2004

Development & Validation of the Acoustic Analogy Code for Jet Noise Predictions

Leonardo A. Bueno
Embry-Riddle Aeronautical University - Daytona Beach

DEVELOPMENT & VALIDATION OF THE ACOUSTIC ANALOGY CODE FOR JET NOISE PREDICTIONS

by

Leonardo A. Bueno

A Thesis Submitted to the College of Aerospace Engineering
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Aerospace Engineering

Embry-Riddle Aeronautical University
Daytona Beach, Florida

DEVELOPMENT & VALIDATION OF THE ACOUSTIC ANALOGY CODE FOR JET NOISE PREDICTIONS

by

Leonardo A. Bueno

This thesis was prepared under the supervision of the candidate's thesis committee chairman, Dr. Golubev, Department of Aerospace Engineering, and has been approved by the members of his thesis committee. A thesis submitted to the College of Aerospace Engineering in partial fulfillment of the requirements for the degree of Master of Science in Aerospace Engineering.

Thesis committee:

- Dr. Golubev
  Committee Chairman
  [illegible]

- Dr. Perrell
  Member
  [illegible]

- Dr. Nakhla
  Member
  [illegible]

[illegible]
Department Chair, Aerospace Engineering

6/29/04
Date

ABSTRACT

Author: Leonardo A. Bueno

Title: Development & Validation of the Acoustic Analogy Code for Jet Noise Predictions

Institution: Embry-Riddle Aeronautical University

Degree: Masters of Science in Aerospace Engineering

Year: 2003

In this thesis, the numerical code for predicting the far-field sound radiated from a localized sound source is developed based on Lighthill's acoustic analogy theory. The acoustic analogy equation allows calculating the sound intensity in terms of the integral over the volume with distributed Lighthill tensor of unsteady flow fluctuations. A FORTRAN code is written to perform the integration using Simpson's numeric technique. The procedure for validating the code against the test case for the sound source in the form of a Gaussian pulse is examined. Future application of the code to predict acoustic radiation from turbulent jets is briefly discussed.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

a_o:	velocity of sound in ambient air.
D:	diameter of the Kirchhoff surface.
e_{ij}:	viscous stresses.
$\vec{f}$:	volume force.
I:	sound intensity.
J_o:	Bessel function of the first kind of order zero.
J_1:	Bessel function of the first kind of order one.
J_2:	Bessel function of the first kind of order two.
LEE:	Linearized Euler Equations
M_J:	Mach number of the jet.
p:	pressure from the source.
P_s:	sound pressure in the far-field.
r:	'r' location along the radial-axis to be considered during integration.
R_{ob}:	radius of the axis of the observer from the jet axis.
SIF:	Surface Integral Formulation
St:	Strouhal's number.
t:	time.
ts:	retarded time.
T_{ij}:	stress tensor.
u:	velocity component of source in the x-axis.
U_e:	mean velocity profile.
v:	velocity component of source in the r-axis.
$\vec{v}$:	velocity vector of source.
x:	'x' location along the x-axis to be considered during integration.
X:	location of observer.
Y:	location of source.
θ:	theta angle from the source to the observer location on the axis parallel to the jet axis.
ρ:	flow density.
σ:	sigma is the argument for the Bessel functions.
τ:	normal stress.
ω:	angular frequency.

1. INTRODUCTION

Aircraft engines produce aerodynamic noise sources, such as the nozzle turbulent jet. As modern aircraft engines are developed, a reduction of the sound becomes more critical. The ability to estimate pressure with precision is a necessity in order to achieve a decrease in its levels. Computational aeroacoustics is an effective way to obtain these predictions as it can be used to generate models of the existent noise. The methods used by computational aeroacoustics aid in the creation of appropriate case models due to their efficiency and cost.

This work addresses the prediction of aircraft noise, particularly the jet noise. There are several methods to predict the noise in the far-field sound propagation region (2). These include the linearized Euler equations (LEE), Kirchhoff's method, surface integral formulation (SIF) and Lighthill's theory. LEE is used in conjunction with large-scale simulation (LSS) because the acoustic wave propagation is governed by the linearized Euler equations, when it is not in the nonlinear sound generation region. In the case of Kirchhoff's method a cylindrical Kirchhoff's surface encloses all the sources and non-linear effects so that outside of this region the mean flow velocity is zero or constant. In order to find the pressure at a point outside the Kirchhoff's surface, it is required to have the pressure and its normal derivative at the surface, and this would require additional post-processing of results from large-scale wave-field simulations. The surface integral formulation (SIF), works in the same way as Kirchhoff's method but it tries to eliminate the need to obtain the pressure's normal derivative on the surface. It uses the method of images to make Green's function zero at the surface therefore eliminating the need for the pressure derivative of the surface (2).

In the present study, Lighthill's theory is used to solve for the pressure in the far field. Considering a compact region of the flow and arranging the Navier-Stokes equations so that they become a non-homogeneous wave equation, allows the decay of the source within the extension of the computational domain. The sound that emanates from a compact source provides a good solution in the case of low Mach numbers but the non-compactness of a source in supersonic jets can create a problem due to non-convergence of the integral (1). Solving Lighthill's integral over the near-field region provides the solution for the pressure in the far field. This allows for a calculation that does not have to extend to the point being calculated but just around the selected near field.

The objective is to develop the code needed to obtain the sound pressure distribution in the far-field. Lighthill's theory is used to write the necessary algorithms. This code develops a solution for the case of a point source for which the pressure equation is given. The pressure is validated as two different types of point sources are considered, the monopole point source and the Gaussian Pulse case.

A numeric method is used to solve Lighthill's theory in order to obtain the results wanted. Writing a program in the FORTRAN language is the solution preferred in order to speed up the calculations, due to high number of steps that have to be repeated. This is a program that can be configured to carry out the numeric technique for different types of point sources. This is done by using separate functions as much as possible, therefore allowing the modular change of only the necessary functions.

2. GOVERNING EQUATIONS

The governing equations of the unsteady flow motion are the compressible Navier-Stokes equations. These are used in computational aeroacoustics to describe the source and the transmission of aerodynamically generated sound. In general, the Navier-Stokes equations can be written as (3):

$$\rho\frac{Du}{Dt} = f_x - \frac{\partial p}{\partial x} + \frac{\partial}{\partial x}\left[\mu\left(2\frac{\partial u}{\partial x} - \frac{2}{3}div\vec{v}\right)\right] + \frac{\partial}{\partial y}\left[\mu\left(2\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)\right] + \frac{\partial}{\partial z}\left[\mu\left(2\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}\right)\right]$$

$$\rho\frac{Dv}{Dt} = f_y - \frac{\partial p}{\partial y} + \frac{\partial}{\partial y}\left[\mu\left(2\frac{\partial u}{\partial y} - \frac{2}{3}div\vec{v}\right)\right] + \frac{\partial}{\partial z}\left[\mu\left(2\frac{\partial u}{\partial z} + \frac{\partial w}{\partial y}\right)\right] + \frac{\partial}{\partial x}\left[\mu\left(2\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)\right]$$

$$\rho\frac{Dw}{Dt} = f_z - \frac{\partial p}{\partial z} + \frac{\partial}{\partial z}\left[\mu\left(2\frac{\partial w}{\partial z} - \frac{2}{3}div\vec{v}\right)\right] + \frac{\partial}{\partial x}\left[\mu\left(2\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}\right)\right] + \frac{\partial}{\partial y}\left[\mu\left(2\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right)\right]$$

Or more compactly they can be written as:

$$\rho\frac{D\vec{v}}{Dt} = \vec{f} - grad(p) + Div(\tau) \qquad \text{(i)}$$

In Lighthill's formulation the general equations of Navier-Stokes are expressed as a non-homogeneous wave equation (2):

$$\frac{1}{a_o^2}\frac{\partial^2 p}{\partial \tau^2} - \nabla^2 p = \frac{\partial^2 T_{ij}}{\partial y_i \partial y_j} + \frac{1}{a_o^2}\frac{\partial^2}{\partial t^2}(p - a_o^2\rho) \qquad \text{(ii)}$$

Where T_{ij} is the stress tensor with the viscous stresses represented as e_{ij}:

$$T_{ij} = \rho u_i u_j - e_{ij}$$

Solution of (2) for the far-field sound pressure can be obtained by using Green's theorem, thus, neglecting the viscous stresses, it can be expressed as the volume integral in the following form;

$$P_s = \frac{1}{4\pi R_{ob} a_o^2} \iiint \left[\frac{\partial^2}{\partial x_i \partial x_j} \{\rho u_i u_j\} + \frac{1}{a_o^2} \frac{\partial^2}{\partial t^2} \{p - a_o^2 \rho\} \right] dV \qquad \text{(iii)}$$

The second source term is usually neglected. The braces indicate in (iii) that the source term is estimated at the time that the various paths required for acoustic waves emanating from the distributed source to arrive at the same time t of the observer location. The retarded time is then expressed as:

$$t_s = t - |\overline{X} - \overline{Y}| / a_o$$

In addition, the spatial derivatives in (iii) can be effectively reflected with time derivative. Following Ref (1), the solution is written;

$$P_s = \frac{1}{4\pi R_{ob} a_o^2} \iiint \frac{\partial^2}{\partial t^2} (\rho C_r^2) r dr dx d\phi$$

$$t_r = t - \frac{R_{ob}}{a_o} + \frac{(x\cos\theta + r\sin\theta\cos\phi)}{a_o}$$

Where;

$$C_r = u\cos\theta + v\sin\theta\cos\phi$$

Thus, the form of the Lighthill integral reduces to:

$$I = C \left| \iint F(x, r, St) \exp(-2\pi i M_j St x \cos\theta) r dr dx \right|^2 \qquad (4)$$

Where:

$$C = \frac{\pi^4}{\left(R_{ob}/D\right)^2} St^4 \left(U_e/a_o\right)^8$$

$$F(x, r, St) = A_o J_o(\sigma) + A_1 J_1(\sigma) + A_2 J_2(\sigma)$$

$$A_o = F_{xx} \cos^2\theta + \frac{1}{2} F_{rr} \sin^2\theta$$

$$A_1 = F_{xr} \sin 2\theta$$

$$A_2 = \frac{1}{2} F_{rr} \sin^2\theta$$

In F(x, r, St), J is the Bessel function of:

$$\sigma = 2\pi St M r \sin\theta$$

3. FORMULATION OF THE ACOUSTIC ANALOGY EQUATIONS FOR JET NOISE PREDICTIONS

The formulation used for the current code starts as the original Lighthill's theory for the far-field pressure, given by:

$$P_s = \frac{1}{4\pi R_{ob} a_o^2} \iiint \left[\frac{\partial^2}{\partial x_i \partial x_j} \left\{ \rho u_i u_j \right\} + \frac{1}{a_o^2} \frac{\partial^2}{\partial t^2} \left\{ p - a_o^2 \rho \right\} \right] dV$$

The second term is again neglected (1), and since accounting for the retarded-time effect would require an excessive amount of computational storage, a compact source is assumed, yielding the form:

$$P_s(x_o, t) = \frac{1}{4\pi R_{ob} a_o^2} \iiint \left[\frac{\partial}{\partial t} (\rho C_r^2) \right]_{t_r} r dr dx d\phi$$

$$C_r = u \cos\theta + v \sin\theta \cos\phi$$

$$t_r = t - \frac{R_{ob}}{a_o} + \frac{(x \cos\theta + r \sin\theta \cos\phi)}{a_o}$$

Note that the compact sound source is located at $(x_o, r_o) = (0,0)$. Thus, it is placed at the center of the volume to be analyzed. Also, the retarded time can be written as:

$$t_r = t - \frac{|\vec{X} - \vec{Y}|}{a_o}$$

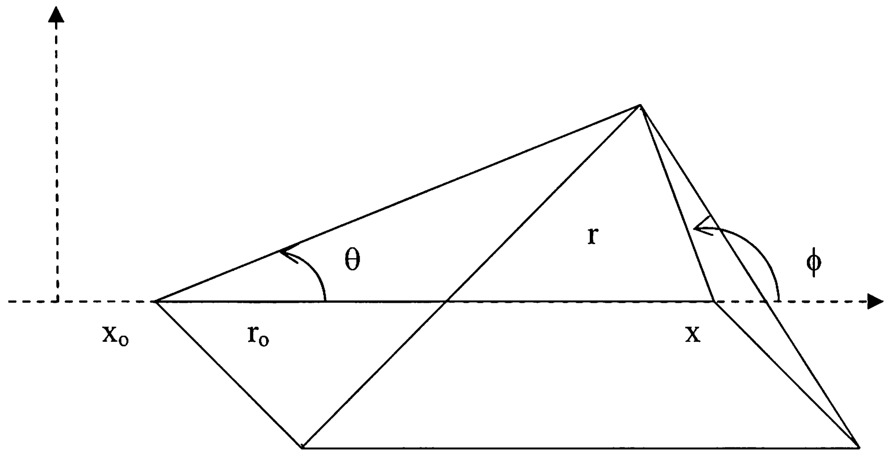

Figure 3.1. Compact source's location.

According to this the equations can be simplified further by assuming that:

$$R = |\vec{X} - \vec{Y}| = \sqrt{R_x^2 + R_r^2}$$

$$R_x = x - x_o ; R_r = r - r_o \cos(\phi_o - \phi)$$

Also it is known that:

$$\cos\theta = \frac{x - x_o}{\sqrt{(x - x_o)^2 + r^2}} \quad \text{and} \quad a^2 = r^2 + r_o^2 - 2rr_o \cos(\phi - \phi_o)$$

In any case, if $r_o = 0$ ➔ $a^2 = r^2$, $R_r = r$, then $R^2 = (x - x_o)^2 + r^2$

Then substituting all this information into the pressure equation yields:

$$p' = \frac{1}{4\pi a_o^2}\iiint \left[\frac{R_x^2}{R^3}\frac{\partial^2}{\partial t^2}(\rho u^2) + \frac{2R_xR_r}{R^3}\frac{\partial^2}{\partial t^2}(\rho uv) + \frac{R_r^2}{R^3}\frac{\partial^2}{\partial t^2}(\rho v^2) + \frac{1}{\rho}\frac{\partial^2}{\partial t^2}(p - a_o^2\rho) \right]_{t_r} rdrdxd\phi$$

$$= \frac{1}{4\pi a_o^2}\iiint \left[\cos^2\theta \frac{\partial^2}{\partial t^2}(\rho u^2) + \sin 2\theta \cos\phi \frac{\partial^2}{\partial t^2}(\rho uv) + \sin^2\theta \frac{\partial^2}{\partial t^2}(\rho v^2) \right]_{t_r} rdrdxd\phi$$

The velocities in this case are defined as: $u = U + u', v = v'$

Where; $u', v' << U$

Substituting the velocities into the pressure equation:

$$p' = \frac{1}{4\pi a_o^2}\iiint \left[\begin{array}{l} \cos^2\theta \frac{\partial^2}{\partial t^2}(U^2 + 2Uu' + u'^2) + \sin 2\theta \cos\phi \frac{\partial^2}{\partial t^2}(Uv' + u'v') \\ + \sin^2\theta \frac{\partial^2}{\partial t^2}(v'^2) \end{array} \right]_{t_r} rdrdxd\phi$$

$$= \frac{1}{4\pi a_o^2}\iiint \left[\cos^2\theta \frac{\partial^2}{\partial t^2}(2Uu') + \sin 2\theta \cos\phi \frac{\partial^2}{\partial t^2}(Uv') \right]_{t_r} rdrdxd\phi$$

If,

$$\left| \begin{array}{l} u' = \tilde{u}(x,r)e^{-\iota\omega t} \\ v' = \tilde{v}(x,r)e^{-\iota\omega t} \end{array} \right. \Rightarrow \text{in retarded coordinates:} \quad \left| \begin{array}{l} u_{ret}' = \tilde{u}(x,r)e^{-\iota\omega(t - \frac{R_{ob}}{a_o} + \frac{(x\cos\theta + r\sin\theta\cos\phi)}{a_o})} \\ v_{ret}' = \tilde{v}(x,r)e^{-\iota\omega(t - \frac{R_{ob}}{a_o} + \frac{(x\cos\theta + r\sin\theta\cos\phi)}{a_o})} \end{array} \right.$$

$$\begin{cases} \frac{\partial^2}{\partial t^2}(2Uu') = -2\omega^2 U\tilde{u}(x,r)e^{-\iota\omega t}e^{\iota\omega\frac{R}{a_o}}e^{-\iota\omega\frac{(x\cos\theta + r\sin\theta\cos\phi)}{a_o}} \\ \frac{\partial^2}{\partial t^2}(Uv') = -\omega^2 U\tilde{v}(x,r)e^{-\iota\omega t}e^{\iota\omega\frac{R}{a_o}}e^{-\iota\omega\frac{(x\cos\theta + r\sin\theta\cos\phi)}{a_o}} \end{cases}$$

Thus, $p' = \frac{-\rho_o \omega^2 U}{4\pi R a_o^2} \iiint (2\hat{u}\cos^2\theta + \hat{v}\cos\phi \sin 2\theta) e^{\imath\omega \frac{R}{a_o}} e^{-\imath\omega\left(\frac{x\cos\theta + r\sin\theta\cos\phi}{a_o}\right)} r dr dx d\phi$

Taking the Fourier conjugant: $\begin{cases} \hat{u} = \tilde{u}(x,r)e^{-\imath\omega t} \\ \hat{v} = \tilde{v}(x,r)e^{-\imath\omega t} \end{cases}$

Then,

$$p' = \frac{-\rho_o \omega^2}{4\pi R a_o^2} e^{\imath\omega\frac{R}{a_o}} \iint \int_0^{2\pi} \begin{Bmatrix} [2\hat{u}U\cos^2\theta][e^{-\imath\omega\frac{r\sin\theta\cos\phi}{a_o}}] + \\ [\hat{v}U\sin 2\theta][\cos\phi e^{-\imath\omega\frac{r\sin\theta\cos\phi}{a_o}}] \end{Bmatrix} e^{-\imath\omega\frac{\cos\theta}{a_o}} r dr dx d\phi$$

Consider the following for Bessel functions:

$$J_n(z) = \frac{i^{-n}}{\pi}\int_0^{\pi} e^{\imath z\cos\phi}\cos(n\phi)d\phi, \qquad \text{hence}$$

$\begin{Bmatrix} \alpha = \phi + \pi \\ \phi = \alpha - \pi \\ d\phi = d\alpha \end{Bmatrix}$ $\qquad \int_{\pi}^{2\pi} e^{\imath z\cos\alpha}\cos(n\alpha)d\alpha = \int_0^{\pi} e^{\imath z\cos(\phi+\pi)}\cos(n\phi + n\hat{u})d\phi$

$\cos(\phi+\pi) = -\cos(\phi)$
$\cos(n\phi + n\pi) = -\cos\phi, (n=1)$
$\cos(2\phi + 2\pi) = \cos 2\phi, (n=2)$

Then if n=0: $\int_{\pi}^{2\pi} e^{\imath z\cos\theta}d\alpha = \int_0^{\pi} e^{-\imath z\cos\theta}d\alpha$

This results in:

$$\int_0^{2\pi} e^{\imath z\cos\phi}d\phi = 2\int_0^{\pi} e^{\imath z\cos\phi}d\phi = 2\pi J_o(z) \leftarrow$$

Hence,

$$\int_0^{2\pi} e^{iz\cos\alpha}\cos(\alpha)d\alpha = \begin{Bmatrix} \alpha = \phi + \pi \\ \phi = \alpha - \pi \end{Bmatrix} = \int_0^{\pi} e^{i(-z)\cos\phi}(-\cos\phi)d\phi = -\int_0^{\pi} e^{i(-z)\cos\phi}\cos\phi d\phi = -\pi i J_1(z)$$

Resulting in:

$$\int_0^{2\pi} e^{iz\cos\phi}\cos\phi d\phi = 2\pi i J_1(z)$$

Thus, if $z = \dfrac{\omega}{a_o} r\sin\theta$

$$\int_0^{2\pi} e^{-iz\cos\phi}d\phi = 2\pi J_o(-z) = 2\pi J_o(z)$$
$$\int_0^{2\pi} e^{-iz\cos\phi}\cos\phi d\phi = 2\pi J_1(-z) = -2\pi J_1(z)$$

So that:

$$p' = -\frac{\rho_o\omega^2}{4\pi a_o^2 R} e^{i\omega\frac{R}{a_o}} \iint\left\{\left[2\tilde{u}U\cos^2\theta\right]2\pi J_0(z) + \left[U\tilde{u}\sin 2\theta\right](-2\pi i J_1(z))\right\} e^{-i\frac{\omega}{a_o}x\cos\theta} rdrdx$$

if $\begin{Bmatrix} A_o = 2\rho_o\tilde{u}U\cos^2\theta \\ A_1 = \rho_o\tilde{v}U\sin 2\theta \end{Bmatrix}$, then $p' = -\dfrac{e^{i\frac{\omega}{a_o}R}}{2R}\left(\dfrac{\omega}{a_o}\right)^2 \iint\left[A_o J_o(z) - iA_1 J_1(z)\right] e^{-i\frac{\omega}{a_0}x\cos\theta} rdrdx$

Note that R. Mankbadi, et al (1), using Lighthill's theory to obtain the far-field sound, derive the sound intensity in the far-field expressed as:

$$I = C \left| \iint F(x,r,St)\exp(-2\pi i M_j St x\cos\theta) rdrdx \right|^2$$

Since the pressure is the objective and it is known that:

$$|p| = \rho_o^2 a_o^4 \sqrt{I}$$

Then following Mankbadi, F(x, r, St) and C are defined as:

$$F(x,r,St) = A_o J_o(\sigma) - iA_1 J_1(\sigma) - A_2 J_2(\sigma)$$

$$C = \frac{\pi}{(R_{ob}/D)^2} St^4 (U_e/a_o)^8$$

In the derivation of this thesis, it is obtained;

$$F(x,r,St) = A_o J_o(\sigma) - iA_1 J_1(\sigma)$$

$$C = \frac{\pi}{(R_{ob}/D)^2} St^4 (U_e/a_o)^8$$

So that;

$$|p| = \rho_o^2 a_o^4 St^2 \left(\frac{U}{a_o}\right)^4 \frac{D}{R}\sqrt{\pi} \iint (A_o J_o(\sigma) - iA_1 J_1(\sigma)) \exp(-2\pi i M_j St x \cos\theta) r dr dx$$

At this point the objective of this dissertation was to develop the code needed to obtain the sound pressure distribution in the far-field using Lighthill's theory by applying the previous equation for sound intensity. This code develops the case for the point source, for which the pressure equation is given.

4. CODE STRUCTURE

The integral solver was developed using the FORTRAN programming language. The modular structure of the code is shown in figure 4.1.

In the code, the same complexity arises because of the highly oscillatory Bessel functions which must be solved for each point of the observer and then integrated to obtain the functions. Code for these is used from Numerical Recipes [11]. The program is divided into simpler blocks and the calculations that are performed in separate functions are:

- Angle of the observer location with respect to the source.
- Integration along the radius. This is the inner integral of the double integral that forms part of the solution.
- The calculations for each step of the integration.

Keeping track of the parameters of each point being calculated, the numerical solutions of the integrals, and the resolution of the Bessel functions, are all critical points to consider when developing the structure of the program. The use of COMMON blocks to keep track of variables throughout the program reduces the number of arguments passed to the functions.

The double integration, as shown in figure 4.2, is performed in two steps. For the inner integral, the Simpson method is used. The numerical algorithm is written as a function called from MAIN. The outer integral is calculated in MAIN itself using a loop to perform the counting, with small increment values used to increase the number of total steps for a constant-length Kirchhoff surface to improve accuracy.

In order to make the MAIN structure easy to see, a flow chart was constructed allowing the general structure to be analyzed. It is a simple program shown in figure 4.1. Embedded block structure simplifies the management of changes. This reduces the size of each module and makes debugging or addition of changes easier.

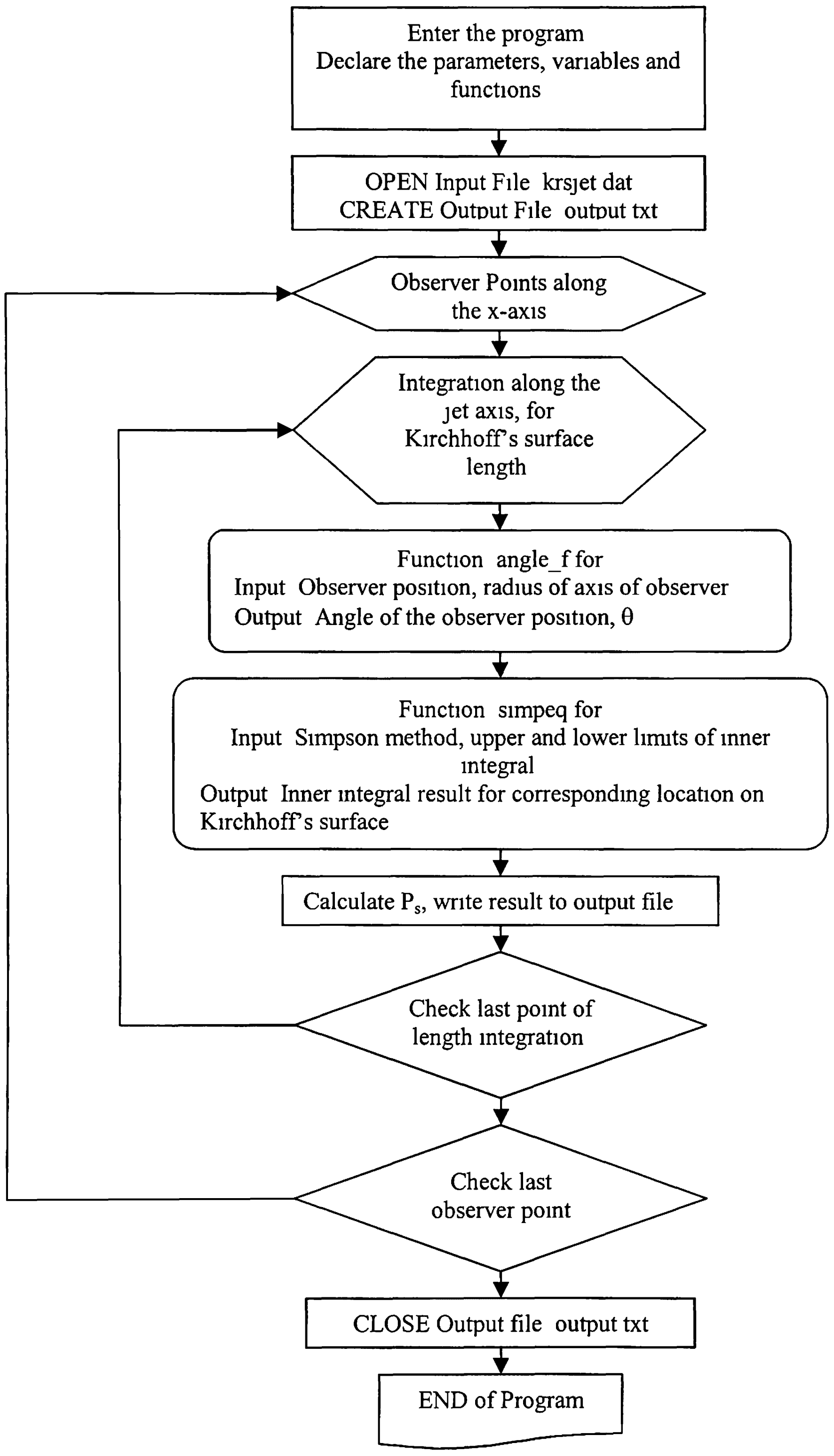
Enter the program
Declare the parameters, variables and functions
OPEN Input File krsjet dat
CREATE Output File output txt
Observer Points along the x-axis
Integration along the jet axis, for Kirchhoff's surface length
Function angle_f for
Input Observer position, radius of axis of observer
Output Angle of the observer position, θ
Function simpeq for
Input Simpson method, upper and lower limits of inner integral
Output Inner integral result for corresponding location on Kirchhoff's surface
Calculate P_s, write result to output file
Check last point of length integration
Check last observer point
CLOSE Output file output txt
END of Program

Figure 4.1. Flow chart of the Main body of the program.

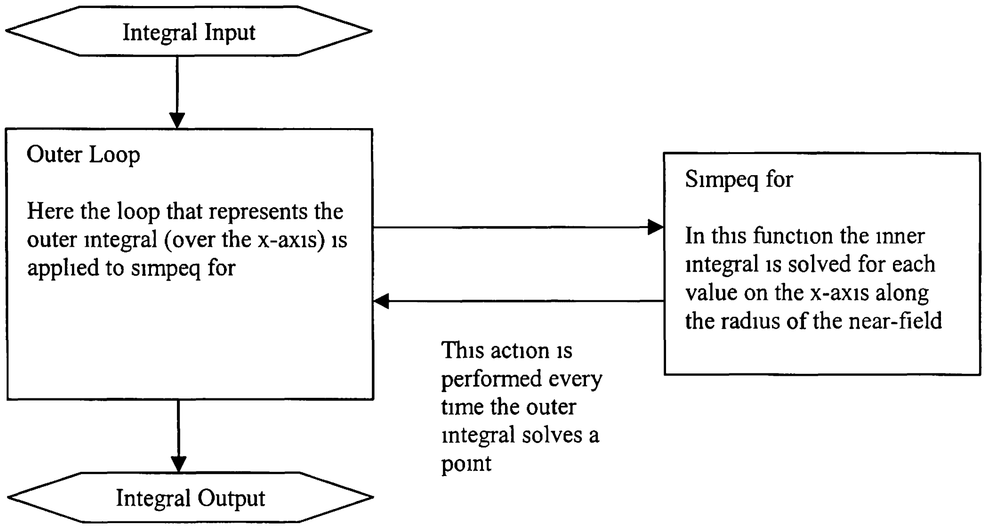
Integral Input
Outer Loop
Here the loop that represents the outer ıntegral (over the x-axıs) ıs applıed to sımpeq for
Sımpeq for
In thıs functıon the ınner ıntegral ıs solved for each value on the x-axıs along the radıus of the near-field
Thıs actıon ıs performed every tıme the outer ıntegral solves a poınt
Integral Output

Figure 4.2. Flow chart of double integration as it is performed by Main.

5. CODE VALIDATION

5.1. Analysis of the Far-field Sound Intensity Equation

In predicting the far-field noise from the near field, a Kirchhoff surface is used in order to enclose all the nonlinear effects and sound sources. The observer location is outside of the Kirchhoff surface, at an angle theta from the source and on the x-axis at a radius R_{ob} from the jet line, in the far field, as can be seen in the next figure:

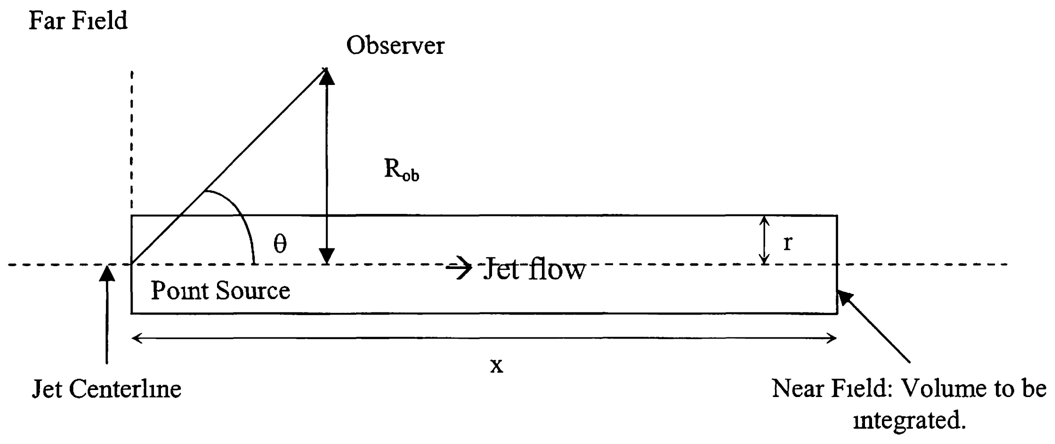

Figure 5.1.1. Kirchhoff surface for the far-field sound pressure calculations.

The objective is to find the sound pressure along points on the x-axis parallel to the jet line. The angle theta and the radius R_{ob} are changed only for the point being analyzed at the moment, and remain constant through the calculation of the pressure.

5.1.1 Code Testing Procedure

The main parameters and functions of the code are tested. They are plotted in the axis for which they are effective at revealing their behavior and any potential errors. The range used is the same that is used while running the code for the final calculations.

The argument of the Bessel function, sigma, is plotted and, since sigma is dependent only the change in r, the other variables remain constant. For this reason, the values for these variables are based on a particular case, for one point, of the final results. Sigma has been simplified to be dependent only on the angular frequency to speed of sound ratio, theta, and location along the radios. At an x-position at 30 points behind the jet and the radius of the observer R_{ob} of 10, theta is calculated. Therefore, plotting sigma as a function of r from the jet line to the radius of the Kirchhoff surface yields:

$$\sigma = \left(\frac{\omega}{a_o}\right) r \sin\theta$$

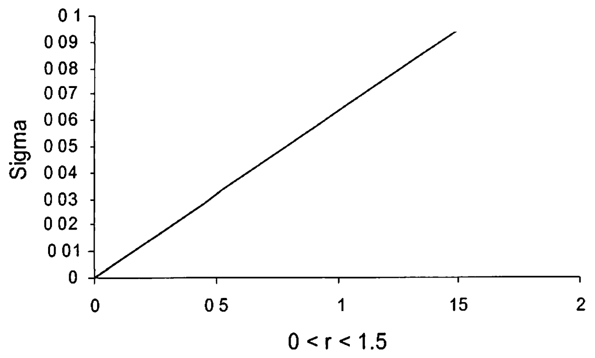

Figure 5.1.1.1. Plot of sigma for 0 < r < r Kirchhoff.

This argument is used in the Bessel functions of the first kind for the corresponding orders. Bessel functions of orders zero, one and two, are the functions used. It is solved by using the Bessel functions' code obtained from Numerical Recipes.

In order to find the solution for Lighthill the functions F_{xx}, F_{xr}, and F_{rr} which are to be found and used in the A_o, A_1, and A_2 correspondently. These functions are:

$$F_{xx} = u^2$$
$$F_{xr} = uv$$
$$F_{rr} = v^2$$

The testing of these functions is done to ensure that they are continuous and that the code does not leave any points out of it. Because the velocity of the flow along the radial axis is zero, F_{rr} turns out to be zero. Then, the functions A_o, and A_1 become:

$$A_o = 2\pi[u^2 \cos\theta]$$
$$A_1 = 2\pi uv \sin 2\theta$$

From this point onward the results obtained are used to calculate the component of the sound pressure integral.

5.2. Pressure Equation for the Monopole Point Source

The case in which the source of the sound is a monopole is now considered. This case represents a source in a near field used to calculate the pressures in the far field.

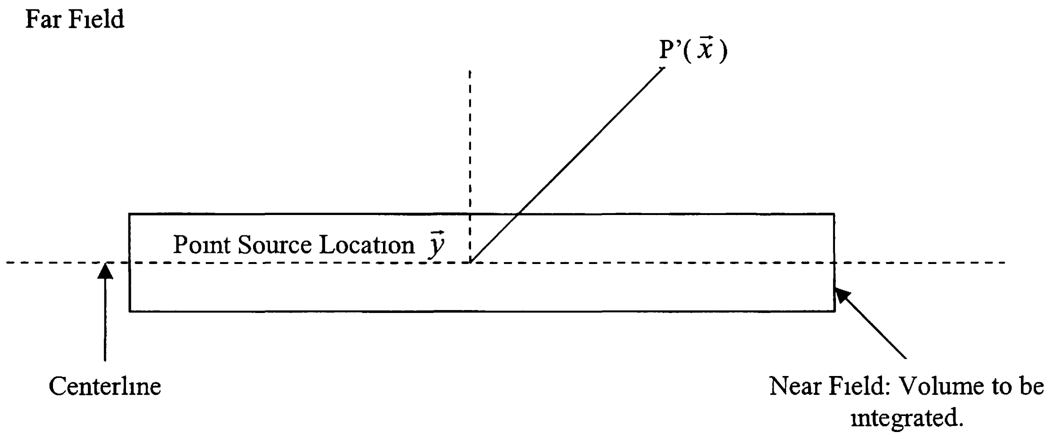
Far Field
P'($\vec{x}$)
Point Source Location $\vec{y}$
Centerline
Near Field: Volume to be integrated.

Figure 5.2.1. Monopole Point Source

First let us consider the wave equation for the source $Q(\vec{y},t)$ in this case which is:

$$\frac{\partial^2 \varphi}{\partial t^2} - a_o{}^2 \nabla^2 \varphi = \Box^2 \varphi = Q(\vec{y},t)$$

To solve this case, the Green function of the wave operator is obtained:

$$\frac{\partial^2 G}{\partial t^2} - a_o{}^2 \nabla^2 G = \delta(\vec{x})\delta(t)$$

Also knowing that:

$$\delta(x - x_o) = \begin{cases} \infty, x = x_o \\ 0, x \neq x_o \end{cases} \qquad \int_{-\infty}^{\infty} \delta(x - x_o) f(x) dx \equiv f(x_o)$$

$$G(\vec{x},t) = \frac{\delta(t - \frac{|\vec{x}|}{a_o})}{4\pi c^2 |\vec{x}|}$$

Then the solution of the wave equation for the source Q is:

$$\varphi(\vec{y},t) = \int_{\vec{x},t} G(\vec{x}-\vec{y}, t-\frac{|\vec{x}-\vec{y}|}{a_o})Q(\vec{x},\tau)d\vec{x}d\tau = \int \frac{\delta\left(t-\frac{|\vec{x}-\vec{y}|}{a_o}\right)}{4\pi a_o^2 |\vec{x}-\vec{y}|}Q(\vec{x},\tau)d\vec{x}d\tau$$

Since,

$$\int_{-\infty}^{\infty} \delta(t-\tau)Q(\vec{x},\tau)d\tau = Q(\vec{x},t-\tau)$$

Resulting in:

$$\varphi(\vec{y},t) = \frac{1}{4\pi a_o^2}\int \frac{Q(\vec{x},t-\frac{|\vec{x}-\vec{y}|}{a_o})}{|\vec{x}-\vec{y}|}d\vec{x}$$

Also, in the case of this point source,

$$Q(\vec{x},t) = Q_o e^{\imath\omega t}\delta(\vec{x})$$

Thus:

$$\varphi(\vec{x},t) = \frac{Q_o}{4\pi a_o^2}\int \frac{d\vec{x}}{|\vec{x}-\vec{y}|}e^{\imath\omega(t-\frac{|\vec{x}-\vec{y}|}{a_o})}\delta(x)$$

$$= \frac{Q_o}{4\pi a_o^2}\frac{e^{\imath\omega(t-\frac{|\vec{y}|}{a_o})}}{|\vec{y}|}$$ In this case, $|\vec{y}| = r$

Then φ becomes:

$$\varphi = \frac{Q_o}{4\pi a_o^2 r}e^{\imath\omega(t-\frac{r}{a_o})}$$

This allows finding the pressure and velocities because of the fact that:

$$P = -\rho_o\frac{\partial\varphi}{\partial t} \quad ; \qquad \hat{V} = \begin{pmatrix} u \\ - \\ v \end{pmatrix} = \nabla\varphi$$

So that for the pressure this yields:

$$P=-\rho_o\frac{\partial\varphi}{\partial t}=-\rho_o\frac{Q_o}{4\pi \mathrm{a}_o^2 r}i\omega e^{i\omega(t-\frac{r}{\mathrm{a}_o})}=\frac{P_s}{r}e^{i\omega(t-\frac{r}{\mathrm{a}_o})}$$

$$P_s=-\rho_o\frac{Q_o}{4\pi \mathrm{a}_o^2}i\omega$$

The velocities are needed to proceed with Lighthill's method.

$$\hat{V}=\nabla\varphi=\frac{\partial\varphi}{\partial r}=e^{i\omega(t-\frac{r}{\mathrm{a}_o})}\left\{-\frac{Q_o}{4\pi \mathrm{a}_o^2}-\frac{i\omega Q_o}{4\pi \mathrm{a}_o^3 r}\right\}=\frac{P_s}{\rho_o}\left(\frac{1}{\mathrm{a}_o r}-i\frac{1}{\omega r^2}\right)e^{i\omega(t-\frac{r}{\mathrm{a}_o})}$$

From here the velocities along the x and r axis are obtained. This is because the velocities come from:

$$u=V\cos\theta=V\frac{x}{r} \qquad v=V\sin\theta=V\frac{y}{r}$$

This yields the result

$$u=\frac{P_s}{\rho_o a_o}\left(\frac{x}{r^2}-i\frac{x}{\overline{\omega}r^3}\right)e^{i\omega(t-\frac{r}{a_o})}$$

$$v=\frac{P_s}{\rho_o a_o}\left(\frac{y}{r^2}-i\frac{y}{\overline{\omega}r^3}\right)e^{i\omega(t-\frac{r}{a_o})}$$

Where: $\overline{\omega}=\frac{\omega}{a_o}$

From here the next step is to find F_{xx}, F_{rx}, and F_{rr} and setup for their used in Lighthill's method. This case however, contains a singularity that will not allow for an accurate estimation of the sound pressure using the Lighthill integral. Because of this, code validations focused on the case of the sound source in the form of the Gaussian pulse.

5.3. Validation of the Pressure Equation for a Gaussian Pulse Point Source

We now examine the case of the sound sources represented by a Gaussian pulse. The pulse is centered at x=0.0 and decays quickly, with the decay controlled by a constant.

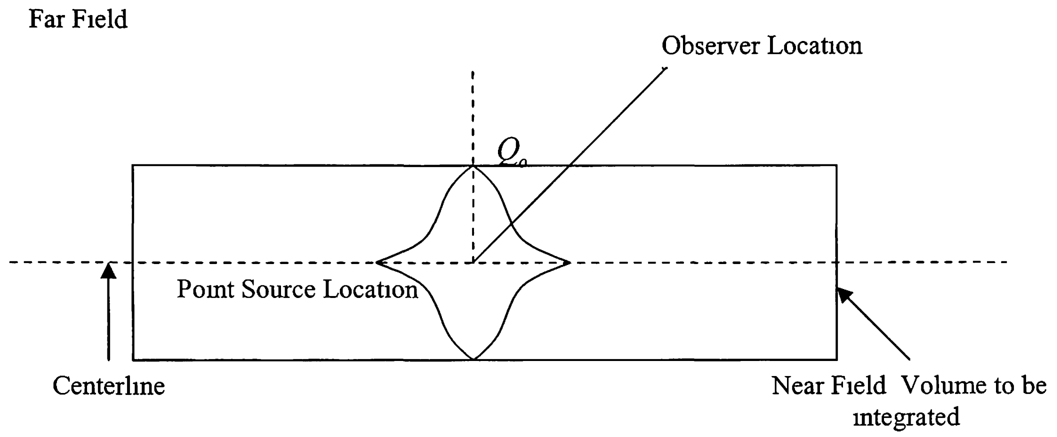

Figure 5.3.1. Gaussian Pulse Point Source

For the Gaussian pulse;

$$Q(\vec{x},t) = Q_o e^{\imath\omega t} e^{-\alpha r^2}$$

Then, for the perturbation velocity potential,

$$\varphi(\vec{y},t) = \int \frac{1}{4\pi a_o^2} \frac{Q(\vec{x}, t - \frac{|\vec{x}-\vec{y}|}{a_o})}{|\vec{x}-\vec{y}|} d\vec{x} = \frac{1}{4\pi a_o^2} \int \frac{Q_o e^{\imath\omega(t-\frac{|\vec{x}-\vec{y}|}{a_o})}}{|\vec{x}-\vec{y}|} e^{-\alpha|\vec{x}|^2} d\vec{x}$$

For the source:

$$\varphi(\vec{y}_s,t) = \int \frac{1}{4\pi a_o^2} \frac{Q_o}{|\vec{x}-\vec{y}_s|} e^{\imath\omega(t-\frac{|\vec{x}-\vec{y}_s|}{a_o})} e^{-\alpha|\vec{x}|^2} d\vec{x}$$

Considering,

$$\hat{V} = \nabla\varphi = \frac{\partial\varphi}{\partial r} \qquad \text{and} \qquad P = -\rho_o \frac{\partial\varphi}{\partial t}$$

For this case of 2D;

$$\varphi(x_s, y_s, t) = \int_0^{2\pi}\int_{-\infty}^{\infty}\int_0^{\infty} \frac{Q_o}{4\pi a_o^2} \frac{e^{i\omega(t - \frac{\sqrt{(x_s-x)^2+(y_s-y)^2}}{a_o}}}{\sqrt{(x_s-x)^2+(y_s-y)^2}} d\varphi(-e^{-\alpha\sqrt{x^2+y^2}})dxdy$$

$$= -2\pi \frac{Q_o}{4\pi a_o^2} \int_{-\infty}^{\infty}\int_0^{\infty} \frac{e^{i\omega(t - \frac{\sqrt{(x_s-x)^2+(y_s-y)^2}}{a_o}}}{\sqrt{(x_s-x)^2+(y_s-y)^2}} (-e^{-\alpha\sqrt{x^2+y^2}})dxdy \qquad (*)$$

To find the pressure this equation is differentiated with respect to time. Since the pressure is known to be:

$$P = -\rho_o \frac{Q_o}{4\pi c^2} \frac{\partial\varphi}{\partial t}$$

Where:

$$\frac{\partial\varphi}{\partial t} = 2\pi \frac{Q_0}{4\pi c^2} \iint \frac{i\omega e^{i\omega(t - \frac{\sqrt{(xs-x)^2+(ys-y)^2}}{c}} e^{-\alpha\sqrt{x^2+y^2}}}{\sqrt{(xs-x)^2+(ys-y)^2}} dxdy$$

This allows the pressure in far field to be calculated directly.

5.3.1. Lighthill Integral for Gaussian Pulse Point Source.

Considering Lighthill's equation in its volume integral form:

$$P = \frac{\rho_o}{4\pi R a_o^2} \iiint \left[\cos^2\theta \frac{\partial^2}{\partial t^2}(u^2) + \cos\varphi \sin 2\theta \frac{\partial^2}{\partial t^2}(uv) + \cos^2\varphi \sin^2\theta \frac{\partial^2}{\partial t^2}(v^2) \right]_{ret\ time} r\,dr\,dx\,d\varphi$$

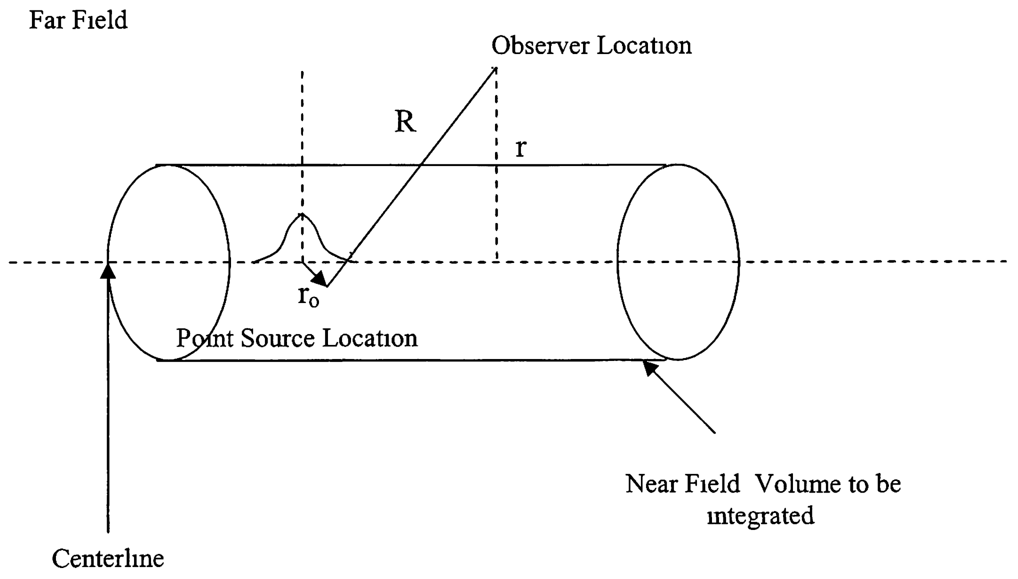

Figure 5.3.1.1. Lighthill case using a Gaussian Pulse.

The retarded time is given by:

$$t_{retarded} = t - \frac{R}{a_o} + \frac{x\cos\theta + r\sin\theta\cos\varphi}{a_o}$$

Also, from the figure it is seen that:

$$R^2 = (x - x_o)^2 + (r^2 - 2r_o r\cos(\varphi_o - \varphi) + r_o^2$$

The velocities are given by:

$$\begin{cases} u = u'(x,r)e^{-\imath\omega t} \\ v = v'(x,r)e^{-\imath\omega t} \end{cases} \quad \text{and in retarded time} \quad \begin{cases} u_{retarded} = u'(x,r)e^{-\imath\omega(t-\frac{R}{a_o}+\frac{x\cos\theta+r\sin\theta\cos\varphi}{a_o})} \\ v_{retarded} = v'(x,r)e^{-\imath\omega(t-\frac{R}{a_o}+\frac{x\cos\theta+r\sin\theta\cos\varphi}{a_o})} \end{cases}$$

Obtaining the derivatives of these velocities n retarded time:

$$\frac{\partial^2}{\partial t^2}(u^2) = \frac{\partial^2}{\partial t^2}(u'^2(x,r)e^{-2\imath\omega t_{retarded}})$$

$$= u'^2(x,r)\frac{\partial^2}{\partial t^2}e^{-2\imath\omega(t-\frac{R}{a_o}+\frac{x\cos\theta+r\sin\theta\cos\varphi}{a_o})},$$

$$\frac{\partial^2}{\partial t^2}u^2 = u'^2(x,r)(-4\omega^2)e^{-2\imath\omega(t-\frac{R}{a_o}+\frac{x\cos\theta+r\sin\theta\cos\varphi}{a_o})}$$

Similarly for the other derivatives:

$$\frac{\partial^2}{\partial t^2}uv = u'(x,r)v'(x,r)(-4\omega^2)e^{-2\imath\omega(t-\frac{R}{a_o}+\frac{x\cos\theta+r\sin\theta\cos\varphi}{a_o})}$$

$$\frac{\partial^2}{\partial t^2}v^2 = v'^2(x,r)(-4\omega^2)e^{-2\imath\omega(t-\frac{R}{a_o}+\frac{x\cos\theta+r\sin\theta\cos\varphi}{a_o})},$$

Substituting in the integral:

$$P = -\frac{4\rho_o\omega^2}{4\pi R a_o^2}\iiint\left[\cos^2\theta u^2 + \cos\varphi\sin 2\theta uv + \cos^2\varphi\sin^2\theta v^2\right]e^{2\imath\omega\frac{R}{a_o}}e^{-2\imath\omega(\frac{x\cos\theta+r\sin\theta\cos\varphi}{a_o})} rdrdxd\varphi$$

$$= -\frac{\rho_o\omega^2}{4\pi R a_o^2}e^{2\imath\omega\frac{R}{a_o}}\iiint\left[\begin{matrix} u^2\cos^2\theta e^{\frac{-2\imath\omega r\sin\theta}{a_o}\cos\varphi} + uv\cos\varphi\sin 2\theta e^{\frac{-2\imath\omega r\sin\theta}{a_o}\cos\varphi} \\ +v^2\cos^2\varphi\sin^2\theta e^{\frac{-2\imath\omega r\sin\theta}{a_o}\cos\varphi} \end{matrix}\right] e^{\frac{-2\imath\omega r\cos\theta}{a_o}} rdrdxd\varphi$$

Denoting

$$\sigma = -2i\bar{\omega} r \sin\theta \, ,$$

and using the representation

$$J_n(z) = \frac{i^{-n}}{\pi} \int_0^{\pi} e^{iz\cos\theta} \cos(n\varphi) d\varphi \ ,$$

The integral simplifies to

$$P = -\frac{\rho_o \bar{\omega}^2}{\pi R} e^{2i\bar{\omega}R} \iint \begin{pmatrix} 2\pi u^2 \cos^2\theta J_o(\sigma) + uv\sin 2\theta(-2\pi i J_1(\sigma)) \\ -v^2 \sin^2\theta(\pi J_0(\sigma) - \pi J_2(\sigma)) \end{pmatrix} e^{-2i\bar{\omega}x\cos\theta} r dr dx$$

Also, u and v come from the derivation of phi from the Gaussian pulse. They are:

$$u = \frac{\partial\varphi}{\partial x_s}, \qquad v = \frac{\partial\varphi}{\partial y_s}$$

Thus, using (* from section 5.3) by substituting we obtain for the source

$$\frac{\partial\varphi}{\partial x_s} = -\frac{Q_o}{2a_o^{\,2}} \int_{-\infty}^{\infty}\int_0^{\infty} \left[\frac{e^{i\frac{\omega}{a_o}\sqrt{(x_s-x)^2+(y_s-y)^2}} \, e^{\alpha\sqrt{x^2+y^2}}}{(x_s-x)^2+(y_s-y)^2} \frac{\omega}{a_o} i - \frac{e^{i\frac{\omega}{a_o}\sqrt{(x_s-x)^2+(y_s-y)^2}} \, e^{\alpha\sqrt{x^2+y^2}}}{\left[(x_s-x)^2+(y_s-y)^2\right]^{\frac{3}{2}}} \right] (x_s - x) dx dy$$

$$\frac{\partial\varphi}{\partial y_s} = -\frac{Q_o}{2a_o^{\,2}} \int_{-\infty}^{\infty}\int_0^{\infty} \left[\frac{e^{i\frac{\omega}{a_o}\sqrt{(x_s-x)^2+(y_s-y)^2}} \, e^{\alpha\sqrt{x^2+y^2}}}{(x_s-x)^2+(y_s-y)^2} \frac{\omega}{a_o} i - \frac{e^{i\frac{\omega}{a_o}\sqrt{(x_s-x)^2+(y_s-y)^2}} \, e^{\alpha\sqrt{x^2+y^2}}}{\left[(x_s-x)^2+(y_s-y)^2\right]^{\frac{3}{2}}} \right] (y_s - y) dx dy$$

This allows the pressure to be written the following way:

$$P = -\frac{\rho_o \bar{\omega}^2}{\pi R} e^{2i\bar{\omega}R} \iint \left(F(x, r, \omega) \right) e^{-2i\bar{\omega}x\cos\theta} r dr dx$$

Where,

$$F(x,r,\omega) = A_o J_o(\sigma) - iA_1 J_1(\sigma) - A_2 J_2(\sigma)$$

and A_o, A_1, and A_2 are written as:

$$A_o = 2\pi[u^2 \cos\theta + \frac{1}{2} v^2 \sin^2\theta]$$
$$A_1 = 2\pi uv \sin 2\theta$$
$$A_2 = \pi v^2 \sin^2\theta$$

Using this setup for Lighthill's equation follows the same pattern as the solution in (1). The program's structure allows for changes such as this to be made easily and altering the code does not represent a challenge, since all that is needed now is the velocities obtained from the Gaussian pulse point source equation.

6. RESULTS

With regards to code development to obtain results, all the steps necessary to put Lighthill's equation into code were taken. The flow chart in section 4 describes the structure of the program. The program is written so that each integral and functions are calculated in separate modules, permitting the user control over the calculations.

The objective is to develop a program to solve Lighthill's equation for the sound pressure at the far-field. The complete code is located in the appendix section. The output gives the pressure in the far field. The program produces a simple two-column output, where the magnitude of pressure in the far field is listed for its x-location. The input to the code the attempted tried test case is

Radius of the Kirchhoff Surface:	radius_kh =	1.5
Length of the Kirchhoff Surface:	length =	30
Omega/Velocity of Sound:	w_a_ratio =	1.131
Radius of the Observer:	r_ob =	10
Diameter of the Nozzle:	d =	1.0
Strouhal's number	St =	0.2

The output of the code was prepared for validation against results for:

$$P = -\rho_o \frac{Q_0}{2a_o^{\,2}} \iint \frac{i\omega e^{\imath\omega(t-\frac{\sqrt{(xs-x)^2+(ys-y)^2}}{a_o}} \quad e^{-\alpha\sqrt{x^2+y^2}}}{\sqrt{(xs-x)^2+(ys-y)^2}} dxdy$$

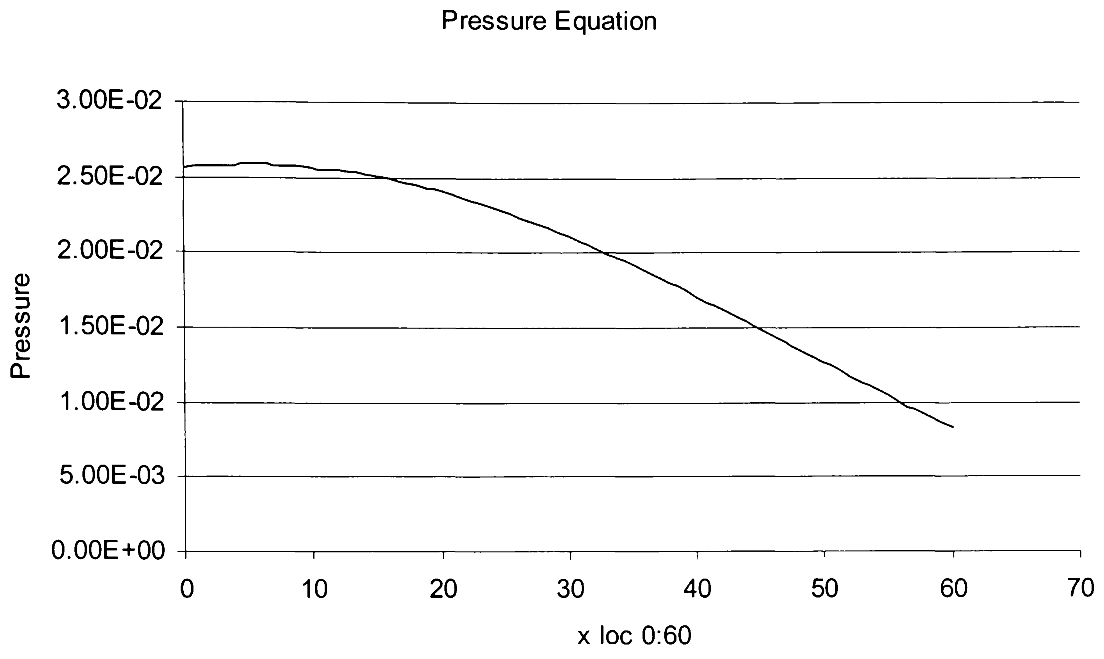

Figure 6.1. Output of the direct calculation for pressure.

The case of the plot obtained using Lighthill's equation, while assuming a Gaussian pulse point source is shown on Figure 6.2. It is evident that the plots are different not only in magnitude but also in behavior. This difference in the behavior of the curve shows that an error has occurred and it likely to be the singularity present throughout the equations.

Plotting the two curves together shows the difference in shape, note that the magnitude is not the same in the following but it serves as a comparison of their shape.

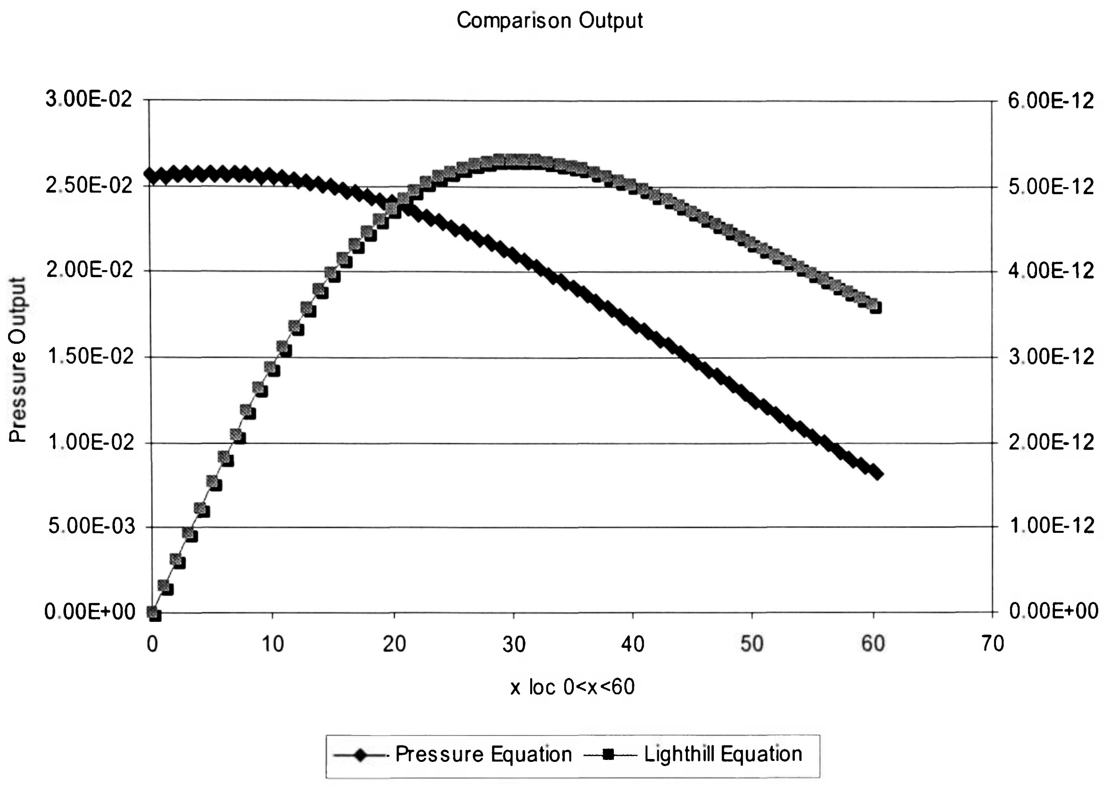

Figure 6.2. Output of the Lighthill's equation calculation for pressure in the far-field.

The general shape of the curve is expected, following an increase in pressure near the source and proceeding to decrease as the distance of the observer increases from the origin. The continuous curve suggests that there are no problems with the integration itself, but its form indicates an error in the calculations. This error is known to exist as the singularity is present in every interval calculated. The elimination of this singularity from the equations will help explore the validation of the code.

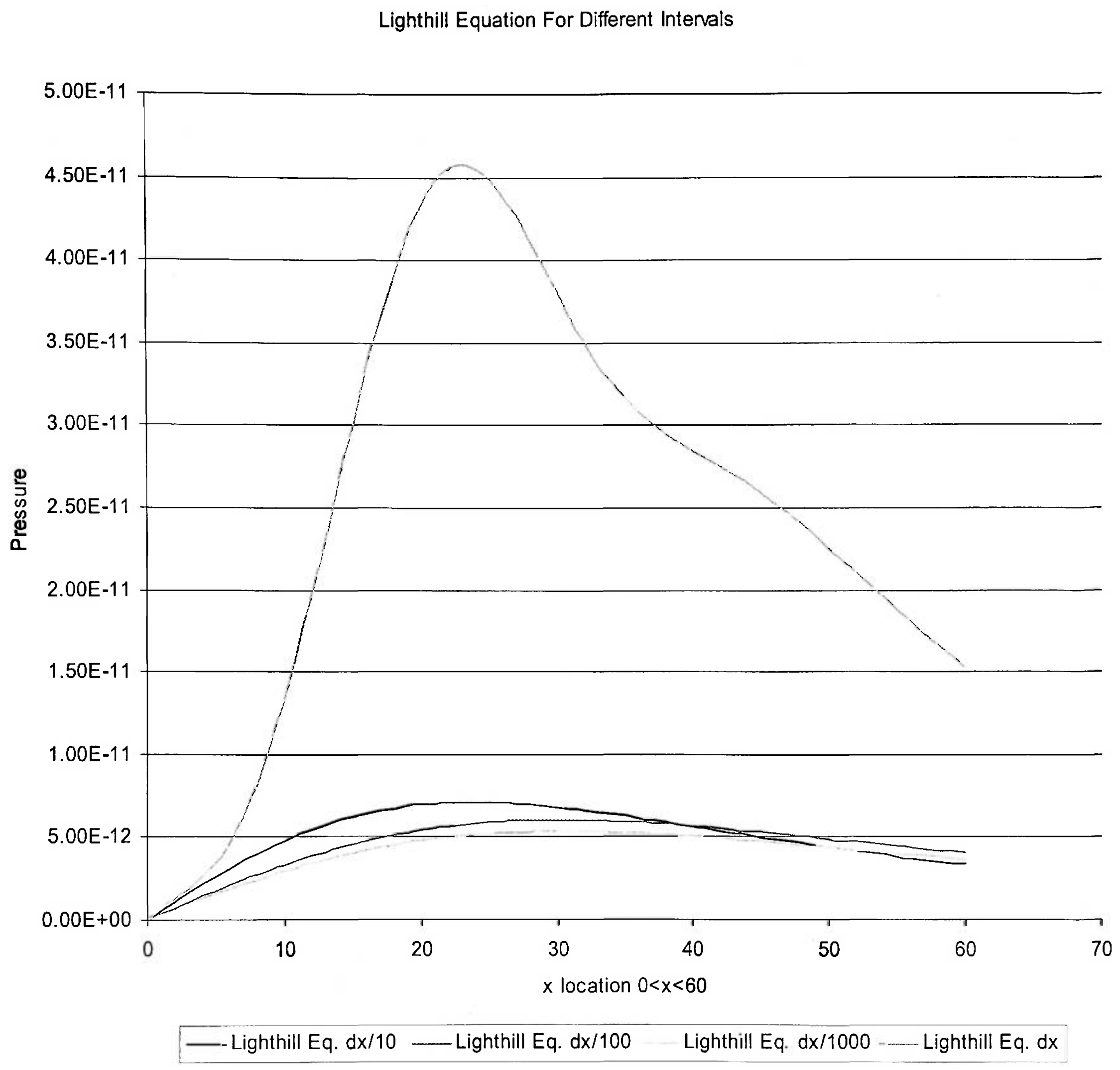

Figure 6.3. Plots at different interval sizes for Gaussian pulse point source.

This is the last test scenario with the highest practical accuracy, if the integration intervals are reduced in size to increase the number of steps, the results that were obtained were not better than the current plot. Therefore the significant increase in CPU usage was not worth the minimal improvement and did not change substantially the plotting of the curve.

When analyzing the shapes of the curves that were obtained using the Gaussian pulse it is important to state that they were considered post the study of the plots corresponding to the monopole point source. The original case was to consider a monopole source for both Lighthill's method and the pressure equation. Using the same parameters the plot was obtained and plotted with the pressure that would also be calculated assuming the monopole source.

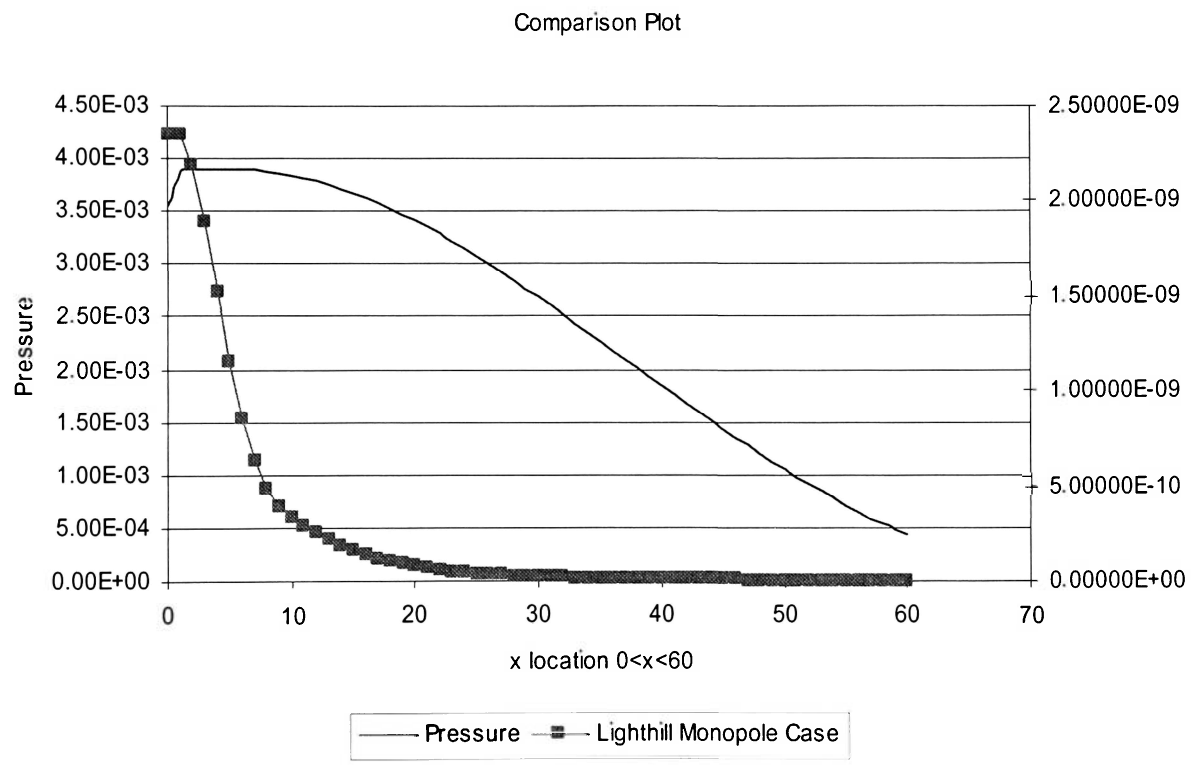

Figure 6.4. Plot of pressure & Lighthill's results for the monopole point source.

These plots show how the pressure equation output behaves in comparison to the Lighthill's equation output. Note the logical decay from the high pressure, near the origin of the point source, and the progressive decay as the observer is farther away from the source. The main problem is the way the curves are shaped, as the result from

Lighthill's solution does not follow the same pressure solution as it should. This difference in the way the curve decays is what led to the development of the equations assuming the Gaussian point source. In order to see if the error occurred only in the particular case in view, other cases were attempted.

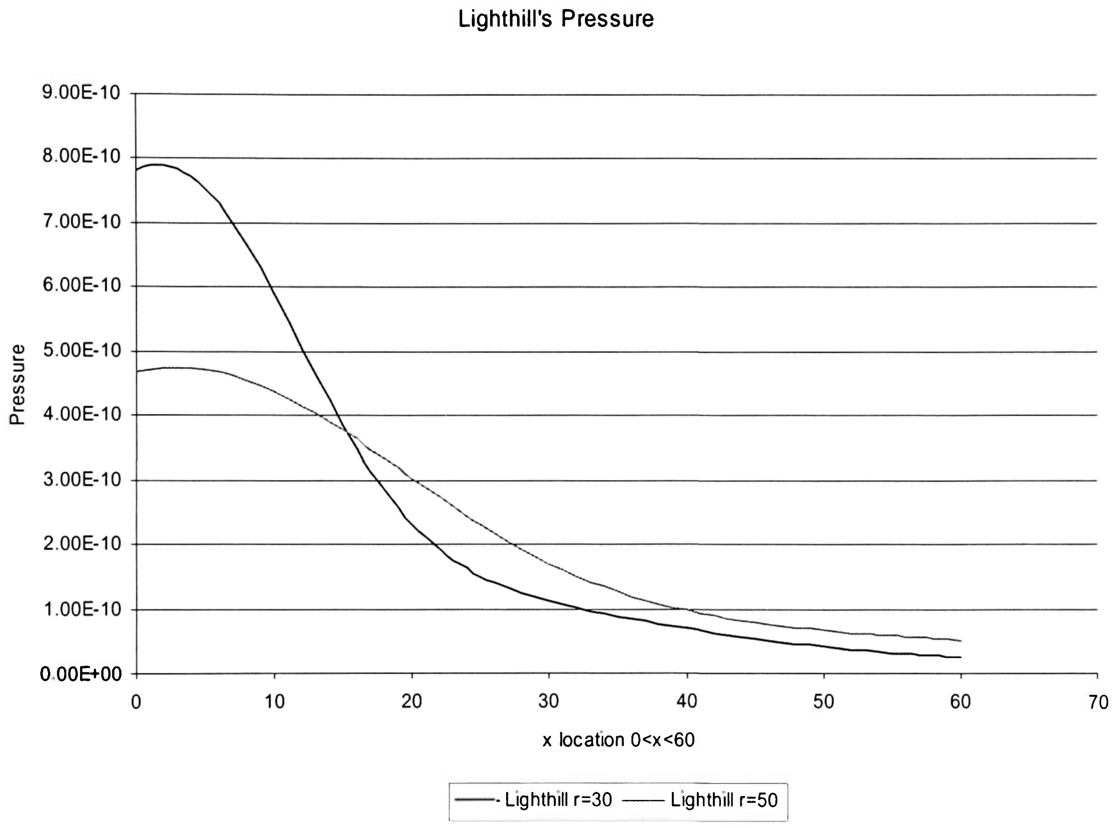

Figure 6.5. Lighthill's solution at different radius from the centerline.

While checking the results, plots were done at other distances from the centerline. This way it is possible to see that the results were consistent with the original Lighthill results, as their shape is similar and they reduce in magnitude as the observer sits farther away from the centerline. However, the difference in the shape from the pressure calculations remained throughout.

7. ANALYSIS AND DISCUSSION

The results showed a large discrepancy in the results between the Lighthill's integral solution and direct computation of pressure. Samples of the code were tested in order to find probable bugs and numerical accuracy was tested by decreasing the size of discreet integration steps. It did not improve the results while the CPU time increased significantly.

At the present, it may be suggested that the primary source of the discrepancy is the remaining singularity in the integral equation for the Gaussian-pulse sound source. Note that the singularity in the source was also the major problem when solving Lighthill's integral for a monopole sound source. By assuming the source to be a Gaussian pulse we attempted to reduce the effect of the singularity since only one point from a continuous distribution had to be carefully removed. However, apparently the error continued to persist in the integration of the pressure equation.

The method that was last used to reduce this error does not correct for it. It is recommended that the issues of proper integration path and singularity removal have to be further explored in future research. If such attempts fail, a new representation for the source equations that provide adequate code validation has to be sought.

8. CONCLUSIONS

This thesis project achieved the following objectives:

1. A numerical code has been developed to performed integration of Lighthill's acoustic analogy equation.
2. Two formulations for the sound source were examined to validate the code, corresponding to the monopole sound source and the Gaussian pulse sound source.
3. In the study of the monopole sound source a singularity occurred in the integration process that prevented from obtaining an accurate numeric solution.
4. In the case of the Gaussian-pulse sound source, we attempted to reduce the effect of the singularity since only one point from a continuous distribution has to be removed from the integration path. Still, the proper numerical solution was not obtained.
5. Recommendations for further research of the integral were formulated. It is suggested that, first, the issue of proper integration path to remove the singularity from the numerical solution has to be explored to provide for code validation. Alternatively, another formulation for the sound source is proposed to achieve the goal.

REFERENCES

1. Mankbadi, R. R., Hayer, M. E., and Povinelli, L. A., "Structure of Supersonic Jet Flow and its radiated Sound" AIAA Journal, Volume 32, No. 5, May 1994.

2. Shih, S. H., Hixon, D. R., Mankbadi, R. R., Pilon, A., and Lyrintzis, A., "Evaluation of Far-field Jet Noise Prediction Methods" AIAA Paper No. 97-0282, 6-9 January 1997, Reno, Nevada.

3. Schlichting, H., and Gersten, K. "Boundary-Layer Theory." 8th Edition, Springer, New York.

4. Data Provided by Tutor: V. Golubev. Fall 2003.

5. Golubev, V. V., Mankbadi, R. R., and Dahl, M. D., "Comparison of Several Approaches to Predict Noise Associated with Jet Acoustic Source Models" AIAA Paper No. 2003-3283, 12-14 May 2003, Hilton Head, South Carolina.

6. Golubev, V. V., Prieto, A. F., Mankbadi, R. R., Dahl, M. D., and Hixon, D. R., "Sound Radiated by a Wave-Like Structure in a Compressible Jet" AIAA Paper No. 2003-1063, 6-9 January 2003, Reno, Nevada.

7. Lyrintzis, A., "Some Recent Advances in the Use of Kirchhoff's Method in Computational Aeroacoustics" NCA- Volume 25, Proceedings of the ASME, Noise Control and Acoustics Division, ASME 1998.

8. Kreyszig, E., "Advanced Engineering Mathematics" 7th Edition, John Wiley & Sons, Inc., New York.

9. Articolo, G. A., "Partial Differential Equations & Boundary Value Problems with Maple V" Academic Press, 1998.

10. Pilon, A. "Development of Improved Surface Integral Methods for Jet Aeroacoustic Predictions" Thesis for University of Minnesota. January 1997.

11. Numerical Recipes. FORTRAN code source database for F77 and F90.

APPENDIX

LIGHTHILL' EQUATION CODE

MAIN.FOR

```
        PROGRAM maın

        PARAMETER (pı = 3 141592)

        INTEGER ıteratıon, ıter_count, ob_poınts, ınum
        INTEGER ı, ıı, ııı
        INTEGER IRULE

        REAL mach, radıous_kh, length_kh, dr, dx, w_a_ratıo, r_ob
        REAL x_pos, x_value, rad_ınt, rad, n, real_part, ı_part, absolut
        REAL n_lth
        REAL A, B, G, H
        REAL anglevalue, angle, ao
        REAL x_ıntegral, ıntensıty, St, d, pressure, R_d

        COMPLEX f, RESULT1, RESULT2, sımpeq, dummy, press_cplx

        EXTERNAL f, sımpeq

        COMMON /ANGLES/ anglevalue
        COMMON /POSITION/ x_pos
        COMMON /ASOUND/ ao
        COMMON /XINT/ x_ıntegral
        COMMON /STROUHAL/ St
        COMMON /DIAMETER/ d

c       Beggınıng of program body

        RESULT1 = (0 0,0 0)
        x_pos = 0 0
        x_value =0 0
        rad = 0 0
        ao = 340 3
        d = 1 0

        OPEN (UNIT=5, FILE='krsjet dat')
        OPEN (UNIT=6, FILE='output txt')

        DO 10 ı=1, 4
                READ (5,*)
10      CONTINUE
        READ (5,*) ıteratıon
        DO 20 ı=1, 10
                READ (5,*)
20      CONTINUE

        DO 3000 ıter_count=1, ıteratıon

        READ (5,*) ob_poınts, ınum, mach, radıous_kh, length_kh,
   &            dr, dx, w_a_ratıo
        WRITE (*,504) ob_poınts, mach, w_a_ratıo
504     FORMAT (5X, 'Observer Poınts', I3, /5X,
   &            'Mach', F10 5, /5X,
   &            'w/ao ratıo', F10 5)
        READ (5,*)

        r_ob = 50 0
```

```
            dummy = (0 0,1 0)

            CLOSE (UNIT=5)

            DO 1500 i=1, ob_points

            IF (i eq 1) THEN

c           Simpson method steps
            n = 30
c           Lower and upper limit in the x-direction
c           A = 0
c           B = 1500
c           Lower and upper limit in the r-direction
            G = 0 0
            H = 1 5

                    DO 900 ii = 0,60

                    x_pos=FLOAT(ii)
                    anglevalue = angle(x_pos, r_ob)

                    RESULT2 = (0 0,0 0)

                    n_lth = 0 0

c                   This loop represents the outer integral dx
c                   Remember to change the IF statements for each dx
                    DO 600 iii=0, 30

                    x_integral = n_lth
c                   FLOAT(iii)

                    RESULT1 = simpeq(n, G, H)

                    IF (iii eq 0) THEN
                            RESULT2 = RESULT2 + RESULT1
                    ELSE IF (iii eq 30) THEN
                            RESULT2 = RESULT2 + RESULT1
                    ELSE
                            RESULT2 = RESULT2 + RESULT1*1 0
                    END IF

                    n_lth = n_lth + 1 0

600                 CONTINUE

                    real_part = REAL(RESULT2)
                    i_part = CABS(RESULT2-real_part)
                    absolut = CABS(RESULT2)

                    R_d = SQRT(x_pos**2 + r_ob**2)

                    press_cplx = (-1 0/(3 14159*R_d))*(w_a_ratio**2)*
  &                         EXP(-2 0*dummy*w_a_ratio*R_d)*RESULT2

                    pressure = CABS(press_cplx)

                    WRITE (*,*) ii, pressure
                    WRITE (6,*) ii, pressure

900                 CONTINUE
```

```
      ELSE
            WRITE(*,*) 'Functionality not available'
      END IF
      CLOSE (UNIT=6)
1500  CONTINUE
3000  CONTINUE
      END
```

ANGLE_F.FOR

```
      REAL FUNCTION angle (x_pos, r_ob)

      REAL x_pos, r_ob
      INTEGER x

      x = INT(x_pos)

      IF (x eq 0) THEN
            angle = 3 1415902/2 0
      ELSE
            angle = atan(r_ob/x_pos)
      END IF

      RETURN
      END
```

SIMPEQ.FOR

```
      COMPLEX FUNCTION simpeq(n, a, b)

      REAL a, b, h, n, x
      COMPLEX sum, f
      EXTERNAL f

c     This is dr, the increment
      h = (b-a)/n

      x = a

      sum = f(x)

      DO i=1, n-2, 2

            x = x+h
            sum = sum + 4*(f(x))
            x = x+h
            sum = sum + 2*(f(x))

      END DO

      x=x+h
      sum = sum + 4*(f(x))

      sum = sum + (f(b))

      simpeq = (b -a)*sum/(3*n)

      END
```

F.FOR

```
      COMPLEX FUNCTION f(r)

      REAL x_pos, r, pi
      REAL iteration, inum
      REAL mach, radious_kh, length_kh, dr, dx, w_a_ratio
      REAL angle, sigma
      REAL d, COS, SIN
      REAL j0, j1, j2, bessj0, bessj1, bessj
      REAL St
      REAL x_integral

      COMPLEX a0, a1, a2, ft
      COMPLEX c_var, exp_part, fxx1, fxr1, frr1, fxx_f,fxr_f, frr_f

      EXTERNAL bessj0, bessj1, bessj
      EXTERNAL fxx_f,fxr_f,frr_f

      COMMON /ANGLES/ angle
      COMMON /POSITION/ x_pos
      COMMON /XINT/ x_integral
      COMMON /STROUHAL/ St
      COMMON /DIAMETER/ d
      COMMON /MNUMBER/ mach

      f=(0 0,0 0)
      pi = 3 14159
      St = 0 2

      OPEN (UNIT=7, FILE='krsjet dat')
      OPEN (UNIT=8, FILE='otheroutput txt')

      DO 10 i=1, 4
            READ (7,*)
10    CONTINUE
      READ (7,*) iteration
      DO 20 i=1, 10
            READ (7,*)
20    CONTINUE

      READ (7,*) ob_points, inum, mach, radious_kh, length_kh,
   &        dr, dx, w_a_ratio

      CLOSE(UNIT=7)

      c_var = (0 0,1 0)

      sigma=w_a_ratio*r*SIN(angle)

      j0 = bessj0(sigma)
      j1 = bessj1(sigma)
      j2 = bessj(2,sigma)

      fxx1 = fxx_f(x_integral, r, w_a_ratio)
      fxr1 = fxr_f(x_integral, r, w_a_ratio)
      frr1 = frr_f(x_integral, r, w_a_ratio)

      a0 = 2 0*fxx1*(COS(angle)**2)

      a1 = fxr1*SIN(2 0*angle)
      a2= 0 0
```

```
          ft = a0*ȷ0 - c_var*a1*ȷ1 - a2*ȷ2

          exp_part = EXP(-1 0*c_var*w_a_ratıo*x_ıntegral*COS(angle))

          f = exp_part*ft*r

          return
c End of Functıon
          END
```

BESSEL.FOR (from Numerical Recipes)

```
 FUNCTION bessȷ0(x)
 REAL bessȷ0,x
 REAL ax,xx,z
 DOUBLE PRECISION p1,p2,p3,p4,p5,q1,q2,q3,q4,q5,r1,r2,r3,r4,r5,r6,
*s1,s2,s3,s4,s5,s6,y
 SAVE p1,p2,p3,p4,p5,q1,q2,q3,q4,q5,r1,r2,r3,r4,r5,r6,s1,s2,s3,s4,
*s5,s6
 DATA p1,p2,p3,p4,p5/1 d0,- 1098628627d-2, 2734510407d-4,
*- 2073370639d-5, 2093887211d-6/, q1,q2,q3,q4,q5/- 1562499995d-1,
* 1430488765d-3,- 6911147651d-5, 7621095161d-6,- 934945152d-7/
 DATA r1,r2,r3,r4,r5,r6/57568490574 d0,-13362590354 d0,
*651619640 7d0,-11214424 18d0,77392 33017d0,-184 9052456d0/,s1,s2,
*s3,s4,s5,s6/57568490411 d0,1029532985 d0,9494680 718d0,
*59272 64853d0,267 8532712d0,1 d0/
 ıf(abs(x) lt 8 )then
   y=x**2
   bessȷ0=(r1+y*(r2+y*(r3+y*(r4+y*(r5+y*r6)))))/(s1+y*(s2+y*(s3+y*
*(s4+y*(s5+y*s6)))))

 else
   ax=abs(x)
   z=8 /ax
   y=z**2
   xx=ax- 785398164
   bessȷ0=sqrt( 636619772/ax)*(cos(xx)*(p1+y*(p2+y*(p3+y*(p4+y*
*p5))))-z*sın(xx)*(q1+y*(q2+y*(q3+y*(q4+y*q5)))))
 endıf
 return
 END

 FUNCTION bessȷ1(x)
 REAL bessȷ1,x
 REAL ax,xx,z
 DOUBLE PRECISION p1,p2,p3,p4,p5,q1,q2,q3,q4,q5,r1,r2,r3,r4,r5,r6,
*s1,s2,s3,s4,s5,s6,y
 SAVE p1,p2,p3,p4,p5,q1,q2,q3,q4,q5,r1,r2,r3,r4,r5,r6,s1,s2,s3,s4,
*s5,s6
 DATA r1,r2,r3,r4,r5,r6/72362614232 d0,-7895059235 d0,
*242396853 1d0,-2972611 439d0,15704 48260d0,-30 16036606d0/,s1,s2,
*s3,s4,s5,s6/144725228442 d0,2300535178 d0,18583304 74d0,
*99447 43394d0,376 9991397d0,1 d0/
 DATA p1,p2,p3,p4,p5/1 d0, 183105d-2,- 3516396496d-4,
* 2457520174d-5,- 240337019d-6/, q1,q2,q3,q4,q5/ 04687499995d0,
*- 2002690873d-3, 8449199096d-5,- 88228987d-6, 105787412d-6/
 ıf(abs(x) lt 8 )then
   y=x**2
   bessȷ1=x*(r1+y*(r2+y*(r3+y*(r4+y*(r5+y*r6)))))/(s1+y*(s2+y*(s3+
*y*(s4+y*(s5+y*s6)))))
```

```
      else
        ax=abs(x)
        z=8 /ax
        y=z**2
        xx=ax-2 356194491
        bessj1=sqrt( 636619772/ax)*(cos(xx)*(p1+y*(p2+y*(p3+y*(p4+y*
     *p5))))-z*sin(xx)*(q1+y*(q2+y*(q3+y*(q4+y*q5)))))*sign(1 ,x)
      endif
      return
      END
      FUNCTION bessj(n,x)
      INTEGER n,IACC
      REAL bessj,x,BIGNO,BIGNI
      PARAMETER (IACC=40,BIGNO=1 e10,BIGNI=1 e-10)
CU    USES bessj0,bessj1
      INTEGER j,jsum,m
      REAL ax,bj,bjm,bjp,sum,tox,bessj0,bessj1
      if(n lt 2)pause 'bad argument n in bessj'
      ax=abs(x)
      if(ax eq 0 )then
        bessj=0
      else if(ax gt float(n))then
        tox=2 /ax
        bjm=bessj0(ax)
        bj=bessj1(ax)
        do 11 j=1,n-1
          bjp=j*tox*bj-bjm
          bjm=bj
          bj=bjp
11      continue
        bessj=bj
      else

        tox=2 /ax
        m=2*((n+int(sqrt(float(IACC*n))))/2)
        bessj=0
        jsum=0
        sum=0
        bjp=0
        bj=1
        do 12 j=m,1,-1
          bjm=j*tox*bj-bjp
          bjp=bj
          bj=bjm
          if(abs(bj) gt BIGNO)then
            bj=bj*BIGNI
            bjp=bjp*BIGNI
            bessj=bessj*BIGNI
            sum=sum*BIGNI
          endif
          if(jsum ne 0)sum=sum+bj
          jsum=1-jsum
          if(j eq n)bessj=bjp
12      continue
        sum=2 *sum-bj

        bessj=bessj/sum
      endif
      if(x lt 0  and mod(n,2) eq 1)bessj=-bessj
      return
      END
```

F_FUNC.FOR

```
COMPLEX FUNCTION fxx_f (x, r, w_a_ratio)

COMPLEX          temp1, u_vel, u_func
REAL x, r
REAL w, ao, ross
REAL bb, mach

EXTERNAL u_func, v_func

COMMON /ASOUND/ ao
COMMON /MNUMBER/ mach

          w = w_a_ratio*ao

          temp1 = (0 0,1 0)

          u_vel = u_func(x,r,w_a_ratio)

          fxx_f= u_vel*u_vel

return
END

COMPLEX FUNCTION fxr_f (x, r, w_a_ratio)

COMPLEX          temp1, u_vel, v_vel, u_func, v_func
REAL x, r
REAL w, ao, ross
REAL bb, mach

EXTERNAL u_func, v_func

COMMON /ASOUND/ ao
COMMON /MNUMBER/ mach

          w = w_a_ratio*ao

          temp1 = (0 0,1 0)

          u_vel = u_func(x,r,w_a_ratio)
          v_vel = v_func(x,r,w_a_ratio)

          fxr_f= u_vel*v_vel

return
END

COMPLEX FUNCTION frr_f (x, r, w_a_ratio)

COMPLEX          temp1, v_vel, v_func
REAL x, r
REAL w, ao, ross
REAL bb, mach

EXTERNAL u_func, v_func

COMMON /ASOUND/ ao
COMMON /MNUMBER/ mach
```

```
          w = w_a_ratio*ao

          temp1 = (0 0,1 0)

          v_vel = v_func(x,r,w_a_ratio)

          frr_f= v_vel*v_vel

      return
      END
```

U_V_PHI.FOR

```
      COMPLEX FUNCTION u_func(x, r, w_a_ratio)

      COMPLEX temp1, num1, num2, phi
      COMPLEX r_phi_loop, x_phi_loop, gauss_f
      REAL dem1, num3, decay
      REAL x, r, x_phi, r_phi, dx_phi, dr_phi
      REAL Qo, ao, counter_x_phi, counter_r_phi
      INTEGER r_int_counter, x_int_counter, c_r_max, c_x_max

      COMMON /ASOUND/ ao

      temp1 = (0 0,1 0)
      r_phi_loop = (0 0,0 0)
      x_phi_loop = (0 0,0 0)
      dx_phi = 0 1
      dr_phi = 0 1
      decay = -0 0001
      Qo = 1 0

      counter_r_phi = -50 0
      r_int_counter = 0

      c_r_max = 100
      c_x_max = 100
      DO i=0, c_r_max
          r_phi = counter_r_phi
          r_int_counter = i

          counter_x_phi = 0 0
          x_int_counter = 0
c -------------------------inner int------------------------------
          DO ii=0, c_x_max
              x_phi = counter_x_phi
              x_int_counter = ii

              dem1 = ( (x-x_phi)**2 + (r-r_phi)**2 )
              num1 = ((temp1*w_a_ratio/dem1)-1 0/(dem1**(3 0/2 0)))
              num2 = EXP(temp1*w_a_ratio*SQRT(dem1))
              num3 = EXP(decay*SQRT(x_phi**2+r_phi**2))
              gauss_f = num1*num2*num3*(x-x_phi)

              IF (x_int_counter eq 0) THEN
                  x_phi_loop = x_phi_loop + gauss_f
              ELSE IF (x_int_counter eq c_x_max) THEN
                  x_phi_loop = x_phi_loop + gauss_f
              ELSE
                  x_phi_loop = x_phi_loop + gauss_f*dx_phi
```

```
                        END IF

                        counter_x_phi = counter_x_phi +  dx_phi
                END DO
c ------------------------inner int------------------------------
                IF (r_int_counter eq 0) THEN
                        r_phi_loop = r_phi_ploop + x_phi_loop
                ELSE IF (r_int_counter eq c_r_max) THEN
                        r_phi_loop = r_phi_loop + x_phi_loop
                ELSE
                        r_phi_loop = r_phi_loop + x_phi_loop*dr_phi
                END IF

                counter_r_phi = counter_r_phi + dr_phi
        END DO

        u_func = -1 0*Qo*r_phi_loop/(2 0*(ao**2))

        return
        END

c ----------------------------------------------------------------------- c
c _____________________________ v function ______________________________ c

        COMPLEX FUNCTION v_func(x, r, w_a_ratio)

        COMPLEX temp1, num1, num2, phi
        COMPLEX r_phi_loop, x_phi_loop, gauss_f
        REAL dem1, num3, decay
        REAL x, r, x_phi, r_phi, dx_phi, dr_phi
        REAL Qo, ao, counter_x_phi, counter_r_phi
        INTEGER r_int_counter, x_int_counter, c_r_max, c_x_max

        COMMON /ASOUND/ ao

        temp1 = (0 0,1 0)
        r_phi_loop = (0 0,0 0)
        x_phi_loop = (0 0,0 0)
        dx_phi = 0 1
        dr_phi = 0 1
        decay = -0 0001
        Qo = 1 0

        counter_r_phi = -50 0
        r_int_counter = 0

        c_r_max = 100
        c_x_max = 100
        DO i=0, c_r_max
                r_phi = counter_r_phi
                r_int_counter = i

                counter_x_phi = 0 0
                x_int_counter = 0
c ------------------------inner int------------------------------
                DO ii=0, c_x_max
                        x_phi = counter_x_phi
                        x_int_counter = ii

                        dem1 = ( (x-x_phi)**2 + (r-r_phi)**2 )
                        num1 = ((temp1*w_a_ratio/dem1)-1 0/(dem1**(3 0/2 0)))
                        num2 = EXP(temp1*w_a_ratio*SQRT(dem1))
```

```
                              num3 = EXP(decay*SQRT(x_phi**2+r_phi**2))
                              gauss_f = num1*num2*num3*(r-r_phi)

                              IF (x_int_counter eq 0) THEN
                                        x_phi_loop = x_phi_loop + gauss_f
                              ELSE IF (x_int_counter eq c_x_max) THEN
                                        x_phi_loop = x_phi_loop + gauss_f
                              ELSE
                                        x_phi_loop = x_phi_loop + gauss_f*dx_phi
                              END IF

                              counter_x_phi = counter_x_phi +  dx_phi
                    END DO
c ------------------------inner int------------------------------
                    IF (r_int_counter eq 0) THEN
                              r_phi_loop = r_phi_ploop + x_phi_loop
                    ELSE IF (r_int_counter eq c_r_max) THEN
                              r_phi_loop = r_phi_loop + x_phi_loop
                    ELSE
                              r_phi_loop = r_phi_loop + x_phi_loop*dr_phi
                    END IF

                    counter_r_phi = counter_r_phi + dr_phi
          END DO

          v_func = -1 0*Qo*r_phi_loop/(2 0*(ao**2))
          return
          END
```

KRSJET.DAT
SOURCE FILE

```
*
*
*number of iterations (with different data)
*
1
*
*number of observers
*type of derivatives (inum=0 => analytical, inum=3 => all numerical)
*Mach number of Kirchhoff surface
*radius of Kirchhoff surface
*length of Kirchhoff surface
*Dr, Dx for numerical normal derivatives
*omega/Vsound
*
*
1,0,1 2d0,0 555d0,1500 d0,0 d0,0 d0,1 131d0
*r, theta for each observer position
10 ,90
50 ,15
50 ,20
50 ,25
50 ,30
50 ,35
50 ,40
50 ,45
50 ,50
50 ,55
50 ,60
50 ,65
```

```
50.,70.
50.,75.
50.,80.
50.,85.
50.,90.
1,0,0.d0,5.d0,150.d0,0.d0,0.d0,1.d0
```

PRESSURE EQUATION CODE

MAIN.FOR

```
PROGRAM Main

REAL x, press, w
REAL b, ao

COMPLEX p_1, p_2, cplx, p_cplx
COMPLEX temp1, num1, num2, phi
COMPLEX r_phi_loop, x_phi_loop, gauss_f
REAL dem1, num3, decay
REAL r, x_phi, r_phi, dx_phi, dr_phi
REAL Qo, counter_x_phi, counter_r_phi
INTEGER r_int_counter, x_int_counter, c_r_max, c_x_max

temp1 = (0 0,1 0)
ao = 343 0

OPEN(UNIT=7,FILE='pressplot txt')

w_a_ratio = 1 131

r = 50 0

DO jj=0, 60

    r_phi_loop = (0 0,0 0)
    x_phi_loop = (0 0,0 0)

    decay = -0 0001
    Qo = 1 0
    x = FLOAT(jj)

    c_r_max = 200
    c_x_max = 100
    dx_phi = 0 1
    dr_phi = 0 1
    counter_r_phi = -100 0

    r_int_counter = 0

    DO i=0, c_r_max
        r_phi = counter_r_phi
        r_int_counter = i

        counter_x_phi = dx_phi
        x_int_counter = 0

        DO ii=0, c_x_max
            x_phi = counter_x_phi
            x_int_counter = ii

            dem1 = ((x-x_phi)**2+(r-r_phi)**2)
            num1 = 1 0/SQRT(dem1)
            num2 = EXP(temp1*w_a_ratio*SQRT(dem1))
            num3 = EXP(decay*SQRT(x_phi**2+r_phi**2))
            gauss_f = num1*num2*num3*temp1*w_a_ratio

            IF (x_int_counter eq 0) THEN
```

```
                                x_phi_loop = x_phi_loop + gauss_f
                        ELSE IF (x_int_counter eq c_x_max) THEN
                                x_phi_loop = x_phi_loop + gauss_f
                        ELSE
                                x_phi_loop = x_phi_loop + gauss_f*dx_phi
                        END IF

                        counter_x_phi = counter_x_phi +  dx_phi
                END DO

                IF (r_int_counter eq 0) THEN
                        r_phi_loop = r_phi_ploop + x_phi_loop
                ELSE IF (r_int_counter eq c_r_max) THEN
                        r_phi_loop = r_phi_loop + x_phi_loop
                ELSE
                        r_phi_loop = r_phi_loop + x_phi_loop*dr_phi
                END IF

                counter_r_phi = counter_r_phi + dr_phi

         END DO

        p_cplx=-1 0*Qo*r_phi_loop/(2 0*ao)

        press = ABS(p_cplx)

        WRITE(7,*) jj, press
        WRITE(*,*) jj, press

END DO

CLOSE(UNIT=7)

END
```