Fold Simplification and Fault Estimation for Inverse Modeling of Fault-Bend Folding

Amanda Nicole Hughes

Advisor: Dr. Christopher D. Connors
Bachelors of Science Honors Thesis
Washington and Lee University
Lexington, Virginia
May 1, 2006

TABLE OF CONTENTS

I. ABSTRACT

It has been long recognized that many folds observed in the field and in seismic reflection data are created as stratigraphic layers are displaced over nonplanar fault surfaces. As material is moved over a bend in a fault, the rock layers are forced to fold, a process termed "fault-bend folding." Fault-bend folding has been geometrically characterized and forward modeled, but a rigorous inverse model does not exist. An inverse model is currently being developed that uses a genetic algorithm to find the fault shape over which displacement of material fault could have created the observed structure given the geometric and kinematic constraints of fault-bend folding. As part of the development of the inverse model, this study aims at creating a reasonable initial population of faults for the genetic algorithm by observing the fold shape and estimating the fault shape based on a few aspects of the fold. By integrating a few simple structural geology concepts into the prediction of fault shapes from fold shapes, a program was created that produces a more intelligent initial population of folds for the genetic algorithm.

The algorithms developed can be separated into two categories: simplification and description of the observed fold and fault shape estimation. Fold simplification is accomplished by an area-based line simplification algorithm. The properties of the fold are observed (line length, spatial X and Z locations) and dip domains are described. The fault shape is then estimated by using the information observed about the fold and dip domains to predict the initial X and Z location of the fault, length of the fault, number and location of fault bends, and slope of fault segments, and the input slip is also estimated. The algorithms were tested on four synthetic fault-bend fold models that were generated by the forward modeling program. The tests explored the estimated fault shapes generated by varying the fault parameter range, line simplification, and complexity independently and together. It was observed that the line simplification is the most sensitive parameter to change, and that variations in fold shape and complexity make it such that there is no single optimal value for any of the parameters used. As such, it was determined by using the test functions developed that by varying all of these parameters within a specified range it is possible to generate an initial population of faults with the desired variability around the known fault shape. Future work includes testing and refining the algorithms and testing their most effective implementation into the inverse model.

II. INTRODUCTION

It has been long recognized that many folds observed in the field and in seismic reflection data are created as stratigraphic layers are displaced over nonplanar fault surfaces. As material is moved over a bend in a fault, the rock layers are forced to fold, a process termed "fault-bend folding" (Figure 1). This process was first geometrically and mathematically described by Suppe (1983). The description was extended to multi-bend fault-bend folds (Medwedeff and Suppe, 1997) and curved hinge fault-bend folds (Suppe, et al., 1997). An understanding of the geometry and kinematics of fault-bend folding is essential in addressing structures observed in the field and in seismic reflection data (Shaw et al, 2005).

The process of fault-bend folding has been kinematically forward-modeled (Hardy and Poblet, 1995), meaning that given an input fault shape, as slip increases at each time step, the deformation can be described by a velocity model of the particles (Figure 1). As expected, the inverse model asks the reverse of this—given an observed fold shape, displacement of material over what fault shape could have created that structure given the geometric and kinematic constraints of fault-bend folding? Computer modeling of fault-bend folding has been addressed by several studies (Contreras and Suter, 1990, Zoetemeijer and Sassi, 1992, Hardy and Poblet, 1995, Egan et al, 1999), but a rigorous inverse model does not exist.

The velocity description of the inverse model has been derived for simple cases (Connors, et al, 2005), and the extension of this model to more complex structures is the motivation for this study. A complete forward fault-bend folding model has been developed (Connors and Upchurch, 2003) that serves as the basis for the inverse code. The inverse code is based upon a genetic algorithm, which functions by the following steps:

1. An initial population of fault layers and input slip are created,
2. The "fitness" of each population member is assessed, based on how closely the fold forward modeled from that set of parameters fits the observed attributes of the objective fold.
3. The most fit members are selected to survive to the next generation,
4. The members that survive are mutated by randomly changing their attributes slightly. In addition, population member attributes are shared or "crossed-over" between members to create some new members with characteristics from multiple members from the previous generation.
5. Steps 2-4 are repeated.

While this model works very well for simple geometries, its ability to address more complex structures is limited because many of the fault parameters are "hard-coded" and have to be manually changed in order to find better inverse solutions.

This study attempts to address this by creating a method of looking at folds and estimating the fault shape based on a few aspects of the fold. By integrating a few simple structural geology concepts into the prediction of fault shapes from fold shapes, a program was created that produces a more intelligent

initial population of folds for the genetic algorithm. This will allow for a prudent narrowing of the parameter space being searched—unreasonable fault shapes will not be included, yet the parameter space is not too specific such that the genetic algorithm might become stuck in local minima. This will allow only reasonable fault options to be searched, and will create an improved ability to analyze more complex structures.

III. THEORY

A. Fold Simplification

Folds can be defined in the traditional way of delineating two dipping "limbs" separated by a hinge (figure 2a); however, it is also useful to think of a fold as a series of "dip domains," or regions of uniform dip, separated by axial surfaces (figure 2b). The latter definition of a fold is useful in describing how fold and fault shape are geometrically related for fault-bend folding.

Line Simplification

The first step in reducing a fold down to its essential components is simplification of the line segment that represents the fold shape. Line simplification is performed by an area-threshold-based filtering algorithm. First, all collinear points are removed. Then, triangles are created by connecting three collinear points on the polyline, and the areas of the triangles are calculated. If the area of a given triangle is above the specified standard deviation of the area of all three-point triangles, then this point is eliminated from the polyline (Visvalingam and Whyatt ,1993).

Describing dip domains

There are multiple ways to describe dip domains that are useful in predicting fault shape. One method is that a dip domain is defined as a dipping region bounded by axial surfaces that dip the same direction. The dip domain

may include several segments that dip varying amounts, but the full dip domain includes all segments that are bounded by axial surfaces that dip the same direction (Figure 3a).

Another useful way to describe dip domains is that each dip domain is a dipping segment that is bounded by a synclinal and an anticlinal hinge (Figure 3b). For example, within a dip domain bounded on the left by a synclinal axis, there may be multiple synclinal hinges within the dip domain, but the dip domain will not end until an anticlinal hinge is encountered.

B. Fault Prediction

Using the above described methods of fold simplification, there are a number of properties of the fault that may be estimated. In some cases, knowledge of more than one folded layer allows one to make more detailed predictions.

Initial position (X) and Layer length

Estimation of the initial X position of the fault is determined by the minimum X value of the folded layer (the leftmost point of the fold) minus the 1.5 times the estimated slip amount. Because the estimated slip is not certain, as slip is consumed and produced at synclinal and anticlinal fault bends, respectively, allowing for 1.5 times the estimated amount ensures that the initial position of the fault is beyond the undeformed extent of the layer.

The length of the fault is the maximum X value of the folded layer (the rightmost point of the fold) minus the initial X value plus half of the slip. This ensures that the last point of the fault is sufficiently beyond the end of the folded layer. Both layer length and initial X position are fixed because they are maximum estimates that are beyond the extents of the fold that define the boundaries. All parameters to follow are not fixed, as varying these parameters is useful in generating a variety of different faults.

Estimate initial height (Z)

Estimation of the initial height of the fault is dependent on the amount of input data for the fitness objectives. If the shapes of multiple folded layers are input, then the predicted Z value for the fault is the lowest Z value in the polyline minus this value times the range for the GRFPms. This is done so that the range covers the area immediately below the lowest Z value of the fold, with the highest possible value being the Z value of the fold.

If information on only one fold layer exists, then the maximum possible Z value is the minimum Z value of the fold minus the range of Z values for the fold. The range of Z values explored is then from this point minus three times the range of Z for the fold to the determined maximum Z value. The multiplier of three is arbitrary, but allows for a significant range of initial Z values to be explored. Since there is only information on one layer of the fold, little can be known about the depth of the fault below the fold, so a wide range of initial heights of the fault is appropriate.

Information about the observed fault may also be input as a fitness objective. If something is known (input as a fault objective) about the first segment of the fault, then the minimum value between the the minimum Z value of the input fault objective and the lowest Z value of the polyline is used for the initial height of the estimated fault (Z value).

Finding active axial surfaces to predict fault bend locations (FRW)

Integral to the concept of fault-bend folding is that as material moves over bends in the fault, folds in the rock layers develop. Thus the location of bends in the fault may be predicted by the shape of the fold. In a syncline/anticline bounded dip domain of a fold, the leading synclinal axis corresponds to an active axial surface and a bend in the fault. Subsequent synclinal bends within the dip domain similarly may be correlated with synclinal bends in the fault. The final anticlinal bend in the dip domain correlates with a passive axial surface; this axial surface represents the current position of the material that sat at the active axial surface when deformation began. Similarly, in an anticline/syncline bounded dip domain of a fold, the leading anticlinal bend in the fault corresponds to an active axial surface and an anticlinal bend in the fold, and subsequent anticlinal bends in the fold dip domain may be correlated with anticlinal bends in the fault, with the final synclinal axis corresponding to a passive axial surface (Figure 4a).

Because the axial surfaces modeled are relatively steep, and because the faults are only meant to be a close approximation around which a range of faults will be generated, it is adequate to approximate the X coordinate of the fault bend

as the X location of the corresponding bend in the fold (the exact location would be described as the point where the fault intersects the axial trace that corresponds to that bend in the fold; this approach may be implemented in the future).

The length of the fault is estimated, as described above, as the range in X values for the fold plus 1.5 times the estimated slip in order to ensure that the estimated fault is well beyond both sides of the undeformed layers. Because this is an overestimation, the relative width of the end segments as estimated is less than they should be; for that reason, the first and last fault segments' fault relative widths are multiplied by 0.3 and 0.2 to account for this error. These values were determined by empirically testing a number of faults and seeing what values best readjusted fault relative widths to account for the error in layer length overestimation.

In approximating the fault shape, it is useful to consider both the simple generalization of the fault shape and the more specific description. In the simple case (hereafter referred to as 'simple'), only the leading active axial surfaces of dip domains are correlated with bends in the fault. This approach provides a rough approximation of the fault shape (figure 4b). In the more general case, intermediate axial surfaces are also correlated with bends in the fault (figure 4b). This approach provides a more detailed approximation of the fault, but includes more points in the fault shape. The number of points in the simplified line controls the number of points that are correlated to fault bends in this case, so

the value used for the fold polyline simplification threshold plays an important role in the number of bends determined in the more complex approach.

Estimating input slip

The input slip may be estimated in several ways that are dependent on analyzing the backlimb of the fold. Slip is increased, or produced, at synclinal fault bends and decreased, or consumed, at anticlinal fault bends, so an approximation of the slip is dependent on the number and nature of the fault bends, which is in turn dependent on the number of points that result from simplification of the fold polyline. For one-synclinal-bend faults (like the one shown in Figure 5a), the length of the backlimb of the fold is a good approximation of the maximum amount of input slip. However, as the number of synclinal bends increases and the fault becomes more rounded, the length of the backlimb is much larger than the actual input slip (Figure 5b). Thus, maximum slip is calculated as the minimum value of the two following possibilities:

1. Length of the back limb (5a, 5b.)
2. The change in height of the fold divided by the sign of the maximum slope (5c.).

Because it is difficult to estimate the input slip in the case of very rounded faults (in a very rounded fault, a very small slip would produce a very long backlimb), because slip is produced and consumed at synclinal and anticlinal fault bends, respectively, the minimum slip estimate is zero.

Estimating fault segment slopes

The relationship between the slope of segments of the fold and segments of the fault in fault-bend folding is complex and is fully characterized by Suppe, 1983. However, to provide a simple approximation of this in order to generate a reasonable and sufficiently diverse initial population, the following algorithm for the estimation of fault slope is implemented:

1. The slope of the first fault segment is approximated as the slope of the first fold segment.
2. The slope of subsequent fault segments is estimated as the sum of the slopes of the current and previous fold segments (skipping flat segments because material is being translated through these regions) (Figure 6).

In cases where fault simplifications have a large number of points, fault slopes quickly grow to an irrationally steep slope. To address this, the maximum fault slope allowed is the maximum slope of any fold segment. While in some cases this leads to an under-estimation of fault slope, this provides reasonable fault shapes in most cases; refinement of this algorithm to address this issue will be addressed in the future. The minimum allowed slope of any fault segment is zero; while this is not always true (particularly in extensional cases), instances of negatively-sloping fault segments in contractional fault-bend folds are rare, and only contractional structures were modeled in this study. This will be addressed in future work as well.

IV. CODE IMPLEMENTATION OF THEORY

The above-described theories were written as several functions in Matlab, and are used in conjunction with previously written functions (Appendix II) to create populations of faults within specified parameter ranges based on the input fold shape (all functions written by the author except simplifyLine, written by C. D. Connors). The main function is 'estGRFPms', which is the abbreviation of the function's purpose—estimation of the generation of random fault parameters. Within this function, the most complex subroutine is the determination of dip domains, which is accomplished by the function 'dsbDipDomains'. Please refer to the program maps of 'estGRFPms' (Figure 7) and 'dsbDipDomains' (Figure 8) to see the flow of function implementation for this Matlab toolbox.

Simplification of the fold polyline is accomplished by the function 'simplifyLine'. Minimum and maximum X and Z values, slopes, and a variety of other quantities that describe the fold are attained by evaluating the function 'getFitObjVal'. From this information, dip domains are described as polylines that are bounded by synclinal and anticlinal bends or include all same-dipping axial surfaces by the function 'dsbDipDomains' (Figure 8). Fault bends are found in the 'simple' case by the function 'dscFaultBendsSimple' and the 'all' case by 'dscFaultBendsAll'. Slip estimates are made by 'estGRSlipPms'. Fault slopes for each segment are estimated by 'dscFaultSlopes', and the relative width of each fault segment is estimated by 'dscFRW'. From these estimations and the range

specified by GRFPmsVal, 'genGRFPms' creates ranges for fault relative width, fault slopes, and Z values.

The function 'testEstGRFPms' is a wrapper that runs 'estGRFPms' with specified range values, simplification values, and 'simple' or 'all' for a specified number of faults and plots them over the fitness objectives and known fault. 'testEstGRFPms2' is the same as 'testEstGRFPms', except that it inputs ranges for the specified values above and randomly selects from those ranges a value for each individual fault in the population. This allows faults in the population to have a range of fold simplifications, ranges for fault parameters, and 'simple' or 'all' designations, creating a diverse fault population.

Help documentation for each of the functions created in this thesis may be viewed in Appendix I. Additionally, these functions may be found in the folder …fbf\src\fbfInv\general\estimateInitial within the fault-bend folding modeling code.

V. RESULTS AND DISCUSSION

Several tests were developed and executed to explore numerous aspects of the fault estimation algorithm. The tests were designed to observe the range of fault shapes that could be generated, the sensitivity of different parameters, limitations of the current algorithm, and biases or tendencies of the algorithm to favor certain solution sets. It was then possible to create hypotheses about the causes of the trends that became apparent from observations of the results of these tests.

It is important to note that the intention of developing this algorithm is to create a broad and diverse range of possible fault shapes that are reasonable given basic properties of the observed fold; it is readily acknowledged that many of the approximations made in the algorithm do not directly predict the precise answer. Rather than predicting the exact solution, the intention is to create a population of reasonable solutions that are similar to the correct one, as it will be valuable to use this population in the genetic algorithm of the inverse model. Because the exact solution is not the primary goal of the algorithm, it is understood that the estimations made in the algorithm introduce some bias into the solution populations.

Fault estimation was conducted on four synthetic fault-bend fold models that were generated by the forward modeling program: one flat-20 degree ramp-flat fault-bend fold (Figure 9), one flat-ramp flat where the ramp had two bends (10 and 30 degree fault slopes for ramp segments) (Figure 10), a rounded fold

(Figure 11), and flat-ramp-flat-ramp-flat (two 20 degree ramps separated by a flat detachment) (Figure 12). In figures 13, 14, 15, 16, 17, 24, 25, 26, 27, and 28 a population of 50 faults is displayed. The 'all' designation is used in the generation of all figures except in figure 23, column 1, and 50% of the time in figures 24, 25, 26, 27, and 28.

A. Varying fault parameter range value

Exploration of the parameter space for the fault parameter range, or size of the window within which fault parameters can be chosen, are shown in figures 13, 14, 15, 16, and 17 for the different fold shapes described above. As shown in Figure 13, points that represent the vertices of the fault cluster near the known fault vertex locations, but as the range value increases, the estimated vertex points spread more evenly around the known fault vertices. The range of the first bend's estimated vertex is flatter because the slope of the first segment is always zero (there is no variability in this value regardless of the fault parameter range specified), as specified by the algorithm, because the slope of the first overlying segment of the fold is also zero.

For each of the fold shapes shown in figures 14-17, as the range value increases, the variability of the fault shape increases, but estimated fault shapes generally cluster around the known fault shape (in red). It is important to notice that in the simple flat-ramp-flat case (Figure 14), faults are fairly evenly distributed above and below the known fault shape, and thus the population of faults created appears to cover the parameter space as desired. However, when

the fault shape becomes more complex, approximations made in the algorithm cause the estimated populations to be biased in certain directions; for instance, because the algorithm estimates the location of an anticlinal fault bend as directly below a leading anticlinal hinge in the fault (instead of down the axial trace), the location of the first point in the upper detachment in figure 15 is estimated as too far to the right; additionally, the fault slope is underestimated because the algorithm used to determine fault slope is inexact.

Similarly, the fault is estimated as too high (above the known fault shape) in the flat-ramp-flat-ramp-flat case (Figure 17) because, while the slopes are estimated correctly, the same error in fault-bend location as described above has caused the relative width of each of the segments to shift slightly. Because the beginning and end segments are multiplied by scalars to account for the overestimation of fault length (described in III.B section 3) this compresses the other segments; multiplying any intermediate flat fault segments by a scalar value should rectify this, as accounting for the overestimation of fault length should be distributed over all flat segments in the fault.

However, it is also evident in this and the following examples (Figures 14-17) that the higher the fault parameter range value, the more population members fall both above and below the known fault shape (there is more symmetry to the range about the known fault), suggesting that a higher value in fault parameter range can negate some of the bias introduced by the approximations made in the algorithm. Because the inverse model will be used to address fold shapes with considerable complexity and the fault shape cannot

be perfectly estimated, a relatively broad range of population members is necessary to find the optimal solution.

It has been observed that at and above a range value of 0.4, the population members become unrealistic and no longer reflect the information known about the fault; thus it appears that using value around 0.3 may create the best range for more complex faults, while it is possible to generate a reasonable population for simpler faults (Figure 14) with a smaller range value.

B. Varying line simplification

The line simplification value may be considered the most sensitive parameter tested, as slight variations in this parameter greatly affect the simplified fold shape, which then controls the nature of the estimated fault shape. The line simplification number represents the cutoff standard deviation of the area of all three-collinear-point-generated triangles, where a point is removed from triangles having a lower area than the specified line simplification value. Line simplification values of 0.01, 0.05, 0.10, 0.50, and 1.00 were tested on the four example fault-bend fold shapes (Figures 18-21).

It is evident that different line simplification values are required to capture the critical points of the fold, and that the nature of the fold shape determines the range of line simplification values that will work. For example, in the case of the simple flat-ramp-flat (Figure 18) fault-bend fold, because the fold shape is simple and angular, higher values (0.50-1.00) for the simplification value find the essential points on the fold, and the estimated fault shape is very close to the

known fault shape. Using a lower line simplification value (0.001-0.10) results in superfluous points being retained in the fold shape, and as a result, the estimated fault shape is also less accurate.

However, when there are two bends in the fault ramp (Figure 19), use of line simplification values of as low as 0.10 and result in elimination of the intermediate bend from the simplified fold shape. Because of the subtle changes in slope in the backlimb of the fold, the area of triangles connecting these points is fairly small such that these points are eliminated with fairly low cutoff values. At higher simplification values, the estimated fault does not have any information about the intermediate bend in the ramp, and is estimated as a single, gently sloping backlimb.

This consequence is also observable in the case of the rounded fold (Figure 20); because the area of collinear points on the rounded limbs is very small, these points are eliminated quickly as line simplification value increases. In this case, because the change in slope is very gradual, the area of collinear triangles are very small, the fold shape is reduced to a single-segment backlimb at a simplification value of 0.16, and high line simplification values (0.50-1.00) result in no intermediate points being retained in the simplified fold shape.

In the case of the flat-ramp-flat-ramp-flat fault bend fold, a relatively low simplification value is necessary, as the fold shape becomes too simple for the fault estimation code to find the second ramp with increasing line simplification values. Part of this may be attributable to the way dip domains are defined; if dip domains were defined as bounded by synclinal and anticlinal hinges instead of

including all segments with same-dipping axial surfaces, this could help address the issue observed in higher line simplification values.

Overall, variation of line simplification value for different fold shapes reveals several general trends. First is that there is no single optimal fold simplification value—the nature of the fold shape determines whether high or low fold simplification values work best and whether the most favorable range of values is narrow or wide for a given fold shape. Additionally, it is probable that use of "simple" or "all" specification and describing dip domains differently may have an important impact on line simplification.

Also illustrated is that a tradeoff exists in estimating more complicated and rounded structures—using a small line simplification value results in an estimated fault shape that may be close to the actual fault shape, but has numerous points in the estimated fault. Conversely, the number of points may be reduced by using a higher simplification value, but critical aspects of the fold may be missed. It is unclear which approach is better—preliminary use of fault populations with numerous points in the inversion modeling code have shown that when initial faults have too many points, better solutions are not found in subsequent generations of the evolution.

C. Complexity: Simple vs. all

It is evident by testing different fault shapes that in different situations it is preferable to use a "simple" estimate of the fault rather than correlating all of the points in the fold to points in the estimated fault. For example, consider a dip domain in which all axial surfaces are dipping the same direction but there are multiple segments within it; the "all" designation would correlate each point in the dip domain with a point on the fault, whereas the "simple" designation would consider only the endpoints of the dip domain.

In the case of a flat-ramp-flat-ramp-flat fault-bend fold (Figure 22 a.), if the line simplification value is too low (top), the "simple" designation results in a better approximation of the fold; however, if a more appropriate line simplification value is used (bottom), this effect is removed and both cases are the same. In the case of the two-bend fault bend fold, if the fold is under-simplified (top), using the "all" designation, the resulting fault is too steep; the segments very quickly reach the threshold slope value. However if the "simple" designation is used, the second bend in the ramp is not observed. If line simplification values are more appropriate (bottom), the "simple" designation still misses the second fault bend, but the "all" designation captures the essential aspects of the fault. Use of syncline/anticline bounded dip domains with the "simple" designation should result in a very good approximation of the fault shape; this will be tested in the future.

D. Random Number Generation ('Seed')

The random number used in generating the fault population members also plays an important role in the appearance of the population. In figure 23, three different random number scenarios are shown; although all of the rest of the parameter values are the same in each test, each population qualitatively appears different because of the different random numbers used in each case. This effect, although subtle, is lessened by increasing the number of population members or repeating the process multiple times.

E. Combination of the above

As is evident by the above discussion of the effect of varying different range values in creating a fault population, there is no single value that works best for estimating all faults; different combinations of fold simplification values, fault parameter ranges, use of "simple" or "all" fold points, and syncline/anticline bounded dip domains or same axial surface dip direction dip domains are necessary to create an initial fault population. Thus it was determined that by randomly selecting a value for each of these parameters out of a specified range for each population member, a diverse fault population may be achieved.

In figures 24, 25, 26, 27, each of the parameters is given a range from which a value is randomly selected in the generation of each of the 50 population members (by using testEstGRFPms2, see Appendix 1). The parameters window is randomly selected as any value between zero and 0.3, and 50% of the time, the "simple" designation is used. The fold simplification range varies for each of

the different folds, as it was shown above that this is the most sensitive parameter and varies the most for each fold shape. In each of the cases, use of a range of values for each of the parameters has created a population that is appropriately distributed around the known fault shape.

VI. FUTURE WORK

There are several areas of future work aside from the issues addressed in the previous section. New algorithms to approximate the relative width and slopes of each fault segment will be tested in an attempt to improve these aspects of the fault prediction process. Integrating the syncline/anticline-bounded dip domain description into fault estimation will also be investigated, as this has been shown to lend more insight into the estimated fault in some cases.

The major area of future work is in using the fault populations generated by this estimation code as the initial population for the inverse fault-bend folding code. This has many challenges associated with it, as faults with different numbers of segments may prove difficult to integrate into the genetic algorithm. This will be tested extensively and will likely lead to investigation of several additional topics, including post-processing/simplifying the estimated fault or using the 'simple' designation and adding complexity in later generations in the genetic algorithm.

Long term goals include establishing a Bayesian belief net or neural net to get the fault estimation algorithm to find what values for the parameters are most effective for certain structures at finding the best estimated faults. This will ultimately lead to the ability to model more complex structures, including imbricates. Additionally, the ability to use more kinds of fitness objective data, such as strike and dip measurements and possibly growth strata, may be developed.

VII. REFERENCES

Connors, C. D., and J. Upchurch, 2003, A forward modeling program for fault-bend folding: GSA Abstracts with Programs, v. 32, p. 146.

Connors, C. D., S. Levy and M. Nelson, 2005, Forward and Inverse Modeling of Fault-Bend Folding (Abstract), International Conference on Fault-Related Folding, Beijing, China.

Contreras, J. and M. Suter, 1990, Kinematic modeling of cross-sectional deformation sequences by computer simulation: Journal of Geophysical Research, v. 95, n. B13, p. 21,913-21,929.

Egan, S.S., S. Kane, T.S. Buddin, G.D. Williams, and D. Hodgetts, 1999, Computer modelling and visualisation of the structural deformation caused by movement along geological faults: Computers & Geosciences, v. 25, n. 3, p. 283-297.

Hardy, S. and J. Poblet, 1995. The Velocity Description of Deformation, Paper 2: Sediment Geometries Associated with Fault-bend and Fault-propagation folds: Marine and Petroleum Geology, v. 12, p. 165-176.

Medwedeff, D.A. and J. Suppe, 1997, Multibend fault-bend folding: Journal of Structural Geology, 19, p. 279-292.

Rowan, M.G., and R. Linares, 2000, Fold evolution matrices and axial surface analysis of fault-bend folds: Application to the Medina anticline, eastern Cordillera, Colombia: AAPG Bulletin, v. 84, no. 6, p. 741-764.

Shaw, J. H., C. D. Connors, and J. Suppe, 2005 *Seismic Interpretation of Contractional Fault-Related Folds*: AAPG Seismic Atlas, Studies in Geology # 53: Amer. Assoc. of Petroleum Geologists, Tulsa, OK, 156 p.

Suppe, J., 1983, Geometry and Kinematics of Fault-Bend Folding: American Journal of Science, v. 283, p. 684-721.

Suppe, J., F. Sábat, J.A. Muñoz, J. Poblet, E. Roca, and J. Vergés, 1997, Bed-by-bed growth by kink band migration: Sant Llorenç de Morunys, Eastern Pyrenees: Journal of Structural Geology, v. 19, p. 443-461.

Visvalingam, M and Whyatt J D, 1993, Line Generalisation by Repeated Elimination of Points, Cartographic Journal, v. 30, no. 1, p. 46 – 51

Zoetemeijer and Sassi, 1992, 2-D reconstruction of thrust evolution using the fault-bend fold method: in McClay, K.R., ed., *Thrust Tectonics*, p. 133-140.

VIII. FIGURES

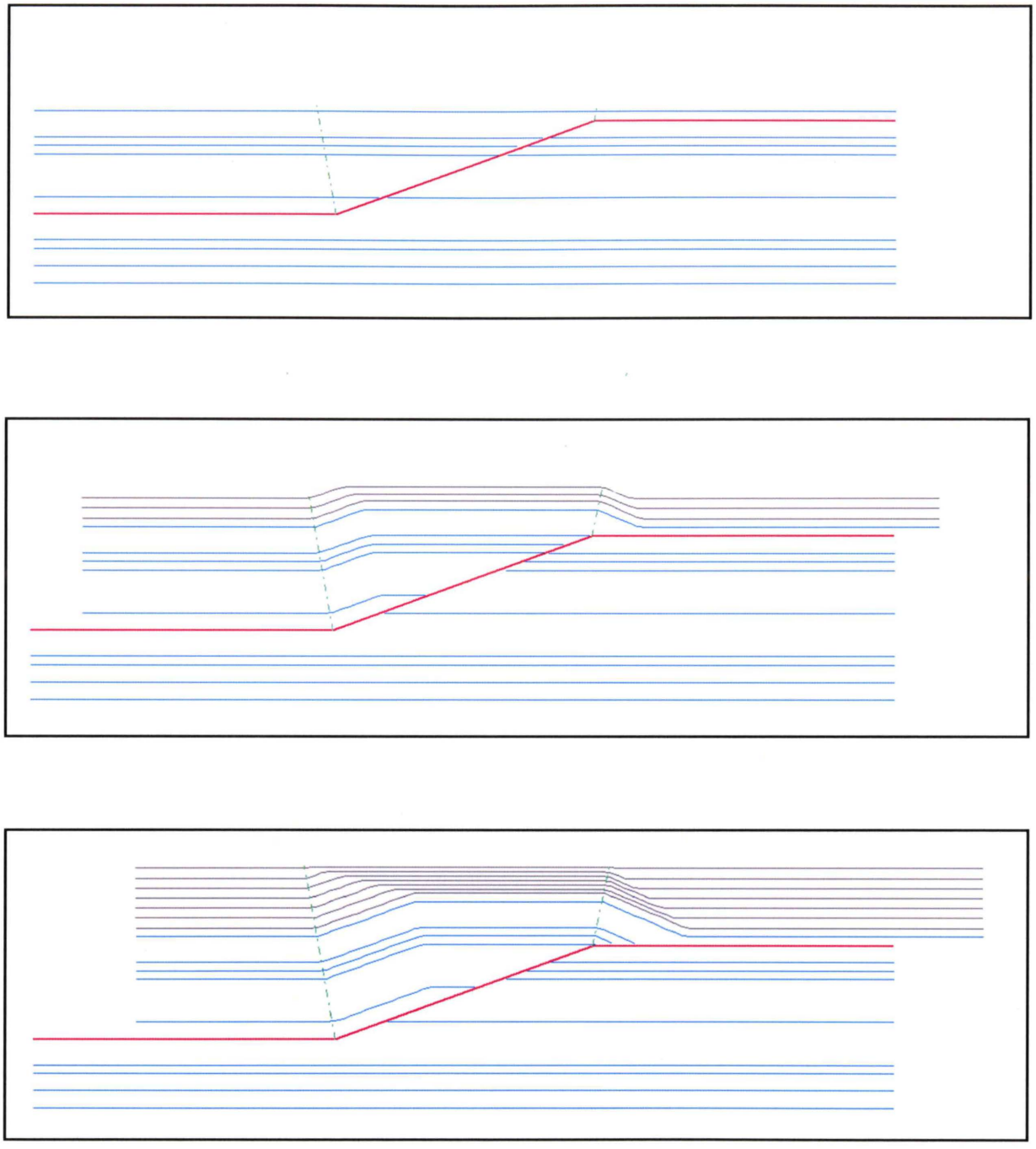


Figure 1. Kinematic model of fault bend folding; As slip increases (top to bottom) more material passes through the active axial surface, where it is translated as its movement becomes parallel to the next fault segment, forming a kink band. Red is the fault, blue are pre-deformation strata, black are syn-deformational "growth" layers, and active axial surfaces are in green.

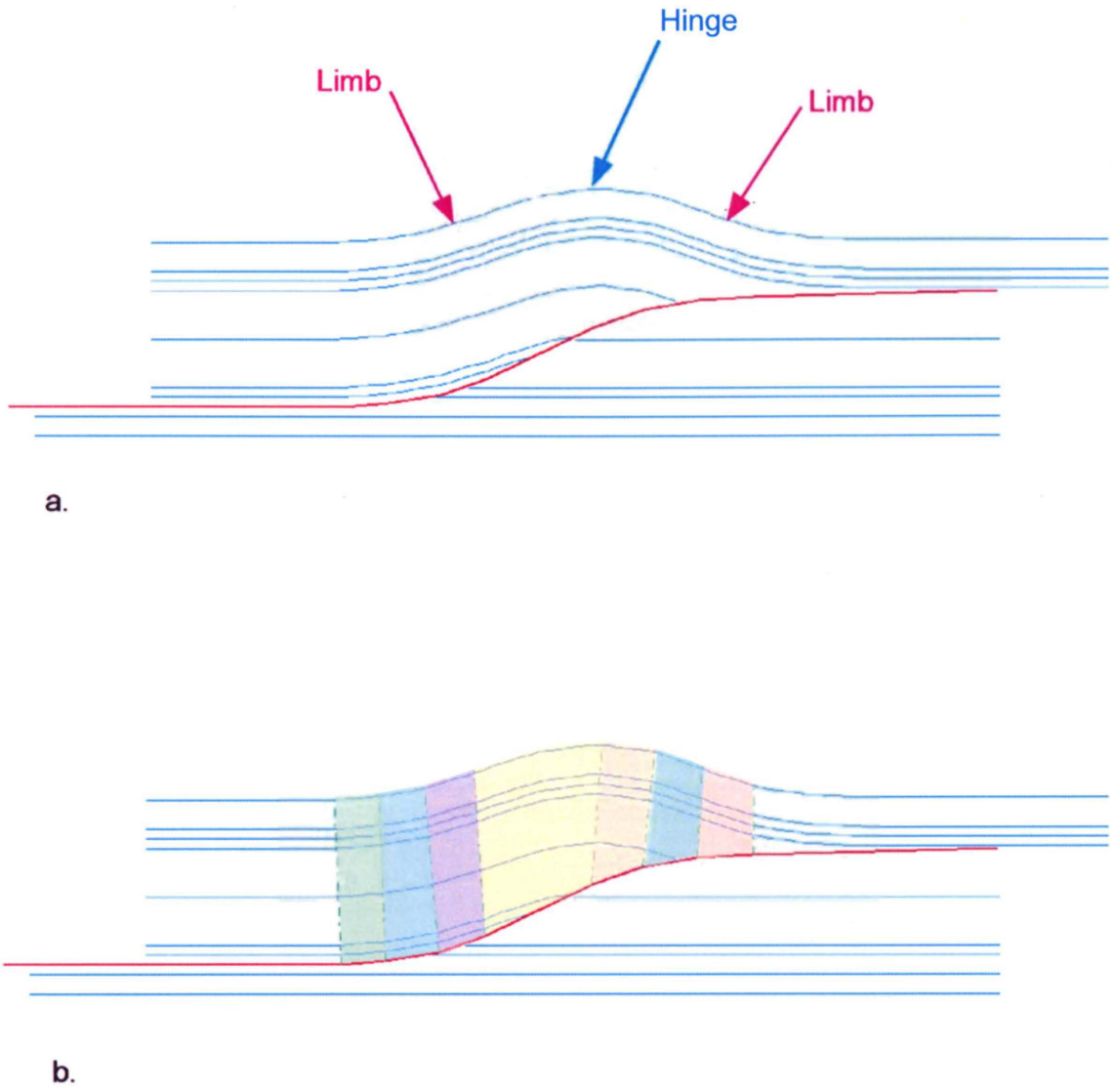
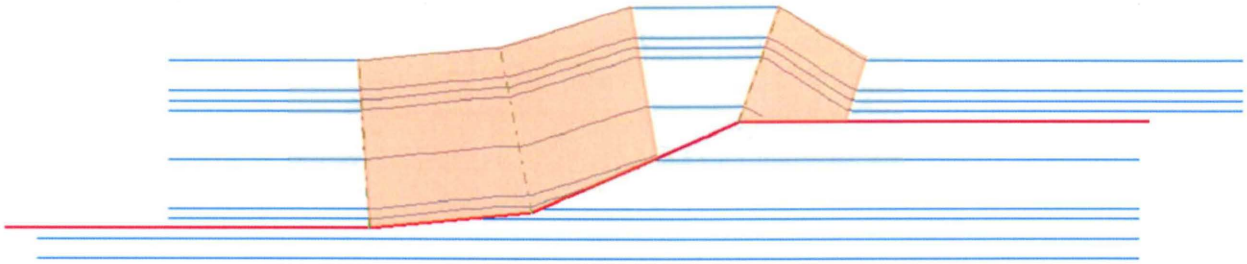


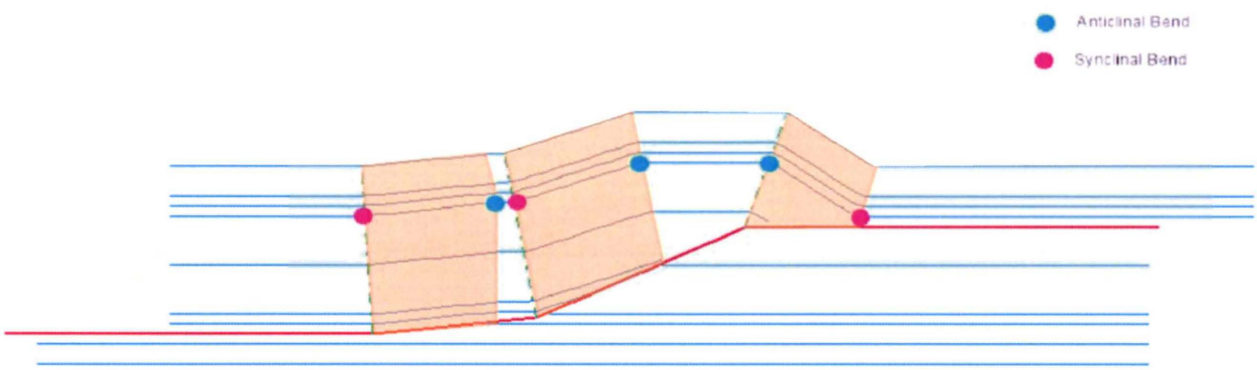
Figure 2.

a. Folds can be described as dipping limbs divided by a hinge region where dip changes quickly.

b. Alternatively, folds can be described as dip domains, regions of similar dip, bounded by axial surfaces. Axial surfaces are green dashed lines, each different colored shaded region is a dip domain.



a.

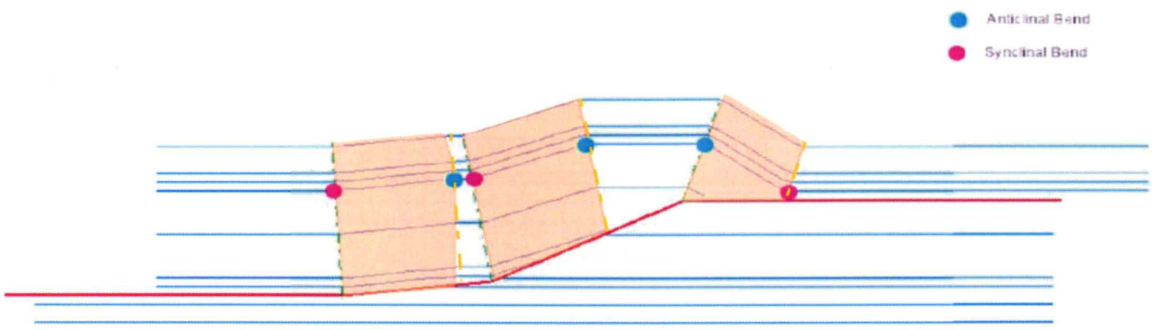


b.

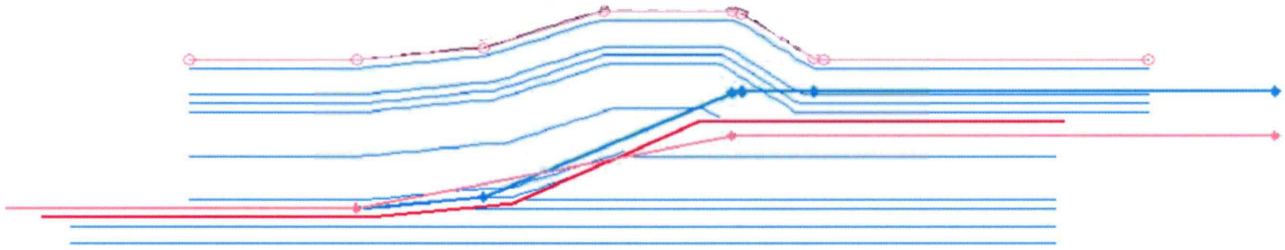
Figure 3. Describing dip domains.

a. Dip domains (peach-colored shaded regions) are bounded by axial surfaces that dip the same direction.

b. Dip domains are bounded by synclinal (magenta points) and anticlinal (blue points) bends.



a.

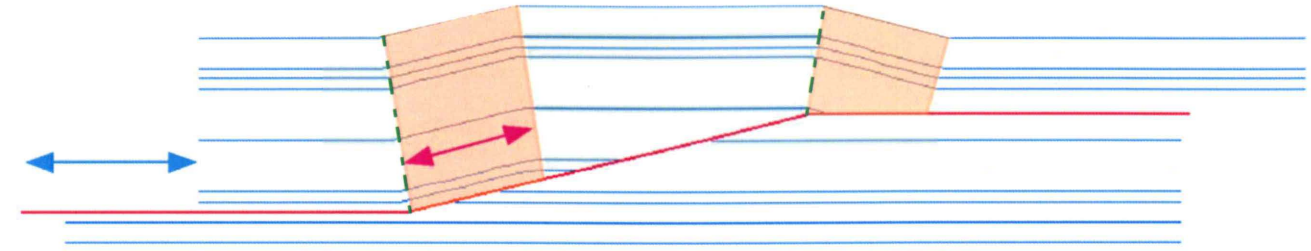


b.

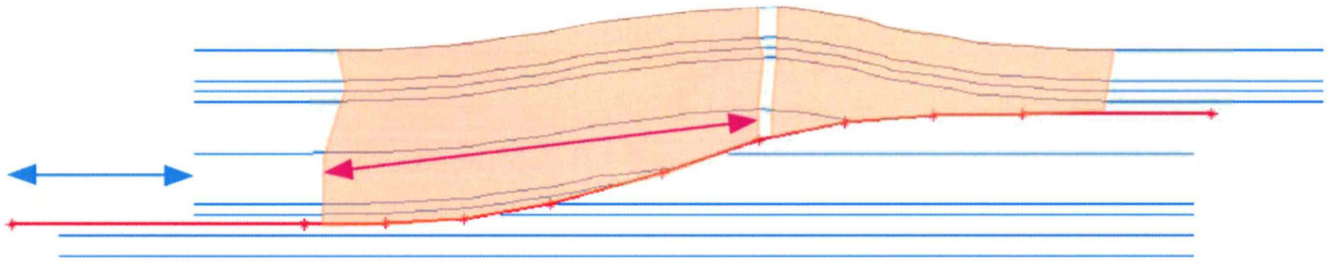
Figure 4. Estimating the location of fault bends from dip domains.

a. Dip domains including all segments bounded by synclinal to anticlinal hinges and anticlinal to synclinal hinges. Each leading synclinal axis in the fold corresponds to a synclinal bend in the fault, and each leading anticlinal axis in the fold corresponds to an anticlinal bend in the fault.

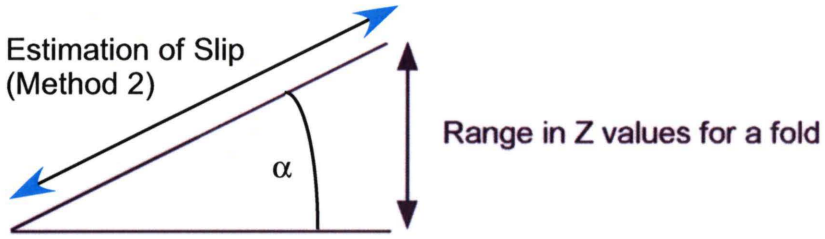
b. Dip domains including all segments bounded by axial surfaces that dip the same direction. Active axial surfaces are green, passive axial surfaces are in yellow. In generalized description of the fault, the leading active axial surfaces are correlated to bends in the fault; the fault is approximated as shown in purple. In the more detailed description of the fault, all intermediate bends are correlated with bends in the fault; fault approximation is in aqua.



a.



b.



c.

Figure 5. Estimation of input slip.

a. In cases of one fault bend, the maximum input slip is approximated as the 1.5 times the length of the sloping back limb (magenta), actual input slip is shown by the blue arrow.

b. In multibend and rounded fault cases, the length determined by the above method far overestimates the slip;

c. Slip is therefore calculated as the minimum of this value (in b.) and the hypotenuse of the triangle formed by the range in Z values for a layer and the maximum positive slope (α) of any segment.

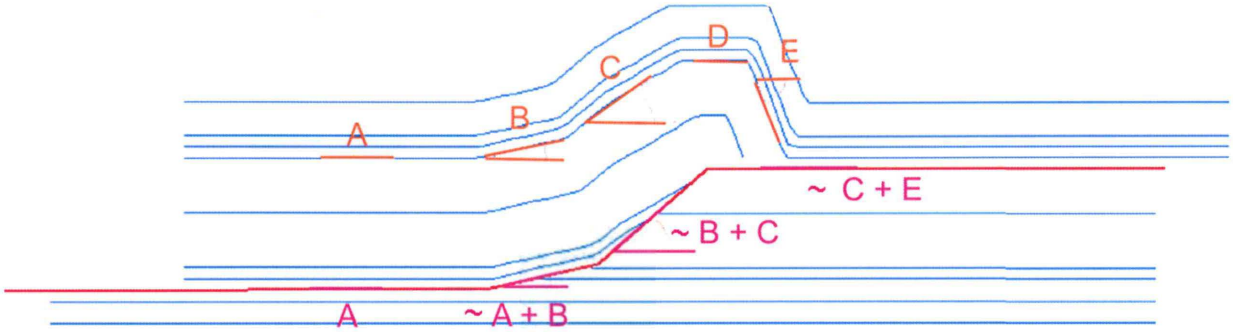


Figure 6. Estimation of fault segment slopes

The slope of the first fault segment is estimated as the slope of the first fold segment.

The slope of subsequent fault segments are estimated as the sum of the slopes of the current and previous fold segment.

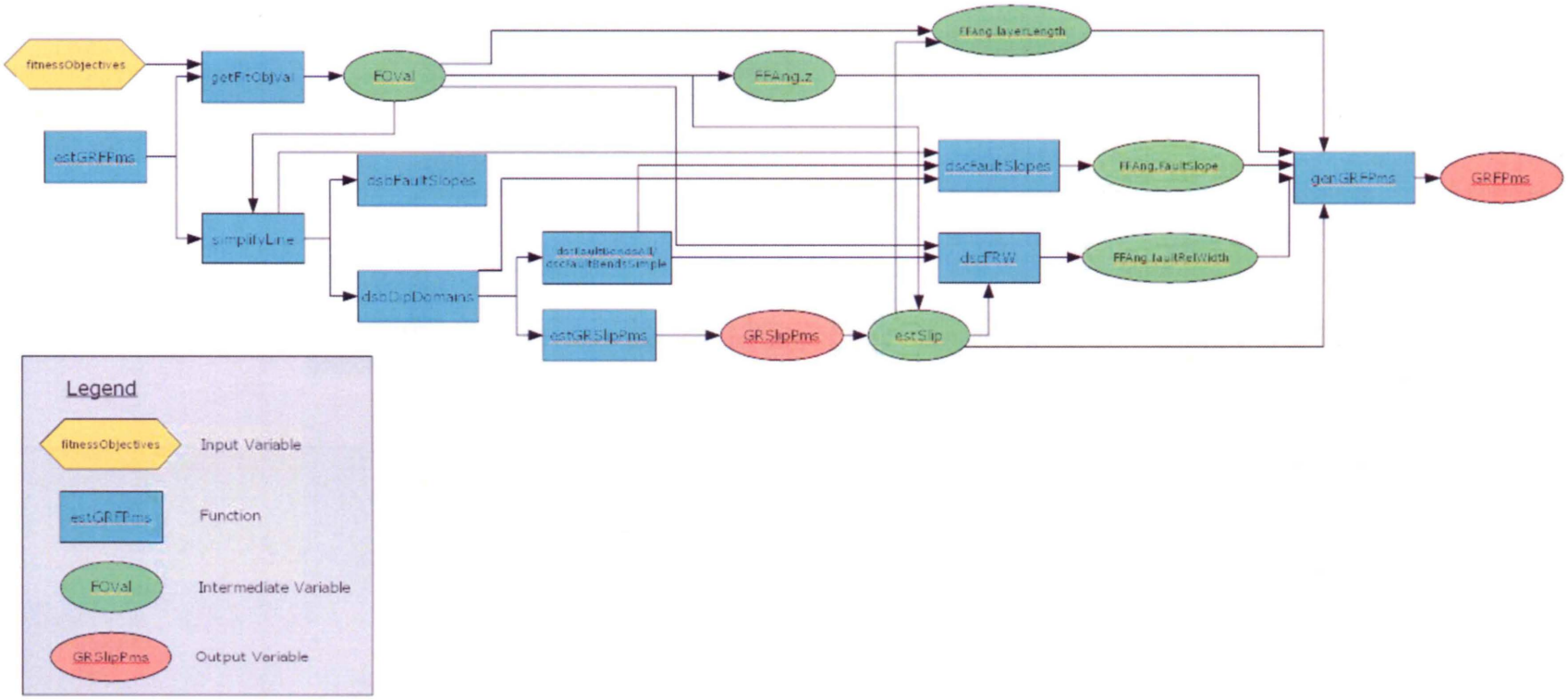


Figure 7. Program Map for estGRFPms. Intermediate variables are only shown for variables generated within the function estGRFPms and not those generated by subroutines executed within estGRFPms; other intermediate variables exist, but are omitted for clarity of presentation--arrows represent where an intermediate product of one function is an input to a subsequent function.

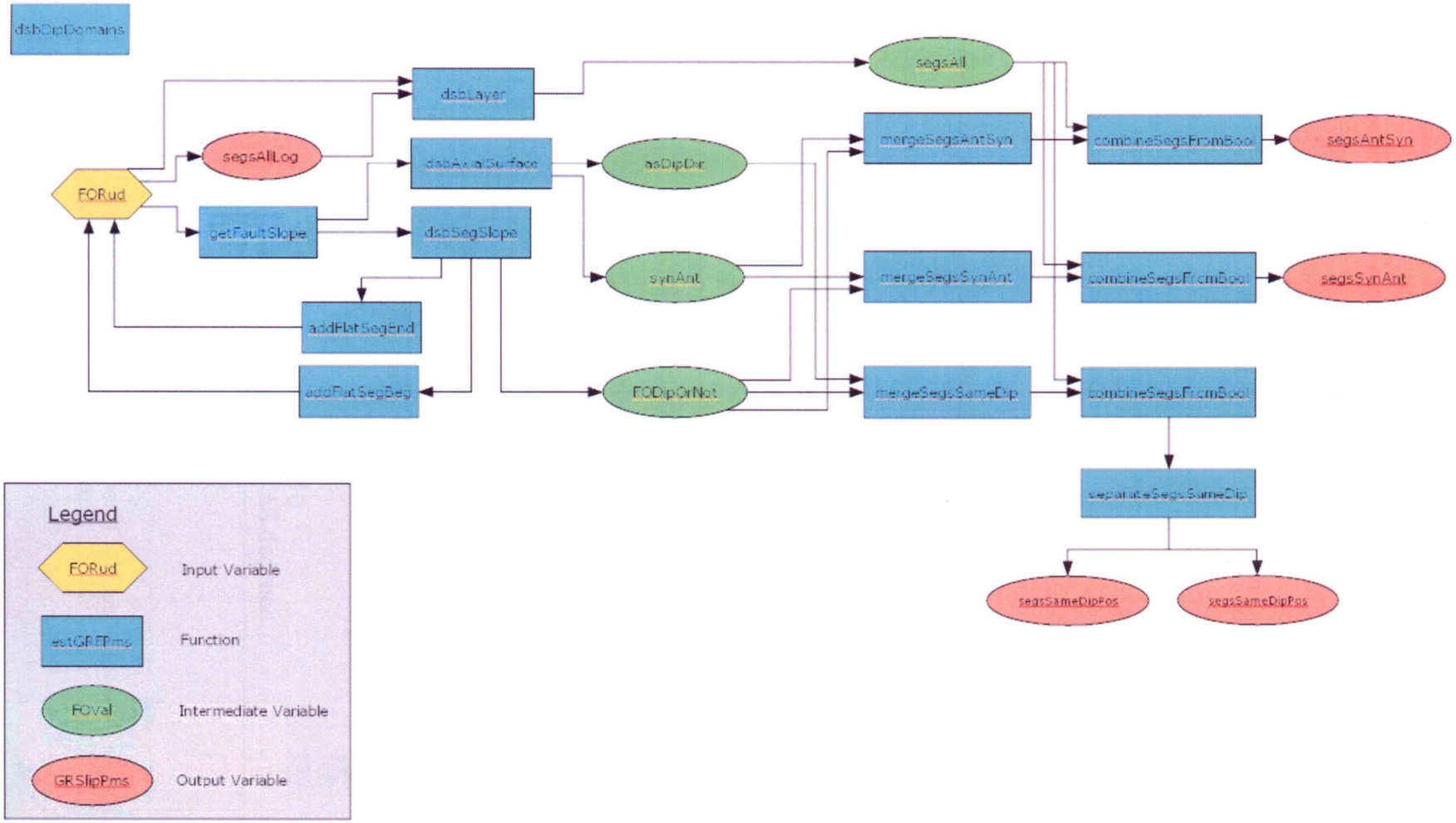
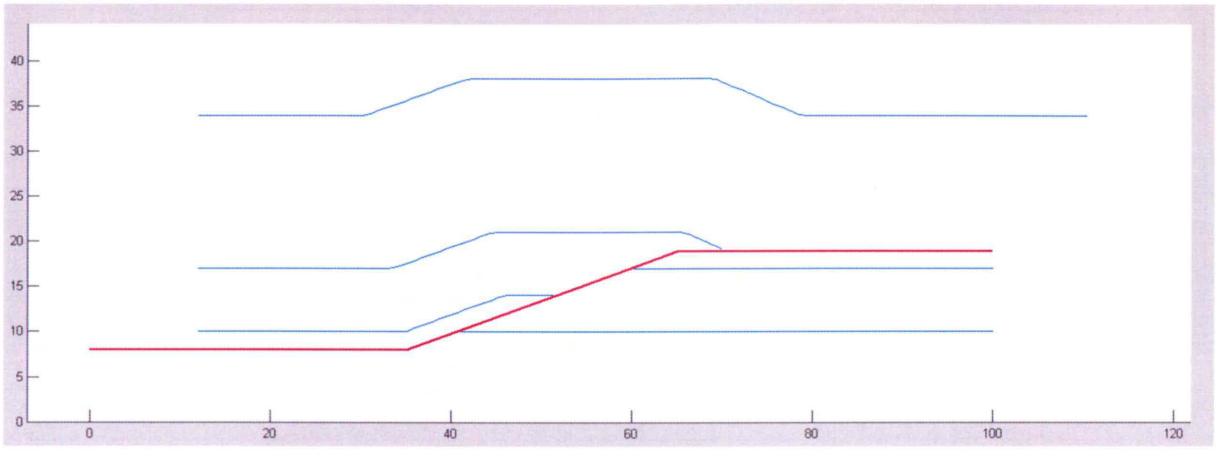


Figure 8. Program map for dsbDipDomains, which executes within the function estGRFPms.



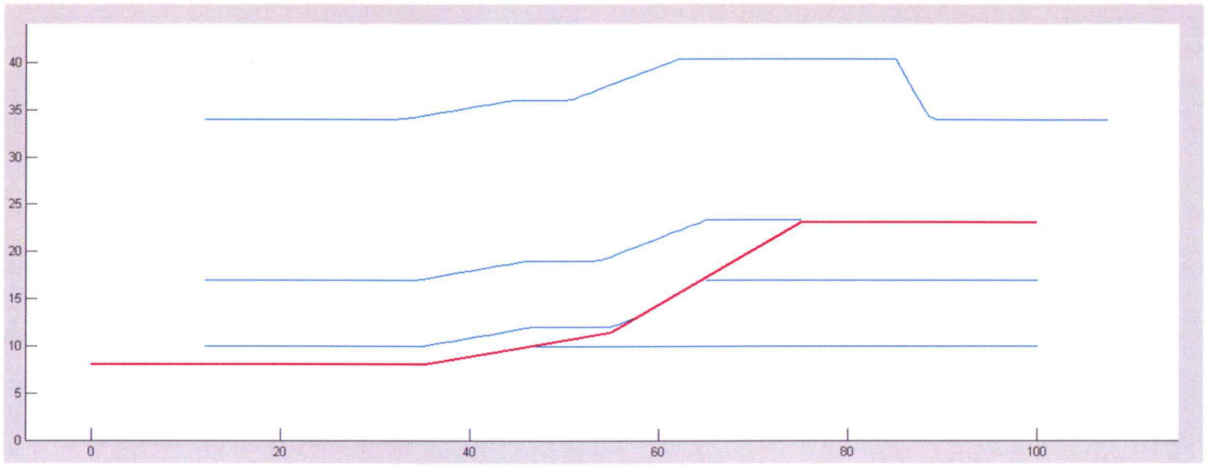
Known fault

Initial X:	0
Initial Z:	8
No. of Segments:	3
Slopes:	0, 20, 0 degrees
Fault relative width:	35, 30, 35

Fitness Objectives (fold)

	Layers		
	Bottom	Middle	Top
Initial X:	12	12	12
Initial Z:	10	17	34
Z Range:	4.10		
Maximum Slope:	21.56 degrees		

Figure 9. Simple flat-ramp-flat fault-bend fold used for testing, with properties of the known fault and fitness objectives given.



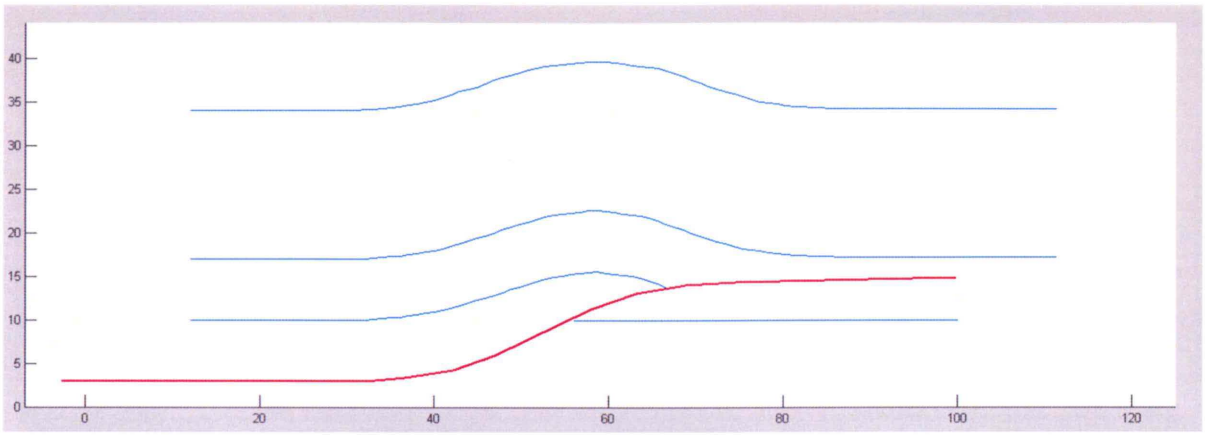
Known fault

Initial X:	0
Initial Z:	8
No. of Segments:	4
Slopes:	0, 10, 30, 0 degrees
Fault relative width:	35, 20, 20, 25

Fitness Objectives (fold)

	Layers		
	Bottom	Middle	Top
Initial X:	12	12	12
Initial Z:	10	17	34
Z Range:	6.37		
Maximum Slope:	22.54 degrees		

Figure 10. Two bend fault-bend fold used for testing, with properties of the known fault and fitness objectives given.



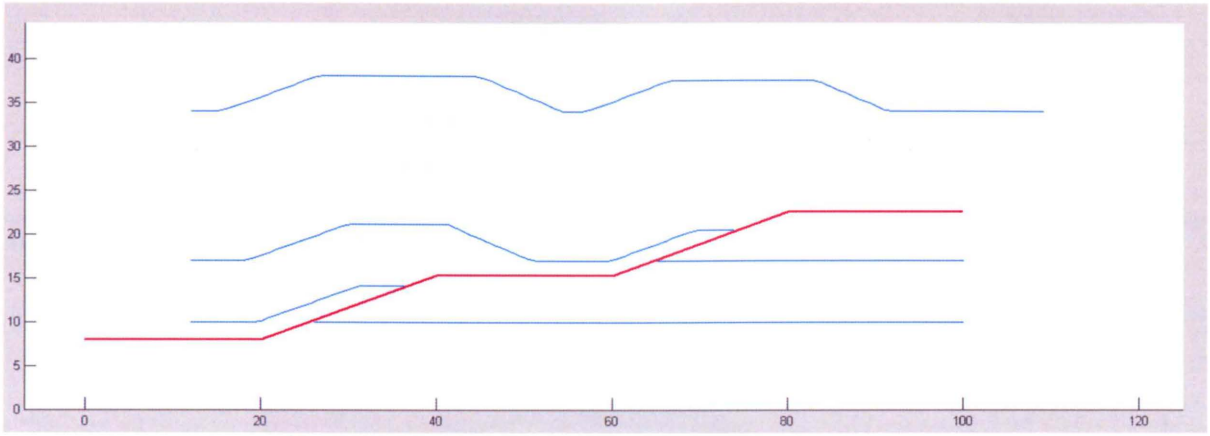
Known fault

Initial X:	0
Initial Z:	8
No. of Segments:	9
Slopes:	0, 4.57, 9.46, 19.09, 25.80, 19.65, 9.16, 3.69, 1.25 degrees
Fault relative width:	34.74, 4.60, 5.51, 4.78, 11.03, 5.15, 5.70, 5.70, 25.37

Fitness Objectives (fold)

	Layers		
	Bottom	Middle	Top
Initial X:	12	12	12
Initial Z:	10	17	34
Z Range:	5.62		
Maximum Slope:	22.87 degrees		

Figure 11. Rounded fault-bend fold used for testing, with properties of the known fault and fitness objectives given.



Known fault

Initial X:	0
Initial Z:	8
No. of Segments:	5
Slopes:	0, 20, 0, 20, 0 degrees
Fault relative width:	20, 20, 20, 20, 20

Fitness Objectives (fold)

	Layers		
	Bottom	Middle	Top
Initial X:	12	12	12
Initial Z:	10	17	34
Z Range:	4.10		
Maximum Slope:	21.56 degrees		

Figure 12. Two flat-ramp-flat composite fault-bend fold used for testing, with properties of the known fault and fitness objectives given.

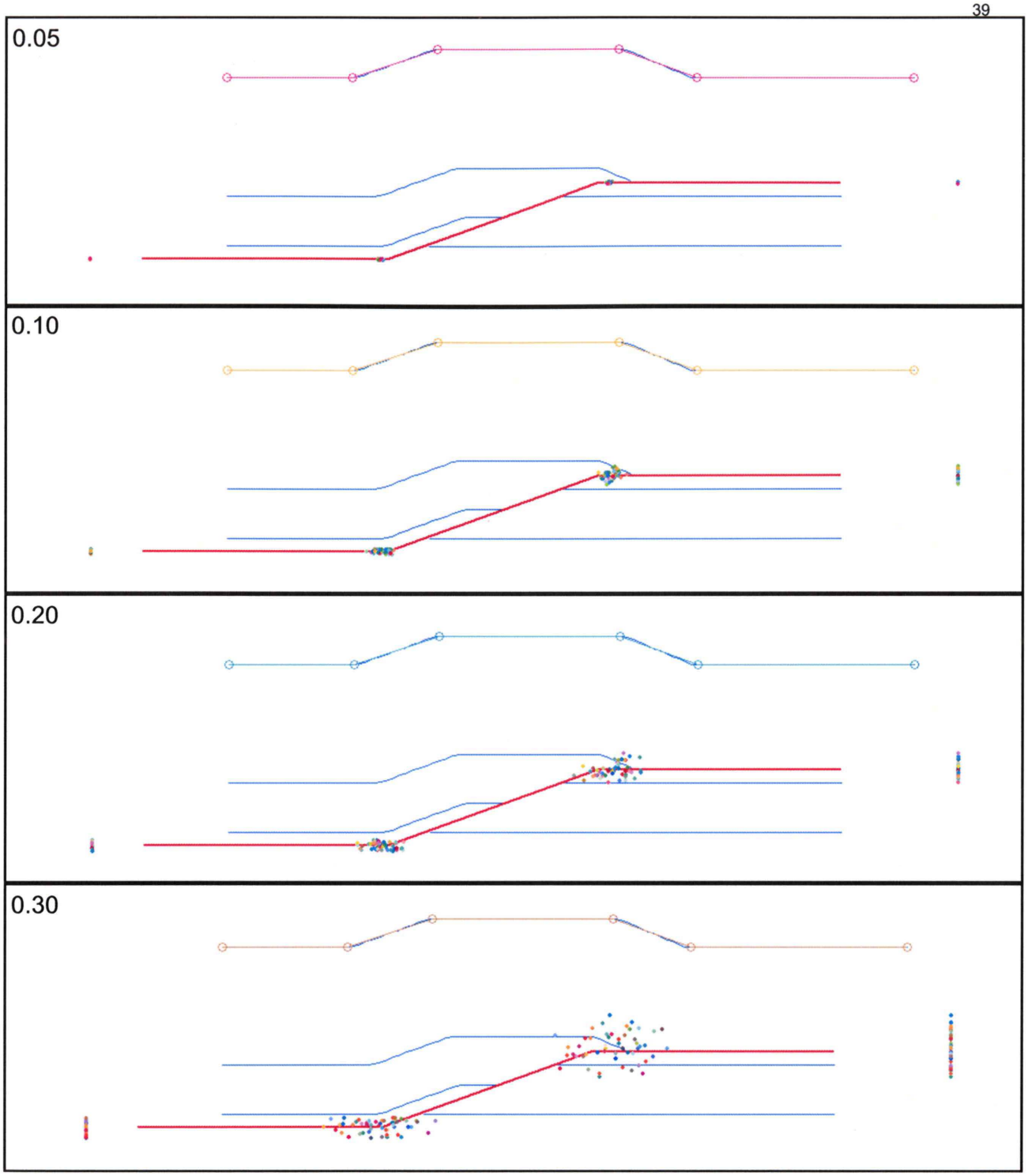


Figure 13. Variation in the range from which the random fault may be generated (GRFPms range). Known fault in red, known fold (fitness objectives) in blue. Line simplification cutoff is 0.3, simplified line is colored line with open circles at vertices. Fault parameter range values are given in the upper left of each figure, and vertices of the estimated fault are plotted as multicolor points. As range increases, the range of randomly generated fault vertices increases its spread. Because the final and initial points X values are fixed, note that variation in these points is only in the Z direction. Additionally, note that variation in the first bend's location is elongated in the X direction.

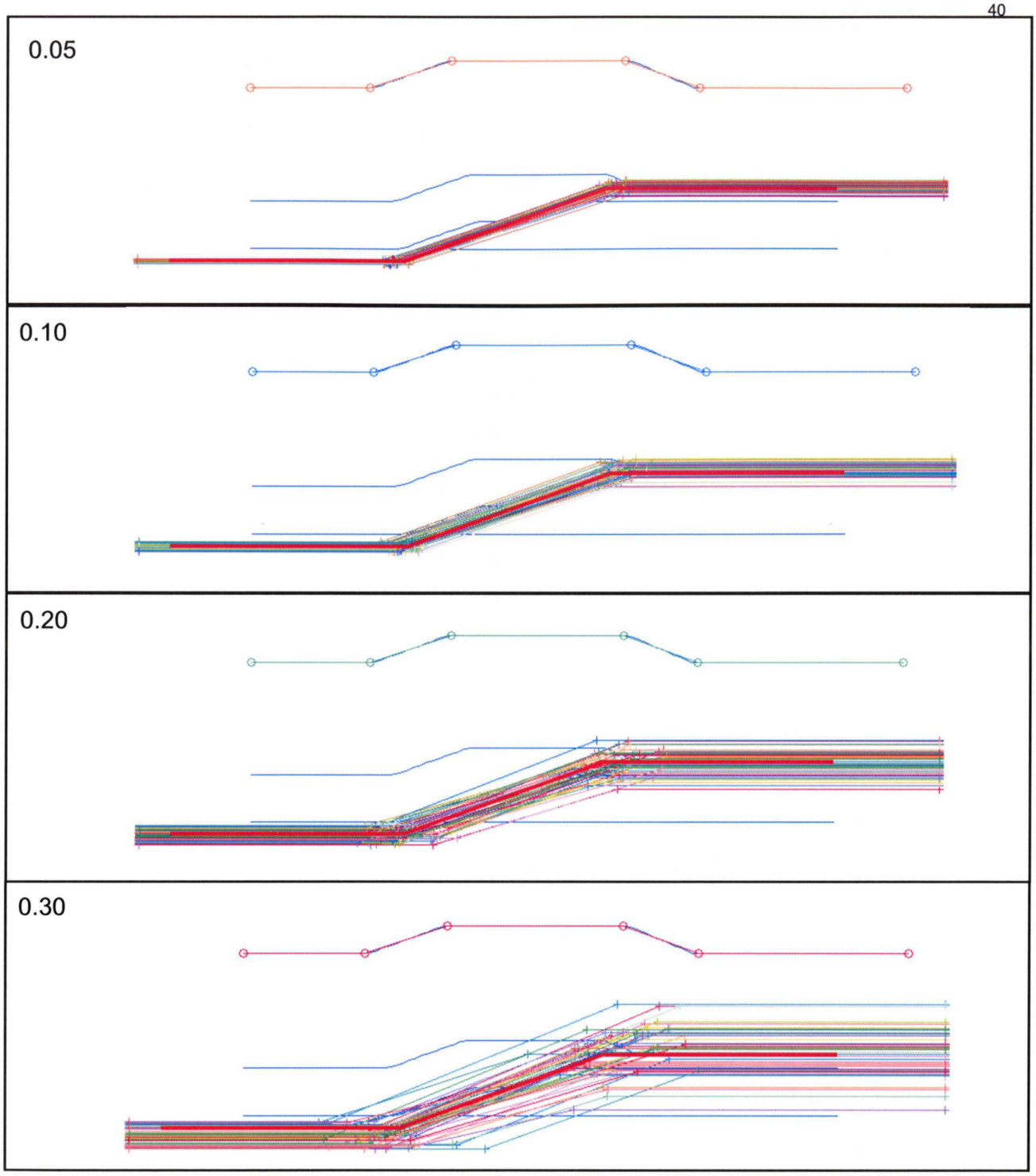


Figure 14. Varying fault parameter range for a fault with one 20 degree ramp. Fault parameter range value is in the upper left for each figure.

Simplification cutoff: 0.30

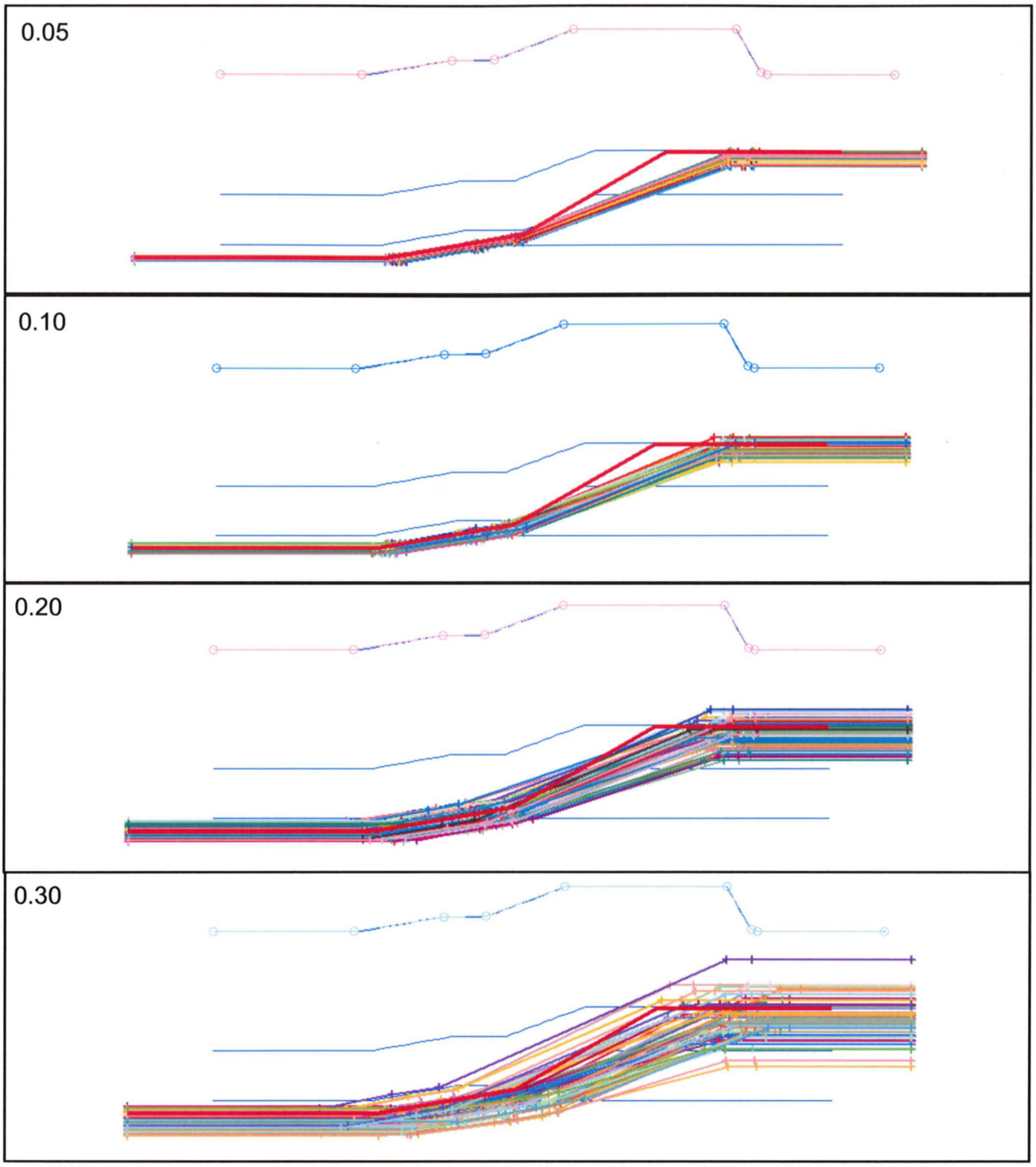


Figure 15. Varying fault parameter range for a fault with two bends, fault ramp segments sloping 10 and 30 degrees. Fault parameter range value is in the upper left for each figure.

Simplification cutoff: 0.07

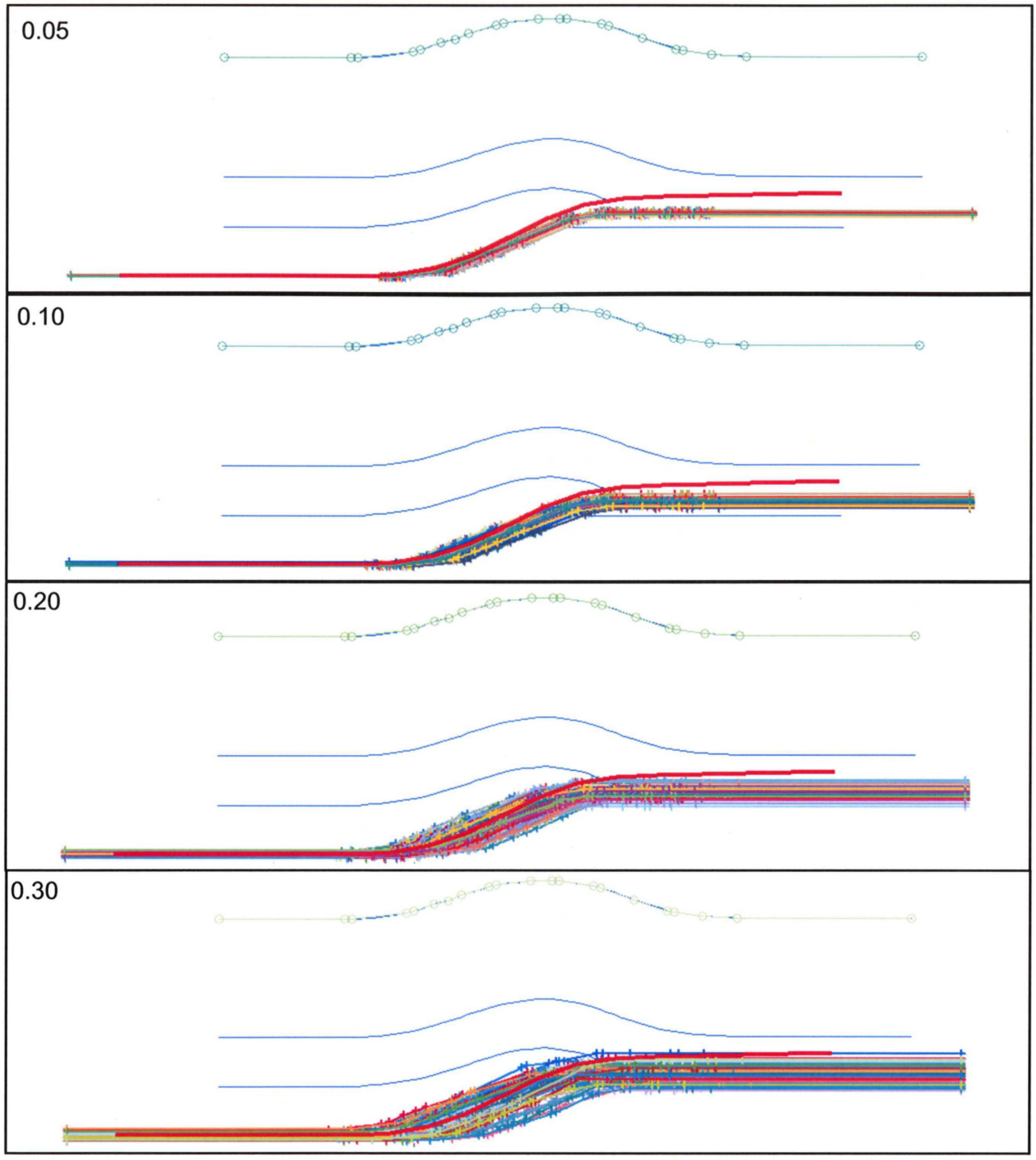


Figure 16. Varying fault parameter range for a rounded fault-bend fold. Fault parameter range value is in the upper left for each figure.

Simplification cutoff: 0.07

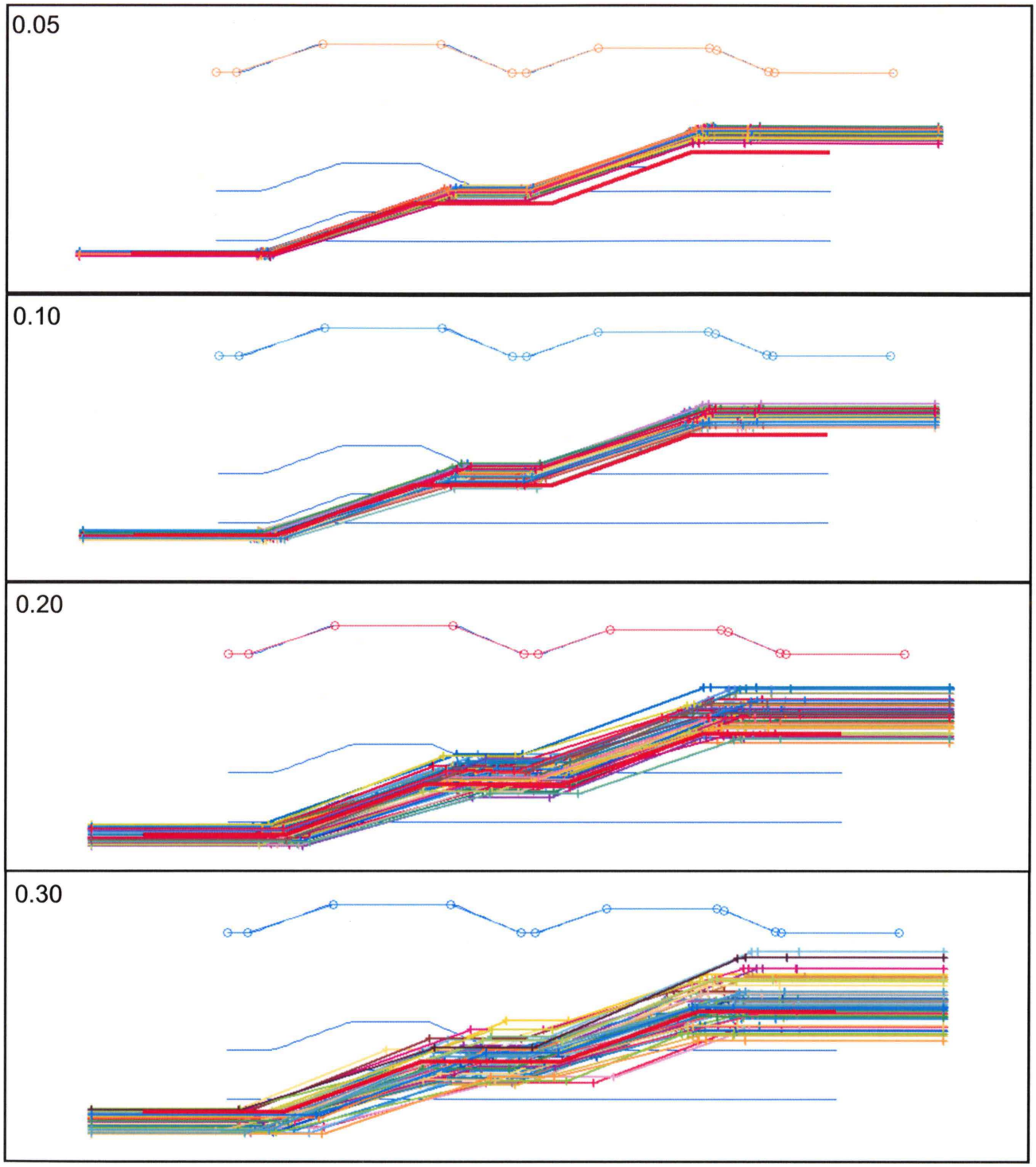


Figure 17. Varying fault parameter range for a two flat-ramp-flat fault-bend fold. Fault parameter range value is in the upper left for each figure.

Simplification cutoff: 0.20

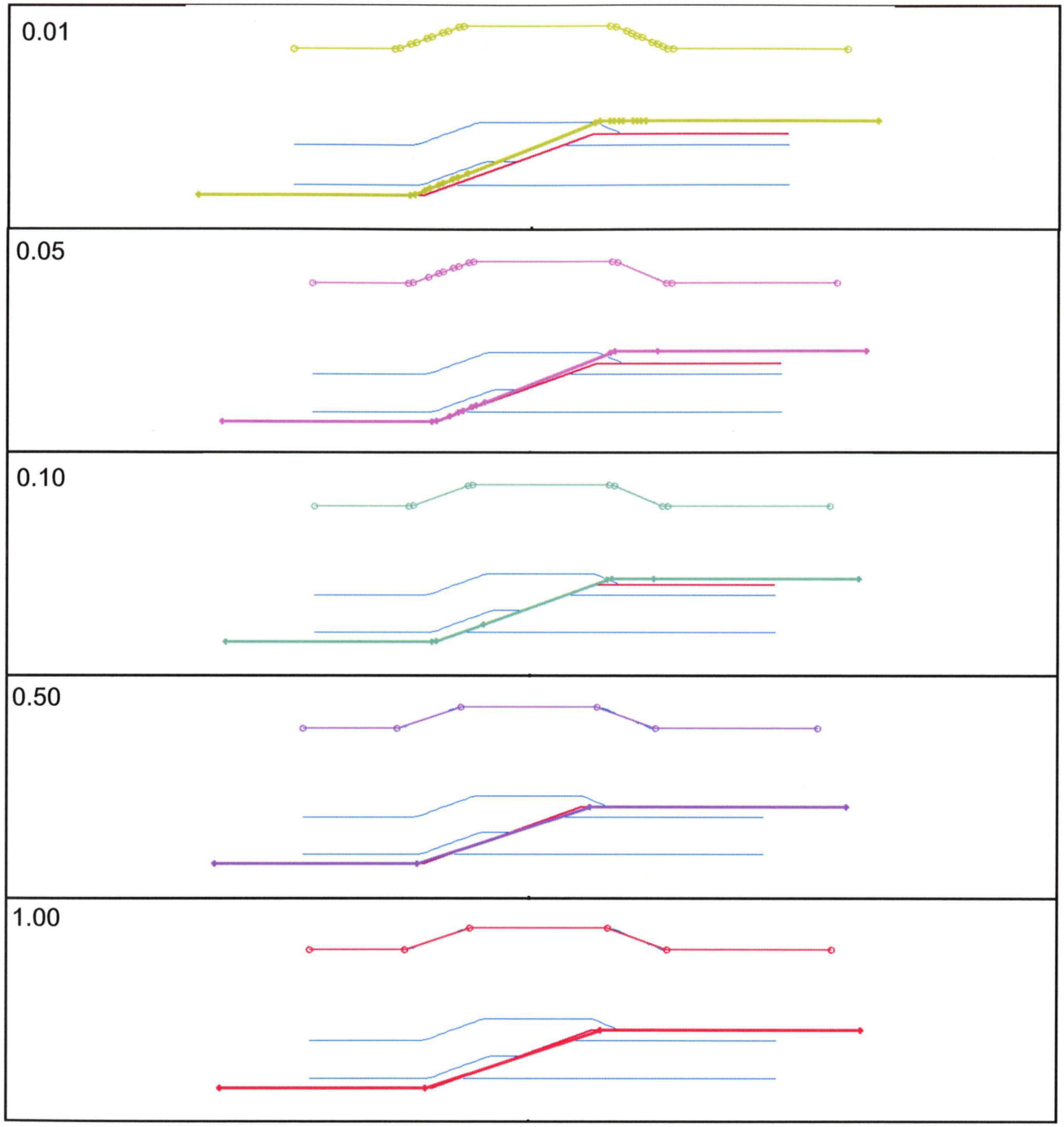


Figure 18. Variation in simplification cutoff for a flat-ramp-flat fault-bend fold. Ramp is 20 degrees. Line simplification cutoff value is in the top left of each figure.

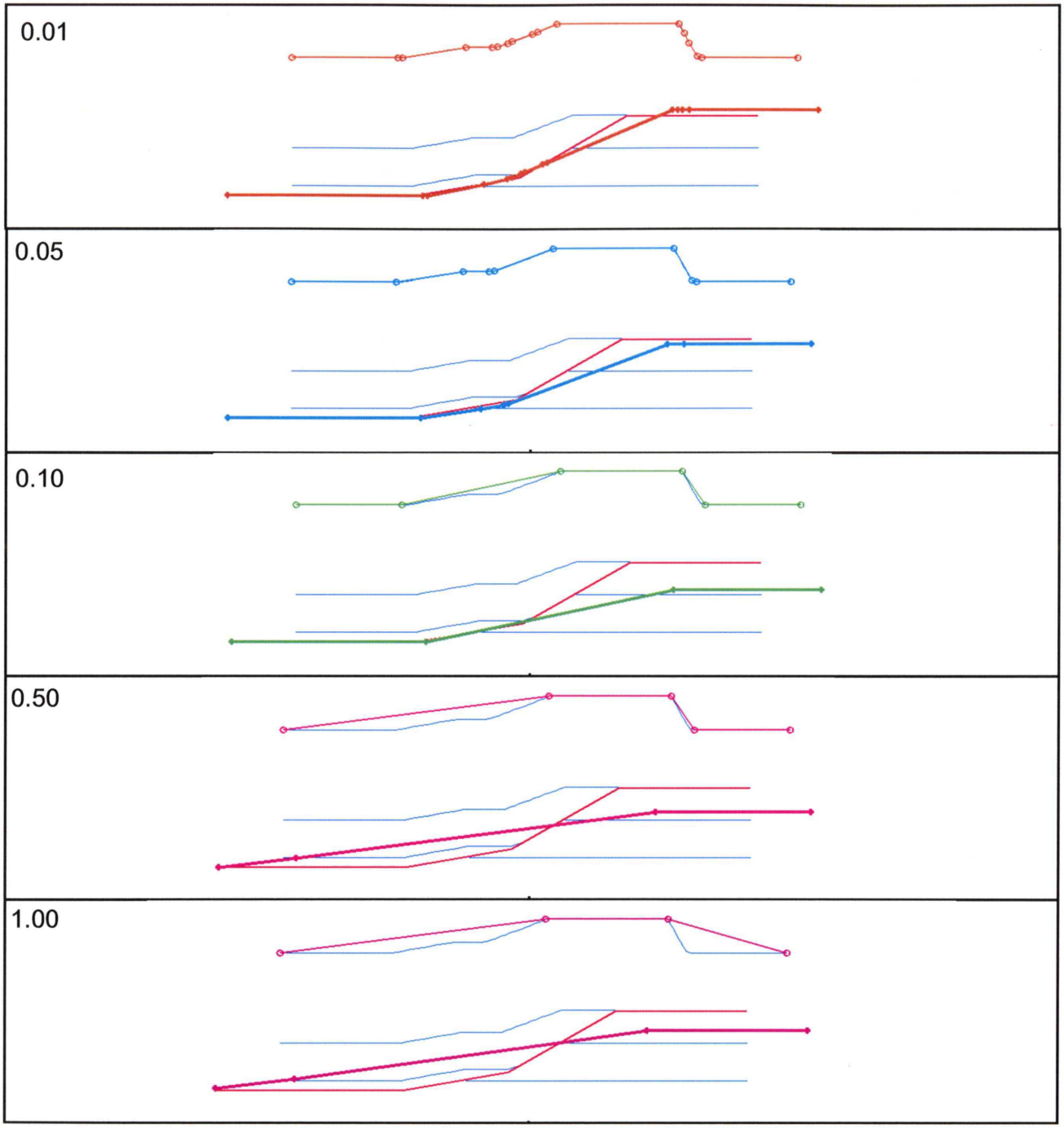


Figure 19. Variation in simplification cutoff for a two-bend fault-bend fold. Ramp angles are 10 and 30 degrees. Line simplification cutoff value is in the top left of each figure.

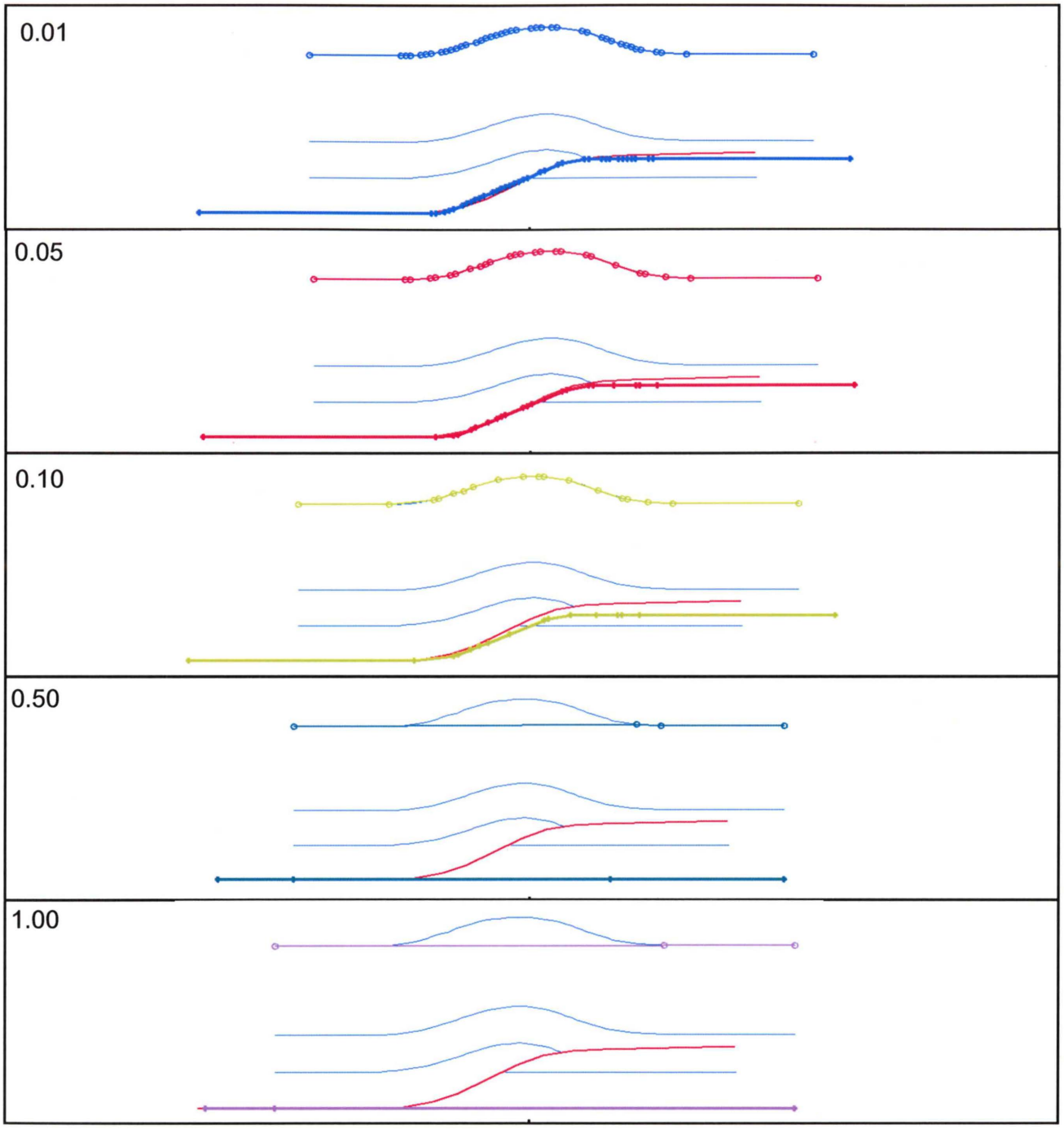


Figure 20. Variation in simplification cutoff for a rounded fault-bend fold. Line simplification cutoff value is in the top left of each figure.

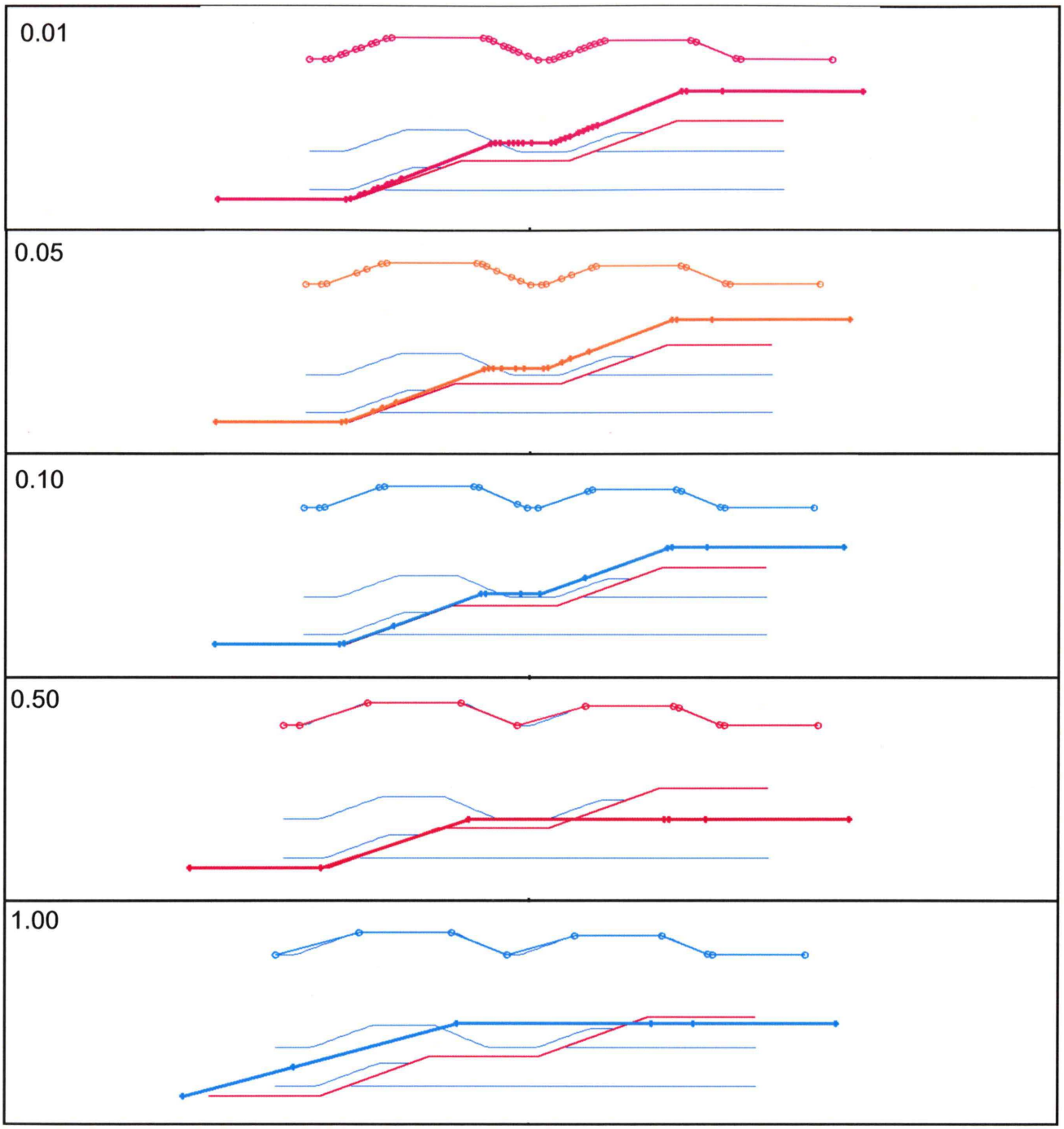


Figure 21. Variation in simplification cutoff for a two ramp fault-bend fold. Ramps are both 20 degrees. Line simplification cutoff value is in the top left of each figure.

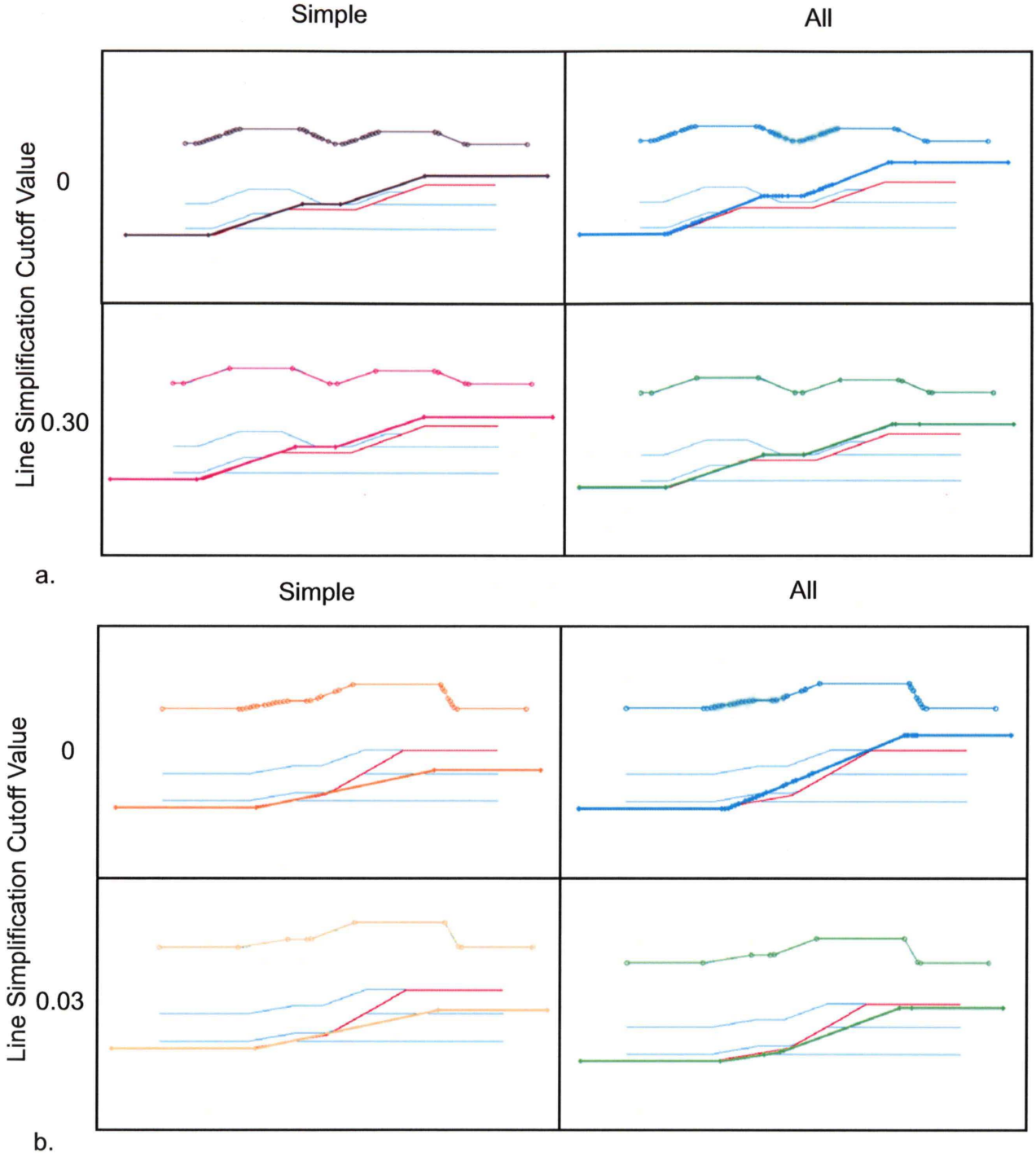


Figure 22. Use of 'simple' or 'all' to predict fault bends as a function of simplification cutoff for the fold.

a. Two ramps: If the fold is under-simplified, 'simple' is a better approximation (top). If fold is simplified at the right threshold, the distinction does not affect the shape of the predict ed fault, although the number of points still varies(bottom).

b. Two bends: If the fold is undersimplified, 'all' creates too steep of a fault with many bends, 'simple' creates a fault that is too gently dipping and does not capture the second bend (top). When the fault is simplified at a better threshold (bottom), 'all' approximates the fault better than 'simple.'

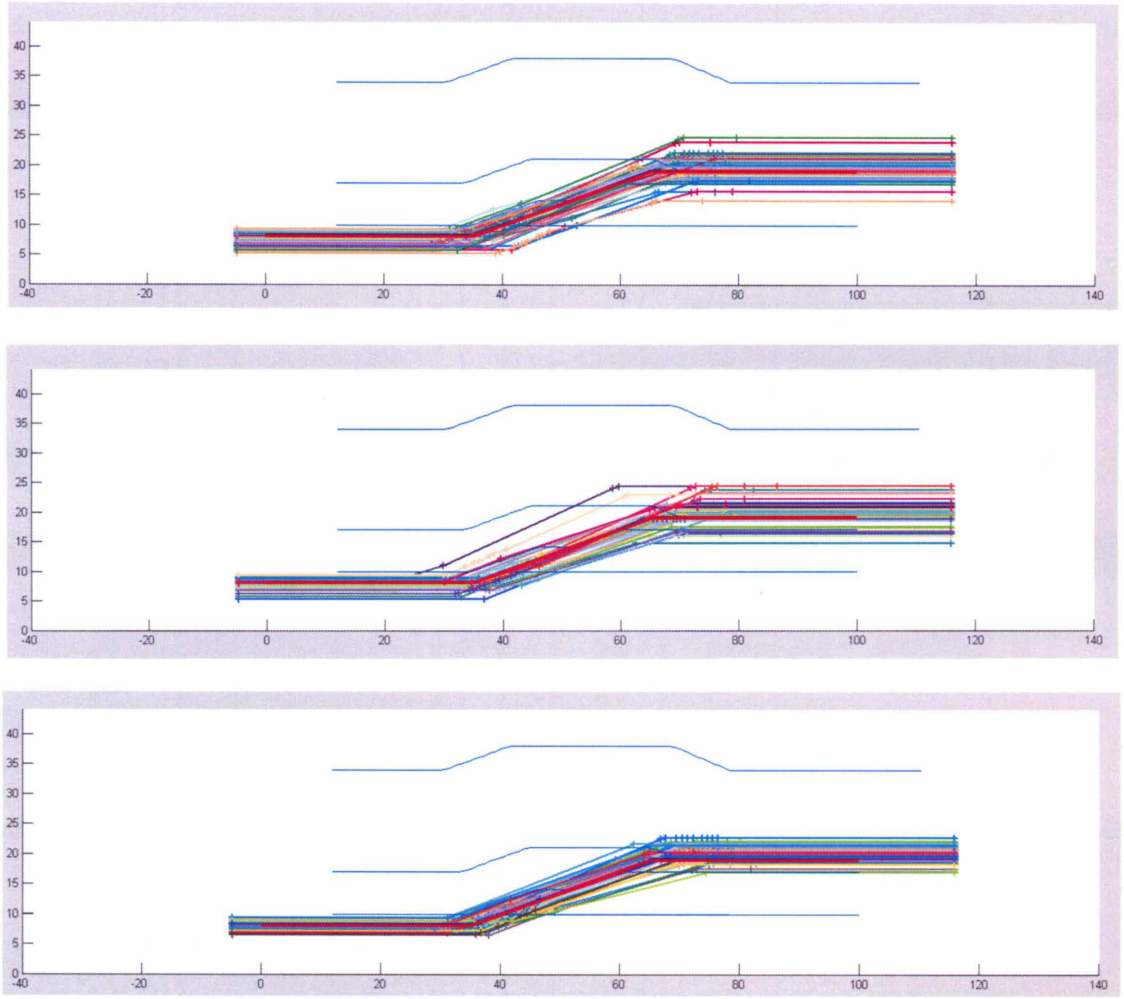


Figure 23. Result of varying all values and parameters, using default ranges for a flat-ramp-flat fault-bend fold. Randomly generated populations of faults may look somewhat different, although the default ranges remain the same (top and bottom). Default Ranges: Fold Simplification Range: [0.01 0.4]; Parameters Range: [0 0.3]; 100% 'all'; 50 faults in the population.

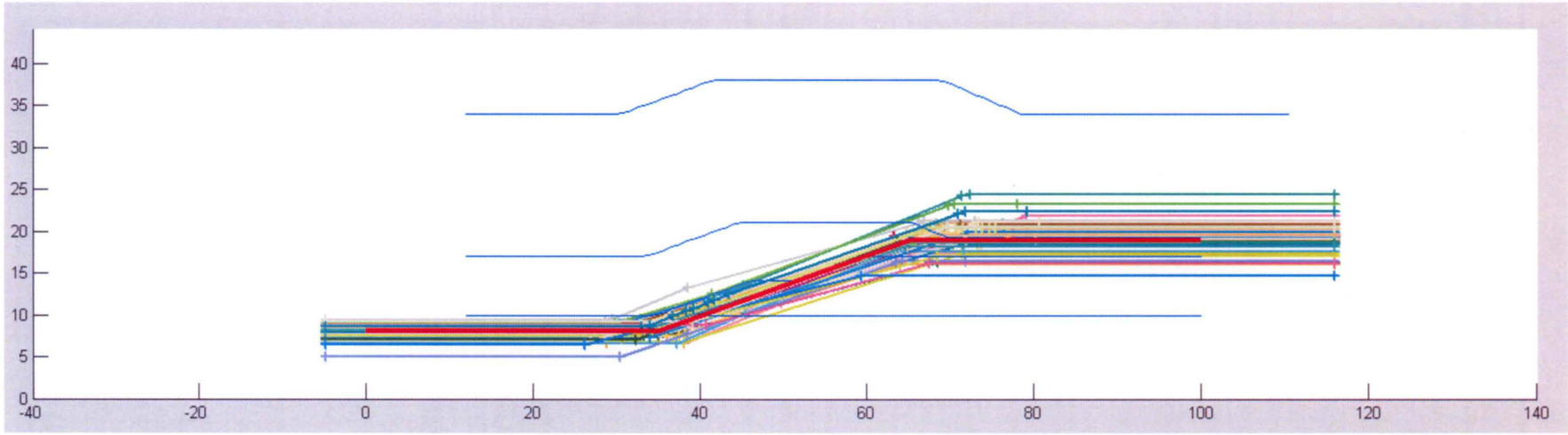


Figure 24. Result of varying all values and parameters for a flat-ramp-flat fault-bend fold, using ranges specified.

Ranges:

Fold Simplification Range: [0.01 0.4];
Parameters Range: [0 0.3];
50% 'all', 50% 'simple;
50 faults in the population.

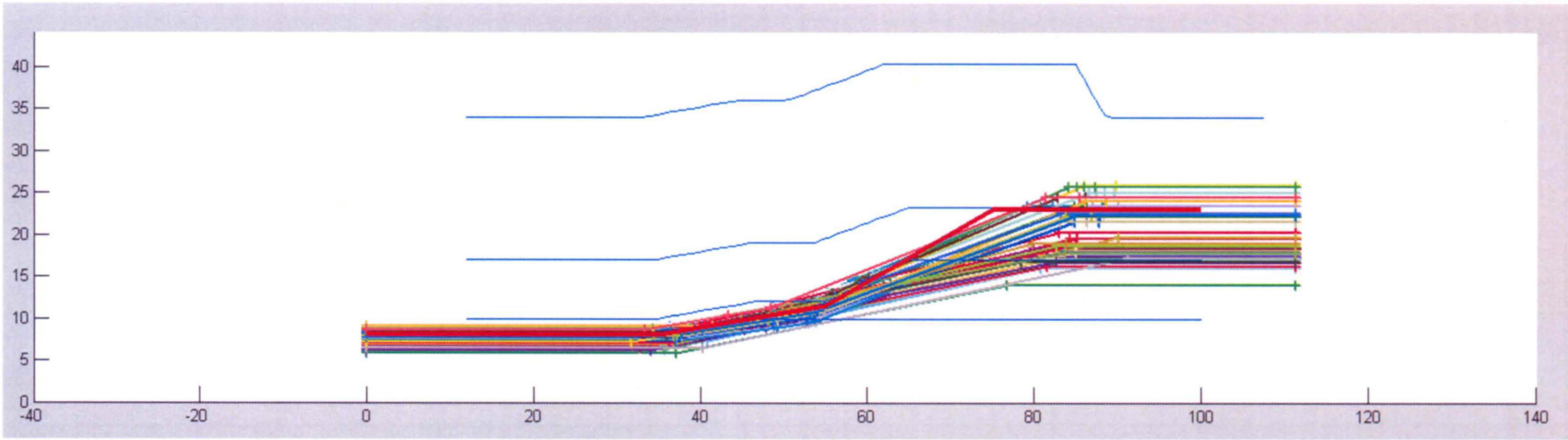


Figure 25. Result of varying all values and parameters for a two-bend fault-bend fold, using the ranges specified below. Note that the Fold Simplification Range has been reduced from other tests, but all other ranges remain the same.

Ranges:

Fold Simplification Range: [0.01 0.1];
Parameters Range: [0 0.3];
50% 'all', 50% 'simple';
50 faults in the population.

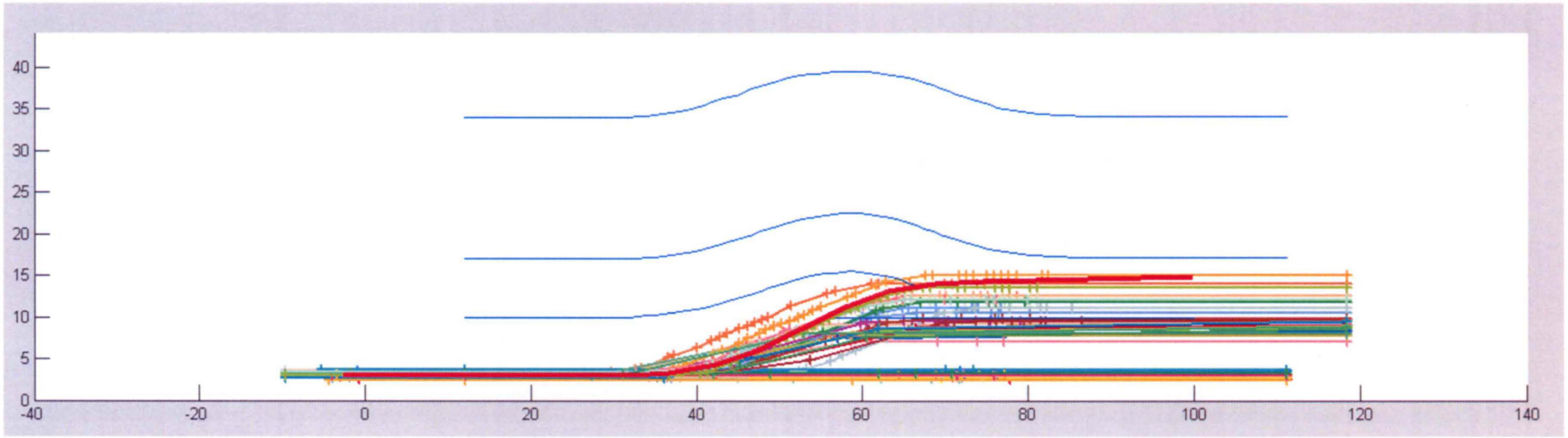


Figure 26. Result of varying all values and parameters for a rounded fault-bend fold, using the ranges specified below.

Ranges:

Fold Simplification Range: [0.01 0.4];
Parameters Range: [0 0.3];
50% 'all', 50% 'simple;
50 faults in the population.

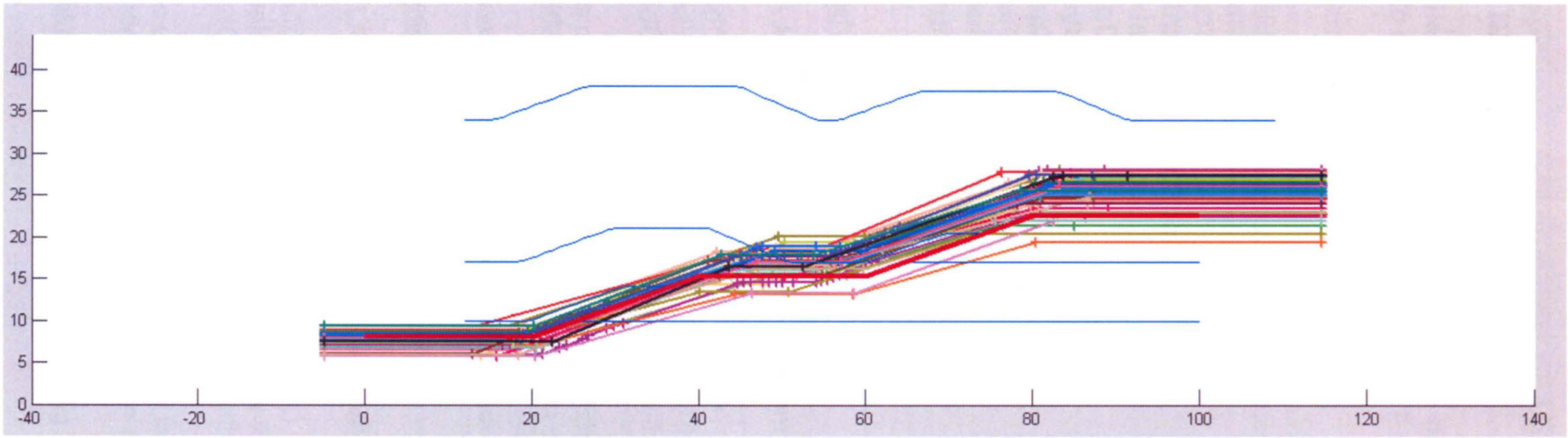


Figure 27. Result of varying all values and parameters for a two flat-ramp-flat composite fault-bend fold, using ranges specified.

Ranges:

Fold Simplification Range: [0.01 0.4];
Parameters Range: [0 0.3];
50% 'all', 50% 'simple;
50 faults in the population.

XI. APPENDIX: DESCRIPTION OF THE ROLES, INPUTS, AND OUTPUTS OF INDIVIDUAL FUNCTIONS

Help files for all functions written in the completion of this toolbox. Please refer to the program maps (Figures 7 and 8).

Program Index:

addFlatSegBeg

ADDFLATSEGBEG adds a flat segment 1 unit long to the beginning of a line

FOrudOut = addFlatSegBeg(FOrudIn)
FOrudIn -- any n x 3 matrix that describes a line/polyline as X-Y-Z
FOrudOut -- the same, but with a 1 unit long flat seg added to the front

this is useful so that the properties of the first real segment can be considered when analyzing the fold in estGRFPms

see also estGRFPms, addFlatSegEnd

addFlatSegEnd

ADDFLATSEGEND adds a flat segment 1 unit long to the end of a line

FOrudOut = addFlatSegEnd(FOrudIn)
FOrudIn -- any n x 3 matrix that describes a line/polyline as X-Y-Z
FOrudOut -- the same, but with a 1 unit long flat seg added to the front

this is useful so that the properties of the last real segment can be considered when analyzing the fold in estGRFPms

see also estGRFPms, addFlatSegBeg

combineSegsFromBool

COMBINESEGSFROMBOOL takes use and append vectors and creates polylines

segsArrayFinal = combineSegsFromBool(segsAll, use, append) takes use and append boolean vectors and creates a cell array of polylines of the dipdomains as specified by the booleans

segsAll	-- an n X 1 cell array 2 X 3 matrices that describe the beginning and end point of each individual segment in the FOrud fold
use	-- whether to use (1) a segment in generating a dip domain or not (0)
append	-- whether it should be appended to a previous segment as part of apolyline (1) or not (0)
segsArrayFinal	-- a cell array of matrices that give the points in the polylines that describe the dip domains as prescribed by the input use and append booleans.

use and append created by mergeSegsSynAnt,mergeSegsAntSyn, and mergeSegsSameDip

also see dsbDipDomains

dsbAxialSurface

DSBAXIALSURFACE describe each hinge as syn/ant and dip of axial trace

[synant, asDipDir] = dsbAxialSurface(FOSlope) takes input slopes of the segments of a polyline and describes each hinge as synclinal/anticlinal and the dip of the axial trace at that hinge

FOSlope	-- a column vector of the slope values for the segments of FORud, a polyline of a simplified fold shape
synant	-- a boolean vector indicating for each point in the polyline whether the bend is synclinal (1) or anticlinal (0)
asDipDir	-- a boolean vector indicating whether the axial trace at a bend is positive sloping (1) or negative sloping (0)

see also dsbSegSlope, dsbDipDomains

dsbDipDomains

DSBDIPDOMAINS describes dip domains of input fold in different ways

function
[segsSynAnt,segsAntSyn,segsSameDipPos,segsSameDipNeg,segsAll]...
= dsbDipDomains(FOrud) describes dip domains as a series of structures of polylines based on different categorizations of dip domains

FOrud -- an n x 3 X-Y-Z point description of the simplified fold

outputs four logicals that are indices to the points in FOrud to use to consider each of the following as dip domains:

segsAll -- consider each segment its own dip domain
segsSameDip -- (same dip) consider all segments that have axial traces with the same dip direction as dip domains (separated into segs with negative slopes (segsSameDipNeg) and positive slopes, segsSameDipPos)
segsSynAnt -- (syncline/anticline bounded) coinsider multiple segments with the same axial trace dip direction that are bounded by a syncline/anticline hinge pair
segsAntSyn -- same as above, but for anticline/syncline-bounded segments

for how these logicals are generated, see mergeSegsSameDip, mergeSegsSynAnt, mergeSegsAntSyn, seperateSegsSameDip, and combineSegsFromBool

dsbLayer

DSBLAYER--takes logical and reduced line, creates cell array of segments

segsarray = dsbLayer(FOrud, segsLog)
from the segments specified in the segsLog, dsbLayer returns the x,y,z coordinates of segments in a cell array

FOrud -- an n x 3 matrix of X-Y-Z locations of the points in the reduced fold polyline
segsLog -- boolean column vector of which segments to use
segsarray -- contains n 2 x 3 vectors that specify the xyz coordinates of the beginning and end points of each segment

see also dsbDipDomains

dsbSegSlope

DSBSEGSLOPE determines whether segs are pos/neg dip and dipping or not

[FODipOrNot, FOSegSlopeSign] = dsbSegSlope(FOslope) creates boolean vectors that indicate whether each segment of the reduced fault is dipping or not and has positive or negative slope

FOslope	-- an nx1 matrix of the slopes of the segments that connect the points that define the reduced objective layer.
FODipOrNot	-- a logical where 1 is dipping, 0 is flat (+/- 2 degrees)
FOSegSlopeSign	-- is a logical where 1 is positive slope, 0 is flat or negative slope

see also dsbAxialSurface, dsbDipDomains

dscFaultBendsAll

DSCFAULTBENDSALL fault bends chosen by all points within fold dip domains

[cvexBends, ccaveBends] = dscFaultBendsAll(segs1,segs2);
finds the number of convex and concave bends in the fault by creating a fault bend that correlates to every bend within a fold's dip domains

segs1	-- cell arrays of matrices of X-Y-Z coordinates of backlimb dip domains (best bet: segsSameDipPos)
segs2	-- cell arrays of matrices of X-Y-Z coordinates of frontlimb dip domains (best bet: segsSameDipNeg)
string	-- a string, specify 'synAnt' or 'sameDip' (default sameDip)
cvexBends	-- column vector that stores the estimated X coordinate of convex fault bends
ccaveBends	-- same, but for concave fault bends

see also dscFaultBendsSimple, dsbDipDomains, dscFRW, estGRFPms

note: code has not been written to meaningfully accomodate the use of 'synAnt' (segsSynAnt/segsAntSyn)--'string' is not used

dscFaultBendsSimple

DSCFAULTBENDSSIMPLE fault bends chosen by first points of fold dip domains

[cvexBends, ccaveBends] = dscFaultBendsSimple(segs1,segs2,string);
finds the number of convex and concave bends by creating a fault bend that correlates with every leading point of a fold's dip domains

segs1	-- cell arrays of matrices of X-Y-Z coordinates of backlimb dip domains (best bet: segsSameDipPos)
segs2	-- cell arrays of matrices of X-Y-Z coordinates of frontlimb dip domains (best bet: segsSameDipNeg)
string	-- a string, specify 'synAnt' or 'sameDip' (default sameDip)
cvexBends	-- column vector that stores the estimated X coordinate of convex fault bends
ccaveBends	-- same, but for concave fault bends

see also dscFaultBendsAll, dsbDipDomains, dscFRW, estGRFPms

note: code has not been written to meaningfully accomodate the use of 'synAnt' (segsSynAnt/segsAntSyn)--

dscFaultSlopes

DSCFAULTSLOPES predicts slopes for each fault segment

[faultSlopes]=dscFaultSlopes(concavebends,convexbends,...
segsPos,segsNeg,FOrud,simpleorall)

concavebends	-- an n x 1 vector of X coordinates of concave bends
convexbends	-- an n x 1 vector of X coordinates of convex bends
segsPos	-- cell arrays of matrices of X-Y-Z coordinates of backlimb dip domains (best bet: segsSameDipPos)
segsNeg	-- cell arrays of matrices of X-Y-Z coordinates of frontlimb dip domains (best bet: segsSameDipNeg)
FOrud	-- an n x 3 matrix of X-Y-Z locations of the points in the reduced fold polyline
simpleorall	-- a string, 'simple' if you want to only use leading bends to determine fault segments, or 'all' if you want to use intermediate bends in the fold to determine fault segments. default 'all'
faultSlopes	-- a column vector of estimated fault slopes

note: the minimum allowed fault slope is 0; the max is the largest slope
of any positively dipping segment--while this is not geometrically true,
it provides good predictions

see also estGRFPms

dscFRW

DSCFRW takes fault bend locations and returns fault relative widths

FRW=dscFRW(convexBends, concaveBends, FOVal,X,estSlip) determines
fault
relative width of each fault segment by looking at fault bend locations
relative to the total fault length

convexBends	-- column vector that stores the estimated X coordinate of convex fault bends
concaveBends	-- same, but for concave fault bends
FOVal	-- structure of polyline properties generated by getFitObjVal
X	-- the X value for the first point of the fault
estSlip	-- scalar of the maximum estimated slip from estGRSlipPms
FRW	-- a column vector of relative widths of each segment

see also dscFaultBendsAll, dscFaultBendsSimple, estGRSlipPms, estGRFPms

estGRFPms

ESTGRFPMS estimate parameters GRFPms to use in findFbf

[GRFPms,GRSlipPms] = estGRFPms(fitnessObjectives,SDorSA,... simpleorall,simpCutoffVal,GRFPmsWind) reads the fitness objectives and determines the GRFPms parameters that will more likely generate a reasonable initial population of faults for findFbf

fitnessObjectives	-- a structure that can contain fields layers (a cell array of layers, each of which is an n X 3 cell array of x y z points that describe the fold layer OR can be just an n X 3 matrix of x y z points) and fault (matrix of x y z points of known fault segments)
SDorSA	-- a string, 'sameDip' if you want to describe dip domains by the "same dip" method, 'synAnt' if you want to describe dip domains by the "syn/ant-bounded method". see dsbDipDomains for more details. default: 'sameDip'
simpleorall	-- a string, 'simple' if you want to only use leading bends to determine fault segments, or 'all' if you want to use intermediate bends in the fold to determine fault segments. see dscFaultBendsSimple and dscFaultBendsAll for more details. default: 'all'
simpCutoffVal	-- a scalar for the cutoff value for simplifyLine default: 0.1
GRFPmsWind	-- a scalar for the range for genGRFPms to generate GRFPms. default: 0.1
GRFPms	-- a structure containing fields x, z, layerLength, faultSlope, and faultRelWidth. x is the X coordinate of the initial est fault point, layerLength is the length of the fault. z is a 1 x 2 vector of the minimum and maximum z values for the initial fault point. faultSlope is an n X 2 matrix with each row giving the minimum and maximum slope for that fault segment, and faultRelWidth is the same size matrix giving the min and max fault relative width for each fault segment.
GRSlipPms	-- a minimum and maximum estimate for the input slip

see also testEstGRFPms, testEstGRFPms2, calls estGRSlipPms

estGRSlipPms

ESTGRSLIPPMS estimates slip based on fold shape

GRSlipPms = estGRSlipPms(segs,FOVal,simpleorall) generates minimum and maximum slip estimates based on fold shape. minimum slip is 0, maximum slip is the max of the two estimations of maximum slip calculated:

method 1: maxSlip is the maximum length of any of the dip domains

method 2: maxSlip is the hypotenuse length of the triangle formed by the range in Z of the fold and the maximum slope of any fold segment

segs	-- cell array of backlimb dip domains (segsSameDipPos or segsSynAnt) a matrix of points will also work
FOVal	-- structure of polyline properties generated by getFitObjVal
simpleorall	-- a string, 'simple' if you want to only use leading bends to determine fault segments, or 'all' if you want to use intermediate bends in the fold to determine fault segments. see dscFaultBendsSimple and dscFaultBendsAll for more details.
GRSlipPms	-- a 1 x 2 of the min and max slip estimates; min slope is 0, max is determined by the length of dip domains

see also estGRFPms

mergeSegsAntSyn

MERGESEGSANTSYN makes bools to make polylines of ant/syn-bounded dip doms

[use, append] = mergeSegsAntSyn(synAnt, FODipOrNot) creates booleans for use in generating dip domains that include all bends within an anticline/syncline pair of bends

the following are boolean column vectors that indicate:

synAnt	-- whether each bend in the fold is a synclinal (1) or anticlinal (0) hinge
FODipOrNot	-- whether a segment is dipping (1) or "flat" (0) (flat +/- 2 degrees).
use	-- whether to use (1) a segment in generating a dip domain or not (0)
append	-- whether it should be appended to a previous segment as part of apolyline (1) or not (0)

see also mergeSegsSynAnt, dsbDipDomains

mergeSegsSameDip

MERGESEGSSAMEDIP creates bools to create polylines of SameDip dip domains

function [use, append] = mergeSegsSameDip(asDipDir, FODipOrNot) creates booleans for use in generating dip domains that include all bends that have axial traces that are dipping in the same direction

the following are boolean column vectors that indicate:

asDipDir	-- a boolean vector indicating whether the axial trace at a point is positive sloping (1) or negative sloping (0)
FODipOrNot	-- whether a segment is dipping (1) or "flat" (0) (flat +/- 2 degrees).
use	-- whether to use (1) a segment in generating a dip domain or not (0)
append	-- whether it should be appended to a previous segment as part of apolyline (1) or not (0)

see also combineSegsFromBool, separateSegsSameDip, dsbDipDomains

mergeSegsSynAnt

MERGESEGSSYNANT makes bools to make polylines of syn/ant-bounded dip doms

[use, append] = mergeSegsSynAnt(synAnt, FODipOrNot) creates booleans for use in generating dip domains that include all bends within an synclinal/anticlinal pair of bends

the following are boolean column vectors that indicate:

synAnt	-- whether each bend in the fold is a synclinal (1) or anticlinal (0) hinge
FODipOrNot	-- whether a segment is dipping (1) or "flat" (0) (flat +/- 2 degrees).
use	-- whether to use (1) a segment in generating a dip domain or not (0)
append	-- whether it should be appended to a previous segment as part of apolyline (1) or not (0)

see also mergeSegsAntSyn, dsbDipDomains

separateSegsSameDip

SEPARATESEGSSAMEDIP divides segsSame dip into + and - dipping dip domains

[segsSameDipPos, segsSameDipNeg] = separateSegsSameDip(segsSameDip) separates SegsSameDip into segsSameDipPos, positively sloping segments include all segments with the same axial surface dip direction, and segsSamdDipNeg, negatively sloping segments of the same.

see also mergeSegsSameDip, combineSegsFromBool, dsbDipDomains

testEstGRFPms

TESTESTGRFPMS wrapper to test estGRFPms, display fault pops for GRFPms

testEstGRFPms(fbfParams,n,simpCutoffVal,GRFPmsVal,simpOrAll,SDorSA)
reads the fbfParams for a fold and specified parameters and runs
estGRFPms to create and plot a population of estimated faults
-- this plots the random variations for one set of GRFPms

fbfParams	-- a structured array of the observed fold, generated from either fbfFor or buildFbfParams
n	-- the number of faults to be generated
simpCutoffVal	-- a scalar for the cutoff value for simplifyLine
GRFPmsVal	-- a scalar for the range for genGRFPms to generate GRFPms
simpleorall	-- a string, 'simple' if you want to only use leading bends to determine fault segments, or 'all' if you want to use intermediate bends in the fold to determine fault segments. see dscFaultBendsSimple and dscFaultBendsAll for more details.
SDorSA	-- a string, 'sameDip' if you want to describe dip domains by the "same dip" method, 'synAnt' if you want to describe dip domains by the "syn/ant-bounded method". see dsbDipDomains for more details.

can output (but currently does not)

faultsArray	-- an array containing the matricies of the X-Y-Z points of each fault shape generated

see also testEstGRFPms2, estGRFPms

testEstGRFPms2

TESTESTGRFPMS2 wrapper, generates a pop of faults for many GRFPms values based on the input specifications

testEstGRFPms2(fbfParams,num,simpCutoff,GRFPmsWind,simpAllAmt) reads the fbfParams for a fold and specified parameters and runs estGRFPms to create and plot a population of estimated faults -- this plots the random variations for many sets of GRFPms--within the ranges of values specified, a GRFPms window, linesimplification cutoff, and 'simple' or 'all' are used, to create a GRFPms structure, then a fault is randomly generated within those parameters.

fbfParams	-- a structured array of the observed fold generated from either fbfFor or buildFbfParams
num	-- the number of faults generated in the pop. default 1.
simpCutoff	-- a 1 x 2 giving the min and max values to use for fold simplification in simplifyLine. default [.01 0.4].
GRFPmsWind	-- a 1 x 2 giving the min and max values to use for the ranges in the GRFPms. default [0 0.3]
simpleAllAmt	-- a 1 x 2 giving the min and max amount of the time that 'all' is used relative to 'simple'. 0 is all 'simple', 1 is all 'all' default [0.5 1]

can output (but currently does not)

faultsArray	-- an array containing the matricies of the X-Y-Z points of each fault shape generated

see also testEstGRFPms, estGRFPms

APPENDIX II: USE OF OTHER NON-BUILT-IN FUNCTIONS

Below are listed the functions, with their occurrence, that are non-built-in but are implemented within the codes listed above:

getFitObjVal (in estGRFPms and estGRSlipPms)

simplifyLine (in estGRFPms and testEstGRFPms)

getFaultSlope (in dsbDipDomains, seperateSegsSameDip, dscFaultSlopes)

getLineSegmentLength (in estGRSlipPms)

getFFAngFromFault (in dscFRW)

genGRFPms (in estGRFPms)

genRandFaultFromAngle (in testEstGRFPms)

VIII. ACKNOWLEDGEMENTS

I would most like to thank Dr. Chris Connors for his countless hours of guidance and willingness to share his insight and experience. Chris consistently goes beyond what is required in his teaching and research, and I am thankful for the opportunity to learn from and work with him. His enthusiasm and talent for teaching and advising has helped me to develop my career goals and research interests.

I would also like to thank the other faculty who I have had at Washington and Lee, including Dr. Peter Adams, Dr. Linda Davis, Dr. David Harbor, Dr. Lisa Greer, Dr. Elizabeth Knapp, Dr. Sara Bier, Dr. Matt Powell, Dr. Fred Schwab and Dr. Ed Spencer. I have enjoyed learning from all of them, and I appreciate their dedication and encouragement. I would also like to thank the Department of Geology for providing outstanding opportunities by offering unique courses and valuable research experiences.