NUMERICAL RECOVERY OF THE LAYERED MEDIUM FROM THE SURFACE DATA

by

PEIQING LI

B.S., Xi'an Jiaotong University, 1984

A MASTER'S THESIS

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Mathematics
KANSAS STATE UNIVERSITY
Manhattan, Kansas

1987

Approved by:

A. Ramm

Major Professor

LD
2668
.T4
MATH
1987
M5
c. 2

CONTENTS

A11207 309024

	Acknowledgements	3
I.	Formulation of the problem	4
II.	The description of the algorithm	6
III.	Numerical results and practical recommendations	9
IV.	Additional remarks	11
	References	12
	Appendix 1	13
	Appendix 2	16
	Appendix 3	17
	Appendix 4	18
	Table 1	23
	Table 2	24
	Table 3	25
	Table 4	26
	Table 5	27
	Table 6	28
	Abstract	

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to Dr. A. G. Ramm for his guidance and advice throughout my graduate program. He has shown me a broad and interesting field of applied mathematics.

I would like to thank Dr. W. D. Curtis and Dr. F. R. Miller for their help.

I would also like to thank ONR for support.

Finally, I wish to thank Reta Williams for her patient and expert typing of this work.

I. Formulation of the problem..

In geophysical prospecting and seismology there are many problems which are governed by the equation:

$$\Delta u(x) + \omega^2 n(z) u(x) = -\delta(x) \tag{1}$$

where $x = (x^1, x_3) \in \mathbf{R}^3$, $x^1 = (x_1, x_2) \in \mathbf{R}^2$, $z = x_3 < 0$, $\omega > 0$. In acoustic prospecting, $\delta(x)$ is a point source situated at the origin, $u(x)$ is the pressure, ω is the wave frequency, and $c(z) = n^{-1/2}(z)$ is the velocity of the wave at the depth z. The inverse problem is to find $n(z)$ from the given data: $u(x, \omega)$, where $x \in P := \{x : x_3 = 0\}$ and $0 < \omega \leq \omega_0$, ω_0 is a fixed small frequency. Let us give a sketch of our approach following [1].

Fourier transform (1) in x^1 to get

$$\tilde{u}'' - \lambda^2 \tilde{u} + q(z)\tilde{u} = -\delta(z),$$

where

$$\begin{aligned} q(z) &= \omega^2 n(z), \\ \tilde{u}' &= d\tilde{u}/dz, \\ \tilde{u}(\lambda, z) &= \int_{-\infty}^{+\infty} e^{-i\lambda \cdot x^1} u(x^1, z) dx^1, \\ \lambda &= (\lambda_1, \lambda_2), \\ \lambda^2 &= \lambda_1^2 + \lambda_2^2. \end{aligned}$$

Assume that $\lambda^2 > q_0 := \omega^2 \max_z |n(z)|$, then the integral equation

$$\tilde{u}(\lambda, z) = g(\lambda, z) + \int_{-\infty}^{+\infty} g(\lambda, z - z') q(z') \tilde{u}(\lambda, z') dz' \tag{2}$$

where $g(\lambda, z) = \frac{e^{-\lambda|z|}}{2\lambda}$, is uniquely solvable by iterations, and $\tilde{u}$ is analytic in λ in the region $Re\lambda > q_0^{1/2}$. Note that

$$\tilde{u}(\lambda, z) = g(\lambda, z) + \omega^2 \int_{-\infty}^{+\infty} g(\lambda, z - z') n(z') g(z') dz' + O(\omega^4), \text{as} \quad \omega \to 0.$$

Therefore

$$f(z, \lambda) = \lim_{\omega \to 0} (\tilde{u} - g)\omega^{-2} = \int_{-\infty}^{+\infty} \frac{e^{-\lambda|z-z'|}}{2\lambda} n(z') \frac{e^{-\lambda|z'|}}{2\lambda} dz'.$$

This low frequency limit was used in [2], [3]. Set $z = 0$, then

$$F(\lambda) := 4\lambda^2 f(0,\lambda) = \int_{-\infty}^{+\infty} e^{-2\lambda|s|} n(z)dz, \quad \lambda > 0. \tag{3}$$

Since the data are given on the plane $P := \{x : x_3 = 0\}$, $\tilde{u}(\lambda,0)$ and $F(\lambda)$ are known. Therefore the inverse problem is reduced to finding $n(z)$ from the knowledge of $F(\lambda)$.

Assume that $n(z) = 1$ for $z > 0$ and $n(z) = n_0 = \text{const}$ for $z < -d$, where d is a certain depth. Then (3) becomes

$$\int_{-d}^{0} e^{-2\lambda|s|} n(z)dz = \psi(\lambda) \tag{4}$$

where $\psi(\lambda) \equiv F(\lambda) - \frac{1}{2\lambda} - \frac{n_0 e^{-2\lambda d}}{2\lambda}$. A change of variables, $2\lambda = p$, $z = -t$, transforms (4) into

$$\int_0^d e^{-pt} h(t)dt = \Phi(p), \quad p > 0$$

where $h(t) \equiv n(-t)$, $\Phi(p) \equiv \psi(\frac{p}{2})$. The function $h(t)$ can be found by the method given in [3], [4]. The problem of inverting the Laplace transform of a compactly supported function from the real axis is solved analytically in [5].

We assume that $n(z) = 1$ for $z > 0$ (above the ground), $n(z) = n_0$, for $z < -d$, $n_0 > 0$, $d > 0$, and $n(z) = n_j$ for $z_j < z < z_{j+1}$, $1 \le j \le m$, $z_1 = -d$, $z_{m+1} = 0$. So there are m homogeneous layers in the region $-d < z < 0$. Therefore (4) yields

$$\sum_{j=1}^{m} n_j [e^{-2\lambda|z_{j+1}|} - e^{-2\lambda|z_j|}] = \phi(\lambda) := 2\lambda\psi(\lambda), \quad \lambda > 0. \tag{5}$$

This problem is ill-posed, it is very sensitive to noise in the data. One wishes to find an efficient way to recover parameters $\{n_j\}_{j=1}^m$, $\{z_j\}_{j=1}^m$ from the knowledge of $\phi(\lambda)$. In this paper, we give a numerical solution to this problem and demonstrate its efficiency. We assume in the beginning that the number m of layers is known. In section IV a method is given for estimating this number from the data.

II. The description of the algorithm.

Let $\{\phi(\lambda_i)\}_{i=1}^N$ be given, where $\{\lambda_i\}_{i=1}^N$ is a set of positive numbers. Replacing λ by λ_i for $i = 1, 2, \ldots, N$ in (5) yields

$$\sum_{j=1}^{m} n_j [e^{-2\lambda_i |x_{j+1}|} - e^{-2\lambda_i |x_j|}] = \phi(\lambda_i), \quad i = 1, 2, \ldots, N. \tag{6}$$

This is a nonlinear system for $\{n_j\}_{j=1}^m$ and $\{x_j\}_{j=1}^m$. It can be solved by an optimization method. Let us describe the method. Set

$$\tilde{\phi}_i(w_1, w_2, \ldots, w_m, y_1, y_2, \ldots, y_m) := \sum_{j=1}^{m} w_j [e^{-2\lambda_i y_{j+1}} - e^{-2\lambda_i y_j}],$$

$$G(w_1, w_2, \ldots, w_m, y_1, y_2, \ldots, y_m) := \sum_{j=1}^{N} f_i^2(w_1, w_2, \ldots, w_m, y_1, y_2, \ldots, y_m),$$

$$f_i = \tilde{\phi}_i - \phi(\lambda_i), i = 1, 2, \ldots, N.$$

We want to find $\{n_j\}_{j=1}^m$ and $\{x_j\}_{j=1}^m$ by minimizing G, the target function. Note that here the variables $w_1, w_2, \ldots, w_m, y_1, y_2, \ldots, y_m$ are not free variables. They satisfy the constraints:

$$\left.\begin{aligned} g_j &:= w_j > 0, \\ g_{j+m} &:= y_j > 0, \\ g_{j+2m} &:= y_{j+1} - y_j > 0, \end{aligned}\right\} j = 1, 2, \ldots, m. \tag{7}$$

Introduce parameters M_j, $j = 1, 2, \ldots, 3m$, then the problem is reduced to the following non-constrained optimization problem:

$$F(w_1, w_2, \ldots, w_m, y_1, y_2, \ldots, y_m) = \min$$

where $F = G + \sum_{i=N+1}^{N+3m} f_i^2$, $f_{i+N} = M_i |\min(0, g_i)|^2$, $i = 1, 2, \ldots, 3m$, are exterior penalty functions [6], and M_i, $i = 1, 2, \ldots, 3m$, are given large positive numbers chosen so that $f_i = 0$, $i = 1, 2, \ldots, 3m$, at the minimizer of F; therefore conditions (7) are satisfied at this minimizer.

Newton's method for minimizing $F(x)$ is

$$x^{(k+1)} = x^{(k)} - [F''(x^{(k)})]^{-1} \cdot \nabla F(x^{(k)})$$

$$x = (x_1, x_2, \ldots, x_{2m}) = (w_1, w_2, \ldots, w_m, y_1, y_2, \ldots, y_m).$$

Here $\nabla F(x)$ is the gradient of F, and F'' is the matrix $\left(\frac{\partial^2 F}{\partial x_i \partial x_j}\right)$, $i = 1,2,\ldots,$ $2m$, $j = 1,2,\ldots,2m$. Newton's method has quadratic convergence near a local minimizer. But it requires calculation of second derivatives of the target function. One can estimate the second derivatives of $F(x)$ by the first derivatives of $f_i(x)$, $i = 1,2,\ldots,M := N + 3m$.

Let $x^{(k)}$ be an approximation of the minimizer x^* of F. To construct $x^{(k+1)}$, we approximate $f_i(x)$ by the linear functions

$$\ell_i^{(k)} := f_i(x^{(k)}) + \sum_{j=1}^{2m} J_{ij}(x^{(k)}) \cdot (x_j - x_j^{(k)})$$

$$J_{ij}(x) = \frac{\partial f_i(x)}{\partial x_j}, i = 1,2,\ldots,M, j = 1,2,\ldots,2m.$$

So in place of $F(x)$, one minimizes $\sum_{i=1}^{M} (\ell_i^{(k)}(x))^2$. Let us denote this function by the same letter $F(x)$. Then

$$F''(x^{(k)}) = 2(J^{(k)})^T(J^{(k)}),$$

$$J^{(k)} = (J_{ij}(x^{(k)})), i = 1,2,\ldots,M, j = 1,2,\ldots,2m$$

$$\nabla F(x^{(k)}) = 2(J^{(k)})^T f(x^{(k)})$$

$$f(x) = \begin{pmatrix} f_1(x) \\ f_2(x) \\ \cdots \\ f_M(x) \end{pmatrix}.$$

Thus the iterative procedure is

$$x^{(k+1)} = x^{(k)} - ((J^{(k)})^T(J^{(k)}))^{-1}(J^{(k)})^T f(x^{(k)}). \tag{8}$$

This scheme is easy to implement and our results show its efficiency. Note that $(J^{(k)})^T(J^{(k)})$ is always ill-conditioned when $x^{(k)}$ is sufficiently close to a minimizer, and $(J^{(k)})^T(J^{(k)})$ sometimes is ill-conditioned even when $x^{(k)}$ is not close to a minimizer. To overcome this, we change the search direction a little, so the modified procedure is

$$x^{(k+1)} = x^{(k)} - [(J^{(k)})^T(J^{(k)}) + \alpha_k I]^{-1}(J^{(k)})^T f(x^{(k)}) \tag{9}$$

where $\alpha_k > 0$ is a parameter. The choice of α_k is described in the outline of the program below; α_k is stored in the cell named α. This is Marquardt's method which is a combination of the least squares method and the gradient method.

Let us outline the algorithm:

One starts with an initial point $x^{(0)}$, the data $\{\lambda_i\}_{i=1}^{N}$, $\{\phi(\lambda_i)\}_{i=1}^{N}$, m, a positive integer, ϵ, a small positive number, d, a positive number, parameters $\alpha_0 > 0$, $\nu > 0$, $\{M_j\}_{j=1}^{3m}$, each M_j is a positive large number. IW is a count number, the meaning of which can be seen clearly from the following outline.

Step 1 : $0 \Rightarrow k$, $\alpha_0 \Rightarrow \alpha$.

Step 2 : $-1 \Rightarrow IW$, $\alpha/\nu \Rightarrow \alpha$, calculate:

$$f(x^{(k)}) = (f_1(x^{(k)}), f_2(x^{(k)}), \ldots, f_M(x^{(k)}))^T,$$

$$F(x^{(k)}) = \sum_{i=1}^{M} f_i^2(x^{(k)})$$

$$J^{(k)} = \begin{pmatrix} \frac{\partial f_1(x^{(k)})}{\partial x_1} & \frac{\partial f_1(x^{(k)})}{\partial x_2} \ldots \frac{\partial f_1(x^{(k)})}{\partial x_{2m}} \\ \frac{\partial f_2(x^{(k)})}{\partial x_1} & \frac{\partial f_2(x^{(k)})}{\partial x_2} \ldots \frac{\partial f_2(x^{(k)})}{\partial x_{2m}} \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ \frac{\partial f_M(x^{(k)})}{\partial x_1} & \frac{\partial f_M(x^{(k)})}{\partial x_2} \ldots \frac{\partial f_M(x^{(k)})}{\partial x_{2m}} \end{pmatrix}.$$

Step 3 : Calculate:

$$P_k = -[(J^{(k)})^T \cdot J^{(k)} + \alpha_k I]^{-1} (J^{(k)})^T f(x^{(k)}),$$
$$x^{(k+1)} = x^{(k)} + P_k \quad \text{and} \quad F(x^{(k)}),$$

where α_k is a real number in cell α.

Step 4 : If $F(x^{(k+1)})$ is less than $F(x^{(k)})$, then pass to the sixth step, otherwise do the next step.

Step 5 : $IW + 1 \Rightarrow IW$, $\alpha\nu \Rightarrow \alpha$. If $\| (J^{(k)})^T f(x^{(k)}) \| \leq \epsilon$, then pass to the seventh step, otherwise go back to the third step.

Step 6 : If $\| (J^{(k)})^T f(x^{(k)}) \| \leq \epsilon$, then go to the seventh step; otherwise, let $k+1 \Rightarrow k$, $x^{(k+1)} \Rightarrow x^{(k)}$, and go back to the second step.

Step 7 : The execution stops.

Since there is no guarantee that the minimizer found by this algorithm is a global one, we demand that the optimal point x^* should satisfy two inequalities $\| (J(x^*))^T f(x^*) \| \leq \epsilon$ and $F(x^*) \leq \epsilon_1$ to be a global minimizer. Here $\epsilon_1 > 0$ is a small number and $\epsilon > 0$ is the number given at the start.

III. Numerical results and practical recommendations.

Rewrite (5) as

$$n_m + (n_{m-1} - n_m)e^{-2\lambda x_m} + (n_{m-2} - n_{m-1})e^{-2\lambda x_{m-1}} + \cdots + (n_1 - n_2)e^{-2\lambda x_2} - n_1 e^{-2\lambda d} = \phi(\lambda).$$

Suppose that there is a large enough λ_{i_0} among $\{\lambda_i\}_{i=1}^N$, then n_m is approximately equal to $\phi(\lambda_{i_0})$. Thus n_m is $\phi(\lambda_{i_0})$, a known quantity, in the numerical examples below. When the data are noiseless, parameters $n_{m-1}, n_{m-2}, \ldots,$ and $\{z_j\}_{j=1}^m$ can be determined by an asymptotic procedure of the peeling off type [3, Appendix 7]. However, this asymptotic approach is not easy to use because of the rapid accumulation of round-off errors and its sensitivity to noise.

It is obvious that one needs some a priori assumptions about parameters $\{n_j\}_{j=1}^m$ and $\{z_j\}_{j=1}^m$. There are three typical cases with noise-free synthetic data or noisy data for which the recovery may be difficult to carry out. The first case is when some layers are very deep, the second is when some layers are very thin, and the third is when two adjacent layers have almost the same parameters n_j. Hence the numerical examples are mainly focused on these difficult cases, yet a few intermediate cases are also presented for comparison.

The results obtained are presented in six tables. Tables 1, 3 and 5 are for the two, three and four layers with noise-free data. Tables 2, 4 and 6 are for two, three and four layers with different noise levels in the data. The results show that the algorithm is applicable to a large range of layered structures and is efficient. In the numerical examples the quantities n_j, λ_j, and z_j are dimensionless. For example $n_j = 2$ means that one has chosen some unit of n_j and $n_j = 2 \times$ (unit of n_j). For a practical problem, if one has a scale system under which the value of each variable falls into the range we describe below, then one can apply the method to this problem.

First, if the thicknesses of layers are not too small and the depths of layers are not too large, then recovery is reasonably accurate. Specifically, we require that the thickness of each layer should not be less than 0.2 and the depth of each layer should not be more than 20. Under these assumptions, the recovery is accurate even for noisy data, the majority of which have noise/signal ratio less than 0.01. The case when layers are thin and deep is more difficult than that when layers are thin and shallow. Also one will not have much difficulty to recover the parameters

$\{n_j, z_j\}$ if there is no big difference in the thicknesses of layers even though they are very thin or they are deep down in the earth.

Secondly, although one can divide both sides of (5) by any number to make $\{n_j\}_{j=1}^m$ larger, the level of noise in $\phi(\lambda)$ is changed at the same time. Therefore, the relative error of recovery remains basically the same. Clearly, the larger n_j, $j = 1, 2, \ldots, m$, the easier the recovery, provided that the level of noise is the same. In fact, the range of n_j for which the recovery is feasible depends highly on the difference between z_j and z_{j+1}. We suggest that n_j be larger than 1, $j = 1, 2, \ldots, m$, if the level of noise is not greater than 0.001. Some restriction one should impose on the difference of the parameters of the adjacent layers: $|n_{j+1} - n_j|$ should be of order 100ϵ where ϵ is the absolute error of the data. No restrictions are needed on $|n_i - n_j|$, $i \neq j-1$, j, $j+1$, i, $j = 1, 2, \ldots, m$.

Thirdly, the sample rate $\Delta\lambda_i$, $i = 1, 2, \ldots, N$ for $\phi(\lambda)$ can be chosen between 0.01 and 0.1. The number of samples can be around 100, and this number should be increased as the number of layers increases. One should not choose large λ as sample points except one point to get n_m and does not have to choose equidistant samples. The rate of convergence of $x^{(k)}$ depends on the choice of samples.

Fourthly, since there are many local minimizers, we may not be able to get a global one if we start with an arbitrary initial point. We recommend to use the criterion

$$\{F(x^*) \leq \epsilon_1 \quad \text{and} \quad \| (J(x^*))^T (f(x^*)) \| < \epsilon\} \tag{10}$$

to decide which minimizer is the global one. In order to get better initial points one can use an auxiliary optimization method, such as pattern search method or simplex method [7], and then apply the above algorithm to the prospective points found by the auxiliary method.

Fifthly, since data are always noisy in practice, it is important to show that the method can handle the cases with noisy data when the noise level is within a certain level. Note that $\phi(\lambda)$ decreases very fast as λ grows. In our examples noise/signal ratio for 88% of the data with noise level 0.001 is ≤ 0.01. The remaining data, though very noisy, do not play a significant role in the recovery. Our results show that the algorithm works well in the presense of noise. In tables 2, 4 and 6, one can see that if the level of noise is increased a little, then the accuracy of the recovered parameters does not decrease much. At the recovered

values of $\{n_j, z_j\}$ every f_i in the target function F is not greater than the noise. Therefore the accuracy of the recovery can not be improved.

We conclude that the algorithm can be used for practical recovery of the layered medium from the seismic data.

IV. Additional remarks..

The algorithm can be generalized for the case when n_j and z_j are complex numbers. The number of the layers can be increased to 10 by the given method according to the results of numerical experiments.

Let us give a method for estimating the number of layers from the data.

Let $T(\lambda) := \sum_{j=1}^{m} n_j e^{-\lambda z_j}$. Apparently, formula (5) and $T(\lambda)$ have the same form. Set $T_k = T(k \cdot \Delta\lambda)$, $\xi_j = e^{-\Delta\lambda \cdot z_j}$, $\Delta\lambda > 0$, $j = 1, 2, \ldots, m$, thus $T_k = \sum_{j=1}^{m} n_j \xi_j^k$, $k = 1, 2, \ldots$. Assume $\xi_i \neq \xi_j$ for $i \neq j$, $i, j = 1, 2, \ldots, m$, then we have

$$\text{rank } A_p = \min(m, p) \qquad \text{for } n_j > 0, j = 1, 2, \ldots, m, \tag{11}$$

and

$$\text{rank } A_p = m \qquad \text{for } p \geq m, \text{and } n_j \neq 0, j = 1, 2, \ldots, m. \tag{12}$$

where rank A is the rank of the matrix A, and

$$A_p = \begin{pmatrix} T_0 & T_1 \ldots & T_{p-1} \\ T_1 & T_2 \ldots & T_p \\ \vdots & \ldots\ldots & \vdots \\ T_{p-1} & T_p \ldots & T_{2p-2} \end{pmatrix}.$$

Hence the number m is the smallest p starting from which rank A_p does not change as p grows. Since each ξ_j, $j = 1, 2, \ldots, m$, exponentially decreases when k grows, A_p becomes ill-conditioned as p increases. Therefore it is difficult to compute its rank. Prony's method can also be used for determining n_j and z_j, $j = 1, 2, \ldots, m$. Prony's method is complicated and very sensitive to the noise in the data. An extensive bibliography on Prony's method can be found in [8] and [3, Appendix 7].

One can use (11) or (12) for estimating m, and then use our algorithm with the criterion (10) for recovery of n_j and z_j.

REFERENCES

1. A. G. Ramm, An inverse problem for the Helmholtz equation in a semi infinite medium. Inverse problems, 3, L19-L22, 1987.
2. A. G. Ramm, Inverse scattering for geophysical problems. Inverse problems, 1, 133-172, 1985.
3. A. G. Ramm, Scattering by obstacles. D. Reidel, Dordrecht, 1986.
4. A. G. Ramm, Signal estimation from incomplete data. Journal of mathematical analysis and applications, 123, 1987.
5. A. G. Ramm, Inversion of the Laplace transform from real axis. Inverse problem, 2, L55-59, 1986.
6. Y.G. Evtushenko, Numerical optimization techniques. Optimization software Inc. Publication Division, New York, 1985.
7. J. Ortega, W. Rheinboldt, Iterative solution of nonlinear equations in several variables. Acad. Press, New York, 1970.
8. J.R. Auton and M.L. Van Blaricum, Investigation of resonance extraction procedures, Vol. I. Mathematics notes, note 79, 1981.

APPENDIX 1

THEOREM 1.. *If* $\lambda^2 > q_0 := \omega^2 \max_x |n(x)|$, *then the integral equation (2) is uniquely solvable by iterations, and* $\tilde{u}$ *is analytic in* λ *in the region* $Re\lambda > q_0^{1/2}$.

PROOF: Let $V\tilde{u} := \int_{-\infty}^{+\infty} g(\lambda, z-z')q(z')\tilde{u}(\lambda, z')dz'$. One has

$$\sup_x \int_{-\infty}^{+\infty} |g(\lambda, z-z')q(z')|dz' \leq \sup_x \frac{q_0}{2\lambda} \int_{-\infty}^{+\infty} e^{-\lambda|x-z'|}dz'$$

$$= \sup_x \frac{q_0}{2\lambda} \frac{2}{\lambda} = \frac{q_0}{\lambda^2}.$$

By Young's inequality, V maps $L^2(-\infty, +\infty)$ into itself, and $||V|| \leq \frac{q_0}{\lambda^2}$. If $q_0 < \lambda^2$, then $||V|| < 1$. Therefore the series

$$\tilde{u}(\lambda, z) = g(\lambda, z) + Vg(\lambda, z) + \ldots + V^k g(\lambda, z) + \ldots \tag{13}$$

gives the unique solution to equation (2) and converges uniformly in z and λ for $\lambda > q_0^{1/2}$. Since for all positive integers k, the function $V^k g(\lambda, z) = \int_{-\infty}^{+\infty} \frac{e^{-\lambda|z-z'|}}{2\lambda} q(z')\, V^{k-1} g(\lambda, z')dz'$ is analytic in λ and the series (13) converges uniformly in the region $\lambda > q_0^{1/2}$, it follows that $\tilde{u}$ is analytic in λ if $\lambda > Re\lambda > q_0^{1/2}$.

In the following theorem, we use the same notations as in the section IV.

THEOREM 2.. *Assume that* $\xi_i \neq \xi_j$ *for* $i \neq j$, $i, j = 1, 2, \ldots, m$, *then (11) and (12) hold.*

We consider (11) first. It is sufficient to prove that $det\ A_p = 0$ *if* $p > m$ *and that* $det\ A_p \neq 0$ *if* $p \leq m$.

Suppose $p > m$, *we want to show that* $det\ A_p = 0$. *Note that* $det\ A_p$ *can be represented as a sum of terms*

$$n_{i_1} n_{i_2} \ldots n_{i_p}\ det \begin{pmatrix} 1 & \xi_{i_2} \ldots & \xi_{i_p}^{p-1} \\ \xi i_1 & \xi_{i_2}^2 \ldots & \xi_{i_p}^p \\ \vdots & \ldots\ldots & \vdots \\ \xi_{i_1}^{p-1} & \xi_{i_2}^p \ldots & \xi_{i_p}^{2p-2} \end{pmatrix}. \tag{14}$$

where $i_k = 1,2,\ldots,m, k = 1,2,\ldots,p$. Since $p > m$, at least two columns of the above matrix are limearly dependent. So the determinant of the matrix is zero. This implies that det $A_p = 0$.

Now we prove that det $A_p \neq 0$ if $p \leq m$. Consider all the terms of the form (14). We divide them into several groups. In each group the p - tuple $(i_1, i_2, \ldots, i_p)$ runs through all the permutations of the integers $i_1, i_2, \ldots, i_p$. We pick up one of these groups and sum all the terms in it. Without loss of generality, we assume that for this group $(i_1, i_2, \ldots, i_p)$ is a permutation of (1, 2, ..., p). Therefore the sum is

$$det \begin{pmatrix} \sum_{j=1}^{p} n_j & \sum_{j=1}^{p} n_j \xi_j \ldots & \sum_{j=1}^{p} n_j \xi^{p-1} \\ \sum_{j=1}^{p} n_j \xi_j & \sum_{j=1}^{p} n_j \xi_j^2 \ldots & \sum_{j=1}^{p} n_j \xi_j^p \\ \vdots & \ldots\ldots & \vdots \\ \sum_{j=1}^{p} n_j \xi_j^{p-1} & \sum_{j=1}^{p} n_j \xi_j^p \ldots & \sum_{j-1}^{p} n_j \xi_j^{2p-2} \end{pmatrix}$$

$$= det \begin{pmatrix} 1 & 1 \ldots & 1 \\ \xi_1 & \xi_2 \ldots & \xi_p \\ \vdots & \ldots\ldots & \vdots \\ \xi_1^{p-1} & \xi_2^p \ldots & \xi_p^{2p-2} \end{pmatrix} det \begin{pmatrix} n_1 & n_1 \xi_1 \ldots & n_1 \xi_1^{p-1} \\ n_2 & n_2 \xi_2 \ldots & n_2 \xi_2^{p-1} \\ \vdots & \ldots\ldots & \vdots \\ n_p & n_p \xi_p \ldots & n_p \xi_p^{p-12} \end{pmatrix}$$

$$= n_1 n_2 \ldots n_p \ det \begin{pmatrix} 1 & 1 \ldots & 1 \\ \xi_1 & \xi_2 \ldots & \xi_p \\ \vdots & \ldots\ldots & \vdots \\ \xi_1^{p-1} & \xi_2^p \ldots & \xi_p^{2p-2} \end{pmatrix}^2 \geq 0.$$

Since $\xi_i \neq \xi_j$ for $i \neq j, i, j = 1,2,\ldots,m$, and $p \leq m$, we infer that det $A_p > 0$ if $p \leq m$. Therefore we conclude that rank A_p = min(m,p) under the assumption (11).

From the above argument, it is clear that det $A_p = 0$ for $p > m$, regardless of the values of n_j. If p = m, then the above argument yields

$$det\ A_m = n_1 n_2 \dots n_m\ det \begin{pmatrix} 1 & 1 \dots & 1 \\ \xi_1 & \xi_2 \dots & \xi_p \\ \vdots & \dots\dots & \vdots \\ \xi_1^{p-1} & \xi_2^p \dots & \xi_p^{2p-2} \end{pmatrix}^2 \geq 0.$$

provided that $n_j \neq 0$, *and* $\xi_i \neq \xi_j$ *for* $i \neq j$, $i, j = 1, 2, \dots, m$. *Therefore (12) follows. This completes the proof of theorem 2.*

APPENDIX 2

In the section III, we point out that the asymptotic approach practically is hard to carry out. Here we give an explanation. Consider two layer case, then (5) becomes

$$n_2 + (n_1 - n_2)e^{-2\lambda z_2} - n_1 e^{-2\lambda d} = \psi(\lambda) \tag{15}$$

Let us first describe the asymptotic approach to get n_2, z_2, n_1 . Below $*$ denotes the recovered parameters. As mentioned before, we use a datum $\psi(\lambda_{i_0})$ to get n_2^* , where λ_{i_0} is a large number. Suppose $\psi(\lambda_{i_1})$ is a datum such that $n_1 e^{-2\lambda_{i_1} d} << (n_1 - n_2)e^{-2\lambda_{i_1} z_2}$ and $\frac{ln|n_1 - n_2|}{\lambda_{i_1}}$ is small enough. Since $d > z_2$, we can have such a datum. Then we approximately have

$$ln|n_1 - n_2| - 2\lambda_{i_1} z_2 = ln|\psi(\lambda_{i_1}) - n_2^*|$$

In order to get z_2 , we eliminate the small quantity $\frac{ln|n_1 - n_2|}{\lambda_{i_1}}$, hence

$$z_2^* = -\frac{ln|\psi(\lambda_{i_1}) - n_2^*|}{2\lambda_{i_1}} \tag{16}$$

From (14), using another datum $\psi(\lambda_{i_2})$, we get

$$n_1^* = \frac{\psi(\lambda_{i_2}) - n_2^* + n_2^* e^{-2\lambda_{i_2} z_2}}{e^{-2\lambda_{i_2} z_2} - e^{-2\lambda_{i_2} d}}$$

Assume now that $n_1 = 4, n_2 = 7, d = 5, z_2 = 2$. Let us analyse more carefully the way we get z_2^* . (16) is more precise when λ becomes larger. For example,

$$\lambda = 10, \quad z_2^* = -\frac{ln(1.2745063 * 10^{-17})}{20} = 1.9450$$

$$\lambda = 20, \quad z_2^* = -\frac{ln(5.4145542 * 10^{-35})}{40} = 1.9725$$

One sees that if λ increases, the numbers under logarithm decrease extremely fast, while the accuracy of z_2^* does not change much. Practically, the round-off error of computers does not allow us to increases λ very much. Moreover we always have noise in $\psi(\lambda_{i_1}) - n_2^*$ which makes the estimation of z_2^* difficult. Therefore we are not able to recover z_2^* by this approach even in this simplest two layer case.

APPENDIX 3

Here is the list of arguments of our program which is appended.

AA - Real work matrix of order $(2k-2)\times(2k-2)$.

AJ - Real work matrix of order $(M0+3k-3)\times(2k-2)$.

AJF - Real work vector of length $(2k-2)$.

AJTJ - Real work matrix of order $(2k-2)\times(2k-2)$.

AJTJJ1 - Real work matrix of order $(2k-2)\times(2k-2)$.

ALAMDA - Input real vector of length $M0$, which contains the points where the data are picked up.

AN - Real work vector of length k. In the end, it contains the estimation of numbers in ANE.

ANE - Input real vector of length k, in which the exact parameters n_j, $j=1,2,\ldots,k$, are stored.

AZ - Real work vector of length k. In the end, it contains the estimation of numbers in AZE.

AZE - Input real vector of length k, in which the exact parameters z_j, $j=1,2,\ldots,k$, are stored.

EPSN - Input real number which is the error of the data.

F - Real work vector of length $M0+3k-3$.

FE - Real work vector of length $M0$, which contains the exact data.

FI1 - Real number which is the value of the target function.

K - Input integer which is the number of the layers.

M0 - Input integer which is the number of the data.

T - Real work number which is the maximum value among all the components of the vector $(J(x^{(k)}))^T f(x^{(k)})$ in formula (7) in each step of iteration.

P - Real work vector of length $(2k-2)$.

X - Real work vector of length $(2k-2)$.

X1 - Real work vector of length $(2k-2)$.

Appendix 4 : The program of the algorithm

```
FILE: T        FORTRAN  A   KANSAS STATE UNIVERSITY VM/SP CMS

C*********************************************************************
C
C   THE PROGRAM FOR SOLVING THE INVERSE LAYER PROBLEM
C
C*********************************************************************
C       THE MAIN PROGRAM
        REAL*8 AN(4),AZ(4),ANE(4),AZE(4),AJ(89,6),AJTJ(6,6),ALAMDA(80),
     *         X(6),X1(6),F(89),FE(80),AJF(6),P(6),AJTJ1(6,6)
        REAL*8 ALPHA,GAMMA,R,DE,EP,EPS,EPSN,FI1,FI,T,AA(5,12),PHI
        DATA ALAMDA/0.020+00,0.04D+00,0.06D+00,0.08D+00,0.10D+00,
     *              0.12D+00,0.14D+00,0.16D+00,0.18D+00,0.20D+00,
     *              0.21D+00,0.22D+00,0.23D+00,0.24D+00,0.25D+00,
     *              0.26D+00,0.27D+00,0.28D+00,0.29D+00,0.30D+00,
     *              0.31D+00,0.32D+00,0.33D+00,0.34D+00,0.35D+00,
     *              0.36D+00,0.37D+00,0.38D+00,0.39D+00,0.40D+00,
     *              0.42D+00,0.44D+00,0.46D+00,0.48D+00,0.40D+00,
     *              0.52D+00,0.54D+00,0.56D+00,0.58D+00,0.50D+00,
     *              0.62D+00,0.64D+00,0.66D+00,0.68D+00,0.70D+00,
     *              0.72D+00,0.74D+00,0.76D+00,0.78D+00,0.80D+00,
     *              0.82D+00,0.84D+00,0.86D+00,0.88D+00,0.90D+00,
     *              0.92D+00,0.94D+00,0.96D+00,0.98D+00,1.00D+00,
     *              1.02D+00,1.04D+00,1.06D+00,1.08D+00,1.00D+00,
     *              1.12D+00,1.14D+00,1.16D+00,1.18D+00,1.20D+00,
     *              1.22D+00,1.24D+00,1.26D+00,1.28D+00,1.30D+00,
     *              1.32D+00,1.34D+00,1.36D+00,1.38D+00,1.40D+00/
        DATA ANE/5.00+00,8.0D+00,3.0D+00,7.0D+00/
        DATA AZE/6.00+00,4.5D+00,3.10+00,1.50+00/
        DATA X/4.9810D+00,7.82D+00,2.980+00,4.540+00,3.0TD+00,1.501D+00/
        K=4
        N=2*K-2
        NN=2*N
        M0=80
        M=M0+3*K-3
        ALPHA=1.0D+00
        GAMMA=5.0D+00
        EP=1.00-09
        EPS=1.00-09
        EPSN=0.0001D+00
        R=500.0D+00
        DO 2 I=1,K-1
        AN(I)=X(I)
2       AZ(I+1)=X(I+K-1)
        AN(K)=ANE(K)+0.10-09
        AZ(1)=AZE(1)
        DO 10 I=1,M0
        CALL FF(ANE,AZE,K,ALAMDA(I),PHI)
        WRITE(6,9) I,PHI
8       FORMAT(2X,2HI=,I6,4X,3HFE=,D20.10)
10      FE(I)=PHI+EPSN
        WRITE(6,20) ANE,AZE
20      FORMAT(1X,3HN1=,D14.6,2X,3HN2=,D14.6,
     *         2X,3HN3=,D14.6,2X,3HN4=,D14.6/
     *         1X,3HZ1=,D14.6,2X,3HZ2=,D14.6,
     *         2X,3HZ3=,D14.6,2X,3HZ4=,D14.6)
        WRITE(6,24) AN,AZ
```

```
FILE: T        FORTRAN  A   KANSAS STATE UNIVERSITY VM/SP CMS

 24    FORMAT(1X,13HINITIAL VALUE/1X,3HN1=,D14.6,3X,3HN2=,D14.6,3X,
     *         3HN3=,D14.6,3X,3HN4=,D14.6/
     * 1X,3HZ1=,D14.6,3X,3HZ2=,D14.6,3X,3HZ3=,D14.6,3X,3HZ4=,D14.6)
       IT=1
       DO 40 I=1,M0
       CALL FF(AN,AZ,K,ALAMDA(I),PHI)
 40    F(I)=PHI-FE(I)
       DO 42 I=M0+1,M
 42    F(I)=0.0D+00
       DO 50 I=1,N
       IF(X(I)) 44,50,50
 44    F(M0+I)=R0X(I)=X(I)
 50    CONTINUE
       DO 56 I=1,K-1
       IF(AZ(I)-AZ(I+1)) 54,56,56
 54    F(M0+N+I)=(AZ(I)-AZ(I+1))*(AZ(I)-AZ(I+1))*R
 56    CONTINUE
 60    WRITE(6,20) AN,AZ
       FI=0.00+00
       DO 70 I=1,M
 70    FI=FI+F(I)**F(I)
       FI=DSQRT(FI/DFLOAT(M))
       WRITE(6,72) FI
 72    FORMAT(1X,5HFSUM=,D16.8)
 30    IW=-1
       IF(ALPHA.LT.0.0000050+00) GOTO 79
       ALPHA=ALPHA/GAMMA
 79    CALL RJ(AN,AZ,K,AJ,N,M,M0,ALAMDA,R)
       DO 100 I=1,N
       DO 80 I1=1,N
 80    AJTJ(I,I1)=0.0D+00
       AJF(I)=0.00+00
100    P(I)=0.00+00
       DO 140 I=1,N
       DO 110 I1=1,N
       DO 110 I2=1,M
110    AJTJ(I,I1)=AJTJ(I,I1)+AJ(I2,I)*AJ(I2,I1)
       DO 120 I3=1,M
120    AJF(I)=AJF(I)+AJ(13,1)**F(I3)
140    CONTINUE
       DO 160 I=1,N
       DO 160 I1=1,N
160    AJTJ1(I,I1)=AJTJ(I,I1)
170    WRITE(6,142) AJF
142    FORMAT(1X,'AJF=',3D20.10)
       DO 180 I=1,N
       DO 180 J=1,N
180    AJTJC(I,J)=AJTJ1(I,J)
       DO 182 I=1,N
182    AJTJC(I,I)=AJTJC(I,I)+ALPHA
       IF(ALPHA.LT.0.01D+00) GOTO 185
       DO 184 I=1,N
       DO 184 J=1,N
184    AJTJC(I,J)=AJTJC(I,J)/ALPHA
185    CALL IVSN(AJTJ,AA,N,NN,DE,EP,IS)
```

```
FILE: T     FORTRAN  A   KANSAS STATE UNIVERSITY VM/SP CMS

      IT=IT+1
      IF(IS.GT.0) GOTO 200
      WRITE(6,190) IS
190   FORMAT(1X,25HTHE MATRIX IS DEGENERATED,I10)
      GOTO 250
200   DO 202 I=1,N
202   P(I)=0.00+00
      DO 210 I=1,N
      DO 210 II=1,N
210   P(I)=P(I)+AJTJ(I,II)*AJF(II)
      WRITE(6,105) P
105   FORMAT(1X,2HPP,3D20.10)
      WRITE(6,106) ALPHA
106   FORMAT(1X,6HALPHA=,D15.8)
      WRITE(6,84) IT
84    FORMAT(5X,3HIT=,I6)
211   I=1
212   IF(P(I)-1.00+00) 213,214,214
213   I=I+1
      IF(I.GT.N) GOTO 219
      GOTO 212
214   DO 215 I=1,N
215   P(I)=P(1)/2.00+00
      GOTO 211
219   DO 220 I=1,N
220   X1(I)=X(I)-P(I)
      DO 222 I=1,K-1
      AN(1)=X1(I)
222   AZ(I+1)=X1(I+K-1)
      DO 230 I=1,M0
      CALL FF(AN,AZ,K,ALAMDA(1),PHI)
230   F(I)=PHI-FE(1)
      DO 231 I=M0+1,M
231   F(I)=0.00+00
      DO 240 I=1,N
      IF(X1(I)) 238,240,240
238   F(M0+I)=X1(I)*X1(1)*R
240   CONTINUE
      DO 246 I=1,K-1
      IF(AZ(1)-AZ(I+1)) 244,246,246
244   F(M0+N+I)=(AZ(1)-AZ(1+1))*(AZ(1)-AZ(1+1))*R
246   CONTINUE
249   WRITE(6,20) AN,AZ
      F11=0.0
      DO 252 I=1,M
252   F11=F11+F(I)*F(I)
      F11=DSQRT(F11/DFLOAT(M))
      WRITE(6,72) F11
      IF(F11.LT.F1) GOTO 270
250   IW=IW+1
      ALPHA=ALPHA*GAMMA
      WRITE(6,260) IW
260   FORMAT(1X,3HIW=,I5)
      T=0
      DO 264 I=1,N
```

```
FILE: T        FORTRAN  A   KANSAS STATE UNIVERSITY VM/SP CMS

      IF(DABS(AJF(I)).LE.T) GOTO 264
      T=DABS(AJF(I))
 264  CONTINUE
      WRITE(6,266) T
 266  FORMAT(1X,5HMAJF=,D20.10)
      IF(T.LT.EPS) GOTO 999
      IF(IW.GT.7) GOTO 999
      GOTO 170
 270  WRITE(6,280) ALPHA
 280  FORMAT(1X,6HALPHA=,DI4.8)
      IF(IT.GT.100) GOTO 999
      T=0
      DO 284 I=I,N
      IF(DABS(AJF(I)).LE.T) GOTO 284
      T=DABS(AJF(I))
 284  CONTINUE
      WRITE(6,266) T
      IF(T.LT.EPS) GOTO 999
      DO 290 I=1,N
 290  X(I)=X1(I)
      FI=FII
      GOTO 30
 999  STOP
      END
C*********************************************************************
      SUBROUTINE RJ(AN,AZ,K,AJ,N,M,MO,ALAMDA,R)
      REAL*8 AN(K),AZ(K),AJ(M,N),ALAMDA(MO)
      REAL*8 R
      DO 10 J=1,K-1
      DO 10 I=I,MO
      AJ(I,JJ)=DEXP(-2.00+00*ALAMDA(I)*AI(J+1))
     *          -DEXP(-2.00+00*ALAMDA(I)*AZ(J))
 10   AJ(I,J+K-I)=2.00+00*(AN(J+I)-AN(J))*ALAMDA(I)
     *          *DEXP(-2.00+00*ALAMDA(I)*AZ(J+I))
      DO 12 I=MO+I,M
      DO 12 J=1,N
 12   AJ(I,J)=0.00+00
      DO 30 I=1,K-1
      IF(AN(I)) 20,26,25
 20   AJ(MO+I,I)=2.00+00*AN(I)*R
 26   IF(AZ(I+I)) 28,30,30
 28   AJ(MO+K-1+I,K-I+I)=2.00+00*AZ(I+1)*R
 30   CONTINUE
      DO 50 I=1,K-1
      IF(AZ(I)-AZ(I+I)) 40,50,50
 40   AJ(MO+N+I,K-2+I)=-2.00+00*(AZ(I+1)-AZ(I))*R
      AJ(MO+N+I,K-1+I)=-AJ(M+N+I,K-2+I)
 50   CONTINUE
      AJ(MO+N+1,K-1)=0.00+00
 130  RETURN
      END
C*********************************************************************
      SUBROUTINE FF(ANS,AZS,K,C,PHI)
      REAL*8 ANS(K),AZS(K)
      REAL*8 C,PHI
```

```
FILE: T          FORTRAN  A   KANSAS STATE UNIVERSITY VM/SP CMS

      PHI=0.0D+00
      DO 10 I=1,K-1
 10   PHI=PHI+ANS(I)*(DEXP(-2.0D+00*C*AZS(I+1))
     *                -DEXP(-2.0D+00*C*AZS(I)))
      PHI=PHI+ANS(K)*(1.0D+00-DEXP(-2.0D+00*C*AZS(K)))
      RETURN
      END
C*********************************************************************
      SUBROUTINE IVSN(A,AA,N,NN,DE,EP,IS)
      REAL*8 A(N,N),AA(N,NN)
      REAL*8 Y,DE,EP,C
      IS=1
      DE=1.0D+00
      DO 6 I=1,N
      DO 7 J=1,N
 7    AA(I,J)=A(I,J)
      DO 8 J=1,N
 8    AA(I,J+N)=0.0
 6    AA(I,I+N)=1.0
      DO 100 I=1,N
      Y=0.0D+00
      K=I
      DO 20 J=I,N
      IF(DABS(AA(J,I)).LE.DABS(Y)*1.0D-10) GOTO 20
      K=J
      Y=AA(J,I)
 20   CONTINUE
      DE=DE*Y
      IF(DABS(Y).GT.EP) GOTO 30
      IS=-1
      RETURN
 30   IF(K-I) 40,50,40
 40   DO 44 J=I,NN
      C=AA(I,J)
      AA(I,J)=AA(K,J)/Y
 44   AA(K,J)=C
      DE=-DE
      GOTO 62
 50   DO 60 J=I,NN
 60   AA(I,J)=AA(I,J)/Y
 62   DO 80 J2=I+1,2*N
      DO 80 J1=I+1,N
 80   AA(J1,J2)=AA(J1,J2)-AA(J1,I)*AA(I,J2)
100   CONTINUE
      DO 120 I=1,N-1
      DO 120 J2=N+1,NN
      DO 120 J1=1,N-I
120   AA(J1,J2)=AA(J1,J2)-AA(J1,N+1-I)*AA(N+1-I,J2)
      DO 140 I=1,N
      DO 140 J=1,N
140   A(I,J)=AA(I,J+N)
      RETURN
      END
C*********************************************************************
```

TABLE 1: TWO LAYERS, NOISELESS DATA †

n_1		4.0000	12.0000	7.2000	12.0000	4.0000	15.0000	15.0000
n_2		2.0000	7.0000	7.0000	7.0000	7.0000	7.0000	7.0000
z_1	(given)	5.0000	30.0000	14.0000	14.0000	19.0000	5.0000	5.0000
z_2		3.0000	24.0000	8.0000	13.0000	18.0000	4.2000	4.8000
n_1^*		4.0001	11.9998	7.2000	11.9800	4.1029	15.0000	15.0742
n_2^*		2.0000	7.0000	7.0000	7.0000	7.0000	7.0000	7.0000
z_2^*		3.0000	23.9993	8.0000	12.9962	17.9654	4.2000	4.8018

† all the results are rounded to four decimals.
* denotes the recovered parameters.

TABLE 2: TWO LAYERS, NOISY DATA

n_1	4.0000	12.0000	7.2000	15.0000	
n_2	2.0000	7.0000	7.0000	7.0000	
z_1 (given)	5.0000	30.0000	14.0000	5.0000	
z_2	3.0000	24.0000	8.0000	4.6000	
n_1^*	3.9990 3.9896	11.9953 11.9543	7.2003	15.0351 15.3714	noise level 0.0001 † noise level 0.001
n_2^*	2.0001 2.0010	7.0001 7.0010	7.0001	7.0001 7.0010	noise level 0.0001 noise level 0.001
z_2^*	2.9987 2.9871	23.9934 23.9354	7.9929	4.6016 4.6146	noise level 0.0001 noise level 0.001

† by noise level we mean the absolute error of the data in tables 2, 4 and 6.

TABLE 3: THREE LAYERS, NOISELESS DATA

n_1	10.0000	10.0000	4.0000	4.0000	8.0000	8.0000
n_2	7.0000	7.0000	5.0000	5.0000	2.0000	2.0000
n_3	4.0000	4.0000	1.0000	1.0000	5.0000	5.0000
z_1 (given)	3.0000	20.0000	4.0000	4.0000	0.9000	4.0000
z_2	2.0000	15.0000	3.8000	1.2000	0.6000	3.7000
z_3	1.0000	13.0000	1.0000	1.0000	0.3000	3.4000
n_1^*	10.0000	9.9978	4.0791	4.0000	7.9998	8.0505
n_2^*	7.0000	6.9404	5.0003	4.9649	1.9997	3.0246
n_3^*	4.0000	4.0000	1.0000	1.0000	5.0000	5.0000
z_2^*	2.0000	14.9726	3.7823	1.2043	0.5998	3.7375
z_3^*	1.0000	12.9841	1.0000	0.9992	0.3001	3.3320

TABLE 4: THREE LAYERS, NOISY DATA

n_1	10.0000	4.0000	4.0000	8.0000	
n_2	7.0000	5.0000	5.0000	2.0000	
n_3	4.0000	1.0000	1.0000	5.0000	
z_1 (given)	20.0000	4.0000	4.0000	0.9000	
z_2	15.0000	3.8000	1.2000	0.6000	
z_3	13.0000	1.0000	1.0000	0.3000	
n_1^*	10.0307 10.4230	4.2136 4.5122	4.0001 4.0053	8.0263	noise level 0.0001 noise level 0.001
n_2^*	7.2731 7.6933	5.0002 4.9955	4.7976 4.7888	2.0486	noise level 0.0001 noise level 0.001
n_3^*	4.0001 4.0010	1.0001 1.0010	1.0001 1.0010	5.0001	noise level 0.0001 noise level 0.001
z_2^*	15.1933 16.2958	3.7468 3.6303	1.2261 1.1991	0.6029	noise level 0.0001 noise level 0.001
z_3^*	13.0488 12.9653	1.0000 0.9984	0.9947 0.9912	0.2985	noise level 0.0001

TABLE 5: FOUR LAYERS, NOISELESS DATA

n_1	5.0000	5.0000	5.0000	5.0000	5.0000	5.0000
n_2	8.0000	8.0000	8.0000	8.0000	8.0000	8.0000
n_3	6.0000	6.0000	6.0000	6.0000	6.0000	6.0000
n_4	10.0000	10.0000	10.0000	10.0000	10.0000	10.0000
z_1 (given)	5.2000	7.0000	7.5000	7.5000	8.0000	4.5000
z_2	3.8000	6.2000	2.4000	6.4000	7.1000	3.3000
z_3	2.5000	1.8000	1.7000	5.3000	6.2000	2.3000
z_4	1.2000	1.0000	1.0000	1.0000	5.2000	1.1000
n_1^*	5.0293	5.0018	5.0005	5.0062	5.0106	4.9584
n_2^*	8.1760	8.0001	8.3189	8.0256	7.4764	7.7602
n_3^*	6.0146	6.0004	6.0622	6.0000	5.6168	5.9792
n_4^*	10.0000	10.0000	10.0000	10.0000	10.0000	10.0000
z_2^*	3.7460	6.1994	2.3692	6.3920	7.2012	3.3662
z_3^*	2.5531	1.8002	1.7692	5.3054	5.6579	2.2304
z_4^*	1.1991	1.0000	0.9973	1.0000	5.1973	1.1012

TABLE 6: FOUR LAYERS, NOISY DATA

n_1	5.0000	5.0000	5.0000	5.0000	5.0000	5.0000	
n_2	8.0000	8.0000	8.0000	8.0000	8.0000	8.0000	
n_3	6.0000	6.0000	6.0000	6.0000	6.0000	6.0000	
n_4	10.0000	10.0000	10.0000	10.0000	10.0000	10.0000	
z_1 (given)	5.2000	7.0000	7.5000	7.5000	8.0000	4.5000	
z_2	3.8000	6.2000	2.4000	6.4000	7.1000	3.3000	
z_3	2.5000	1.8000	1.7000	5.3000	6.0000	2.3000	
z_4	1.2000	1.0000	1.0000	1.0000	5.2000	1.1000	
n_1^*	5.0369	5.0430	5.0011	5.0021	5.2246	5.0721	noise level 0.0001
	5.1971			6.1799			noise level 0.001, 0.005+
n_2^*	8.1152	8.0007	8.3245	7.7782	7.3716	7.8455	noise level 0.0001
	8.3570			7.4567			noise level 0.001, 0.005
n_3^*	5.9956	5.9911	6.0363	5.9997	5.6422	5.9527	noise level 0.0001
	5.8726			5.9922			noise level 0.001, 0.005
n_4^*	10.0001	10.0001	10.0001	10.0001	10.0001	10.0001	noise level 0.0001
	0.0010			10.0050			noise level 0.001, 0.005
z_2^*	3.7520	6.1881	2.3657	6.4472	7.1332	3.2890	noise level 0.0001
	3.5874			5.0568			noise level 0.001, 0.005
z_3^*	2.5229	1.7979	1.7603	5.2356	5.5113	2.2177	noise level 0.0001
	2.4738			4.4969			noise level 0.001, 0.005
z_4^*	1.2007	1.0007	0.9989	1.0001	5.1684	1.1031	noise level 0.0001
	1.2127			1.0015			noise level 0.001, 0.005

+ The data in the second column have noise level 0.0001 and 0.001, and the data in the fifth column have noise level 0.0001 and 0.005.

NUMERICAL RECOVERY OF THE LAYERED MEDIUM FROM THE SURFACE DATA

by

PEIQING LI

B.S., Xi'an Jiaotong University, 1984

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Mathematics
KANSAS STATE UNIVERSITY
Manhattan, Kansas

1987

ABSTRACT.

In this paper an inverse problem is formulated and its numerical solution is given. The problem is of interest, for example, in geophysics. Numerical results show that the algorithm is efficient for noisy data. Practical recommendations are given. The computer code which solves the inverse problem is appended.