USGS
science for a changing world

Computer Programs for Forward and Inverse Modeling of Acoustic and Electromagnetic Data

By Karl J. Ellefsen

Open-File Report 2011–1124

U.S. Department of the Interior
U.S. Geological Survey

U.S. Department of the Interior
KEN SALAZAR, Secretary

U.S. Geological Survey
Marcia K. McNutt, Director

U.S. Geological Survey, Reston, Virginia 2011

For product and ordering information:
World Wide Web: http://www.usgs.gov/pubprod
Telephone: 1-888-ASK-USGS

For more information on the USGS—the Federal source for science about the Earth, its natural and living resources, natural hazards, and the environment:
World Wide Web: http://www.usgs.gov
Telephone: 1-888-ASK-USGS

Suggested citation:
Ellefsen, K.J., 2011, Computer programs for forward and inverse modeling of acoustic and electromagnetic data: U.S. Geological Survey Open-File Report 2011–1124, 11 p. [Available at http://pubs.usgs.gov/of/2011/1124/].

Contents

Figures

Computer Programs for Forward and Inverse Modeling of Acoustic and Electromagnetic Data

By Karl J. Ellefsen

Abstract

A suite of computer programs was developed by U.S. Geological Survey personnel for forward and inverse modeling of acoustic and electromagnetic data. This report describes the computer resources that are needed to execute the programs, the installation of the programs, the program designs, some tests of their accuracy, and some suggested improvements.

1. Introduction

A suite of computer programs was developed by U.S. Geological Survey (USGS) personnel for forward and inverse modeling of acoustic and electromagnetic data. “Acoustic data” refer to waves that can be simulated with the acoustic wave equation. “Electromagnetic data” refer to waves that can be simulated with Maxwell’s equations. The source is either an infinitesimal electric dipole or an infinitesimal magnetic dipole. For both types of data, the forward modeling comprises calculation of Green’s functions, traces, traveltimes for the first-arriving wave, and amplitudes for the first arriving wave. For acoustic data and electromagnetic data generated with just an electric dipole, the inverse modeling comprises phase inversion and amplitude inversion.

This suite of programs performs only the numerical calculations for forward and inverse modeling. That is, the suite cannot be used for other essential procedures in forward and inverse modeling such as setting up models, plotting traces, plotting residuals, and so on.

The mathematical theory related to forward and inverse modeling of acoustic data is described in Ellefsen (2009). The mathematical theory related to forward and inverse modeling of electromagnetic data is described in Ellefsen and others (2007), Ellefsen and others (2009), and Ellefsen and others (2010 and 2011). Consequently, these theories are not repeated here. Instead, this report describes the computer resources that are needed to execute the programs, the installation of the programs, the program designs, some tests of their accuracy, and some suggested improvements.

The use of the programs is demonstrated with four examples that accompany this report. These four examples provide extensive, but not exhaustive, documentation of the input and output files. If more information about the input and output parameters is required, the code must be read. To this end, the reader must have some understanding of the C++ programming language.

2. Installation

2.1 Computer Resources

The programs, especially the program used for inversion, require a lot of computer memory (that is RAM, Random Access Memory). The amount depends upon the size of the model, the number of sources, and the number of receivers. Therefore, an exact minimum requirement cannot be specified. Nonetheless, for the four examples accompanying this report, approximately 4.5 gigabytes of memory is needed, beyond the memory that is needed for the operating system.

The programs used for calculation of Green’s functions and for inversion include a software module that solves a large system of linear equations (Ellefsen and others, 2009). This solution is calculated with software from the Intel Math Kernel Library (*http://www.intel.com/software/products/mkl*), which requires multiple computer processors to speed the calculations. Thus, the minimum number of processors is two. The optimal number of processors is difficult to specify because some limited tests suggest that it depends on the sparsity of the matrix in the system of linear equations.

The programs were compiled and executed within the Microsoft Windows 7 operating system that uses 64 bits. The 64-bit system is required because only it is capable of accessing large amounts of computer memory.

2.2 Procedures

Copy directories "Examples," "WaveformInv," "UtilityPrograms," and "Libraries" into a suitable location on your computer. The executable files are located in directories "WaveformInv\x64\Release" and "UtilityPrograms\x64\Release."

If you want to improve or maintain the programs, then you will have to compile and link the computer code. In this case, there are several additional installation steps that must be performed. The programs use a nonstandard C++ container that is called "ptr_vector." This particular container is from the Boost library (version 1.46.1 or later), which can be downloaded from the Boost web site *http://www.boost.org/*. Container ptr_vector as well as most of the Boost library are header files. If you want to use only this particular container, then there is no need to compile the non-header parts of the library.

The programs use components of Intel's Math Kernel Library to solve a large, sparse system of linear equations and to compute Fourier transforms. Thus, the easiest way to compile and link the programs is to use this library (version 10.3 or later) If you do not want to use this library, then consider using the Pardiso solver (for the system of linear equations), which is available from the web site *http://www.pardiso-project.org/*. In this case, you will probably have to edit the C++ classes that serve as interfaces. These classes are located in "Libraries\SparseLinearSystem." In addition, you will have to provide code to compute fast Fourier transforms and edit the C++ classes that serve as interfaces. These classes are located in "Libraries\MyUtilities."

The programs were developed with Microsoft Visual Studio 2008 (MVS), and the directories "WaveformInv," "UtilityPrograms," and "Libraries" are organized as Visual Studio solutions. Consequently, the easiest way to compile and link the programs is to use MVS.

3. Example Calculations

Within directory "Examples" are four sub-directories: Subdirectory "CalcRadarTraces" demonstrates forward modeling of crosswell radar data. Subdirectory "PhaseInvRadarTraces" demonstrates phase inversion of crosswell radar traces. Subdirectory "AmplitudeInvRadarTraces" demonstrates amplitude inversion of crosswell radar traces. Subdirectory "PhaseInvRefractionTimes" demonstrates phase inversion of seismic refraction traveltimes.

In directory "Examples" is a file called "ReadMe.txt," which provides information that is common to all examples. In each subdirectory is a file that is also called "ReadMe.txt," which provides information that is unique to that example.

4. Design

4.1 Organization of Files

The files with source code are located in the following directories: "UtilityPrograms," "WaveformInv3," and "Libraries." You can explore the files in these directories using a program such as "Windows Explorer." However, it is recommended that you use MVS because you can view the organization of the files (that is, the classes), the classes themselves, the properties of the classes, and so on. For the remainder of this section, it is assumed that you are using MVS.

The MVS solution "WaveformInv3\WaveformInv.sln." includes seven projects: (1) Project "DataVolume" is a library that comprises the classes that store and manipulate the data, which include traces, traveltimes, amplitudes, and so on. (2) Project "LinearizedSoln" is a library that comprises the classes that are needed to calculate a single iteration of an inversion. (3) Project "Misc" is a library that comprises the various miscellaneous classes that are needed for different programs. (4) Project "Model" is a library that comprises the classes that store and manipulate the model. (5) Project "WaveEqn" is a library that comprises the classes that simulate wave propagation and also calculates various wave-related quantities. (6) Project "GFCalculator" is a program that calculates Green's functions, and (7) project "WaveformInv" is a program that performs phase and amplitude inversion.

The MVS solution "UtilityPrograms\UtilityPgms.sln" includes four projects: (1) Project "GreensFunctionsWriter" is a program that converts the binary file produced by program GFCalculator into a text file. (2) Project "ReferenceValuesReset" is a program that changes the values for the reference model. This capability is occasionally helpful when performing a series of inversions. (3) Project "TACalculator" is a program that computes traveltimes and amplitudes of the first-arriving wave using the Green's functions calculated by program GFCalculator. (4) Project "TraceMaker" is a program that computes traces using the Green's functions calculated by program GFCalculator.

The MVS solution "Libraries\Libraries.sln" includes four projects: (1) Project "LamchForSuperLu" is a library that includes code for project SuperLu. (This code must be separated from that for SuperLu because it is compiled differently). (2) Project "MyUtilities"is a library that includes a wide variety of classes and functions that are needed by other programs. (3) Project "SparseLinearSystem" is a library whose classes serve as interfaces to other libraries that solve the large, sparse system of linear equations. (4) Project "SuperLu" is a library that comprises the public-domain software package "SuperLu," which is used to solve a large, sparse system of linear equations (see *http://crd.lbl.gov/~xiaoye/SuperLU/*). It is not used anymore because the solver in the Intel Math Kernel Library is much faster.

4.2 Program Designs

Program designs are shown graphically, using the unified modeling language, which is summarized in Fowler (2004). Likewise, object oriented programming is explained in many modern textbooks such as Stroustrup (1997). Consequently, neither the unified modeling language nor object oriented programming are described in this report.

All programs are executed from the Microsoft Windows console, which is also called the "Command Prompt." There are two reasons for using only the Windows console. First, the goal of the research project was to develop methods for forward and inverse modeling of radar data. The simplest and fastest means of achieving this goal was to use only the Windows console, not a graphical user interface. Second, code using the Windows console can be readily ported to other operating systems (for example, Linux) — no changes are needed in the code. In contrast, code with a graphical user interface probably would be difficult to port.

The class diagram for program WaveformInv is shown in figure 1. Details about each class, such as its methods and variables, can be obtained from the class diagrams available in MSV. Consequently, the details are not in the figures. The code for WaveformInv is very simple — its function is to instantiate class Iterative_Soln, which repeatedly minimizes the objective function until convergence occurs. A single minimization for the current objective function is performed by class Linearized_Soln. The appellation "linearized" indicates that the solution is obtained by linearizing the objective function about the current model. Class Linearized_Soln comprises four classes: (1) Class DataResidualEqn computes that portion of the gradient of the objective function that is due to the data residuals. This class comprises two subsidiary classes. DataVolume stores the data and the functions that perform various computations on the data. WaveEqn stores the functions that perform various computations related to wave propagation. (Class WaveEqn is large and somewhat complicated, so the details of this class are shown in figure 2). (2) Class ModelDeviationEqn computes that portion of the gradient of the objective function that is due to the difference between the current model and the reference model. (3) Class Gradient combines the two portions of the gradient into a single gradient and then processes the gradient to make it suitable for the minimization. (4) Class Model comprises the model itself and the functions that manipulate it (for example, the functions that update the model).

Each class in WaveformInv performs just one set of related calculations in the minimization of the objective function. For example, class gradient performs just those calculations related to the gradient. If necessary, a class calls subsidiary classes to perform subsidiary calculations. For example, class DataResidualEqn calls WaveEqn to compute Green's functions and various wave-related quantities. With this object-oriented design, related calculations are grouped together. Consequently, the likelihood of errors is reduced, and the code is easier to maintain than it would be otherwise.

Figures 3, 4, and 5 are sequence diagrams that show how three different quantities are calculated for program WaveformInv. These three sequence diagrams do not cover all important calculations. Each diagram is annotated to help you understand the steps in the calculations. Figure 4 pertains to one algebraic term that is used to calculate the step length in the conjugate gradient algorithm (Mora, 1987). In this figure, vector $\mathbf{c}$ is the conjugate direction; scalar ξ scales the conjugate direction; vector $\mathbf{d}(\mathbf{m})$ is the data (either phases or log-amplitudes) for the current model $\mathbf{m}$; vector $\mathbf{d}(\mathbf{m}+\xi\mathbf{c})$ is the corresponding data for the perturbed model $\mathbf{m}+\xi\mathbf{c}$; matrix $\mathbf{D}$ contains the Frechét derivatives, matrix $\mathbf{C}_d$ is the data covariance matrix, and scalar $(1/\xi^2)(\mathbf{D}\xi\mathbf{c})^T\mathbf{C}_d^{-1}\mathbf{D}\xi\mathbf{c}$ is the calculated term, where T indicates the transpose operation.

The class diagram for program GFCalculator is shown in figure 6. The code for GFCalculator is very simple: its function is to substantiate class GFCalculator_Impl. Class GFCalculator comprises three subsidiary classes: (1) DataVolume stores the data (namely, the Green's functions) and the functions that perform various computations on the data. (2) WaveEqn comprises the functions that perform various computations related to wave propagation. (3) Class Model stores the model itself. This design uses many of the classes that also are used for WaveformInv. The advantage of reusing code is that it reduces the effort to maintain the code. Furthermore, this reuse exploits the advantages of object-oriented design, which was previously described for program WaveformInv.

The designs of the other programs (namely, GreensFunctionsWriter, ReferenceValuesReset, TACalculator, and TraceMaker) are very simple, so they are not shown.

All code is written in standard C++, so all compilers that conform to the standard should be able to compile and link the code.

5. Discussion

5.1 Testing

Initial tests of forward modeling were conducted with homogeneous models for which analytic solutions exist and with layered models for which semi-analytic solutions exist. The analytic or semi-analytic solutions were used to calculate Green's functions, and these functions were compared to Green's functions calculated with program GFCalculator. Two such comparisons for radar data are published in Ellefsen and others (2009). Similar comparisons for acoustic data were performed but not published. In all comparisons, the Green's functions calculated with program GFCalculator were deemed accurate.

One extension of the previous tests involved calculating traces from the Green's functions using program TraceMaker. The traces based on the analytic or semi-analytic solutions were compared to the traces based on the solution from

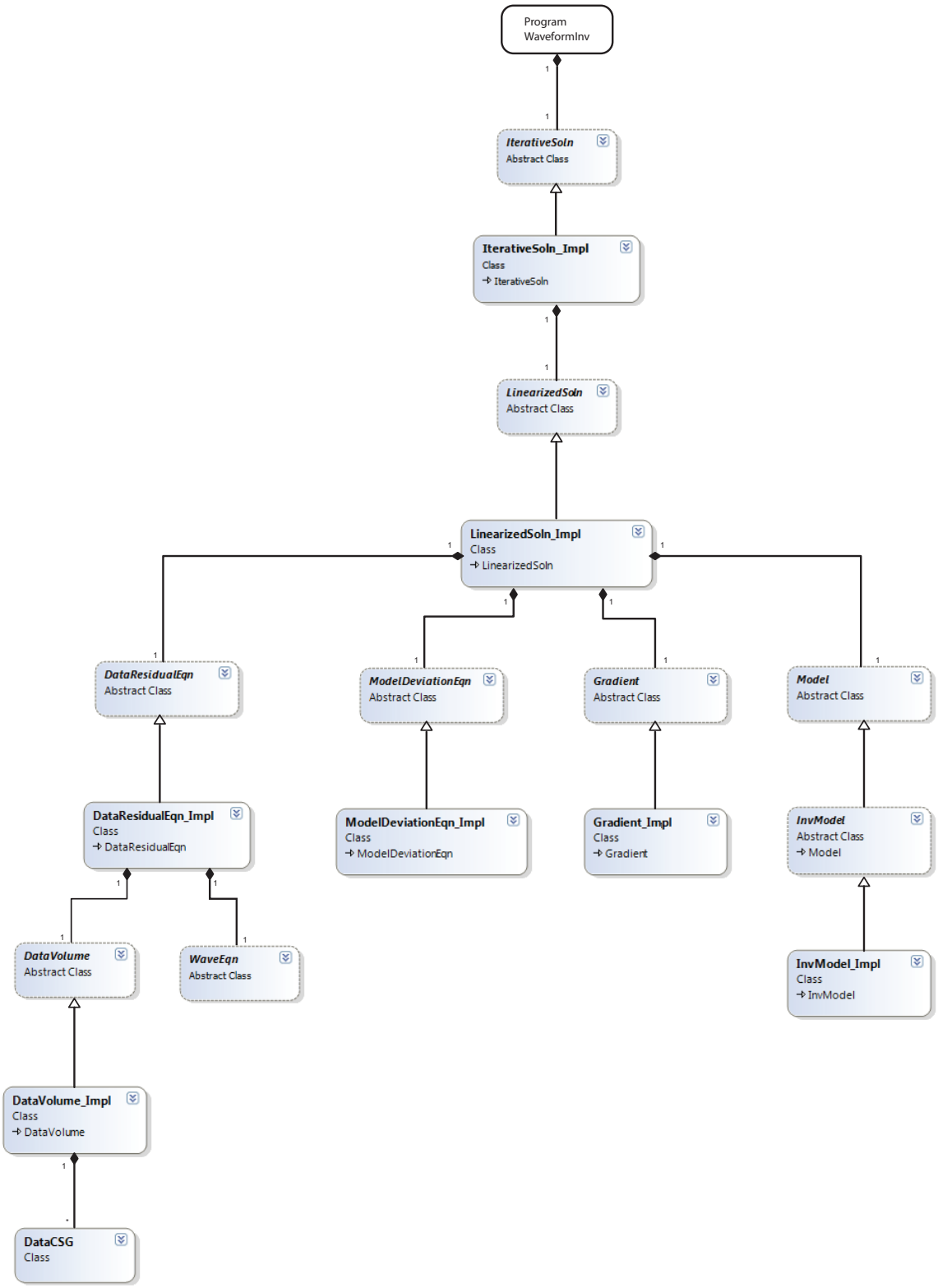

Figure 1. Class diagram for program WaveformInv. Minor classes were omitted. See figure 2 for the classes associated with WaveEqn.

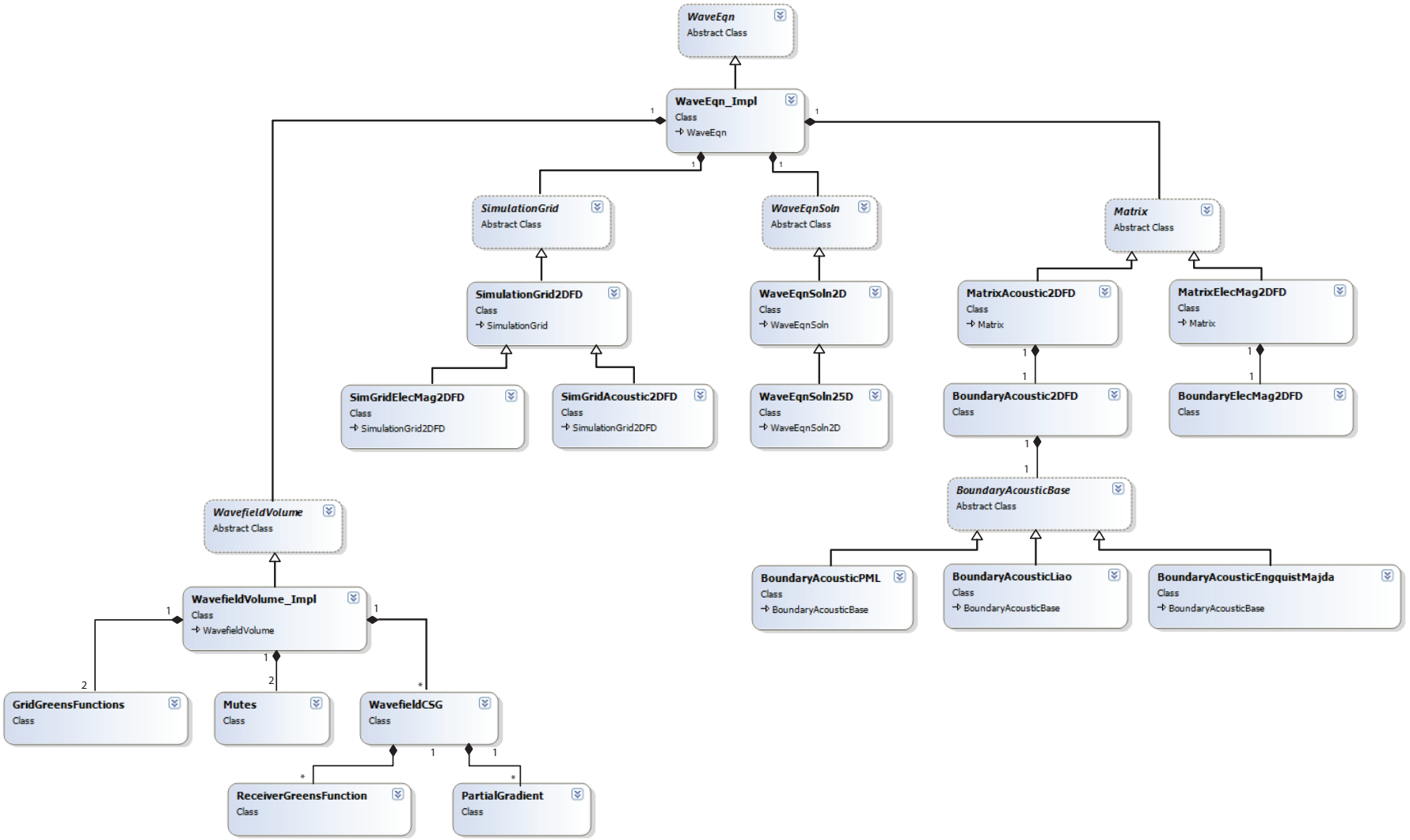

Figure 2. Class diagram for library WaveEqn. Minor classes and structures were omitted.

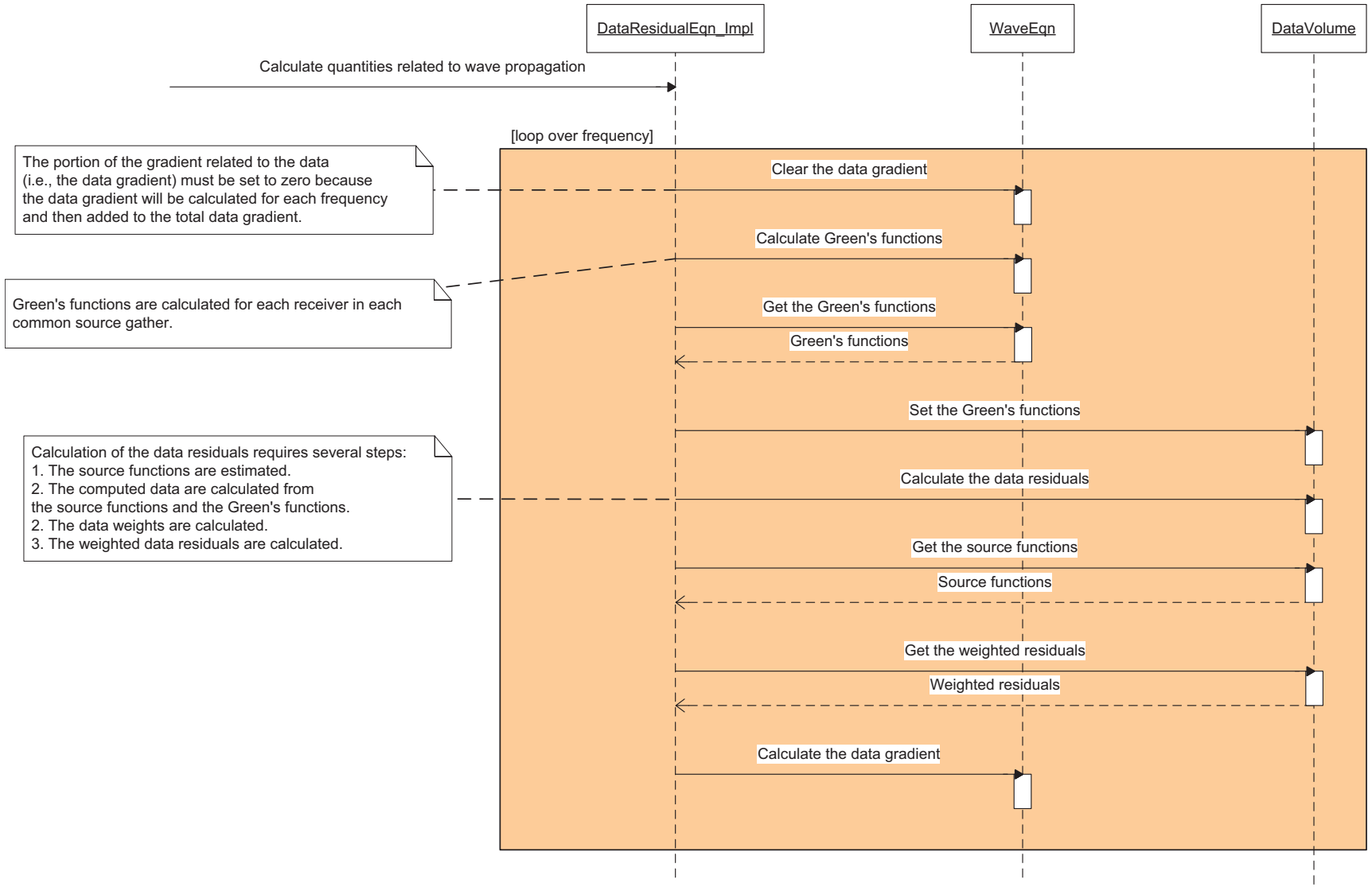

Figure 3. Sequence diagram showing how various quantities related to wave propagation are calculated during an inversion.

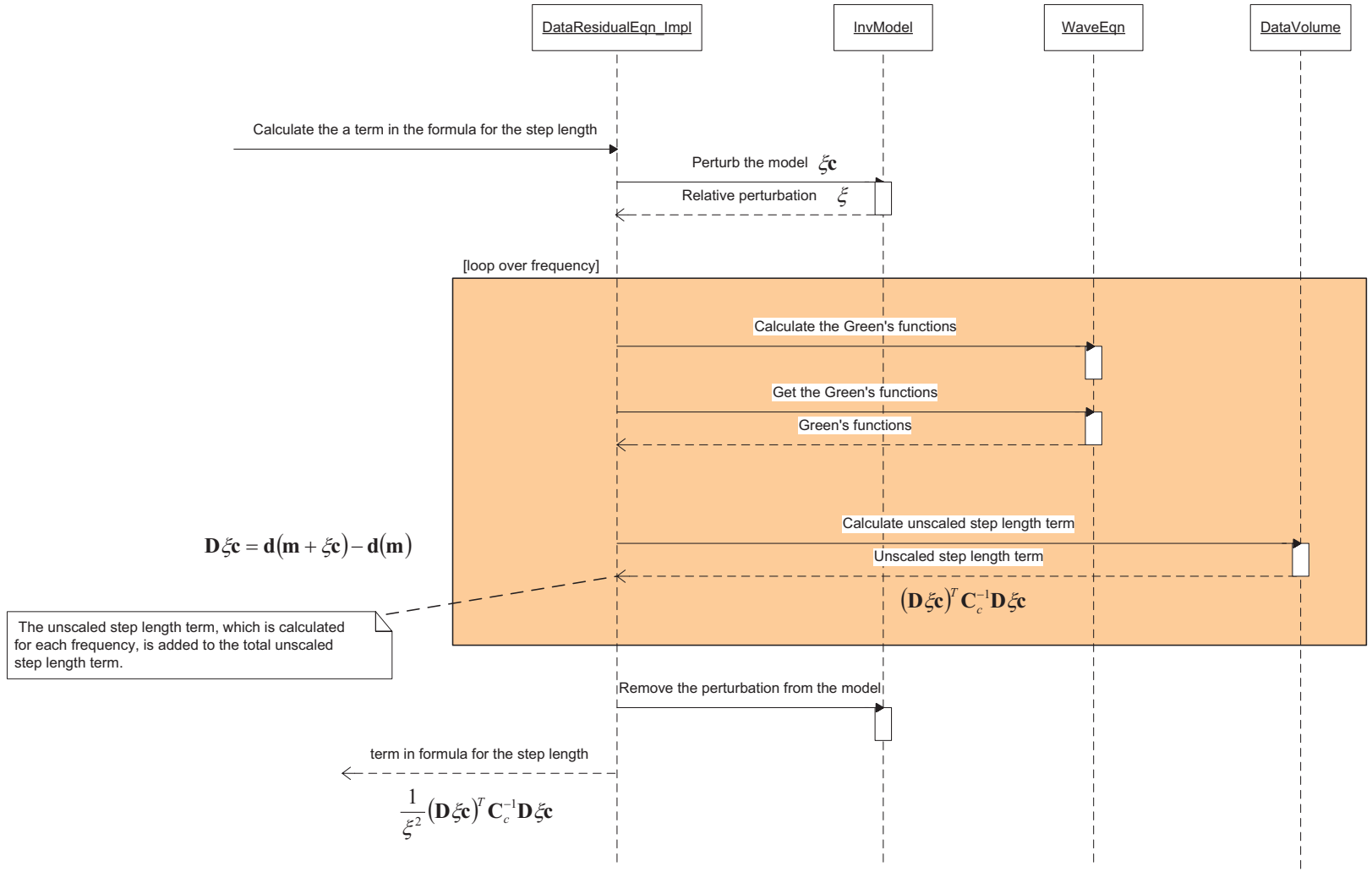

Figure 4. Sequence diagram showing how the step length is calculated during an inversion.

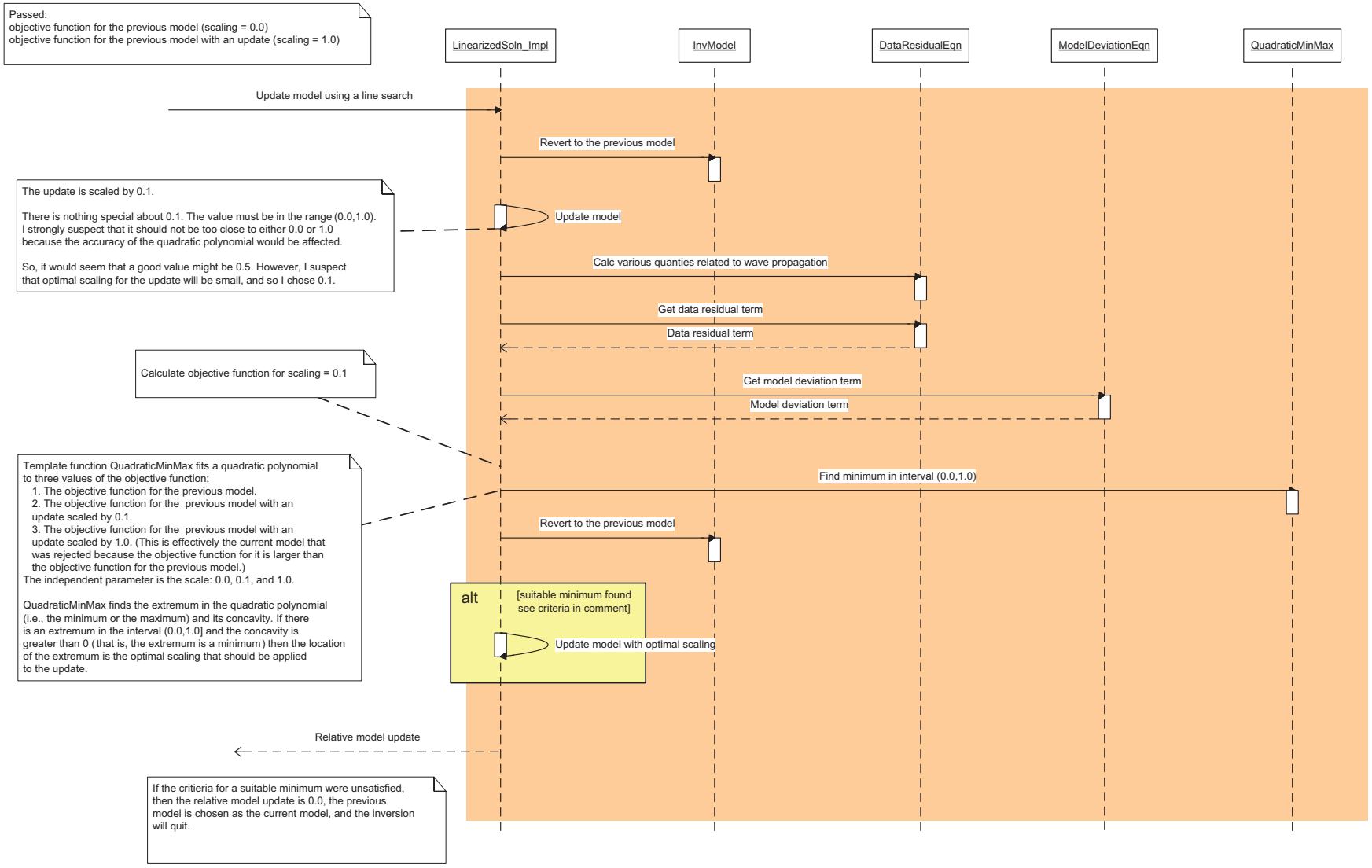

Figure 5. Sequence diagram showing how a model update is calculated with a line search.

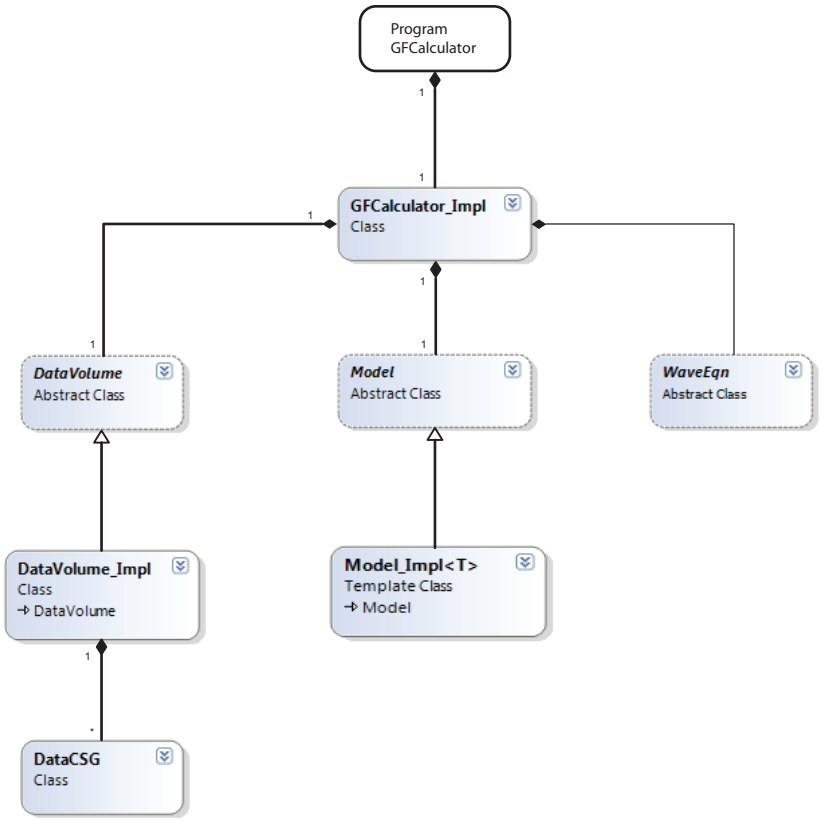

Figure 6. Class diagram for program GFCalculator. Minor classes were omitted. See figure 2 for the classes associated with WaveEqn.

GFCalculator. In all comparisons, the GFCalculator traces were deemed accurate. The results of these tests were not published.

Another extension of the previous tests was forward modeling of crosswell radar data, which were collected between two wells that penetrate igneous and metamorphic rock. For this test, the calculated traces were compared to the field traces. The traveltimes were identical, and the trends in the amplitudes were identical. This result suggests that programs GFCalculator and TraceMaker are working properly. Additional details of the test are described in Ellefsen and others (2009), and the files used for the forward modeling are in subdirectory Examples\CalcRadarTraces.

Initial tests of inverse modeling were conducted by calculating traces with programs GFCalculator and TraceMaker for a hypothetical model. Program WaveformInv processed these traces to estimate a model, which was compared to the original hypothetical model. This test evaluated the portion of program WaveformInv that performed the inversion, but not the portion that calculated the Green's functions. (That is, the portion that calculated the Green's function was used for both forward and inverse modeling. Consequently, errors in this portion would not be apparent). Despite this shortcoming, these initial tests showed that most of program WaveformInv worked properly. Furthermore, these tests helped establish the spatial resolution of the inversions. An example of these initial tests for crosswell radar data is published in Ellefsen and others (2007).

Moderately extensive tests were performed for phase inversion of seismic refraction traveltimes. One set of tests used hypothetical models for which traveltimes were calculated with a finite-difference implementation of the eikonal equation. (That is, the traveltimes were not computed with programs GFCalculator and TACalculator.) The traveltimes were processed with program WaveformInv to estimate a model, which was compared to the correct model. In all cases, the estimated model was close to the correct model, when the spatial resolution of the inversion was accounted for. Additional information about these tests is published in Ellefsen (2009) and Burton and Ellefsen (2011). Another set of tests involved field data. Traveltimes were processed with phase inversion (using program WaveformInv) and with traditional traveltime tomography; both processing methods used the same starting model. The estimated models and the residuals for both processing methods were compared, and the results are published in Ellefsen (2009). For this test, the files used for the phase inversion are in subdirectory Examples\PhaseInvRefractionTimes.

Finally, phase and amplitude inversion were used to process crosswell radar collected in a sand tank. For the phase inversion, the model anomalies corresponded to the

significant heterogeneities within the tank; similar results were obtained for the amplitude inversion. In addition, the dielectric permittivities were calculated from the model estimated with the phase inversion. This range of permittivities was consistent with the value for a small sample that was independently measured in a laboratory (Peters and others, 2010). Additional details are published in Ellefsen and others (2011). The files used for the phase inversion are in subdirectory Examples\PhaseInvRadarTraces, and the files used for the amplitude inversion are in subdirectory Examples\AmplitudeInvRadarTraces.

All tests were conducted for complete programs. Although such tests are very important, tests of program components like individual classes and libraries are also important but were not performed. Such component tests should be developed because they would find errors and thereby significantly improve the reliability of the code. Procedures for developing component tests are described in McConnell (2004).

5.2 Suggested Improvements

The suite of programs could be improved in several different ways. First, the programs should be extended to make them capable of processing seismic traces. This extension involves developing code to simulate waves with the elastic wave equation. The seismic models must be capable of representing the ground surface, including a surface with variable topography. Second, the programs should be rewritten to make them execute in parallel and thereby decrease the execution time. (This revision applies to all code except that from the Intel Math Kernel Library, which already executes in parallel). Third, program WaveformInv should be revised to reduce the amount of required memory.

Additional improvements are listed in file WaveformInv3\Improvements.txt.

5.3 Miscellaneous Issues

Recall that the suite of computer programs is capable of forward modeling of electromagnetic waves generated by an infinitesimal magnetic dipole (see the Introduction , section 1). This type of simulation has not been tested yet. Also, the code for inverse modeling of such data has not been written yet, although it readily could be.

The known bugs are listed in file WaveformInv3\Bugs.txt.

6. Acknowledgments

The U.S. Environmental Protection Agency (USEPA), through its Office of Research and Development, partially funded and collaborated in the research described here under Interagency Agreement DW–14–93964101–1 with the U.S. Geological Survey. Additional funding was provided by the Mineral Resources Program of the U.S. Geological Survey. Philip J. Brown II and Craig W. Moulton of the USGS reviewed this report.

7. References

Burton, B.L. and K.J. Ellefsen, 2011, Phase inversion of refraction traveltimes, in Proceedings of the Annual Symposium on the Application of Geophysics to Engineering and Environmental Problems, April 10–14, 2011, Charleston, South Carolina: Environmental and Engineering Geophysical Society. (Available on CD at *http://www.eegs.org/*).

Ellefsen, K.J., 2009, A comparison of phase inversion and traveltime tomography for processing near-surface refraction traveltimes: Geophysics, v. 74, no. 6, WCB11–WCB24, doi:10.1190/1.3196857.

Ellefsen, K.J., Mazzella, A.T., Moulton, C.W., and Lucius, J.E., 2007, An algorithm for waveform inversion of crosswell radar data, in Proceedings of the Annual Symposium on the Application of Geophysics to Engineering and Environmental Problems, April 1–5, 2007, Denver, Colorado: Environmental and Engineering Geophysical Society. (Available on CD at *http://www.eegs.org/*).

Ellefsen, K.J., Croizé, Delphine, Mazzella, A.T. , McKenna, J.R. , 2009, Frequency-domain Green's functions for radar waves in heterogeneous 2.5D media: Geophysics, v. 74, no. 3, J13–J22, doi:10.1190/1.3092776.

Ellefsen, K.J., Mazzella, A.T., Horton, R.J., and McKenna, J.R., 2010, Frequency domain, waveform inversion of laboratory crosswell radar data: Society of Exploration Geophysicists Expanded Abstracts, v. 29, p. 4019–4023, doi:10.1190/1.3513697.

Ellefsen, K.J., Mazzella, A.T., Horton, R.J., and McKenna, J.R., 2011, Phase and amplitude inversion of crosswell radar data: Geophysics, v. 76, no. 3, G1–G12, doi:10.1190/1.3554412.

Fowler, Martin, 2004, UML distilled—A brief guide to the standard object modeling language (3d ed.): Boston, Pearson Education, Inc., 175 p.

McConnell, Steve, 2004, Code complete: Microsoft Press, 960 p.

Mora, Peter, 1987, Nonlinear two-dimensional elastic inversion of multioffset seismic data: Geophysics, v. 52, no. 9, p. 1211–1228.

Peters, B., Moulton, C.W., Ellefsen, K.J., Horton, R.J., and McKenna, J.R., 2010, High-frequency, crosswell radar data collected in a laboratory tank: U.S. Geological Survey Digital Data Series 486. Accessed on Jan. 2011 at *http://pubs.er.usgs.gov/usgspubs/ds/ds486*.

Stroustrup, Bjarne, 1997, The C++ programming language (3d ed.): Reading, Mass., Addison-Wesley Publishing Co., 1,040 p.

Publishing support provided by:
Denver Publishing Service Center

For more information concerning this publication, contact:
Center Director, USGS Crustal Geophysics and Geochemistry Science Center
Box 25046, Mail Stop 964
Denver, CO 80225
(303) 236-1312

Or visit the Crustal Geophysics and Geochemistry Science Center Web site at: *http://crustal.usgs.gov/*