An Accurate and Efficient Algorithm for Parameter Estimation of 2D Acoustic Wave Equation

Wenyuan Liao

***Abstract*—An efficient and accurate numerical method has been proposed in this paper to estimate the acoustic coefficient in a 2D acoustic wave equation. The inverse problem is formulated as a PDE-constrained optimization problem, in which the forward problem is numerically solved by both efficient finite difference schemes. To generate the gradient of the quadratic misfit function with respect to the unknown coefficient, the widely used automatic differentiation tool TAMC is used. Finally a gradient-based optimization algorithm is applied to minimize the misfit function. Numerical results are presented to show that the proposed method is effective and robust in estimating the acoustic coefficient.**

***Index Terms*—Inverse problem; wave equation; finite difference method; adjoint method; optimization.**

I. Introduction

The identification of model coefficients in an acoustic wave equation through observational data plays a crucial role in applied mathematics, geo-science, physics and many other areas. This technique has been widely used to determine the unknown property of a medium in which the wave is propagated by measuring data on its boundary or a specified location in the domain. A great deal of work have been done and many direct and indirect methods have been reported in the past decades [1-4,10]. In the wave equation, the unknown coefficient which characterizes the property of the medium is important to the physical process but usually cannot be measured directly, or very expensive to be measured, thus some mathematical method is needed to estimate it.

In this paper, we extend the method developed in [5] to a 2D acoustic wave equation subject to extra boundary condition. The wave equation has been widely used in modeling wave propagation in mediums such as earth. It is quite often that geophysicists need to know the internal structure of a certain area of the earth. However, due to the high drilling cost, it is impossible to drill many wells and examine the rock samples. Fortunately, the underlying mathematical model that describes the wave propagation is well understood, and it is much easier and more cost-effective to record the wave information on the surface of the domain of interest. Such technique forms exactly an inverse problem of wave equation. The basic question is: is it possible to recover the coefficient from the measurements on the surface of the interested part of the earth? As one can expect, there is no simple answer for this question, as a matter of fact, the complete mathematical solution of this questions is quite complicated, and is still partially answered or even unanswered at all.

One difficulty in solving an inverse problem is the instability, which means that a set of almost perfect observational data could produce a totally wrong result. Although the problem is still not solved completely, some important progresses on the stability were made in the past. According to Isakov, the inverse problem is stable if some smooth conditions for the domain, initial and boundary conditions are satisfied. More details regarding the well-posedness of the inverse problem can be found in [8]. For homogeneous wave equation, the analytical solution can be obtained for certain initial and boundary conditions. However, for the general case, the analytical solution is difficult to get, thus we should use numerical methods to solve the forward problem when an inverse problem is solved. In this paper, we applied a second-order accurate central finite difference approximation for time derivative and a fourth-order finite difference schemes to approximate u_{xx} and u_{yy} .

An accurate gradient is critical in minimizing the cost functional. Herein we apply automatic differentiation tool **TAMC** to generate discrete adjoint, then use it to calculate the gradient. The rest of this paper is organized as the following. In section 2, we formulate the inverse problem as a PDE-constrained optimization problem, followed by the description of an efficient finite difference scheme to solve the wave equation. In section 4 the automatic differentiation tool **TAMC** is briefly introduced, while the optimization package is introduced in section 5. Some numerical results are presented in section 6 and finally some conclusions are addressed.

II. Mathematical Formulation of the Inverse Problem

Let us consider the following 2D wave equation

$$u_{tt} = c(x,y)(u_{xx} + u_{yy}) + S(x,y,t) \quad (1)$$

satisfying the initial and boundary conditions:

$$u(x,y,0) = u_0(x,y), (x,y) \in \Omega \quad (2)$$

$$u_t(x,y,0) = u_1(x,y), (x,y) \in \Omega \quad (3)$$

$$u(x,y,t) = \phi(x,y,t), (x,y,t) \in \Omega \times (0,T) \quad (4)$$

and subject to the extra boundary condition:

$$\partial u / \partial n = \varphi(x,y,t), (x,y,t) \in \partial\Omega \times (0,t) \quad (5)$$

where $c(x,y)$ is the unknown acoustic coefficient, **n** is the outer normal orientation of $\partial\Omega$, $u_0(x,y)$, $u_1(x,y)$, $\phi(x,y,t)$ and $\varphi(x,y,t)$ are functions that satisfy some regularity conditions, $\partial\Omega$ is the boundary of the square domain Ω. For the sake of simplicity, throughout this paper, we assume that

Manuscript received July 12, 2011; revised August 10, 2011.

Wenyuan Liao,Department of Mathematics and Statistics, University of Calgary Calgary, Alberta, T2N 1N4, Canada (E-mail: wliao@ucalgary.ca)

$\Omega = [0, L] \times [0, L]$, with L be a positive constant.

If the coefficient $c(x, y)$ is known, the problem (1)-(5) is obviously over-specified because of the extra boundary condition in (5). However, since the coefficient $c(x, y)$ is unknown, a unique solution may exist and can be obtained through solving an inverse problem governed by the wave equation. Here we assume that the initial and boundary conditions are sufficiently smooth to guarantee a unique solution. More details regarding the existence and uniqueness can be found in [8,18].

The numerical procedure to estimate $c(x, y)$ and corresponding solution $u(x, y, t)$ is inherently an iterative procedure, which includes three main steps:

1. Starting from an initial guess $c^0(x, y)$, the forward problem defined in (1)-(4) is solved.

2. The numerical result from step 1 is compared with the condition in Eq. (5) to calculate the cost functional, then the gradient (derivative) of the cost functional with respect to $c^0(x, y)$ is calculated from the adjoint.

3. Optimization algorithm is applied to minimize the cost functional, and the three steps are repeated till convergence is obtained.

Here the cost functional $J(c(x, y))$ is defined as

$$J(c(x,y)) = \lambda \left\| c(x,y) \right\|_2 + \mu \left\| \nabla c(x,y) \right\|_2 + \int_0^T \int_{\partial\Omega} \left(\varphi(x,y,t) - \frac{\partial \tilde{u}(c(x,y); x, y, t)}{\partial n} \right)^2 ds dt \qquad (6)$$

where $\tilde{u}(c(x, y), x, y, t)$ is the solution of the initial-boundary value problem (1-4) based on $c(x, y)$, ds is the line integral along the boundary of Ω, λ and μ are non-negative regularity parameters. Note that in real computation, the integral in (6) is implemented by some high-order numerical methods such as Simpson's formula.

A detailed discussion on how to choose these parameters λ and μ to ensure the existence and uniqueness, and to improve the accuracy and efficiency of the solution is available in [5]. In general, if the user has no prior information regarding $c(x, y)$, they can simply set the parameters as zeros, although it may cause the inverse problem be ill-posed.

III. Finite Difference Method for the Wave Equation

For special initial and boundary conditions, the initial-boundary value problem defined in (1)-(4) can be solved analytically, but it becomes almost impossible when a general initial and boundary value problem is considered. Thus, accurate and efficient numerical schemes are needed to solve the wave equation.

To simplify the discussion, we assume that the domain Ω is divided into a uniform $N \times N$ grid with step sizes $h = h_x = h_y = L/(N-1)$. For convenience, we also assume that time step size $\Delta t = T/(M-1)$ is used for time integration. The numerical solution of $u(x, y, t)$ at grid point (x_i, y_j) and time level $t_k = k\Delta t$ is denoted as $u_{i,j}^k$. It is well-known that the central finite difference operators δ_x^2/h^2 and δ_y^2/h^2 approximate u_{xx} and u_{yy} with only second-order accuracy, where δ_x^2/h^2 and δ_y^2/h^2 are defined as:

$$\delta_x^2 u_{i,j} = u_{i+1,j} - 2u_{i,j} + u_{i-1,j}, \qquad (7)$$

$$\delta_y^2 u_{i,j} = u_{i,j+1} - 2u_{i,j} + u_{i,j-1}, \qquad (8)$$

In order to obtain fourth-order approximation in space, we replace δ_x^2 and δ_y^2 by $\delta_x^2(1 - \delta_x^2/12)$ and $\delta_y^2(1 - \delta_y^2/12)$ respectively.

Based on the approximation, the original wave equation in (1) is transformed to the following ODE system

$$(u_{i,j})_{tt} = \frac{c_{i,j}}{h^2}\left(\delta_x^2 + \delta_y^2 - \frac{\delta_x^4 + \delta_y^4}{12} \right) u_{i,j} + s_{i,j}(t) \qquad (9)$$

with $c_{i,j} = c(x_i, y_j), u_{i,j}(t) = u(x_i, y_j, t), s_{i,j}(t) = s(x_i, y_j, t)$.

Depending on the discretization of time derivative, the numerical methods for the ODE system (9) can be classified into two categories: explicit and implicit schemes. The explicit scheme is efficient but only conditionally stable, so in general small time step size is required. The implicit scheme is usually unconditionally stable but a linear or nonlinear algebraic system needs to be solved at each time step, which makes it less efficient. A detailed comparison can be found in [6,7]. In this paper, we apply the central finite difference scheme to the second-order time derivative, and obtain the following formula:

$$\frac{u_{i,j}^{k+1} - 2u_{i,j}^k + u_{i,j}^{k-1}}{\Delta t^2} = \frac{c_{i,j}}{h^2}\left(\delta_x^2 + \delta_y^2 - \frac{\delta_x^4 + \delta_y^4}{12} \right) u_{i,j} + s_{i,j}(t) \qquad (10)$$

which is second-order accurate in time and fourth-order accurate in space. Note that two initial conditions are needed in (10) but only $u_0(x, y)$ is given explicitly in (2), so we need to find a second-order accurate approximation of the second initial condition. Namely we have

$$\begin{aligned} u_{i,j}(\Delta t) &= u_{i,j}(0) + \Delta t \left. \frac{\partial u_{i,j}}{\partial t} \right|_{t=0} + \frac{\Delta t^2}{2} \left. \frac{\partial^2 u_{i,j}}{\partial t^2} \right|_{t=0} \\ &= u_0(x_i, y_j) + \Delta t u_1(x_i, y_j) + \frac{\Delta t^2}{2} u_2(x_i, y_j), \end{aligned} \qquad (11)$$

where $u_2(x_i, y_j) = c_{i,j}((u_0)_{xx} + (u_0)_{yy})_{i,j} + s(x_i, y_j, 0)$, and the truncation error of the approximation is Δt^3.

Once the numerical solution based on an initial guess $c^0(x, y)$ is available, we can use the observational data to calculate the cost functional. Since the extra boundary condition in Eq. (5) is Neumann boundary condition, the numerical solution cannot be used directly. We firstly apply Taylor expansion at the boundary points to approximate $\partial u / \partial n$. Since the numerical scheme for the forward problem is second-order in space, the interpolation scheme should be second-order accurate at least. For example, the following schemes can be used to approximate the boundary conditions:

$$\left. \frac{\partial u}{\partial x} \right|_{\substack{t=t^n \\ x=0}} = (-3u_{1,j}^n + 4u_{2,j}^n - u_{3,j}^n)/(2\Delta x), \; j = 1, 2, \cdots N$$

$$\left. \frac{\partial u}{\partial x} \right|_{\substack{t=t^n \\ x=L}} = (u_{N-2,j}^n - 4u_{N-1,j}^n + 3u_{N,j}^n)/(2\Delta x), \; j = 1, 2, \cdots N$$

$$\left. \frac{\partial u}{\partial y} \right|_{\substack{t=t^n \\ y=0}} = (-3u_{i1,1}^n + 4u_{i,2}^n - u_{i,3}^n)/(2\Delta y), i = 1, 2, \cdots N$$

$$\left. \frac{\partial u}{\partial y} \right|_{\substack{t=t^n \\ y=L}} = (u_{i1,N-2}^n - 4u_{i,N-2}^n + 3u_{i,N}^n)/(2\Delta y), i = 1, 2, \cdots N$$

where $u_{i,j}^n$ is the numerical solution at the grid point (x_i, y_j) at time level t^n, based on the initial guess $c^0(x,y)$. The time integral is implemented by Simpson's rule, which is fourth-order accurate. So eventually the cost functional is a function of $c(x,y)$.

IV. Discrete Adjoint Generated By Automatic Differentiation

The success of the gradient-based optimization relies on an accurate derivative of the cost functional with respect to the unknown coefficient. In this section we use **TAMC** to generate the discrete adjoint which will be used to calculate the gradient.

In real computation, $c(x,y)$ has to be discretized on a finite grids thus we can only estimate it on a finite number of grid points, although theoretically it can be a continuous function in some function space. Here we assume that $c(x,y)$ is estimated on the same grid as that on which the wave equation is solved.

If the size of the unknown coefficient is small, one can use finite difference quotient, which is a simple technique to calculate the gradient. However when the size of the unknown parameters increases, this method becomes too expensive to be practical.

The adjoint method has been widely used to calculate the gradient for large-size problem. In principle both continuous and discrete adjoint methods can be used to derive the gradient. Theoretically, there is no formal advantage of one method over another one in any general sense, however one method may be better suited to a given application.

In the method of continuous adjoint, the adjoint equation, which is derived using methods such as calculus of variations, is numerically solved to generate the gradient. While in the method of discrete adjoint, the numerical scheme for the direct problem defined in (1)-(4) is considered as the forward model, thus this approach actually computes the derivatives of the numerical solution, rather than approximating the derivatives of the exact solution. To derive the discrete adjoint, one can either take the adjoint of the forward numerical scheme directly, or apply automatic differentiation tool such as **TAMC** [11-13]. Since the direct approach is time-consuming and error-prone, the automatic differentiation, which builds a new augmented program based on the program to solve the forward problem, had been widely used. Another advantage of automatic differentiation is its quick and simple implementation, since the user just needs to provide a numerical program to calculate the cost functional and the automatic differentiation tool will generate an adjoint program which will compute the gradient.

V. Optimization Algorithms

We now provide a brief introduction to the optimization algorithm that will be used to solve the PDE-constrained optimization problem. **L-BFGS**, which stands for 'limited memory BFGS, will be used mainly because it is very efficient in solving optimization problem of large size parameters. It is well-known that **L-BFGS**, which implements Quasi-Newton method, uses the Broyden-Fletcher-Goldfarb-Shanno update to approximate the Hessian matrix. Consequently, unlike the regular **BFGS** algorithm, the **L-BFGS** only maintains a very short history of the past updates, thus the requirement on storage has been reduced significantly.

Let *f(x)* be the function to be minimized, *g(x)* be the gradient, and x_k be the result of k-th iteration, we can define $s_k = x_{k+1} - x_k$ ... and $y_k = g_{k+1} - g_k$, the L-BFGS method is defined as the following:

Step 1: Set initial guess X_0 and let K=0, choose parameter m, which is the number of BFGS corrections that are to be kept. Choose a sparse symmetric and positive definite matrix H_0, which approximates the inverse Hessian matrix of *f(x)*. Choose two parameters $\tilde{\tau}$ and τ as $0 < \tilde{\tau} < 1/2$ and $\tilde{\tau} < \tau < 1$.

Step 2: Compute $d_k = -H_k g_k$ and $x_{k+1} = x_k + \alpha_k d_k$, where α_k is a parameter satisfying the Wolf conditions:

$$f(x_k + \alpha_k d_k) \le f(x_k) + \tilde{\tau}\alpha_k g_k^T d_k \text{ and}$$

$$g(x_k + \alpha_k d_k)^T d_k \ge \tau g_k^T d_k .$$

Step 3: Let $\overline{m} = \min\{k, m-1\}$, and update the matrix H_k $\overline{m}+1$ times using the pairs $\{y_j, s_j\}_{j=k-\overline{m}}^{k}$ and the following formula:

$$\begin{cases} V_k = I - y_k s_k^T / (y_k^T s_k) \\ H_{k+1} = V_k^T H_k V_k + s_k s_k^T / (y_k^T s_k) \end{cases}$$

Step 4: Set $k = k+1$ and go to Step 2.

A detailed description of the algorithm can be found in [15,16]. It has been shown [17] that this package is very robust and converges faster than many other popular methods such as the widely used standard conjugate gradient method.

For comparison, we also tested three conjugate gradient optimization algorithms: the Fletcher-Reeves method, the Polak-Ribiere method and the Positive Polak-Ribiere method. Conjugate gradient method has been widely used to solve nonlinear optimization problem. Again suppose $f(x)$ is a smooth function to be minimized, and $g(x)$ is its gradient. Let x_0 be the initial guess, $g_0 = g(x_0)$, and x_k be the k-th iteration, the conjugate gradient method is implemented as the following:

$$d_k = \begin{cases} -g_k & k = 0 \\ -g_k + b_k d_{k-1} & k \ge 1 \end{cases}$$

$$x_{k+1} = x_k + \alpha_k d_k$$

where b_k is a scalar and α_k is a step length determined by means of a one-dimensional search. The three methods that are implemented in the package **CG+**[14] are almost identical with the only difference in the formula to calculate b_k.

In the Fletcher-Reeves method, $b_k = \|g_k\|^2 / \|g_{k-1}\|^2$, in the Polak-Ribiere method, $b_k = \langle g_k, g_k - g_{k-1} \rangle / \|g_{k-1}\|^2$, while in the positive Polak-Ribiere method, b_k is calculated as

$$b_k = \max\{\langle g_k, g_k - g_{k-1}\rangle / \|g_{k-1}\|^2, 0\}$$

thus only non-negative b_k is allowed in this method.

It is worthy pointing out that the user can use any other gradient-based optimization algorithm to solve the optimization problem, since the proposed computational method is very flexible, and a new component can be easily integrated. Also the definition of the cost functional may affect the choice of the optimization algorithm as well. For more details, the readers are referred to the papers mentioned above.

VI. Numerical Results and Discussions

We solve the initial-boundary value problem given in (1)-(5) defined on $[0,1]\times[0,1]$ and $t \in [0,1]$. Here $S(x,y,t)$, initial and boundary conditions are chosen accordingly so that $u(x,y,t)=e^{-t}(\sin(x)+\cos(y))$ and $c(x,y)=1/(1+x^2+y^2)$. In this example, we set the two regularity parameters in (6) be zeros, as we assume that we have no prior information about $c(x,y)$. We first validate the discrete adjoint generated by **TAMC** then illustrate the efficiency and accuracy of the proposed computational method.

To validate the discrete adjoint generated by **TAMC**, we first calculate the finite difference quotient approximations of $\partial J / \partial c_{i,j}$ at several selected grid points with various values of ε. The approximated derivatives are then compared with the results generated by **TAMC** at the same grid points. The numerical test is conducted on the grid with $\Delta t = 1/40$ and $h_x = h_y = 1/20$. Such grid is chosen to ensure the stability of the numerical method in (10), and that the numerical solution of the wave equation is accurate enough. Finally, the grid cannot be too fine otherwise the computational cost will be an issue. The results in Table I show that **TAMC** generates very accurate discrete adjoints. The first column contains the grid points where the adjoint is calculated, while column 2 to 4 contain the finite difference approximations of the adjoints by using various values of ε. As we can see, when ε is reduced from 0.1 to 0.001, the finite difference approximation converges to the results generated by **TAMC**, which is presented in column 5. Note that here we listed the results on 5 selected grid points only although it is possible to do so on all grid points, due to the space limit. However it is sufficient to show that **TAMC** is an effective and accurate tool to generate adjoint.

TABLE I: Comparison of Derivatives by Various Methods

(x_i, y_j)	$\varepsilon = 0.1$	$\varepsilon = 0.01$	$\varepsilon = 0.001$	TAMC
(0.1, 0.1)	-0.01536	-0.01617	-0.01613	-0.01614
(0.2,0.2)	-0.26358	-0.27542	-0.27667	-0.27680
(0.3, 0.3)	-0.17491	-0.18262	-0.18343	-0.18352
(0.4,0.4)	-0.14826	-0.15486	-0.15555	-0.15563
(0.5, 0.5)	-0.09188	-0.09596	-0.09639	-0.09644
(0.6,0.6)	-0.04497	-0.04697	-0.04718	-0.04721
(0.7, 0.7)	-0.01437	-0.01501	-0.01508	-0.01509
(0.8,0.8)	-0.00129	-0.00134	-0.00135	-0.00135
(0.9, 0.9)	0.00167	0.00122	0.00121	0.00121

As a measure of the accuracy of the computational method, we consider the difference in both maximal and root mean square norms, which are defined as the following

$$MAX = \max_{1\le i,j\le N} |c(x_i, y_j) - \tilde{c}_{i,j}| \tag{12}$$

$$RMS = \sqrt{\sum_{i,j=1}^{N} c(x_i, y_j) - \tilde{c}_{i,j})^2 / N^2} \tag{13}$$

where $\tilde{c}_{i,j}$ is the estimated value of $c(x_i, y_j)$.

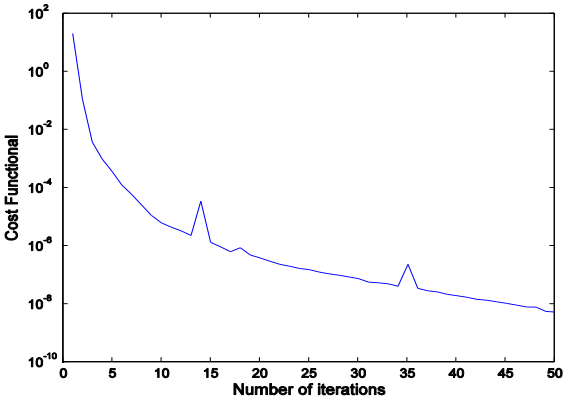

Fig. 1.Cost functional vs number of iterations

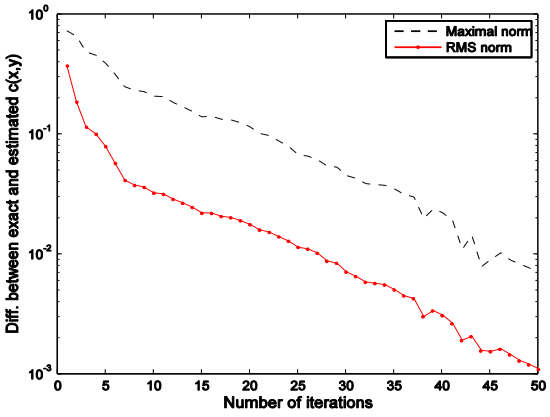

Fig. 2.Accuracy of the results vs number of iterations

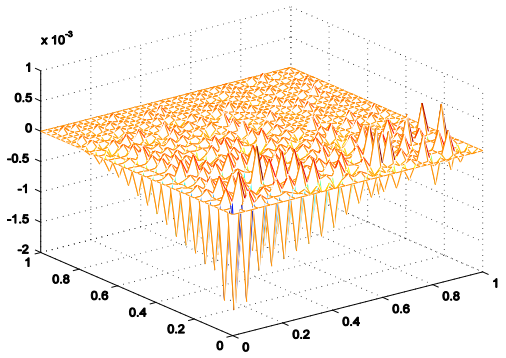

Fig. 3.2D view of the difference

The result presented in Fig. 1 shows that the gradient generated by **TAMC** is accurate and effective, since the cost functional reduces to zero rapidly(It is below 10^{-8} after 50 iterations). Consequently, the estimated $c(x, y)$ is also accurate as we can see from Fig. 2 that, both maximal and root mean square norms of the error are reduced to the level of 10^{-3} after 50 iterations. Finally, to see how accurate the

estimated $c(x, y)$ is, we plot the 2D view of the difference between the exact and estimated coefficient $c(x, y)$ in Fig. 3, which clearly demonstrates that the estimated coefficient is very accurate.

To investigate the impact of the optimization algorithm on the result, we include the results by using L-BFGS and three Conjugate-Gradient algorithms: the Fletcher-Reeves (F-R) method, the Polak-Ribiere(P-R) method and the positive Polak-Ribiere (P P-R) method in the following table. The comparison is conducted when the RMS error for each case satisfies the specified tolerance, but the total number of function evaluations are different thus the total CPU time for each case varies significantly. Remark: The CPU time in Table 2. is the average of three runs.

TABLE II:PERFORMANCE COMPARISON AMONG VARIOUS ALGORITHMS

RMS <	Number of function evaluations			
	L-BFGS	F-R	P-R	P P-R
1.00E-002	26	53	55	55
5.00E-003	42	89	83	84
1.00E-003	57	109	112	116
	CPU Time (Seconds)			
	L-BFGS	F-R	P-R	P P-R
1.00E-002	9.23	18.56	18.82	19.71
5.00E-003	17.28	38.11	36.79	37.71
1.00E-003	33.42	62.19	63.52	63.83

The data in Table II clearly shows that L-BFGS produces the most accurate result while the other three conjugate gradient algorithms are comparable. Note that the comparison is conducted for this specified problem, thus it is possible that a different conclusion will be obtained for a different problem. Nevertheless, here our goal is to show that the performance of the numerical method relies on the choice of the optimization algorithm.

VII. CONCLUSIONS

A computational method to estimate the acoustic coefficient $c(x_i, y_j)$ of a 2D acoustic wave equation is developed in this paper. Numerical results show that the proposed computational method provides a fast and robust approach to estimate the acoustic coefficient in the wave equation. It is well-known that the performance of the method depends on various factors such as the numerical schemes for solving the forward problem, the method to generate the adjoint code, the choice of regularity parameters, the optimization algorithm used to minimize the cost functional, etc, and these issues will be further investigated in our future work.

ACKNOWLEDGMENTS

This work is supported by the Natural Sciences & Engineering Research Council of Canada (NSERC).

REFERENCES

[1] A. I. Prilepko and D. G. Orlovskii, "Determination of the evolution parameter of an equation and inverse problems of mathematical physics, I and II", J. Differential Eq. 21(1) (1985) 119-129.
[2] A.G. Ramm and J. Sjostrand, "An inverse problem of the wave equation", *Mathematische Zeitschrift*, 206 (1991), 119 - 130.
[3] M.L. Gerver, "Inverse problem for the One-dimensional wave equation", *Geophysical Journal of the Royal Astronomical Society*, 21 (1970), 337- 357.
[4] G.Q. Xie, "A new iterative method for solving the coefficient inverse problem of the wave equation", *Communications on Pure and Applied Mathematics*, 39(3)(1986), 307 - 322.
[5] W. Liao, "A computational framework to estimate acoustic coefficient in a wave equation using boundary measurements", submitted.
[6] H. Lim, S. Kim and Jim Douglas Jr, "Numerical methods for viscous and nonviscous wave equations", Applied Numerical Mathematics, 57(2007), 194-212.
[7] S. Kim and H. Lim, "Higher-order schemes for acoustic wave-form simulation", Applied Numerical Mathematics, 57(2007), 402-414.
[8] V. Isakov, Inverse problems for partial differential equations, Springer, 2005.
[9] G. Shubin and J. Bell, "A modified equation approach to construction fourth order methods for acoustic wave equations", *SIAM J. Sci. Statisti. Comput.* 8 (1987) 135 - 151.
[10] T. Lin and R. Ewing, "Direct numerical method for an inverse problem of Hyperbolic equations", Numerical Methods for Partial Differential Equations, 8, pp. 551-574, 1992.
[11] Ralf Giering. "Tangent Linear and Adjoint Model Compiler", Users Manual, 1997.
[12] Ralf Giering and Thomas Kaminski. "Recipes for Adjoint Code Construction", Article in ACM Trans. Math. Software, 1998.
[13] Ralf Giering, "Tangent linear and Adjoint Model Compilor", http://www.autodiff.com/tamc.
[14] J. C. Gilbert and J. Nocedal, "Global Convergence Properties of Conjugate Gradient Methods for Optimization", (1992), *SIAM J. on Optimization*, 2, 1, pp. 21 -42.
[15] J. Nocedal. "Updating Quasi-Newton Matrices with Limited Storage". *Mathematics of Computation* 35, pp. 773-782, 1980.
[16] J. Nocedal. Department of Electrical & Computer Engineering, Northwestern University, http://www.ece.northwestern.edu/nocedal/lbfgs.html
[17] D.C. Liu and J. Nocedal. "On the Limited Memory Method for Large Scale Optimization", (1989), *Mathematical Programming B*, 45, 3, pp. 503-528.
[18] G.M.L. Gladwell, Inverse problems in vibration, Martin Nihhoff Publishers, Dordrecht, 1986.