```
clear all; close all;

n = 10;
p = 0.5;

Nexp = 100;
results = zeros(Nexp,1);

for exp = 1:Nexp
    fprintf('Experiment %03d...\n',exp);
    A = getRandomSignedGraph(n,p);

    Niter = 1e6;
    for iter = 1:Niter
        balanced = isGraphBalanced(A);
        if (balanced)
            break;
        end

        % I) Pick a triad at random
        triad_nodes = randsample(n, 3, false); % Sample without replacement
        triad_edges = [triad_nodes circshift(triad_nodes,-1)];

        % II) Check if triad is balanced. Use the fact that, in a balanced
        %       triangle, the product of the signs of the three edges is
        %       positive.
        triad_balanced = 1;
        for k = 1:3
            triad_balanced = triad_balanced * ...
                             A(triad_edges(k,1),triad_edges(k,2));
        end
        triad_balanced = triad_balanced > 0;

        % III) If the triad is not balanced, then choose one of the edges
        %       at random and flip its sign
        if (~triad_balanced)
            k = randi(3,1);
            A(triad_edges(k,1),triad_edges(k,2)) = ...
                -A(triad_edges(k,1),triad_edges(k,2));

            A(triad_edges(k,2),triad_edges(k,1)) = ...
                -A(triad_edges(k,2),triad_edges(k,1));
        end
    end

    results(exp) = balanced;
end
```

```
% Returns a signed adjacency matrix of a complete graph on n nodes,
%   where each edge has probability p (1-p) of being + (-)
function A = getRandomSignedGraph(n,p)

A = 1/2*eye(n); % We'll later set A = A + A'
for i = 1:(n-1)
    for j = (i+1):n
        if (rand()<p)
            A(i,j) = 1;
        else
            A(i,j) = -1;
        end
    end
end

A = A + A';
```

```
function balanced = isGraphBalanced(A)

N = size(A,1);
Ap = A > 0; % "Positive adjacency matrix"

% First, we identify the candidate faction by looking
%   at the edges of node 1. We encode the candidate factions
%   into vectors g1 and g2.
%-------------------------------------------------------------
g1 = zeros(N,1); g1(1) = 1;
g2 = zeros(N,1);
for i = 2:N
    if (A(1,i) > 0)
        g1(i) = 1;
    else
        g2(i) = 1;
    end
end

% Second, we check if the candidate faction vectors are eigenvectors
%   of the positive adjacency matrix, with eigenvalue equal to the
%   size of the respective faction
%-------------------------------------------------------------
n1 = sum(g1);
n2 = sum(g2);

balanced = all(Ap*g1 == n1*g1) && ...
           all(Ap*g2 == n2*g2);
```