Contents


```
% Code Summary:
% Now tackling the problem of force chains
% This will be a very lengthy description. I initially tried to optimise
% the data set by not using adjacency matrices. Prabhu Sir said that you
% don't check for all particles. For the next time step, you check the
% particles it was already in contact with and few more particles. I can
% see it's implementation but still feel my case is much faster. There are
% multiple plots all throughout, either of force chains, or an
% implementation of Peters, Muthuswamy (2005) considering particles in the
% quasilinear particle assembly. There are also my attempts to solve it on
% my own before that. In the pink and cyan graph, I tried to colour the
% nodes which have degree>=2 and their corresponding edges. The 2nd plot is
% a kind of radial averaging. The red and black plot is a joint attempt at
% "all" particle contacts and their joining. It is impossible due to
% obvious reasons why you cannot derive any useful information from it.
```

Clearing previous data

```
clc;
close all;
clear variables;
```

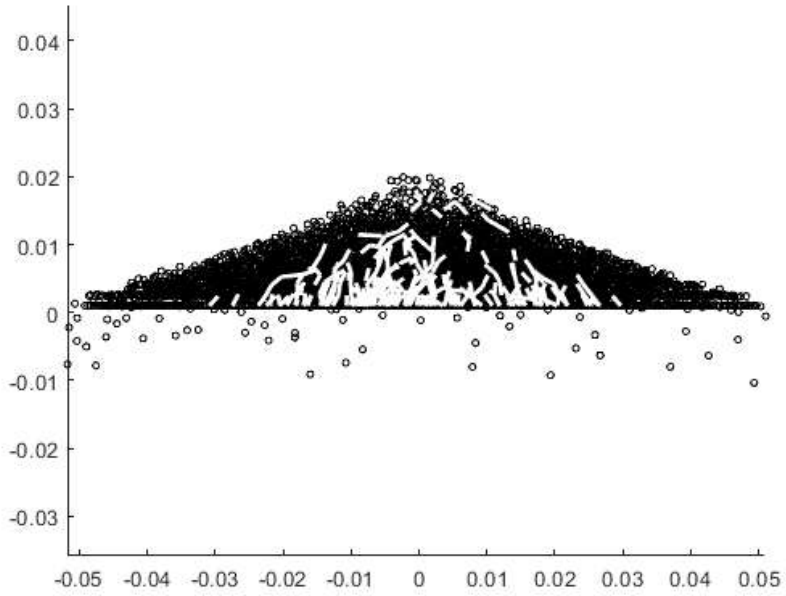
Data loading

```
file = importdata("post\particles_1000000.liggghts", " ", 9);
data = file.data;
clear file;

% id = data(:, 1);
% % [~, I] = sort(id, "ascend");
% data = data(I, :);
X = data(:, 3);
Y = data(:, 4);
Z = data(:, 5);
radius = data(1, end-1);
dp = 2*radius;

zbool = (Z < 0.02);
x = X(zbool); y = Y(zbool); z = Z(zbool);

dist_particles = [];

% For all particles
for i=1:length(x)

    x1 = x(i); y1 = y(i); z1 = z(i);

    xfluc = x1 - x; yfluc = y1 - y; zfluc = z1 - z;

    dist = sqrt(xfluc.^2 + yfluc.^2 + zfluc.^2);
    bool = dist<dp;
    contact_particles = find(bool==1);

    for j=1:length(contact_particles)
```

```
        if(i == contact_particles(j))
            continue
        else
            xcomp = x1 - x(contact_particles(j));
            ycomp = y1 - y(contact_particles(j));
            zcomp = z1 - z(contact_particles(j)); % I have computed the x, y, z components
            % Now, I'll resolve them along the x, y, z directions as per
            % the contact force component as it'll act in the centre to
            % centre direction

            xcompf = (dp - contact_particles(j))*xcomp/(sqrt(xcomp^2 + ycomp^2 + zcomp^2));
            ycompf = (dp - contact_particles(j))*ycomp/(sqrt(xcomp^2 + ycomp^2 + zcomp^2));
            zcompf = (dp - contact_particles(j))*xcomp/(sqrt(xcomp^2 + ycomp^2 + zcomp^2));
            dist_particles = [dist_particles; [i, contact_particles(j), abs(dist(contact_particles(j)) - dp), xcompf, ycompf, zcompf]];
        end
    end
end

[~, I] = sort(dist_particles(:, 3), "descend"); % We sort it using the amt of overlap

dist_particles =  dist_particles(I, :);
% The presence of non-zero numbers being lesser than dp implies the overlap
% yes was effective and it has considered it while running simulations
```

Plot

```
threshold = floor(size(dist_particles, 1)/100);

[~, I] = sort(z);

figure;
scatter(x(I), z(I), 10, "black");
axis equal
hold on
for i=1:threshold
    names = [dist_particles(i, 1), dist_particles(i, 2)];
    plot(x(names), z(names), "LineWidth", 2, "LineStyle", "-", "Color","white")
    dist_particles(i+1, :) = [];
    hold on
end

% saveas(gcf, "Full heap")
```

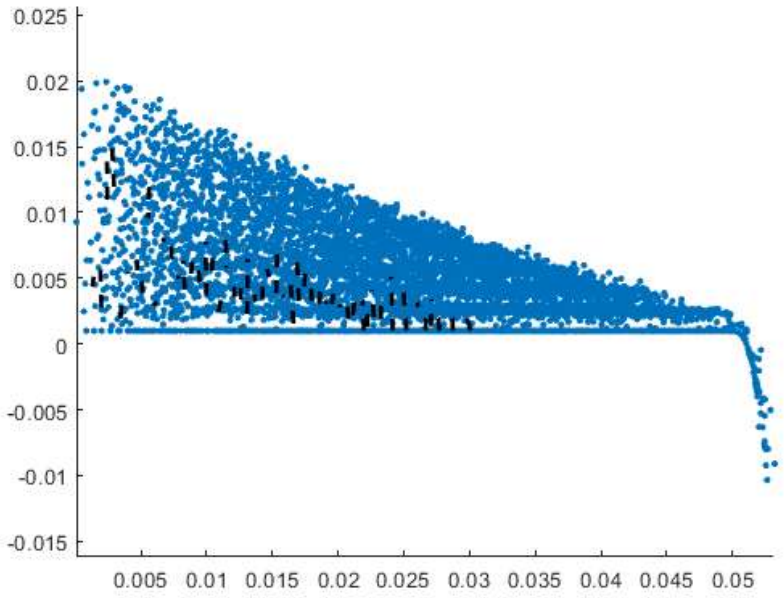


For the particles present in the 1:threshold region, I am going to check if any of their radii belong to specific regions, and then average them out.

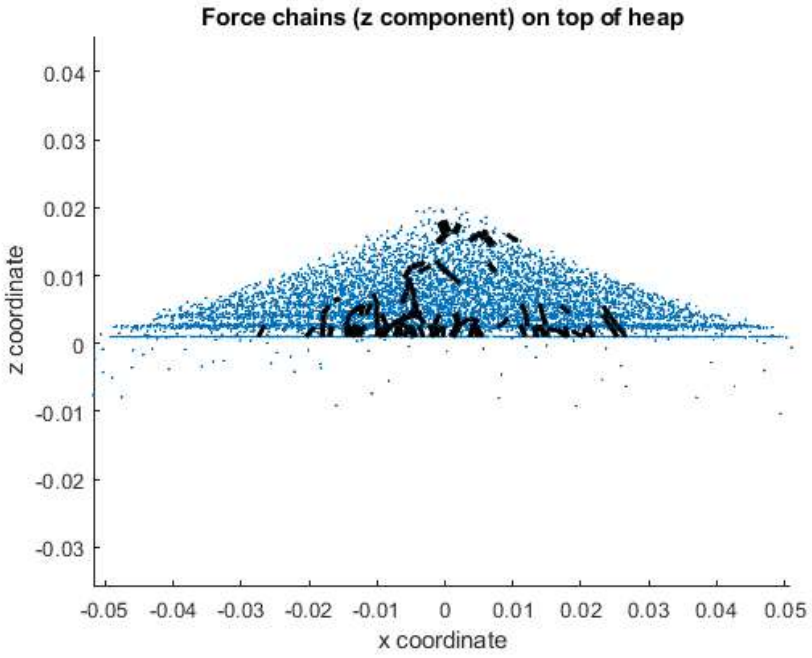
```
xth = x(dist_particles(1:threshold, 1));
yth = y(dist_particles(1:threshold, 1));
zth = z(dist_particles(1:threshold, 1));

rth = sqrt(xth.^2 + yth.^2);

r = sqrt(x.^2 + y.^2);
rbin = linspace(0, max(r), 100);

for i=1:length(rbin) - 1
    rows = find(rth > rbin(i) & rth < rbin(i+1));
    if (isempty(rows))
        newdata(i, :) = [0, 0];
    else
        newdata(i, :) = [mean(rth(rows)), mean(zth(rows))];
    end
end

xth = x(dist_particles(1:threshold, 2));
yth = y(dist_particles(1:threshold, 2));
zth = z(dist_particles(1:threshold, 2));

rth = sqrt(xth.^2 + yth.^2);

for i=1:length(rbin) - 1
    rows = find(rth > rbin(i) & rth < rbin(i+1));
    % rows = rth(rows==1);
    if (isempty(rows))
        newdata1(i, :) = [0, 0];
    else
        newdata1(i, :) = [mean(rth(rows)), mean(zth(rows))];
    end
end
```

One more plot

```
figure;
scatter(r, z, 10, "filled");
hold on

for i=1:length(rbin)-1
    if (newdata(i, 1) == 0 || newdata1(i, 1) == 0)
        continue
    end
    r1 = [newdata(i, 1), newdata1(i, 1)];
    z1 = [newdata(i, 2), newdata1(i, 2)];
    plot(r1, z1, "LineStyle", "--", "LineWidth", 2, "Color", "black");
    hold on
end
axis equal
set(gca, 'Color', [255, 255, 51]/256)
```



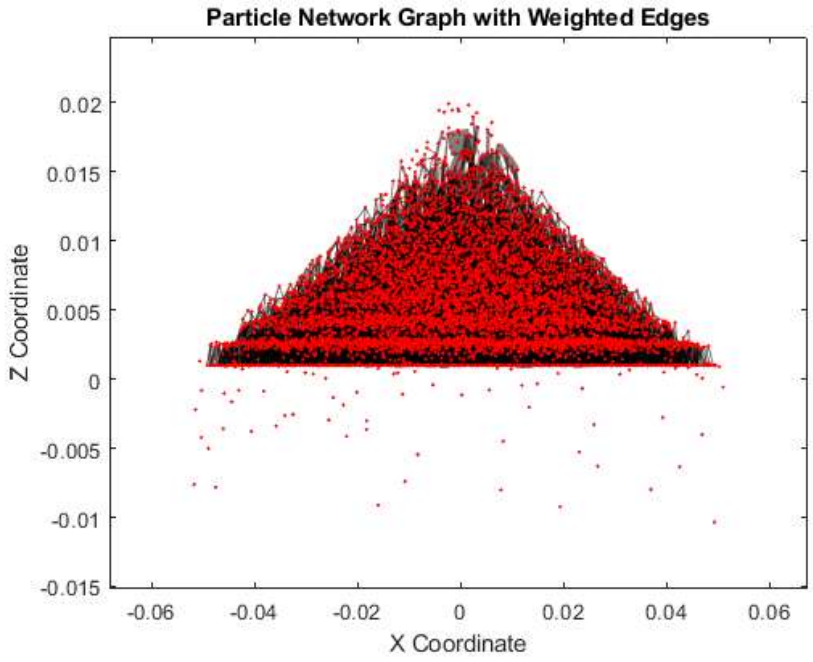

```
overlap = 10^5*dist_particles(:, 3);
figure;
scatter(x, z, 0.5)
hold on
axis equal
% for the given x, z coordinates the force is of the given nature. so plot
% accordingly like x, z should match
width = overlap(1:100);

for i=1:100

    names = [dist_particles(i, 1), dist_particles(i, 2)];
    scatter(x(names), z(names), 1, "filled")

    hold on
    plot(x(names), z(names), "Color","black", "LineWidth", 5*width(i))
    hold on

end
xlabel("x coordinate")
ylabel("z coordinate")
title("Force chains (z component) on top of heap")
```



Method using adjacency matrices

```
adj = zeros(size(x, 1), size(x, 1));

for i=1:size(adj, 1)
    for j=1:size(adj, 2)
        adj(i, j) = dp - sqrt((x(i) - x(j))^2 + (y(i) - y(j))^2 + (z(i) - z(j))^2);
        % adj(j, i) = adj(i, j);
        if (i==j)
            adj(i, j) = 0;
        end
        if (adj(i, j) <= 0)
            adj(i, j) = 0 ;
        end
    end
end
```

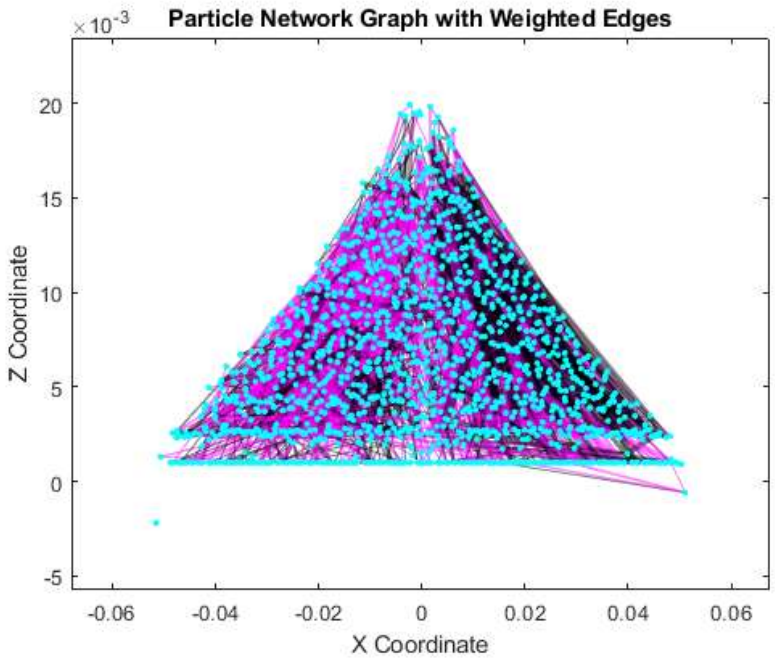
```
figure;
G = graph(adj);
h = plot(G, 'XData', x, 'YData', z);
LWidths = 10*G.Edges.Weight / max(G.Edges.Weight);
h.LineWidth = LWidths;
h.MarkerSize = 1;
h.NodeColor = 'r';
h.EdgeColor = [0 0 0];

xlabel('X Coordinate');
ylabel('Z Coordinate');
title('Particle Network Graph with Weighted Edges');
```



My method/way of plotting for a 2D plane

```
% Consider it for a 2D plane and then do the above things for the adjacency
% matrix in general.

ybins = linspace(min(y), max(y), 100);
ymin = ybins(45); ymax = ybins(55);

rows = find(y > ymin & y < ymax);
x1 = x(rows); y1 = y(rows); z1 = z(rows);
part_of_chain = zeros(size(x1, 1), 1);

adj1 = zeros(size(x1, 1), size(x1, 1));

for i=1:size(adj1, 1)
    for j=1:size(adj1, 2)
        adj1(i, j) = dp - sqrt((x(i) - x(j))^2 + (y(i) - y(j))^2 + (z(i) - z(j))^2);
        if (i==j)
            adj1(i, j) = 0;
        end
        if (adj1(i, j) <= 0)
            adj1(i, j) = 0 ;
        end
    end
end

figure;
G = graph(adj1);
h = plot(G, 'XData', x1, 'YData', z1);
LWidths = 5*G.Edges.Weight / max(G.Edges.Weight);
h.LineWidth = LWidths;

nodeDegrees = degree(G);
nodesDegree2 = find(nodeDegrees == 3);
edgeColors = repmat([1 0 1], numedges(G), 1);
nodeColors = repmat([1 0 1], numnodes(G), 1);

% Iterate over nodes with degree 2 and color their edges yellow
for i = 1:length(nodesDegree2)
    % Find edges connected to this node
    connectedEdges = findedge(G, nodesDegree2(i), 1:numnodes(G));

    % Color these edges yellow
    for j = 1:length(connectedEdges)
        if connectedEdges(j) ~= 0 % valid edge index
            edgeColors(connectedEdges(j), :) = [0 0 0]; % yellow
```

```
            nodeColors(connectedEdges(j), :) = [0 0 0];
        end
    end
end

h.MarkerSize = 2;
h.NodeColor = 'cyan';
h.EdgeColor = edgeColors;

xlabel('X Coordinate');
ylabel('Z Coordinate');
title('Particle Network Graph with Weighted Edges');
set(gca, 'Color', [0.9, 0.9, 0.99])
```

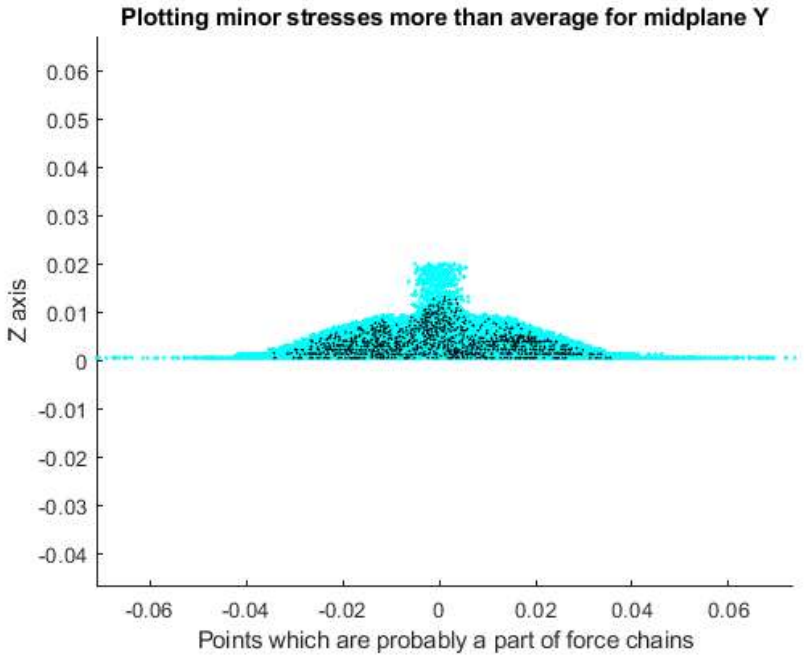


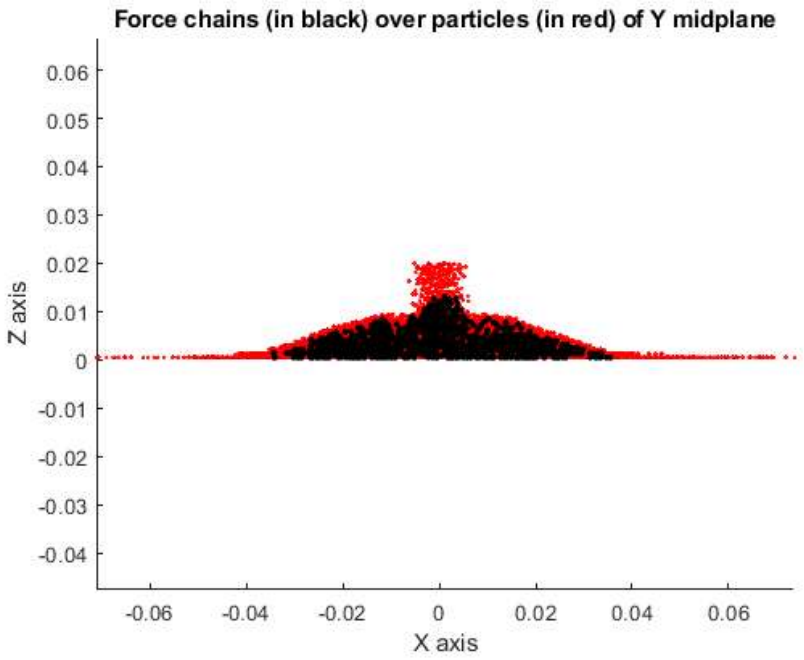
Trying out a minor implementation of the Muthuswamy, Peters paper from 2005

In the paper, they've considered a 2D assembly and only considered the existence of sigma 1, sigma 3 and sigma 13. But as all 6 components exist in this 3D setup, I used eig to find the invariant eigenvalues using all the components and then took the smallest once (mod of it) as the minor principal stress is the most compressive principal stress

```
% alpha = 0 implies perfectly linear force chains

file1 = importdata("particles_137400.liggghts", " ", 9);
data1 = file1.data;
clear file1;

X1 = data1(:, 3);
Y1 = data1(:, 4);
Z1 = data1(:, 5);
radius1 = data1(1, end-1);
dp1 = 2*radius1;
alpha = pi/18;

ybins = linspace(min(Y1), max(Y1), 100);
ybool = (Y1 > ybins(48) & Y1 < ybins(52));

zbool = Z1 < 0.020;
ybool = ybool & zbool;

xpl = X1(ybool); ypl = Y1(ybool); zpl = Z1(ybool);
% Now calculate the stresses and their eigenvalues

minor_stresses = zeros(length(xpl), 2);
k=1;
for i=1:length(X1)
```

```
    if (ybool(i)==1)
        tauxx = data1(i, 12); tauyy = data1(i, 13); tauzz = data1(i, 14);
        tauxy = data1(i, 15); tauxz = data1(i, 16); tauyz = data1(i, 17);
        stress_tensor = [tauxx, tauxy, tauxz;
                         tauxy, tauyy, tauyz;
                         tauxz, tauyz, tauzz];
        [~, D] = eig(stress_tensor);
        D = sort(diag(D), "ascend");
        minor_stresses(k, 1) = abs(D(3));
        minor_stresses(k, 2) = i; % This is so that we can get ready retrieval from X1 if needed
        k = k + 1;
    end
end

stressbool = minor_stresses(:, 1) > mean(minor_stresses(:, 1));
stress_take = minor_stresses(stressbool, :); % This contains the minor stresses which are larger than the average
% They also contain the X1 coordinates as well/ so that we can readily
% reference them as will be needed

xplot = X1(stress_take(:, 2));
zplot = Z1(stress_take(:, 2));
yplot = Y1(stress_take(:, 2));
id = stress_take(:, 2);

figure
scatter(xpl, zpl, 2, "cyan")
axis equal
hold on
scatter(xplot, zplot, 0.5, "black", "filled", "Marker", "o", "MarkerEdgeColor", "flat")
xlabel("Points which are probably a part of force chains")
ylabel("Z axis")
title("Plotting minor stresses more than average for midplane Y")

% now, xplot and zplot contain the particles' locations which are higher
% than the mean
% Now what I'll do is that I'll create an adjacency matrix and then in that
% put the value of the cos(alpha) < xxx < 1 and if that is correct then
% colour it with black I guess, toh then, joining the blacks and getting a
% pattern will be easy. also, as this is an implementation right from a
% paper, there are no issues as well!

adjacency = zeros(length(xplot));

for i=1:size(adjacency, 1)
    for j=1:size(adjacency, 2)
        % i
        l = sqrt(((xplot(i) - xplot(j))^2 + (yplot(i) - yplot(j))^2 + (zplot(i) - zplot(j))^2));
        vector = [xplot(i) - xplot(j), yplot(i) - yplot(j), zplot(i) - zplot(j)];
        if l==0
            continue
        end
        adjacency(i, j) = abs(vector(3)/l);
        if l >= dp || adjacency(i, j) < cos(alpha)
            adjacency(i, j) = 0;
        end
    end
end

network = graph(adjacency);
figure;
scatter(xpl, zpl, 2, "red")
hold on
plot(network, "NodeColor", [0 0 0], "EdgeColor", [0 0 0], 'XData', xplot, 'YData', zplot)
axis equal
xlabel("X axis")
ylabel("Z axis")
title("Force chains (in black) over particles (in red) of Y midplane")
```

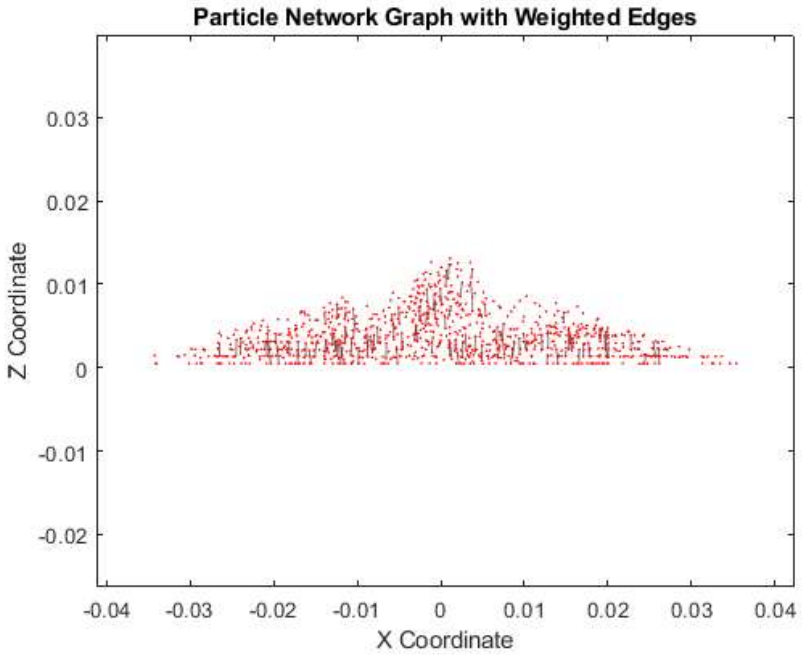




min(degree) = 2 and continue? Seems to run into some issue though

```
figure;
axis equal
h = plot(network, "NodeColor", [0 0 0], "EdgeColor", [0 0 0], 'XData', xplot, 'YData', zplot);

nodeDegrees = degree(network);
nodesDegree2 = find(nodeDegrees >= 1);
edgeColors = repmat([1 0 1], numedges(network), 1);
nodeColors = repmat([1 0 1], numnodes(network), 1);

% Iterate over nodes with degree 2 and color their edges yellow
for i = 1:length(nodesDegree2)
    % Find edges connected to this node
    connectedEdges = findedge(network, nodesDegree2(i), 1:numnodes(network));

    % Color these edges yellow
    for j = 1:length(connectedEdges)
        if connectedEdges(j) ~= 0 % valid edge index
```

```
            edgeColors(connectedEdges(j), :) = [0 0 0]; % yellow
            nodeColors(connectedEdges(j), :) = [0 0 0]  ;
        end
    end
end

h.MarkerSize = 0.5;
h.NodeColor = 'red';
h.EdgeColor = edgeColors;

xlabel('X Coordinate');
ylabel('Z Coordinate');
title('Particle Network Graph with Weighted Edges');
set(gca, 'Color', [0.9, 0.9, 0.99])

axis equal
```



Published with MATLAB® R2023a