Determination of Elastic Moduli from Measured Acoustic Velocities

J Michael Brown
Earth and Space Sciences
University of Washington
Seattle WA 98195
brown@ess.washington.edu

draft October 15, 2013

Abstract

The function, `Velocities2Cij`, allows determination of elastic moduli and their uncertainties from body wave and surface wave phase velocities measured in anisotropic materials. Three inverse algorithms are provided: the gradient-based methods of Levenberg-Marquardt and Backus-Gilbert, and a non-gradient-based Nelder-Mead simplex approach. Velocity data can be mixed or not (body wave velocities alone, surface wave velocities plus axes compressibility constraints, or joint body and surface wave velocities). Both experimental data and synthetic results are tested using the algorithms. Although all implemented approaches succeed in finding solutions, the Levenberg-Marquardt method consistently demonstrates greater skill and speed. In contrast, the simplex method, while slower to find solutions, is less susceptible to convergence problems sometimes encountered with linearized gradient-based methods. Several published datasets are reexamined. Uncertainties based on the diagonal of the covariance matrix and those estimated using Monte Carlo simulations are generally in accord and agree with most published estimations. The current package of functions therefore provides a robust, validated, and flexible environment for analysis of ultrasonic, Brillouin, or Impulsive Stimulated Light Scattering datasets.

Keywords: elastic constants, elastic moduli, acoustic waves, surface waves, body waves, ultrasonic, Brillouin, impulsive stimulated scattering, non-linear least-square parameter estimation

Highlights:

- A convenient numerical framework is provided to analyze measured body and surface wave velocities to determine elastic moduli
- Three optimization methods are provided: gradient-based least-square optimizers of Levenberg-Marquardt and Backus-Gilbert and the Nelder-Mead simplex optimizer
- The methods are validated against published and synthetic data. Results appear robust and accurate

Introduction

Determinations of the elastic moduli for anisotropic crystals figure into several science and technical agendas including condensed matter physics (evaluating inter-atomic forces), material sciences (determining technical properties of materials), and the geosciences (interpreting Earth's seismic velocity structure). In the case of high symmetry crystals, analytic equations provide relatively simple relationships between moduli and velocities measured in a small number of specified directions (Every 1980). However, both in the case of low symmetry crystals (requiring a large number of measurements to constrain larger numbers of moduli) and when measurements are made in arbitrary directions relative to symmetry elements, numerical inversion of velocities to moduli is necessary. An interest in low symmetry crystals (*e.g.* more than 50% of all minerals are either monoclinic or triclinic) and the use of surface wave acoustic measurements (Brown et al 2006) provides impetuous to develop and document methods that can be applied to the determination of moduli under a variety of experimental conditions for all crystal symmetries.

In an early example of computer-aided numerical analysis (Aleksandrov et al 1974) all 13 elastic moduli required for monoclinic elasticity of common rock forming minerals were reported. As noted in Brown et al (2006), that work, which was based on a small number of measurements, was unable to identify and characterize the large uncertainties in the reported moduli. Weidner and Carleton (1977) ushered in a modern era of moduli determination for low symmetry minerals using Brillouin spectroscopy. They gave details of a specialized numerical method based on Backus-Gilbert inversion (1968, 1970) to determine elastic moduli. Motivated by the need to analyze measurements of body wave and surface wave velocities obtained by Impulsive Stimulated Light Scattering, several strategies to determine moduli and to characterize uncertainties have been reported (Brown et al. 1989, 2006, 2014, Abramson et al. 1994, 1997, 1999, Chai et al 1997, Collins and Brown 1998, Crowhurst et al 2001, Waeselmann et al 2014). Here experience developed in the course of these studies is documented. Cross-comparisons of different inverse techniques have not previously been reported nor have results been validated through use of a common set of published and synthetic examples.

Here a set of utilities and a suite of inverse techniques are assembled into a package of MATLAB functions that are transportable to all common computer platforms. A small set of command line instructions allows flexible optimization and visualization of results. The underlying approaches to the inverse problem are articulated and sets of actual and synthetic velocities are assembled to test and explore the capabilities of the functions. Also included are functions to create graphical representations of fits and model predictions.

Methods

Forward Problem

All inverse techniques require a well-defined forward calculation and determination of acoustic phase velocities as a function of elastic moduli, density, and propagation direction is relatively straightforward. Given the 4th order tensor elastic moduli, C_{ijkl}, and the material density ρ, with velocities, $\boldsymbol{v}$ (equal to $\boldsymbol{k}/\omega$ where $\boldsymbol{k}$ is the wave vector and ω is the frequency), elastic wave propagation is governed (*e.g.* Auld 1973 and many other basic texts) by:

$$\rho \frac{\partial^2 u_r}{\partial t^2} = C_{lrms} \frac{\partial^2 u_s}{\partial x_l \partial x_m} \quad (1)$$

where subscripts refer to the three Cartesian coordinates and u_i are displacements. For body waves, a trial solution in the form of a plane wave $u_r = U_o \exp\left(i(k_i x_i - \omega t)\right)$ when substituted into equation 1, leads to a secular equation that can be solved for velocities:

$$det|A_{rs} - \rho v^2 \delta_{rs}| = 0 \quad (2)$$

where the Christoffel matrix A_{rs} is defined in terms of the elastic moduli tensor and direction cosine components, n_i, as $C_{rlsm}\ n_l\ n_m$ (using the Einstein summation convention). The three eigenvalues of the matrix defined in equation 2 give ρv^2 for the three (quassi) longitudinal and (quasi) transverse modes while the eigenvectors define wave polarizations.

In the case of wave propagation on surfaces of anisotropic materials, Rayleigh-like surface acoustic waves (SAW) exist for all propagation directions and pseudo-surface waves (PSAW) (waves that leak acoustic energy into the sample interior) can propagate under more restrictive conditions (Maznev et al. 1999). Equation 1 can be numerically solved by application of appropriate boundary conditions (Farnell 1970). The computational procedure developed by Every et al. (1998) is used in the current analysis. An elastic Greens function G_{ij} solution is found for a line-source forcing function. The procedure is general and can be applied to any combination of crystal symmetries and orientations.

Using impulsive stimulated light scattering, (Chai et al, 1997, Abramson et al 1999, Crowhurst et al 2001) SAW and PSAW have been observed at 1 bar and in high pressure experiments. Crowhurst and Zaug (2004) note additional surface skimming quassi-longitudinal modes. Brown et al (2006) determined all elastic moduli of a triclinic mineral from observations of SAW and PSAW. As recommended by Maznev et al. (1999) and further tested in Crowhurst and Zaug (2004) and Brown et al (2006), the intensities of observed signals correlate best with the off-diagonal elastic Greens function tensor element $|G_{13}|^2$.

Inverse Problem

The inverse process of determining elastic moduli from measured elastic wave velocities is undertaken within the framework of non-linear least-square parameter estimation (Aster et al 2012). Three methods, described below in greater detail, have shown skill in solving the current problem. Additional methods and ideas are briefly mentioned. Generally, increments of parameters are found relative to an initial guess that reduces the misfit as measured by the square of deviations between data and model prediction. The process is repeated until misfit ceases to decrease. Parameters that provide the smallest value of misfit are taken to be the solution if experimental uncertainty and misfit are in accord. Regularization (the use of additional constraints) can help optimization by steering solutions in appropriate direction and/or stabilizing ill-conditioned numerical problems.

A “local” solution may exist that has larger misfit than the true “global” minimum. In such cases, *a priori* knowledge of experimental uncertainty is invoked to reject the solution. Finding the smallest misfit is possible using any method that successfully increments parameters to reduce misfit. A grid search of the entire hyper-surface of misfit while computationally tedious would also locate the global minimum. Gradient methods (based on a local determination of misfit derivatives) are computationally more efficient. However, such methods can be trapped in regions with low gradients of misfit or in local minima. Differences in numerical strategies to find global minima in misfit can be characterized, as illustrated in figure 1, from “exploitive” (following a gradient defined path to achieve smaller misfit) to “exploratory” (brute force grid search). Gradient methods, lying near the exploitive axis, while typically requiring the fewest calculations, are most susceptible to being trapped in local minima.

The framework of gradient-based approaches is to either calculate local derivatives of misfit and move in the direction of smaller misfit (steepest descent method) or to undertake a parabolic expansion of the misfit surface and thus locate the minimum in a single step (Gauss-Newton method). The hybridization of these approaches that underlies the numerical algorithm of Levenberg-Marquardt (Marquardt 1963) is described below. A modification of this that includes an additional layer of regularization based on the Backus-Gilbert (1968, 1970) approach is also described.

Key concepts of gradient-based least-square solutions are noted here. A model $f(m)$ with discrete parameters m is adjusted to best represent data y_{obs}. Adjustments to the model can be determined by expanding f relative to initial parameters m_o:

$$f_i(m) = f_i(m_o) + \frac{\partial f_i}{\partial m_k}\delta m_k + \cdots \tag{3}$$

where higher order derivatives are ignored and subscript i is the index relating to the i^{th} data point and k is the index for the k^{th} discrete model parameter. The matrix of partial derivatives $\partial f_i/\partial m_k$ of the model with respect to model parameters, the Jacobian, is represented as $\boldsymbol{J}$. The Jacobian determines the steepest descent direction and the simplest estimation of model increments in the direction of smaller misfit is given by:

$$\delta m = \boldsymbol{J^t}[y_{obs} - f(m_o)] \qquad 4$$

where y_{obs}, $f(m_o)$ and δm are vectors and $\boldsymbol{J}$ is a matrix.

In order to derive the Gauss-Newton method, the least-square problem is expressed as the minimization of misfit S where:

$$S = \|y_{obs} - f(m_o) - \boldsymbol{J}\delta m\|^2 \qquad 5$$

Double brackets with superscript 2 imply summation of squared differences. Setting the derivative of equation 5 to zero with respect to model parameters m and neglecting derivatives of f beyond the first leads to the linearized least-square solution for increments of model parameters:

$$\delta m = (\boldsymbol{J^t J})^{-1}\,\boldsymbol{J^t}[y_{obs} - f(m_o)] \qquad 6$$

where superscript t is the transpose operation. If m_o is linearly close to the minimum in misfit and if the neglected higher-order derivatives of $f(m)$ with respect to m are small, then equation 6 should allow convergence to the true minimum in one step.

The insight provided in the Levenberg-Marquardt method (Marquardt 1963) is that a gradient-based descent path is preferred far from the misfit minimum and that the step size should be scaled by the local curvature (*i.e.,* larger steps for the smaller curvature expected far from the minimum). The linearized Gauss-Newton solution and a modified steepest descent method are then combined in a single increment estimator as:

$$\delta m = [\boldsymbol{J^t J} + \lambda diag(\boldsymbol{J^t J})]^{-1}\boldsymbol{J^t}[y_{obs} - f(m_o)] \qquad 7$$

where the factor λ is adjusted. Large values of λ are used when far from the minimum emphasizing the gradient estimator of equation 4 and small values are used near the minimum such that equation 7 tends towards equation 6. The schedule for changing λ is arbitrary and can be "tuned" to provide better performance for specific problem classes.

Weidner and Carleton (1977) determined moduli increments through an implementation of Backus-Gilbert (1968, 1970) regularization of equation 7. Backus-Gilbert regularization was originally formulated for ill-posed inverse problems consisting of continuous model functions rather than for relatively well-posed models consisting of discrete parameters. The power of the Backus-Gilbert approach lay in its ability to determine model resolution at arbitrary points rather than for any special ability to estimate discrete parameters. "Resolving power" is not a well-defined concept in the case of discrete parameters. The basic idea of Backus-Gilbert regularization is that some linear combination of observations and model derivatives should best determine increments of a specified model parameter while having little or no influence on other model parameters. The increment equation is given as:

$$\delta m = \alpha[y_{obs} - f(m_o)] \quad 8$$

where each row of the matrix α is constructed independently for each parameter. To determine α, a matrix consisting of components of the Jacobian and model misfit is inverted separately for each model parameter. The matrix for each parameter follows equation 7 with additional rows and columns to implement the regularization constraint.

Nothing unique to elasticity is found in the application of Backus-Gilbert regularization to fitting velocity data. Aster et al (2012) note that Backus-Gilbert techniques have not been widely adopted as a result of their numerical complexity and a perception that the method has no clear advantage over other approaches. In examples discussed in this paper, the Backus-Gilbert method shows less skill in incrementing model parameters than the standard Levenberg-Marquardt method. However, any method that successfully adjusts parameters to achieve smaller data misfit can provide an adequate approach. The method is essentially a gradient-based algorithm and is included in the current library.

A non-gradient method, the simplex algorithm of Nelder-Mead (1965), is also included in the current collection of algorithms to expand the repertoire of methods that can be applied to the problem. This algorithm tends to sample a larger portion of the misfit hyper-surface and is colloquially called an "amoeba" fitter. Rather than calculating local gradients of misfit, a collection of models is used to define a volume in the misfit hyperspace (with as many dimensions as parameters). Each model in the current set forms a vertex of the multidimensional shape. "Pseudopods" (based on symmetry operations such as reflection and contractions of a current vertex) are extended in various directions to see if a smaller misfit can be found. When smaller misfit is found, the largest misfit in the current collection of models is discarded, thus moving the volume of trial models in a direction of smaller misfit. The volume expands or contracts as it moves and, if successful, eventually centers and shrinks around the misfit minimum. Simplex methods are generally less susceptible to being trapped in local minima. Even if the starting point is a local minimum, the expanded search region represented by the multidimensional collection of vertexes provides an opportunity to move into a region with a gradient adequate to steer the iterative process to a better minimum.

Multi-start strategies are characterized by initiating optimization at more than one location. In situations with a finite number of local minima, a properly designed algorithm can be implemented that recognizes when a particular search is trending to a previously discovered minimum. Thus, not all searches need be followed to completion. Here, with each search requiring relatively little computer time, mult-start is implemented simply by restarting optimization from a new and randomly generated location until a satisfactory solution is identified.

Other methods such as genetic (Gallagher and Sambridge 1994) and simulated annealing (Kirkpatrick et al 1983) illustrated in Figure 1 have the ability to locate the global minimum in cases where misfit surfaces are complex and may have many local minima. The cost is typically in the need to sample more possible solutions and thus these algorithms extend towards the exploratory side of the figure where computational effort is larger. The generally exploitive approaches used in the current application have demonstrated an ability to find elastic moduli in all test cases. Thus, the more computational intensive methods do not appear to be necessary.

A number of situations can cause the process of incrementing moduli to stall at an unacceptable solution. Reasons for this can be identified. (1) Large experimental scatter relative to variation with direction provides an opportunity for "non-linear" scatter in estimated parameters. Several "local" minima may adequately fit the data. Thus, the curvature about each minimum may not reflect actual uncertainties in the moduli and calculated gradients may not point in the direction of better solutions. (2) The subset of experimentally determined velocities might not include data that are adequately sensitive to one or more of the parameters (the data do not adequately span the parameter space). (3) The data may be sensitive to a particular linear combination of parameters such that the combination is better constrained than are individual values. (4) Velocities can accidentally be assigned to incorrect acoustic branches. A common problem arises when deciding how to associate shear wave observations with calculated polarized waves. In the case of surface waves, differentiating between Rayleigh waves and various pseudo surface waves requires experience and some trial and error experimentation.

Uncertainties are estimated on the basis of the curvature of the misfit where $\mathbf{J}^t\mathbf{J}$ is taken to adequately represent the second derivative of the model with respect to model parameters. The inverse of $\mathbf{J}^t\mathbf{J}$ is the covariance matrix and the diagonal of the covariance matrix when appropriately weighted by experimental uncertainties gives the estimate of parameter variances. An alternate approach is to undertake Monte Carlo simulations (Astor et al 2012). An ensemble of alternate synthetic data sets, each with a distribution of propagation directions equivalent to experiment, is created. Velocities calculated from a reference set of moduli are perturbed with random error having the same statistical distribution as observed in experiments. Each member of the ensemble is inverted and provides an independent estimate of the model as if an entirely independent data set had been collected. In well-determined systems, the standard deviation of the ensemble of synthetic moduli should agree with the covariance estimate of uncertainty.

Implementation

The inverse algorithms described in the previous section have been implemented within the numerical environment of MATLAB. Although a GUI could be constructed, here the workflow is accomplished at the command line by invoking a small number of functions. The analysis steps are (1) load experimental data into the workspace as a structure containing heterogeneous information (both text and numerical), (2) execute the fitting function (once or multiple times), (3) graphically examine the quality of the fits, (4) save results in a compact (structure) form. Once data are appropriately organized (wave polarizations are correctly identified and problem data are appropriately weighted by their estimated experimental uncertainty), the process of optimization is nearly instantaneous on modern desktop computers.

Data Organization

All input data and fitting options are contained in a single structure, called "Input" that is passed between all functions. Units are GPa for moduli, TPa^{-1} for compliances, km/s for velocities, and gm/cm^{-3} for density. Angles are in degrees. Sets of example data-containing functions are included that use published or synthetic data and organize the information into the requisite structure. Files with names `mkStrXXX`, where XXX is a descriptive label for the material, illustrate construction of the input data structure. These files can serve as templates in working with new or different data sets. It is relatively easy to copy and paste data tables from other sources into an edit window. Within the `mkStrXXX` function, data are parsed into appropriate structure variables. In a few published data sets, propagation directions have been in relatively broadly distributed directions. Most data, however, are obtained from propagation directions in planes determined by sample slices or rotations about a single axis. Orientations for such sample slices are conveniently represented using three Euler angles. All measurements sharing a common axis of rotation are therefore grouped as one "sample" within the structure.

The input structure contains two major divisions: "Data" and "opts". All information in the data side (velocities, uncertainties, measured Euler angles and/or direction cosines, sample density, chemistry, comments, published moduli) is not changed during the optimization process. The "opts" side of the structure contains information that may be changed during optimization or is set by the user to control the optimization process.

Included on the data side of the structure are the "trust regions" for moduli and Euler angles. By defining a region of sensible results, optimization is better guided. For example, the requirement that the elastic moduli tensor be positive definite requires that some moduli have positive values. In many cases, the trust region can be set broadly. In some cases, constraining the region of acceptable solutions provides assistance and is justified by *a priori* knowledge. If solutions tend to converge at the edge of the specified trust region, the user should re-evaluate the assumed extent of the region.

Results of invoking the optimization are placed in the structure "Results". Included in this structure is the input structure plus all relevant details of the optimization. This structure can be saved as a record of both what was assumed for data, what approach was used for optimization, and what resulted from the optimization including the optimized moduli, their uncertainties, velocity predictions, and deviations between data and predictions.

Workflow Example

Optimization begins by loading an input structure into the workspace:

```
[Input,Co,ea]=mkStrCPX('p');
```

where `Input` is the data structure, `Co` is an initial guess for elastic moduli, and `ea` is an initial set of Euler angles if data are taken in planes that are represented by rotation about an Euler axis. In the case of data characterized only by direction cosines, this variable is returned empty. `Co` can be moduli that represent *a priori* knowledge or can be set to default values. In this example, data from (Collins and Brown 1998) represent results for a monoclinic pyroxene mineral with 13 unique elastic moduli. The input string `'p'` results in `Co` being initialized to the published moduli. Any other input string will results in `Co` being set to default silicate moduli: longitudinal moduli (C_{11}, C_{22}, C_{33}) set to 100 GPa. Other moduli that are non-zero in the case of orthorhombic symmetry (C_{12}, C_{13}, C_{23}, C_{44}, C_{55}, C_{66}) are set to a nominal value of 50 GPa. The remaining uniquely monoclinic moduli are set to zero.

The following command returns results based on the published moduli:

```
[Cout,eaout,Results]=Velocities2Cij(Input,Co,'n',ea,'n','LM');
```

The ouput is:

```
rms misfit =20.8 m/s  chisqr =   1.01  elapsed time  0.0 s
```

The function `Velocities2Cij` has input variables `Input` (the data structure), `Co` (an initial guess for moduli), and `ea` (the euler angles for a data set characterized by the orientation of sample slices). The other items in the input list control the optimization process. The string following `Co` can be set to `'y'` to optimize moduli, `'n'` (do not optimize moduli), and `'r'` (optimization starting from randomly generated moduli that are uniformly distributed within the trust region for each moduli). The second string applies to the Euler angles and can also be set to `'y'`, `'n'` or `'r'` with the same meaning. When both strings are set to `'n'` the function returns values and statistics based in the input moduli and Euler angles. In the standard workflow with each invocation of the function, one either optimizes for moduli or for Euler angles. A simultaneous optimization for both moduli and Euler angles is not implemented. The third string defines the optimizer used: 'NM' for Nelder-Mead, 'LM' for Levenberg-Marquardt, and 'BG' for Backus-Gilbert. A last (optional) input variable, when set to zero, suppresses all command line output during execution of the function.

The function returns `Cout` (optimized moduli), `eaout` (optimized euler angles), and `Results` (the structure containing all information about data used in the optimization, the resulting optimized values, and associated statistics. Saving `Results` preserves all information related to the optimization effort. The structure `Results` is also used as input to the visualization functions.

The command line output gives the *rms* (root-mean-square) misfit (a common figure of merit) and chisqr (the reduced *chi-square* χ^2 which is the sum of the square of misfits weighted by uncertainty and normalized by the number of data). That χ^2 is close to one is appropriate if uncertainty has been correctly characterized and data errors are random and the optimization has found an appropriate solution.

A test of optimization can be undertaken by setting the first string flag to `'r'`, thus initiating optimization from a random (within the trust region) set of moduli:

```
>> [Cf,eaout,Results,Ct]=Velocities2Cij(Input,Cout,'r',ea,'n','LM',1);
      iteration    chisqr      optimality       lambda      relaxation
          0       5690.679      1.747e+02      1.000e-02     1.000e+00
          1        666.197      7.542e+00      1.000e-03     1.250e+00
          2        132.162      4.041e+00      1.000e-04     1.562e+00
          3         96.509      3.694e-01      1.000e-05     1.953e+00
          4         88.670      8.841e-02      1.000e-05     1.953e+00
          5         33.515      1.646e+00      1.000e-06     2.441e+00
         12         33.515      2.984e+04      1.000e-02     1.000e+00
         13          2.907      1.053e+01      1.000e-03     1.250e+00
         14          1.205      1.412e+00      1.000e-04     1.562e+00
         15          1.048      1.499e-01      1.000e-05     1.953e+00
         16          1.048      8.459e-05      1.000e-06     2.441e+00
         22          1.048      9.541e+05      1.000e-02     1.000e+00
         23          0.990      5.842e-02      1.000e-03     1.250e+00
         24          0.990      4.476e-05      1.000e-04     1.562e+00
         28          0.990      1.010e+06      1.000e-02     1.000e+00
         29          0.990      2.203e-06      1.000e-03     1.250e+00
      rms misfit =20.6 m/s  chisqr =   0.99  elapsed time  0.6 s
```

The first column shows the number of iterations during optimization. Iteration steps that do not improve misfit are not shown. The second column gives the current *chi-square* misfit. "Optimality" is the fractional change in misfit from step to step and is used as the convergence criteria. The fourth column gives current values of λ (the Levenberg-Marquardt parameter). It is adjusted by an order of magnitude up or down depending on the success of the current step in reducing misfit. The fifth column (relaxation) shows the current value of an extra parameter that multiplies the estimated increment of parameters (δm in equation 7). A properly chosen schedule of relaxation adjustment provides both faster convergence and an ability to avoid incrementing parameters into unphysical (not positive definite) regimes during optimization. The schedules for both λ and relaxation were adjusted on the basis of tests of several data sets.

In this example, the randomly generated starting model was clearly far from the optimal solution. Fourteen steps were required to approach a χ^2 near 1. The fitter struggled between step 5 and 12. Here the schedule for decreasing λ from the initial steepest descent approach was too rapid and Gauss-Newton linearization could not find the minimum. As a result of not improving misfit, λ was increased by 4 orders of magnitude between steps 5 and 12. The value of χ^2 then improved in a single step from 33.5 to 3. That iterations 15 through 29 show nearly the same misfit is a consequence of the chosen convergence criteria and possible problems associated with the linearization used to derive equation 7. Non-linearity (second derivatives of the model with respect to parameters) that is not accurately accounted for in equation 7 can cause gradient-based methods to struggle in locating the true minimum in misfit.

The optional output variable `Ct` shown above returns the values of the randomly generated starting moduli. The same initial guess is therefore available in the workspace to test other optimization method. Setting the input moduli to Ct and changing the optimization flag to 'y', and the method string to 'BG' invokes the Backus-Gilbert regularization:

```
>> [Cf,eaout,Results,Ct]=Velocities2Cij(Input,Ct,'y',ea,'n','BG',1);
iteration    chisqr      optimality      variance     relaxation
    0       5690.679      1.747e+02       2.037e+00     3.000e-01
    1       1812.880      2.139e+00       1.044e+00     3.750e-01
    2        781.117      1.321e+00       5.573e-01     4.688e-01
    3        417.900      8.691e-01       3.042e-01     5.859e-01
    4        265.884      5.717e-01       1.774e-01     7.324e-01
    5        177.133      5.010e-01       1.034e-01     9.155e-01
    6        110.901      5.972e-01       5.457e-02     1.144e+00
    7         85.868      2.915e-01       3.453e-02     1.431e+00
    8         79.864      7.518e-02       2.967e-02     1.788e+00
   18         79.864      1.252e+04       2.967e-02     3.000e-01
   19         75.886      5.243e-02       2.823e-02     3.750e-01
   20         74.819      1.425e-02       2.742e-02     4.688e-01
   21         74.062      1.023e-02       2.670e-02     5.859e-01
   22         72.973      1.493e-02       2.594e-02     7.324e-01
   23         71.377      2.235e-02       2.507e-02     9.155e-01
   24         69.178      3.179e-02       2.408e-02     1.144e+00
   25         66.436      4.127e-02       2.299e-02     1.431e+00
   26         62.410      6.452e-02       2.159e-02     1.788e+00
   27         21.170      1.948e+00       7.750e-03     2.235e+00
   38         21.170      4.724e+04       7.750e-03     3.000e-01
   39         12.881      6.435e-01       5.206e-03     3.750e-01
   40          7.389      7.433e-01       3.132e-03     4.688e-01
   41          3.821      9.336e-01       1.644e-03     5.859e-01
   42          1.760      1.172e+00       7.534e-04     7.324e-01
   43          1.102      5.973e-01       4.654e-04     9.155e-01
   44          1.002      9.935e-02       4.261e-04     1.144e+00
   45          0.997      5.544e-03       4.251e-04     1.431e+00
   54          0.997      1.003e+06       4.251e-04     3.000e-01
rms misfit =20.5 m/s  chisqr =   1.00  elapsed time  1.5 s
```

Following the Weidner and Carleton implementation, the Backus-Gilbert inversion optimizes the *rms* misfit rather than the *chi-square* misfit. As a result, here χ^2 is slightly larger and the `rms` misfit is slightly smaller. The schedule for modifying the relaxation parameter was tuned to provide the best performance for these test cases.

Although the Backus-Gilbert method converged, the total number of iteration steps and the elapsed time are larger than for Levenberg-Marquardt. In all test cases, Backus-Gilbert shows less "skill" in optimizing moduli. More often than when using Levenberg-Marquardt, it stalls at unacceptable misfit. In such cases, restarting the optimization from different starting points has allowed successful optimization. As observed for the Levenberg-Marquardt method, the optimizer can struggle near the minimum in misfit (here 10 iteration steps were taken at nearly the same level of misfit).

The Nelder-Mead optimization is invoked with the same randomized initial model:

```
>> [Cf,eaout,Results,Ct]=Velocities2Cij(Input,Ct,'t',ea,'n','NM',1);
      iteration      chisqr
          0         5690.68
        700          127.95
       1400            2.01
       2100            0.99
       2616            0.99
      rms misfit =20.6 m/s  chisqr =   0.99  elapsed time  7.5 s
```

The elapsed time is greater and the misfit surface has been sampled at more locations – over 2600 distinctly different sets of model parameters (a new set for each iteration step) were examined. The current implementation of the simplex method is provided within the standard MATLAB environment. An independent implementation based on widely available source code (*e.g.* Press et al 2007) might provide an opportunity to better “tune” the algorithm for better performance in this application by making use of the trust region to scale increments of the parameters. Since Nelder-Mead does not calculate numerical gradients, it does not suffer linearization problems near the minimum in misfit. In some test cases, the best moduli found by gradient methods can be slightly improved through further optimization using the Nelder-Mead algorithm.

The ability to optimize Euler angles is often necessary since several mechanical steps may separate an x-ray alignment of a crystal with its placement in an experiment to determine acoustic velocities. As noted by Every (1980), the three angles necessary to describe an orientation in laboratory coordinates are simply additional parameters to optimize. In many cases, the acoustic data constrain the orientations better than do direct measurements of orientation. Here a test is performed to explore the ability of velocity data sets to constrain the Euler angles. Below, the orientations of Euler angles are intentionally randomized with a variance of 4 degrees:

```
>> ea
      ea =
          7.9000  269.0000  345.2000
         89.6000   85.4000    7.3000
          3.2000  193.4000  345.5000
>> ear=ea+4*randn(3,3)
      ear =
          3.2023  276.1789  343.8573
         84.2397   83.4797    7.1015
          5.0633  193.0063  345.0603
```

These Euler angles with synthetic “experimental error” are then optimized against the velocity data:

```
>> [Cf,eaout,Results,Ct]=Velocities2Cij(Input,Cout,'n',ear,'y','LM',1);
```

```
      rms misfit =20.9 m/s  chisqr =   1.01  elapsed time  0.1 s
>> eaout
      eaout =
          7.8206  269.1834  345.5780
         89.6646   85.3551    7.2877
          3.1631  193.3522  345.1303
```

The recovered Euler angles are nearly equal to the initial values. When both optimized moduli and Euler angles are required, experience has shown that even with completely unknown Euler angles, a process of alternation between fitting for moduli and fitting for Euler angles converges to the correct results. Implementation of a simultaneous optimization for both moduli and Euler angles is possible but has not yet been necessary.

Visualization of predictions versus data is accomplished through the use of plotting functions BWPlot (for body wave data) and SWPlot (for surface wave data). Both are invoked with the same input parameters. BWPlot is demonstrated here:

```
BWPlot(Results,plt_win,pltprcnt)
```

Where `Results` is the output structure, `plt_win` is a user specified frame number where the figure is shown, and `pltprcnt` is the percentage range for display of deviations between data and predictions. The resulting plot is shown as figure 2.

Examples

Coesite: (file `mkStrCoesite)` The pioneering data set of Weidner and Carleton (1977) is provided here. Coesite is monoclinic and thus requires 13 elastic moduli. Measurements were made in 96 directions. Not all polarizations were observed in any one direction. Six of the data deviated so strongly from their best fit that even though listed in the table these points were excluded from the fit. They reported an *rms* misfit of 151 m/s. This misfit is approximately an order of magnitude larger than current experimental standards.

Direction cosines and observed velocities were copied directly from the paper into the example file `mkStrCoesite.m`. Experimental uncertainties (180 m/s for shear waves and 130 m/s for compressional waves) were assigned based on the observed misfit reported in the paper. Examination of the data indicates that most of the direction cosines lie on planes. Thus, a set of Euler angles could in principle be used to describe the propagation directions. Two initial sets of moduli are provided. In the function call to `mkStrCoesite`, setting `Cflg` to `'p'` returns the published moduli in the variable `Co`. Any other string or no input arguments returns a default silicate set of moduli. The commands below demonstrate loading the data, checking that the published results are duplicated, and then attempting further optimize using both Levenberg-Marqardt and Backus Gilbert methods. The last command determines moduli uncertainties on the basis of a Monte Carlo test. Results from these commands which are saved in the `Results` structure are summarized in Table 1.

```
>> [Input,Cout,ea]=mkStr_Coesite('p');
>> [Cf,eaout,Results]=Velocities2Cij(Input,Cout,'n',ea,'n','LM',1);
      rms misfit =151.6 m/s  chisqr =   1.01  elapsed time  0.0 s
>> [Cf,eaout,Results]=Velocities2Cij(Input,Cout,'y',ea,'n','LM',1);
      iteration    chisqr      optimality       lambda      relaxation
         0          1.006      9.940e+05       1.000e-02     1.000e+00
         1          0.995      1.130e-02       1.000e-03     1.250e+00
         2          0.995      1.038e-05       1.000e-04     1.562e+00
         6          0.995      1.005e+06       1.000e-02     1.000e+00
         7          0.995      1.071e-06       1.000e-03     1.250e+00
      rms misfit =151.1 m/s  chisqr =   0.99  elapsed time  0.2 s

>> [Cf,eaout,Results]=Velocities2Cij(Input,Cout,'y',ea,'n','BG',1);
      iteration    chisqr      optimality      variance     relaxation
         0          1.006      9.940e+05       2.312e-02     3.000e-01
         1          1.001      4.589e-03       2.301e-02     3.750e-01
         2          1.000      1.551e-03       2.295e-02     4.688e-01
         3          0.999      5.934e-04       2.292e-02     5.859e-01
         4          0.999      2.244e-04       2.291e-02     7.324e-01
         5          0.999      2.976e-05       2.290e-02     9.155e-01
        12          0.999      1.001e+06       2.290e-02     3.000e-01
      rms misfit =150.8 m/s  chisqr =   1.00  elapsed time  0.4 s
>> [uncerts,Cfs,rms]=MonteCarloStats(Input,1000,Cout,1,0);
```

As previously noted Levenberg-Marquardt and Backus-Gilbert methods converge to slightly different *rms* and χ^2 misfits. Differences between the published moduli and moduli determined here are small relative to uncertainty. A general observation is that too many significant figures were reported in the original paper and in the current table. Some parameters are uncertain by more than their value. The previously reported uncertainties agree with the uncertainties estimated here. The first two uncertainty columns are calculated from the covariance matrix (based on numerical derivatives). Differences are expected for such approximate finite difference calculations. The last column gives the Monte Carlo estimates of uncertainties based on a thousand synthetic models that have the same distribution of propagation directions and the same distribution of (assumed to be random) misfits. These results are in adequate agreement with the covariance based estimations. That the current calculations for velocities agree with model velocities reported in the original paper and that all uncertainty estimates are in adequate agreement provides support that the numerical framework used here is internally consistent and reliable.

Clinopyroxene: (`mkStrCPX`) Collins and Brown (1998) reported velocities measured using Impulsive Stimulated Light Scattering on three slices of a mantle-derived clinopyroxene. The current analysis (discussed above) duplicates published results as shown in Table 2.

Glaucophane: (`mkStrGlaucophane`) Bezacier et al (2010) reported velocities and moduli for this important monoclinic mineral. Although only direction cosines are given in the paper, Euler angles for three separate rotations about their crystals were determined (the cross product of any two directions in a plane define the normal direction). The file `[Input,Cout,ea]=mkStrGlaucophane(Cflg)` returns the published moduli in `Cout` if `Cflg ='p'`; In the command line, if `Input.Data.dcosflg` is set to 1, only published direction cosines are used in the analysis. If `Input.Data.dcosflg` is set to 0, Euler angles are used. In this second case, it is possible to optimize the Euler angles.

Undertaking Backus Gilbert optimization from default moduli (far from the published moduli) recovers the published *rms* misfit and moduli (Table 3).

```
>> [Cf,eaout,Results]=Velocities2Cij(Input,Cout,'y',ea,'n','BG',1);
   iteration    chisqr      optimality      variance     relaxation
       0       1989.274      5.017e+02       9.880e-01     3.000e-01
       1        455.036      3.372e+00       7.068e-01     3.750e-01
       2        283.879      6.029e-01       4.043e-01     4.688e-01
       3        183.428      5.476e-01       2.199e-01     5.859e-01
       4        111.689      6.423e-01       1.241e-01     7.324e-01
       5         61.102      8.279e-01       6.702e-02     9.155e-01
       6         17.393      2.513e+00       1.777e-02     1.144e+00
       7          4.451      2.908e+00       4.363e-03     1.431e+00
       8          2.419      8.401e-01       2.380e-03     1.788e+00
       9          2.056      1.763e-01       2.042e-03     2.235e+00
      20          2.056      4.863e+05       2.042e-03     3.000e-01
      21          2.009      2.333e-02       2.009e-03     3.750e-01
      22          1.978      1.568e-02       1.991e-03     4.688e-01
      23          1.959      9.689e-03       1.983e-03     5.859e-01
      24          1.949      5.465e-03       1.980e-03     7.324e-01
      25          1.944      2.694e-03       1.979e-03     9.155e-01
      26          1.941      1.081e-03       1.978e-03     1.144e+00
      27          1.941      3.501e-04       1.978e-03     1.431e+00
      28          1.941      9.756e-05       1.978e-03     1.788e+00
      38          1.941      5.153e+05       1.978e-03     3.000e-01
      39          1.941      8.605e-07       1.978e-03     3.750e-01
      rms misfit =44.3 m/s  chisqr =   1.09  elapsed time  0.6 s
```

However, uncertainties given in the original paper and listed in Table 3 are not in agreement with the current covariance-based estimate, the Monte Carlo based estimate, or examination of the data. On the basis of the distribution of propagation directions and data scatter, the reported uncertainties for several moduli (C_{15}, C_{25} C_{35}, C_{46}) appear too small while others (for example, C_{22} and C_{33} relative to C_{11}) are too large.

If Euler angles are optimized, the *rms* misfit of this data set can be further reduced by 17%. The change in Euler angles of a few degrees for all slices provides a hint that a systematic experimental difference might exist between the orientations determined by x-ray and orientations assigned for propagation directions.

Monoclinic Potassium Feldspar: (`mkStrKspar`) Surface acoustic waves have been measured using Impulsive Stimulated Light Scattering. Results for potassium feldspars are given in Waeselmann et al (2014). Here synthetic velocities are calculated for propagation directions achieved in laboratory experiments. The model moduli are rounded to whole numbers for easier evaluation of inverse results. Random variance is added to the calculated velocities to create synthetic data with scatter that is comparable to that observed in experiments (around 10 m/s). The advantage of this synthetic data set is that the underlying model (both moduli and Euler angles) are known. The inverse process is checked to confirm that it correctly recovers the model parameters.

Surface wave inversions are not as robust as those for body waves. In the absence of additional constraints (*e.g.* high pressure x-ray determinations of axis compressibility) the longitudinal moduli (C_{11}, C_{22}, C_{33}) covary strongly with the off-axis longitudinal moduli (C_{12}, C_{13}, C_{23}). Even with side constraints, the inversions can stall at unacceptable levels of misfit. However, a multi-start approach (*i.e.* restarting optimization from another random initial model) has proven effective in locating optimal solutions. In the example given below, the optimization was initiated several times in order to find one set of initial guesses that converged. If the boundaries of the trust region are reduced based on *a priori* knowledge (*e.g.* providing bounds for moduli based on properties of similar minerals), the percentage of successful inversions from random starting models increases. Shown below is the convergence path for the synthetic feldspar data with a side constraint based on the axes compressibilities.

```
>> [Cf,eaout,Results,Ct]=Velocities2Cij(Input,Cout,'r',ea,'n','LM',1);
      iteration    chisqr      optimality       lambda      relaxation
         0       62883.136      1.490e+01       1.000e-02     1.000e+00
         1       11487.109      4.474e+00       1.000e-03     1.250e+00
         2         544.873     2.008e+01       1.000e-04     1.562e+00
         3         327.501     6.637e-01       1.000e-05     1.953e+00
         4          26.277     1.146e+01       1.000e-03     1.250e+00
         5           2.929     7.971e+00       1.000e-04     1.562e+00
         6           1.630     7.969e-01       1.000e-05     1.953e+00
         7           1.621     5.469e-03       1.000e-06     2.441e+00
        16           1.621     6.168e+05       1.000e-02     1.000e+00
        17           1.178     3.766e-01       1.000e-03     1.250e+00
        18           1.163     1.280e-02       1.000e-04     1.562e+00
        19           1.163     2.315e-04       1.000e-03     1.250e+00
        20           1.162     6.991e-05       1.000e-04     1.562e+00
        26           1.162     8.603e+05       1.000e-02     1.000e+00
        27           1.162     8.111e-06       1.000e-03     1.250e+00
      rms misfit =10.4 m/s  chisqr =   1.16  elapsed time 15.6 s
```

The total number of steps to solution is similar to those shown for body wave examples. However, the numerical approach necessary for SAW and PSAW analysis requires many more calculations and the elapsed time is an order of magnitude greater. Table 4 lists the input moduli and moduli resulting from this inversion. Covariance and Monte Carlo based uncertainty estimates are also listed. The moduli recovered through the inverse process adequately agree with the moduli used to create the synthetic data. Extensive testing has shown that this is generally the case as is further supported by the observation that Monte Carlo uncertainty estimates typically agree with covariance-based estimates.

Hornblende: (`mkStrHornblende`) In this example a mixed set of body wave and surface wave data is provided. The measured velocities are based on Impulsive

Stimulated Light Scattering experiments. The number of measurements of transverse body waves was inadequate to provide a robust solution for the elastic moduli solely on the basis of body wave data. Thus, additional surface wave measurements were undertaken. The combination of measured compressional velocities that are strongly dependent on the longitudinal moduli and surface waves velocities that are strongly dependent on other off-diagonal moduli provides a robust dataset. The data are loaded with the command:

```
[Input, Co,ea]=mkStrHornblende('p')
```

Inverse results are shown in Table 5. Uncertainties based separately on body waves, surface wave and for the joint fit are shown. The large uncertainties based only on surface waves reflect strong covariance between moduli rather than any intrinsic error. The complementary contributions in the combined data set create a final set of moduli with significantly reduced uncertainty. Here all moduli for this low symmetry crystal have 2σ uncertainties less than 1 GPa.

Summary

Functions are implemented in the MATLAB numerical environment that allow flexible analysis of acoustic wave velocities to determine elastic moduli. The package will run under all standard operating systems and hardware if MATLAB is available. In the case of surface wave analysis, two FORTRAN source files must be compiled and linked to MATLAB as MEX-files. Several inverse methods are provided since no one method and no single optimization attempt will always find the optimal solution. Example data sets are provided. These allow a user to gain experience in finding optimal moduli and provide templates to organize new data in need of interpretation.

The methods are tested using both published and synthetic data sets. The Levenberg-Marquardt method shows greater skill and speed in finding optimal solutions relative to the Backus-Gilbert inverse technique. Although the Nelder-Mead simplex method is slower, in some cases it can find a slightly better solution since the linearization inherent in the gradient-based methods fails if second derivatives of the model with respect to parameters are inadequately represented.

Uncertainties based on the diagonal of the covariance matrix and those estimated using Monte Carlo simulations are generally in accord and agree with most published estimations. The current package of functions therefore provides a robust, validated, and flexible environment for analysis of ultrasonic, Brillouin, or Impulsive Stimulated Light Scattering datasets.

Acknowledgements: Partial support was provided by the National Science Foundation through grant EAR-0711591. Permission from A. Every to include portions of his source code in the current application is acknowledged and appreciated. Discussions with K. Creager and A. Abramson have helped refine this paper.

Appendix

Description of functions called by Velocities2Cij

These functions are nested within the main function `Velocities2Cij` so that necessary variables are globally available and do not need to be passed in function calls.

Functions that accomplish the optimization include:

```
[Co,misfit,~,output]=fminsearch(func,Co,options);
```

This Nelder-Mead optimization function is built into MATLAB. Inputs include `func` (a string defining the function to be called that returns misfit). `Co` is the starting set of moduli. A list of user-controlled options can be found in MATLAB documentation.

```
[Cout,chisqr]=LM_LSQR(Cin)
[Cout,chisqr]=BackusGilbert(Cin)
[eaout,chisqr]=LM_LSQR_ea(eain,ix,lb,ub)
```

These invoke the Levenberg-Marquardt or Backus-Gilbert optimization with obvious input and output variables. `LM_LSQR_ea` uses the Levenberg-Marquardt method to optimize Euler angles for a single round of data (as defined by the input ix). lb and ub are vectors containing upper and lower bounds for the Euler angles.

Three functions calculate misfit and Jacobians for (1) body wave data, (2) surface wave data, or (3) data sets including both body and surface wave data.

```
[chisqr,J,dvbw,rms,npflg] = BW_calc(Co)
[chisqr,J,dvsw,rms,npflg] = SW_calc(Co)
[chisqr,J,dvbwsw,rms,npflg] = EC_calc(Co)
```

where `Co` is the current set of moduli being adjusted. Output by the functions are the reduced `chisqr`, `J` (the Jacobian), `dv` (a list of deviations between data and the model), `rms` (the root-mean-square misfit of the current model), and `npflg` (which is set equal to 1 if the current elastic moduli are not positive definite). The numerical derivatives are evaluated as single sided finite differences with a fixed increment of the independent variable. More computationally intensive (and presumably more accurate) methods to evaluate derivatives (double sided and adaptive increments) were evaluated and did not demonstrably improve performance or significantly change results.

The following functions are not nested within `Velocities2Cij`.

```
[veldat,sigdat,dcos,idfnt]=Data2matrixBW(InptStrct,ifit)
[veldat,sigdat,dcos,comp,dcomp]=Data2matrixSW(InptStrct,ifit)
```

These functions unpack selected data (controlled by `ifit`) from the input structure and return appropriate vectors or matrixes. `ifit` is a vector defining which data in a set including more than one round of velocities, are to be used. Body wave velocities sets include up to three velocities for each propagation direction. Since in practice not all three phases are observed, the missing data are listed in the data structure as NaN (not a number) and `idfnt` is a vector of indexes into the velocity matrix giving locations of velocities that are not NaN. `comp` and `dcomp` are vectors of x-ray determined compliances and their uncertainties.

```
Cout=Ci2Cij(Cin,sym)
```

This function symmetrically coverts between vector and matrix representations of elastic moduli. (vector in –> matrix out or matrix in –> vector out). The input sym is a string declaring the symmetry associated with the moduli.

```
cout=Tnsr2Mtrx(cin)
```

This function symmetrically converts between tensor and matrix representations of the elastic moduli (matrix in -> tensor out or tensor in –> matrix out).

```
cout=rotateCij(cin,atr)
```

This function rotates the coordinate system for a set of elastic moduli. Input either matrix or tensor moduli and a rotation matrix atr. The output moduli, cout, are in the same representation (vector or matrix) as the input.

```
[J, velc]=jacobianSW(Co,iconst,Input,Cflg)
[J, velc]=jacobianBW(Co,iconst,sym,dcos,idfnt,rho,Cflg)
```

These functions return the Jacobians (`J`) and model velocities (`velc`) given elastic moduli, a list of which moduli are allowed to vary (`iconst`), the input data structure, and a flag (`Cflg`) to determine whether derivatives are with respect to moduli or compliances.

```
[chisqr,J,dv,sigdat]=jacobian_ea(InputStrct,ix,Co)
```

This function is designed to return the Jacobian for the euler angles of a particular sample (defined by index `ix`).

```
ea = inv_eiler(OM)
OM = eiler(ea);
```

These functions convert between Euler angle (ea) and the orientation matrix (OM) representations of a crystal coordinates relative to laboratory coordinates.

```
dcos=angles2dcos(a,ea)
```

This function takes laboratory rotational angles of a sample (`a`) and the associated Euler angles for that sample (`ea`) and calculates the direction cosines at each angle.

```
out=KG_calc(C,Mc,Ms,sym)
```

This function determines isotropic Voigt-Reuss moduli and their uncertainties for the moduli `C` and covariance matrixes for moduli (`Mc`) and compliances (`Ms`) for symmetry sym.

```
c=Crand(TrustRegion)
```

Given a matrix defining the trust region for elastic moduli (lower and upper bounds), this function provides a random set of moduli (uniformly distributed over the range for each modulus). It checks that the moduli are positive definite.

```
[velocities,eigvec] = xstl(dcos,rho,C)
```

This function calculates velocities and polarizations of body waves with propagation directions given by dcos, density rho, and moduli matrix C. The output values for each direction of propagation are sorted by velocity.

```
SWout=SurfaceWaveVel(Input,Co,SWflg)
```

This function is gateway to calculations of surface wave velocities. `Input` is the standard input data structure (which contains information needed for the surface wave calculations. `Co` are the moduli, and `SWflg` is set to 'v' to return velocities for specified propagation directions or 's' to calculate a grid of surface wave excitation intensities as a function of velocity and direction. The output structure contains different results depending on the input flag. This function calls mex files (compiled FORTRAN). The source code is based on "PANGIM" (Every 1998). "modevel.F" was modified from "PANGIM" to return the velocity associated with peaks in the intensity spectra. "modeconv.F" returns spectral intensities on a grid of velocities and directions of propagation.

```
[uncert,Cfs,rms]=MonteCarloStats(Input,nsyn,Co,0);
```

This function creates a number `nsyn` of random velocity data sets (each with the same propagation directions and experimental variance as data described in `Input`). Each synthetic data set is inverted for the optimized moduli that are returned in matrix Cfs (size is nsyn by the number of moduli). The rms misfit is returned in vector rms and twice the standard deviation of each modulus is returned in uncert.

```
BWPlot(Results,ifig,pltdev)
SWPlot(Results,ifig,pltdev)
```

These functions plot results for body waves (if data for individual samples lie in planes defined by euler angles) and surface waves. The number of subplots is adjusted depending on how many samples are in the data set. ifig sets the window to plot in. pltdev is the range in percent for the deviations plot.

References

Auld, B. A. (1973) *Acoustic Fields and Waves in Solids Vol. 1*, Wiley, New York.

Abramson, E. H., L. J. Slutsky, and J. M. Brown, Elastic constants, interatomic forces and equation of state of β-oxygen at high pressure, *J. Chem, Phys.,* 100, 4518-4526, 1994.

Abramson, E. H., J. M. Brown, L. J. Slutsky, and J. Zaug (1997) The elastic constants of San Carlos olivine to 17GPa, *J. Geophys. Res.,* 102, 12,253-12,263.

Abramson, E. H., J. M. Brown, and L. J. Slutsky (1999) Applications of impulsive stimulated scattering in the earth and planetary sciences, *Ann. Rev. Phys. Chem.,* 50, 279-313, 1999

Aleksandrov, K.S., U.V. Alchikov, B.P. Belikov, B.I. Zalavskii, A.I. Krupnyi (1974) Velocities of elastic waves in minerals at atmospheric pressure and increasing precision of elastic constants by means of EVM, *Izv. Acad. Sci. USSR Geol. Ser*., 10, 15–24

Aster, R., B. Borchers, and C. Thurber, *Parameter estimation and inverse problems*, Academic Press, 2012.

Backus, G.E., and J. F. Gilbert (1968), The resolving power of gross Earth data, *Geophys. J. R. Astro. Soc.,* 16, 169–205.

Backus, G. E. and J. F. Gilbert (1970) Uniqueness in the inversion of gross earth data, *Phil. Trans. R. Soc. Land.,* 266, 123-192.

Bezacier, L., B. Reynard, J. D. Bass, J. Wang , D. Mainprice (2010) Elasticity of glaucophane, seismic velocities and anisotropy of the subducted oceanic crust, *Tectonophysics*, 494, 201–210

Brown, J. M., L. J. Slutsky, K. A. Nelson, and L-T. Cheng (1989) Single crystal elastic constants for San Carlos Peridot: An application of impulsive stimulated scattering, *J. Geophys. Res., 94,* 9485-9492.

Brown, J. M., E.H. Abramson, R. L. Ross (2006) Triclinic elastic constants for low albite, *Phys. Chem. Minerals,* DOI 10.1007/s00269-006-0074-1.

Brown, J. M., R. Angel, N. Ross (2014) Elasticity of plagioclase feldspars, *J. Geophys, Res.,* in review

Brown, J. M., (2014), Elasticity of common amphiboles, in preparation

Chai, M., J. M. Brown, and L. J. Slutsky (1997) The elastic constants of a pyrope-grossular-almandine garnet to 20 GPa, *Geophys. Res. Lett*, 24, 523-526.

Collins, M. C., and J. M. Brown (1998) Elasticity of an upper mantle clinopyroxene, *Phys. Chem. Minerals*, 26, 7-13.

Crowhurst, J. C., E. H. Abramson, L. J. Slutsky, J. M. Brown, J. M. Zaug, and M. D. Harrell (2001) Surface acoustic waves in the diamond anvil cell: An application of impulsive stimulated light scattering, *Phys. Rev. B*, 64, 100103-6.

Crowhurst J. C, J. M. Zaug (2004) Surface acoustic waves in germanium single crystals, *Phys. Rev. B*, 69, 52301-1-4.

Every, A. G. (1980) General closed-form expressions for acoustic waves in elastically anisotropic solids, *Phys. Rev. B*, 22, 1746-1760.

Every A. G., Kim K. Y. , A. A. Maznev (1998) Surface dynamic response functions of anisotropic solids, *Ultrasonics* ,36, 349–353.

Farnell, G. W. (1970) Properties of elastic surface waves, In: Mason W.P., Thurston R.N. (eds) *Physical Acoustics*, Academic, New York, pp 109–166

Gallagher, K. and M. Sambridge (1994) Genetic algorithms: a powerful tool for large-scale nonlinear optimization problems, *Computers & Geosciences,* 20, 1229-1236.

Kirkpatrick, S., C. D. Gelatt, M. P. Vecchi (1983) Optimization by simulated Annealing, *Science*, 220, 671–680, doi:10.1126/science.220.4598.671.

Marquardt, D. (1963) An algorithm for least-squares estimation of nonlinear parameters, *SIAM Journal on Applied Mathematics* 11, 431–441, doi:10.1137/0111030.

Maznev A. A., A. Akthakul, K. A. Nelson (1999) Surface acoustic modes in thin films on anisotropic substrates*. J. Appl. Phys.,* 86, 2818–2824.

Nelder, J. A., and R. Mead (1965) A simplex method for function minimization, *Computer Journal* , 7, 308–313. doi:10.1093/comjnl/7.4.308.

Press, W. H., S. A. Teukolsky, W. T. Vetterling, B. P. Flannery (2007) *Numerical Recipes: The Art of Scientific Computing* (3rd ed.). New York: Cambridge University Press. ISBN 978-0-521-88068-8.

Waeselmann, N., J. M. Brown, R. Angel, N. Ross, (2014), Elasticity of alkaline feldspars in preparation

Weidner, D. J. and H.R. Carleton, (1977) Elasticity of coesite, *J. Geophys. Res.*, 82, 1334-1346.

	Weidner and Carleton	Current Backus-Gilbert	Current Levenberg-Marquardt	Weidner and Carleton 2σ	Covariance 2σ	Monte Carlo 2σ
C_{11}	160.8	160.8	161.3	5.8	4.1	4.8
C_{12}	82.1	81.6	80.5	8.4	7.6	6.0
C_{13}	102.9	102.5	103.1	12.2	10.7	10.3
C_{15}	-36.2	-36.0	-35.9	3.6	2.9	3.0
C_{22}	230.4	230.5	230.6	5.2	3.9	5.3
C_{23}	35.6	31.9	34.1	16.2	17.1	14.6
C_{25}	2.6	4.3	5.0	8.0	7.6	5.6
C_{33}	231.6	232.3	231.6	8.8	6.6	8.5
C_{35}	-39.3	-40.1	-39.9	4.8	3.9	4.6
C_{44}	67.8	67.3	67.8	6.0	6.8	4.4
C_{46}	9.9	9.6	9.4	4.0	3.8	2.5
C_{55}	73.3	73.3	73.2	4.6	4.3	2.9
C_{66}	58.8	58.5	58.1	3.6	3.3	2.2
Misfit (*rms*)	152	151	151			

Table 1. Elastic moduli and uncertainties for Coesite based on velocities reported by Weidner and Carleton 1977. Voigt notation moduli are listed in the first column. Published moduli are in the second column. Current results using the Backus-Gilbert and the Levenberg-Marquardt inverse techniques are listed in the next two columns. 2σ uncertainties are given in the last three columns based on published results, covariance estimates, and Monte Carlo estimates.

	Collins and Brown	Current Levenberg-Marquardt	Collins and Brown 2σ	Covariance 2σ	Monte Carlo 2σ
C_{11}	237.8	238.0	0.9	1.3	1.4
C_{12}	83.5	84.0	1.3	1.4	1.1
C_{13}	80.0	79.8	1.1	1.3	1.1
C_{15}	9.0	9.2	0.6	0.8	0.8
C_{22}	183.6	184.3	0.9	1.2	1.1
C_{23}	59.9	59.4	1.6	1.6	1.7
C_{25}	9.5	9.9	1.0	1.0	1.0
C_{33}	229.5	229.3	0.9	1.1	1.0
C_{35}	48.1	48.2	0.6	0.7	0.7
C_{44}	76.5	76.8	0.9	1.0	0.8
C_{46}	8.4	8.4	0.8	0.8	0.7
C_{55}	73.0	73.0	0.4	0.4	0.5
C_{66}	81.6	81.1	1.0	1.2	1.2
Misfit (*rms*)	20.8	20.6			

Table 2. Elastic moduli and uncertainties for clinopyroxene based on velocities reported by Collins and Brown (1998). Moduli in Voigt notation are listed in the first column. Published moduli are listed in the second column. Current results using the Levenberg-Marquardt inverse technique are listed in the next column. 2σ uncertainties are given in the last three columns based on published estimates, covariance estimates and Monte Carlo estimates.

	Bezacier et al	Current Backus-Gilbert	Current Levenberg-Marquardt	Bezacier et al 2σ	Covariance $2\ \sigma$	Monte Carlo $2\ \sigma$
C_{11}	122.3	122.2	121.5	1.9	1.8	1.4
C_{12}	45.7	45.6	44.6	1.1	2.3	2.0
C_{13}	37.2	37.2	37.4	1.0	2.7	2.4
C_{15}	2.3	2.4	2.7	0.1	1.1	1.0
C_{22}	231.5	231.5	229.7	4.8	2.8	2.9
C_{23}	74.9	74.9	75.8	2.0	2.8	2.9
C_{25}	-4.8	-4.7	-5.1	0.1	2.9	2.8
C_{33}	254.6	254.6	256.0	5.8	3.1	2.9
C_{35}	-23.7	-23.7	-24.1	0.3	1.6	1.5
C_{44}	79.6	79.7	79.4	0.9	1.1	1.0
C_{46}	8.9	8.9	9.2	0.1	1.1	0.9
C_{55}	52.8	52.8	53.0	0.5	0.8	0.7
C_{66}	51.2	51.2	51.3	0.4	0.7	0.7
Misfit (*rms*)	44.3	44.3	37.0			

Table 3. Elastic moduli and uncertainties for Glaucophane based on velocities reported by Bezacier et al (2010). Moduli in Voigt notation are listed in the first column. Published moduli are given in the second column. Current results using the Backus-Gilbert and the Levenberg-Marquardt inverse techniques are listed in the next two columns. Euler angles were also optimized for the Levenberg-Marquardt results. 2σ uncertainties are given in the last three columns - the published estimate, the current covariance estimate and the current Monte Carlo estimate.

	Model	Inverse	Covariance 2σ	Monte Carlo 2σ
C_{11}	85.0	84.9	0.2	0.1
C_{12}	50.0	50.0	0.5	0.4
C_{13}	40.0	40.1	0.6	0.4
C_{15}	-1.0	-0.9	0.1	0.1
C_{22}	160.0	162.9	3.5	1.8
C_{23}	20.0	17.4	2.9	1.5
C_{25}	-10.0	-10.5	0.6	0.4
C_{33}	165.0	166.9	2.6	1.6
C_{35}	10.0	10.3	0.6	0.4
C_{44}	20.0	20.0	0.1	0.1
C_{46}	-12.0	-11.9	0.1	0.1
C_{55}	20.0	20.1	0.2	0.1
C_{66}	30.0	29.8	0.2	0.2
Misfit (*rms*)	11.1	10.4		

Table 4. Elastic moduli and uncertainties for a synthetic alkaline feldspar based on surface wave velocity propagation directions in Waeselmann et al 2014. Moduli in Voigt notation are listed in the first column. Model moduli are given in the second column. Inverse results using the Levenberg-Marquardt inverse technique are listed in the next column. 2σ uncertainties are given in the last two columns based on covariance and Monte Carlo estimates.

	Hornblende	Covariance 2σ	Monte Carlo 2σ	Body wave 2σ	Surface wave 2σ
C_{11}	133.2	0.5	0.5	0.6	11.6
C_{12}	53.8	0.9	0.7	1.7	8.1
C_{13}	48.4	0.7	0.6	0.8	7.2
C_{15}	-1.0	0.3	0.3	0.3	2.8
C_{22}	189.3	0.7	0.6	0.8	15.1
C_{23}	61.2	0.8	0.8	1.4	10.1
C_{25}	-8.8	1.0	0.8	3.6	4.5
C_{33}	227.6	0.7	0.7	0.8	23.3
C_{35}	-31.1	0.4	0.4	0.4	4.2
C_{44}	73.7	0.4	0.4	0.7	1.6
C_{46}	4.3	0.4	0.4	1.7	0.7
C_{55}	47.2	0.2	0.2	0.3	1.2
C_{66}	48.5	0.2	0.2	1.3	0.5
Misfit (*rms*)	13.1				

Table 5. Elastic moduli and uncertainties for hornblende based on velocities reported by Brown 2014. Moduli in Voigt notation are listed in the first column. Moduli are given in the second column. 2σ uncertainties are given in the last three columns based on published results, results based on the covariance matrix and results based on the Monte Carlo method.

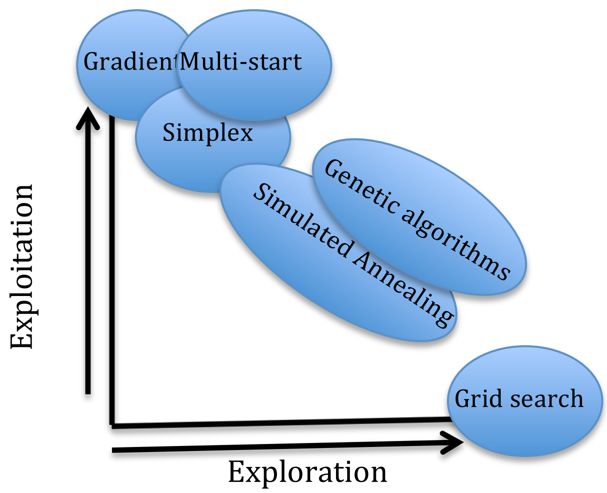

Figure 1. Schematic representation of inverse methods (adapted from a presentation by M. Sambridge). The vertical axis suggests the relative contribution of local gradients in determination of directions to move to improve model misfit. The horizontal axis suggests an increased number of evaluations of the forward problem. Inverse methods that rely on local gradients explore more limited regions of the parameter space (only that part of the space lying along a path from larger to smaller misfit) while a full grid search relies on massive sampling of the entire parameter space. The simplex method, while not directly calculating local gradients works to move "downhill". In multi-start methods, more regions of the parameter space are explored while still making use of local gradients. Both genetic algorithms and simulated annealing are less dependent on local gradients and rely more on extensive sampling of the parameter space.

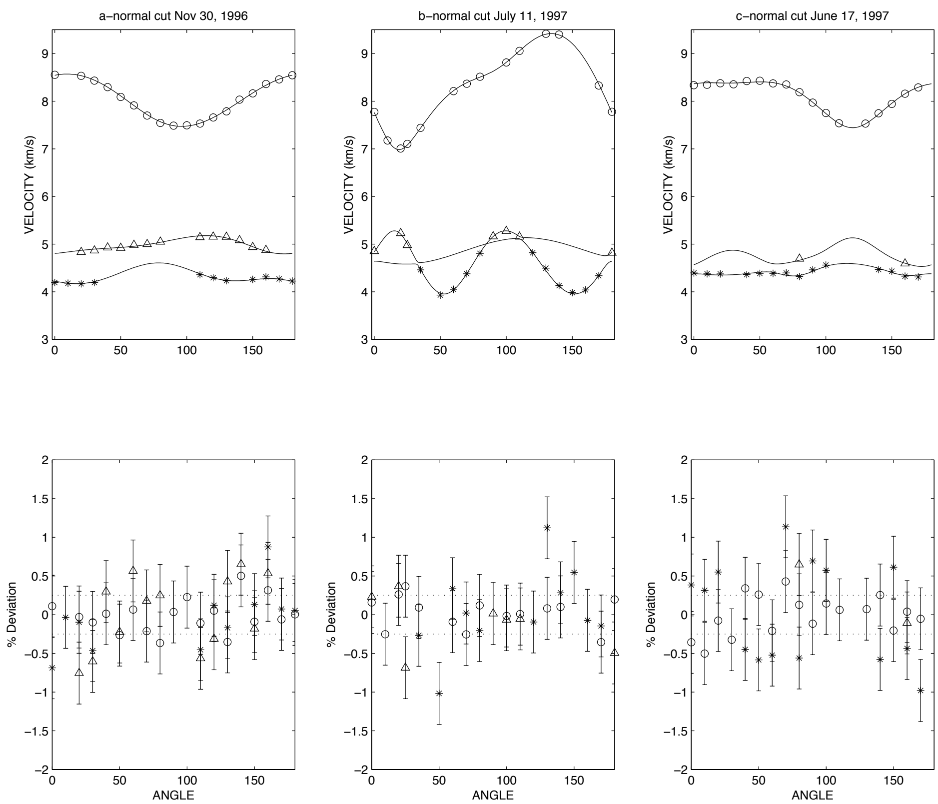

Figure 2. Model predictions, velocities and deviations between observations and predictions for clinopyroxene. These plots were generated using the MATLAB function BWPlot. Velocities were measured in planes perpendicular to three crystallographic directions (normal to a*, b, and c). The upper panels show measured velocities and model predictions. The lower panels show percentage deviations of data from predictions. For reference dashed lines at +/- 0.3% are shown.

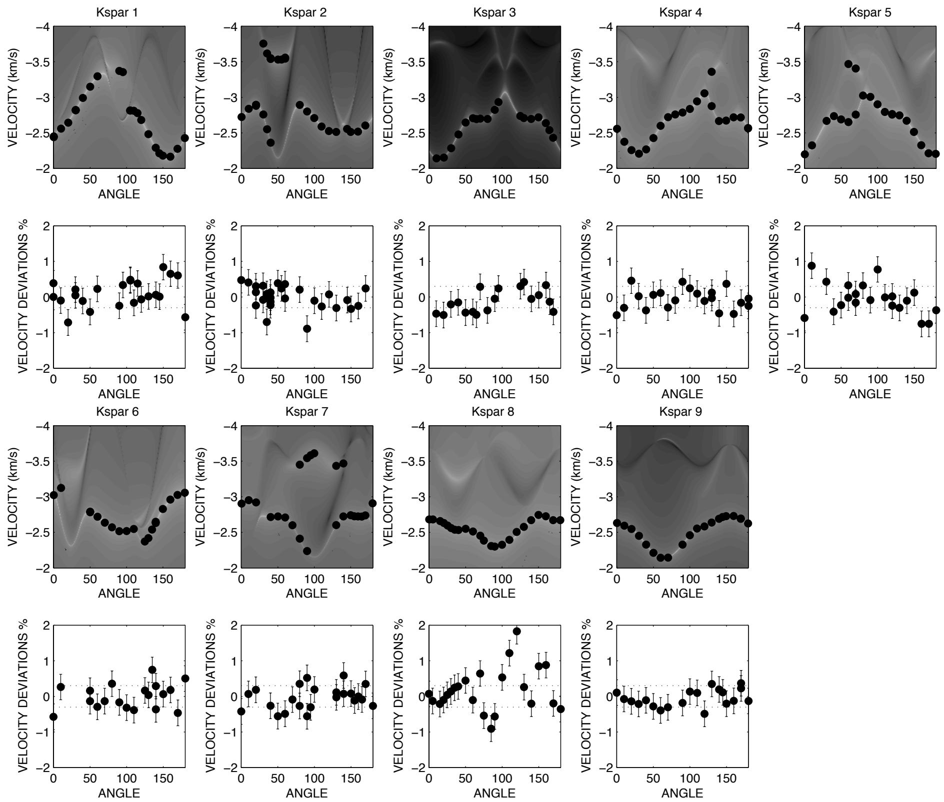

Figure 3. Model predictions, velocities and deviations between observations and predictions for synthetic alkaline feldspar. These plots were generated with the MATLAB function SWPlot. The upper panels show measured SAW and PSAW velocities as filled circles. The log of the elastic Green's function tensor element G_{13} is shown in the gray scale. Lighter means greater phase amplitude. Below each velocity panel is a plot of percentage deviations of data from predictions. For reference, dashed lines at +/- 0.3% are shown.