DFNSIM: A New Program for Simulating Discrete Fracture Networks

Eric B. Niven and Clayton V. Deutsch

A new program called DFNSIM has been created for the simulation of Discrete Fracture Networks (DFNs). DFNSIM works by simulating a pool of more fractures than are required and finding a subset of fractures that best respects the target fracture spacing, orientation and intensity. Fractures in the DFN are classified into two subsets: activated or deactivated. The DFN consists of activated fractures. The program starts with an initial DFN and iterates by randomly activating and deactivating fractures. An objective function is used to measure the effectiveness of the iterations and is based on the perpendicular distance to the nearest fracture, the nearest fracture inter-angle and the DFN fracture intensity. The objective function is minimized by accepting activations and deactivations that reduce the objective function. This paper reviews the DFNSIM algorithm in detail. The data search strategy and the method for calculating perpendicular distance are reviewed. The modeling parameters from the parameter file are also discussed.

1. Introduction

Although there are several computer codes available for simulation of DFNs (Hartley 1998, Golder Associates 2010), these programs do not attempt to match input distributions of fracture spacing and the angle between the fractures.

This paper introduces a new program called DFNSIM, which can be used to simulate a DFN using an iterative algorithm that allows the resulting DFN to better match distributions of fracture spacing and nearest fracture inter-angle. The nearest fracture inter-angle is the angle between the poles of two fracture planes, which are the closer to each other than any other fractures. The concept of *nearest fracture inter-angle* is described in more detail in Niven and Deutsch (2010a).

This paper is part of a series of four articles in this report that pertain to fracture modeling. This paper presents a new computer program to simulate DFNs using the methodology described in (2010a). Niven and Deutsch (2010c) discuss and present evidence that some natural fracture networks cannot be accurately modeled using traditionally created DFNs. Finally, Niven and Deutsch (2010b) present an example of the program and methodology applied to a fracture modeling problem from Northern Alberta.

2. The DFNSIM Algorithm

The first step in the DFNSIM algorithm is to create a pool of fractures. However, more fractures are created for the pool than are required. A fracture intensity multiplication factor is used to describe the ratio of the total amount of fractures required in the pool to the amount of fractures required to satisfy the desired fracture intensity. Thus, if the intended fracture intensity is 10 and the fracture multiplication factor is 2, 20 fractures will be created and assigned to the pool of fractures. The pool of fractures is created using random fracture locations and randomly choosing fracture orientations from an input distribution (see Niven and Deutsch (2010a) for more details on the creation of the pool of fractures).

Next, fractures are selected at random and assigned to the *activated* group of fractures until the number of activated fractures equals the desired fracture intensity (10 in this case). Those fractures that are not activated are termed *deactivated*. Figure 1 illustrates the concept of the activated and deactivated fractures along with the pool of fractures. There are 20 fractures in the pool. 10 fractures are activated and 10 are deactivated.

At this point, the activated fractures represent the *initial DFN*, which is a starting point DFN that is iterated upon later. When the required number of iterations has been completed, the fractures in the activated group represent the *final DFN*. After the final DFN is generated the deactivated fractures are thrown away.

Search Strategy

After the initial DFN is created, the search strategy is set-up. From the location of each fracture centroid, a search is required to identify the nearest fractures in order to calculate fracture spacing and similarity of orientation. The search is designed to run once at the beginning of the program. Fractures are located by the coordinates of their centroids. The search strategy is as follows:

1. The Superblock search from SGSIM is set-up to find the fracture centroids.
2. After the Superblock search is set up, a fracture is visited.
3. At that fracture location, the Superblock search identifies the nearest *nclose* fracture centroids.
4. The Euclidean distance between centroids is a poor measure for fracture spacing and is only used to initially identify close fractures. The perpendicular distance from the visited fracture to the nearest fractures is calculated. This is demonstrated in Figure 2. The program calculates the average perpendicular distance to the *nclose* number of fractures identified by the Superblock search.
5. A certain number of the *nclose* nearest perpendicular fractures and their perpendicular distances are stored in memory for later calculations. The parameter *keepn*, specifies the number of closest perpendicular fractures and distances stored for each fracture visited.
6. This is repeated for each fracture in the pool of fractures.

In the "Data Search Parameters" section of the parameter file (discussed later), search parameters can be set to improve the speed of the search for large amounts of fractures.

Finding the nearest fractures by perpendicular distance

The super block search identifies and sorts, by Euclidean distance between centroids, all fracture centroids that fall within the specified superblock search radius. However, in order to assess fracture spacing DFNSIM uses the average perpendicular distance to the nearest fracture. Figure 2 illustrates this. Note that the two perpendicular distances between two fractures, d_{p1} and d_{p2}, are usually not equal. Thus, DFNSIM uses an average of the two perpendicular distances, $d_{p,ave}$ as a measure of fracture spacing. Paper 102 (Niven and Deutsch 2010a) discusses the calculation of the nearest fracture. After the perpendicular distances are calculated the histogram of perpendicular distance to the nearest fracture is built.

Calculation of Nearest Fracture Inter-Angle

After the initial search, the pool of fractures is processed to calculate the fracture inter-angle histogram:

1. A fracture is visited.
2. The fracture with the smallest perpendicular distance to the current fracture is identified from the stored records.
3. The angle between the two fracture poles is calculated and termed the *nearest fracture inter-angle* and is stored in memory.
4. Each fracture is visited and the nearest fracture inter-angle for each is stored.
5. Then the histogram of nearest fracture inter-angle is constructed.

The angle between the two fracture poles (the inter-angle) is calculated as follows:

$$\theta_{\text{inter-angle}} = \arccos\left(\frac{\mathbf{a}\cdot\mathbf{b}}{|\mathbf{a}||\mathbf{b}|}\right) \tag{1}$$

At this point, the histograms of the nearest fracture inter-angle and perpendicular distance to the nearest fracture have been constructed, so the objective function can be calculated for the initial DFN.

The objective function is presented and discussed in paper 102 (Niven and Deutsch 2010a). After the objective function is calculated for the initial DFN, DFNSIM iterates, by changing the *activation* of each fracture one-at-a-time.

1. A fracture is selected randomly and visited. There are two random path options, which are discussed later in this article.
2. The activation state of the current fracture is switched from its current state. Note that changing the activation state changes the fracture intensity to be one greater or less than it was previously.
3. Activating or deactivating a fracture also changes which fractures are closest. DFNSIM sorts through the list of fractures and updates the nearest fractures as a result of the activation change.

4. Histograms of perpendicular distance to the nearest fracture and nearest fracture inter-angle are rebuilt with the recently updated nearest fracture information.
5. The objective function is calculated with the new histograms and the new fracture intensity as a result of the activation change.
6. If the objective function decreases as a result of the activation change, the activation change is accepted.
7. Go to 3, until all fractures have been visited in the random path or the desired number of fractures are visited.
8. Visiting each fracture once may not be enough to achieve reasonable results. Line 50 in the parameter file specifies the number of iteration loops (discussed below) so that each fracture may be visited a number of times.

The final DFN is written out in two formats, one of which is in the .fab format, which can be imported into the popular fracture modeling software FRACMAN. The histograms for the final DFN are also written out.

3. The Parameter File

Figure 3 shows an example of the DFNSIM parameter file. The parameters are organized into logical blocks. The first section specifies the grid-domain using standard GSLIB conventions and the random number seed. The next section specifies data search parameters.

```
11 ***Data Search Parameters***
12 1    1      1              -x,y and z block discretization
13 0                          -max per octant (0-> not used)
14 2000 2000 2000             -maximum search radii
15 0.0    0.0    0.0          -angles for search ellipsoid
16 -1                         -number of nearest data to perp. search, or <0 for all
17 3000000                    -Band width for perpendicular fracture search
18 3000000                    -Perpendicular distance to search
```

Lines 12-15 are superblock search parameters that are also specified in other GSLIB programs that use the superblock search. See Deutsch and Journel (1998) for more information on the superblock search.

Lines 16-18 specify search parameters related to the perpendicular distance calculation as shown in Figure 4. The superblock search identifies *nclose* data points (in this case, 5 fracture centroids are identified in the figure) based on the Euclidean distance from the current fracture centroid to the other centroids. However, the perpendicular distance from the fracture being searched to the other fractures needs to be calculated. The parameters above allow the user calculate the perpendicular distances for a subset of the *nclose* fracture centroids identified by the superblock search. These parameters help to speed up the program by limiting the number of perpendicular distance calculations. In addition, situations where the Euclidean distance is much larger than the perpendicular distance are avoided (such as when two fractures are nearly in line with each other but are separated by significant distance in the direction of the fracture's azimuth). Line 16, specifies the parameter *keepn*, which tells the program how many of the nearest data to calculate the perpendicular distance for. Line 17 specifies a band width for search in the plane of the fracture such that only centroids found within the bandwidth are subject to the perpendicular distance calculation. Line 18, specifies the distance to search for data in the direction parallel to the normal vector of the fracture for the perpendicular distance calculation.

```
20 ***Output Fracture Files***
21 fracdata.out               -Output GSLIB Fracture Data
22 fracdata.fab               -Output FracMan Fracture Data
23 fracdata_optimized.out     -Optimized output GSLIB Fracture Data
24 fracdata_optimized.fab     -Optimized output FracMan Fracture Data
25 1                          -fracture set names / numbers
```

Lines 21 to 25 in the parameter file specify the output files. The activated fractures are output for the initial DFN and the final DFN in two formats, one of which can be imported into the popular commercial fracture modeling software, FracMan. The other output files are in GeoEAS format using ASCII characters.

```
27  ***Input Fracture Distributions***
28  spacingdist.out             -Input Actual Fracture Spacing Distribution
29  angledist.out               -Input Actual Inter-Fracture Angle Distribution
```

Lines 28 and 29 specify the two input histogram distributions that must be input into DFNSIM. The first is the target distribution of the perpendicular distance to the nearest fracture. The second is the target distribution for the nearest fracture inter-angle.

```
31  ***Joint Set Parameters***
32  0                           -Load initial DFN from file? (1=yes, 0=no)
33  mapdata.out                 -File with Initial DFN
34  intensity1.out              -Fracture Intensity Input File
35  1 1                         -intensity real. to use, max # of realizations in file
36  0                           -use locally varying joint orientation (1=yes,0=no)
37  orientation1.out            -If "Yes", specify locally varying joint orientation file
38  146.53 11.80                -Mean pole trend and pole trend st. dev.
39  0     0.001                 -Mean pole plunge and pole plunge st. dev.
40  65 25 2                     -Fracture Length Mean, St. Dev. and minimum value
41  2 0.002 0.002               -Fracture Height Mean, St. Dev. and minimum value
42  0.5 0.1 0.01                -Mean Fracture Aperture, St. Dev. and minimum value
43  2                           -random z location for fractures? (1=yes,2=no)
44  2                           -Fracture Intensity Multiplication Factor
```

The joint set parameters section specifies the parameters for building the pool of fractures. Lines 32 and 33 can be used to specify an initial DFN if one already exists, instead of generating one automatically. Line 34 refers to the fracture intensity file. If there is more than one realization of intensity in the file, the correct realization and the maximum number of realizations are specified on the next line.

The user can specify locally varying fracture orientation on line 36 and 37. If there is no locally varying fracture orientation, lines 38 through 42 specify input normal distributions for fracture orientation and size parameters such as pole dip direction, pole dip, length, height and aperture, respectively. Note that fracture length, height and aperture require three values. The mean and standard deviation define the normal distribution from which those variables are sampled from and a minimum possible value truncates the distribution on the lower end if the user wishes to avoid extremely small fractures or apertures.

Line 43 specifies whether or not the fractures should have a random z (vertical) location within the grid blocks. If the user specifies '2' for that variable, the z-direction, the fractures will be located in the middle of the blocks in the z-direction (elevation). This feature is handy for two dimensional problems. Line 44 specifies the fracture intensity multiplication factor. This is the ratio between the total number of fractures created for the pool of fractures and the number of activated fractures.

```
46  ***Optimization Variables***
47  3                           -Optimize by intensity and: 1=spacing, 2=angle, 3=all three
48  0.000035                    -objective function parameter: Intensity constant
49  850                         -Number of iterations per loop
50  4                           -Number of iteration loops
51  2                           -Random Path Option:1=random,2=random without replacement
```

The next section of parameters (see directly above) generally relate to the optimization and iteration process of DFNSIM. The objective function is made up of three components which relate to: 1) the perpendicular distance to the nearest fracture, 2) the nearest fracture inter-angle and 3) the fracture intensity. DFNSIM always includes the intensity portion of the objective function, but is capable of ignoring either spacing or inter-angle as necessary. One example where inter-angle might not be important is if the desired fractures have a very narrow distribution of orientation.

Line 48 specifies an intensity constant for the objective function (see Niven and Deutsch (2010a)). If the intensity constant is too large proposed fracture activations/deactivations will always be rejected since the fracture intensity portion of the objective function always starts at zero (since the initial DFN intensity equals the target intensity). A properly set intensity constant allows enough activations or deactivations to be accepted while keeping the final fracture intensity close to the target intensity. In the map example from Niven and Deutsch

(2010b), the target fracture intensity is 425 and the chosen intensity constant allows the fracture intensity to increase or decrease by approximately 4 or 5. It may take some trial and error to find a suitable intensity constant.

Lines 49 and 50 specify the number of iterations per loop and the number of loops. A good idea would be to set the number of iterations per loop to be equal to the total number of the fractures in the pool so that each fracture can be activated/deactivated once. However, one loop may not be enough to reach a good solution, so the user can specify as many loops as needed. Line 51 specifies the random path option. The random path can be completely random, which means that a portion of fractures (per loop) will not be visited and some will be visited more than once. If random path option '2' is specified, the random path locations are specified, in advance, without replacement so that every fracture is visited exactly once per loop, but in a random order. If more than one loop is specified, a new random path is specified for each loop. The 2nd random path option is preferred, since it ensures that every fracture is visited in fewer iterations.

```
53  ***Output Files***
54  1                           -output objective function progress? (1=yes, 0=no)
55  iterations.out              -File for iteration objective function progress
56  results.out                 -File for spacing and inter-angle results
```

Finally, the last section of the parameter file specifies the two output files: 1) with the objective function progress for each iteration, and 2) the initial and final distributions of perpendicular distance to the nearest fracture and nearest fracture inter-angle.

An example application of DFNSIM can be found in Niven and Deutsch (2010b).

4. Convergence and Time Trials

Figure 5 shows the convergence results from the fracture map example in paper 206 (Niven and Deutsch 2010b). The objective function decreases rapidly during the first 900 iterations, which roughly corresponds to the first full iteration loop. Beyond the first iteration loop, objective function improvement is much slower. This finding echoes results from other applications of the program to real examples. Although DFNSIM is a new program and has undergone limited testing to this point, the author has never found an example where more than five iteration loops were needed to achieve stable convergence.

The map example from paper 206 took 12.5 seconds to run with four iteration loops. However, the run time could have been decreased further by adjusting the search parameters to limit the number of distance calculations.

By increasing the amount of super blocks used in the search, very large fracture networks can be simulated in a reasonable amount of time. Table 1 shows the results of a time trial for the data search alone. As the number of fractures increases, the number of grid blocks is also increasing such that the number of grid blocks stays the same. For each trial there are 20 fractures per grid block. In each case, the number of super blocks is one quarter the number of grid blocks (one half in the x and y directions). As long as the number of fractures in each grid block is kept constant, the search time increases linearly with increasing amounts of fractures (**Figure 6**).

Table 1: Search time trial results

# of Fractures in Model	Grid blocks in x and y directions	Super blocks in x and y directions	Total number of grid blocks	# of fractures per grid block	Search time (min)
2,000	10	5	100	20	0.02
50,000	50	25	2,500	20	0.13
200,000	100	50	10,000	20	0.53
800,000	200	100	40,000	20	2.15
1,800,000	300	150	90,000	20	4.78
5,000,000	500	250	250,000	20	13.83

5. Conclusions

DFNSIM, a new program for simulating discrete fracture networks, is introduced. DFNSIM works by simulating a pool of more fractures than are required. The program iterates to find a subset of the pool of fractures that better matches the desired fracture spacing and nearest fracture inter-angle.

An objective function minimization is used to measure the quality of fit between the distributions of target fracture spacing, nearest fracture inter-angle and fracture intensity and distributions from the DFN created by the program.

References

Deutsch, C.V. and Journel, A.G. 1998. GSLIB: Geostatistical software library and user's guide. Oxford University Press, .

Golder Associates. 2010. FracMan7: Interactive discrete feature, data analysis, geometric modeling and exploration simulation. Golder Associates, .

Hartley, L.J. 1998. NAPSAC (release 4.1) technical summary document. AEA Technology, Oxon, UK.

Niven, E.B. and Deutsch, C.V. 2010a. A new approach to DFN simulation. Paper 102, Report 12, Centre for Computational Geostatistics, University of Alberta, Edmonton, Alberta.

Niven, E.B. and Deutsch, C.V. 2010b. An example application of DFNSIM in modeling surface lineaments. Paper 206, Report 12, Centre for Computational Geostatistics, University of Alberta, Edmonton, Alberta.

Niven, E.B. and Deutsch, C.V. 2010c. On the randomness of natural fractures. Paper 207, Report 12, Centre for Computational Geostatistics, University of Alberta, Edmonton, Alberta.

Figures

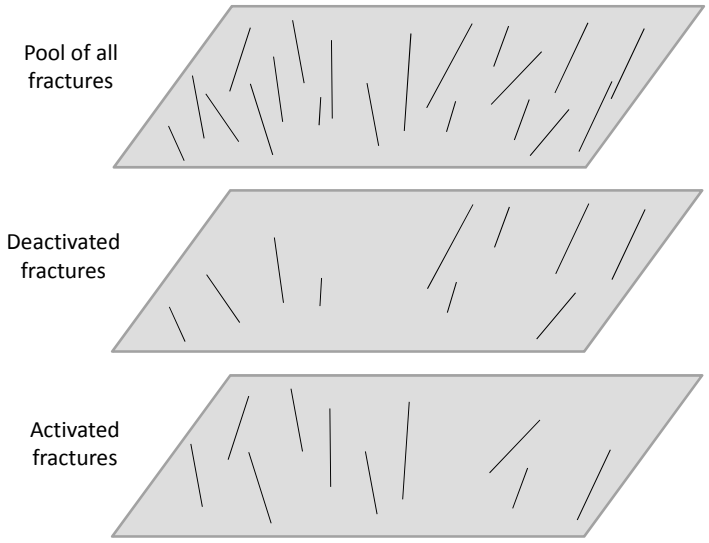

Figure 1: A 2D illustration of the activated, deactivated and pool of fractures. There are 10 activated and 10 deactivated fractures. Thus, there are 20 fractures in the pool. The activated fractures represent the DFN under construction. The deactivated fractures are thrown away when the desired number of iterations is complete.

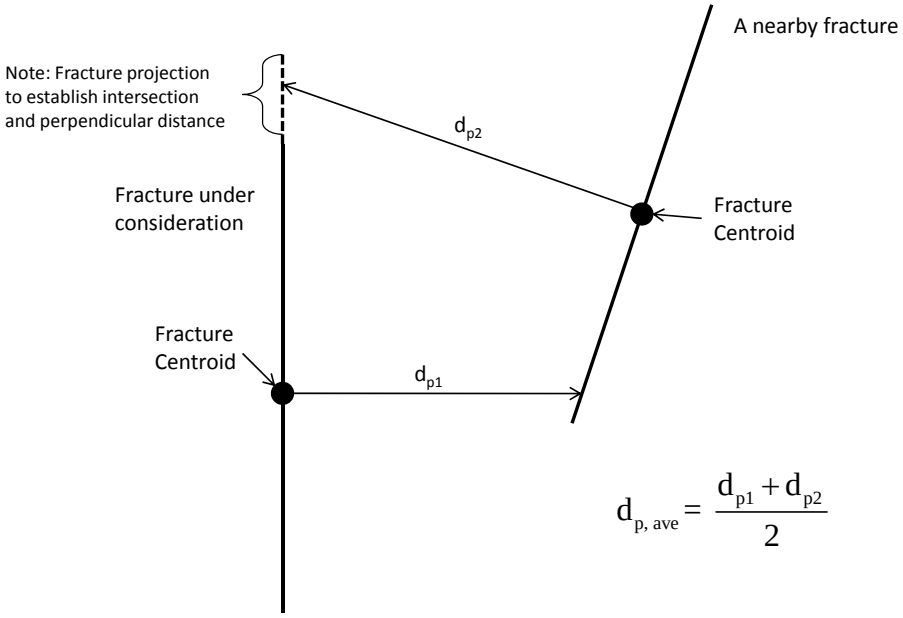

Figure 2: The procedure for calculating the perpendicular distance to the nearest fracture.

```
Parameters for DFNSIM
**********************


START OF MAIN PARAMETERS:
1 503 1006                    -nx,xmn,xsiz
1 525 1050                    -ny,ymn,ysiz
1  0.05 0.1                   -nz,zmn,zsiz
6911                          -random number seed
------------------------------------------------------------------------
***Data Search Parameters***
1    1      1                -x,y and z block discretization
0                             -max per octant (0-> not used)
2000 2000 2000                -maximum search radii
0.0    0.0    0.0             -angles for search ellipsoid
-1                            -number of nearest data to perp. search, or <0 for all
3000000                       -Band width for perpendicular fracture search
3000000                       -Perpendicular distance to search
------------------------------------------------------------------------
***Output Fracture Files***
fracdata.out                  -Output GSLIB Fracture Data
fracdata.fab                  -Output FracMan Fracture Data
fracdata_optimized.out        -Optimized output GSLIB Fracture Data
fracdata_optimized.fab        -Optimized output FracMan Fracture Data
1                             -fracture set names / numbers
------------------------------------------------------------------------
***Input Fracture Distributions***
spacingdist.out               -Input Actual Fracture Spacing Distribution
angledist.out                 -Input Actual Inter-Fracture Angle Distribution
------------------------------------------------------------------------
***Joint Set Parameters***
0                             -Load initial DFN from file? (1=yes, 0=no)
mapdata.out                   -File with Initial DFN
intensity1.out                -Joint Intensity Input File
1 1                           -intensity real. to use, max # of realizations in file
0                             -use locally varying joint orientation (1=yes,0=no)
orientation1.out              -If "Yes", specify locally varying joint orientation file
146.53 11.80                  -Mean pole dip direction and st. dev
0     0.001                   -Mean pole dip and st. dev
65 25 2                       -Fracture Length Mean, St. Dev and minimum
2 0.002 0.002                 -Fracture Height Mean, St. Dev and minimum
0.5 0.1 0.01                  -Mean Fracture Aperture, St. Dev and minimum
2                             -random z location for fractures? (1=yes,2=no)
2                             -Fracture Intensity Multiplication Factor
------------------------------------------------------------------------
***Optimization Variables***
3                             -Optimize by intensity and: 1=spacing, 2=angle, 3=all three
0.000035                      -objective function parameter: Intensity constant
850                           -Number of iterations per loop
4                             -Number of iteration loops
2                             -Random Path Option:1=random,2=random without replacement
------------------------------------------------------------------------
***Output Files***
1                             -output objective function progress? (1=yes, 0=no)
iterations.out                -File for iteration objective function progress
results.out                   -File for spacing and inter-angle results
```

Figure 3: DFNSIM parameter file.

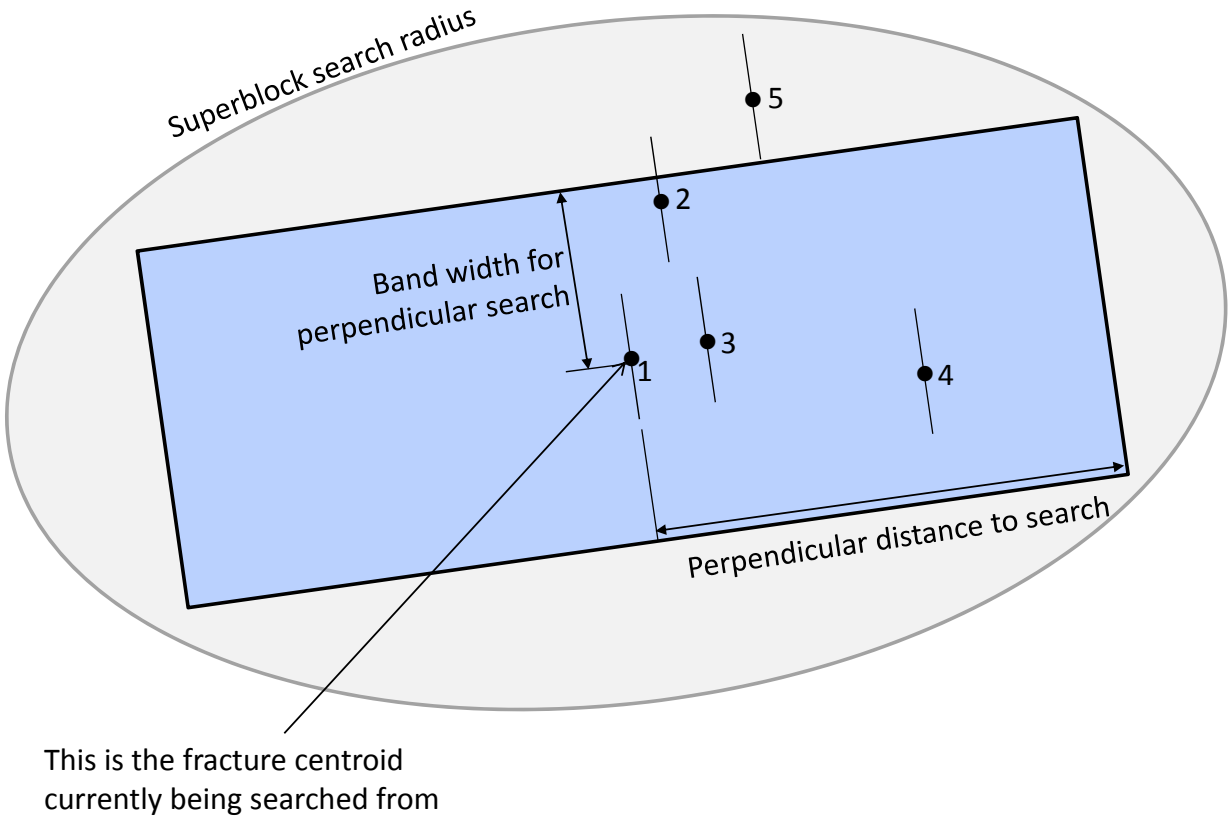

Figure 4: The relationship between the superblock search and the perpendicular search. The program currently at location 1 and is searching for other fractures nearby. The superblock search identifies all five fracture centroids. However, the perpendicular distance is only calculated for the first four centroids due to the choice of band width and perpendicular distance to search. Note that fracture 3 is the closest to fracture 1 using Euclidean distance, but when perpendicular distance is used fracture 2 is closer to 1 than 3 is.

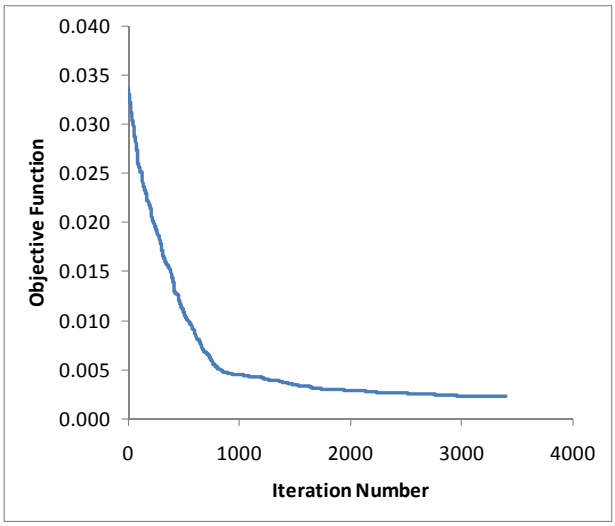

Figure 5: Convergence results from fracture map example from paper 206.

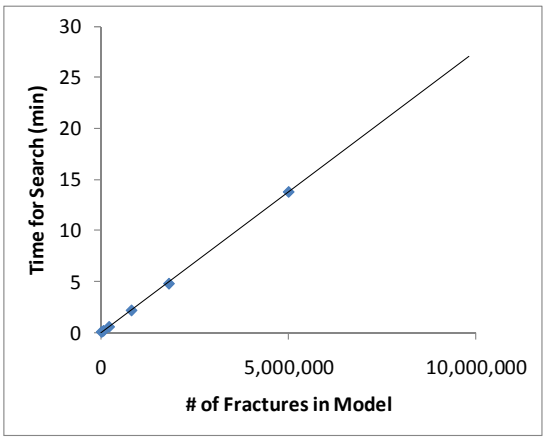

Figure 6: Search time trial results for increasing numbers of fractures.