2

Matlab Programs: Alghalandis Fracture Network Modelling (AFNM) Package

2.1 Introduction

Alghalandis Fracture Network Modelling (AFNM) is a package of computer codes in Matlab language syntax which consists of a) functions to generate fracture networks in two and three dimensions based on stochastic modelling principals (e.g., discrete fracture network modelling framework); b) functions to characterise synthesised or imported two- and three-dimensional fracture networks including intersection analysis, density measures, connectivity indices, clustering and many others; c) functions for highly simplification of visualisation of two- and three-dimensional fracture networks; and d) functions to generically utilise the above stages and to extend their use for practical applications, to provide stable framework for further developments, and tools to save the resulting maps, tables and information in appropriate formats readable by many common standard software applications. The intension of this endeavour was to unify the computation stages i.e., framework and even more the core functions such to get the highest productivity. With the hope readers will find this package handy and useful for evaluation of the concepts proposed in this thesis, developing their own ideas and experiencing fracture network modelling concepts.

A couple of external Matlab single codes or packages are linked here without inclusion of their code including "*geom2d*[10]" and "*geom3d*[11]" (edition 2011, by David Legland), "*circstat*[12]" (edition 2011, by Philipp Berens), "*kde2d.m*[13]" (edition 2009, by Zdravko Botev), "*smoothn.m*[14]" (edition 2010, by Damien Garcia), "*vol3d.m*[15]" (edition 2009, by Oliver Woodford), and "*dict.m*[16]" (edition 2008, by Doug Harriman).

---

10 http://www.mathworks.com.au/matlabcentral/fileexchange/7844-geom2d
11 http://www.mathworks.com.au/matlabcentral/fileexchange/24484-geom3d
12 http://www.mathworks.com.au/matlabcentral/fileexchange/10676-circular-statistics-toolbox-directional-statistics
13 http://www.mathworks.com.au/matlabcentral/fileexchange/17204-kernel-density-estimation
14 http://www.mathworks.com.au/matlabcentral/fileexchange/25634-easy-n-fast-smoothing-for-1-d-to-n-d-data
15 http://www.mathworks.com.au/matlabcentral/fileexchange/22940-vol3d-v2
16 http://www.mathworks.com.au/matlabcentral/fileexchange/19647-dict

2.2 License

The provided computer codes in this chapter are copyrighted. They are made publically available for use under the following license and conditions.

2.3 List of functions

Name	Description	Name	Description
`Angles2D`	*angles of 2D lines*	`Centers2D`	*centres of 2D lines*
`Lengths2D`	*lengths of 2D lines*	`GenFNM2D`	*2D fracture network*
`ClipLines2D`	*clips 2D lines*	`LinesXLines2D`	*two line sets intersections*
`LinesX2D`	*intersection analysis on 2D lines*	`LinesToClusters2D`	*clusters of lines*
`Density2D`	*true density of 2D lines*	`Histogram2D`	*2D histogram (density)*
`RandLinesInPoly2D`	*random line sampling*	`Sup2D`	*creates 2D support*
`SupCSup2D`	*two 2D supports' connectivity*	`SupXLines2D`	*intersections between a support and 2D lines*
`SupXNLines2D`	*sup intersects 2D lines*	`P21G`	*P21 gridded measure*
`ConnectivityIndex2D`	*CI*	`ConnectivityField2D`	*CF*

BreakLinesX2D	*break lines at their intersections*	Rotate2D	*rotates 2D points*
SortPoints2D	*topological sort of points*	Isolated2D	*checks if a 2D line isolated*
Backbone2D	*backbone of 2D lines*	IsolatedLines2D	*checks isolation for all 2D lines*
BackboneToNodesEdges2D	*backbone to graph*	Expand2D	*expands 2D matrix*
Resize2D	*resize 2D matrix*	DrawLines2D	*draws 2D lines*
LinesToXYnan2D	*convert lines to X and Y*	ExpandAxes2D	*expands 2D axes*
Titles2D	*title, labels, grid … for axes in 2D*	RandPoly3D	*random 3D polygons*
GenFNM3D	*3D facture network*	Sup3D	*creates 3D support*
ClipPolys3D	*clips 3D polygons*	PolysX3D	*intersection analysis on 3D polygons*
PolysXPolys3D	*polygons' intersections*	PolyXPoly3D	*intersection between two 3D polygons*
SupCSup3D	*two 3D supports' connectivity*	BBox3D	*bounding box of 3D points*
Expand3D	*expand 3D matrix*	Resize3D	*resizes 3D matrix*
SaveToFile3D	*save 3D result to file*	SavePolysToVTK3D	*saves 3D polygons as VTK file*
SetAxes3D	*sets 3D axes*	DrawPolys3D	*draws 3D polygons*
DrawSlices3D	*draws 3D slices*	VolRender3D	*draws volume render of 3D volumetric data*
Scale	*scales data*	ToStruct	*converts to `struct` format*
Clusters	*cluster analysis*	CheckClusters	*checks clusters for errors*
Labels	*labels*	Relabel	*relabels cluster labels*
Stack	*stacks cell data*	Group	*groups data based on common elements*
FarthestPoints	*two farthest points*	PDistIndices	*generates indices for `pdist` function*
Occurrence	*occurrence of points*	ConnectivityMatrix	*connectivity matrix (CM)*
FullCM	*full form of CM*	FNMToGraph	*converts fracture network to graph*
LoadColormap	*loads colormap*	SaveColormap	*saves current colormap*
SecondsToClock	*seconds to clock*	Colorise	*colourise given data*
ShowFNM	*shows fracture network*	Round	*rounds data*

2.4 Functions for Two Dimensional Cases

2.4.1 Angles2D

```
% Angles2D
% returns angles of 2D lines (fracture)
%
% Usage :
%    ags = Angles2D(lines)
%
% input : lines     (n,4)
% output: ags       (n) in radian
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function ags = Angles2D(lines)
ags = atan2(lines(:,4)-lines(:,2),lines(:,3)-lines(:,1));
```

`Angles2D` computes orientation angles of two-dimensional fractures (line segments) with correct sign for each quadrant. Lines are represented as a two-dimensional array of size $n \times 4$ here and in all other functions.

2.4.2 Centers2D

```
% Centers2D
% returns center points of 2D fracture lines
%
% Usage :
%    cts = Centers2D(lines)
%
% input : lines     (n,4)
% output: cts       (n,2)
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function cts = Centers2D(lines)
cts = 0.5*[lines(:,3)+lines(:,1),lines(:,4)+lines(:,2)];
```

`Centers2D` computes centre of fracture lines. The centring points can be used to compute the **DFC (FCD)**, for example.

2.4.3 Lengths2D

```
% Lengths2D
% returns lengths of 2D fracture lines
%
% Usage :
%    lhs = Lengths2D(lines)
%
```

```
% input : lines     (n,4)
% output: lhs       (n)
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function lhs = Lengths2D(lines)
if isempty(lines)
    lhs = 0;
else
    lhs = hypot(lines(:,3)-lines(:,1),lines(:,4)-lines(:,2));
end
```

Lengths2D calculates the Euclidean lengths between the two endpoints of lines. This function can be used for estimation of **P21** measure, for example.

2.4.4 GenFNM2D

```
% GenFNM2D
% generates 2D fracture network
%
% Usage :
%    [lines,olines] = GenFNM2D(n,theta,kappa,minl,maxl,rgn)
%
% input : n         number of fracture lines, default=150
%         theta     main orientation, default=0
%         kappa     Fisher dispersion factor, default=0: omnidirectional
%         minl      minimum length of fracture lines, default=0.05
%         maxl      maximum length of fracture lines, default=1
%         rgn       region of study, default=[0,1,0,1] i.e., unit square
% output: lines     fracture lines after clipping by rgn
%         olines    original lines
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function [lines,olines] = GenFNM2D(n,theta,kappa,minl,maxl,rgn)
if nargin<6; rgn = [0,1,0,1]; end
if nargin<5; maxl = 1; end
if nargin<4; minl = 0.05; end
if nargin<3; kappa = 0; end
if nargin<2; theta = 0; end
if nargin<1; n = 150; end
pts = rand(n,2);                                    %locations~ U(0,1)
ags = circ_vmrnd(theta,kappa,n);                    %oreint.~von-Mises(theta=0,kappa=0)
lhs = Scale(exprnd(1,n,1),minl,maxl);               %lengths~ Exp(mu=1)
[dx,dy] = pol2cart(ags,0.5*lhs);
olines = [pts(:,1)-dx,pts(:,2)-dy,pts(:,1)+dx,pts(:,2)+dy];  %original
lines = ClipLines2D(olines,rgn);                    %clipped by region of study
```

GenFNM2D synthesises two-dimensional fracture networks according to stochastic modelling principals in which for locations, orientations and lengths finite samples from desired random distribution functions are drawn. In this implementation for locations a simple uniform distribution is chosen while for orientations von-Mises distribution is used. The lengths are generated based on an exponential distribution and are truncated into a specific range by scaling. In use, one may adapt the code easily to benefit from other distributions.

2.4.5 ClipLines2D

```
% ClipLines2D
% returns clipped 2D fracture lines by a given rectangle (box)
%
% Usage :
%    clines = ClipLines2D(lines,box)
%
% input : lines     (n,4)
%         box       (4), default=[0,1,0,1]
% output: clines    (n,4)
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function clines = ClipLines2D(lines,box)
if nargin<2; box = [0,1,0,1]; end
[m,n] = size(lines);
clines = zeros(m,n);
for i = 1:m
    clines(i,:) = clipEdge(lines(i,:),box);
end
```

ClipLines2D provides handy tool to apply clipping to two-dimensional fracture lines by a given rectangle.

2.4.6 LinesXLines2D

```
% LinesXLines2D
% finds intersection indices and points between two sets of 2D fracture lines
%
% Usage:
%    [xpss,idss] = LinesXLines2D(lines1,lines2)
%
% input : lines1    (m,4)
%         lines2    (n,4)
% output: xtss      intersection points, (cell)
%         idss      intersecting lines' indices, (cell)
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function [xpss,idss] = LinesXLines2D(lines1,lines2)
m = size(lines1,1);
n = size(lines2,1);
idss = cell(m,1);
xpss = cell(m,1);
```

```
u = 0;
for i = 1:m                                             %for all lines in lines1
    idx = zeros(1,1);
    xps = zeros(1,2);
    found = false;
    k = 0;
    for j = 1:n                                         %for all lines in lines2
        xpt = intersectEdges(lines1(i,:),lines2(j,:));
        if ~isfinite(xpt(1)); continue; end
        k = k+1;
        idx(k) = j;
        xps(k,:) = xpt;
        found = true;
    end
    if found                                            %record if there was any
intersection
        u = u+1;
        idss(u) = {{i,idx}};
        xpss(u) = {xps};
    end
end
idss = idss(1:u);                                       %compaction
xpss = xpss(1:u);
```

`LinesXLines2D` is made available to apply intersection analysis between two sets of lines. The resulting intersection points and the identity number of intercepting lines are reported in the format of Matlab “cell” data type. The implementation is optimised to consume minimum memory requirement and also to deliver high performance.

2.4.7 LinesX2D

```
% LinesX2D
% finds intersection indices and points for a set of 2D fracture lines
%
% Usage :
%    [xts,ids,La] = LinesX2D(lines)
%
% input : lines     (n,4)
% output: xts       intersection points, (m,2)
%         ids       intersecting lines indices, (m,2)
%         La        cluster labels (n)
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function [xts,ids,La] = LinesX2D(lines)
n = size(lines,1);
m = n*(n-1)/2;                                      %max possible number of intersections
xts = zeros(m,2);
ids = zeros(m,2);
k = 0;
for i = 1:n-1                                           %apply optimum iteration
    for j = i+1:n
        xpt = intersectEdges(lines(i,:),lines(j,:));
```

```
        if ~isfinite(xpt(1)); continue; end
        k = k+1;
        xts(k,:) = xpt;                           %intersection points
        ids(k,:) = [i,j];                         %intersecting lines' indices
    end
end
xts = xts(1:k,:);                                 %compaction
ids = ids(1:k,:);
La = Labels(Clusters(num2cell(ids,2)),n);         %fracture cluster labels
```

LinesX2D is to compute intersections between all lines in a set of fracture lines. This function also computes fracture clusters with appropriately assigned labels. It plays an important role in analysing connectivity measures, for example.

2.4.8 LinesToClusters2D

```
% LinesToClusters2D
% finds clusters of 2D fracture lines (inter-connections)
%
% Usage :
%    La = LinesToClusters2D(lines)
%
% input : lines     (n,4)
% output: La        (n) cluster labels
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function La = LinesToClusters2D(lines)
[~,~,La] = LinesX2D(lines);
```

LinesToClusters2D wraps **LinesX2D** for cases in which only label of clusters are required rather than complete intersection information.

2.4.9 Density2D

```
% Density2D
% computes true density of 2D fracture network
%
% Usage :
%    [DN,x,y] = Density2D(lines,gm,gn)
%
% input : lines     (n,4)
%         gm        grid dimension vertically
%         gn        grid dimension horizontally
% output: DN        (gm,gn)
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function [DN,x,y] = Density2D(lines,gm,gn)
w = 1/gn;                                         %sampling cell's width
h = 1/gm;                                         %sampling cell's height
```

```
DN = zeros(gm,gn);
for i = 1:gm
    for j = 1:gn
        sup = [(j-1)*w,j*w,(i-1)*h,i*h];
        DN(i,j) = SupXNLines2D(sup,lines);
    end
end
x = [w/2,1-w/2];                                  %for usage in imagesc(x,y,DN)
y = [h/2,1-h/2];
```

Density2D computes the true density of fracture lines in the region of study based on a grid ($gm \times gn$) sampling.

2.4.10 Histogram2D

```
% Histogram2D
% computes 2D histogram (~density) of points
%
% Usage :
%    out = Histogram2D(pts,nx,ny)
%
% input : pts       (n,2)
%         nx        grid dimension on axis X
%         ny        grid dimension on axis Y
% output: out       (ny,nx)
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function out = Histogram2D(pts,nx,ny)
if nargin<3
   nx = 7;                                        %default grid size
   ny = 7;
end
k = [nx,ny];
bin = zeros(size(pts));
for i = 1:2                                       %iterate for X and Y coordinates
    minx = min(pts(:,i));
    egs = minx+((max(pts(:,i))-minx)/k(i))*(0:k(i));
    [~,t] = histc(pts(:,i),[-Inf,egs(2:end-1),Inf],1);
    bin(:,i) = min(t,k(i));
end
out = accumarray(bin(all(bin>0,2),:),1,k)';
```

Histogram2D provides a superbly fast evaluation of density of points based on the concept of histogram classification. For some cases this function is superior to **KDE** estimation if the local variation is more important than global smoothing.

2.4.11 RandLinesInPoly2D

```
% RandLinesInPoly2D
% generates random 2D sampling lines inside a 2D polygon
%
% Usage :
```

```
%    lines = RandLinesInPoly2D(n,h,dh,ang,da,poly)
%
% input : n          number of lines
%         h          length of lines
%         dh         length's tolerance
%         ang        angle
%         da         angle tolerance
%         poly       (k,2)
% output: lines      (n,4)
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function lines = RandLinesInPoly2D(n,h,dh,ang,da,poly)
if nargin<6; poly = [[0,0];[1,0];[1,1];[0,1]]; end      %default: unit square
if nargin<5; da = 0; end
if nargin<4; ang = 0; end
if nargin<3; dh = 0; end
if nargin<2; h = 0.1; end
if nargin<1; n = 100; end
px = poly(:,1);
py = poly(:,2);
lines = zeros(n,4);
i = 0;
while i<n
    x1 = rand(1,1);
    y1 = rand(1,1);
    rr = (2*rand(1,2)-1);
    [dx,dy] = pol2cart(ang+rr(1)*da,h+rr(2)*dh);
    x2 = x1+dx;
    y2 = y1+dy;
    if inpolygon([x1,x2],[y1,y2],px,py)           %if line is inside the polygon
        i = i+1;
        lines(i,:) = [x1,y1,x2,y2];
    end
end
```

RandLinesInPoly2D is to generate n random lines inside of a given polygon. The length of lines is defined by a scalar h with variation dh. The orientation follows scalar ang with tolerance of da both in radian.

2.4.12 Sup2D

```
% Sup2D
% creates a support at point(x,y) with width(w) and height(h)
%
% Usage :
%    sup = Sup2D(x,y,w,h)
%
% input : x,y        coordinates of the center of support
%         w,h        width and height
% output: sup        (4,4), a rectangle
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
```

```
% Updated: Nov 2013
function sup = Sup2D(x,y,w,h)
sup = [[0,0,w,0];[w,0,w,h];[w,h,0,h];[0,h,0,0]];
sup(:,[1,3]) = sup(:,[1,3])+x-0.5*w;
sup(:,[2,4]) = sup(:,[2,4])+y-0.5*h;
```

Sup2D generates a two-dimensional support (here rectangle).

2.4.13 SupCSup2D

```
% SupCSup2D
% test for connectivity between two 2D supports via fracture network
%
% Usage :
%    C = SupCSup2D(lines,La,sup1,sup2)
%
% input : lines     (n,4), fracture network
%         La        (n) cluster labels for lines
%         sup1      support 1, box,e.g., [0,1,0,1]
%         sup2      support 2
% output: C         true/false
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function C = SupCSup2D(lines,La,sup1,sup2)
xC1 = SupXLines2D(sup1,lines,La);                  %cluster info of sup1 from fnm
xC2 = SupXLines2D(sup2,lines,La);                  %cluster info of sup2 from fnm
C = ~isempty(intersect(xC1,xC2));                  %if they intersect?
```

SupCSup2D evaluates whether the two given supports are connected to each other or not. This is a pretty and robust implementation of connectivity assessment between the two supports. Note that the cluster information of fracture network (*La*) is used for the evaluation.

2.4.14 SupXLines2D

```
% SupXLines2D
% find cluster labels for a support intersecting 2D fracture network
%
% Usage :
%    xC = SupXLines2D(sup,lines,La)
%
% input : sup       (4), e.g., [0,1,0,1]
%         lines     (n,4)
%         La        cluster labels of fractures
% output: xC        intersected clusters
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function xC = SupXLines2D(sup,lines,La)
xC = unique(La(sum(ClipLines2D(lines,sup),2)>0));   %utilises clipping concept
```

SupXLines2D is used to determine all fracture clusters (including isolated fractures) intersecting a support.

2.4.15 SupXNLines2D

```
% SupXNLines2D
% finds number of intersected 2D fractures by a 2D support
%
% Usage :
%    xN = SupXNLines2D(sup,lines)
%
% input : sup       (4), e.g., [0,1,0,1]
%         lines     (n,4)
% output: xN        number of intersected lines
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function xN = SupXNLines2D(sup,lines)
xN = sum(sum(ClipLines2D(lines,sup),2)>0);         %utilises clipping concept
```

SupXNLines2D determines the total number of intersecting fractures for a support.

2.4.16 P21G

```
% P21G
% evaluates P21 measure of 2D fracture network (regular grid)
%
% Usage :
%    [tLs,xLs,x,y] = P21G(lines,gn,gm)
%
% input : lines     (n,4)
%         gn        grid dimensions, horizontally
%         gm        grid dimensions, vertically
% output: tLs       (n) cell, total length
%         xLs       (n) cell, all lengths
%         x,y       extent for plotting
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function [tLs,xLs,x,y] = P21G(lines,gn,gm)
if nargin<3
    gn = 25;
    gm = gn;
end
w = 1/gn;                                          %cell width
h = 1/gm;                                          %cell height
xLs = cell(gm,gn);
tLs = zeros(gm,gn);
for i = 1:gm
    for j = 1:gn
        sup = [(j-1)*w,j*w,(i-1)*h,i*h];
        cls = ClipLines2D(lines,sup);
        lhs = Lengths2D(cls(sum(cls,2)>0,:));
```

```
        xLs(i,j) = {lhs};                               %store all lengths
        tLs(i,j) = sum(lhs);                            %total length per cell
    end
end
x = [w/2,1-w/2];
y = [h/2,1-h/2];
```

`P21G` is to compute **P21** measure based on regular grid cell sampling.

2.4.17 ConnectivityIndex2D

```
% ConnectivityIndex2D
% computes connectivity index (CI) on 2D fracture networks
%
% Usage :
%    [CI,x,y] = ConnectivityIndex2D(lines,La,gm,gn,cm,cn)
%
% input : lines     (n,4)
%         La        cluster labels (n)
%         gm        grid dimension vertically
%         gn        grid dimension horizontally
%         cm        target cell i index
%         cn        target cell j index
% output: CI        (gm,gn)
%         x,y       extents
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function [CI,x,y] = ConnectivityIndex2D(lines,La,gm,gn,cm,cn)
w = 1/gn;
h = 1/gm;
CI = zeros(gm,gn);
xCs = cell(gm,gn);
for i = 1:gm                                   %extract cluster info for all supports
    for j = 1:gn
        sup = [(j-1)*w,j*w,(i-1)*h,i*h];
        xCs(i,j) = {SupXLines2D(sup,lines,La)};   %store
    end
end
xC1 = xCs{cm,cn};                               %target cell cluster info
if ~isempty(xC1)                                %if target cell is not isolated
    for i = 1:gm
        for j = 1:gn
            com = intersect(xC1,xCs{i,j});      %any common cluster?
            if ~isempty(com); CI(i,j) = 1; end
        end
    end
end
x = [w/2,1-w/2];                                %for usage in imagesc(x,y,CI)
y = [h/2,1-h/2];
```

`ConnectivityIndex2D` evaluates the connectivity index (**CI**) on a two-dimensional fracture network based on grid cell sampling.

2.4.18 ConnectivityField2D

```
% ConnectivityField2D
% computes connectivity field (CF) for 2D fracture network
%
% Usage :
%    [CF,x,y] = ConnectivityField2D(lines,La,gm,gn,rm,rn)
%
% input : lines     (n,4)
%         La        cluster labels (n)
%         gm        grid dimension vertically
%         gn        grid dimension horizontally
%         rm        range for cells vert.
%         rn        range for cells horiz.
% output: CF        (gm,gn)
%         x,y       extents
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Oct 2013
function [CF,x,y] = ConnectivityField2D(lines,La,gm,gn,rm,rn)
w = 1/gn;
h = 1/gm;
sm = length(rm);
sn = length(rn);
CF = zeros(gm,gn);
xCs = cell(gm,gn);
for i = 1:gm
    for j = 1:gn
        sup = [(j-1)*w,j*w,(i-1)*h,i*h];
        xCs(i,j) = {SupXLines2D(sup,lines,La)};   %all supports' clusters information
    end
end
for i = 1:sm                                      %outer loop for all target cells
    for j = 1:sn
        xC1 = xCs{i,j};
        k = 0;
        for ii = 1:gm
            for jj = 1:gn
                com = intersect(xC1,xCs{ii,jj});
                if ~isempty(com); k = k+1; end    %record number of connected cells
            end
        end
        CF(rm(i),rn(j)) = k;
    end
end
x = [w/2,1-w/2];                                  %for usage in imagesc(x,y,CF)
y = [h/2,1-h/2];
```

ConnectivityField2D computes connectivity field (CF) on a two-dimensional fracture network based on grid cell sampling.

2.4.19 BreakLinesX2D

```
% BreakLinesX2D
% breaks 2D lines at their intersection points
%
% Usage :
```

```
%    olins = BreakLinesX2D(lines)
%
% input : lines     (n,4)
% output: olins     cell
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function olins = BreakLinesX2D(lines)
[xts,ids,~] = LinesX2D(lines);
gxs = Group(xts,ids,size(lines,1));
n = size(gxs,1);
olins = cell(n,1);
for i=1:n
    ots = SortPoints2D([gxs{i};lines(i,1:2);lines(i,3:4)]);
    olins{i} = [ots(1:end-1,:),ots(2:end,:)];
end
```

BreakLinesX2D breaks given lines at their intersection points. This function is used for generating backbone structure.

2.4.20 Rotate2D

```
% Rotate2D
% rotates 2D points about a center by given angle
%
% Usage :
%    ots = Rotate2D(pts,cnt,ang)
%
% input : pts       (n,2)
%         cnt       center of rotation (2)
%         ang       angle of rotation
% output: ots       (n,2)
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function ots = Rotate2D(pts,cnt,ang)
if nargin<3; ang = 0; end
if nargin<2; cnt = [0,0]; end
ots = [pts(:,1)-cnt(1),pts(:,2)-cnt(2)]*[cos(ang),sin(ang);-sin(ang),cos(ang)];
ots = [ots(:,1)+cnt(1),ots(:,2)+cnt(2)];
```

Rotate2D applies rotation to points about a given centre and angle (in radian).

2.4.21 SortPoints2D

```
% SortPoints2D
% sorts 2D points topologically
%
% Usage:
%    ots = SortPoints2D(pts)
%
% input : pts       (n,2)
% output: ots       (n,2)
```

```
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function ots = SortPoints2D(pts)
p1 = FarthestPoints(pts);                          %farthest point as ref
d = pdist2(pts,p1);
[~,idx] = sort(d);
ots = pts(idx,:);
```

SortPoints2D sorts points based on their topological arrangement, that is, they can be connected to create a polyline without self-intersection.

2.4.22 Isolated2D

```
% Isolated2D
% checks if a line at index i from lines is isolated
%
% Usage :
%    b = Isolated2D(i,lines,tol)
%
% input : i          index
%         lines      (n,4)
%         tol        tolerance, default=1e-9
% output: b          boolean
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Oct 2013
function b = Isolated2D(i,lines,tol)
if nargin<3; tol = 1e-9; end
p1s = lines(:,1:2);
p2s = lines(:,3:4);
pts = [p1s;p2s];
b = (Occurrence(p1s(i,:),pts,tol)<=1) | (Occurrence(p2s(i,:),pts,tol)<=1);
```

Isolated2D determines if a line is isolated from (unconnected to) others in fracture network.

2.4.23 Backbone2D

```
% Backbone2D
% returns backbone of 2D fracture network
%
% Usage :
%    bbn = Backbone2D(lines,process,rgn,tol)
%
% input : lines      (n,4)
%         process    if true to break lines into segments
%         rgn        the region of study as polyline (k,4)
%         tol        tolerance
% output: bbn        (m,4)
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
```

```
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function bbn = Backbone2D(lines,process,rgn,tol)
if nargin<4; tol = 1e-9; end
if (nargin<3) || isempty(rgn)
    rgn = [0,0,1,0; 1,0,1,1; 1,1,0,1; 0,1,0,0];   %default: a unit square
end
if nargin<2; process = false; end
if process
    bbn = Stack(BreakLinesX2D([lines;rgn]));
else
    bbn = lines;
end
while true
    B = IsolatedLines2D(bbn,tol);
    if ~any(B); break; end                          %break if no isolated line anymore
    bbn = bbn(~B,:);                                %update backbone
end
```

Backbone2D extracts the backbone structure of given fracture network.

2.4.24 IsolatedLines2D

```
% IsolatedLines2D
% check isolation for all 2D fracture lines
%
% Usage :
%    B = IsolatedLines2D(lines,tol)
%
% input : lines     (n,4)
%         tol
% output: B         (n) boolean
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function B = IsolatedLines2D(lines,tol)
if nargin<3; tol = 1e-9; end
p1s = lines(:,1:2);
p2s = lines(:,3:4);
pts = [p1s;p2s];
n = size(lines,1);
B = false(n,1);
for i=1:n
    B(i) = (Occurrence(p1s(i,:),pts,tol)<=1) | (Occurrence(p2s(i,:),pts,tol)<=1);
end
```

IsolatedLines2D checks isolation for all lines in fracture network.

2.4.25 BackboneToNodesEdges2D

```
% BackboneToNodesEdges2D
% returns (nodes,edges) extracted from 2D backbone
%
```

```
% Usage :
%    [nodes,edges] = BackboneToNodesEdges2D(bbn)
%
% input : bbn       backbone
% output: nodes
%         edges
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function [nodes,edges] = BackboneToNodesEdges2D(bbn)
nodes = dict();
edges = bbn;
for i=1:size(bbn,1)
    p1 = bbn(i,1:2);                    %endpoints as keys for nodes
    p2 = bbn(i,3:4);
    nodes(p1) = unique([nodes(p1),i]);   %indices of edges associated with nodes
    nodes(p2) = unique([nodes(p2),i]);
end
```

BackboneToNodesEdges2D produces list of nodes and edges (*graph structure*) from given backbone.

2.4.26 Expand2D

```
% Expand2D
% expands a 2D matrix by nx,ny
%
% Usage :
%    Y = Expand2D(X,nx,ny)
%
% input : X         2D matrix
%         nx,ny     increase in dimensions
% output: Y         expanded matrix
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Oct 2013
function Y = Expand2D(X,nx,ny)
if nargin==1; nx = 1; ny = 1; end
[m,n] = size(X);
Y = zeros(m+ny,n+nx)+X(end,end);
Y(1:m,1:n) = X;
```

Expand2D expands given 2D matrix by nx and ny.

2.4.27 Resize2D

```
% Resize2D
% resizes 2D matrix in shape (m,n)
%
% Usage :
%    B = Resize2D(A,m,n)
%
% input : A         input 2d matrix
```

```
%         m,n       desired dimensions
% output: B         resized matrix (m,n)
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Oct 2013
function B = Resize2D(A,m,n)
[a,b] = size(A);
if nargin<2; m = 70; end
if nargin<3; n = m; end
zm = m/a;
zn = n/b;
B = A(floor((0:end*zm-1)/zm)+1,floor((0:end*zn-1)/zn)+1);
```

Resize2D resizes a given matrix into shape of (m,n).

2.4.28 DrawLines2D

```
% DrawLines2D
% draws quickly 2D fracture lines
%
% Usage :
%    DrawLines2D(lines,La,rgb,age)
%
% input : lines     (n,4)
%         La        cluster labels
%         rgb       color, default=[0,0,0]
%         age       axes, grid settings, check `Titles2D`
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function DrawLines2D(lines,La,rgb,age)
if nargin<4; age = '=['; end
if nargin<3; rgb = [0,0,0]; end
if nargin<2; La = []; end
if isempty(La) || all(La==0)
    [X,Y] = LinesToXYnan2D(lines);
    if all(rgb==0); rgb = [0,0,0]; end
    plot(X,Y,'-','Color',rgb)
else
    [X,Y] = LinesToXYnan2D(lines(La<0,:));        %isolated fractures
    plot(X,Y,'-','Color',[0.5,0.5,0.5])
    hold on
    for i=1:max(La)                               %fracture clusters
        [X,Y] = LinesToXYnan2D(lines(La==i,:));
        plot(X,Y,'-','Color',rand(3,1),'LineWidth',1.5);
    end
end
Titles2D(age)
```

DrawLines2D draws efficiently large number of fracture lines.

2.4.29 LinesToXYnan2D

```
% LinesToXYnan2D
% builds [X,Y,nan] from lines for `plot` to highest efficiency
%
% Usage :
%    [X,Y] = LinesToXYnan2D(lines)
%
% input : lines     (n,4)
% output: X         x coordinates of lines
%         Y         y coordinates of lines
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function [X,Y] = LinesToXYnan2D(lines)
n = size(lines,1);
X = [lines(:,[1,3]),NaN(n,1)]';
Y = [lines(:,[2,4]),NaN(n,1)]';
```

LinesToXYnan2D builds $(data, NaN)$ structure for increased efficiency in drawing large number of 2D lines.

2.4.30 ExpandAxes2D

```
% ExpandAxes2D
% expands current axes by rate (+:relative,-:absolute)
%
% Usage:
%    ExpandAxes2D(rate)
%
% input : rate      to expand axes symmetrically
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function ExpandAxes2D(rate)
a = axis;
if rate>0                                          %relative expansion
    fx = a(2)-a(1);
    fy = a(4)-a(3);
else                                               %absolute expansion
    fx = -1;
    fy = -1;
end
axis([a(1)-rate*fx,a(2)+rate*fx,a(3)-rate*fy,a(4)+rate*fy]);
```

ExpandAxes2D expands two-dimensional axes by factor of given rate.

2.4.31 Titles2D

```
% Titles2D
% sets titles, grid etc for current 2D axes
%
```

```
% Usage :
%    Titles2D(tl,xl,yl,ext,rgn,age)
%
% input : tl        title
%         xl        xlabel
%         yl        ylabel
%         ext       limits of axes
%         rgn       region of study
%         age       aspect,limits,grid switches
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function Titles2D(age,tl,xl,yl,ext,rgn)
if nargin<6; rgn = [0,1,0,1]; end
if nargin<5; ext = [-0.03,1.03,-0.03,1.03]; end
if nargin<4; yl = 'Y'; end
if nargin<3; xl = 'X'; end
if nargin<2; tl = ''; end
if nargin<1; age = '=['; end
hold on
drawBox(rgn,'k-','LineWidth',1.5)
if isempty(strfind(age,'-'))
    title(tl)
    xlabel(xl);
    ylabel(yl);
else
    axis off
end
box on
if strfind(age,'=')>0; axis image; end
if strfind(age,'+')>0; grid on; end
if strfind(age,'[')>0; axis(ext); end
hold off
```

Titles2D simplifies setting the titles, labels, grid, extent and region information for current 2D axes.

2.5 Functions for Three Dimensional Fracture Networks

2.5.1 RandPoly3D

```
% RandPoly3D
% generates randomly shaped and distributed 3D polygons
%
% Usage :
%    plys = RandPoly3D(n,dax,day,daz)
%
% input : n         number of polygons
%         dax,day,daz rotation angle range around X, Y and Z axes
% output: ply       (n,4,3)
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Oct 2013
```

```
function plys = RandPoly3D(n,dax,day,daz)
if nargin<4; daz = 2*pi; end
if nargin<3; day = 2*pi; end
if nargin<2; dax = 2*pi; end
if nargin<1; n = 1; end
plys = zeros(n,4,3);
for i=1:n
    r = rand(4,1);
    ply = [r(1),0,0; 0,r(2),0; r(3),1,0; 1,r(4),0];   %random polygon with 4 vertices
    cnt = polygonCentroid3d(ply);
    T = composeTransforms3d(...
        createRotationOx(cnt,rand*dax),...
        createRotationOy(cnt,rand*day),...
        createRotationOz(cnt,rand*daz));
    plys(i,:,:) = transformPoint3d(ply(:,1),ply(:,2),ply(:,3),T);
end
```

RandPoly3D generates three-dimensional polygons following the given angle variations about triple axes.

2.5.2 GenFNM3D

```
% GenFNM3D
% generates 3D fracture network
%
% Usage:
%    [clys,plys] = GenFNM3D(n,dax,day,daz,s,rgn)
%
% input : n          number of polygons
%         dax,day,daz rotation angle range around X, Y and Z axes
%         s          scale
%         rgn        region of study, default= unit cube
% output: clys       (n), cell, clipped polygons by region of study rgn
%         plys       (n), cell
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function [clys,plys] = GenFNM3D(n,dax,day,daz,s,rgn)
if nargin<6; rgn = [0,1,0,1,0,1]; end
if nargin<5; s = 0.25; end
if nargin<4; daz = pi; end
if nargin<3; day = pi; end
if nargin<2; dax = pi; end
if nargin<1; n = 1; end
plys = cell(n,1);
for i=1:n
    ply = [rand-0.5,-0.5,0; -0.5,rand-0.5,0; rand-0.5,0.5,0; 0.5,rand-0.5,0];
    pt = rand(3,3);
    T = composeTransforms3d(...
        createRotationOx([0,0,0],(2*rand-1)*dax),...
        createRotationOy([0,0,0],(2*rand-1)*day),...
        createRotationOz([0,0,0],(2*rand-1)*daz),...
        createScaling3d(s,s,s),...
        createTranslation3d(pt(1),pt(2),pt(3)));
    plys{i} = transformPoint3d(ply(:,1),ply(:,2),ply(:,3),T);
end
clys = ClipPolys3D(plys,rgn);
```

GenFNM3D synthesises a three-dimensional fracture network (polygonal shape fractures).

2.5.3 Sup3D

```
% Sup3D
% creates a 3D support (box)
%
% Usage:
%    sup = Sup3D(cnt,dim)
%
% input : cnt       a point(3)
%         dim       [width,height,depth]
% output: sup       cell, six sides of a cube
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function sup = Sup3D(cnt,dim)
if nargin<2; dim = [1,1,1]; end
if nargin<1; cnt = [0.5,0.5,0.5]; end
sup = cell(6,1);
sup{1} = [0,0,0; 0,0,1; 0,1,1; 0,1,0];    %left
sup{2} = [1,0,0; 1,0,1; 1,1,1; 1,1,0];    %right
sup{3} = [0,0,0; 1,0,0; 1,1,0; 0,1,0];    %bottom
sup{4} = [0,0,1; 1,0,1; 1,1,1; 0,1,1];    %top
sup{5} = [0,0,0; 0,0,1; 1,0,1; 1,0,0];    %front
sup{6} = [0,1,0; 0,1,1; 1,1,1; 1,1,0];    %back
d = cnt-[0.5*dim(1),0.5*dim(2),0.5*dim(3)];
for i=1:6
    sup{i}(:,1) = sup{i}(:,1)*dim(1)+d(1);
    sup{i}(:,2) = sup{i}(:,2)*dim(2)+d(2);
    sup{i}(:,3) = sup{i}(:,3)*dim(3)+d(3);
end
```

Sup3D creates 3D cubic support.

2.5.4 ClipPolys3D

```
% ClipPolys3D
% clips 3D polygons by a box
%
% Usage :
%    clys = ClipPolys3D(plys,box)
%
% input : plys      cell(n)
% output: clys      cell(m)
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function clys = ClipPolys3D(plys,box)
if nargin<2; box = [0,1,0,1,0,1]; end
x1 = box(1); x2 = box(2); y1 = box(3);
y2 = box(4); z1 = box(5); z2 = box(6);
```

```
xy1 = createPlane([x1,y1,z1],[0,0,-1]);
xy2 = createPlane([x1,y1,z2],[0,0,1]);
xz1 = createPlane([x1,y1,z1],[0,-1,0]);
xz2 = createPlane([x1,y2,z1],[0,1,0]);
yz1 = createPlane([x1,y1,z1],[-1,0,0]);
yz2 = createPlane([x2,y1,z1],[1,0,0]);
n = size(plys,1);
clys = cell(n,1);
for i=1:n
    ply = plys{i};
    ply = clipConvexPolygon3dHP(ply,xy1);
    ply = clipConvexPolygon3dHP(ply,xy2);
    ply = clipConvexPolygon3dHP(ply,xz1);
    ply = clipConvexPolygon3dHP(ply,xz2);
    ply = clipConvexPolygon3dHP(ply,yz1);
    ply = clipConvexPolygon3dHP(ply,yz2);
    if all(ply(1,:)==ply(end,:)); ply = ply(1:end-1,:); end
    clys{i} = ply;
end
```

ClipPolys3D clips 3D polygons by given cube.

2.5.5 PolysX3D

```
% PolysX3D
% finds all intersections between 3D polygons
%
% Usage:
%    [xts,ids,La] = PolysX3D(plys)
%
% input : plys      cell(n)
% output: xts       cell
%         ids       cell
%         La        cluster labels
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function [xts,ids,La] = PolysX3D(plys)
n = size(plys,1);
m = n*(n-1)/2;
xts = cell(m,1);
ids = cell(m,1);
k = 0;
for i = 1:n-1
    for j = i+1:n
        xpt = PolyXPoly3D(plys{i},plys{j});
        if isempty(xpt); continue; end
        k = k+1;
        xts{k} = xpt;                          %intersection points
        ids{k} = int32([i,j]);                 %intersecting lines indices
    end
end
xts = xts(1:k);
ids = ids(1:k);
La = Labels(Clusters(ids),n);                  %fracture cluster labels
```

PolysX3D conducts intersection analysis for given 3D fractures resulting in information of intersection points, inter-connected fracture indices and fracture clusters.

2.5.6 PolysXPolys3D

```
% PolysXPolys3D
% finds intersection (points,indices) between two sets of 3D polygons
%
% Usage:
%    [xts,pts,ids] = PolysXPolys3D(plys1,plys2)
%
% input : plys1     cell of polygons
%         plys2     cell of polygons
% output: xts       (k,3), cell
%         pts       stacked intersection points, i.e., (k,2)
%         ids       indices, cell
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function [xts,pts,ids] = PolysXPolys3D(plys1,plys2)
m = length(plys1);
n = length(plys2);
xts = cell(m,1);
ids = [];
u = 0;
for i=1:m
    pts = cell(n,1);
    idx = zeros(n,1);
    k = 0;
    for j=1:n
        xpt = PolyXPoly3D(plys1{i},plys2{j});     %intersecting
        if isempty(xpt); continue; end
        k = k+1;
        pts{k} = xpt;                              %points
        idx(k) = j;                                %polygon index
    end
    if k==0; continue; end
    u = u+1;
    xts{u} = pts(1:k);
    ids = union(ids,idx(1:k));                     %indices
end
xts = xts(1:u);
pts = zeros(0,3);
k = 0;
for i=1:u
    cps = xts{i};                                  %stacking all intersection points
    for j=1:size(cps,1)
        ets = cps{j};
        for w=1:size(ets,1)
            k = k+1;
            pts(k,:) = ets(w,:);
        end
    end
end
```

PolysXPolys3D assesses inter-connection between two sets of three-dimensional fractures. It can be used for example to find connectivity information between a three-dimensional support (e.g., cube) and a fracture network.

2.5.7 PolyXPoly3D

```
% PolyXPoly3D
% finds intersection points between two 3D polygons
%
% Usage :
%    xts = PolyXPoly3D(ply1,ply2)
%
% input : ply1      (n,3)
%         ply2      (m,3)
% output: xts       (k,3)
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function xts = PolyXPoly3D(ply1,ply2)
edges = [ply1,circshift(ply1,[-1,0])];            %create edges
pln = createPlane(ply2(1:3,:));
xts = intersectEdgePlane(edges,pln);
xts = xts(sum(isnan(xts),2)==0,:);
if ~isempty(xts)
    pts = planePosition(ply2,pln);
    its = planePosition(xts,pln);
    ins = xor(isPointInPolygon(its,pts),polygonArea(pts)<0);
    xts = xts(ins,:);
end
```

PolyXPoly3D results in intersection points between two three-dimensional polygons.

2.5.8 SupCSup3D

```
% SupCSup3D
% checks if two 3D suports are connected
%
% Usage :
%    out = SupCSup3D(sup1,sup2,plys,La)
%
% input : sup1      cell of polygons
%         sup2      cell of polygons
%         plys      cell of polygons
%         La        cluster labels
% output: out       boolean
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function out = SupCSup3D(sup1,sup2,plys,La)
[~,~,ids1] = PolysXPolys3D(sup1,plys);            %fractures intersection indices
```

```
[~,~,ids2] = PolysXPolys3D(sup2,plys);
if isempty(ids1) || isempty(ids2)
    out = false;
else
    out = ~isempty(intersect(La(ids1),La(ids2)));
end
```

SupCSup3D evaluates connectivity between two three-dimensional supports (e.g., cubes).

2.5.9 BBox3D

```
% BBox3D
% finds min and max of polygon in each axis X, Y and Z
%
% Usage :
%    [mins,maxs] = BBox3D(plys)
%
% input : plys      cell
% output: mins      min values for X, Y and Z
%         maxs      max values for X, Y and Z
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function [mins,maxs] = BBox3D(plys)
mins = min(cell2mat(cellfun(@min,plys,'UniformOutput',false)));
maxs = max(cell2mat(cellfun(@max,plys,'UniformOutput',false)));
```

BBox3D finds bounding box for given three-dimensional points / polygons.

2.5.10 Expand3D

```
% Expand3D
% expands a 3D matrix by nx,ny,nz
%
% Usage :
%    Y = Expand3D(X,nx,ny,nz)
%
% input : X         3D matrix
%         nx,ny,nz  increase in dimensions
% output: Y         expanded matrix
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function Y = Expand3D(X,nx,ny,nz)
if nargin==1; nx = 1; ny = 1; nz = 1; end
[m,n,o] = size(X);
Y = zeros(m+ny,n+nx,o+nz)+X(end,end,end);
Y(1:m,1:n,1:o) = X;
```

Expand3D expands a given three-dimensional matrix.

2.5.11 Resize3D

```
% Resize3D
% resizes 3D matrix into shape (m,n,o)
%
% Usage :
%    B = Resize3D(A,m,n,o)
%
% input : A         input 3d matrix
%         m,n,o     desired dimensions
% output: B         resized matrix (m,n,o)
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function B = Resize3D(A,m,n,o)
[a,b,c] = size(A);
if nargin<2; m = 70; end
if nargin<3; n = m; end
if nargin<4; o = n; end
zm = m/a;
zn = n/b;
zo = o/c;
B = A(floor((0:end*zm-1)/zm)+1,floor((0:end*zn-1)/zn)+1,floor((0:end*zo-1)/zo)+1);
```

Resize3D resizes three-dimensional matrix to a given shape. This function is useful to combine different shapes of input matrices to generate E-Type maps, for example.

2.5.12 SaveToFile3D

```
% SaveToFile3D
% saves 3D data into text file
%
% Usage :
%    SaveToFile3D(fname,x)
%
% input : fname     filename
%         x         3D matrix
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function SaveToFile3D(fname,x,index)
if nargin<3; index = false; end
if ~index
    dlmwrite(fname,x,'delimiter',',')             %just data, no indices
else
    fut = fopen(fname,'w');
    [m,n,o] = size(x);
    for i=1:m
        for j=1:n
            for k=1:o
                fprintf(fut,sprintf('%d, %d, %d, %0.6f\n',i,j,k,x(i,j,k)));
            end
```

```
        end
    end
    fclose(fut);
end
```

SaveToFile3D exports three-dimensional data to an *ASCII* file.

2.5.13 SavePolysToVTK3D

```
% SavePolysToVTK3D
% saves 3D polygons as standard VTK file (ASCII format)
%
% Usage :
%    SavePolysToVTK3D(plys,colors,fname)
%
% input : plys      cell
%         colors    (x,3[4]), with or without alpha values
%         fname     filename
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function SavePolysToVTK3D(plys,colors,fname)
nply = length(plys);
plyn = cellfun(@length,plys,'UniformOutput',false);
npnt = sum(cell2mat(plyn));
fut = fopen(fname,'w');
fprintf(fut,'# vtk DataFile Version 3.0\n');      %header...
fprintf(fut,'Polygons by Younes Fadakar Alghalandis\n');
fprintf(fut,sprintf('ASCII\nDATASET POLYDATA\nPOINTS %d float\n',npnt));
for i=1:nply
    ply = plys{i};
    fprintf(fut,sprintf('%0.6f %0.6f %0.6f\n',ply'));
end
fprintf(fut,sprintf('POLYGONS %d %d\n',nply,npnt+nply));
k = 0;
for i=1:nply                                      %polygon data
    fprintf(fut,strcat(sprintf('%d ',plyn{i},k:k+plyn{i}-1),'\n'));
    k = k+plyn{i};
end
fprintf(fut,sprintf('POINT_DATA %d\n',npnt));
fprintf(fut,'COLOR_SCALARS lut 4\n');
if size(colors,2)==3                              %if no alpha provided, set all to 1
    colors(:,4) = 1;
end
for i=1:nply
    for j=1:plyn{i}
        fprintf(fut,strcat(sprintf('%0.3f ',colors(i,:)),'\n'));
    end
end
fclose(fut);
fprintf(1,'Polygons were saved as file %s.\n',fname);   %report on screen
```

SavePolysToVTK3D exports polygons to a file as **VTK** ASCII format. VTK format is industry standard for three-dimensional data and can be visualised and manipulated by many available free software applications such as ***ParaView***.

2.5.14 SetAxes3D

```
% SetAxes3D
% sets and adjust axes into 3D view
%
% Usage :
%    SetAxes3D(mins,maxs)
%
% input : mins,maxs min and max values of X, Y and Z axes
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function SetAxes3D(mins,maxs)
if nargin==0;
    mins = [0,0,0];                                  %defaults
    maxs = [1,1,1];
end
hold on
plot3([mins(1),maxs(1)],[mins(2),mins(2)],[mins(3),mins(3)],'-',...
      'LineWidth',1.5,'Color',[0.7,0,0]);
plot3([mins(1),mins(1)],[mins(2),maxs(2)],[mins(3),mins(3)],'-',...
      'LineWidth',1.5,'Color',[0,0.7,0]);
plot3([mins(1),mins(1)],[mins(2),mins(2)],[mins(3),maxs(3)],'-',...
      'LineWidth',1.5,'Color',[0,0,0.7]);
text(0.5*(mins(1)+maxs(1)),mins(2),mins(3),'X','BackgroundColor',[0.7,0,0],'Color','w
')
text(mins(1),0.5*(mins(2)+maxs(2)),mins(3),'Y','BackgroundColor',[0,0.7,0],'Color','w
')
text(mins(1),mins(2),0.5*(mins(3)+maxs(3)),'Z','BackgroundColor',[0,0,0.7],'Color','w
')
camproj('perspective')
set(gca,'CameraPosition',[-1*maxs(1),-2*maxs(2),1.5*maxs(3)]);
axis(reshape([mins;maxs],1,[]));
axis image
grid on
box on
```

SetAxes3D sets the current view into three-dimensional perspective view with axes and labels automatically adjusted.

2.5.15 DrawPolys3D

```
% DrawPolys3D
% draws quickly 3D polygons
%
% Usage :
%    DrawPolys3D(plys,La,rgba,axes)
%
% input : plys      cell
%         rgba      [r,g,b,a]
%         clus      La
%         axes      if true to draw set axes and adjust into 3D
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
```

```
% Updated: Nov 2013
function DrawPolys3D(plys,La,rgba,axes)
if nargin<4; axes = true; end
if nargin<3; rgba = [0.5,0,0.1,0.5]; end
if nargin<2; La = []; end
hold on
cmap = colormap(jet);
for i=1:length(plys)
    ply = plys{i};
    if ~isempty(La) && La(i)<0                    %isolated fractures
        patch(ply(:,1),ply(:,2),ply(:,3),[0.5,0.5,0.5],'FaceAlpha',0.5,...
              'EdgeColor','none');
    else
        fvc = zeros(length(ply),3);
        if isempty(La)                             %no cluster labels info provided
            fvc(1,:) = rgba(1:3);
        else
            fvc(1,:) = cmap(int32(double(La(i))/double(max(La))*64),:);
        end
        h = patch(ply(:,1),ply(:,2),ply(:,3),0,'FaceAlpha',rgba(4));
        set(h,'FaceVertexCData',fvc);
    end
end
if axes
    [mins,maxs] = BBox3D(plys);                    %bounding box of polygons
    SetAxes3D(mins,maxs);
end
```

`DrawPolys3D` draws three-dimensional polygons (fracture network). If clusters' labels (*La*) were provided fractures will be colourised according to their associated clusters.

2.5.16 DrawSlices3D

```
% DrawSlices3D
% draws 3D slices of 3D volume data
%
% Usage :
%    h = DrawSlices3D(data,a,axes)
%
% input : data      3D array
%         a         transparency
%         axes      if true to set and adjust axes into 3D
% output: h         handle to slice objects
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function h = DrawSlices3D(data,a,axes)
if nargin<3; axes = true; end
if nargin<2; a = 1; end
[m,n,o] = size(data);
[x,y,z] = meshgrid(0:m,0:n,0:o);
h = slice(x,y,z,Expand3D(data),m/2,n/2,o/2);
shading flat
if a~=1
    if a<0
```

```
        set(h,'EdgeColor','none','FaceColor','interp');
        alpha(abs(a));
    else
        for i=1:length(h)
            set(h(i),'alphadata',get(h(i),'cdata'),'facealpha',a);
        end
    end
end
if nargout==0; clear h; end
if axes; SetAxes3D([m,n,o]); end
```

DrawSlices3D draws three-dimensional slices on the middle of each of axes.

2.5.17 VolRender3D

```
% VolRender3D
% view volume render of 3D volumetric data
%
% Usage :
%    VolRender3D(data,a,axes)
%
% input : data      array (m,n,o)
%         a         alpha factor
%         axes      if true to draw set axes and adjust into 3D
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function VolRender3D(data,a,axes)
if nargin<3; axes = true; end
if nargin<2; a = 1; end
[m,n,o] = size(data);
mdl = Vol3D('CData',data);
alphamap('rampup');
alphamap(a.*alphamap);
if axes; SetAxes3D([0,0,0],[m,n,o]); end
```

VolRender3D renders three-dimensional data (volumetric) with adjustable alpha (transparency value).

2.5.18 Vol3D

```
function [model] = vol3d(varargin)                          → Vol3D
By Woodford O, 2011
```

2.6 Generic Functions

2.6.1 Scale

```
% Scale
% scales (maps) X into range (a to b)
%
% Usage :
%     Y = Scale(X,a,b)
%
% input : X          any array
%         a          minimum bound of the output
%         b          maxmium bound of the output
% output: Y          same as X but mapped
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function Y = Scale(X,a,b)
if nargin<3; a = 0; b = 1; end
Y = double(X-min(X(:)))/double(range(X(:)))*(b-a)+a;
```

Scale scales data to a given bounds.

2.6.2 ToStruct

```
% ToStruct
% builds `struct` data type from data
%
% Usage :
%     S = ToStruct(data)
%
% input : data       any array
% output: S          struct
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function S = ToStruct(data)
S = struct();
m = size(data,1);
for i = 1:m
   S.(sprintf('S%d',i)) = data(i,:);
end
```

ToStruct provides “*struct*” format for the given data.

2.6.3 KDE

```
function [bandwidth,density,X,Y] = kde2d(data,n,MIN_XY,MAX_XY)    → KDE
By Botev Z.I @ botev@maths.uq.edu.au
```

KDE applies kernel density estimation.

2.6.4 Smooth

```
function [z,s,exitflag,Wtot] = smoothn(varargin)                → Smooth
By Garcia D @ http://www.biomecardio.com/matlab/smoothn.html
```

Smooth applies smoothing on the given data.

2.6.5 dict

```
classdef dict < handle                                           → dict
By Harriman D @ doug.harriman@gmail.com
```

dict provides "*dict*" structure.

2.6.6 Clusters

```
% Clusters
% clusters items based on common elements
%
% Usage :
%    C = Clusters(S)
%
% input : S         cell
% output: C         cell
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function C = Clusters(S)
if isempty(S); C = {}; return; end
while true
    m = length(S);
    united = zeros(m,1);                            %nothing is clustered yet
    C = cell(m,1);
    u = 0;
    for i = 1:m-1
        if united(i); continue; end;
        p = S{i};
        for j = i+1:m
            q = S{j};
            com = intersect(p,q);                   %common elements
            if ~isempty(com)
                united(j) = 1;
                p = union(p,q);
                S{i} = p;
            end
        end
        u = u+1;
        C{u} = p;
    end
    if ~united(m)                                   %copy the last item if not united
        u = u+1;
        C{u} = S{m};
```

```
    end
    C = C(1:u);
    if any(united)
        S = C;                                          %all are united, i.e., clustered
    else
        C = S;
        break
    end
end
```

Clusters determines fracture clusters in the fracture network by means of intersection indices (see also **LinesX2D**). This function is highly efficient and also generic for two- and three-dimensional fracture clustering.

2.6.7 CheckClusters

```
% CheckClusters
% checks if clusters are OK, have no missing common element
%
% Usage :
%    OK = CheckClusters(C)
%
% input : C          cell
% output: OK         boolean
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function OK = CheckClusters(C)
X = horzcat(C{:});
OK = (length(unique(X))==length(X));
```

CheckClusters checks cluster information for any inconsistency due to any remaining unclassified elements.

2.6.8 Labels

```
% Labels
% extracts labels from clusters
%
% Usage :
%    La = Labels(C,n)
%
% input : C          cell of Clusters
%         n          number of fractures
% output: La         labels for all fractures
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function La = Labels(C,n)
La = zeros(n,1);
```

```
for i = 1:length(C)
    La(C{i}) = i;
end
f = (La==0);                                        %isolated fractures
La(f) = -(1:sum(f));                                %relabeling
```

`Labels` assigns unique label for each cluster. Isolated fractures are assigned a unique negative label for each.

2.6.9 Relabel

```
% Relabel
% relabel cluster labels according to their number of elements
%
% Usage :
%    Ra = Relabel(La)
%
% input : La         labels
% output: Ra         relabeled output
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function Ra = Relabel(La)
k = max(La);                                        %highest label
frq = zeros(k,1,'int32');
for i=1:k                                           %frequency of each label
    frq(i) = sum(La==i);
end
[~,idx] = sort(frq);                                %sort based on their frequencies
Ra = La;
for i=1:k
    Ra(La==idx(i)) = i;                             %apply relabing
end
```

`Relabel` is to sort fracture cluster labels based on the cardinality of each cluster.

2.6.10 Stack

```
% Stack
% stacks values of cell, i.e., results in array
%
% Usage :
%    S = Stack(C)
%
% input : C          cell
% output: S          array
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function S = Stack(C)
S = cat(1,C{:});
```

Stack stacks data in the give cell structure and produces an array.

2.6.11 Group

```
% Group
% groups intersection indices and points
%
% Usage :
%    [gxs,gds] = Group(xts,ids,n)
%
% input : xts       intersection points (m,2)
%         ids       intersection indices (m,2)
%         n         number of fractures
% output: gxs,gds   (n) cell
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function [gxs,gds] = Group(xts,ids,n)
gds = cell(n,1);
gxs = cell(n,1);
for i=1:size(ids,1)
    I = ids(i,1);
    J = ids(i,2);
    gds{I} = [gds{I},J];
    gds{J} = [gds{J},I];
    gxs{I} = [gxs{I};xts(i,:)];
    gxs{J} = [gxs{J};xts(i,:)];
end
```

Group groups given intersection points based on their associated fracture indices.

2.6.12 FarthestPoints

```
% FarthestPoints
% finds two farthest points in a set of nD points
%
% Usage :
%    [p1,p2] = FarthestPoints(pts)
%
% input : pts       (n,2)
% output: p1,p2     two farthest points
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function [p1,p2] = FarthestPoints(pts)
[~,idx] = max(pdist(pts,'euclidean'));
[I,J] = PDistIndices(size(pts,1));
p1 = pts(I(idx),:);
p2 = pts(J(idx),:);
```

FarthestPoints finds two farthest points from each other in a given set of n-dimensional points.

2.6.13 PDistIndices

```
% PDistIndices
% finds indices of results from `pdist` function
%
% Usage :
%    [I,J] = PDistIndices(n)
%
% input : n         number of points
% output: I,J       indices

% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Oct 2013
function [I,J] = PDistIndices(n)
[I,J] = find(tril(ones(n),-1));
```

PDistIndices produces indices information for "*pdist*" function.

2.6.14 Occurrence

```
% Occurrence
% finds number of occurrence of a point in set of nD points
%
% Usage :
%    k = Occurrence(pt,pts,tol)
%
% input : pt        point
%         pts       points
%         tol       tolerance of distance
% output: k         occurrence number
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Oct 2013
function k = Occurrence(pt,pts,tol)
if nargin<3; tol = 1e-9; end
[m,n] = size(pts);
k = true(m,1);
if tol~=0
    for i=1:n
        k = k & (abs(pts(:,i)-pt(i))<tol);        %relative match
    end
else
    for i=1:n
        k = k & (pts(:,i)==pt(i));                %absolute match
    end
end
k = sum(k);
```

Occurrence determines occurrence of any point in a set of n-dimensional points.

2.6.15 ConnectivityMatrix

```
% ConnectivityMatrix
```

```
% computes connectivity matrix of fracture network
%
% Usage :
%    cm = ConnectivityMatrix(ids,n,full,mat,fnm)
%
% input : ids       intersection indices
%         n         number of fractures
%         full      if true returns full matrix
%         mat       if false sparse form of results
%         fnm       fracture network
% output: cm        (n,n)
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Oct 2013
function cm = ConnectivityMatrix(ids,n,full,mat,fnm)
if nargin==5                                    %to find indices if not provided
    if iscell(fnm)
        [~,ids,~] = PolysX3D(fnm);              %3D fracture network
    else
        [~,ids,~] = LinesX2D(fnm);              %2D fracture network
    end
    n = size(fnm,1);
end
if nargin<4; mat = true; end
if nargin<3; full = false; end
cm = zeros(n,n);
for i=1:size(ids,1)
    if iscell(ids)
        I = ids{i}(1);
        J = ids{i}(2);
    else
        I = ids(i,1);
        J = ids(i,2);
    end
    cm(I,J) = 1;
    if full; cm(J,I) = 1; end
end
if ~mat; cm = sparse(cm); end
```

ConnectivityMatrix generates connectivity matrix based on intersection indices for a fracture network. If fracture network was provided it applies intersection analysis to find intersection indices. The function accepts two- or three-dimensional fractures networks.

2.6.16 FullCM

```
% FullCM
% returns full form of connectivity matrix (cm)
%
% Usage :
%    fcm = FullCM(cm)
%
% input : cm        sparse/matrix of connectivity
% output: fcm       full matrix of cm
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
```

```
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function fcm = FullCM(cm)
if issparse(cm)
    fcm = full(cm);
else
    fcm = cm;
end
fcm = fcm+fcm';
```

FullCM builds full connectivity matrix based on sparse or triangular connectivity matrix.

2.6.17 FNMToGraph

```
% FNMToGraph
% creates Graph from fracture network
%
% Usage :
%    [G,cm] = FNMToGraph(ids,n,fnm)
%
% input : ids       intersection indices
%         n         number of fractures
%         fnm       fracture network 2D or 3D
% output: G         Graph
%         cm        connectivity matrix
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function [G,cm] = FNMToGraph(ids,n,fnm)
if nargin==3
    if iscell(fnm)
        [~,ids,~] = PolysX3D(fnm);                  %3D fracture network
    else
        [~,ids,~] = LinesX2D(fnm);                  %2D fracture network
    end
    if isempty(ids); G = empty; return; end
    n = size(fnm,1);
end
cm = ConnectivityMatrix(ids,n,false,false);
G = biograph(cm,num2str(linspace(1,n,n)'));        %Matlab graph structure
```

FNMToGraph generates graph structure based on intersection indices.

2.6.18 LoadColormap

```
% LoadColormap
% updates current colormap from file
%
% Usage :
%    LoadColormap(fname)
%
% input : fname     filename
```

```
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function LoadColormap(fname)
load(fname,'cmap');
set(gcf,'Colormap',cmap);
```

LoadColormap loads a given “*colormap*” file and applies it to current figure.

2.6.19 SaveColormap

```
% SaveColormap
% saves current colormap as file
%
% Usage :
%    SaveColormap(fname)
%
% input : fname     filename
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function SaveColormap(fname)
cmap = get(gcf,'Colormap');
save(fname,'cmap');
```

SaveColormap saves current “*colormap*” to a file.

2.6.20 SecondsToClock

```
% SecondsToClock
% converts seconds to clock format as string
%
% Usage :
%    clk = SecondsToClock(snd)
%
% input : snd       seconds
% output: clk       clock string
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function clk = SecondsToClock(snd)
h = floor(snd/3600);
m = floor((snd-(h*3600))/60);
s = rem(snd,3600)-m*60;
clk = sprintf('%02d:%02d:%05.2f',h,m,s);
```

SecondsToClock converts given seconds to “time (clock) format”.

2.6.21 Colorise

```
% Colorise
% returns colors based on given data
%
% Usage :
%    colors = Colorise(x,cmap)
%
% input : x          (n)
%         cmap       colormap
% output: colors     (64,3)
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function colors = Colorise(x,cmap)
if nargin<2; cmap = colormap(jet); end
y = int32(Scale(x,1,64));
colors = cmap(y,:);
```

`Colorise` maps given data into a specified "*colormap*".

2.6.22 ShowFNM

```
% ShowFNM
% shows 2D or 3D fracture network
%
% Usage :
%    ShowFNM(fnm,La)
%
% input : fnm        (n,4) for 2D or cell for 3D
%         La         cluster labels
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function ShowFNM(fnm,La)
if nargin<2; La = []; end
if iscell(fnm)
    cla
    DrawPolys3D(fnm,La);                    %3D fracture networks
else
    cla
    DrawLines2D(fnm,La);                    %2D fracture networks
end
```

`ShowFNM` visualises given two- or three-dimensional fracture network.

2.6.23 Round

```
% Round
% rounds x to an arbitrary (dp) decimal
%
% Usage :
%    y = Round(x,dp)
```

```
%
% input : x          any
%         dp         decimal point, default=no decimal
% output: y          rounded output
%
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Nov 2013
function y = Round(x,dp)
if nargin<2; dp = 1.0; end
y = round(x/dp)*dp;
```

Round rounds data up to a given precision.

2.7 Example Full Programs

2.7.1 Example: Simulation of 2D Connectivity Index

By means of the provided functions Connectivity Index (CI) can be easily evaluated for two-dimensional fracture network model. The following full program code demonstrates the required stages and setup. Figure 2.1 shows the resulting maps.

```
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Oct 2013

clear all
clc

%% Simulation for Connectivity Index (CI)
n = 500;
simN = 30;
gm = 25;                                    %grid dimension vertically
gn = 25;                                    %grid dimension horizontally
cm = int32(floor(gm/2)+1);
cn = int32(floor(gn/2)+1);
CI = zeros(gm,gn);
tic
kappa = 10;
for i = 1:simN
    lines = GenFNM2D(n,3*pi/4,kappa,0.05,0.5);
    La = LinesToClusters2D(lines);
    ci = ConnectivityIndex2D(lines,La,gm,gn,cm,cn);
    CI = CI+ci;
    fprintf(1,'Real#:%04d, Total Elapsed Time:<%s>\n',i,SecondsToClock(toc));
end
CI = CI/simN;

%% Visualisation
clf
subplot(131);
```

```
[X,Y] = LinesToXYnan2D(lines);
plot(X,Y,'k-')
Titles2D('-=[')

colormap(jet);
subplot(132);
w = 0.5/gn;
h = 0.5/gm;
imagesc([w,1-w],[h,1-h],CI);
set(gca,'YDir','normal');
Titles2D('-=[');

subplot(133);
sCI = Smooth(CI);
contourf(linspace(0,1,25),linspace(0,1,25),sCI,20);
shading flat
Titles2D('-=[');

print('-dpng','-r600','CI2D Example');                      %to export result as image
```

As can be seen in Fig. 2.1 the simulated fracture network is anisotropic towards North-West, South-East. This is due to the setting in the code i.e., main orientation $3 \times \frac{pi}{4}$ and κ equal to 10. The resulting CI maps show the anisotropy.

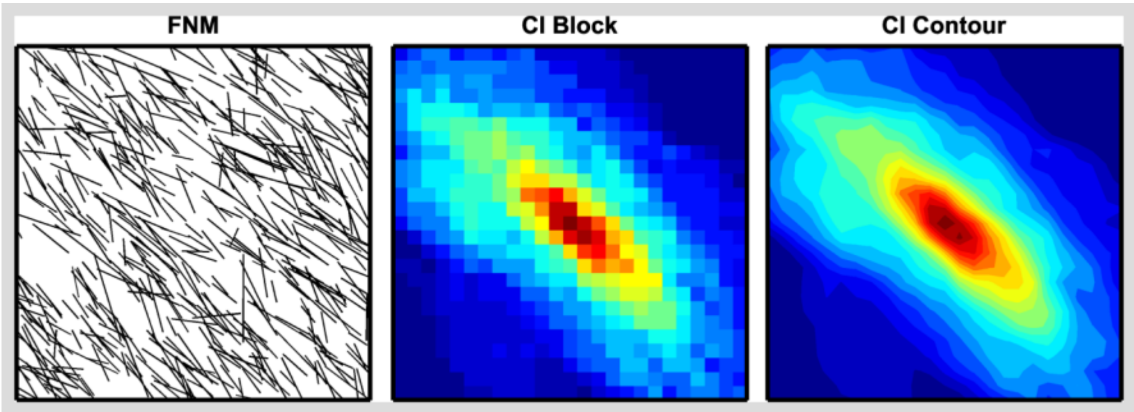

Figure 2.1: Results of evaluating CI on anisotropic fracture network.

2.7.2 Example: Two-dimensional Line Sampling

Line sampling is common stage for evaluating connectivity measures including CI and CF. The following full program code shows how generating line samples can be conducted by means of provided functions in the AFNM package.

```
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Oct 2013
```

```
clear all
clc

lines = RandLinesInPoly2D(1000,0.09,0.01,0,pi/2);      %generating line samples
[X,Y] = LinesToXYnan2D(lines);
clf
plot(X,Y,'k-')
Titles2D()
```

Note that any function provides quick help on its parameters and usage upon request by right-click in Matlab environment as shown for **RandLinesInPoly2D** in Fig. 2.2.

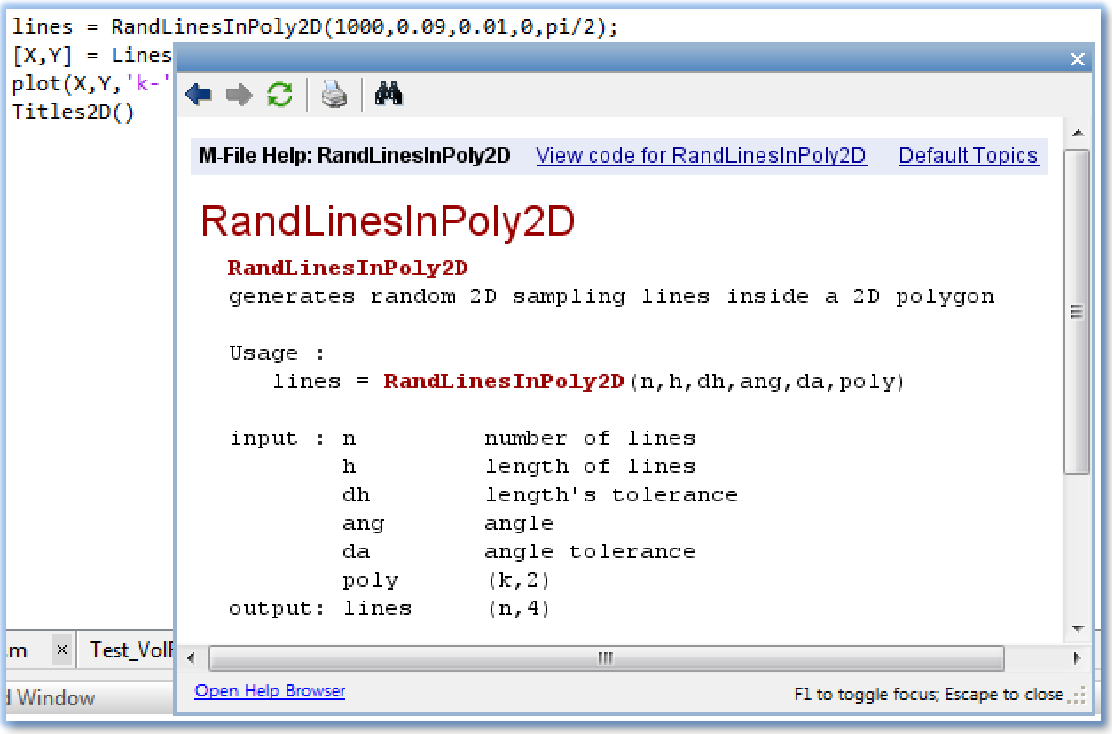

Figure 2.2: Quick help is available any time for all the functions.

Variation in the function **RandLinesInPoly2D** parameters results in various setting of line samples as the examples shown in Fig. 2.3.

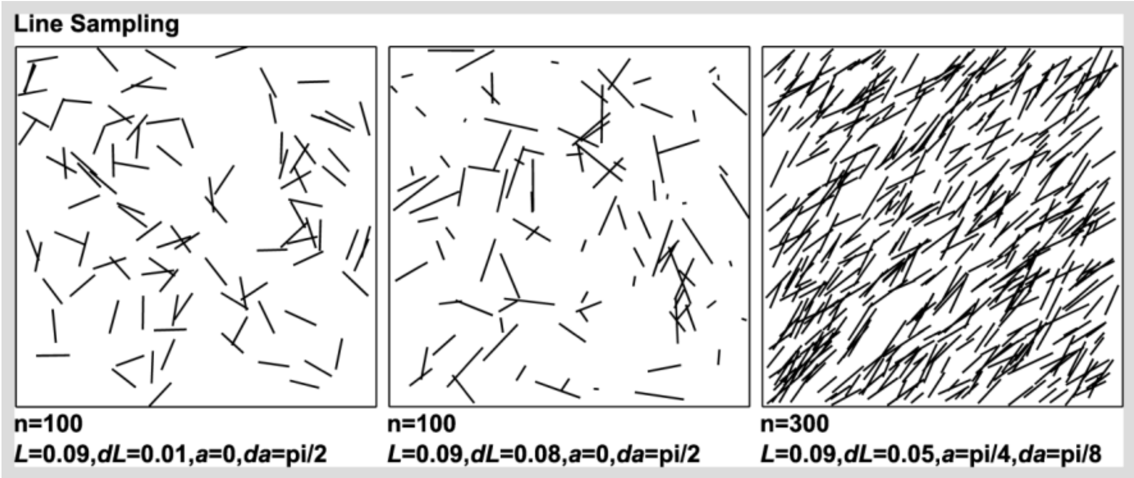

Figure 2.3: Various setting of line samples can be used in the determination of directional connectivity measures, for example.

2.7.3 Example: Simulation of 3D Connectivity Index

The evaluation of three-dimensional CI is simple and straightforward by means of the provided functions. The following full program code evaluates CI on a simulation of 30 realisations from a three-dimensional fracture network model. Each realisation includes 70 three-dimensional polygonal fractures which are randomly located and oriented in a unit cube.

```
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Oct 2013

clear all
clc

%% CI3D
w = 0.2; h = 0.2; d = 0.2;
m = 1/w; n = 1/h; o = 1/d;
i = floor(m/2)+1; j = floor(n/2)+1; k = floor(o/2)+1;
sup1 = Sup3D([i/m-w/2,j/n-h/2,k/o-d/2],[w,h,d]);
CI = zeros(m,n,o);
tic
for s=1:30                                          %simulation number
    plys = GenFNM3D(70,deg2rad(15),deg2rad(15),0);  %anisotropic
    [~,~,La] = PolysX3D(plys);
    for i=1:m
        for j=1:n
            for k=1:o
                pt = [i/m-w/2,j/n-h/2,k/o-d/2];
                sup2 = Sup3D(pt,[w,h,d]);
                CI(i,j,k) = CI(i,j,k)+SupCSup3D(sup1,sup2,plys,La);
```

```
            end
        end
    end
    fprintf(1,'Real.#: %d, Total Elapsed Time:<%s>\n',s,SecondsToClock(toc));
end

clf
DrawPolys3D(plys);
```

The resulting CI matrix which is three-dimensional volumetric data can be visualised by means of **VolRender3D** and **DrawSlices3D** (see also Fig. 2.4).

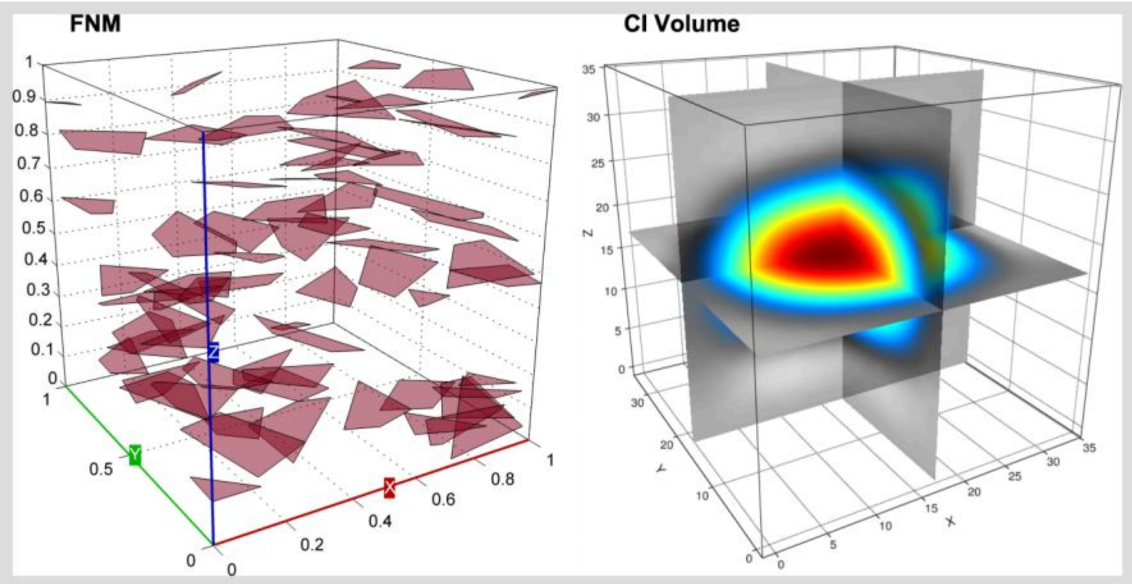

Figure 2.4: An example of CI3D on anisotropic fracture network.

2.7.4 Example: Intersection Analysis and Fracture Clusters

Conducting the intersection analysis and assessing fracture clusters are easy tasks by means of the provided functions in the AFNM as shown in the following full program code.

```
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Oct 2013

clear all
clc

%% Fracture Network models
fnm2 = GenFNM2D(150,0,0,0.01,0.7);
```

```
fnm3 = GenFNM3D(150,pi/3,pi/3,0,0.25);

%% Intersection Analysis >> Clusters
[xts2,ids2,La2] = LinesX2D(fnm2);
[xts3,ids3,La3] = PolysX3D(fnm3);

%% Visualisations
clf
subplot(121);
ShowFNM(fnm2,La2);                % a generic visualisation function which handles
                                  %     automatically 2D and 3D fractures and
subplot(122);                     %     and associated cluster data
ShowFNM(fnm3,La3);
```

The numeric results for the two-dimensional fracture network are as follows.

```
| xts1 =                | ids1 =          | La1 =     |
|    0.4720    0.7168   |      2      18  |       -1  |
|    0.2915    0.5062   |      2      61  |        1  |
|    0.4091    0.6434   |      2      73  |        1  |
|      :         :      |      :       :  |        :  |
```

For the three-dimensional fracture network the results are as follows.

```
| xts2 =                | ids2 =          | La2 =     |
|      [1x3 double]     |    [1x2 int32]  |       -1  |
|      [1x3 double]     |    [1x2 int32]  |        1  |
|      [2x3 double]     |    [1x2 int32]  |        2  |        %two intersection points
|           :           |         :       |        :  |
```

Fracture clusters are visualised by the help of clusters labels (*La*) passed to function **ShowFNM** (Fig. 2.5).

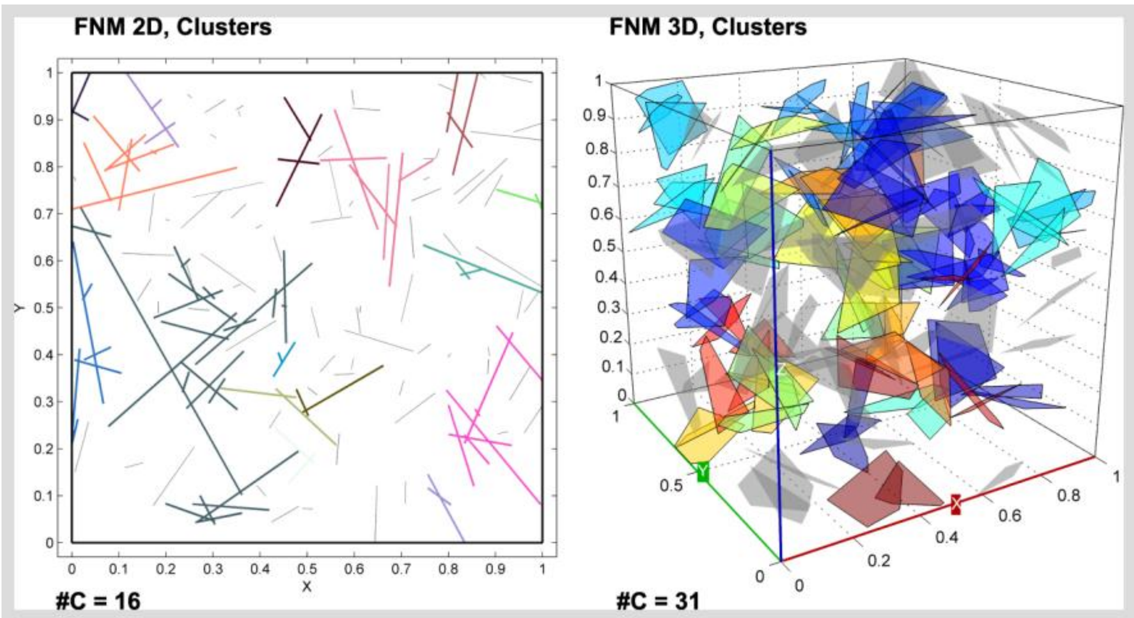

Figure 2.5: Fracture clustering is conducted on two- and three-dimensional fracture networks.

2.7.5 Example: Density Analysis

The density of fracture network can be found via different methods including density of fracture centroids (DFC, FCD). Here however the following program code demonstrates better solution which is based on cell sampling, i.e., fracture density (Fn). The function **Density2D** conducts the evaluation and results in density matrix which can be visualised as block or contour maps.

```
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Oct 2013

clear all
clc

%%
n = 200;
gn = 20;
sf = gn;
lines = GenFNM2D(n,0,0,0.05,0.5);
[dn,x,y] = Density2D(lines,gn,gn);

sdn = Smooth(dn,1);
[X,Y] = LinesToXYnan2D(lines);

clf
subplot(121);
imagesc(x,y,dn);
set(gca,'YDir','normal');
hold on
plot(X,Y,'k-','LineWidth',0.7)
Titles2D('-=[')

subplot(122);
contourf(0:1/(sf-1):1,linspace(0,1,sf),sdn,30);
shading flat
hold on
plot(X,Y,'k-','LineWidth',0.7)
Titles2D('-=[')
```

The output of the above program code is shown in Fig. 2.6. The same concept can be used for three-dimensional fracture networks.

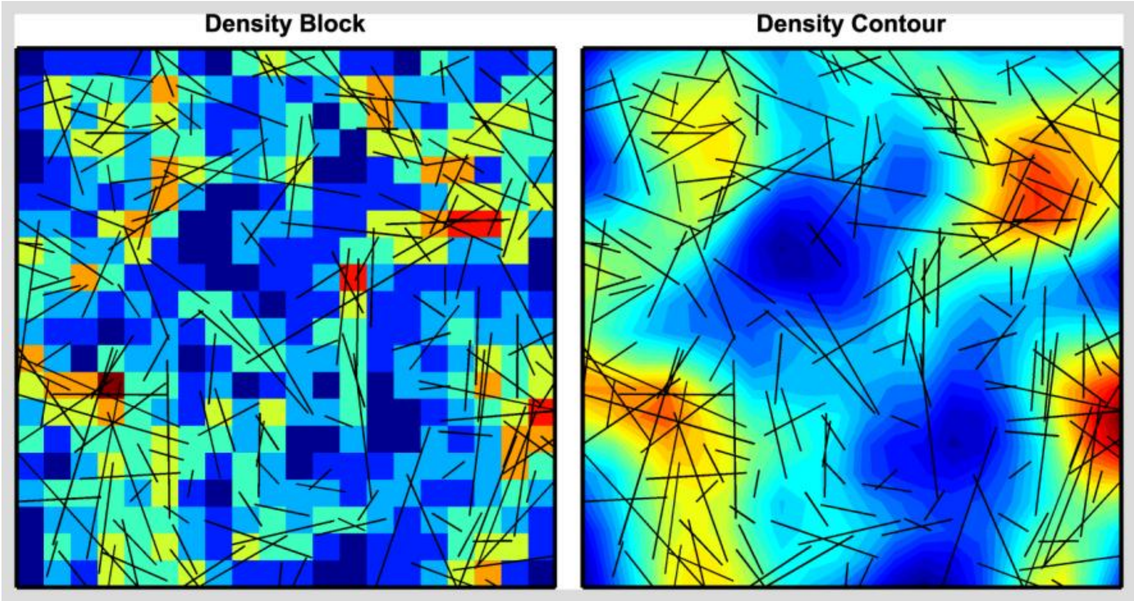

Figure 2.6: An example of density of fracture network; block and contour maps.

2.7.6 Example: Backbone Extraction

Backbone of two-dimensional fracture networks can be efficiently extracted by means of the function **Backbone2D**.

```
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Oct 2013

clear all
clc

fnm2 = GenFNM2D(150);                              %fracture network

clf
subplot(121);
DrawLines2D(fnm2,0,0,'-=[');                       %visualisation of fnm

bbn = Backbone2D(fnm2,true);                       %backbone extraction

if ~isempty(bbn)
    subplot(122);
    DrawLines2D(bbn,0,0,'-=[');                    %visualisation of the backbone
end
```

The resulting backbone from the above code is shown in Fig. 2.7.

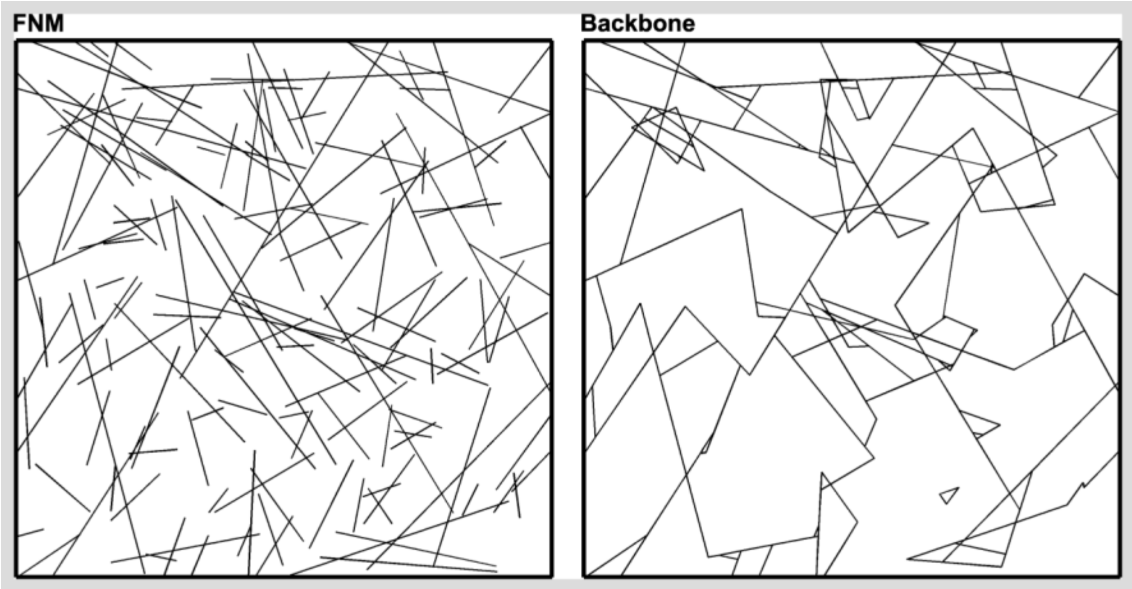

Figure 2.7: An example of backbone of fracture network.

The **BackboneToNodesEdges2D** function was used in the following program code to determine the popularity (centrality) of each node in the *graph* network.

```
% Part of package: Alghalandis Fracture Network Modelling (AFNM)
% Author: Younes Fadakar Alghalandis
% email: younes.fadakar@yahoo.com
% Copyright (c) 2011-2012-2013-2014 Younes Fadakar Alghalandis
% All rights reserved.
% Updated: Oct 2013

clear all
clc

lines = GenFNM2D(300);                          %2D fracture network
bbn = Backbone2D(lines,true,[],0);              %backbone
[nodes,edges] = BackboneToNodesEdges2D(bbn);    %nodes and edges: graph structure
nn = cellfun(@numel,nodes.values);              %popularity of nodes
pts = Stack(nodes.keys);                        %node locations

clf
subplot(121);
DrawLines2D(lines,0,[0,0,0],'-=[');

subplot(122);
DrawLines2D(bbn,0,[0,0,0],'-=[');
hold on
scatter(pts(:,1),pts(:,2),10,nn,'filled');      %colourised by popularity
```

In Fig. 2.8(*right*) blue, green and red dots correspond to 2, 3 and 4 connected edges respectively.

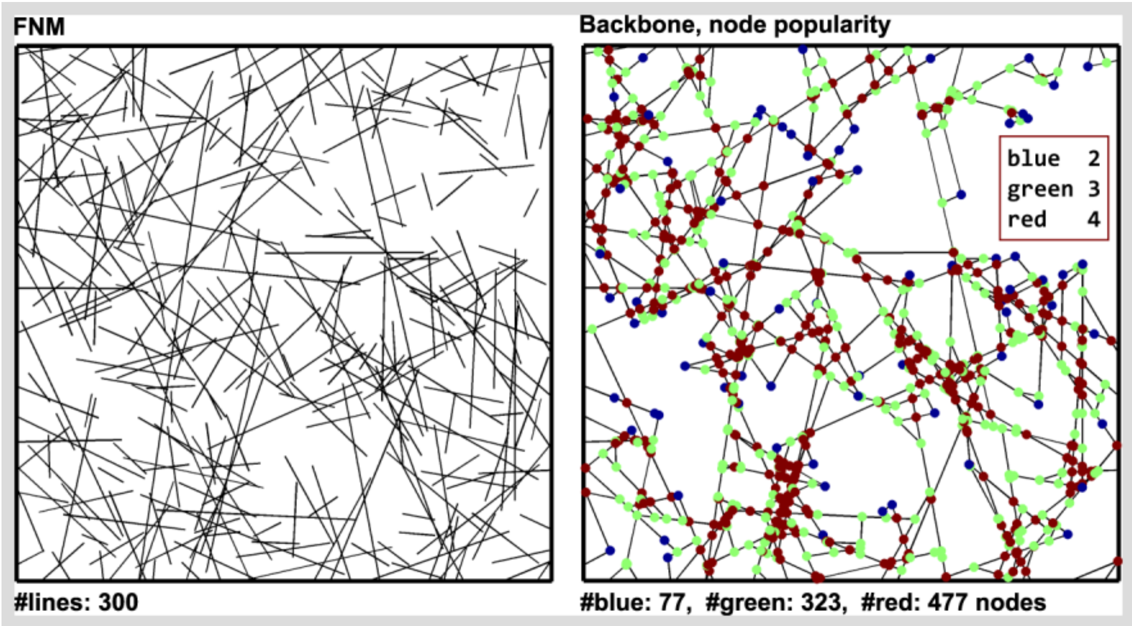

Figure 2.8: An example of backbone of fracture network, node centrality evaluation.