NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

JAVA-BASED IMPLEMENTATION OF MONTEREY-MIAMI PARABOLIC EQUATION (MMPE) MODEL WITH ENHANCED VISUALIZATION AND IMPROVED METHOD OF ENVIRONMENTAL DEFINITION

by

Ha, Yonghoon

December 2000

Thesis Advisors: Don Brutzman
Kevin B. Smith

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE December 2000	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE
Java-Based Implementation of Monterey-Miami Parabolic Equation (MMPE) Model with Enhanced Visualization and Improved Method of Environmental Definition

5. FUNDING NUMBERS

6. AUTHOR HA, YONGHOON

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT

The Monterey-Miami Parabolic Equation (MMPE) Model is a full-wave underwater acoustic propagation model that utilizes the split-step Fourier marching algorithm. Previously the MMPE model was implemented in Fortran language and ran with a simple command line interface either in a Unix or DOS command window. After the Fortran code was run, the resulting binary data output file was post-processed using Matlab routines to extract specific field data and present the results in graphical form. This approach requires the user to have installed both Matlab and Fortran compilers. The MMPE model and associated acoustic processing tools are now rewritten in the object-oriented language Java. This new version of the MMPE model built within a Windows framework is called WinMMPE. Integrating the model, the post-processing calculations and the graphics generation together with a graphic user interface has produced a more attractive tool for users. A user-friendly, efficient, and accurate full-wave acoustic propagation model with enhanced visualization can make it easier to assess the spatial transmission loss in the underwater acoustic environment.

14. SUBJECT TERMS: underwater acoustic propagation, acoustic modeling, Java, parabolic equation, MMPE, WinMMPE, VRML, 3D, sonar visualization

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

Approved for public release; distribution is unlimited.

JAVA-BASED IMPLEMENTATION OF MONTEREY-MIAMI PARABOLIC EQUATION (MMPE) MODEL WITH ENHANCED VISUALIZATION AND IMPROVED METHOD OF ENVIRONMENTAL DEFINITION

Ha, Yonghoon
Lieutenant, Republic of Korea Navy
B.S., Republic of Korea Naval Academy, 1994
B.S., Seoul National University, 1997

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ENGINEERING ACOUSTICS

from the

NAVAL POSTGRADUATE SCHOOL

December 2000

Author: _____

Ha, Yonghoon

Approved by: _____

Don Brutzman, Thesis Advisor

_____

Kevin B. Smith, Thesis Advisor and
Chair, Engineering Acoustics Academic Committee

ABSTRACT

The Monterey-Miami Parabolic Equation (MMPE) Model is a full-wave underwater acoustic propagation model that utilizes the split-step Fourier marching algorithm. Previously the MMPE model was implemented in Fortran language and ran with a simple command line interface either in a Unix or DOS command window. After the Fortran code was run, the resulting binary data output file was post-processed using Matlab routines to extract specific field data and present the results in graphical form. This approach requires the user to have installed both Matlab and Fortran compilers. The MMPE model and associated acoustic processing tools are now rewritten in the object-oriented language Java. This new version of the MMPE model built within a Windows framework is called WinMMPE. Integrating the model, the post-processing calculations and the graphics generation together with a graphic user interface has produced a more attractive tool for users. A user-friendly, efficient, and accurate full-wave acoustic propagation model with enhanced visualization can make it easier to assess the spatial transmission loss in the underwater acoustic environment.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

Two years have been passed in the Naval Postgraduate School. Thank God! I have survived. Now a thousand emotions crowd my mind. Much more regret crosses over me, rather than satisfaction. However, I cherish all of my endeavors in school and I am grateful to earn a M.Sc. in Engineering Acoustics at NPS. Whenever I had difficulties, many people helped me out and cheered me up.

First of all, I would like to thank my mother, Jul-Ja Park. Even though she stayed in Korea, her heart has been with me. She has always said, "I believe you. You can make it!" Sometimes these words were terrifying, but they were making me do all the best.

Assistant Professor Don Brutzman and Associate Professor Kevin B. Smith, my thesis advisors, pulled me out to the most interesting area, "Virtual World" with Java and VRML, and underwater acoustic models. They taught me a lot of theory and always encouraged me to program a new module just as I was completing the last one. To be honest, it was hard for me to keep up with them. I hope that my advisors will find another student to continue developing my thesis program, WinMMPE. My thanks must go to them.

Don Mcgregor, a faculty member of "Modeling and Virtual Environment Simulation" group, was an instructor of Java class and network communication class. Whenever I had difficulty in Java programming, he gave me good advice. He is so kind to me. I thank him very much. Also I thank Sang-Kyu Han, a visiting researcher, for good advice about underwater acoustics.

My life in Monterey started with these friends, Amy, Anne, Lizann, Sharna, Brad, and Brian. I cannot imagine anything in Monterey for two years without them. If it were not for them, I would only have had schoolwork during that period. They let me know young American culture and what the memory of international friendship. With all my heart, I give thanks to my friends.

I. INTRODUCTION

The parabolic equation method for predicting underwater acoustic propagation was introduced to the underwater acoustics community by Tappert (1977). The parabolic equation (PE) is a one-way wave equation approximation to the full Helmholtz wave equation that defines the continuous-wave (CW), single frequency acoustic field in general media. Because the PE is still a full-wave equation, it includes all of the full-wave effects (e.g., diffraction) that are neglected in more rudimentary ray-based models. It is also well suited to handle range-dependent media as it naturally computes the coupling between vertical normal modes of the waveguide. Additionally, the one-way formulation of the PE allows for implementation with highly efficient numerical algorithms. The most common methods are the split-step Fourier (PE/SSF) method (Hardin and Tappert, 1973), the implicit finite difference (IFD-PE) method (Lee, et al., 1981), and the finite element (FEPE) method (Collins, 1990).

Of these methods, the split-step Fourier method was adopted in developing the University of Miami Parabolic Equation (UMPE) Model. The UMPE Model was implemented by Smith and Tappert (1993). Since Prof. Smith moved to Monterey in 1995, the UMPE Model was upgraded to incorporate some improved numerical techniques and rewritten. This new version was then named the Monterey-Miami Parabolic Equation (MMPE) Model.

The MMPE Model was written in Fortran and implemented using a command line structure available in either a Unix or DOS command window. A Fortran compiler is required to create the executable program. Its output file is a binary data file composed of the complex values of the acoustic field function for various grid points in range, depth, and azimuth for multiple frequencies, in general. From this data, acoustic pressure and transmission loss can be extracted for a variety of simulated receiver locations within the computational grid. This post-processing of the binary data file and the subsequent graphical manipulation of the results are performed using Matlab.

Many underwater acoustic models are written in Fortran language. The compiled executable files run and output data files of the acoustic prediction are produced. This is then processed using a graphics package to display the results of interest. Such command line interface programs have little advantage in Windows operating systems. First of all, users have to handle the programs separately. Most users would probably prefer using an integrated software package because it is easier to handle. Second, the program's command line interface and the Matlab command window are text based. Users are required to type text in the command line. It is not a user-friendly computing environment. Third, in standard Fortran the size of data arrays must be declared before they are used. Developers must always consider the maximum array size that may be needed. This can sometimes make the programs waste unnecessary memory depending on some input data, or users might want to use more input data than the maximum declared. Fourth, the Fortran program is a stand-alone program. It is implemented in an independent computing environment. It is not always easy to combine with other software or systems, and it requires a Fortran compiler (not standard on all computer systems).

To address these issues, the MMPE model must be re-programmed. It should be an object-oriented program, and the propagation model and post-processing graphical program should be integrated as one. It should have a new graphic user interface (GUI) and should be convertible to any kind of system. For these reasons, the choice for the programming language is Java. Java is an object-oriented language and network language. Java can dynamically create the data array size because it treats a run time object system. It is also possible to be multi-threaded, which will allow it to run several tasks in parallel within the same program. Lately, the Java language has become very popular among many programmers since it was announced in 1995. It is open-source and freely served by Sun Micro Systems Corporation. More and more class libraries have been and continue to be developed.

The main goal of this thesis is to develop a user-friendly, efficient, and accurate full-wave acoustic propagation model with enhanced visualization. This will be

accomplished by integrating the MMPE Model and graphic functions into a single program with a new graphic user interface (GUI). The integrated software will make it easier for users, such as the war fighter in an operational situation, to assess the acoustic environment of the battle space.

Chapter II provides a summary of the theory of the MMPE model and its implementation and graphic manipulation. It derives the general form of the parabolic equation to be used and describes the algorithm used in the MMPE model. The post-processing and graphic manipulation routines are also defined. In Chapter III, the design of the Java-based WinMMPE and its functions are presented. The details of the GUI are also presented. The model implementation, post-processing, and graphic manipulation are described. In Chapter IV, the output results of the WinMMPE are described. Sample outputs of WinMMPE are compared with data from the MMPE Model. This chapter also presents a discussion of the efficiency in terms of main memory used for implementation and running time. Finally, Chapter V provides general conclusions and recommendations for further work. Suggestions for methods of improving efficiency and the addition of new modules are provided.

THIS PAGE IS INTENTIONALLY LEFT BLANK

II. MONTEREY-MIAMI PARABOLIC EQUATION (MMPE) MODEL

A. INTRODUCTION

This chapter provides a general description of the Monterey-Miami Parabolic Equation (MMPE) Model, its algorithm, and its implementation. All contents in this chapter are summarized from the report of the University of Miami Parabolic Equation (UMPE) Model by Smith and Tappert (1993) and the recent article of Smith (2000) that analyzed the accuracy of the MMPE Model. Accordingly, the nomenclature used in this chapter is the same as in those reports. For more details, the reader should refer to those papers.

B. THE GENERAL THEORY

We begin by considering a general time-harmonic, continuous-wave (CW) acoustic field at angular frequency ω in a cylindrical coordinate system (r, z, φ),

$$P(r,z,\varphi,\omega t) = p(r,z,\varphi)e^{-i\omega t} \quad . \tag{2.1}$$

Combining Eq. (2.1) with the inhomogeneous wave equation for a point source (Jensen, et al, 2000),

$$\rho\nabla\cdot\left(\frac{1}{\rho}\nabla P\right) - \frac{1}{c^2}\frac{\partial^2 P}{\partial t^2} = -4\pi\delta\ (\vec{x}-\vec{x}_s)\ , \tag{2.2}$$

leads to the Helmholtz equation for the CW acoustic field,

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial p}{\partial r}\right)+\frac{1}{r^2}\frac{\partial^2 p}{\partial \varphi^2}+\frac{\partial^2 p}{\partial z^2}+k_0^2 n^2(r,z,\varphi)p = -4\pi P_0\delta(\vec{x}-\vec{x}_s)\ . \tag{2.3}$$

In this equation, k_0 is the reference wavenumber defined as

$$k_0 = \frac{\omega}{c_0} \ , \tag{2.4}$$

c_0 is the reference sound speed, $n(r,z,\varphi)$ is the acoustic index of reference defined as

$$n(r,z,\varphi) = \frac{c_0}{c(r,z,\varphi)} \ , \tag{2.5}$$

and $c(r,z,\varphi)$ is the acoustic sound speed that varies throughout space, in general. The source function is defined as a point source with reference source level P_0 at a reference distance of $R_0 = 1\,\mathrm{m}$, and

$$\delta(\vec{x}) = \frac{1}{2\pi r}\delta(z - z_s)\delta(r) \ , \tag{2.6}$$

where $r = 0$, $z = z_s$ refer to the position of the source. Solutions to Eq. (2.1) may be computed over a bandwidth of frequencies, and the results can then be Fourier synthesized to produce predictions of the arrival time structure due to broadband pulse propagation.

In most ocean environments, the influence of the azimuthal coupling term is negligible and may be ignored. To further simplify Eq. (2.3), we shall also neglect the source term initially. We may then define

$$p(r,z) = \frac{1}{\sqrt{r}}u(r,z) \ , \tag{2.7}$$

and substitution into Eq. (2.3) yields the free-field wave equation

$$\frac{\partial^2 u}{\partial r^2} + \frac{\partial^2 u}{\partial z^2} + k_0^2\left(n^2 + \frac{1}{4k_0^2 r^2}\right)u = 0 \ . \tag{2.8}$$

Next, we introduce the operators

$$P_{op} = \frac{\partial}{\partial r} \tag{2.9}$$

and

$$Q_{op} = \left(n^2 + \frac{1}{k_0^2} \frac{\partial^2}{\partial z^2} \right)^{1/2} . \tag{2.10}$$

In the far field ($k_0 r >> 1$), Eq. (2.8) then becomes

$$\left(P_{op}^2 + k_0^2 Q_{op}^2 \right) u = 0 . \tag{2.11}$$

Expanding Eq. (2.11) into incoming and outgoing wave equations, we obtain

$$- ik_0^{-1} \frac{\partial u}{\partial r} = Q_{op} u \tag{2.12}$$

for the outgoing wave when the incoming wave is negligible. Now u can be defined in terms of the PE field function, $\psi(r,z)$, as

$$u = \psi(r,z) e^{ik_0 r} . \tag{2.13}$$

Substituting this into Eq. (2.12), we finally obtain the PE for the field function,

$$\frac{\partial \psi}{\partial r} = -ik_0 \psi + ik_0 Q_{op} \psi = -ik_0 H_{op} \psi , \tag{2.14}$$

where

$$H_{op} = 1 - Q_{op} , \tag{2.15}$$

is an operator having Hamiltonian form. This operator defines the evolution of the PE field function in range.

C. THE SPLIT-STEP FOURIER (SSF) ALGORITHM

We introduce the PE field function, ψ, by formally defining the relationship with the acoustic pressure field as

$$p(r,z) = P_0\sqrt{\frac{R_0}{r}}Q_{op}^{-1/2}\psi(r,z)e^{ik_0 r} \ , \tag{2.16}$$

normalized such that at $r = R_0$, $|\psi| = 1$ and $|p| = P_0$. The computational evolution of the field is now accomplished via a marching algorithm of the form

$$\psi(r+\Delta r) = \Phi(r)\psi(r) \ . \tag{2.17}$$

From Eq. (2.14), the propagator $\Phi(r)$ can be shown to have the approximate form

$$\Phi(r) \approx e^{-ik_0\overline{H}_{op}(r)\Delta r} \tag{2.18}$$

where

$$\overline{H}_{op}(r) = \frac{1}{\Delta r}\int_r^{r+\Delta r} dr' H_{op}(r') \ . \tag{2.19}$$

If the range step Δr is small enough, the operator may be assumed to be relatively constant over a range step. Evaluating the operator at mid-step, for instance, gives

$$\overline{H}_{op}(r) = H_{op}\left(r+\frac{1}{2}\Delta r\right) . \tag{2.20}$$

From Eq. (2.10), it is observed that the operator Q_{op}, and therefore H_{op}, are pseudo-differential operators due to the square root. An approximation must then be made for numerical implementation. In order to implement the resulting parabolic equation using the split-step Fourier method (Hardin and Tappert, 1973), it is necessary to separate the index of refraction term from the vertical derivative term, i.e.,

$$H_{op} \approx T_{op} + U_{op} \ . \tag{2.21}$$

It has been shown (Jensen, et al., 2000) that a solution which is second order accurate in range step size, Δr, is obtained when the propagator function is defined as

$$\Phi(r) = e^{-ik_0\frac{\Delta r}{2}U_{op}(r+\Delta r)} e^{-ik_0\Delta r T_{op}} e^{-ik_0\frac{\Delta r}{2}U_{op}(r)} . \tag{2.22}$$

Using the wide-angle operator splitting introduced by Thomson and Chapman (1983), we define

$$T_{op} = 1 - \left[1 + \frac{1}{k_0^2}\frac{\partial^2}{\partial z^2}\right] \tag{2.23}$$

and

$$U_{op} = -(n-1) . \tag{2.24}$$

Because U_{op} is a scalar operator, it forms a diagonal matrix which can simply be multiplied by the PE field function at each range step. However, T_{op} is a differential operator, is not diagonal, and couples the solution between depth points. The split-step Fourier algorithm overcomes this by recognizing that the corresponding operator in the vertical wavenumber domain, $\hat{T}_{op}$, *is* diagonal and can be treated using a simple multiplication.

In the PE/SSF implementation, generally, a multiplication of the z-space operator $e^{-ik_0\frac{\Delta r}{2}U_{op}(r)}$ is defined at the initial range-step. The result is then transformed to the vertical wavenumber, k_z, domain. After a multiplication by the k_z-space operator $e^{-ik_0\Delta r\hat{T}_{op}}$, the solution is transformed back to the z-domain where a multiplication of the z-space operator $e^{-ik_0\frac{\Delta r}{2}U_{op}(r+\Delta r)}$ is defined at the end of the range step. A fast Fourier transform (FFT) algorithm is used to accomplish the transformation which assumes a convention

$$\psi(z) = FFT\left(\hat{\psi}(k_z)\right) . \tag{2.25}$$

The representation of the PE/SSF implementation is then

$$\psi(r+\Delta r, z) = e^{-ik_0\frac{\Delta r}{2}U_{op}(r+\Delta r,z)} \text{ x } FFT\left\{e^{-ik_0\Delta r\hat{T}_{op}(k_z)} IFFT\left[e^{-ik_0\frac{\Delta r}{2}U_{op}(r,z)}\psi(r,z)\right]\right\}, \tag{2.26}$$

where, in k_z-space,

$$\hat{T} = 1-\left[1-\left(\frac{k_z}{k_0}\right)^2\right]^{1/2} . \tag{2.27}$$

D. THE BOUNDARY CONDITIONS

The ocean surface in the MMPE model is defined by a Dirichlet boundary condition consistent with a pressure release boundary,

$$\psi(z=0) = 0 . \tag{2.28}$$

For this boundary condition, an image ocean method is used whereby an exact copy of the environment is defined for negative depths and the field function for these depths is defined by

$$\psi(-z) = -\psi(z) . \tag{2.29}$$

Accordingly, the data array is twice as large as that needed to describe the real acoustic field.

In the MMPE model, shear wave propagation in the bottom is treated by an equivalent fluid technique. Thus, the propagation of shear waves are not computed but the effect of shear on the boundary reflection coefficient is included. Specifically, the equivalent fluid approximation of Tindle and Zhang (1992) is employed at each bottom interface.

It is assumed that the interface between the bottom of the water column and the top of the basement, or sediment, layer is characterized by a discontinuity due to a sharp contrast in sound speed. Specifically, the sound speed at the interface is defined by

$$c(z) = c_w(z) + [c_b(z) - c_w(z)]H(z - z_b) \ , \tag{2.30}$$

where c_w and c_b are the sound speed of water and bottom, respectively, and the Heaviside step function is defined by

$$H(\varsigma) = \begin{cases} 0, & \varsigma < 0 \\ 1/2, & \varsigma = 0 \\ 1, & \varsigma > 0 \end{cases} \ , \tag{2.31}$$

where $\varsigma \equiv z - z_b$. In order to implement within the numerical scheme, it is necessary to approximate this by a smooth, continuous function. For that purpose, this function is approximated by a hyperbolic tangent function,

$$\overline{H}(\zeta) = \frac{1}{2}\left[1 + \tanh\left(\frac{\zeta}{2L_c}\right)\right] \tag{2.32}$$

or, equivalently,

$$\overline{H}(\zeta) = \left(1 + e^{-\zeta/L_c}\right) \ , \tag{2.33}$$

where L_c is the sound speed mixing length.

In order to create the most realistic interface condition for reflections from a sound speed discontinuity, Smith (2000) showed empirically that the most accurate results were obtained by defining L_c as a fraction of a wavelength, specifically $\lambda/10$. For sampling purposes, the default minimum value for L_c is set at $L_{c,\min} = \Delta z$. Thus, $L_c = \max(\Delta z, \lambda/10)$, so the most accurate results are obtained efficiently when $\Delta z \le \lambda/10$.

We must now consider the effect of the bottom density contrast, something which was neglected in the definition of the wave equation given in Eq. (2.2). A proper treatment including density produces the same expressions given above if we replace the index of refraction n by an "effective" index of refraction,

$$n'^2 = n^2 + \frac{1}{2k_0^2}\left[\frac{1}{\rho}\nabla^2\rho - \frac{3}{2}\left(\frac{1}{\rho}\nabla\rho\right)^2\right] . \tag{2.34}$$

Then, the pressure field $p(r,z)$ is defined by

$$p(r,z) = P_0\sqrt{\frac{\rho R_0}{\rho_0 r}}Q_{op}^{-1/2}\psi(r,z)e^{ik_0 r} , \tag{2.35}$$

where ρ_0 is a reference density defined by $\rho_0 = 1.0$ g/cm^3. The environmental propagator function in Eq. (2.24) must now be defined as

$$U_{op}(z) = U_1(z) + U_2(z) , \tag{2.36}$$

where $U_1(z)$ is the same as the previously defined environmental potential function in Eq. (2.24) and $U_2(z)$ is a new operator that includes the effect of the density discontinuity.

If we assume that $\rho = \rho(z)$ only, the density is defined by

$$\rho(z) = \rho_w + (\rho_b - \rho_w)H(z - z_b) , \tag{2.37}$$

where ρ_w and ρ_b are the density of water and bottom, respectively. As before, we must replace the Heaviside step function by a smooth, continuous function scaled by a density mixing length, L_ρ. Within the MMPE Model, the function U_2 is approximated by

$$U_2(z) \approx -\frac{\varepsilon}{k_0^2}\frac{\partial^2}{\partial z^2}H(z - z_b) , \tag{2.38}$$

where

$$\varepsilon = \left[\frac{1-(\rho_w/\rho_b)^{1/2}}{1+(\rho_w/\rho_b)^{1/2}}\right]. \tag{2.39}$$

For the density mixing function, a cubic spline over the finite interval $-L_\rho \le \zeta \le L_\rho$ is defined. Then $\overline{H}(\zeta)$ is defined by

$$\overline{H}(\zeta) = \begin{cases} 0 & , \quad \zeta \le -L_\rho \\ \frac{2}{3}\left(1+\frac{\zeta}{L_\rho}\right)^3 & , \quad -L_\rho \le \zeta \le -\frac{L_\rho}{2} \\ \frac{1}{2}+\frac{\zeta}{L_\rho}-\frac{2}{3}\left(\frac{\zeta}{L_\rho}\right)^3 & , \quad -\frac{L_\rho}{2} \le \zeta \le -\frac{L_\rho}{2} \\ 1-\frac{2}{3}\left(1-\frac{\zeta}{L_\rho}\right)^3 & , \quad \frac{L_\rho}{2} \le \zeta \le L_\rho \\ 1 & , \quad \zeta \ge L_\rho \end{cases}. \tag{2.40}$$

The first derivative of this function is

$$\overline{H}'(\zeta) = \overline{\delta}(\zeta) = \begin{cases} 0 & , \quad \zeta \le -L_\rho \\ \frac{2}{L_\rho}\left(1+\frac{\zeta}{L_\rho}\right)^2 & , \quad -L_\rho \le \zeta \le -\frac{L_\rho}{2} \\ \frac{1}{L_\rho}\left[1-2\left(\frac{\zeta}{L_\rho}\right)^2\right] & , \quad -\frac{L_\rho}{2} \le \zeta \le -\frac{L_\rho}{2} \\ \frac{2}{L_\rho}\left(1-\frac{\zeta}{L_\rho}\right)^2 & , \quad \frac{L_\rho}{2} \le \zeta \le L_\rho \\ 0 & , \quad \zeta \ge L_\rho \end{cases}, \tag{2.41}$$

where $\overline{\delta}(\zeta)$ is the smooth approximation to the Dirac delta function. The second derivative is then

$$\overline{H}''(\zeta)=\overline{\delta}'(\zeta)=\begin{cases} 0 & , \quad \zeta \le -L_\rho \\ \frac{4}{L_\rho^2}\left(1+\frac{\zeta}{L_\rho}\right) & , \quad -L_\rho \le \zeta \le -\frac{L_\rho}{2} \\ -\frac{4}{L_\rho^2}\left(\frac{\zeta}{L_\rho}\right) & , \quad -\frac{L_\rho}{2} \le \zeta \le -\frac{L_\rho}{2} \\ -\frac{4}{L_\rho^2}\left(1-\frac{\zeta}{L_\rho}\right) & , \quad \frac{L_\rho}{2} \le \zeta \le L_\rho \\ 0 & , \quad \zeta \ge L_\rho \end{cases} . \tag{2.42}$$

Equations (2.38) and (2.42) provide the necessary expressions for computing the density potential function with a cubic spline polynomial smoothing function.

In order to create the most realistic interface condition for reflections from a density discontinuity, Smith (2000) showed empirically that the most accurate results were obtained by defining $L_\rho \sim 2\lambda$. For sampling purposes, the default minimum value for L_ρ is set at $L_{\rho,\min} = 5\Delta z$. Thus, $L_\rho = \max(5\Delta z, 2\lambda)$, so the most accurate results are obtained efficiently when $\Delta z \le 2\lambda/5$.

E. THE SOURCE FUNCTION

Near the source, we may assume that the effect of density is negligible and the operator in Eq. (2.16) is near unity. The acoustic pressure near the source is then defined simply by

$$p = P_0\sqrt{\frac{R_0}{r}}\psi e^{ik_0 r} , \tag{2.43}$$

where R_0 is a reference range defined by 1m and P_0 is the source pressure at R_0. The source level, *SL*, becomes

$$SL = 20\log\left(\frac{P_0}{P_r}\right) dB \; re \; P_r, R_0 , \tag{2.44}$$

where P_r is a reference pressure, typically defined by $P_r = 1\,\mu Pa$.

From Eq. (2.43), $\psi(r=0, z)$ can be expressed as

$$\psi(r=0,z) = \lim_{r\to 0} \frac{1}{P_0}\sqrt{\frac{r}{R_0}}p(r,z)\ . \tag{2.45}$$

In the vicinity of a point source, the pressure field takes the form of a spherical Green's function, i.e.

$$p = \frac{a}{4\pi R}e^{ik_0 R} \quad , \quad R = \sqrt{r^2+z^2} \quad , \quad a = P_0 R_0\ , \tag{2.46}$$

where a is defined by requiring $|p| = \frac{P_0}{4\pi}$ at $R = R_0$. If we then define the PE field starting condition for a point source at depth z_s by

$$\psi(r=0,z) = \alpha\delta(z-z_S)\ , \tag{2.47}$$

then it can be shown that the proper normalization is

$$\alpha = \sqrt{\frac{iR_0}{2\pi k_0}}\ . \tag{2.48}$$

To represent this field in the vertical wavenumber, k_z-domain, we perform a Fourier transformation of Eq. (2.47) to produce

$$\hat{\psi}(r=0,k) = -2i\alpha\sin(k_z z_S)\ , \tag{2.49}$$

which also includes the influence of the image source.

For the wide angle approximation in the vertical wavenumber domain, we need to add two factors in Eq. (2.49). First of all, to produce the correct solution in the far-field, the factor $\left(1-\frac{k_z^2}{k_0^2}\right)^{-1/4}$, $(|k_z| < k_0)$ needs to be added to the wide angle source (Thomson

and Bohun, 1988). Secondly, in order to take advantage of the MMPE depth grid convention, defined by

$$z_n = \begin{cases} \left(n-\frac{1}{2}\right)\Delta z\,, & n=1,\frac{N}{2} \\ -\left(N-n+\frac{1}{2}\right)\Delta z\,, & n=\frac{N}{2}+1,N \end{cases}, \tag{2.50}$$

we need to add a phase term in the wavenumber domain of $e^{ik_0\frac{\Delta z}{2}}$. Finally then, the PE starting field in the k_z-domain for the wide angle source is given by

$$\hat{\psi}(r=0,k_z) = -2i\sqrt{\frac{iR_0}{2\pi k_0}}\sin(k_z z_S)\left(1-\frac{k_z^2}{k_0^2}\right)^{-1/4} e^{ik_z\frac{\Delta z}{2}}\,. \tag{2.51}$$

F. IMPLEMENTATION

There are two aspects of the implementation of the MMPE Model. The first is the propagation model implementation that produces a binary output file. The second aspect is then the post-processing of this binary data file and the graphical representation of the results. This is treated using Matlab routines. The results of the output PE solution will be discussed in Chapter IV. In this section, the run-time process and the output structure is discussed.

1. Model Implementation

The MMPE Model can be run using a DOS command line interface as illustrated in Figure 2.1 (a). Prior to execution, several ASCII input files must be defined. The primary input data file is named "pefiles.inp". Within this file, all other filenames that provide details of the environment (sound speed profiles, bathymetry, etc.) are defined. Of all file names, only the specific name "pefiles.inp" is required. Also within this main file are specifications for the output data file (number and range of output depths and

ranges) as well as optional declarations of computational parameters (FFT size, range and depth grid sizes, etc.).

When run, the propagation model asks the user to choose either an accurate mode or an efficient mode of calculation (unless the grid sizes have been explicitly specified in "pefiles.inp"). In the former, the depth grid is 1/10 of a wavelength and the range grid is a wavelength. In the latter, the depth grid is half a wavelength and the range grid is three times a wavelength. The model calculates the PE field depending on the user's choice. During the calculation, the range step displays on the screen, along with the current frequency and radial bearing being computed, as illustrated in Fig. 2.1 (b). The result of the calculation is the binary output with filename defined in "pefiles.inp."

The structure of the output binary file is shown in Figure 2.2. For all but the final records (containing bathymetry information), each record length is twice the number of points stored in depth. The factor of two accounts for the real and imaginary parts of the complex PE field function. The first record contains header information providing details about the remainder of the data and the calculation as follows: the total length of the data file; the reference sound speed; the number of frequencies computed; the center frequency; the frequency bandwidth; the number of range points output; the minimum range for output; the maximum range for output; the range step size used in the calculation; the number of depth points output; the minimum depth for output; the maximum depth for output; the number of radials; the angular difference between radials; the bottom depth at the source location; the deep bottom depth at the source location; and the source depth.

For multiple frequency and multiple radial calculations, the model first computes the solution out in range for the initial frequency and radial. This begins by computing the starting field, which is the same for all radials at that frequency. It then computes the solution for all subsequent radials at that frequency. After computing the solution out in range for every radial, it then begins the calculations for the subsequent frequencies. The result of the ordering is the binary data file organization depicted in Figure 2.2.

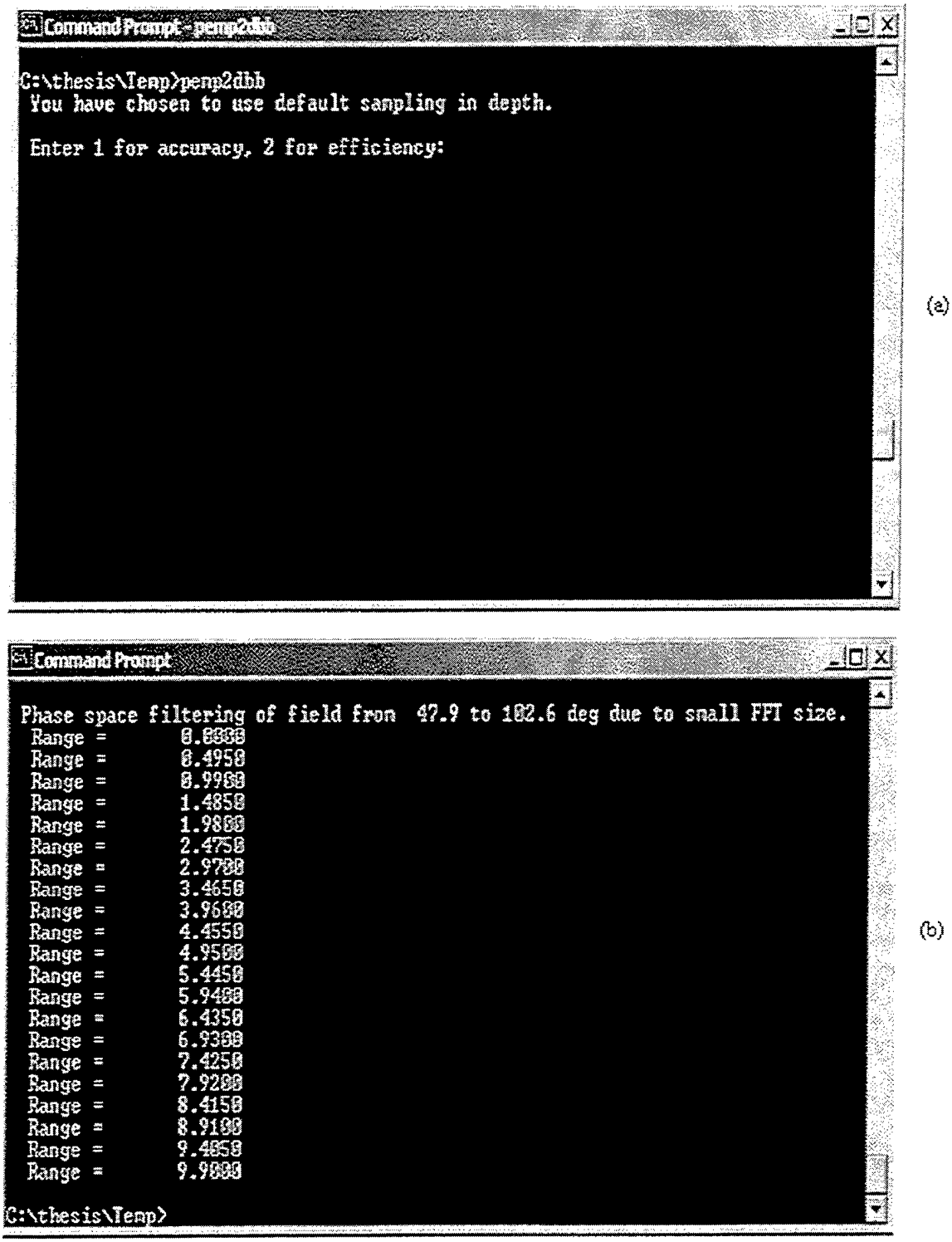


Figure 2.1: The MMPE model run-time environment: (a) initiation of model run; (b) model output while running.

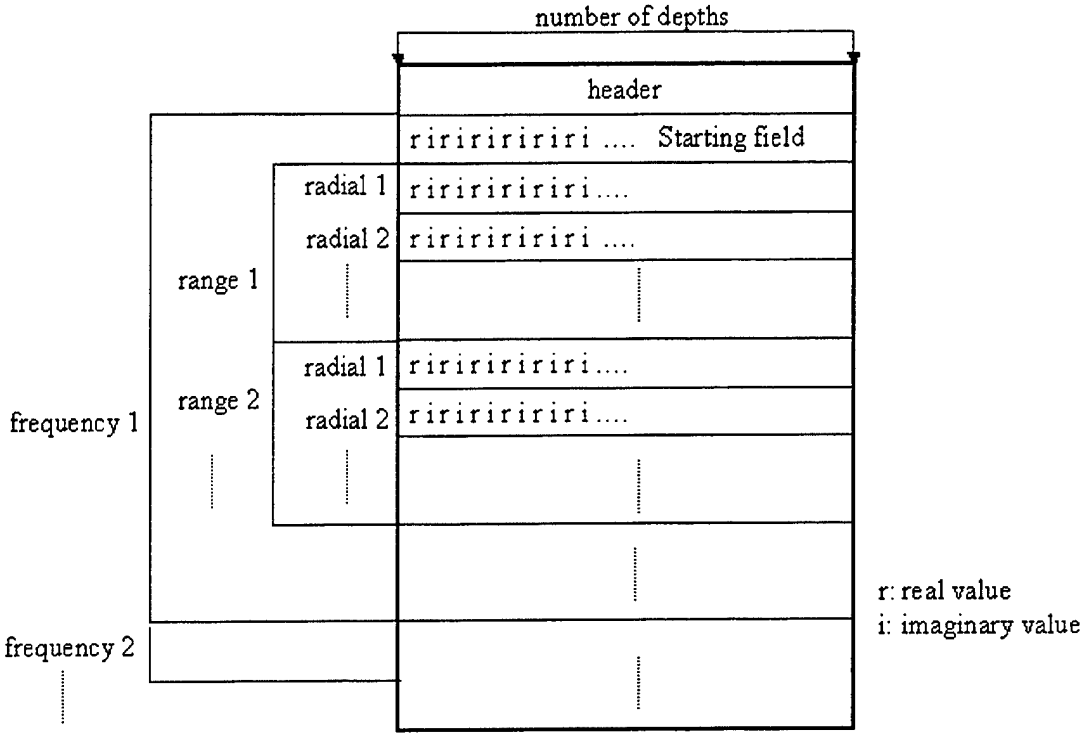


Figure 2.2: The structure of the MMPE output binary data file.

Note that the data ordering places adjacent radials next to each other rather than adjacent range steps as might be expected from the actual run-time ordering. This was done to maintain consistency with other versions of MMPE, which incorporate true 3D effects with azimuthal coupling. In those versions, all radials are computed at each range step. Also note that the complex PE field function values are stored as real and imaginary data pairs.

2. Post-Processing and Graphic Manipulation

The post-processing implementation runs in the Matlab command window. The initialization routine is called "peout1", which requests the user to enter the name of the binary output file of the MMPE Model to be processed. After entering the file name, the data of the output file is loaded to the variables in "peout1" as shown in Figure 2.3 (a). The main processing routine, "peout2", can then be run multiple times without re-running the initialization routine so long as the same file is being processed. After typing

"peout2", six options for processing are given, as shown in Figure 2.3 (b). The user then chooses the option for the desired processing and data extraction. For the option chosen, all acoustic quantities of interest are computed, and a corresponding transmission loss field is plotted. The outputs will be discussed in more detail in Chapter IV.

For all output options of interest here, transmission loss (TL) is defined by

$$TL(r,z,\varphi) = 20\log_{10}\left(\frac{|p(r,z,\varphi)|}{P_0}\right) \text{ dB re 1m}, \tag{2.52}$$

where P_0 is the source pressure amplitude measured at the reference distance of $R_0 = 1\text{m}$. From the definition of the pressure field in terms of the PE field function in Eq. (2.16), this may be rewritten

$$\begin{aligned} TL(r,z,\varphi) &= 20\log_{10}\left(\frac{1}{\sqrt{r}}|\psi(r,z,\varphi)|\right) \\ &= 20\log_{10}|\psi(r,z,\varphi)| - 10\log_{10} r \text{ dB re 1m} \end{aligned} \tag{2.53}$$

Thus, the solution of the PE field function provides all the information necessary to compute TL with the exception of the simple cylindrical spreading factor.

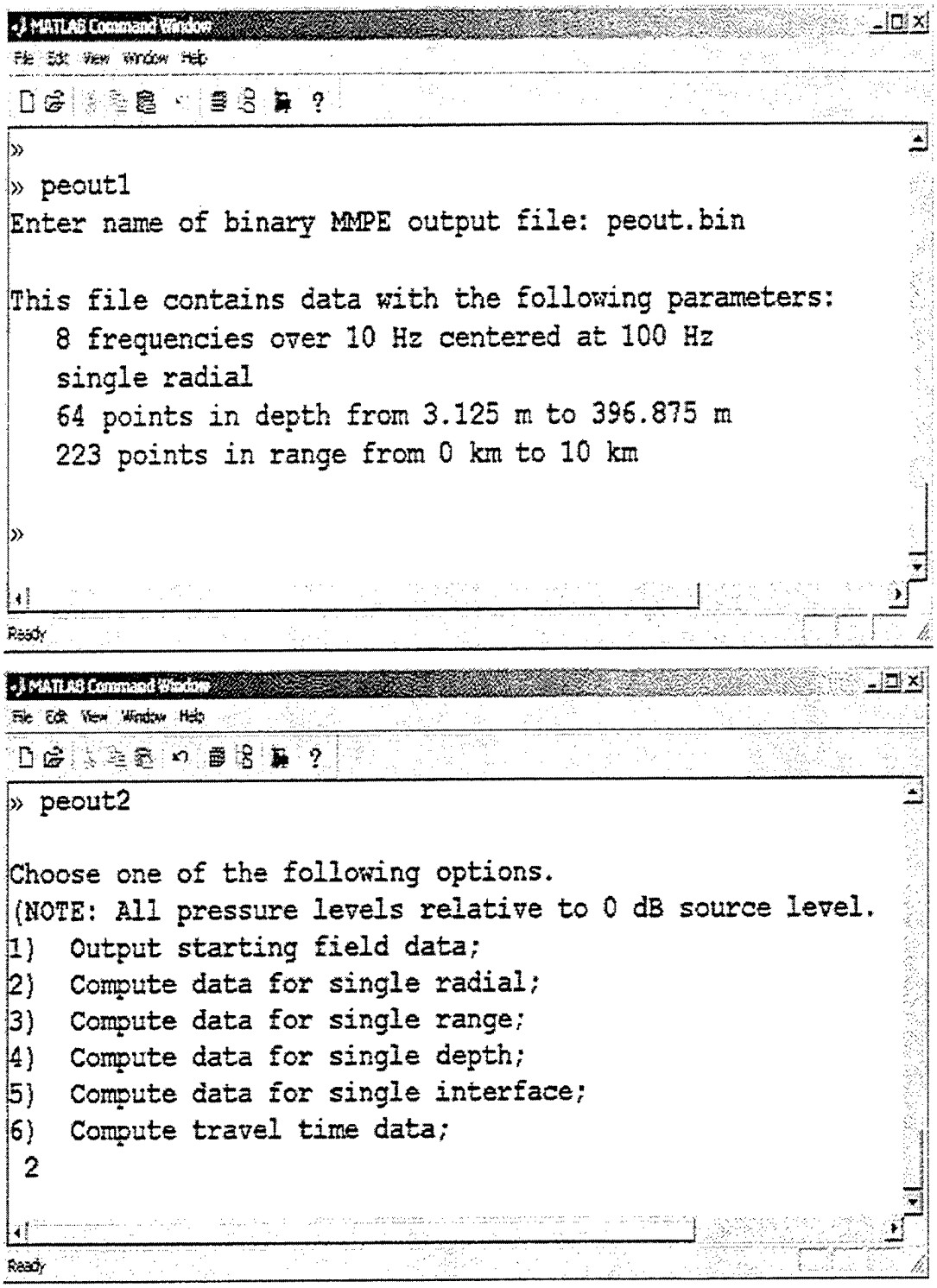


Figure 2.3: Post-processing and graphic implementation: (a) loading the MMPE output file; (b) choosing one of the processing options.

THIS PAGE IS INTENTIONALLY LEFT BLANK

III. WINDOWS MMPE (WINMMPE)

A. INTRODUCTION

As described in the previous chapter, the MMPE model can be executed via Unix shell or DOS command. Then, the post-processing and the graphic manipulation are implemented in MATLAB. Since the final goal of this thesis is integrating those functions and providing a better interface for users, a Windows version of the MMPE satisfying these goals has been created. It is called WinMMPE. This chapter describes its design and functions. Also, its implementation is described in terms of the model and the graphic manipulation. The WinMMPE application is executed by clicking its icon (Figure 3.1).

Figure 3.1 – Icon of WinMMPE

B. DESIGN AND FUNCTIONS

WinMMPE is composed of five frames, consisting of the main frame, model control frame, input data editor frame, graphic control frame, and graphic output frame. Each of the frames has its own design and functions. This section describes how they have been designed and what kinds of functions they have.

1. Main Frame

A double click of the left mouse button on the icon makes the WinMMPE display the main frame (Figure 3.2). It provides a full Windows graphic user interface for all WinMMPE functionality. There are five menus, which are "File" menu, "Edit" menu, "View" menu, "Window" menu, and "Help" menu. Each menu has several menu items and all are shown in Table 3.1. In the "File" menu, there are five menu items. "New

Source" menu item renders model control frame with main input file, "default.inp." "Open Source" menu item renders an open file chooser. Users can choose a ".inp" file as a main input file. If other formats are chosen instead of ".inp," file errors will occur. "Save" and "Save As" menus are used to save the main input file with the given file name, or to save it with another name if desired. "Exit" menu item makes WinMMPE terminate. The "Edit" menu has three menu items: "Cut," "Copy," and "Paste." When editing input files in the "Input File Editor" frame, these menu items are supposed to work for the frame. However, they do not work yet and are left for future work.

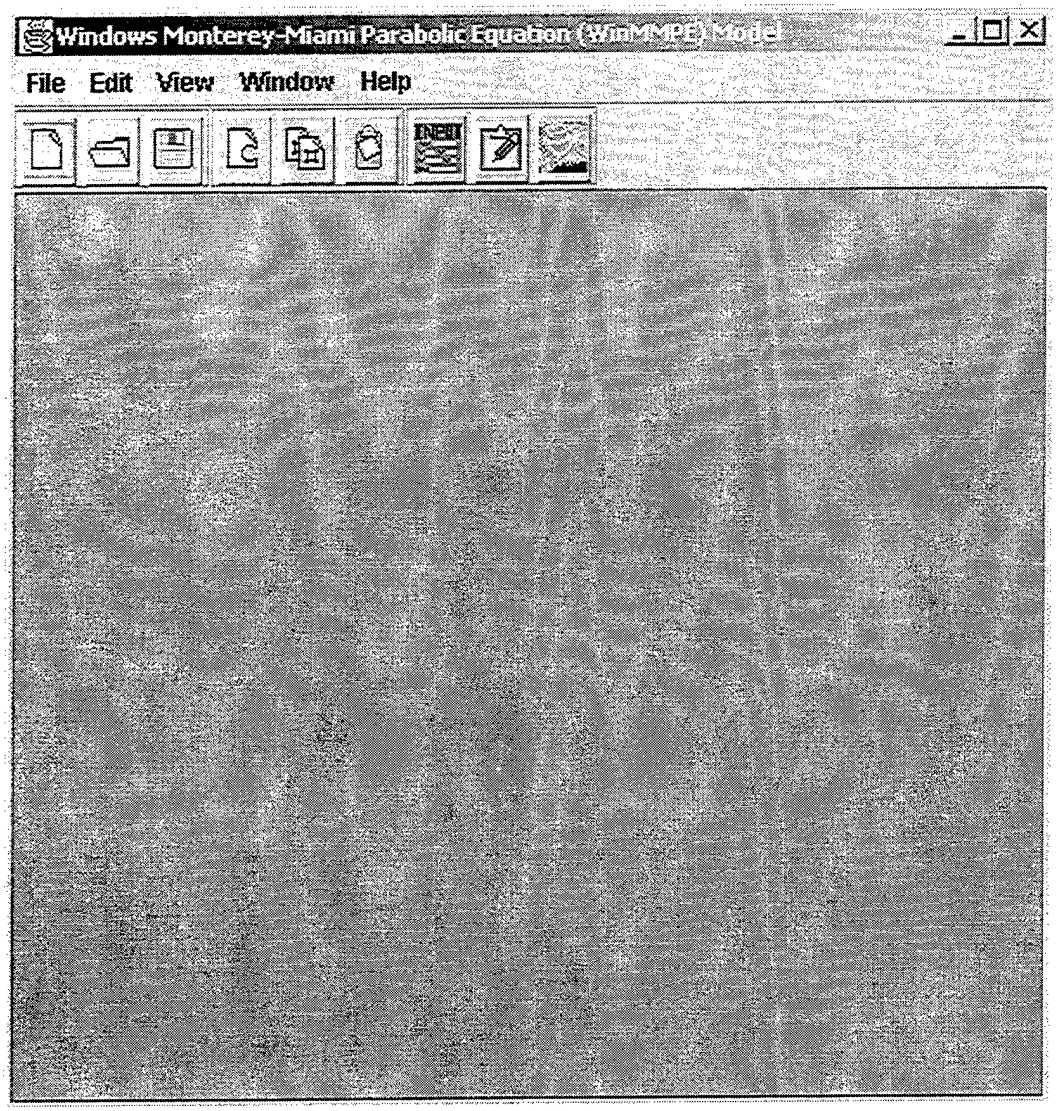


Figure 3.2 – WinMMPE Main Frame

Menu	Menu Items
File	New Source
	Open Source
	Save
	Save As
	Exit
Edit	Cut
	Copy
	Paste
View	Model Control Frame
	Input File Editor Frame
	Graphic Control Frame
Window	Opened Frames
Help	Web Help
	About WinMMPE

Table 3.1 – Menu and Menu Items of Main Frame

In the "View" menu, there are three menu items: "Model Control" frame, "Input File Editor" frame, and "Graphic Control" frame. Whenever any of these menu items is selected, a file chooser is opened. For "Model Control" frame and "Input File Editor" frame, it shows ".inp" file as option. For "Graphic Control" frame, it shows ".bin" file as option. The "Window" menu shows the opened frames as menu items. Users can check how many window frames are on the main frame and can pop up the frame they select. The "Help" menu has two menu items: "Web Help" and "About WinMMPE." The former lets users give the information about menu and each menu item of WinMMPE. The latter is the general explanation of WinMMPE.

The nine buttons in the tool bar below the menu bar are seen in Figure 3.1. From left to right, they are: "New Source" button, "Open Source" button, "Save" button, "Cut" button, "Copy" button, "Paste" button, "Model Control" frame button, "Input Data Editor" frame button, and "Graphic Control" frame button. Each button is able to work like the corresponding menu item previously described.

2. Model Control Frame

This frame launches the main input data file and the sound source input file, which have the same form as the ones used in the MMPE model. It comprises seven panels: "Output Data Files" panel, "Depth Info" panel, "Range Info" panel, "Basic Info" panel, "Source Info" panel, mode selection panel, and buttons panel (Figure 3.3). The first four panels load the main input data file, and the "Source Info" panel loads the sound source input file. In the "Output Data Files" panel, the output file name and log file name can be given. The text fields in the "Depth Info" and "Range Info" panels decide the depth and range information for the output data. The text fields in the "Basic Info" panel provide parameters for running the model, for example, FFT size. If the value of "Number of Depth Points" is zero, depth grid size is determined according to accurate or efficient mode. If it is not zero, it is assigned to the variable, deciding the FFT size. The value of "Absolute Max Depth" is the maximum depth of the deep bottom. If this value is zero, it is reset to the two times the maximum depth of the water/bottom interface.

The radio buttons in the mode selection panel for depth and range are used in deciding the grid size of depth and range. The buttons panel has five buttons: "Run" button, "Plot" button, "Input File Info" button, "Clear" button, and "Help..." button. The "Run" button starts executing the model. After the model computations are completed, the output binary file is created and the "Graphic Control" frame is automatically opened. The "Plot" button renders to open the "Graphic Control" frame with the output binary file given in the text field of the output file name. The "Input File Info" button renders to open the "Input File Editor" frame with the input files recorded in the main input file. The "Clear" button clears all text fields. The "Help..." button shows the explanation about this frame.

3. Input Data Editor Frame

This frame loads five input data files, specifically sound speed profile (SSP) data file, bottom bathymetry data file, bottom property data file, deep bottom bathymetry data file, and deep bottom property data file. Like the main input file and the sound source

input file, these files have the same form as the ones used in the MMPE model. The format of the data files is described in Smith and Tappert (1993). The "Input Data Editor" frame has a tabbed pane with five panels as shown in Figure 3.4. Each panel has a file name text field, "Save" button, "..." button which renders a file chooser to open an input data file, and a text area.

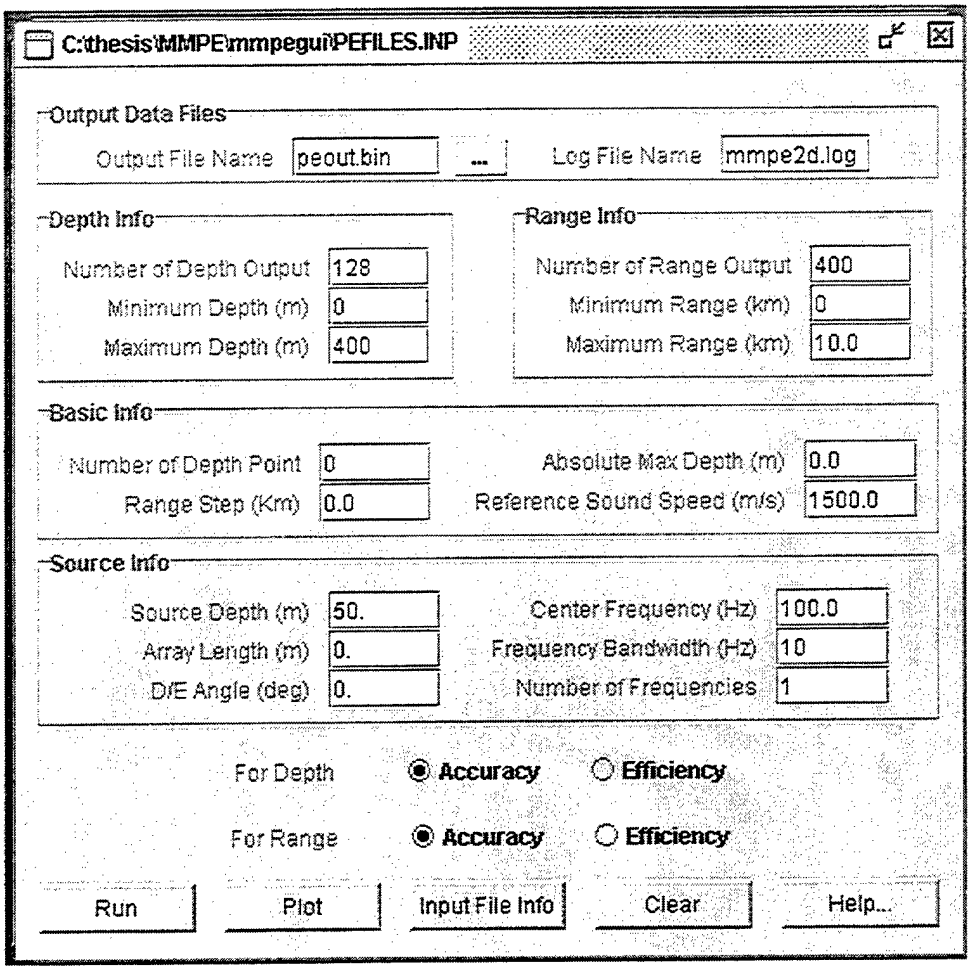


Figure 3.3 – Model Control Frame

User evaluation found that the "Input File Editor" frame does not work well as an ASCII file editor. JTextArea, a Java class in the Java standard library, provides an editing capability for the input files. However using this class is not desirable because it sets the size of rows and columns with two dimensional character arrays. Therefore,

future work needs to avoid using JTextArea class. JEditorPane might well be used instead. The problems in this frame are left for future work.

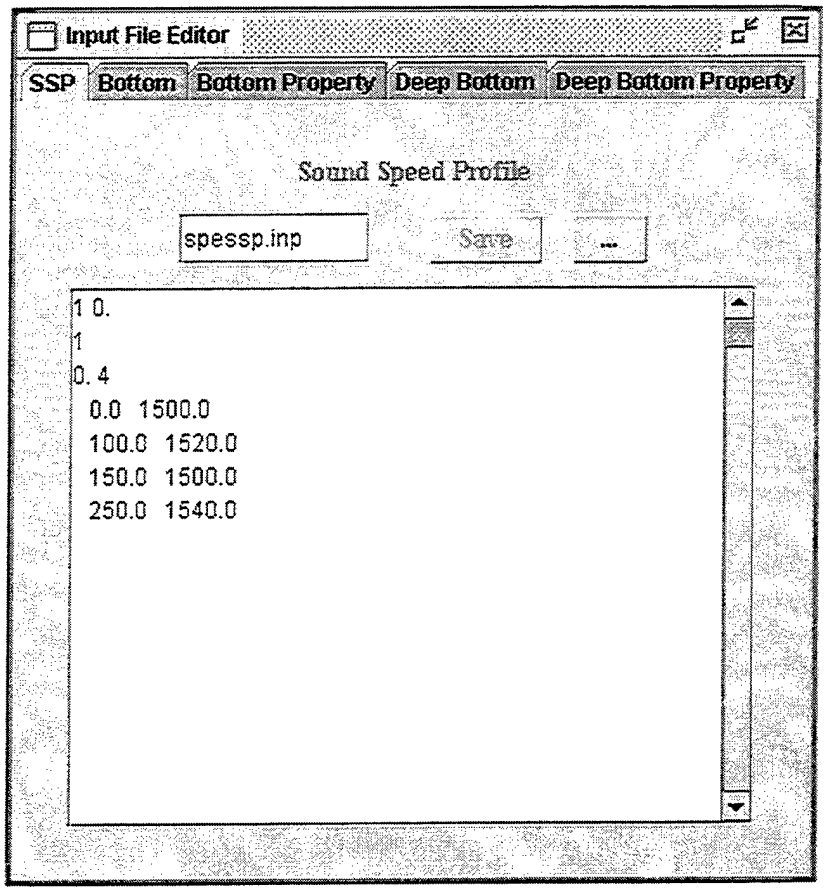


Figure 3.4 – Input File Editor Frame

4. Graphic Control Frame

As described in Chapter II, in the post-processing of MMPE, there are six kinds of options for graphic manipulation. In WinMMPE, four options have been worked so far: "Single Radial," "Single Depth," "Single Interface" and "Source Field." The "Graphic Control" frame has a panel and a tabbed pane and is shown in Figure 3.5. The output binary file is chosen in the panel. With the "..." chooser button the output binary file that users want to use can be selected and loaded. The tabbed pane has the four panels with each graphic option. The graphic output from each graphic option and the components of each panel in details are described later in this chapter.

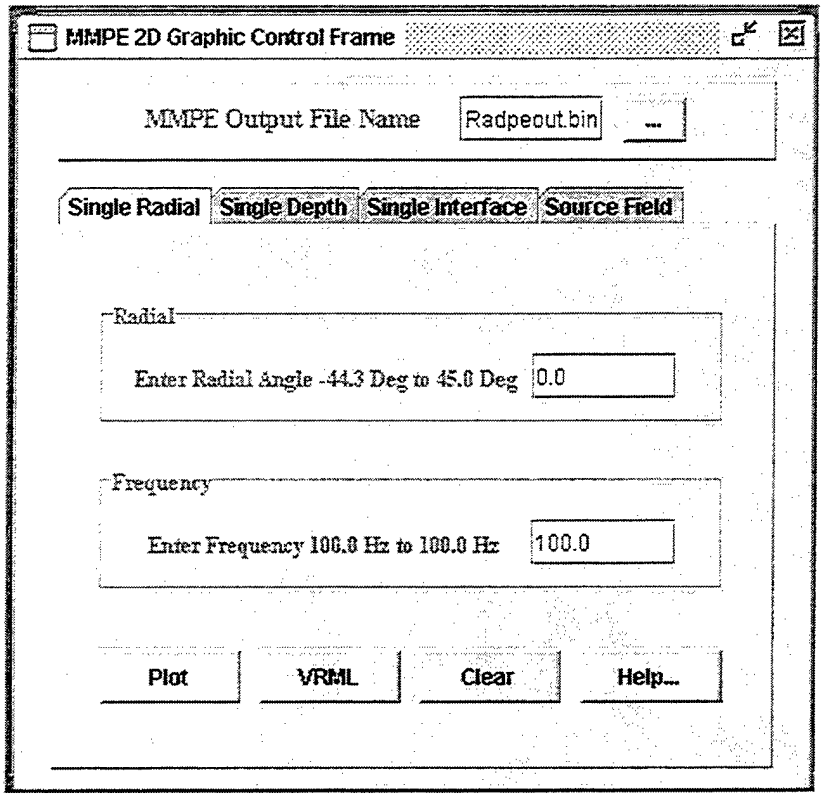


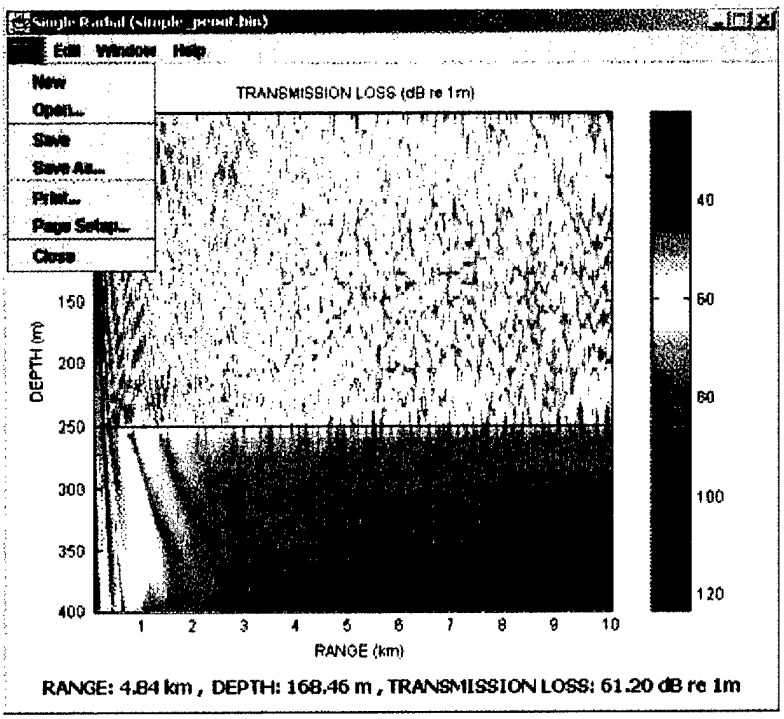
Figure 3.5 – Graphic Control Frame

5. Graphic Output Frame

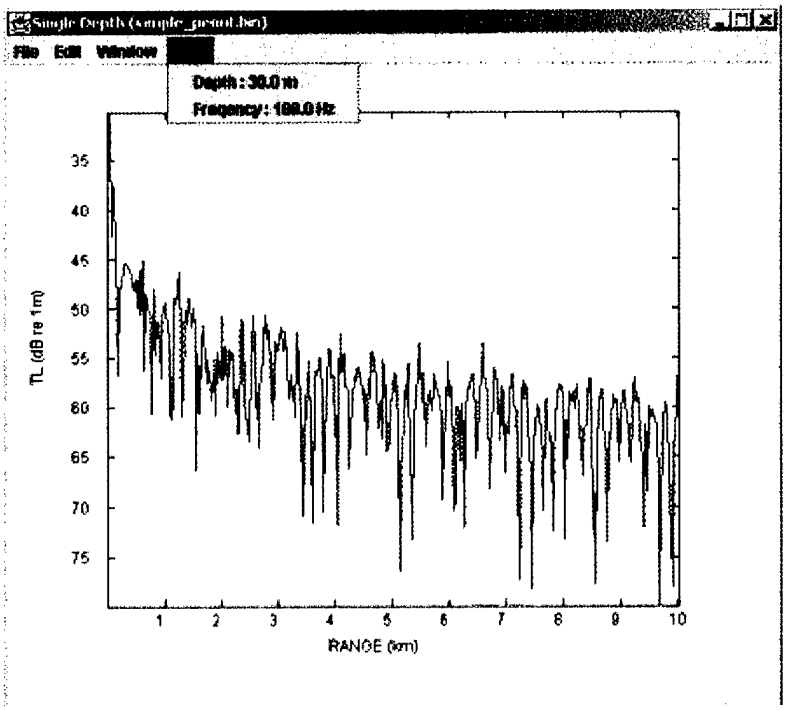
The above three frames, "Model Control" frame, "Input File Editor" frame, and "Graphic Control" frame are displayed inside the main frame, while the "Graphic Output" frame is displayed independently on the main frame and is shown in Figure 3.6. The title bar of the frame shows the chosen graphic option from the "Graphic Control" frame and the output file name. There are four menus: "File," "Edit," "Window," and "Help." As shown in Figure 3.6 (a), the "File" menu has seven menu items: "New," "Open," "Save," "Save As," "Print," "Page Setup," and "Close." Of these menu items, "Print," "Page Setup," and "Close" do work and the others do not work. In addition, "Edit" and "Window" menus do not work, either. They are left for future work. The "Help" menu shows the user's choice from the option in the "Graphic Control" frame. It is shown in Figure 3.6 (b).

Two kinds of graphic output can be launched in this frame. One is a color map image of a transmission loss field (Figure 3.6 (a)). If the user would like to know the value of a transmission loss at a point inside the image, the user can put the cursor at that position and then click. The value is shown at the bottom label of the graphic frame. The bar of color index with respect to transmission loss is also given. Another type of plot is a single line of the transmission loss (Figure 3.6 (b)). The number and the value of the scale in the x-axis and y-axis of the plot are decided by the value depending on the difference of minimum value and maximum value.

To develop the algorithm for the color map shown in Figure 3.6 (a), a lot of experimental trial and error regarding treatment of RGB values was performed. The final color method is shown in Figure 3.7. The red, green, and blue values of the color map start from 0.5, 0, and 0, respectively at the minimum TL. They finish at 0, 0, 0.5 at the maximum TL.



(a)



(b)

Figure 3.6 – Graphic Output Frame

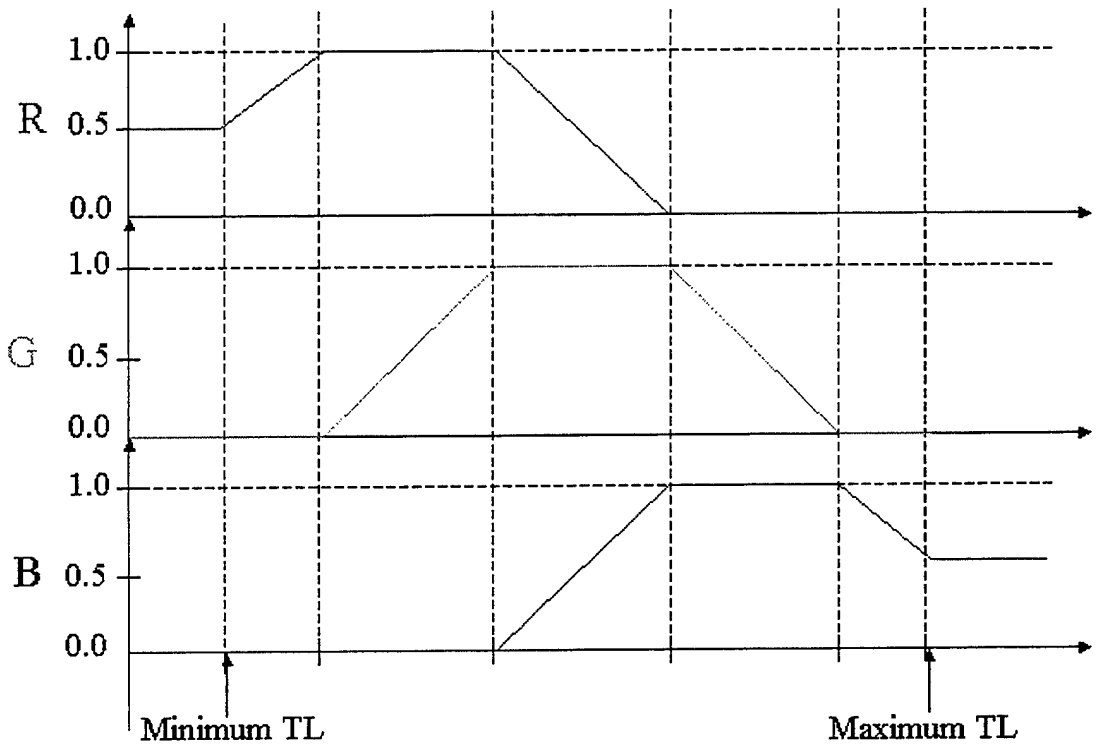


Figure 3.7 – Algorithm of Color Map

C. IMPLEMENTATION

As described in the previous section, all of the functionality in WinMMPE is initiated by clicking a button in the main frame. Model execution is initiated by clicking the "Run" button in the "Model Control" frame. The graphic manipulation is initiated by the "Plot" button in the "Graphic Control" frame. For graphic image generation, each of four graphic options is described in this section. This section does not describe what kinds of data used, but rather how to manipulate the graphic outputs.

1. Model Execution

As soon as the model starts running, processing outputs appear in the DOS command window as shown in Figure 3.8. The program then produces a binary output file. The file has the same structure as the one from the MMPE model. Ordinarily the binary serialization file in Java uses big-endian floating-point representations as a default type in the standard Java library supported by Sun Microsystems Corp. However, in the

WinMMPE, the little-endian representation is used for the binary output file to be compatible with the Windows version of Matlab for post-processing of the MMPE model data. The little endian java class is referenced at http://mindprod.com/products.html - LEDATASTREAM.

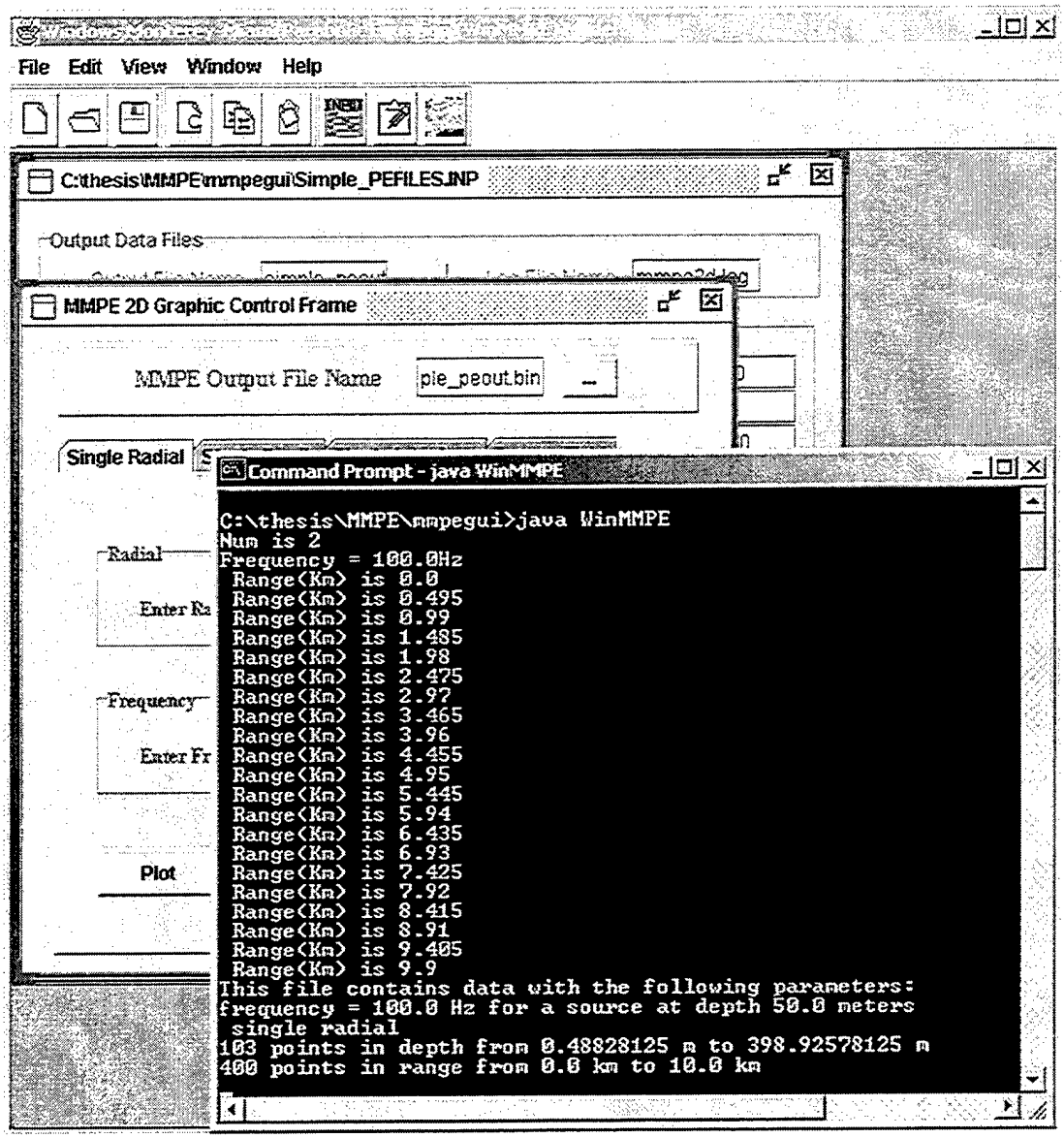


Figure 3.8 – Model Running in the WinMMPE

2. Post-processing and Graphic Manipulation

The "Single Radial" post-processing option in the "Graphic Control" frame is shown in Figure 3.5. The graphic output is produced by Eq. (2.53). It has a "Radial" section, a "Frequency" section, and four buttons: "Plot," "VRML," "Clear," and "Help." Each section has an input text field. Whenever the output file is loaded, 0.0 degrees and the center frequency are set as defaults in the "Radial" and "Frequency" fields, respectively. The "Plot" button renders a "Single Radial" graphic output. The title bar of the frame shows the phrase, "Single Radial," and the output file name. The acoustic transmission loss field at a single radial along the depth and the range is shown in Figure 3.9. The single line is the bottom bathymetry as a function of range. It is possible to show the deep bottom bathymetry as well, if it exists in the data.

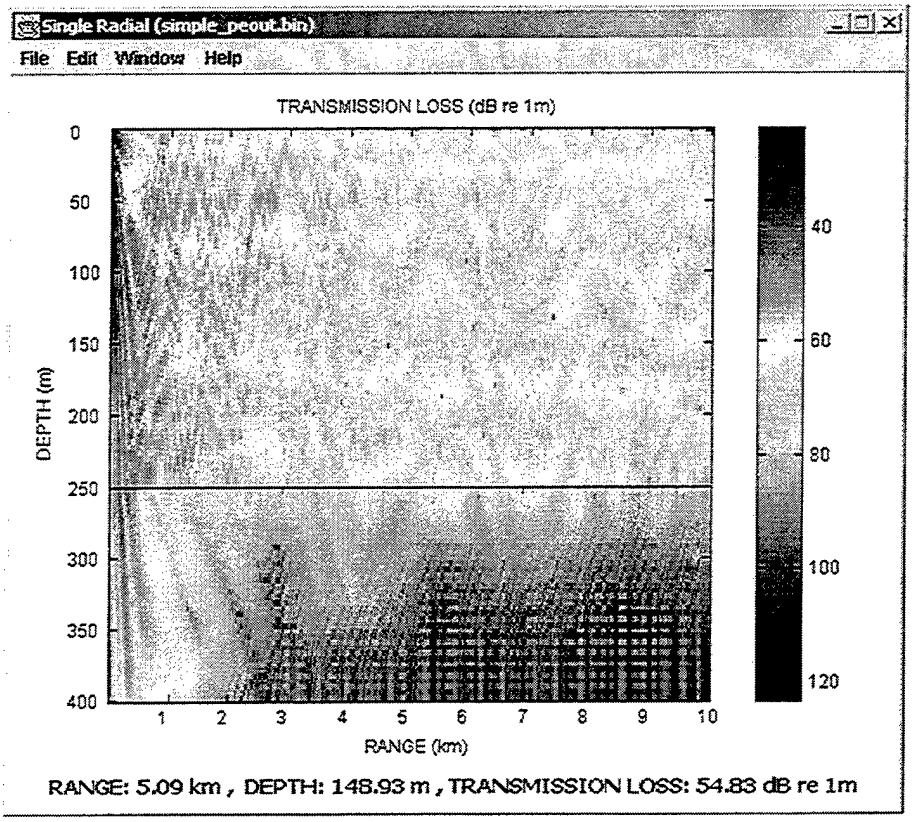


Figure 3.9 – Single Radial Graphic Output

The "VRML" button is only in the "Single Radial" option. This button renders to create four ASCII data files for showing a VRML scene graph. They are "point" for point data, "color" for color data, and "indexPoint" for coordIndex and colorIndex in VRML. These files are combined in a VRML file, "mmpeVRML.wrl." The WinMMPE then renders a web browser set as default on the computer. The web browser must be able to launch the "VRML" file with the extension ".wrl." Then the 3D scene of the acoustic transmission loss field is simulated, as depicted in Figure 3.10. In fact, the data can be viewed with the true aspect ratio in terms of real range and real depth. Note that the scene of Figure 3.10 is minimized by 1/1000 for range and by 1/100 for depth. Currently the 3D scene does not have much meaning since it is simply a single 2D acoustic field in 3D space. Future work expanding the 3D model is discussed in Chapter V.

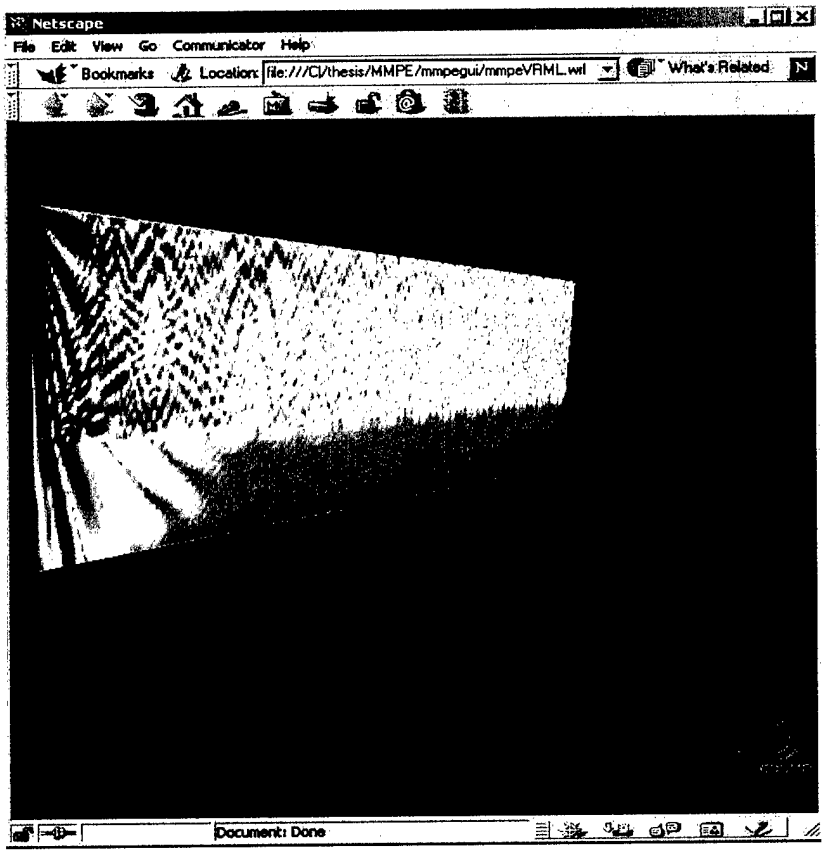


Figure 3.10 – Acoustic Field in VRML

The GUI of the next option, "Single Depth", is shown in Figure 3.11. The graphic output is produced by Eq. (2.53). It has a "Depth" section, a "Frequency" section, and three buttons. When the output file is loaded, the source depth and center frequency are set as defaults in the "Depth" and "Frequency" fields, respectively. The "Plot" button renders a graphic output of "Single Depth." Figure 3.12 shows two cases of graphic outputs. Figure 3.12 (a) shows a multi-radial acoustic field and Figure 3.12 (b) shows a single radial acoustic field. Note that if multiple radials exist in the data file, the plot automatically displays the TL field of radial versus range.

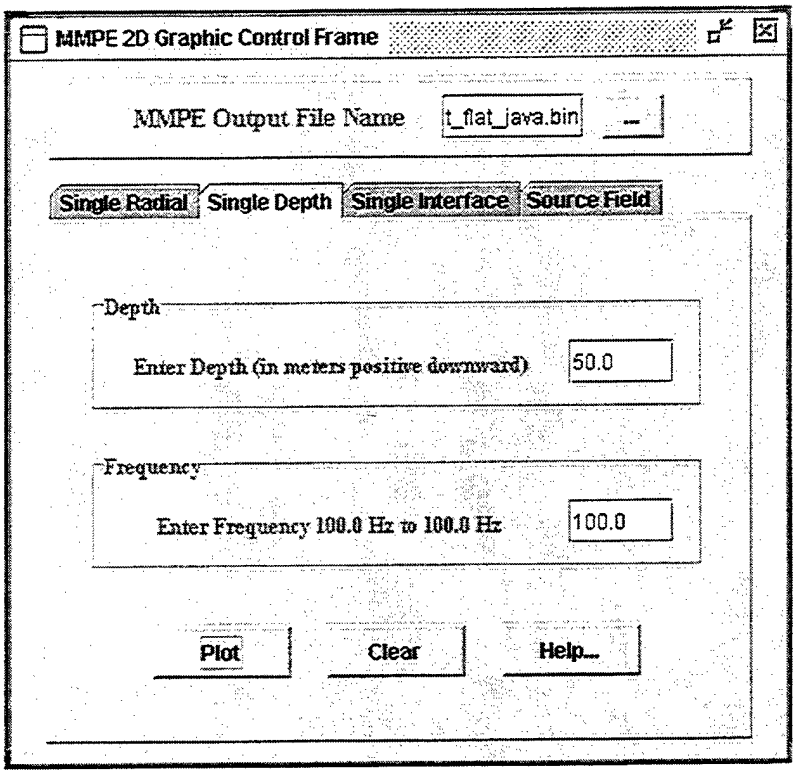
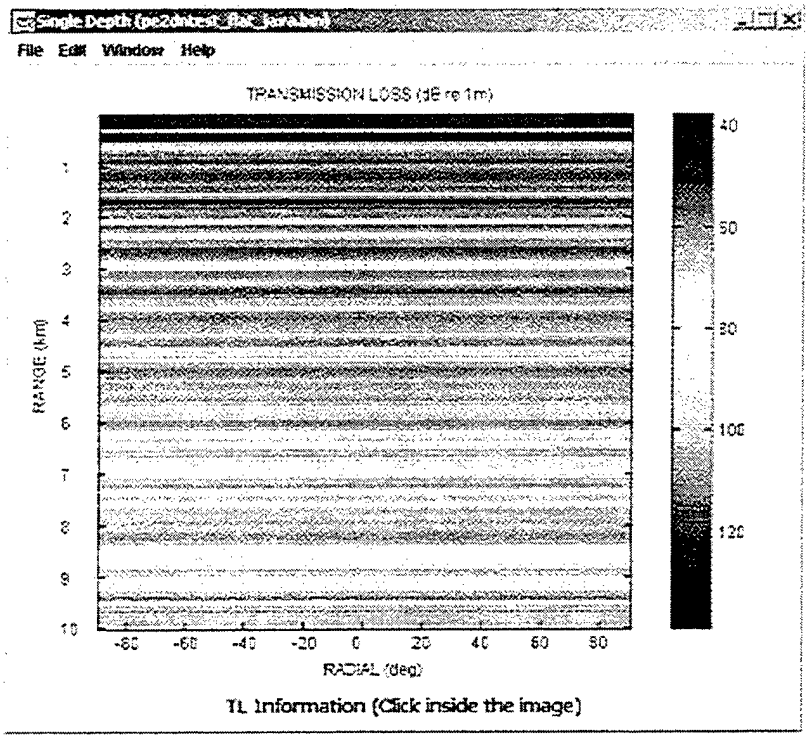
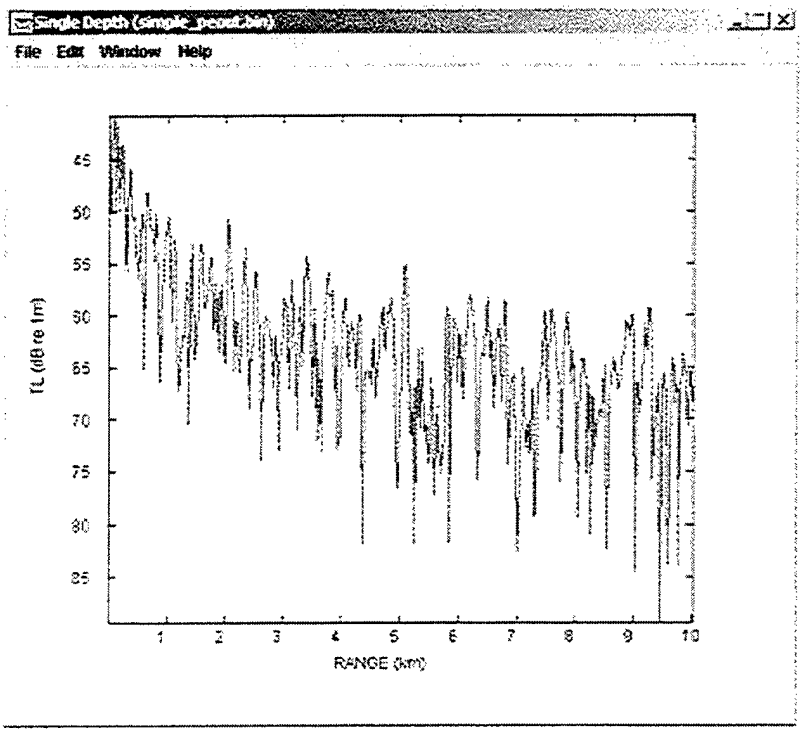


Figure 3.11 – The Option "Single Depth" in Graphic Control Frame



(a)



(b)

Figure 3.12 – Single Depth Graphic Output: (a) Multi-Radial Acoustic Field; (b) Single-Radial Acoustic Field

The GUI of the "Single Interface" is shown in Figure 3.13. The graphic output is produced by Eq. (2.53). It has a "Depth" section, a "Frequency" section, and three buttons. Unlike the "Depth" section in the "Single Depth" graphic option, it has two radio buttons for choosing one of either "Water/Bottom Interface" or "Bottom/Basement Interface," instead of an input text field. When an output file is loaded, "Water/Bottom" and center frequency are set as defaults in the "Depth" and "Frequency" fields, respectively. The "Plot" button renders a graphic output of "Single Interface." Figure 3.14 (a) shows a multi-radial acoustic field at "Water/Bottom Interface" and Figure 3.14 (b) shows a single radial acoustic field at "Bottom/Basement Interface," as shown in the title bar of each figure.

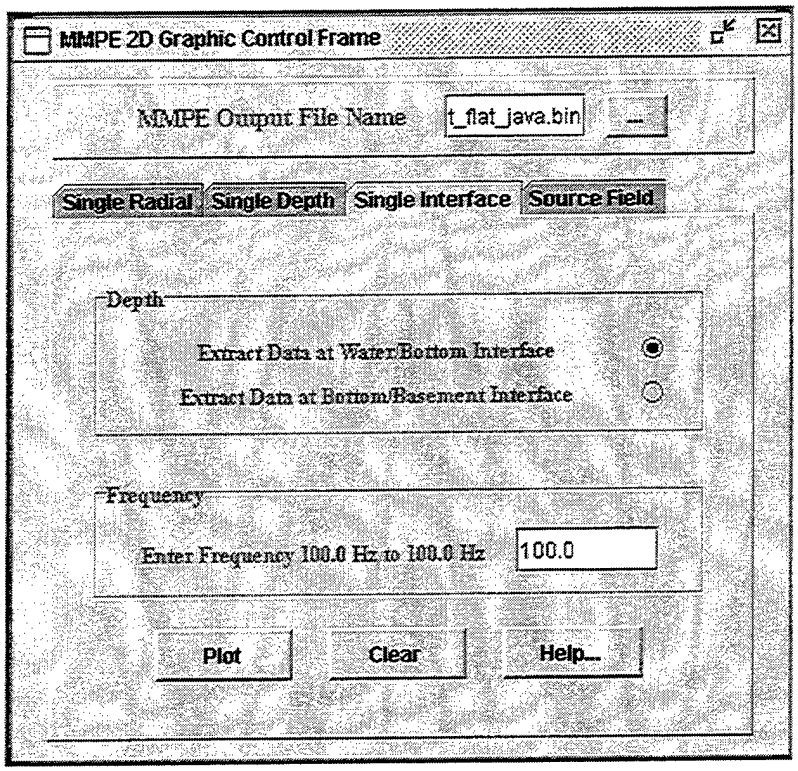
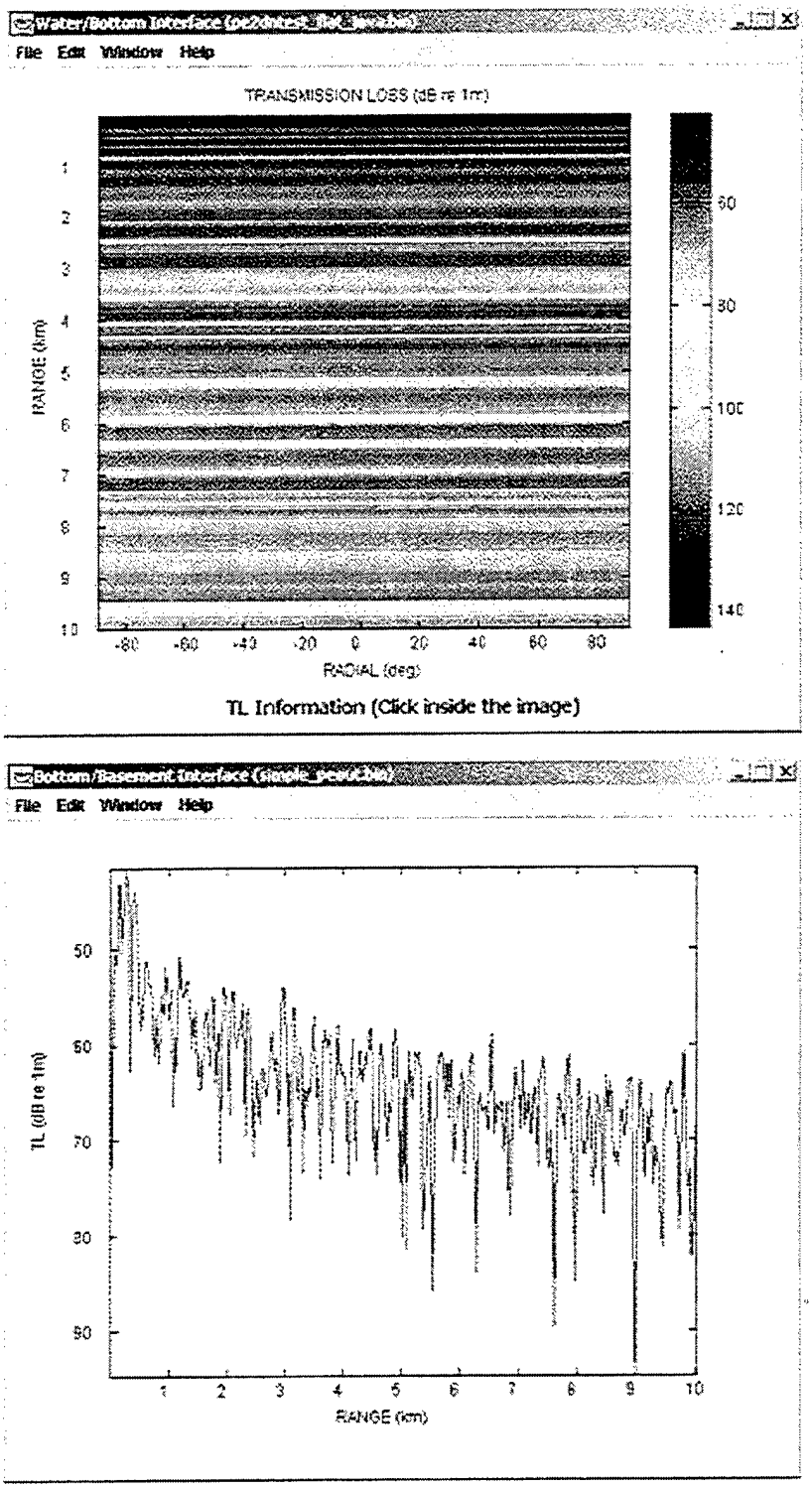


Figure 3.13 - The Option "Single Interface" in Graphic Control Frame



(a)

(b)

Figure 3.14 – Single Interface Graphic Output: (a) Multi-Radial Acoustic Field; (b) Single-Radial Acoustic Field

Figure 3.15 shows the GUI of the "Source Field." The graphic output is produced by Eq. (2.53). It has only "Plot" and "Help..." buttons. The "Plot" button renders the two kinds of graphic outputs, one of which is broadband and the other is for a single frequency data file. The former is shown in Figure 3.16 (a) and the latter is shown in Figure 3.16 (b). Note that if multiple frequencies exist in the data, this option automatically creates a frequency versus depth TL field plot.

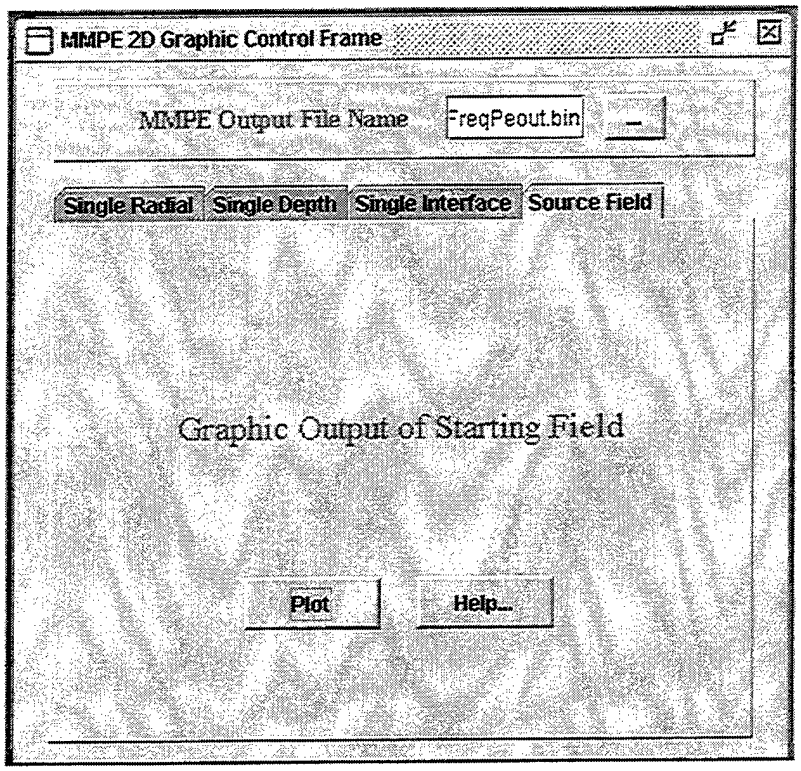
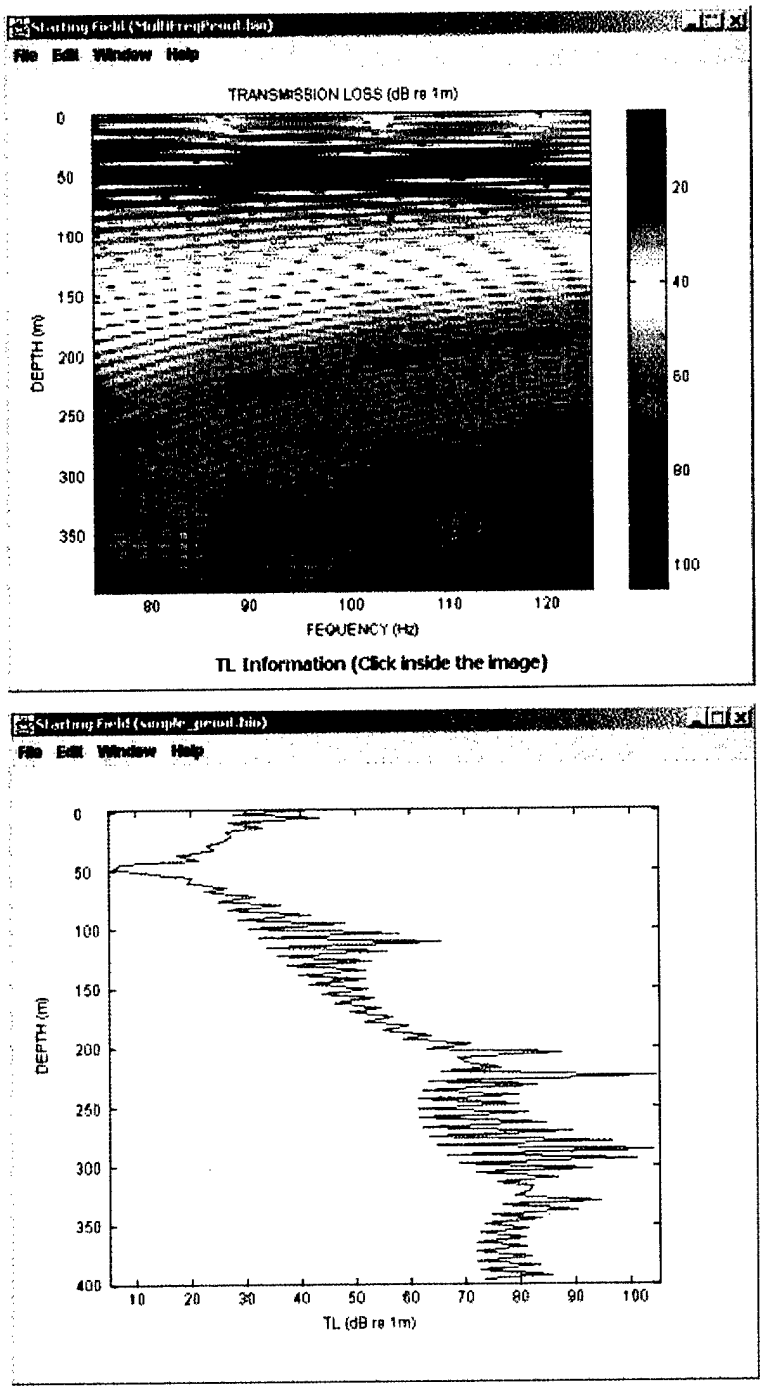


Figure 3.15 – The Option "Source Field" in Graphic Control Frame



(a)

(b)

Figure 3.16 – Source Field Graphic Output: (a) Broadband Acoustic Field; (b) Single-Frequency Acoustic Field

THIS PAGE IS INTENTIONALLY LEFT BLANK

IV. OUTPUT RESULTS

A. INTRODUCTION

In Chapters II and III, the implementation and data processing of the MMPE and WinMMPE models have been described. In this chapter, a comparison of the model output data, the values of transmission loss, and graphic outputs between MMPE and WinMMPE are made in order to confirm the accuracy of the new model. The efficiency of WinMMPE compared with MMPE is also examined. Specifically, this comparison is based upon the amount of memory used and the computational run-times.

As examples of typical cases, two sets of environmental input data files have been defined. The first case is based on a simple, range-independent Pekeris waveguide. The water column is isospeed with sound speed 1500 m/s. The bottom is at constant depth 250 m, and consists of a homogeneous, fluid half space with sound speed 1600 m/s, density 1.2 g/cm^3, and attenuation 0.15 dB/km/Hz. The source is at a depth of 100 m and transmits a CW tone of 100 Hz. The second case is based on real environmental measurements taken during the ONR-sponsored Primer experiment off the east coast of New Jersey in 1996 (Lynch et al, 1997). The source was at a depth of 270 m and the acoustic energy propagated up the continental slope onto the shelf. The acoustic frequency for this run was 400 Hz. Hereafter, the first test case (Pekeris waveguide) will be referred to as Case I, and the second test case (Primer) will be referred to as Case II.

As shown in Table 4.1, Case I has only one profile within each of the environmental input data files. The total number of environmental data values are 20. By contrast, Case II has 319,872 different sound speed profiles and 29,952 different bathymetry data points. The total number of environmental data values are 699664.

	Test Case	
Input Data Type	Case I	Case II
Sound Speed Profile Data (2 data values)	1	128 radials × 49 ranges × 51 depths
Bottom Data (2 data values)	1	128 radials × 234 ranges
Bottom Property Data (7 data values)	1	1
Deep Bottom Data (2 data values)	1	1
Deep Bottom Property Data (7 data values)	1	1

Table 4.1 – The Number of Input Data Profiles

Both the input data and specification of parameters within "pefiles.inp" decide the matrix of the output data from the model. Specifically, all radials defined by the environment and all frequencies defined in the source input file are output. However, only the number of depth and range points requested between certain limits defined within "pefiles.inp" are stored. Thus, Case I has a two-dimensional matrix, as shown in Figure 4.1 (a), of size 103×200. Case II, on the other hand, has a three-dimensional data matrix, as shown in Figure 4.1 (b), of size 128×128×200.

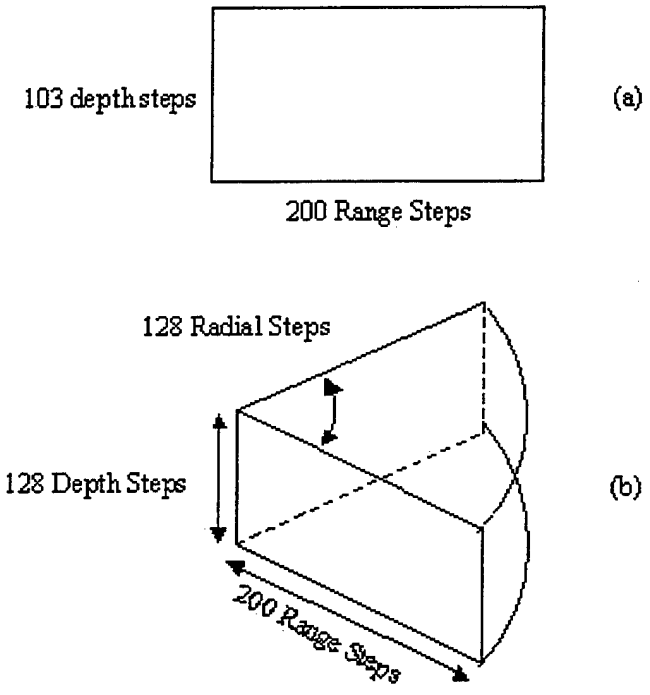


Figure 4.1 – Matrix dimensions of output data for (a) Case I and (b) Case II.

B. COMPARISON OF OUTPUT DATA

The Fortran version of MMPE primarily uses single precision floating point values with each data occupying 4 bytes. In contrast, the Java implementation of WinMMPE uses 8-byte, double precision data types. There are two reasons for using double precision. One is that the default variable type in Java is double precision. If a developer would like to use single precision variables instead, the cast operator with the "float" command must be added in the definition of the value. Another reason to use double precision is that more precise outputs can be obtained. However, the output data from both models is single precision.

This section shows the comparison of output data from the MMPE model and the WinMMPE for the given environmental input data. Some of the values to be examined will be the maximum difference of the PE field function and the maximum difference of transmission loss (TL) between MMPE and WinMMPE for each graphic. In addition, more general comparisons of the graphic outputs from each model are made.

1. Single Radial

We begin by examining the differences between single radial results. The PE field function as defined in Eq. (2.16) is a complex valued vector. As shown in Table 4.2, the maximum amplitude difference of the PE field function for Case I is 2.4780×10^{-6} while the maximum difference in TL is 0.0054 dB re 1m. These differences are negligible and can be attributed to the differences between the single precision and double precision implementations. Figure 4.2 shows the "Single Radial" graphic outputs from MMPE using the Matlab post-processing routines and WinMMPE.

In Case II, the amplitude of the maximum difference of the PE field function is 1.4950×10^{-4} while the maximum difference in TL is 7.5633 dB re 1m (Table 4.2). This large TL difference suggests there may be problems between the solutions. However, upon closer inspection, it is found that all of the significant differences occur in regions where the PE field function is extremely small, i.e. at discrete locations within the bottom

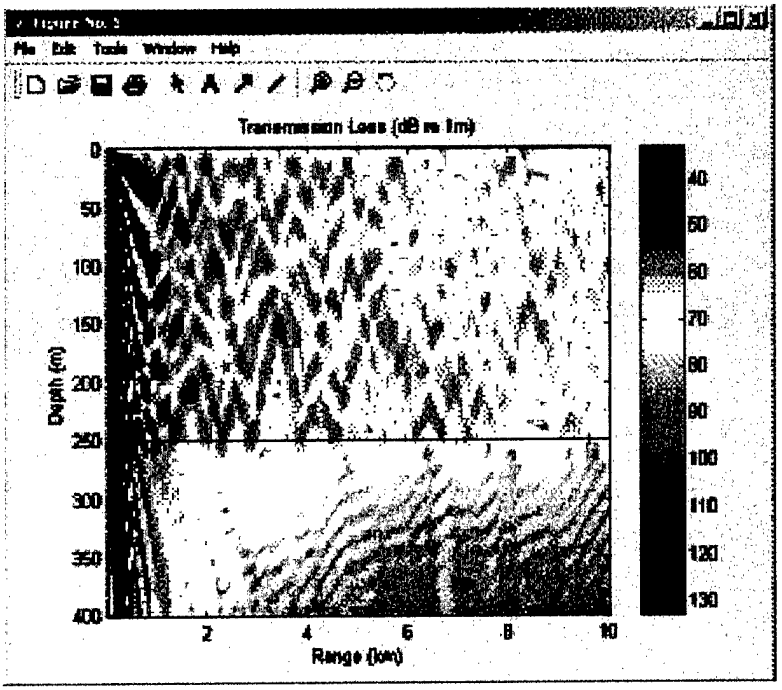
volume. Table 4.3 lists all such points in range and depth where the TL difference exceeds 3 dB. In total, there exist 8 points over 128 depth and 200 range points. Figure 4.3 shows the "Single Radial" graphic outputs from MMPE and WinMMPE along the central radial. These plots confirm that the locations of the discrepancies are in regions of extremely low field strength. Again, this could be due simply to the difference between single and double precision implementations. Overall, these plots indicate that WinMMPE is working properly and producing accurate results.

Test Case	Data	Maximum Difference (|MMPE – WinMMPE|)
Case I	PE Field Function	2.4780×10^{-6}
	TL (dB re 1m)	0.0054
Case II	PE Field Function	1.4950×10^{-4}
	TL (dB re 1m)	7.5633

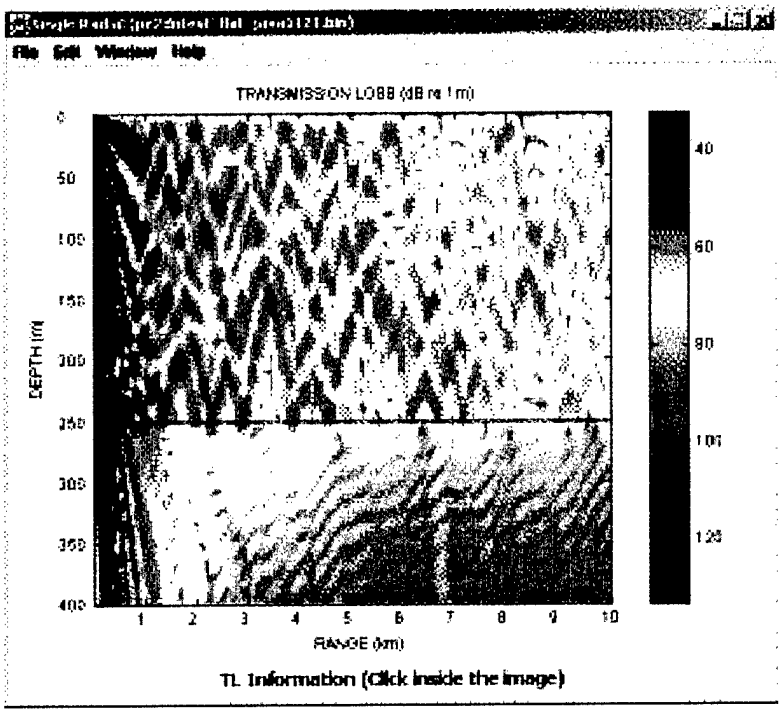
Table 4.2 – Maximum difference between the MMPE Model and the WinMMPE Model for Single Radial.

Depth (m)	Range (km)	TL (dB re 1m)
219.5313	14.3036	3.0050
341.4063	18.9741	7.5633
363.2813	20.4337	3.7791
391.4063	20.4337	4.4738
307.0313	23.9366	5.3957
297.6563	42.6188	3.0633
269.5313	46.4136	4.3954
257.0313	49.3327	3.4411

Table 4.3 – Positions where TL difference exceeds 3 dB for Case II.

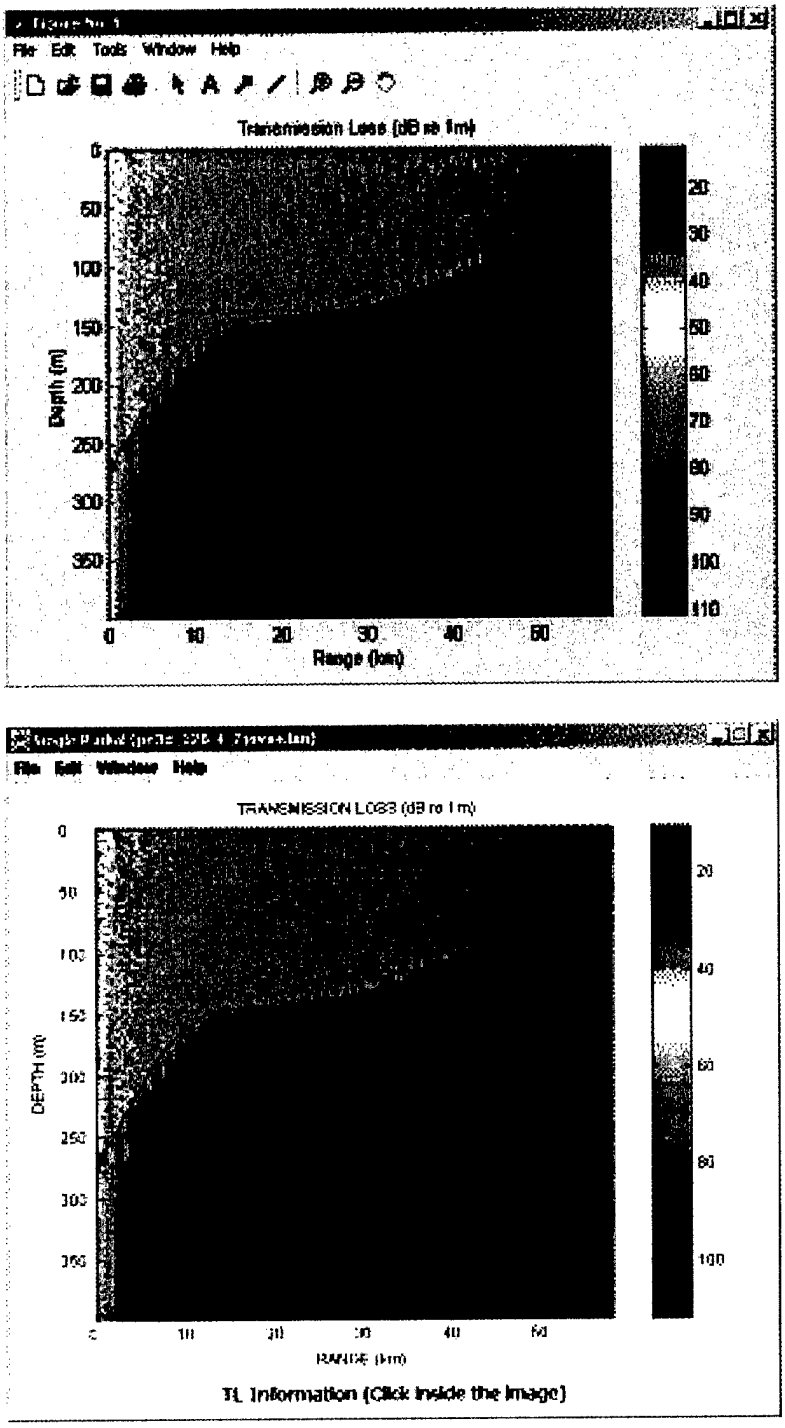


(a)



(b)

Figure 4.2 – Single Radial graphic output for Case I: (a) from MMPE; (b) from WinMMPE.



(a)

(b)

Figure 4.3 - Single Radial graphic outputs for Case II: (a) from MMPE; (b) from WinMMPE.

2. Single Depth

We now consider the data produced by selecting the single depth output option. The maximum amplitude difference of the PE field function for Case I is 2.1393×10^{-6} while the maximum difference of TL is 0.0010 dB re 1m. These values are recorded in Table 4.4. Again, these differences may be considered negligible. Figure 4.4 shows the "Single Depth" graphic outputs from MMPE and WinMMPE at the source depth.

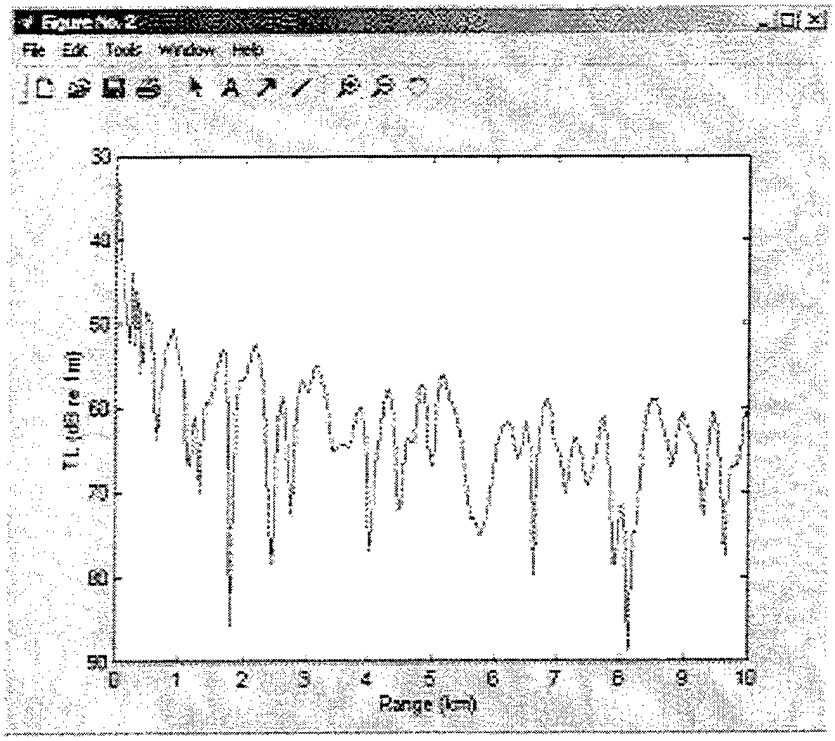
For Case II, the maximum amplitude difference of the PE field function is 0.0016 while for TL the difference is 6.5112 dB re 1m. As before, the positions in space where the TL difference exceeds 3 dB are determined and listed in Table 4.5. In this case, there exist 9 points total over 128 radial and 200 range points. Figure 4.5 shows the "Single Depth" graphic outputs from MMPE and WinMMPE at the source depth. As expected, these large differences only occur where the acoustic field has extremely low values. The similarity of the plots indicates that WinMMPE is providing accurate solutions.

Test Case	Data	Maximum Difference (|MMPE – WinMMPE|)
Case I	PE Field Function	2.1393×10^{-6}
	TL (dB re 1m)	0.0010
Case II	PE Field Function	0.0016
	TL (dB re 1m)	6.5112

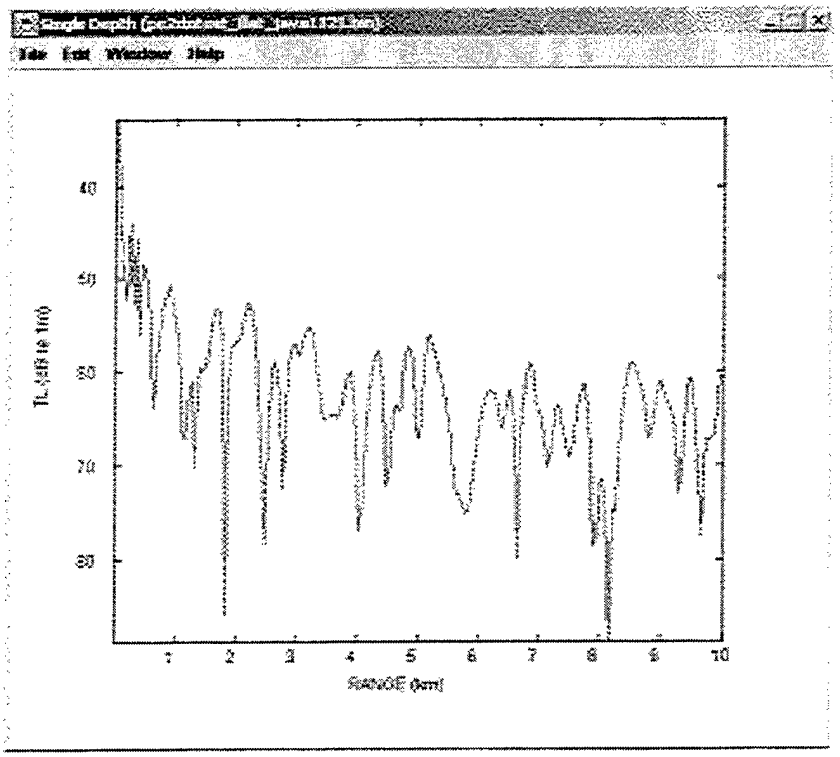
Table 4.4 – Maximum difference between MMPE and WinMMPE for Single Depth.

Radial (degree)	Range (km)	TL (dB re 1m)
-40.0781	54.0033	3.3533
-23.9063	57.5062	3.3075
-0.7031	46.7055	3.2034
0	46.4136	3.0249
11.2500	51.6680	5.6628
16.1719	27.4395	4.2953
26.0156	11.3845	6.5112
30.2344	25.6880	5.0002
40.0781	14.8874	4.9601

Table 4.5 – Positions where TL difference exceeds 3 dB for Case II.



(a)



(b)

Figure 4.4 - Single Depth graphic outputs for Case I: (a) from MMPE; (b) from WinMMPE.

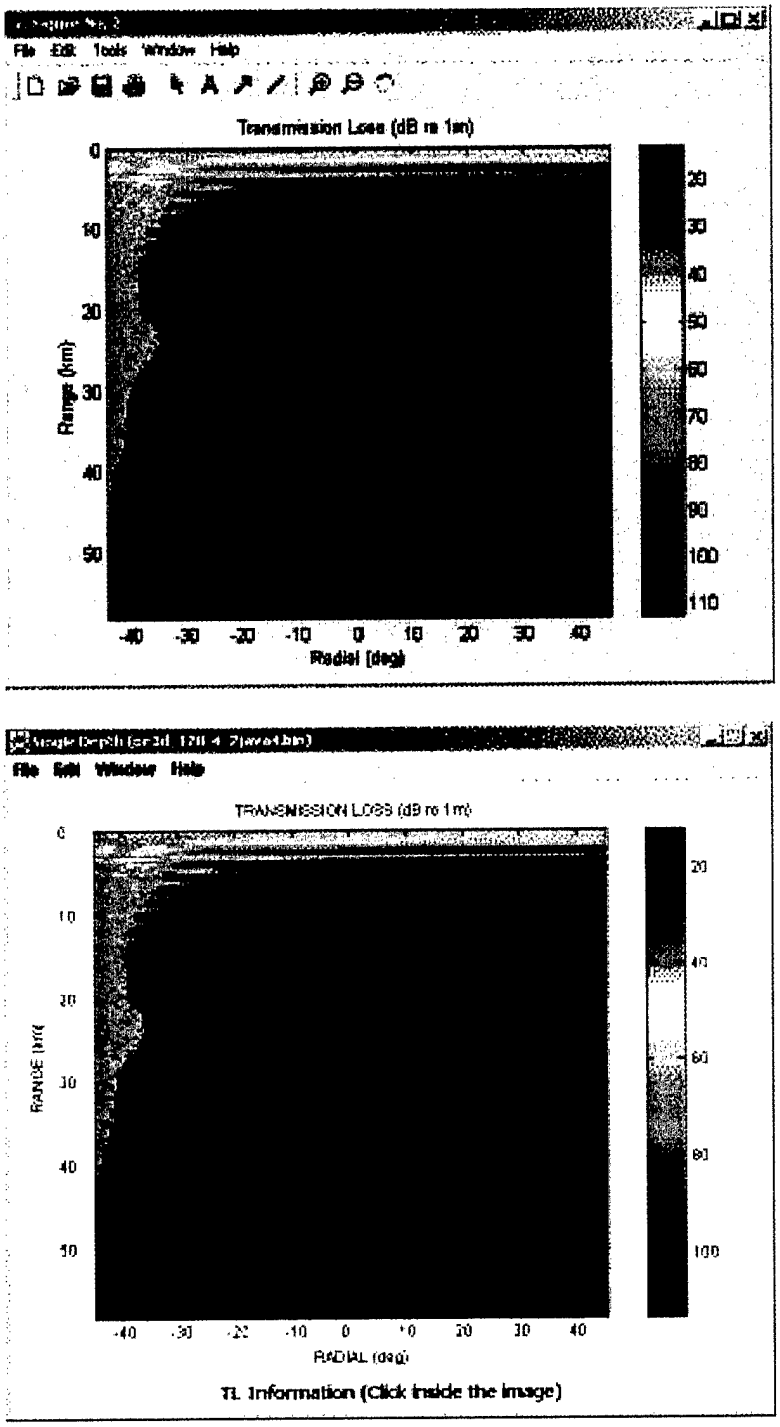


Figure 4.5 - Single Depth graphic outputs for Case II: (a) from MMPE; (b) from WinMMPE.

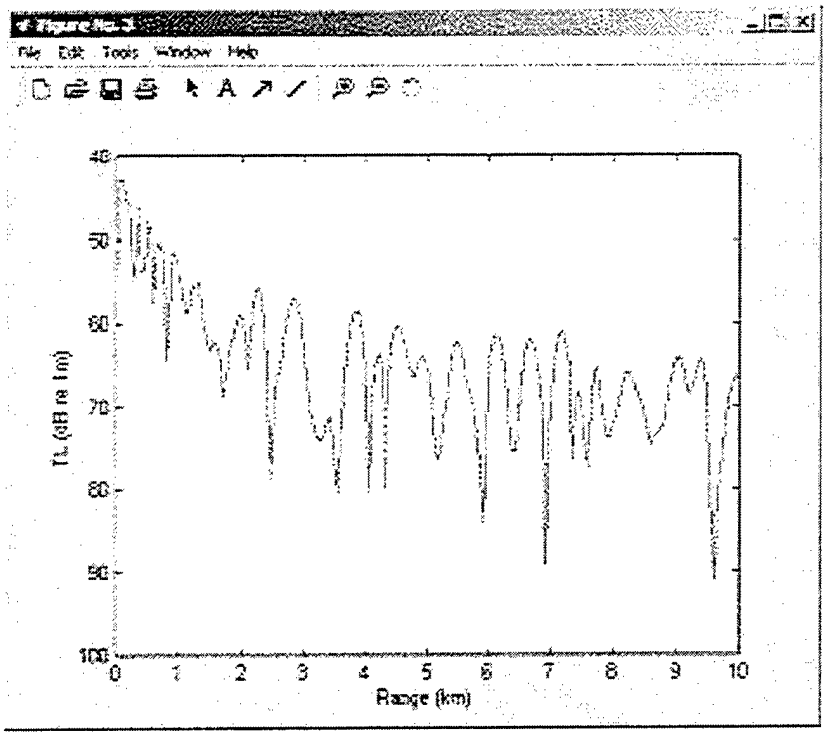
3. Single Interface

As was described in Chapter III, there are two radio buttons in the "Single Interface" graphic frame which allow the user to extract data either from the "Water/Bottom" interface or the "Bottom/Basement" interface. Because the environments examined here do not contain the deep bottom/basement interface, we will only examine the "Water/Bottom" interface option in this section. Table 4.6 shows that the maximum amplitude difference of the PE field function for Case I is 1.5451×10^{-6} while the difference in TL is 3.6884×10^{-4} dB re 1m, both of which may be considered negligible. Figure 4.6 shows the "Single Interface" graphic outputs from MMPE and WinMMPE at the "Water/Bottom" interface.

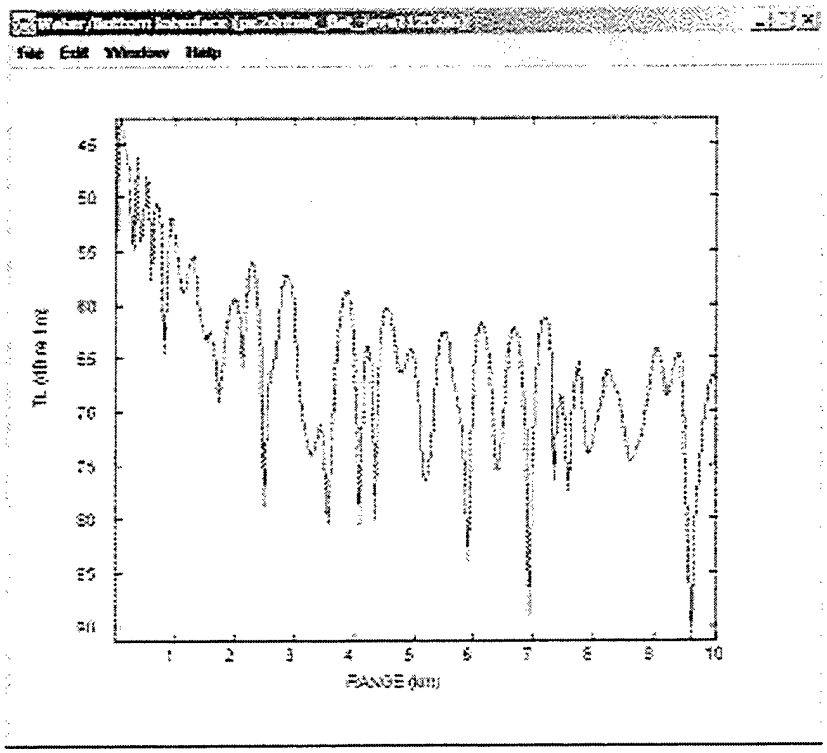
For Case II, the maximum amplitude difference of the PE field function is 0.0017 and the maximum difference in TL is 0.5484 dB re 1m, as stated in Table 4.6. Because we are explicitly examining values along the interface, we do not encounter the extremely small values of the acoustic field that were problematic before. Furthermore, the largest differences occur in the TL "nulls", which may also be affected by the double precision implementation. As a result, even these differences for Case II may be considered negligible. Figure 4.7 shows the "Single Interface" graphic outputs from MMPE and WinMMPE at the "Water/Bottom" interface.

Test Case	Data	Maximum Difference (|MMPE – WinMMPE|)
Case I	Acoustic Field Function	1.5451×10^{-6}
	TL (dB re 1m)	3.6884×10^{-4}
Case II	Acoustic Field Function	0.0017
	TL (dB re 1m)	0.5484

Table 4.6 – Maximum difference between MMPE and WinMMPE for Water/Bottom Interface.

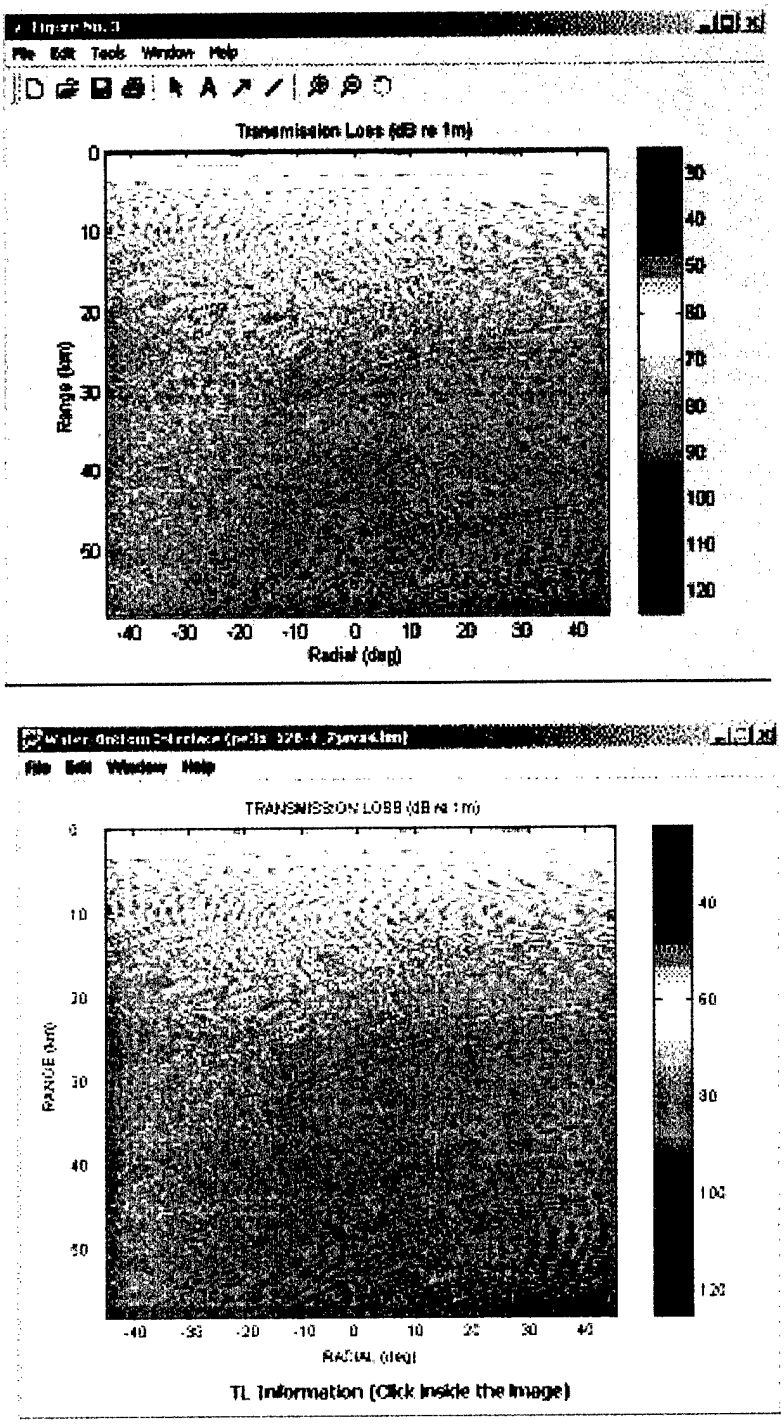


(a)



(b)

Figure 4.6 – Water/Bottom Interface graphic outputs for Case I: (a) from MMPE; (b) from WinMMPE.



(a)

(b)

Figure 4.7 – Water/Bottom Interface graphic outputs for Case II: (a) from MMPE; (b) from WinMMPE.

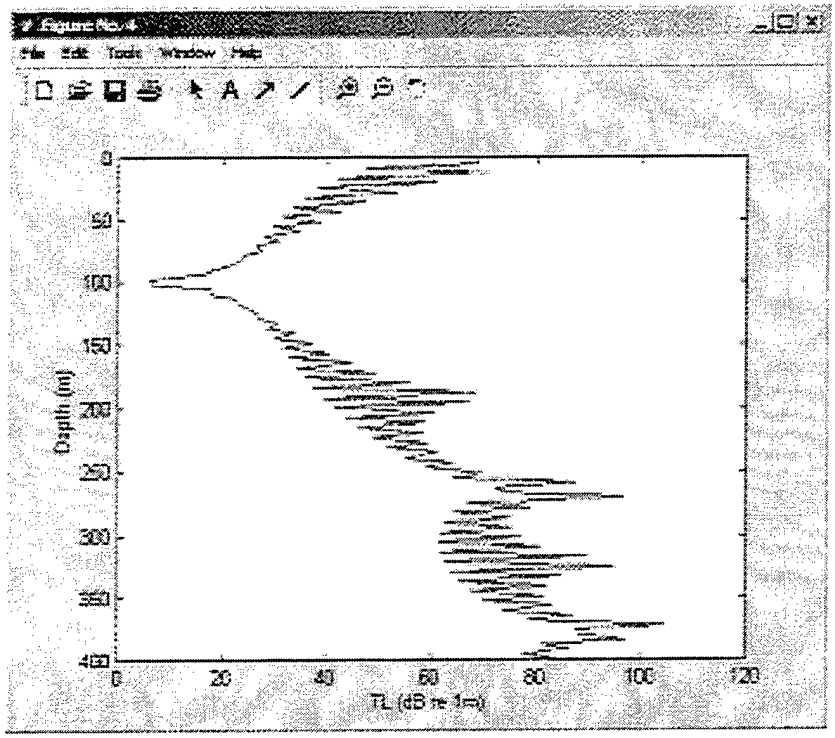
4. Source Field

We now examine the differences in the computed starting fields at zero range. Table 4.7 shows that the maximum amplitude difference of the PE field function for Case I is 2.5333×10^{-7} and the maximum difference in TL is 0.0664 dB re 1m. For Case II, the maximum amplitude difference of the PE field function is 3.1593×10^{-6} while the maximum difference in TL is 0.2548 dB re 1m. In both cases, the differences may be considered negligible.

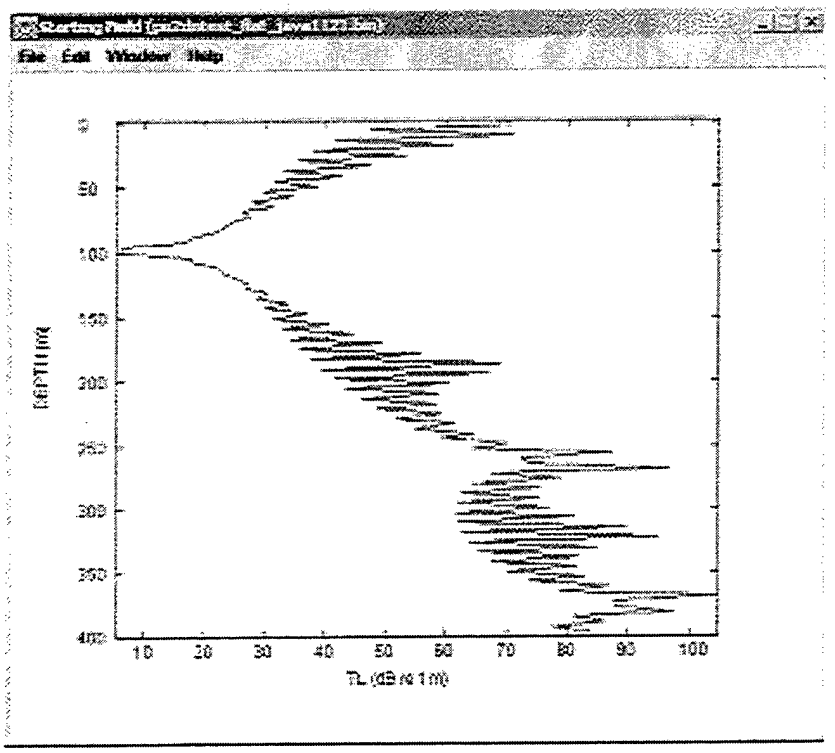
Figure 4.8 displays the "Source Field" graphic outputs from MMPE and WinMMPE at the initial range for Case I. The corresponding graphic outputs for Case II are provided in Figure 4.9.

Test Case	Data	Maximum Difference (|MMPE – WinMMPE|)
Case I	Acoustic Field Function	2.5333×10^{-7}
	TL (dB re 1m)	0.0664
Case II	Acoustic Field Function	3.1593×10^{-6}
	TL (dB re 1m)	0.2548

Table 4.7 – Maximum difference between MMPE and WinMMPE for Source Field.

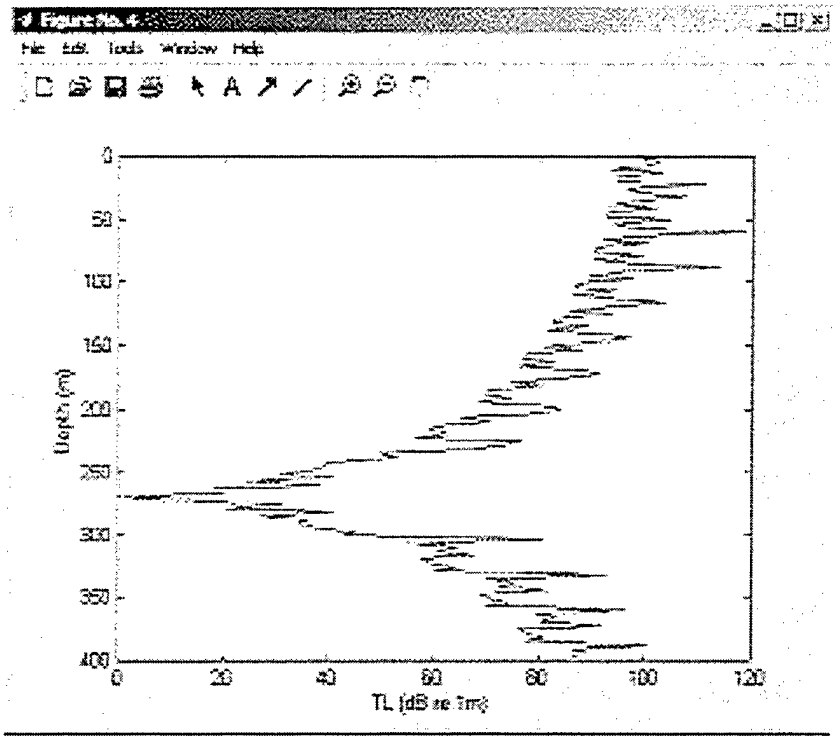


(a)

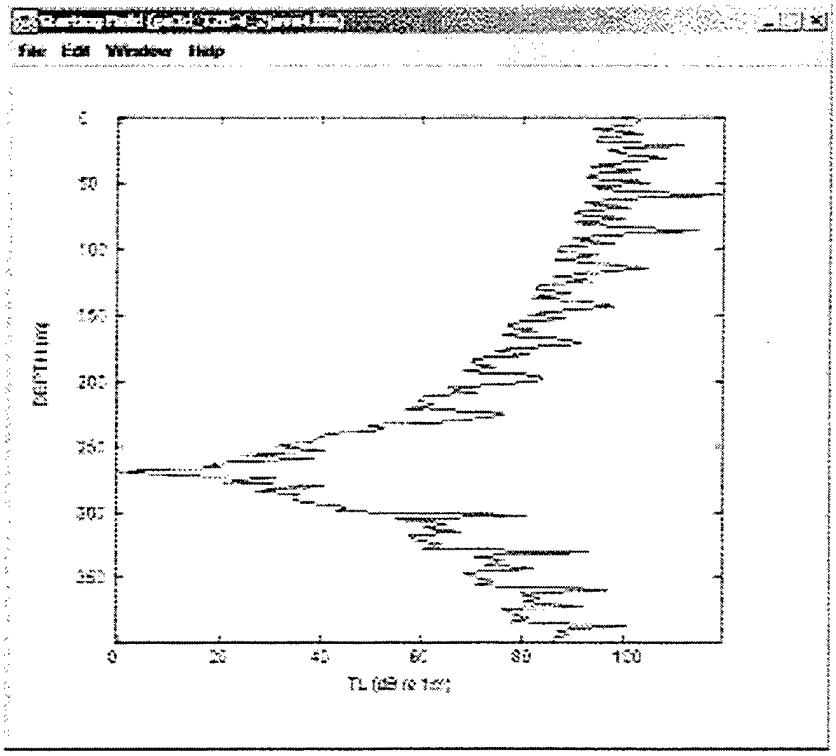


(b)

Figure 4.8 – Source Field graphic outputs for Case I: (a) from MMPE; (b) from WinMMPE.



(a)



(b)

Figure 4.9 – Source Field graphic outputs for Case II: (a) from MMPE; (b) from WinMMPE.

C. EFFICIENCY

The environment defining Case I was very simple with only a small number of input parameters. It was also range-independent, so the propagation operator only needed to be defined once and could be used throughout the calculation. Case II, on the other hand, had a very large number of environmental input parameters and was range-dependent. This required an update of the propagation operator at every range step. Therefore, it is to be expected that Case II will take longer to run than Case I. Furthermore, the amount of memory used for Case II should be larger than for Case I.

This section describes the efficiency of the MMPE Model and the WinMMPE Model as both a propagation model and a graphic tool in terms of main memory usage and running time. All testing was run on a single system with the following specifications:

- Intel Pentium II Processor, 366 MHz;
- Main Memory, 392 Mbytes;
- Operating System, Microsoft Windows 2000;
- Digital Visual Fortran Compiler, Standard Edition, Version 5.0.A;
- Matlab, Version 5.3.1 (R11.1) with Signal Processing Toolbox;
- Java2 SDK, Standard Edition, Version 1.3.0;
- Forte For Java, Community Edition, Version 1.0, Update Release 1.

The main memory on this system is adequate to avoid the need for swapping with the hard disk. Thus, although faster systems are available today, this should provide a reasonable platform to compare model performance.

1. Main Memory

As mentioned in Chapter I, programmers using standard Fortran must define the size of data arrays at the beginning of the program. This fixes the size of all arrays throughout the calculation, regardless of their usage. More recent versions of Fortran compilers can handle dynamic memory allocation, but this will not be available on all platforms. In this analysis, therefore, we will assume the Fortran program has a fixed size of data arrays for input data. If users want to use larger input data sets, the program must be re-coded to accommodate the larger data array sizes. In general, when a very simple data set is used which only occupies a fraction of the fixed allocated space, the rest of the memory is wasted.

In Java, however, programmers create the data arrays dynamically. This allows the program to allocate memory space only when it needs it and only as much as it needs. This provides for a much more efficient use of system resources than the Fortran version. Even though the Java data are 8-byte double precision variables as compared to the mostly 4-byte single precision variables in the Fortran version, this dynamic allocation of memory is expected to significantly reduce the memory requirements of the program.

The amount of memory used in each version of the model is shown in Table 4.8. These values were obtained from the "Task Manager" program of the operating system. Although these numbers do not represent exact values, they do provide a reasonable estimate of the memory usage of each model. For MMPE, both Case I and Case II roughly took 120 Mbytes. The memory usage of WinMMPE is due to both basic model functions (e.g., the GUI objects), and the dynamic allocation for run-time variables. The basic functions took about 18 Mbytes. The dynamic allocation portion took about 4 Mbytes for Case I and 17 Mbytes for Case II.

Model	Main Memory	
	Case I	Case II
MMPE (Fortran)	≈ 120 Mbytes	≈ 120 Mbytes
WinMMPE (Java)	≈ 18 Mbytes (basic) + 4 Mbytes (dynamic)	≈ 18 Mbytes (basic) + 17 Mbytes (dynamic)

Table 4.8 – Main memory used in the models for Case I and Case II.

For the graphical outputs and post-processing, the amount of memory used in WinMMPE is almost the same as that used in Matlab. This is because Matlab uses double precision and dynamically declares the size of data arrays with given sizes from the output binary file.

2. Running Time

Because the model is based on the split-step Fourier algorithm, a significant fraction of the run-time is due to the fast Fourier transforms (FFT) between depth and vertical wavenumber. In Case I, the number of depth points, i.e., number of terms in the FFT, is 1024. In Case II, the number of depth points is 2048. For each range step, two FFT's must be performed. Case I requires 222 range steps as compared to 15490 range steps for Case II. Furthermore, for multiple radials or multiple frequencies, the same number of FFT's must be performed for each range calculation. The total number of FFT's for some calculations can then become significant. Table 4.9 shows how many times the FFT module in the model is called for each case. Note that Case II has over 8900 times more FFT calls than Case I.

	Number of FFT calls	
Loop	Case I	Case II
Radials	1	128
Frequencies	1	1
Ranges × 2	444	30980
Total = Radials × Frequencies × Ranges × 2	444	3965440

Table 4.9 – Number of FFT calls in the PE/SSF algorithm for Case I and Case II.

Table 4.10 shows the run-times of the models for Case I and Case II. This measurement was achieved by watching the timer in the operating system. In some cases, it is hard to tell how long the model takes to complete, especially in the case of small input data and relatively few range steps. For Case I, the MMPE Model spent less than a second and the WinMMPE Model spent less than two seconds. Thus, it is hard to compare the true relative run-times. Case II, however, took a much longer time to run. The MMPE Model took about 3 hours and 30 minutes. In contrast, the WinMMPE Model took 6 hours and 30 minutes. It appears, therefore, that WinMMPE takes about twice as long to complete a calculation as MMPE. This factor of two can be a significant issue for very large calculations. It is quite possible that the problem is, in part, due to the Java code method of calling the "getFFT( )" in the "FFT.java" class in order to implement the FFT. The Java class was designed like the Fortran FFT subroutine in the MMPE Model, which is based on a standard numerical technique (Press, et al., 1992). It is recommended that the FFT routine be re-coded in an object-oriented programming style. This will be discussed further in Chapter V.

Model	Running Time	
	Case I	Case II
MMPE (Fortran)	< 1sec	≈ 3hrs 30min
WinMMPE (Java)	< 2 sec	≈ 6hrs 30min

Table 4.10 – Run-times of the MMPE and WinMMPE Models for Cases I and II .

The run-time of the post-processing and graphic presentation is listed in Table 4.11. For Case I, the Matlab routines spent a maximum of 2 seconds generating the "Single Radial" output while the WinMMPE Model spent only 3 seconds. The run-times for Case II are nearly the same for the two models. For the "Single Depth" graphic output, the run-time of the WinMMPE Model is typically 4 seconds. The Matlab routines, on the other hand, consistently took about 41 seconds to complete. For the "Single Interface" graphic output, WinMMPE generally took about 24 seconds while Matlab still took about 41 seconds to complete. The run-time of the other optional graphic output, "Source Field," was not significant.

It should be noted that these run-times do not account for the time taken by the user to run the propagation model and the post-processing routines separately. Time is also taken to manually enter parameter values on the command line interfaces for the MMPE Model and the associated Matlab routines, as well as the time taken to fill in the parameter fields within the WinMMPE Model. The WinMMPE Model at least has the advantage of a unified program within which GUI's exist to control the propagation model, post-processing, and graphic presentations. As a result, the operator's skill and familiarity with the programs can affect the run-times.

Graphic Tool	Run-Time	
	Case I	Case II
Matlab	Single Radial: < 2sec Single Depth: < 1sec Single Interface: < 1sec Source Field: < 1sec	Single Radial: ≈ 3sec Single Depth: ≈ 41sec Single Interface: ≈ 45sec Source Field: < 1sec
WinMMPE (Java)	Single Radial: < 3sec Single Depth: < 1sec Single Interface: < 1sec Source Field: < 1sec	Single Radial: ≈ 3sec Single Depth: ≈ 4secs Single Interface: ≈ 24secs Source Field: < 1sec

Table 4.11 – Run-times of the post-processing and graphic presentations for Case I and Case II

V. CONCLUSIONS & RECOMMENDATIONS

There are a variety of underwater acoustic propagation models available for the general user these days. These include models based on ray theory, normal modes, wavenumber integration techniques, and parabolic equation models. The ONR-sponsored web site, http://oalib.saic.com, provides copies of many current research models used in the professional community. However, these models are not usually very user-friendly for the untrained. None of the models examined provide a convenient graphical user interface and few have been developed in an object-oriented language. To address these issues and provide a more user-friendly operating environment, a Windows version of the MMPE Model, referred to as the WinMMPE Model, was developed.

The MMPE Model is really composed of two parts. The propagation component was written in Fortran, while the post-processing and graphical presentations were treated using Matlab routines. The WinMMPE Model, on the other hand, incorporated both parts into a single program written in Java. In order to compare the accuracy and efficiency of the WinMMPE Model with the previous MMPE Model, two test cases were defined. The first case was a simple, range-independent, single radial Pekeris waveguide with very few environmental input parameters. The second case was a much more complicated, range-dependent, 3D environment with multiple radials based on measured environmental data taken near the Mid-Atlantic Bight off the coast of New Jersey.

A comparison of the data indicated that WinMMPE was producing results nearly identical to MMPE. The largest differences were observed in regions where the acoustic field strength was extremely small and so could be due to simple round-off error differences. This is certainly likely since the default data type in the Java implementation of WinMMPE is double precision while the default type in the Fortran implementation of MMPE is only single precision. In regions where the acoustic field strength was large, the differences were only fractions of a dB.

A comparison of the efficiency of each model in terms of the memory usage and run-time was also presented. Due to the initialization of all variable array sizes, the Fortran-based MMPE Model used the same amount of memory for both test cases. In contrast, the Java-based WinMMPE Model dynamically allocated the necessary memory. The MMPE Model always required about 120 Mbytes, as compared to the roughly 25 - 30 Mbytes needed to run the propagation component of the WinMMPE Model. With respect to the run-times, is seemed that the WinMMPE Model generally took about twice as long to compute the propagation solution as the MMPE Model. This could be due in part to the increase from single precision to double precision. It was also suggested that the FFT routine was not well suited for implementation within Java. Finally, it was found that there was essentially no difference in run-time for the post-processing and graphic presentation of results.

There are five essential recommendations for future work. The first recommendation concerns the FFT routine. As suggested above, the FFT Java class should be re-created or replaced with an existing FFT Java open source on the Internet. There is currently no FFT Java class in the standard library provided by Sun Microsystems Corp. However, more and more class libraries are becoming available over the Internet. One of them is provided on the web site http://www.fusion.kth.se/courses/jbone. Complex classes must be combined with it as well. Therefore, some parts of the algorithm must be changed since the treatment of real and imaginary data must be combined to treat a complex variable.

Second, there are additional graphic output modules that need to be added in WinMMPE. The Matlab routines of the MMPE Model have two more options than the WinMMPE Model, specifically "Single Range" and "Travel Time". The first of these provides another view of the data to help the user evaluate the acoustical environment. The latter option is of great interest for any future development since it provides for the Fourier synthesis of broadband data to produce predictions of pulse propagation in the time domain. There are also options that allow the user to produce plane-wave beamformed results, thereby providing the user with arrival angle versus time data.

Furthermore, there are some menu items in the graphic frame that do not yet work, as mentioned in Chapter III.

Third, the problems associated with the "Input File Editor" frame were described in the "Design and Functions" section of Chapter III. Even though that frame was not a major work in this thesis, it is recommended that the frame be changed to work like the well-functioned standard Windows editor. Additionally, the functions of the menu items in the "Edit" menu of the main frame should be upgraded to work as expected.

Fourth, it would be a great service to the general acoustics community if the WinMMPE Model was available over the Internet using Java applets. The WinMMPE Model might provide useful information about the underwater acoustic environment for scientists who are not necessarily acoustics experts, or even for the general public interested in scientific modeling and visualization. It could even be used by schools to interest school kids in science and the ocean environment.

Finally, 3D visualization of underwater sound propagation can be an interesting and useful project in the future. Holliday (1998) worked on real-time 3D sonar modeling and visualization with a ray-based model. His work provided an application-programming interface (API) for real-time 3D sonar modeling and visualization of sound propagation. Future work could consider a similar 3D sonar visualization with the WinMMPE Model. Figure 3.10 was used to illustrate the scene of an acoustic field for a single radial. This simple example was not, however, a true 3D acoustic field since it only provided a single plane of 3D structures to visualize. Hopefully, Figure 3.10 can be a starting point to develop the 3D visualization of the actual 3D acoustic data provided by WinMMPE.

THIS PAGE IS INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

1. Collins, M. D., "Benchmark calculations for higher-order parabolic equations," J. Acoust. Soc. Am. 87, pp. 1535-1538, 1990.

2. Jensen, F.B., Kuperman, W.A., Porter, M.B. and Schmidt, H., *Computational Ocean Acoustics*, AIP Press, Woodbury, New York, 2000.

3. Hardin, R. H. and Tappert, F. D., "Applications of the split-step Fourier method to the numerical solution of nonlinear and variable coefficient wave equations," SIAM Rev. 15, p. 423, 1973.

4. Holliday, T.M., *Real-Time 3D Sonar Modeling and Visualization*, Master's Thesis, Naval Postgraduate School, Monterey, California, June 1998.

5. Lee, D., Botseas, G., and Papadakis, J. S., "Finite-difference solution to the parabolic wave equation," J. Acoust. Soc. Am. 70, pp. 795-800, 1981.

6. Lynch, J.F., Gawarkiewicz, G.G., Chiu, C.-S., Pickart, R., Miller, J.H., Smith, K.B., Robinson, A.R., Brink, K.H., Beardsley, R., Sperry, B., and Potty, G., "Shelfbreak PRIMER – An integrated acoustic and oceanographic field study in the Middle Atlantic Bight," Proceedings of International Conference on Shallow Water Acoustics, Beijing, China, 21-25 April, pp. 205 – 212, 1997.

7. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P., *Numerical Recipes in FORTRAN: The Art of Scientific Computing*, Cambridge University Press, New York, New York, pp. 498-504, 1992.

8. Smith, K.B. and Tappert, F.D., "UMPE: The University of Miami Parabolic Equation Model, Version 1.0," Marine Physical Laboratory Technical Memo 432, 1993.

9. Smith, K.B., "Convergence, stability, and variability of shallow water acoustic predictions using a split-step Fourier parabolic equation model," Proceedings of the Shallow Water Acoustic Modeling (SWAM `99) Workshop, 8–10 September 1999, J. Comp. Acoust. (Special Issue, in press), 2000.

10. Tappert, F. D., "The parabolic approximation method," in Lecture Notes in Physics, Vol. 70, *Wave Propagation and Underwater Acoustics* (edited by Keller, J. B. and Papadakis, J. S.), Springer-Verlag, New York, pp. 224–287, 1977.

11. Thomson, D.J. and Chapman, N.R., "A wide-angle split-step algorithm for the parabolic equation," J. Acoust. Soc. Am., Supplement Vol. 83, S118, 1983.

12. Thomson, D. J. and Bohun, C. S., "A wide-angle initial field for the parabolic equation models," J. Acoust. Soc. Am. 83, p. S118, 1988.

13. Tindle, C. T. and Zhang, Z. Y., "An equivalent fluid approximation for a low shear speed ocean bottom," J. Acoust. Soc. Am. 91, pp.3248-3256, 1992.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5101

3. Prof. Don Brutzman, Code PH/Sk 2
   Department of Physics
   Naval Postgraduate School
   Monterey, CA 93943-5002

4. Prof. Kevin B. Smith, Code PH/Sk 3
   Department of Physics
   Naval Postgraduate School
   Monterey, CA 93943-5002

5. Dr. Jeff Simmen (Code 321OA) 1
   Office of Naval Research
   800 N. Quincy Street
   Arlington, VA 22217

6. Dr. Philip S. Barry 1
   Chief, S&T Initiatives Division
   Defense Modeling and Simulations Office
   1901 N. Beauregard Street, Suite 500
   Alexandria, VA 22311

7. Robert J Barton III 1
   Fraunhofer Center for Research in Computer Graphics (CRCG)
   321 South Main Street
   Providence, RI 02903

8. Erik Chaum 1
NAVSEA Undersea Warfare Center
Division Newport
Code 2231, Building 1171-3
1176 Howell Street
Newport, RI 02841-1708

9. John Lademan 1
Electronic Sensors and Systems Sector
Northrop Grumman Corporation
PO Box 1488 – MS 9030
Annapolis, MD 21404

10. CAPT Arnold O. Lotring, USN 1
Commanding Officer
Naval Submarine School
Code 00, Naval Submarine School
Post Office Box 700
Groton, CT 06349-5700

11. CAPT John C. Mickey, USN 1
PEO for Submarines (PMS401)
2531 Jefferson Davis Highway
NC3, Room 3W30
Arlington, VA 22242-5161

12. CAPT Don Gerry USN 1
Deputy, Submarine Development Squadron TWELVE
Naval Submarine Base New London
Groton CT 06340

13. George Phillips 1
CNO, N6M1
2000 Navy Pentagon
Room 4C445
Washington, DC 20350-2000

14. Jung, Dusan 5
230 Tunisia Rd.
Seaside, CA 93955