FaultResampler 1.4

Disclaimer

FaultResampler 1.4 is distributed as is and the authors will not be held liable for any research damages resulting from the use of this program or README. The program will be updated periodically, with updates posted on http://myuiowa.edu/wbarnhart/programs.html. This code is intended for non-commercial use, and the authors ask that users please use the following citation when implementing this program in their research:

Barnhart, W.D., Lohman, R.B. (2010) Automated fault model discretization for inversions for coseismic slip distributions, Journal of Geophysical Research, doi:10.1029/2010JB007545

You should also use this reference to address all conceptual questions regarding the theory driving the FaultResampler 1.4 algorithm. Please direct all questions, comments, and suggestions to Bill Barnhart (William-barnhart-1@uiowa.edu). Also, if you find bugs, please inform us so that we might fix them during our periodic updates.

Table of Contents

2. Getting Started

FaultResampler 1.4 is a program designed to intelligently and efficiently generate a fault discretization for distributed slip inversions. We design the programs so that the size of individual dislocations within the fault plane are sized to the approximate area over which slip on that dislocation would be smoothed. Using the model resolution matrix, FaultResampler 1.4 derives fault discretizations that accurately reflect the ability of available data to resolve slip or the lack thereof. We thanks you for exploring our code and ask that you please contact us (William-barnhart-1@uiowa.edu) with any questions or bugs you may have or find.

In addition to Matlab, you will need the following tools in your path:

Matlab OPTIMIZATION TOOLBOX

Triangular Green's Function Generator

Algorithm written by Brendan Meade that calculates Green's functions for triangular dislocations. Program is available at http://summit.fas.harvard.edu/~meade/meade/Software.html. Driven in FaultResampler 1.4 by make_green_meade_tri.m.

Below, all commands to be executed from the Matlab command line are denoted in RED

After downloading the FaultResampler.tar file, untar the file and add it to your Matlab path:

```
tar xvf FaultResampler.tar
addpath(genpath('location_of_FaultResampler_folder/FaultResampler'));
```

Add the Meade triangular dislocation code to your path

```
addpath(genpath('location_of_Meade_code/name_of_Meade_code_folder'));
```

The finite element meshing code (mesh2d, version 24) from Darren Engwirda is now included in the Fault Resampler distribution. For referencing, or to see more information about the code, visit: http://www.mathworks.com/matlabcentral/fileexchange/25555-mesh2d-automatic-mesh-generation. Specific changes have been made to this code to be compatible with Fault Resampler

In the Fault Resampler distribution, you will find the following primary codes:

faultResampler.m
Driver code that will be run once you have set up your data loading files

loadResampData.m
Loads data sets for use and puts them into appropriate variables for the inversion and resampling codes run later. This code has the capability to load resampled interferograms and/or GPS offsets. The following sections will discuss the necessary conventions and data structures you will need in order to properly run faultResampler.m. This code also loads the predetermined fault geometries (not fault meshes) that will be used throughout the discretization algorithm.

makeStartFault_multi.m
Derives the initial starting mesh for the fault geometry or geometries loaded during loadResampData.m.

calcScales_multi.m
This code first runs the inversion code invertJRI.m to calculate the regularized Green's functions given the current fault discretization. It then calculates the model resolution matrix and derives a new size function of smoothing scales for the fault plane through a non-linear inversion in which we fit a best-fit gaussian curve.

makeNewMesh_multi.m
Takes new scales derived in calcScales_multi.m and generates a newly discretized fault plane.

checkScales_multi.m
Checks the predetermined termination criterion and either continues iterations or terminates iterations.

resampOptions.m
This script simply defines certain options used for Matlab functions such as lsqlin.m, lsqnonlin.m, as well as the meshing routine mesh2d.m. Here, you may also define the range of smoothing parameters to be explored by the inversion as well as the termination tolerance.

inversionFixedRake.m
inversionFreeRake.m
Generates Green's functions and determines the appropriate regularization parameter. This code uses the either the jRi criterion (automated) or an L-curve to determine the appropriate regularization parameter. There is also the option to begin producing figures to check how the progress of your discretization.

make_green_meade_tri.m
Driver code for the triangular dislocation code from Brendan Meade (see below)

triSmooth.m
Generates a Laplacian smoothing matrix for regularizing a grid of triangular dislocations.

ver2patchtri.m
Creates a structure patchstruct that contains information about the mesh geometry in a geologic reference frame. This is necessary for both deriving Green's functions and plotting slip distributions.

hfun1.m
A code necessary to run the meshing routine that states which input arguments will be used as the size functions for meshing.

gausfun2.m
Equation for fitting a Gaussian curve. Necessary for the nonlinear inversion of smoothing scales.

Other codes included

The following codes are supplementary and assist in plotting or placing observations in the correct data structures so that they can be loaded into Fault Resampler:

makeGPScorrect.m
A code not run in the main driver portion of the code, this is a supplemental code that will put GPS data into a data structure than can then be correctly loaded by loadResampData.m.

yrev.m
Reverses the direction of the y-axis, same as the Matlab command set(gca, 'ydir', 'reverse')

3. Data Input Formats

loadResampData.m includes several variables that you will need to change based on the data sets you are using and other aspects of the inversion you would like to control:

```
datafiles
faultfiles
rake_type
smooth_method
reg_method
data_type
rake
flag
```

datafiles

.mat files that include information on data locations, displacement magnitudes, uncertainties, and look direction or GPS component. These can be either GPS or InSAR displacements. If using horizontal displacements from pixel tracking methods, use the GPS construct described below. Multiple data sets can be loaded, and these can be InSAR, GPS, or both. **All geographic coordinates must be in UTM and all data and covariance values must be in meters.**

If you have X,Y, data, and look vectors in a series of matrices, you can write these to the appropriate format using the code **writeDatastruct.m.** You will need to save the resulting structure as a .mat file that will then be entered into **loadResampData.m**

```
savestruct = writeDatastruct(X,Y,data,S,'InSAR','10S');
save savestruct data.mat
```

If you downsample interferograms using the **resampler** tool provided on http://myuiowa.edu/wbarnhart/programs.html, you will not need to run **writeDatastruct**

The data file structure that is contained in your data.mat files should follow the following format:

```
savestruct =
           data: [1 x np struct]
           np: np
           covstruct: [1x1 struct]
           zone: 'UTM ZONE'
```

The structure should be named "savestruct" and contain to sub-structures, "data" which contains information about the data and "covstruct" contains information about the data covariance. "np" is the number of data points for that particular data set and "zone" is the UTM zone in which the data is found. The sub-structures must have the following fields and sizes:

```
savestruct.data =
           1 x np structural array
           X [1 x np]
           Y [1 x np]
           S [3 x np]
           data [1 x np]
           boxx [5 x np]
```

boxy [5 x np]

"X" and "Y" are the geographic coordinates of data points, "S" is the line-of-sight vector, "data" is the data values, and "boxx" and "boxy" are boxes of the resampled interferogram. If you are using GPS data, only "boxx" and "boxy" still must be defined, but can be filled with NaNs or zeros to simply act as place-fillers. These variables may be removed from the code entirely, but figure commands should be changed to scatter plots using "X" and "Y" instead of plots using the "patch" command.

savestruct.covstruct =
cov [np x np]

"cov" is the full data covariance matrix. This sub-structure may contain other variables, but "cov" is the only variable used in this code.

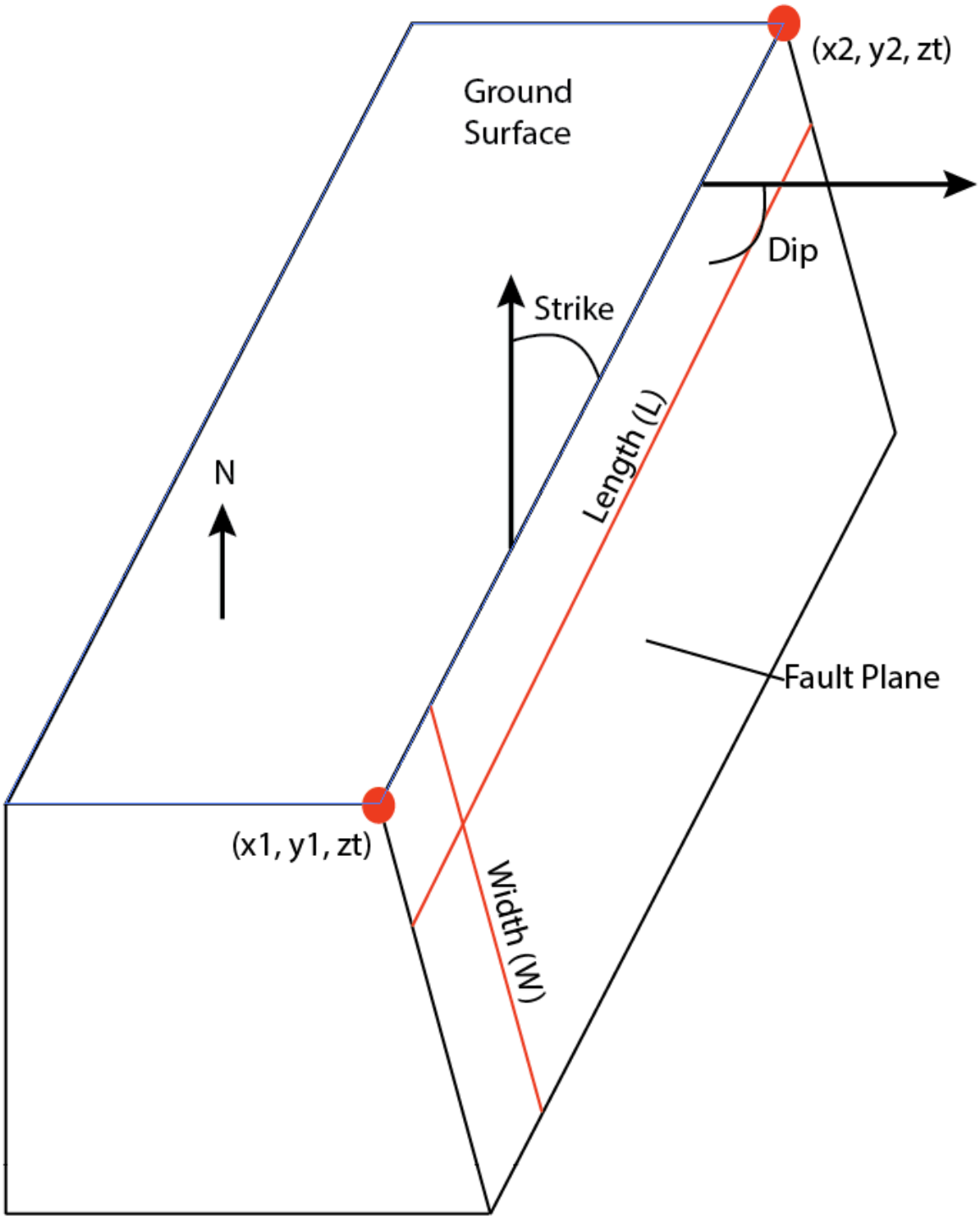
savestruct.zone contains the UTM zone in the form "11S", similar to the output of the matlab built-in command utmzone.m

faultfiles

We define the "fault geometry" as a rectangular fault plane that we wish to discretize into triangular dislocations. The fault geometries must be derived through other means, such as inversions for geometry, surface observations, or geological mapping. As such, you will need to load a fault.mat that contains the following structure, titled "faultstruct":

faultstruct =
strike
dip
W
L
rake
vertices
zt

Each variable will describe a single, planar portion of a fault. This may be the entire fault when using a single plane to model slip, or a single portion of a more complicated fault geometry. "strike" is the strike, between 0 and 360 degrees, measured clockwise from north. We use a right-hand rule convention for this. "dip" is the dip or the plane, measured using a right hand rule convention. "W" is the down-dip width of the fault plane in meters. "L" is the along-strike length of the fault in meters. "rake" is the slip direction measured from the strike direction. This variable may be left free to vary. "vertices" is a 2x2 matrix containing the geographic coordinates of the top two vertices of the fault plane. The first row contains the x-values and the 2nd row contains the y-values, and the points (columns) are in order as you travel in the direction of the strike direction (i.e., for a strike of due north, the first point will be south of the the 2nd point). "zt" is the depth, in meters, to the TOP of the fault plane. The conventions used for depth is positive is depth down into the earth (e.g. 10 km would be 10 km deep, whereas -10 km would be 10 km above the ground surface.



Graphical depiction of variables that need to be included in the fault structure. Vertices = [x1 y1; x2 y2];

The fault structure may contain multiple fault planes. To do this, the variable faultstruct should be a 1 x # of planes structural array, where each individual array contains all of the components described above.

rake_type

Fault Resampler has the ability to do both free rake and fixed rake (rake constrained to a single slip direction) slip inversions. rake_type denotes which option is to be used, and should be either ‘free’ or ‘fixed’. If the ‘fixed’ option is used, assign the appropriate rake value to the variable rake

smooth_method

Fault Resampler is currently programmed to regularize using either minimum moment regularization (no spatial smoothing) or a Laplacian smoothing matrix (spatially smoothed). Indicate the smoothing method you want here by imputing either ‘mm’ or ‘laplacian’

reg_method

Fault Resampler will choose a smoothing constant to be used by the smoothing method using one of two options: automatically choosing the constant based on the jRi criterion of Barnhart&Lohman 2010 or by allowing the user to manually choose a constant from an L-curve. Choose your option here by imputing either ‘jRi’ or ‘lcurve’

data_type

This variable will tell plotting commands what data type is used. Input ‘InSAR’, ‘GPS’, or ‘Mixed’

rake

Input desired rake (slip direction) to be used in fixed rake inversions. This value will be ignored for free rake inversions

flag

A plotting signifier. flag=1 will provide data misfit and slip distribution plots at every inversion iteration, flag=0 will only plot the final results

Work Flow

Once all data.mat files are created and their names/locations added into loadResampData.m, Fault Resampler can be run. Begin by running faultResampler.m, the driver code. This will first load the data.mat files using loadResampData.m. Data structures and covariance structures are extracted one at a time, and renamed into the variable resampstruct. Green's functions for the potential orbital ramp is then created, which may be adjusted according to the needs of different data sets (i.e., InSAR vs. GPS). loadResampData.m then extracts fault data from the fault.mat files and places all data variables into a structure with the appropriate conventions for Brendan Meade's Green's function code. Lastly, resampOptions.m is run, defining functions options for use later. Within loadResampData.m, a number of other important operations are carried out with respect to the data and Green's functions. From the data covariance matrix, we calculate the inverse Cholesky factorization ("Cdinv") which we then use to weight the data ("Dnoise") such that it will have uniform variance, and later the Green's functions. The Green's functions are augmented by the ramp calculated in this function.

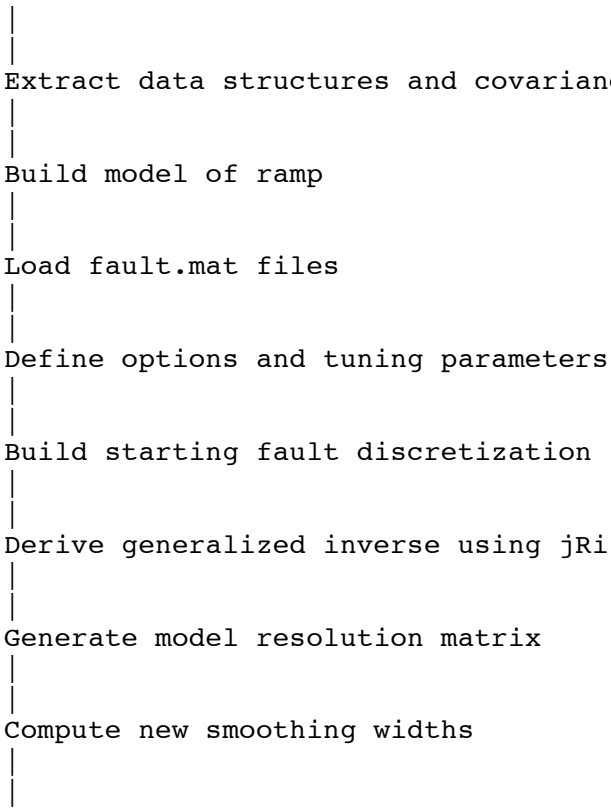
The next function called by faultResampler.m is makeStartFault_multi.m which will create the intial mesh for the first slip inversion. A variable "node" is defined which is the geometry of the fault in no coordinate frame. It refers to the length and width of the fault which will be meshed, which is then projected by ver2patchtri.m into a geographic reference frame. The variable "newscales"

should be adjusted (within makeStartFault_multi.m) so that there are at least several dislocations in the starting model but this starting value will not affect the final discretization.

After creating the starting fault discretization, the iterative process begins. First, calcScales_multi.m conducts a distributed slip inversion. Using the chosen regularization method, a smoothing parameter is chosen that regularizes the Green's functions calculated for the data distribution and the fault discretization of that particular iteration. Using the generalized inverse ("Gg") and Green's function weighted by the Cholesky inverse ("G"), calcScales_multi.m generates the model resolution matrix, "R." For each dislocation, a nonlinear inversion is then used to compute the best fit of a Gaussian curve to the values in R with the distance to all other dislocations. The Gaussian width of each iteration is saved in the variable "newscales."

Once all new smoothing scales are calculated, the new mesh is generated in checkScales_multi.m. The mesh is generated in the same way that the original starting mesh was generated in makeStartFault_multi.m, but "newscales" is no longer an arbitrary matrix of dislocation sizes. Once the new mesh is generated, the percent difference of disclocations in the new and previous meshes is calculated and compared to the tolerance level. If the percent difference is less than the tolerance, then iterations are terminated and the newest discretization is the final discretization. If the difference is greater than the tolerance, then another iteration is conducted with the newest discretization as the starting discretization. When iterations are completed, a final distributed slip inversion is completed using inversionFixedRake.m or inversionFreeRake.m.

Work Flow



Rediscretize
|
|
Check termination criterion
|
|
Terminate or reiterate at deriving generalized inverse

Tuning Parameters

A number of parameters and variables in FaultResampler 1.3 may be adjusted or tuned to better adapt the inversion process to a particular data set. Adjusting the following parameters may also help troubleshoot some problems you may encounter. Below, we define the variable, what functions it's found in, and how it might be changed to better suit your problem.

rampg (loadResampData.m) The ramp model. May be adjust to reflect a linear, bilinear, quadratic, etc. ramp. As is now, it's a quadratic ramp. You may also need to adjust the final ramp model if you are including both resampled InSAR and GPS data.

options2 (resampOptions.m) An option for lsqlin.m. This can be adjusted so better convergence to a single answer. This may become important if you are using free rake.

options3 (resampOptions.m) An option for mesh2d.m . By setting this to a higher number, dislocations may be less equilateral. A smaller number will allow mesh2d.m to throwout more dislocations so the final mesh will have more equilateral dislocations, but not necessarily best reflect the size function.

lambdas (resampOptions.m) The regularization factors that will be considered during jRi regularization. These will need to be adjusted depending on the noise in your data.

tol (resampOptions.m) The termination tolerance. This is a percent difference between the number of dislocations in subsequent iterations.

smooth (invertJRI.m) Laplacian smoothing matrix. May be replaced by any other regularization technique the user wants, such as minimum moment weighted least squares.

Demos

A demo is available that inverts a resampled Insar-like data set of a shallow slip distribution. Use data_demo.mat for the datafiles and fault_demo.mat for the faultfiles. To view the input slip distribution and data distribution, uncomment the plotting codes at the bottom of loadResampData.m.