao;
%         end
%     end
%     % save rprim rprim
%     for ii=1:microphone_number  %% loop for number of micro
%         for i=1:s_xint(1,1)     %% loop for position of the source
%             deltaLfc(ii,i) =
      10*log10(1+((r(ii,i)/rprim(ii,i))*valamp_number)^2+2*((r(ii,i)/rp
      rim(ii,i))*valamp_number)*(sin(pi*to(ii,i)*(Freq_number(1,i+1)-
      Freq_number(1,i)))/(pi*to(ii,i)*(Freq_number(1,i+1)-
      Freq_number(1,i))))*cos(pi*to(ii,i)*(Freq_number(1,i+1)+Freq_numb
      er(1,i))-valphase_number));
%             if to(ii,i)==0
%                 deltaLfc(ii,i) = 6;
%             end
%             if i>2 && deltaLfc(ii,i-1)<deltaLfc(ii,i)
%                 deltaLfc(ii,i-1) = (deltaLfc(ii,i-2) +
      deltaLfc(ii,i))/2;
%             end
%         end
%     end
%     % for i=1:microphone_number%% loop for number of micro
%     %     f_micnb(i,1)=i;
%     % end
%     % f_micnb = (1:microphone_number)';
%     % a='deltaLfc,Nozzle n';
%     % b=48+lfen;
%     % cint=strcat(a,b);
%     % DLM=str2mat(cint);
%     % l_micnb={'Micro number:'};
%     % xlswrite('data',l_micnb,DLM,'A2')
%     % xlswrite('data',f_micnb,DLM,'A3')
%     % l_fc={'fc'};
%     % xlswrite('data',l_fc,DLM,'B1')
%     % xlswrite('data',Freqc_number,DLM,'C1')
%     % xlswrite('data',deltaLfc,DLM,'C3')
%     % save deltaLfc deltaLfc
% end
```

```
%
% if file_number==2
%     % load AA
%     % load lfen
%     s_AA=size(AA);
%     rprim    = zeros(microphone_number,s_AA(1,1));
%     to       = zeros(microphone_number,s_AA(1,1));
%     deltaLfc = zeros(microphone_number,s_AA(1,1),s_AA(1,2));
%     for ii=1:microphone_number  %% loop for number of micro
%         for i=1:s_AA(1,1)       %% loop for position of the source
%             rprim(ii,i)=((Xpeq_loop (ii,1))^2+(Xpeq_loop (ii,2)-
      X(i,2))^2+(Xpeq_loop (ii,3)+X(i,3))^2)^(1/2);
%             to(ii,i)=(rprim(ii,i)-r(ii,i))/ao;
%         end
%     end
%     for iii=1:microphone_number %% loop for number of micro
%         for i=1:s_AA(1,1)       %% position of sources
%             for ii=1:s_AA(1,2)  %% Centered frequency
%
      deltaLfc(i,ii,iii)=10*log10(1+((r(iii,i)/rprim(iii,i))*valamp_num
      ber)^2+2*((r(iii,i)/rprim(iii,i))*valamp_number)*(sin(pi*to(iii,i
      )*(Freq_number(1,ii+1)-
      Freq_number(1,ii)))/(pi*to(iii,i)*(Freq_number(1,ii+1)-
      Freq_number(1,ii))))*cos(pi*to(iii,i)*(Freq_number(1,ii+1)+Freq_n
      umber(1,ii))-valphase_number));
%             end
%         end
%     %     f_micnb(iii,1)=iii;
%     end
%     % f_micnb = (1:microphone_number)';
%     for i=1:microphone_number
%         test=test+1;
%         if test==10
%             test=0;
%             test2=test2+1;
%         end
%         % if i<10
%         %     a='deltaLfc micro n';
%         %     b=48+i;
%         %     c=',Nozzle n';
%         %     d=48+lfen;
%         %     cint=strcat(a,b,c,d);
%         %     DLM=str2mat(cint);
%         %     l_micnb={'Micro number:'};
%         %     xlswrite('data2',l_micnb,DLM,'A2')
%         %     xlswrite('data2',f_micnb,DLM,'A3')
%         %     l_fc={'fc'};
%         %     xlswrite('data2',l_fc,DLM,'B1')
%         %     xlswrite('data2',Freqc_number,DLM,'C1')
%         %     xlswrite('data2',deltaLfc(:,:,i),DLM,'C3')
%         % else
%         %     a='deltaLfc micro n';
%         %     b=48+test2;
%         %     c=48+test;
%         %     d=',Nozzle n';
%         %     e=48+lfen;
%         %     cint=strcat(a,b,c,d,e);
```

```
%         %     DLM=str2mat(cint);
%         %     l_micnb={'Micro number:'};
%         %     xlswrite('data2',l_micnb,DLM,'A2')
%         %     xlswrite('data2',f_micnb,DLM,'A3')
%         %     l_fc={'fc'};
%         %     xlswrite('data2',l_fc,DLM,'B1')
%         %     xlswrite('data2',Freqc_number,DLM,'C1')
%         %     xlswrite('data2',deltaLfc(:,:,i),DLM,'C3')
%         % end
%     end
%     % save deltaLfc deltaLfc
% end
%
% end
%
%
      %.....................................................................
      ..........
%
% function [rprim] =
      ground_reflection2_2(X,microphone_number,engine_number,loop_numbe
      r,octave_size,automatic_method,lfen_number,Xpeq_loop,alpha_loop,r
      prim,Lwsb,AA,deltaf_number,SPLsbpint,St)
% %%%% This method take the directivity of the reflection sources into
% %%%% consideration (we assume that the impedance of the ground is
      infinite)
% global SPLsbptotcm SPLsbptotsdcm
%
% % load X
% % load s_xp = microphone_number
% % load s_xpe = engine_number
% % load valalpha = alpha_ramp % not used in this program so why load
      it ????
% % load loop = loop_number
% % load Freqc = Freqc_number % not used in this program so why load it
      ????
% % load valaa = octave_size
% % load autoc = automatic_method
% % load lfen = lfen_number
% test=0;
%
% if loop_number==5
%     % load Xp = Xpeq_loop
%     Xr(:,1)=X(:,1);
%     Xr(:,2)=X(:,2);
%     Xr(:,3)=-X(:,3);
%     %%% length of the radius line and determination of teta
%     s_xr=size(Xr);
%     rprim = zeros(microphone_number,s_xr(1,1));
%     gamma_ar = zeros(microphone_number,s_xr(1,1));
%     gamma_br = zeros(microphone_number,s_xr(1,1));
%     gamma_cr = zeros(microphone_number,s_xr(1,1));
%     gammar = zeros(microphone_number,s_xr(1,1));
%     for ii=1:microphone_number  %% loop for number of micro
%         for i=1:s_xr(1,1)       %% loop for position of the source
%             rprim(ii,i)=((Xpeq_loop(ii,1))^2+(Xr(i,2)-
      Xpeq_loop(ii,2))^2+(Xr(i,3)-Xpeq_loop(ii,3))^2)^(1/2);
```

```
%             if Xpeq_loop (ii,2)==Xr(i,2) || Xr(i,2)==0
%                 if Xpeq_loop(ii,2)==Xr(i,2)
%                     gammar(ii,i)=90;
%                 end
%                 if Xr(i,2)==0
%                     gamma_ar(ii,i) = abs(Xpeq_loop(ii,3)-Xr(i,3));
%                     gamma_br(ii,i) = abs(Xpeq_loop(ii,2)-Xr(i,2));
%                     gamma_cr(ii,i) =
    gamma_br(ii,i)/cos(atan(abs(Xpeq_loop(ii,1))/gamma_br(ii,i)));
%                     % why is the same array writen twice ????????
%                     gammar(ii,i) =
    atan(gamma_ar(ii,i)/gamma_cr(ii,i))*180/pi;
%                     gammar(ii,i) = -gammar(ii,i)-alpha_loop+90;
%                 end
%             else
%                 gamma_ar(ii,i) = abs(Xpeq_loop(ii,3)-Xr(i,3));
%                 gamma_br(ii,i) = abs(Xpeq_loop(ii,2)-Xr(i,2));
%                 gamma_cr(ii,i) =
    gamma_br(ii,i)/cos(atan(abs(Xpeq_loop(ii,1))/gamma_br(ii,i)));
%                 gammar(ii,i) =
    atan(gamma_ar(ii,i)/gamma_cr(ii,i))*180/pi;
%             end
%         end
%     end
%
%     % for ii=1:microphone_number %% loop for number of micro
%     %     for i=1:s_xr(1,1) %% loop for position of the reflected
    sources
%     %         tetar(ii,i)=180-gammar(ii,i)-alpha_loop; %%% (need to
    show at the user)
%     %     end
%     % end
%     tetar = 180-gammar-alpha_loop;
%     % save tetar tetar
%     %%% Show teta and load DI from figure 10
%     if automatic_method==0
%         for i=1:microphone_number
%             test=test+1;
%             if test==10
%                 test=0;
%                 test2=test2+1;
%             end
%             % if i<10
%             %     a='DI refl. micro n';
%             %     b=48+i;
%             %     c=',Nozzle n';
%             %     d=48+lfen_number;
%             %     cint=strcat(a,b,c,d);
%             %     DIM=str2mat(cint);
%             %     l_teta={'teta(slices)'};
%             %     xlswrite('data2',l_teta,DIM,'A2')
%             %     xlswrite('data2',tetar(i,:)',DIM,'A3')
%             %     l_st={'Strouhal numbers'};
%             %     xlswrite('data2',l_st,DIM,'B1')
%             %     xlswrite('data2',St',DIM,'C1')
%             % else
%             %     a='DI refl. micro n';
```

```
%             %     b=48+test2;
%             %     c=48+test;
%             %     d=',Nozzle n';
%             %     e=48+lfen_number;
%             %     cint=strcat(a,b,c,d,e);
%             %     DIM=str2mat(cint);
%             %     l_teta={'teta(slices)'};
%             %     xlswrite('data2',l_teta,DIM,'A2')
%             %     xlswrite('data2',tetar(i,:)',DIM,'A3')
%             %     l_st={'Strouhal numbers'};
%             %     xlswrite('data2',l_st,DIM,'B1')
%             %     xlswrite('data2',St',DIM,'C1')
%             % end
%         end
%     end
%     % save rprim rprim
% end
%
% if loop_number==6
%     % load Lwsb
%     % load AA
%     % load rprim
%     % load deltaf = deltaf_number
%     % delta_fb=deltaf'; %%% bandwidth of the frequency band for
      "bandwidth" function
%     % fb=Freqc_number; %%% center frequencies
%     AA_size=size(AA);
%     % test=0;
%     % test2=0;
%     if automatic_method==0
%         for i=1:microphone_number
%             % if i<10
%             %     a='DI refl. micro n';
%             %     b=48+i;
%             %     c=',Nozzle n';
%             %     d=48+lfen_number;
%             %     cint=strcat(a,b,c,d);
%             %     DIM=str2mat(cint);
%             %     DIr_int(:,:,i)=xlsread('data2',DIM);
%             %     DIr(:,:,i)=DIr_int(3:end,3:end,i);
%             % else
%             %     a='DI refl. micro n';
%             %     b=48+test2;
%             %     c=48+test;
%             %     d=',Nozzle n';
%             %     e=48+lfen_number;
%             %     cint=strcat(a,b,c,d,e);
%             %     DIM=str2mat(cint);
%             %     DIr_int(:,:,i)=xlsread('data2',DIM);
%             %     DIr(:,:,i)=DIr_int(3:end,3:end,i);
%             % end
%         end
%         % save DIr DIr
%     else
%         [DIr] = fill_DI_r_meth_2(tetar,St,microphone_number);
%         % load DIr
%     end
```

```
%
%     %%%% Sound pressure level in each frequency band at each point
      contributed
%     %%%% by each each slice
%     SPLsbpr =
      zeros(X_flow_axis_length,frequency_band_center_length,microphone_
      number);
%     SPLsbpr_10 =
      zeros(X_flow_axis_length,frequency_band_center_length,microphone_
      number);
%     SPLsbprint =
      zeros(X_flow_axis_length,frequency_band_center_length,microphone_
      number);
%     SPLbp = zeros(1,frequency_band_center_length,microphone_number);
%     SPLbp_10 =
      zeros(1,frequency_band_center_length,microphone_number);
%     for iii=1:microphone_number    %% loop for number of micro
%         for i=1:X_flow_axis_length       %% position of sources
%             for ii=1:frequency_band_center_length  %% Centered
      frequency
%                 SPLsbpr(i,ii,iii) = Lwsb(i,ii)-
      10*log10(rprim(iii,i)^2)-11+DIr(i,ii,iii);
%                 SPLsbpr_10(i,ii,iii) = SPLsbpr(i,ii,iii)/10;
%                 SPLsbprint(i,ii,iii) = 10^(SPLsbpr_10(i,ii,iii));
%             end
%         end
%
%         %%%% Sound pressure level in each frequency band at each
      point by
%         %%%% logarithmic summation of conditions from each of the
      slices
%         % load SPLsbpint
%         for ii=1:frequency_band_center_length     %% Centered
      frequency
%             SPLbp(1,ii,iii) =
      10*log10(sum(SPLsbpint(:,ii,iii))+sum(SPLsbprint(:,ii,iii)));
%             SPLbp_10(1,ii,iii) = SPLbp(1,ii,iii)/10;
%         end
%     end
%
%     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     %%%%% Sound pressure level frequency spectrum
%     SPLbptot = zeros(frequency_band_center_length,1);
%     SPLbptotsd = zeros(frequency_band_center_length,1);
%     SPLbptotcm =
      zeros(frequency_band_center_length,microphone_number,lfen_number)
      ;
%     SPLbptotsdcm =
      zeros(frequency_band_center_length,microphone_number,lfen_number)
      ;
%     for iii=1:microphone_number    %% loop for number of micro
%         for i=1:X_flow_axis_length       %% position of sources
%             for ii=1:frequency_band_center_length  %% Centered
      frequency
%                 SPLbptotcm(ii,iii) = SPLbp(1,ii,iii);
%                 SPLbptot(ii,1) = SPLbp(1,ii,iii);
```

```
%                 SPLbptotsdcm(ii,iii) = SPLbptotcm(ii,iii)-
    10*log10(deltaf_number(ii,1));
%                 SPLbptotsd(ii,1) = SPLbptot(ii,1)-
    10*log10(deltaf_number(ii,1));
%                 if lfen_number>1
%                     SPLsbptotcm(ii,iii,lfen_number) =
    SPLbptotcm(ii,iii);
%                     SPLsbptotsdcm(ii,iii,lfen_number) =
    SPLbptotsdcm(ii,iii);
%                 end
%             end
%         end
%         % iiii=iii+microphone_number*(lfen_number-1);
%         h='Microphone:';
%         hh=',Exhaust number:';
%         hhh=strcat(h,48+iii,hh,48+lfen_number);
%         hhhh=str2mat(hhh);
%         figure('Name',hhhh);
%
    semilogx(Freqc_number,SPLbptot,'b',Freqc_number,SPLbptotsd,'r');
%         legend(strcat('Bandwidth: 1/',num2str(octave_size),'
    octave'),'Spectral density');
%         xlabel('Frequency in Hertz','FontSize',16)
%         ylabel('Sound pressure level in dB','FontSize',16)
%         title('\it{Frequency Spectrum}','FontSize',16)
%         xlim([10 20000])
%         ylim([0 160])
%         grid on
%     end
%
%     %%%%%%%%%% save SPLbptot to sum the SPL for different
%     %%%%%%%%%% exhaust
%     % [filename, pathname, filterindex] = uiputfile('*.mat', 'Save
    sound pressure level frequency spectrum');
%
%     % if filterindex == 1
%     %     chemin=cd;
%     %     cd(pathname);
%     %
    save(filename,'SPLbptotcm','Freqc_number','SPLbptotsdcm','deltaf'
    );
%     %     cd(chemin);
%     % end
%     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     %%%% Overall sound pressure level at any point p over the entire
    spectrum
%
%     % save SPLbp_10 SPLbp_10
%     SPLbpint =
    zeros(1,frequency_band_center_length,microphone_number);
%     SPLoap = zeros(1,1,microphone_number);
%     SPLoapint = zeros(microphone_number,1);
%     for iii=1:microphone_number  %% loop for number of micro
%         for ii=1:frequency_band_center_length    %% Centered
    frequency
%             SPLbpint(1,ii,iii) = 10^(SPLbp_10(1,ii,iii));
```

```
%         end
%         SPLoap(1,1,iii) = 10*log10(sum(SPLbpint(1,:,iii)));
%         SPLoapint(iii,1) = SPLoap(1,1,iii);
%     end
%     % for i=1:microphone_number
%     %     f_micnb(i,1)=i;
%     % end
%     % f_micnb = (1:microphone_number)';
%     % a='SPLoap';
%     % b=',Nozzle n';
%     % c=48+lfen_number;
%     % cint=strcat(a,b,c);
%     % SPL=str2mat(cint);
%     % l_micnb={'Micro number:'};
%     % l_SPLoap={'SPLoap with the reflection of the ground'};
%     % xlswrite('data2',l_micnb,SPL,'A1')
%     % xlswrite('data2',f_micnb,SPL,'A2')
%     % xlswrite('data2',l_SPLoap,SPL,'B1')
%     % xlswrite('data2',SPLoapint,SPL,'B2')
%     if lfen_number==1
%         if engine_number>1
%             SPLsbptotcm_10 =
    zeros(frequency_band_center_length,microphone_number,engine_numbe
    r);
%             SPLsbptotcmint =
    zeros(frequency_band_center_length,microphone_number,engine_numbe
    r);
%             SPLtotcmint = zeros(frequency_band_center_length,1,1);
%             SPLtotcmint1 =
    zeros(frequency_band_center_length,microphone_number,1);
%             SPLtotcmint2 =
    zeros(frequency_band_center_length,microphone_number,1);
%             SPLtotcmint_10 =
    zeros(frequency_band_center_length,microphone_number,1);
%             SPLtotcmintsd = zeros(frequency_band_center_length,1,1);
%             SPLtotoap = zeros(1,microphone_number);
%             for iii=1:microphone_number    %% loop for number of
    micro
%                 for ii=1:frequency_band_center_length      %%
    Centered frequency
%                     for i=1:engine_number  %% exhaust nozzle number
%                         SPLsbptotcm_10(ii,iii,i) =
    SPLsbptotcm(ii,iii,i)/10;
%                         SPLsbptotcmint(ii,iii,i) =
    10^(SPLsbptotcm_10(ii,iii,i));
%                     end
%                 end
%                 for ii=1:frequency_band_center_length      %%
    Centered frequency
%                     SPLtotcmint1(ii,iii,1) =
    10*log10(sum(SPLsbptotcmint(ii,iii,:)));
%                     SPLtotcmint(ii,1,1) = SPLtotcmint1(ii,iii,1);
%                     SPLtotcmintsd(ii,1,1) = SPLtotcmint(ii,1,1)-
    10*log10(deltaf_number(ii,1));
%                     SPLtotcmint_10(ii,iii,1) =
    SPLtotcmint1(ii,iii,1)/10;
```

```
%                     SPLtotcmint2(ii,iii,1) =
      10^(SPLtotcmint_10(ii,iii,1));
%                 end
%                 % iiii=iii+microphone_number*lfen_number;
%                 h='Sum of all the nozzles; Microphone:';
%                 hh=strcat(h,48+iii);
%                 hhh=str2mat(hh);
%                 figure('Name',hhh);
%
      semilogx(Freqc_number,SPLtotcmint,'b',Freqc_number,SPLtotcmintsd,
      'r');
%                 legend(strcat('Bandwidth: 1/',num2str(octave_size),'
      octave'),'Spectral density');
%                 xlabel('Frequency in Hertz','FontSize',16)
%                 ylabel('Sound pressure level in dB','FontSize',16)
%                 title('\it{Frequency Spectrum}','FontSize',16)
%                 xlim([10 20000])
%                 ylim([0 160])
%                 grid on
%                 %%% sum of all the frequency
%                 SPLtotoap(1,iii) =
      10*log10(sum(SPLtotcmint2(:,iii,1)));
%             end
%             % l_SPL={'Sum of the SPL created by each exhaust
      nozzle'};
%             % xlswrite('data2',l_SPL,'SPLoap total','B1')
%             % xlswrite('data2',l_micnb,'SPLoap total','A2')
%             % xlswrite('data2',f_micnb,'SPLoap total','A3')
%             % xlswrite('data2',SPLtotoap','SPLoap total','B3')
%         end
%         % Excel = actxserver('Excel.application');
%         % Excel.Visible = 1;
%         % winopen data2.xls
%     end
% end
%
% end
%
      %.............................................................
      ..........
%
% function [DIr] = fill_DI_r_meth_2(tetar,St,microphone_number)
% global directivity_section
% %%%% fill in DIr method 2
%
% % load tetar
% % load St
% % "Xp" is not needed in this function so why load it?????
% % load Xp = Xpeq_loop
% % load s_xp = microphone_number
%
% s_tetar = size(tetar);
% s_St   = size(St);
% DIr = zeros(s_tetar(1,2),s_St(1,1),microphone_number);
%
% %%% Approximation of the equation of the curve
% for i=1:microphone_number    %% number of micro
```

```
%     for ii=1:s_teta(1,2)     %% number of sources
%         for iii=1:s_St(1,1)  %% frequency
%             if directivity_section==1 %%%% Wesley's curve fitting
      method (summer 2013)
%                 St_log = log10(St(iii,1)) + 3;
%                 Tran1 = 23 + 12.*St_log;
%                 Tran2 = 90 + 12.*St_log;
%                 if teta(i,ii)<=Tran1
%                     DIr(ii,iii,i) = ((400-
      70.*St_log)/1000).*(teta(i,ii) - 35 - 8.*St_log) + 13.0 -
      2.6.*St_log;
%                 elseif (teta(i,ii)>=Tran1) && (teta(i,ii)<=Tran2)
%                     DIr(ii,iii,i) = (-(280-
      35.*St_log)/1000).*(teta(i,ii) - 35 - 10.*St_log) + 6.5 -
      1.0.*St_log;
%                 elseif teta(i,ii)>Tran2
%                     DIr(ii,iii,i) = (-(100-
      0.*St_log)/1000).*teta(i,ii) - 0.3 + 2.*St_log;
%                 else
%                     error('ERROR "fill_DI_r_meth_2" function in
      "Acoustic Loads subroutines 2.13" line ~1768\n')
%                 end
%             elseif directivity_section==2 %%% Wilby Memo # 4: Eq.
      4.12
%                 DIr(ii,iii,i) = 34.61059 + 3.32662*log10(St(iii)) ...
%                                 + 0.326131*(log10(St(iii)))^2 ...
%                                 - 6.215516*(tetar(ii)/10) ...
%                                 + 0.3097785*(tetar(ii)/10)^2 ...
%                                 - 0.00596348*(tetar(ii)/10)^3;
%             elseif directivity_section==3 %%%% Wilby Memo # 4: Eq.
      4.13
%                 DIr(ii,iii,i) = 18.89991 + 4.0511*log10(St(iii)) ...
%                                 + 0.3898576*(log10(St(iii)))^2 ...
%                                 - 3.24416*(tetar(ii)/10) ...
%                                 - 0.1596216*(tetar(ii)/10)^2 ...
%                                 - 0.00305944*(tetar(ii)/10)^3;
%             elseif directivity_section==4 %%%% Wilby Memo # 4: Eq.
      4.14
%                 if tetar(ii)<170
%                     delta = 0;
%                 else
%                     if St(iii,1)<0.01
%                         delta = -28.9339;
%                     elseif St(iii,1)>0.04345
%                         delta = 0;
%                     else
%                         delta = 61.7691 + 45.3515*log10(St(iii,1));
%                     end
%                 end
%                 DIr(ii,iii,i) = 0.088828 + 7.411814*log10(St(iii,1))
      ...
%                                 + 1.607279*(log10(St(iii,1)))^2 +
      delta;
%             elseif directivity_section==5 %%%% Plotkin and Sutherland
      Directivity
%                 A = 0.37;
%                 B = 310;
```

```
%                 C = 9;
%                 D = 0.698;
%                 E = 1.67;
%                 M_ec = 0.75;    %typical eddy convection Mach No. for
      a heated jet
%                 SnDI = 0.0515;  %value of Strouhal's for which DIr is
      a maximum
%                 % DI_max   = 4.74 - 0.698*log10(St(iii,1)); % maximum
      directivity index
%                 % Beta_max = 57.4 - 9.61*log10(St(iii,1));  % maximum
      beta
%                 Beta_eff = (tetar(ii) -
      9.61*log10(St(iii,1)/SnDI))*(pi/180); % radians
%                 %%%%%%%%%%%Note:  Check SnDI theory/value
%                 DIr(ii,iii,i) = 10*log10(A*(1 + (cos(Beta_eff))^4)
      ...
%                                 / ((((1 - M_ec*cos(Beta_eff))^2 ...
%                                 + 0.3*M_ec^2)^2.5)*(1 ...
%                                 + B*exp(-C*Beta_eff)))) ...
%                                 - D*log10(St(iii,1)) - E;
%             else
%                 error('ERROR "directivity section" logic in "fill DIr
      meth 2" in "Acoustic Loads subroutines 2.09" \n')
%             end
%         end
%     end
% end
% % save DIr DIr
%
% end
%
%
      %.....................................................................
      ..........

function [P_Po_mag] =
      Diffraction_off_a_Cylinders(X_flow_axis_beta,X_flow_axis_length,
      ...

      frequency_band_center,frequency_band_center_length, ...
                                                    microphone_number,ao)
% P_Po_mag(X_flow_axis_beta,frequency_band_center,microphone_number);
% X_flow_axis_beta(microphone_number,X_flow_axis_length)
% size(frequency_band_center) % (1,31)

% see "Sound Diffraction by Rigid Spheres and Circular Cylinders"
%      Francis M. Wiener, May 1947
% or  "Force Due to Diffraction of Sound Wave on Small-Diameter
%      Cylindrical Particles"
%      H. Czyz and T. Gudra, April 1992

% From function "Acoustic_Loads_subroutines_2_13"
global pressure_doubling_display microphone_r_data microphone_h_data

fprintf('Diffraction Equation Calculations \n')
% pause
```

```
% size(microphone_r_data)
% size(microphone_h_data)
% size(X_flow_axis_beta)
% size(frequency_band_center)
% X_flow_axis_length
% error('STOP')

% a_radius = microphone_r_data;
% angle_frequency = 2.*pi.*frequency_band_center;
% k_wave_number = angle_frequency./ao;
% ka = k_wave_number.*a_radius
% max_m = round(1.5*ka)
% pause

matlabpool(6)

t_rate_total = 0;
P_Po_mag =
      zeros(X_flow_axis_length,frequency_band_center_length,microphone_
      number);
for C=1:microphone_number % loop for microphones
    % C
    for A=1:X_flow_axis_length % loop for assumed source locations
        tic
        if A>1
            fprintf('Loop #:%4.0f of %4.0f, %4.0f of %4.0f , time to
      complete: %3.2f min.
      \n',C,microphone_number,A,X_flow_axis_length,t_rate*(X_flow_axis_
      length-A)/60)
        end
        %for B=1:frequency_band_center_length% loop for centered band
      frequencies
        parfor B=1:frequency_band_center_length% loop for centered band
      frequencies

            Beta_deg = X_flow_axis_beta(C,A);
            if Beta_deg<0.001
                Beta_deg=0.001;
            end
            Beta_rad = Beta_deg*pi/180;

            Phi_deg = microphone_h_data(C,1);
            Phi_rad = Phi_deg*pi/180;

            a_radius = microphone_r_data(C,1);
            angle_frequency = 2.*pi.*frequency_band_center(1,B);
            k_wave_number = angle_frequency./ao;
            ka = k_wave_number*a_radius*sin(Beta_rad);
            max_m = round(1.5*ka);
            if max_m<25
                max_m=25;
            end

            ka_1 = 4/(pi*ka);
            m = 0;
```

```
            E_m = 1/2;
            besselh_m = (m*besselh(m,2,ka))/ka - besselh(m+1,2,ka);
            sum_Hankel = E_m*cos(m*Phi_rad)./besselh_m;
            m = m+1;
            while m<max_m
                E_m = 1*1i.^m;
                besselh_m = (m*besselh(m,2,ka))/ka - besselh(m+1,2,ka);
                sum_Hankel = sum_Hankel +
      E_m*cos(m*Phi_rad)./besselh_m;
                m = m+1;
            end
            P_Po_mag(A,B,C) = (abs(ka_1*sum_Hankel)).^2; % pressure
      squared
            %
      P_Po_mag(X_flow_axis_beta,frequency_band_center,microphone_number
      );
        end
        t_rate = toc;
        t_rate_total = t_rate_total + t_rate/60;
    end
end
matlabpool close
t_rate_total

if pressure_doubling_display==1
    P_Po_mag_dB = 10*log10(P_Po_mag);
%     size_Freqc_number = size(frequency_band_center);
%     size_P_Po_mag_dB = size(P_Po_mag_dB);
    fprintf('Pressure Doubling Values: \n\n')
    P_Po_mag_dB
%     fprintf('Center Freq. (Hz)')
%     fprintf(': Mic. 1 (dB) ')
%     if size_P_Po_mag_dB(1,2)>1
%         for a=2:size_P_Po_mag_dB(1,2)
%             fprintf(':      %1.0f ',a)
%         end
%     end
%     fprintf(' \n')
%     for b=1:size_Freqc_number(1,1)
%         fprintf(' %15.2f ',frequency_band_center(1,b))
%         fprintf(': %11.2f ',P_Po_mag_dB(b,1))
%         if size_P_Po_mag_dB(1,2)>1
%             for a=2:size_P_Po_mag_dB(1,2)
%                 fprintf(':  %2.2f  ',P_Po_mag_dB(b,a))
%             end
%         end
%         fprintf('\n')
%     end
end

end
%.....................................................................
      ....
```

```
% function [delta] =
      Multi_Diffraction_off_a_Cylinders(microphone_number,microphone_da
      ta,X_length,X)
%
% % Assuming a flow only in the y-z direction
%
% for i=1:microphone_number
%     for ii=1:X_length
%         delta_num(1,i,ii) = X(ii,2) - microphone_data(i,2);
%         delta_den(1,i,ii) = microphone_data(i,3) - X(ii,3);
%         % This will work for 3D.
% %         delta_num(:,i,ii) = microphone_data(i,:) - X(ii,:);
% %         delta_den(1,i,ii) = norm(delta_num(:,i,ii));
%         if delta_den(1,i,ii)<0
%             delta(1,i,ii) = -
      (180/pi).*atan(delta_num(1,i,ii)./delta_den(1,i,ii));
%         elseif delta_den(1,i,ii)>=0
%             delta(1,i,ii) = 180 -
      (180/pi).*atan(delta_num(1,i,ii)./delta_den(1,i,ii));
%         else
%             error('ERROR in "Multi Diffraction off a Cylinders" ~line
      1859')
%         end
%     end
% end
%
% end

%.....................................................................
      ....
```

Appendix 2 Wang Statistical Energy Analysis Model Matlab Code: Wang_SEA.m

```
function [TL,f_o] = Wang_SEA
% clear all
% close all
% clc

C_o = 343; %m/s
% C_o = realsqrt(1.4*287*(273.16+21.1)); % m/s Noise Control
rho_o = 1.21; % kg/m3 Noise Control
% rho_o = 1.205; % kg/m3 www.engineeringtoolbox.com
R = 0.3048; % m
L = 2.0066;
S = 2.*pi*R*L; % surface area m2
h = 0.00127; % m (0.050")
% h = 0.0013208; % m (0.052")
rho = 7850; % kg/m3, Mechanics of Materials 6th - J. Gere
% rho = 2.156.*(0.4536 ./0.0929)./h; % kg/m3 www.engineeringtoolbox.com
E = 2E11; % 200 GPa, Mechanics of Materials 6th - J. Gere
% E = 29.5E6.*6894.8; % Pa, www.engineeringtoolbox.com
% E = 26.1E6.*6894.8; % Pa
Poission = 0.28; % Mechanics of Materials 6th - J. Gere
% Poission = 0.303; % www.engineeringtoolbox.com
% Poission = 0.48;
m_s = rho*h;

% f_r = 2625.0;
f_r = (1./(2.*pi.*R)).*((E./rho).^0.5); % Wang 2635.6
% f_c = 10540.6;
f_c = (C_o.*C_o.*(12.^(0.5))./(4.*pi.*pi.*f_r.*R.*h); % 10,118
% w_c =  2.*pi.*f_c;

%.....................................................................
....
%%%%%%% Bandwidth analysis %%%%%%%
% From function "Acoustic_Loads_subroutines_2_13"
frequency_maximun = 20000;
frequency_center = 1000;
frequency_minimum = 20;
frequency_octave_size = 6;
iii=1;
ii=0;

bandwidth_ratio = 2.^(1/frequency_octave_size);

Freqc = zeros(1,9);
Freqc(1,1) = frequency_center;
while Freqc(1,1)>=frequency_minimum
    Freqc(1,1) = Freqc(1,1)/bandwidth_ratio;
end
while Freqc(1,iii)<=frequency_maximun
    iii=iii+1;
    Freqc(1,iii) = bandwidth_ratio*Freqc(1,iii-1);
end
```

```
deltafc = zeros(1,(iii-1));
while ii<iii
    ii=ii+1;
    deltafc(1,ii) = Freqc(1,ii).*((bandwidth_ratio-
1)./(bandwidth_ratio).^(1/2));
end

frequency_band_width = deltafc'; % The band width of each 1/N octaves
section based on Freqc.
frequency_band_center = Freqc';   % This is the center frequency of the
bandwidth divided into 1/N octaves.
% Freq_number or Freq were the lower frequency of a spectrum bandwidth
divided into 1/N octaves.

% frequency_band_width = 10;
% frequency_band_center =
[frequency_minimum:frequency_band_width:frequency_maximun]';

frequency_band = zeros(length(frequency_band_center),5);
frequency_band(:,1) = frequency_band_center - frequency_band_width./2;
frequency_band(:,2) = frequency_band_center;
frequency_band(:,3) = frequency_band_center + frequency_band_width./2;
w_frequency_band = 2.*pi.*frequency_band;
v_frequency_band = w_frequency_band/(2.*pi.*f_r);

%.....................................................................
....

f_o = frequency_band(:,2);
f_o_length = length(f_o);
w_o = 2.*pi.*f_o;
v_o_array = w_o/(2.*pi.*f_r);
k_o_array = w_o./C_o;

m = 0:1:300;
n = 0:1:300;
m_length = length(m);
n_length = length(n);
k_temp = (h.*h.*R.*R./(12.*(1-Poission.*Poission))).^(0.25);
% ka_C = m+0.5;
% P_array = 1-2./(ka_C.*pi);
% k_a_array = (m.*pi./(L.*ka_C)).*k_temp;
k_a_array = (m.*pi./L).*k_temp;
k_c_array = (n./R).*k_temp;

v_mn = zeros(m_length,n_length);
E_rad_array = zeros(f_o_length,2);
D1=0;
E1=1;
v_frequency_band_below=0;
v_frequency_band_above=0;
for B1=1:m_length
    k_a = k_a_array(B1);
%     P = P_array(B1);
```

```
%     P1 = (1-Poission.*Poission)./(1-Poission.*Poission.*P);
%     P2 = (1-Poission.*P.*P)./(P.*(1-Poission));
%     k_a2 = k_a.*k_a;
%     k_a4 = k_a2.*k_a2;
    for C1=1:n_length
        k_c = k_c_array(C1);
%         k_c2 = k_c.*k_c;
%         k_c4 = k_c2.*k_c2;
        k_temp = (k_a.*k_a+k_c.*k_c).*(k_a.*k_a+k_c.*k_c);
        v_mn(B1,C1) = sqrt(k_temp+k_a.*k_a.*k_a.*k_a./k_temp);
%         v_mn(B1,C1) =
sqrt(k_a4+k_c4+2.*k_a2.*k_c2.*P+k_a4./(P1.*(k_a4+k_c4)+2.*k_a4.*k_c2.*P
2));
        while D1==0
            if isnan(v_mn(B1,C1))
                v_frequency_band_below = v_frequency_band_below + 1;
                D1=1;
            elseif v_mn(B1,C1)<=v_frequency_band(1,1)
                v_frequency_band_below = v_frequency_band_below + 1;
                D1=1;
            elseif v_mn(B1,C1)>=v_frequency_band(f_o_length,3)
                v_frequency_band_above = v_frequency_band_above + 1;
                D1=1;
            elseif (v_mn(B1,C1)>=v_frequency_band(E1,1)) &&
(v_mn(B1,C1)<=v_frequency_band(E1,3))
                v_frequency_band(E1,4) = v_frequency_band(E1,4) + 1;
                k_o = k_o_array(E1);
                k_m = m(B1).*pi./L;
                k_n = n(C1).*pi./(2.*pi.*R);
%                   k_o.*k_o
%                   k_a.*k_a+k_c.*k_c
                k_check = (k_o.*k_o)>(k_m.*k_m+k_n.*k_n);
                v_frequency_band(E1,5) = v_frequency_band(E1,5) +
k_check; % acoustically fast
                ff = f_o(E1)./f_c;
                ff2 = (ff).^(0.5);
                kk = (k_a.*k_a+k_c.*k_c).^(0.5);
                piL = pi.*L.*((12.*(1-Poission.*Poission)).^(0.25));
                E_rad_temp = (((h.*R).^(0.5)).*(log(...
                    (1+ff2)./(1-ff2)+(2.*ff2)./(1-ff)))./(piL.*kk.*(1-
ff2));
                E_rad_array(E1,1) = E_rad_array(E1,1) + E_rad_temp; %
acoustically slow
                D1=1;
            elseif E1>f_o_length
                error('ERROR')
            else
                %nothing
            end
            E1=E1+1;
        end
        D1=0;
        E1=1;
    end
end
% total_number = 301*301
```

```
% total_number_mn = v_frequency_band_below + v_frequency_band_above +
sum(v_frequency_band(:,4))
n_w_o = v_frequency_band(:,4)./frequency_band_width;
hhhhh = figure;
axes1 = axes('Parent',hhhhh,...
    'Position',[0.10 0.10 0.80 0.75],...
    'XLim',[20 20000],...
    'XScale','log',...
    'GridLineStyle',':',...
    'Box','on',...

'xtick',[10:10:90,100:100:900,1000:1000:9000,10000:10000:100000],...

'XTickLabel',{'10';'';'';'';'';'';'';'';'';'100';'';'';'';'';'';'';'';'
';'1000';'';'';'';'';'';'';'';'';'';'10000';'';'';'';'';'';'';'';'';'10000
0'});
line(frequency_band(:,2),n_w_o,'Color','r','LineStyle','-
','Parent',axes1,'LineWidth',0.5);
xlabel(axes1,'Frequency (Hz)','Color',[0,0,0],'FontSize',12)
ylabel(axes1,'Modal Density (Modes/Hz)','Color',[0,0,0],'FontSize',12)

E_rad = zeros(f_o_length,1);
E_rad_array(:,2) = v_frequency_band(:,5)./v_frequency_band(:,4);
for F1=1:f_o_length
    test_F1 = real(E_rad_array(F1,2));
    if test_F1>=1 % acoustically fast
        E_rad(F1) = 1;
    elseif (test_F1<1) && (test_F1>0) % acoustically fast and slow
        E_rad(F1) = test_F1;
    elseif (test_F1==0) || isnan(test_F1)% acoustically slow
        E_rad(F1) = real(E_rad_array(F1,1));
    else
        error('ERROR')
    end
end

E_mech = zeros(f_o_length,1);
for G1=1:f_o_length
    test_G1 = v_o_array(G1,1);
    if test_G1<=0.5
        E_mech(G1) = 0.01.*(1./(11.224489795918.*test_G1 +
12.612244897959)).*(f_r./f_c);
    elseif (test_G1>0.5) && (test_G1<=0.9)
        E_mech(G1) =  (5.5E-4).*(f_r./f_c);
    elseif (test_G1>0.9) && (test_G1<1.1)
        E_mech(G1) =  0;%0.00001.*(1./1.5).*(f_r./f_c);
    elseif test_G1>=1.1
        E_mech(G1) = -(5E-8).*f_o(G1)+0.001;
    else
        error('ERROR')
    end
end

R_rad = rho_o.*C_o.*E_rad.*S;%
% E_mech_old = -(5E-8).*f_o+0.001;
% E_mech = 0.0012;
```

```
R_mech = S.*m_s.*E_mech.*w_o;%

% figure
% plot(v_o_array,E_mech)

% test0 = (1./E_mech).*(f_r./f_c);
% figure
% plot(v_o_array,test0)

% E_rad_plot = 10.*log10(E_rad);
% for G1=1:f_o_length
%     if isfinite(E_rad_plot(G1))
%         %nothing
%     else
%         E_rad_plot(G1)=-60;
%     end
% end
% figure
% plot(f_o,E_rad)

% Transmission of Sound Through a Cylindrical Shell and a Light
Aircraft Fuselage (Dissertation ) - Yiren Simon Wang - 1982

TL_res = -10.*log10((8.*pi.*pi.*C_o.*C_o.*n_w_o.*R_rad.*R_rad)./...
         (w_o.*w_o.*m_s.*S.*S.*(2.*R_rad+R_mech)));
%

TL_nr = zeros(f_o_length,1);
TL = zeros(f_o_length,1);
for A1=1:f_o_length
    v_o = v_o_array(A1,1);
    hR2 = (h/R).*(h/R);
    rho_C_o2 = rho_o.*rho_o.*C_o.*C_o;
    v_o_fr_fc2 = (v_o.*f_r./f_c).*(v_o.*f_r./f_c);
    v_o_fr_fc2_12 = (1-v_o_fr_fc2).*(1-v_o_fr_fc2);
    v_o_fr_fc2_15 = (1-v_o_fr_fc2).^0.5;

    if v_o<=1
        TL_nr(A1) = 8.33.*log10( (v_o.*v_o.*hR2.*E.*rho/(4.*rho_C_o2))
.* v_o_fr_fc2_12 + 2.3 )-3 ...
                  +20.*log10( (pi./2).*asin( (v_o.*v_o_fr_fc2_15).^0.5
) );
        TL(A1) = -10.*log10( 1./(10.^(TL_res(A1)./10)) +
1./(10.^(TL_nr(A1)./10)) );
    elseif (v_o>1.0) && (v_o<(f_c/f_r))
        TL_nr(A1) = 8.33.*log10( (v_o.*v_o.*hR2.*E.*rho/(4.*rho_C_o2))
.* v_o_fr_fc2_12 + 2.3 )-3;
        TL(A1) = -10.*log10( 1./(10.^(TL_res(A1)./10)) +
1./(10.^(TL_nr(A1)./10)) );
    elseif v_o>(f_c/f_r)
        TL(A1) = -10.*log10( 1./(10.^(TL_res(A1)./10)) + 0);
    else
        error('ERROR')
    end
end
```

```
% TL = -10.*log10( 1./(10.^(TL_res./10)) + 1./(10.^(TL_nr./10)) );
% TL1 = 10.*log10( TL_res.*TL_nr2 + TL_nr1 );
% TL = 10.*log10(TL_res + TL_nr);
% figure
% plot(v_o_array,TL_res,'b',v_o_array,TL_nr,'g',v_o_array,TL,'r')

% figure
%
plot(v_o_array,10.*log10(TL_res),'k',v_o_array,10.*log10(TL_nr1),'b',v_
o_array,10.*log10(TL_nr2),'g',v_o_array,TL,'r')
% plot(f_o,TL_nr,'r',f_o,TL_nr1,'g',f_o,TL_res,'b', f_o,TL,'k')
end
```