Random Neighborhood Graphs as Models of Fracture Networks on Rocks: Structural and Dynamical Analysis (Appendices)

Ernesto Estrada and Matthew Sheerin

Department of Mathematics & Statistics, University of Strathclyde, 26 Richmond Street, Glasgow G1 1XQ

Appendix

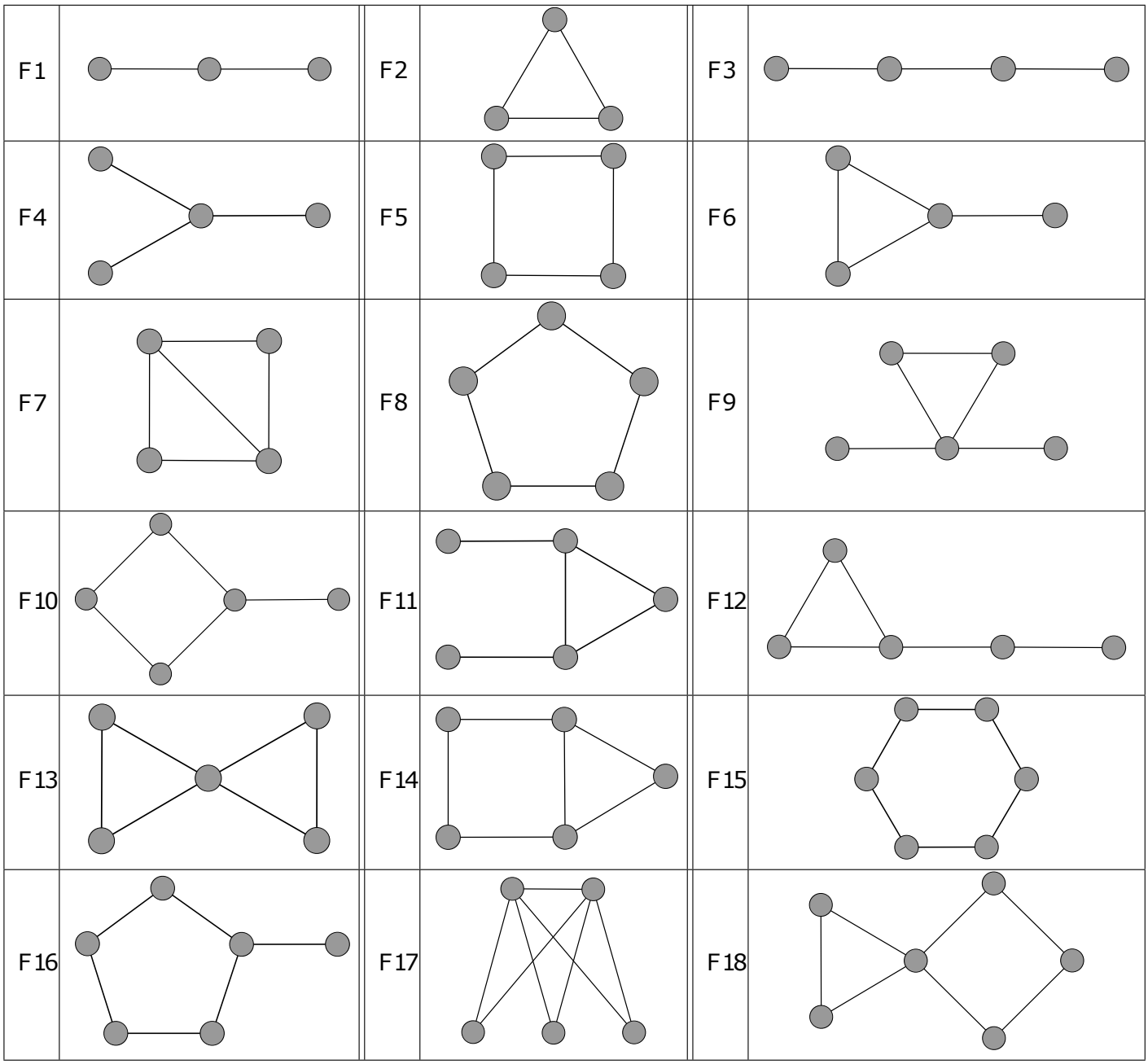

Fig. .1 Illustration of the structure of the small subgraphs used as structural descriptors in this work.

```
A=A-diag(diag(A));
n=length(A);
u=ones(n,1);
t=diag(A^3);
k=A*u;
k1=k-1;
k2=k-2;
m=sum(k)/2;

%Auxiliary functions

Q=A^2.*A;
P=0.5*A^2.*(A^2-1);
q=diag(A^5);
b=0.5*(q-5*t-2*(t.*k2)-2*Q*k2-2*(0.5*A*t-Q*u));
R=(1/6)*(A^2.*(A^2-1).*(A^2-2));

%Fragments according to book

F1=0.5*(k'*k1);
F2=(1/6)*sum(t);
F3=0.5*(k1'*A*k1)-3*F2;
F4=(1/6)*(k.*k1)'*k2;
F5=(1/8)*(trace(A^4)-4*F1-2*m);
F6=0.5*(t'*k2);
F7=0.25*u'*(Q.*(Q-A))*u;
F8=(1/10)*(trace(A^5)-10*F6-30*F2);
F9=0.25*(k2.*(k2-1))'*t;
F10=u'*(P-diag(diag(P)))*k2-2*F7;
F11=0.5*(k2'*Q*k2)-2*F7;
F12=0.5*u'*(A^2-diag(diag(A^2)))*t-6*F2-2*F6-4*F7;
F13=0.25*t'*(0.5*t-1)-2*F7;
F14=0.5*u'*(Q.*(A^3.*A))*u-9*F2-2*F6-4*F7;
F15=(1/12)*(trace(A^6)-2*m-12*F1-24*F2-6*F3-12*F4-48*F5-36*F7-12*F10-24*F13);
F16=k2'*b-2*F14;
F17=0.5*(u'*(R.*A)*u);
F18=0.5*(u'*(P-diag(diag(P)))*t)-6*F7-2*F14-6*F17;

%Results in the form of a column vector

S=[F1 F2 F3 F4 F5 F5 F7 F8 F9 F10 F11 F12 F13 F14 F15 F16 F17 F18]';
```

Algorithm 1: Matlab code used for the calculation of the 18 small subgraphs illustrated in Fig. (.1).

```
function [graph,p] = beta(n,beta,lengths,p)
%Output 'graph' as adjacency list and list of coordinates 'p'
%can use cell2mat on output graph to get adjacency matrix
%
%beta=0 is the complete graph %beta=1 is the Gabriel graph
%beta=2 is the RNG
%
%If no lengths given, a unit square is assumed.
%If one length given, unit area is assumed.
%
%functions being called use beta differently, so the value is converted via
%beta_other=(1-beta)/(1+beta): [0,inf]->[1,-1]
%
%You can also supply the coordinates 'p' yourself as an n*2 matrix

if ~exist('beta','var')
    error('Please supply a value for beta')
end

if ~exist('n','var')
    error('Please supply the number of nodes')
end
if beta<0
    error('Beta must be non-negative')
end

if ~exist('lengths','var')
    lengths=[1 1];
elseif length(lengths)==1
    lengths=[lengths 1/lengths];
end

if ~exist('p','var')
    p=[unifrnd(0,lengths(1),n,1) unifrnd(0,lengths(2),n,1)];
end

if beta==0
    graph=CompleteGraph(n);
elseif beta==1
    dataDist=distFast(p,p);
    [~,indall]=sort(dataDist,'ascend');
    graph=GabrielGraph(p,indall);
elseif beta==2
    dataDist=distFast(p,p);
    graph=RelativeNeighborhoodGraph(dataDist);
else
    beta=(1-beta)/(1+beta);
    dataDist=distFast(p,p);
    graph=LuneBetaSkeletonGraph(p,dataDist,beta);
end
end
```

Algorithm 2: Matlab code used for creating rectangular β-skeleton graphs, which makes use of an available toolbox (**?**]).