InvertTrishear

v. 1.0

by David Oakley © 2015

License and Disclaimer

This software (InvertTrishear, Setup_InvertTrishear, and associated Matlab scripts) is licensed for non-commercial use only. Commercial users please contact me (David Oakley). Use of this software in any academic or other publications should be acknowledged. Users of the software may transfer copies of the software and users' manual to other non-commercial users, but may not sell it or offer it as an inducement to buy any product.

The software is provided as is with no warrantee, explicit or implicit and without guarantee of fitness for any particular purpose. Not all aspects of the software have been thoroughly tested, and some bugs are likely to be present. The author will not be liable for any damages caused by defects in this software or in the users' manual.

Table of Contents

1. Introduction

InvertTrishear is a program for fitting trishear fault propagation fold models to data. Several different types of data, fault geometry, and inversion methods can be used (see section 2). The program uses the trishear velocity field equations of Zehnder and Allmendinger (2000) and the rate of change of dip equation of Oakley and Fisher (in press, "Inverse trishear modeling of bedding dip data using Markov Chain Monte Carlo methods"). The method of inverse modeling, or restoring trishear folds in order to test possible models, comes from Allmendinger (1998). The various data inversion methods that the program is capable of using come from different sources, which are given in the relevant sections of this manual. Some understanding of trishear fold kinematics is advised when using this program, so that it is not treated as a black box. The papers cited in this paragraph will provide a good starting point for the interested user.

Several conventions used by the program should be noted in order to avoid difficulties:

Trishear Velocity Field: The velocity field used is that of Zehnder and Allmendinger (2000), which is not the only possible trishear velocity field. Users of Move should note that the trishear velocity field used in that software (from Hardy and Ford, 1997) is slightly different from the Zehnder and Allmendinger (2000) solution. Differences are typically small, but noticeable.

Fault Parallel Flow: Outside of the trishear zone, the program assumes fault parallel flow. Fold axes bisect bends in the fault, and slip is conserved across fault bends. This will match the fault-bend folding theory of Suppe (1983) only when stratigraphic horizons and bedding dips are parallel to the lower fault segment.

Dip Directions: The program uses the convention that bedding dips down to the right in cross-section are positive and those down to the left are negative. This holds for dip data and for regional dips.

Fault Dips: The fault ramp angle is defined using an angle from 0° to 180°, measured up from the horizontal x-axis. Thus faults that dip down to the left fall in the range 0° to 90°, and those that dip down to the right fall in the range 90° to 180°.

If you have questions or comments about this software, or if you discover any bugs in it, contact me (David Oakley) at doo110@psu.edu.

2. Model Setup

Running InvertTrishear requires specifying the data to be used, the fault model, the parameter space, the data inversion algorithm, and a variety of options. This can either be done directly when running InvertTrishear (in which case the options chosen will not be saved) or by using Setup_InvertTrishear, which creates a text file that InvertTrishear can read. Using

Setup_InvertTrishear is recommended in most cases, as it creates a record of what options were used and allows for reuse of the options file. The questions asked by Setup_InvertTrishear are nearly identical to those asked by InvertTrishear if inputting options manually, and any differences are noted below.

The program runs in a console or command prompt window and has no graphical interface. To setup a run, you will be asked a series of questions or prompted to enter values or file names. All model runs require creating data files and a parameters file, in addition to specifying options for the program. These are simply text files.

2.1 Options file name (Setup_InvertTrishear only)

When running Setup_InvertTrishear, the first prompt is:

Enter name of a file to save the options to:

This is the name of a text file, including the file extension, in which the options specified during setup will be saved. This file will be created in the same folder that Setup_InvertTrishear is in.

2.2 Data

2.2.1 Number of Data Types

The next prompt is:

Enter number of different data types:

For this prompt, enter a number between 1 and 3, specifying how many different types of data will be used to constrain your model.

2.2.2 Data Types

After entering the number of data types, you will be prompted to specify what types of data they are:

Data Are (list all that apply):
(1) beds
(2) dips
(3) points on a fault

Enter any or all of the numbers 1, 2, 3 separated by commas. The total number of data types specified must match the number of data types entered for the previous prompt.

2.2.3 Data Files

For each data type chosen, you will be prompted to enter a data file name:

'Input bedding file name (with extension)'

'Input dip file name (with extension)'

'Input fault file name (with extension)'

As the prompts suggest, you must include a file extension with the file name. The data files are all text files, listing the coordinates of the data points in the 2-dimensional Cartesian coordinate system of the cross section, along with any other necessary information, such as dip. Data must be projected on a cross-section and converted to a 2-dimensional cooordinate system prior to using this program.

It is best to leave a blank line at the end of each data file. Failure to do this, may result in the last line not being read. Adding too many blank lines, however, may cause the program to crash.

2.2.3.1 Beds

Bed data consists of points along a marker bed or contact. They may be the digitized trace of a bed, as from a seismic interpretation, or they may be contact points mapped on the surface. The data file is a two column text file and may contain multiple beds. At the beginning of each bed, put the word "bed" in the first column and a name to identify that bed in the second column. Below that, list the coordinates of all points in the bed, with x coordinates in the first column and y coordinates in the second. Repeat for all additional beds.

2.2.3.4 Dips

Dip data represent the apparent dip of a bed within the plane of the cross section. Strike and dip measurements must be projected to the cross section in another program (or by hand). The dip data file contains three columns: x-coordinates, y-coordinates and dips. No headings are needed.

The convention used by the program is that dips are positive if dipping down to the right and negative if dipping down to the left.

2.2.3.5 Point on a Fault

These data consist of points known (from field, well, or seismic data) to lie on the fault. Any number of points may be used. The data file is simply a two column text file, listing x and y coordinates. No headings are necessary, since there can only be one fault in a given model.

It is important to note that the program only calculates distance between the points and an infinite line through the fault. Thus if a point is above the tip of a blind fault but in line with the fault, the error will be only the distance from the point to the line rather than that from the point to the fault tip. Users should be careful of this in situations where it could cause a problem.

2.3 Fault Tip

After entering all data files, the next prompt is to choose which fault tip position to solve for.

Tip position is:
(1) Initial
(2) Final

The initial tip position is the position of the fault tip before propagation begins, and the final tip position is the position of the fault tip after fault propagation (and slip). The model can search for

either one, but the parameter space should be chosen accordingly. If one is known, the other can be calculated based on the fault geometry.

2.4 Fault Types

The program is capable of modeling three different types of faults. You will be prompted to choose a fault type.

Choose Fault Type
(1) Straight Fault Ramp
(2) Ramp from Horizontal Detachment
(3) Fault with a bend in it

2.4.1 Straight Fault Ramp

This is the simplest fault type: just a straight ramp extending infinitely far down. It will produce a monocline, with a trishear zone in the forelimb. Model parameters are fault tip position (x, y), ramp angle, total slip, P/S, ϕ, and the concentration factor s.

2.4.2 Ramp from Horizontal Detachment

This is a simple step fault-propagation fold, in which the fault ramp steps up from a horizontal detachment. This will produce a fold with a straight backlimb, a flat crest, and a curved, trishear fold forelimb. In addition to the parameters for the straight fault ramp, this model adds detachment depth as an additional parameter.

After choosing this model, you will see the prompt:

Must detachment depth be at initial fault tip depth? (0/1)

Enter either 0 (no) or 1 (yes). If no, the initial position of the fault tip may be anywhere above the detachment. If yes, the fault tip must propagate up from an initial position at the detachment. If you answer yes, you should not include detachment depth in the parameters file, since the program knows it will be the same as the initial y tip position.

2.4.3 Fault with a Bend in it

This fault model has two straight-segments. Unlike the horizontal detachment model, neither segment is required to be horizontal. Currently, the program only allows models in which the upper segment is steeper than the lower one. Models in which the lower segment is steeper will be rejected. This model adds two additional parameters: the elevation at which the bend occurs and the ramp angle of the lower segment.

Two prompts will appear after choosing this model:

Must depth of fault bend be at initial fault tip depth? (0/1)

This is analogous to the detachment depth at initial fault tip prompt. 0 is no and 1 is yes. It may be useful if, for example, you wish to have a fault propagating up from a low-angle, but not horizontal detachment. If choosing yes, you do not have to include bend elevation as a parameter. Note that this option is only available if you are solving for the initial fault tip position.

Can initial tipy be below fault bend? (0/1)

This prompt appears only if choosing no to the above prompt or if the fault tip position to solve for is final. For this prompt, choosing no (0) indicates that the initial position of the fault tip must be above the fault bend. Choosing yes (1) allows the fault tip to start above or below the fault bend.

2.5 Parameters File

The next prompt is:

Input Parameter File Name (with extension)

This requires you to enter the name of a file defining the parameter space, which you must set up by hand. This is simply a text file, with three columns specifying the minimum, maximum, and step size values for each parameter (in that order), followed by two additional lines. All model parameters must have a minimum value, maximum value, and step size specified. If a specific parameter is known or assumed (such as knowing the ramp angle or assuming that s = 1), then simply enter the minimum and maximum as the same value and enter any step size. The meaning of the step size varies depending on the inversion method chosen. Even in methods for which it is not needed, some value must be put here. For grid based methods, the difference between maximum and minimum values must be a multiple of the step size.

The order of parameters is as follows. Not all parameters will be present for all fault models. Note that the parameter phi is half the trishear apical angle, not the full angle.

Tip x coordinate
Tip y coordinate
Total slip (must be positive)
Ramp angle (upper segment in multi-segment models)
Phi
P/S
s
Ramp angle (lower segment)
Detachment or bend y coordinate.

Following the parameters are two additional lines, with only one column each. The first is the sense of slip, which should be 1 for thrust faults and -1 for normal faults. Since normal faults do not currently work in the program, it should always be 1, and you should only use the program for thrust faults.

The second extra line is for the increment of slip. This is used in the trishear zone and in any other case where slip must be calculated incrementally. This is simply a number, and will be in whatever units the section coordinates are in. A smaller step size will make for a more accurate trishear model but will take longer.

2.6 Inversion method

This prompt allows one to choose the inversion method to use, from any of seven possibilities allowed by the program.

Choose Method:
(1) Grid Search
(2) Grid Monte Carlo
(3) Monte Carlo from a normal distribution
(4) Metropolis-Hastings Algorithm
(5) Adaptive Metropolis
(6) Robust Adaptive Metropolis
(7) Adaptive Parallel Tempering

2.6.1 Grid Search

This is the grid search method described by Allmendinger (1998). A multi-dimensional grid of values is specified in the parameter file, and the program test all models on the grid. This method is best suited to situations in which some of the trishear parameters are known, and the others lie within a limited range. For many dimensions and large ranges, either the number of models will be very large and thus the model will take a long time to run, or the grid will be too sparse to adequately sample the parameter space.

2.6.2 Grid Monte Carlo

This is a Monte Carlo simulation that randomly draws samples from a grid of values, specified by the parameter file. If choosing this model, you will see the prompt

Enter number of models to run:

For this simply enter the number of models. Typically, a larger parameter space will require more models to sample adequately. This method requires fewer model evaluations than the grid search to achieve a similar quality of result, but it can still grow unwieldy when the parameter space and number of dimensions are large.

2.6.3 Monte Carlo from a normal distribution

This method also takes random samples, but it does so from a multivariate normal distribution. Samples are drawn from the continuous parameter space, not from a grid, so the step size in the parameter file is ignored. Choosing this method will bring up too additional prompts:

Enter number of models to run:

and

Enter filename to read proposal distribution from:

The first is simply the number of samples to take. The second asks for a text file which will define the normal distribution from which samples are to be drawn. The file must have a number of columns (N) equal to the number of model parameters. The first line lists the mean values for

each parameter for the normal distribution. This must be followed by a blank line. After that, the next N lines and N columns give the covariance matrix for the multivariate normal distribution.

This method is useful if one already has a good estimate of the best model parameters. It is not useful for parameters that are poorly constrained a priori or for multimodal probability density functions.

2.6.4 Metropolis-Hastings Algorithm

This is the Metropolis-Hastings Markov chain Monte-Carlo (MCMC) method. The proposal distribution is a multivariate normal distribution, with standard deviations given by the step size for each parameter and with covariance assumed to be zero. In my experience, I tend to get very low acceptance rates, but in some situations this algorithm may work well. Nonzero covariance might help, and may be added to a future release, but in general it is difficult to choose a good proposal distribution without some prior knowledge of the expected target distribution.

When choosing this algorithm, one will see the prompts

Enter number of models to run:

and

Should initial model be (1) random or (2) specified?

The number of models is self-explanatory. A random initial model will be chosen from a uniform distribution over the parameter space. A specified initial model is useful if one has some prior constraint on the expected values of the model parameters. If choosing a specified initial model, one will then see

Enter initial values for all N parameters:

where N will be replaced by the number of model parameters. Initial values for all parameters should then be entered in order (same order as the parameters file) separated by commas. Initial values should be within the limits of the parameter space.

2.6.5 Adaptive Metropolis

This is the adaptive metropolis (AM) algorithm of Haario et al. (2001). The proposal distribution is a multivariate normal distribution, with initial standard deviations given by the step size for each parameter and with initial covariance assumed to be zero, but the covariance matrix is adapted as the run proceeds. The scaling parameter (s_d) is $(2.4)^2/d$, as suggested by Haario et al. (2001), where d is the number of dimensions of the parameter space, which is the number of parameters, excluding any parameters for which minimum and maximum values are the same. In my experience, I have not had very much success with this algorithm, but I have not tested it extensively.

Additional questions that the user will be prompted to answer for this algorithm include

Enter number of models to run:

and

Should initial model be (1) random or (2) specified?

both of which are the same as for the Metropolis-Hastings algorithm. If the initial model is specified, this is followed by

Enter initial values for all N parameters:

These questions are followed by the prompt

Enter model number after which to begin using adapted covariance matrix

The algorithm will initially run as a standard Metropolis-Hastings algorithm, up to a number of models specified by the user, at which point it will begin to adapt the covariance matrix.

2.6.6 Robust Adaptive Metropolis

This is the robust adaptive metropolis (RAM) algorithm of Vihola (2012). As for the AM algorithm, the proposal distribution is a multivariate normal distribution, with initial standard deviations given by the step size for each parameter and with initial covariance assumed to be zero, but the covariance matrix is adapted as the run proceeds. In this case the covariance matrix is adapted to target an acceptance rate of 0.234. I have found that this algorithm works well when the target distribution is unimodal but not as well for multimodal distributions.

The prompts for this algorithm are the same as for AM:

Enter number of models to run:

and

Should initial model be (1) random or (2) specified?

If the initial model is specified, this is followed by

Enter initial values for all N parameters:

2.6.7 Adaptive Parallel Tempering

This is the adaptive parallel tempering (APT) algorithm of Miasojedow et al. (2013). This algorithm uses multiple chains at progressively higher "temperatures," meaning that the higher temperature chains can more easily move about the parameter space. The target density, π is replaced with a tempered density, π^{β}. β is 1 for the lowest temperature level and approaches 0 at the higher levels. Chains are allowed to swap states, which results in the higher temperature chains exploring the parameter space to find probability maxima and the lower temperature chains exploring the vicinity of these maxima. The final results that are saved are the points visited by the lowest temperature ($\beta = 1$) chain. The covariance matrix of each chain is adapted using the RAM algorithm targeting a 0.234 acceptance ratio, and the β values are adapted as well, targeting an acceptance ratio of 0.234 for swaps. I have found this algorithm to be the most consistently useful. It is especially preferred for distributions that may be multimodal, such as when there is a large parameter space with poor initial constraint on the parameter values. The

drawback to this algorithm is that it is slower than the AM and RAM algorithms, although it remains much faster than the grid search.

As for the preceding three methods, the program prompts the user to

Enter number of models to run:

In this case, the number of models entered is the number per chain and is the number of results that will be saved. The total number of models that will actually be run by the software is this number times the number of temperature levels. Thus, this algorithm will take longer than others to obtain an equivalent number of results.

After this, as for similar methods, one is asked

Should initial model be (1) random or (2) specified?

If the initial model is specified, this is followed by

Enter initial values for all N parameters:

This is followed by another prompt:

Enter number of temperature levels to use:

This is simply the number of chains that will be run and allowed to swap states. The number must be an integer greater than or equal to 2. More temperature levels can better search the parameter space to identify multiple probability maxima and avoid getting stuck in local maxima. The more temperature levels that are used, however, the longer the program will take to run.

2.7 Objective Function

After specifying an algorithm to use, the user will be prompted to choose what objective function the data should be fit to.

Objective Function:
(1) Fit to flat line.
(2) Fit to known dip.
(3) Fit for dip.
(4) Fit to known line

This choice determines how the error is calculated for bed and dip data.

2.7.1 Fit to flat line

In this case, the data are fit to a horizontal line. For beds, the best-fit horizontal line to the restored points is calculated for each bed, and the error is the distance from each point to this line. For dips, the error is the difference between each dip and 0.

2.7.2 Fit to known dip

In this case, the expected dip of the restored data is known, but it is not necessarily 0. This is useful if there is a regional dip to bedding that is independent of the structure being modeled. This option will prompt the user to

Enter regional dip in degrees. (Negative is down to the left, positive right)

This should be a number between 0 and 90, which should be negative if the restored regional dip is down to the left and positive if it is down to the right. For beds, the best-fit line with this dip will be calculated (slope of the line is -tan(dip), given the sign convention). Errors are the distances between the restored points and this line. For dips, errors are the difference between the restored dip and the expected regional dip.

2.7.3 Fit for dip

This option allows the program to calculate the best-fit dip and fit to that. For beds, a best-fit line (slope and intercept) will be calculated for each restored bed and errors calculated from the distance of restored points to that line. For dip data, the error will be the difference between each dip and the average dip. An important limitation to this method is that each bed will be fit separately and dip data will be fit separately from bed data. Thus it does not fit a single best regional dip to all the data. Typically, it should be used only if data are points along a single bed or are dip data only. A second limitation is that the program does not currently output the regional dip that was fit.

2.7.4 Fit to known line

This option is only available if one of the chosen data types was beds. For this method, one must know both the expected regional dip of restored beds and the y-intercepts of the lines to which all beds in the beds data file should be fit. Errors are the distance from the restored point to the known line for points along a bed and the difference between the restored dip and the known regional dip for dip data. If the regional dip is 0°, then a known line equates to knowing the correct restored / undeformed elevations of horizons, but if it is not then known restored / undeformed elevations at a given point must be extrapolated to calculate a y-intercept at $x = 0$ within the cross-section coordinate system. The uncertainty introduced by this extrapolation is not currently considered by the model, so extrapolating over long distances is not recommended.

Choosing this option will lead to the prompts

Enter regional dip in degrees. (Negative is down to the left, positive right)

which is the same as for the Fit to known dip option. This is followed by

Enter number of beds:

This prompt only appears in Setup_InvertTrishear, not when entering options manually in InvertTrishear, since in the latter case the program will already know the number of beds. For this, simply enter the number of different beds in the bed data file. After this, you will be asked to

Enter y intercepts of all N beds in order

where N will be replaced by the number of beds. For this prompt, enter the y-intercepts of the beds, in the same order that they occur in the beds data file, separated by commas.

2.8 Results to Calculate

The next prompt is to choose the type of results to calculate and save for each model.

Results To Calculate:
(1) RMS Error
(2) Probability (unnormalized) for uncorrelated data
(3) Chi-square statistic (Cardozo, 2005)
(4) Probability (unnormalized) for data correlated along a bed

This determines how the goodness of fit of each model will be evaluated. If more than one data type is being used to constrain the model and/or if one is using any of the Markov chain and similar methods (Metropolis, AM, RAM, or APT), one must choose probability, either uncorrelated (2) or correlated (4). Option 4 will only be available if beds were chosen as one of the data types.

2.8.1 RMS error

This is the commonly used root-mean-square error. It provides a straight-forward way to calculate how well a model fits to data of any given type. It cannot be chosen, however, if there are multiple data types, since a single RMS error cannot be calculated for them all together.

2.8.2 Uncorrelated Probability

With this option, each error between data and model is used to calculate a probability, assuming Gaussian errors with a standard deviation specified for each data types. All the probabilities are multiplied together, allowing a single probability to easily be calculated for multiple data types. Metropolis algorithms decide whether to accept or reject a model based on a comparison of its probability with that of the current state of the chain, and therefore this option or option (4) must be used with those algorithms. This option assumes that errors in the data are not correlated with each other. The probability that is calculated and saved will not be the true probability of the model, as it is not from a normalized probability density function. During later processing, however, the results of the entire run can be normalized, and a properly normalized probability density function produced.

If one chooses this option, it will be followed by additional prompts. The first is

Do you want to propagate errors? (0/1):

0 is no and 1 is yes. If no, the error in the restored section is used to calculate a probability, with a standard deviation specified by the user that is the same for all data points. If yes, the standard deviation specified by the user is considered to be for the data in the deformed state (in which they were measured), and this error is thus propagated through the restoration for each point or dip. In most cases, since uncertainty in the deformed state can be better estimated than in the restored state, I recommend choosing yes (1).

After this, the user will see prompts to enter uncertainty for each data type being used in the model.

Enter Uncertainty in Bed Data:

Enter Uncertainty in Dip Data:

Enter Uncertainty in Fault Point Data:

For each of these, simply enter a number, which represents the one standard deviation uncertainty in the data. This value will be used to calculate the probabilities for the data (after error propagation, if applicable).

2.8.3 Chi-square statistic

This is the chi-square statistic used by Cardozo (2005) to evaluate models containing both bed and dip data. See that paper for further details. This option is not allowed if the data contain points on the fault. It can be used for bed or dip data alone but is intended to be used for both together.

2.8.4 Correlated Probability

This option is similar to option (2), except that the errors for bed data are correlated along the lengths of the bed. Dip and fault point data are treated the same as for option (2). This option is only available if bed data is one of the data types. Error calculation is calculated using the spherical variogram model, as described in Cardozo and Aanonsen (2009) and requires a correlation length to be specified. The distance along the bed is calculated as the distance from one data point to the next. Therefore, this option is only appropriate when there are points at short intervals along the trace of a bed, as for a bed digitized from a seismic image. It is not so useful for contacts mapped on the surface, which may be separated by long distances.

This option gives the

Do you want to propagate errors? (0/1):

prompt, which is the same as for option (2). It is then followed by

Enter Uncertainty in Bed Data:

which is in turn followed by

Enter correlation length for bed data:

This is the correlation length for the spherical variogram model. It is assumed to be in whatever units all other distance measurements in your cross-section are in. After this, one will see the prompts

Enter Uncertainty in Dip Data:

Enter Uncertainty in Fault Point Data:

if these types of data are included in the model. These prompts are the same as in options (2).

2.9 Errors file name

The last prompt is

Input file name to save errors to.

For this, you must enter a file name, including the file extension, for a text file in which to save the results of the run, which will include the RMS error, probability, or chi-square statistic calculated for each model that was tested. This file will be created in the folder in which you are running Setup_InvertTrishear, and it will overwrite any existing file with the same name.

3. Running the Program

To run the program, copy the InvertTrishear.exe program into a folder with the data files, parameters file, and the options file created by Setup_InvertTrishear. Then run InvertTrishear.exe. You will then see the prompt:

Choose source of run options:
(1) Input manually
(2) Read from file

Choosing (1) will allow you to answer the same questions as in Setup_InvertTrishear, after which the program will run with those options, but without saving them. Choosing (2) will allow you to enter the name of the options file you created with Setup_InvertTrishear and run with those options.

As it reads in the data files, the program will tell you how many beds, dip data, and points on the fault it has read in, so make sure these are the numbers you expect. If there is an error when reading the options file, the program will crash and close. If this happens, the best thing to do is to run InvertTrishear again choosing (1) Input manually and to enter the same options as in the options file in order to determine where the crash occurs. If the program crashes when loading one of the data files or the parameters file, check that this file is formatted correctly.

Depending on the number of models to be run, the amount of data, the total slip, and other factors, the time necessary to run the program can vary from minutes to hours. The program will periodically print out the number of models that have been run so far, and will print out the total time at the end of the run.

The program uses OpenMP to parallelize some parts of the computation. It will automatically use the maximum number of processes allowed on the processor it is running on, which will typically be twice the number of cores if the processor allows hyperthreading and equal to the number of cores otherwise. This means that the program is likely to use a large percentage of your CPU. Grid search, grid Monte Carlo, and normal distribution Monte Carlo, will use just about 100% of the CPU. It is recommended that you do not run the program while also using other CPU intensive programs. The program uses very little memory, however, so you are unlikely to have to worry about that.

When the program finishes running, it will print out some information that may be useful and will depend on the algorithm chosen. To close the program, simply press Enter after it has finished running.

4. Analyzing the results

The output of InvertTrishear will be a large text file containing results. For the grid search, this will list only the output values (RMS error, probability, or chi-squared), and the position of each value in the list will correspond to its position in the multidimensional grid, from which its parameter values can be derived. A Matlab script (ReadGridtxtErrs) is provided that reads this text file into a multidimensional array corresponding to the grid. For methods other than grid search, the text file produced will contain this same output value followed by the model parameters, in the same order that parameters are specified in the parameters file. A Matlab script (ReadMCtxtErrs) is also provided to read these results, saving them in an array of size (number of models) x (number of parameters + 1). In both cases, the text file is likely to be large, so reading it into Matlab and resaving as a smaller .mat file is recommended.

The following Matlab scripts are provided to help analyze the data. See the scripts themselves for further descriptions and listings of the arguments they take. Some are directly useful for analyzing results and some are just functions called by other scripts. Good scripts to get started on analyzing results are BasicMCMCPlotting or MCMC_plot_general for MCMC results and TrishearPDF_grid for grid search results. IndivModel is useful for analyzing an individual model, such as a best fit model, in more detail.

BasicMCMCPlotting: Plots histograms and figures showing the path taken by the Markov chain for any of the MCMC inversion methods.

BestFit_Grid: Calculates the best fit model from grid search results and provides the subscript for that model.

ContourMCMC: Produces a contour plot of a 2D histogram of MCMC results over two different parameters.

ind2subMat: Converts index of a multidimensional matrix in Matlab into subscripts stored in a single matrix.

IndivModel: Allows the user to analyze an individual model. Note that not all types of models have been thoroughly tested.

InputGoodOnly: Asks for input from the user and rejects any values not in a list of possible inputs.

MCMC_plot_general: Plots histograms of results from any of the MCMC methods as subplots within a single plot window.

Params_from_Errs_Grid: Turns the grid search results array or a subset of it into a list of parameter values more like the output for other methods.

QuadIntegrate: Integrates a grid search probability density function across a chosen dimension, using Simpson quadrature.

ReadBeds: Reads a bed data file into Matlab.

ReadGridtxtErrs: Reads the output text file produced by InvertTrishear for a grid search.

ReadMCtxtErrs: Reads the output text file produced by InvertTrishear for all Monte Carlo or Markov chain Monte Carlo methods (anything except grid search).

trishear_func: Trishear function for points, for a straight fault.

trishear_func_bend: Trishear function for points, for a fault with a bend in it.

trishear_func_decol: Trishear function for points, for a fault with a ramp from a horizontal detachment.

trishear_func_dip: Trishear function for dips, for a straight fault.

trishear_func_dip_bend: Trishear function for dips, for a fault with a bend in it.

trishear_func_dip_decol: Trishear function for dips, for a fault with a ramp from a horizontal detachment.

TrishearPDF_grid: Calculates a probability density function for results of a grid search.

TrishearPDF2D_grid: Calculates a 2D probability density function for results of a grid search.

xy_to_ze: Converts from the Cartesian coordinate system of the cross section to the trishear coordinate system.

ze_to_xy: Converts from the trishear coordinate system to the Cartesian coordinate system of the cross section.

References

Allmendinger, R.W., 1998. Inverse and forward numerical modeling of trishear fault-propagation folds. Tectonics 17, 640-656.

Cardozo, N., 2005. Trishear modeling of fold bedding data along a topographic profile. Journal of Structural Geology 27, 495-502.

Cardozo N., Aanonsen, S., 2009. Optimized trishear inverse modeling. Journal of Structural Geology 31, 546-560.

Haario, H., Saksman, E., Tamminen J., 2001. An adaptive Metropolis algorithm. Bernoulli 7, 223-242.

Hardy, S., Ford, M., 1997. Numerical modeling of trishear fault propagation folding. Tectonics 16, 841-854.

Suppe, J., 1983. Geometry and kinematics of fault-bend folding. American Journal of Science 283, 684-721.

Vihola, M., 2012. Robust adaptive Metropolis algorithm with coerced acceptance rate. Statistics and Computing 22, 997-1008.

Zehnder, A.T., Allmendinger, R.W., 2000. Velocity field for the trishear model. Journal of Structural Geology 22, 1009-1014.