Q1.A)

```
function D = getDegreeMatrix(W)
   % Calculates the Degree Matrix for a given adjacency matrix
          D = zeros(size(W, 1), size(W, 2));
      for i=1:length(W)
         D(i,i) = sum(W(i,:))
      end
   % Fill in D here
End
```

Q1.B)

```
function L = getNormalizedLaplacian(W, D)
   % Calculates the Normalized Laplacian for an adjacency matrix W
   % and degree matrix D
   W = full(W);
   D(D==0) = eps; % avoid dividing by zero
   L = zeros(size(W, 1), size(W, 2));
   % Calculate L here
   L = D - W;
   L = inv(D) * L
end
```

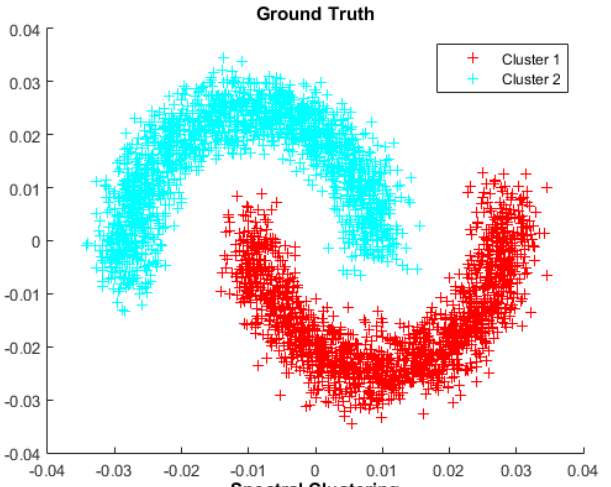
Q1c

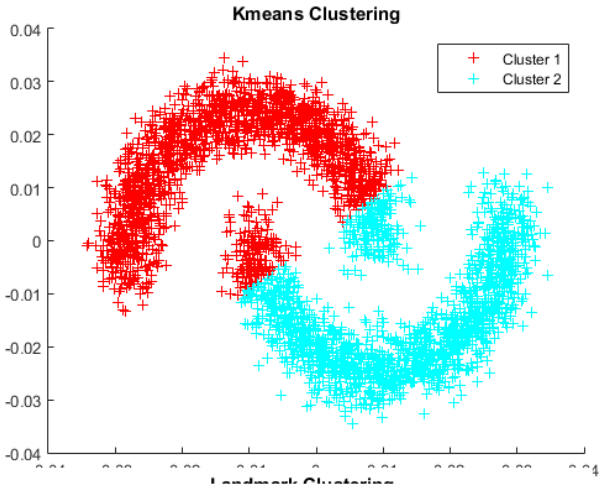
Use the runExperimentsQ1.m _le to plot all the data points, colored by
their cluster label from the Spectral Clustering method. Include your plots in your handin. Qualitatively, what is the difference between the decision surfaces of the two methods? Where does this difference come from?

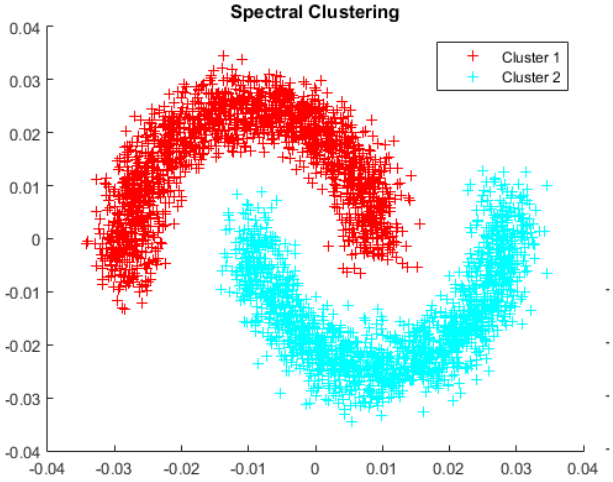
The decision surface for spectral clustering is a non linear whereas for kmeans its linear.

This is basically because Kmeans uses a centroid based approach such that the clusters are formed by the perpendicular bisector of the line joining the centroids.

Spectral Clustering clusters the data set into non linear clusters. Spectral Clustering performs dimensionality reduction(Feature transformation) before clustering the data into fewer dimensions.







Q1.D. Number of Queries (10 points) How many pairwise distance queries does the spectral clustering Method.

Number of Queries

Running Spectral Clustering
Pairwise distance queries for the spectral clustering = ((3380) ^2) * 2 = 22848800

Running K-means Clustering
Pairwise distance queries does k-means = ((3380)^2) * 6 = 68546400

Q2.a)

function [L, n_queries] = LandmarkSelection(q, iter, S)
% Complete the LandmarkSelection algorithm here.

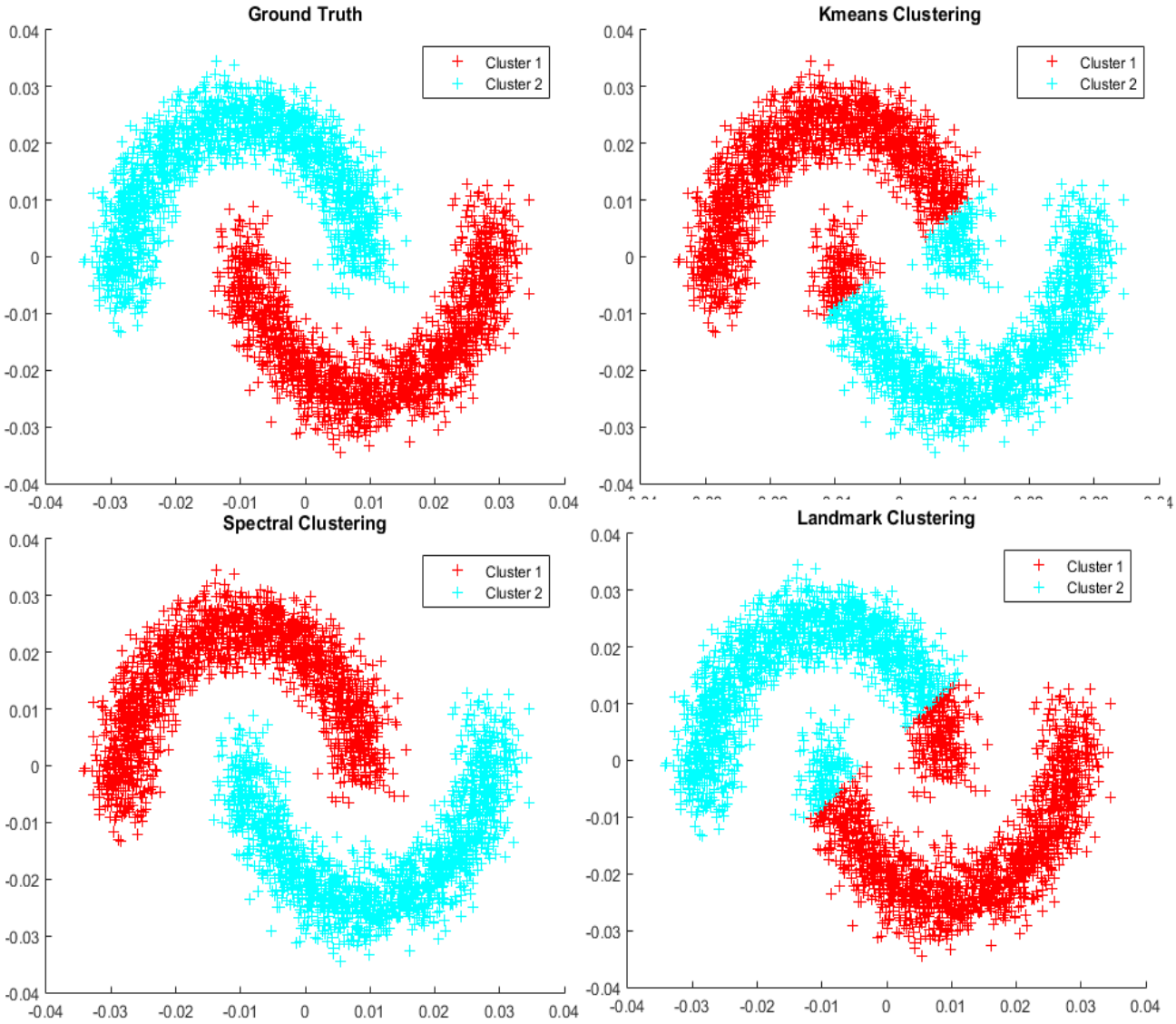
```
    rng(0)
    n_queries = 0;
    iter = floor(iter);
    q = floor(q);
    L = zeros(iter, 1);

    % Choose l in S uniformly at random;
    n = size(S, 1);
    l = randi(n);
            L(1) = l;
    [d_min, new_queries] = query(l, S);
    n_queries = n_queries + new_queries;

    for i = 1:(iter-1)
                        % FILL IN HERE
   [s,indx] = sort(d_min);
      pts = indx(n-q+1:n);
      l = pts(randi(q))
      L(i+1) = l;
      [d_min_new, new_queries] = query(l, S);
      n_queries = n_queries + new_queries;
      for j = 1:length(d_min)
          if(d_min_new(j)<d_min(j))
              d_min(j)= d_min_new(j);
          end
      end
            end
end
```

Q2.b)

```
function [idxs, n_queries, flag] = ExpandLandmarks(k, s_min, n_prime, L, S)
% k - number of clusters to find
% s_min - the minimum number of points in each ball
% n_prime - the minimum number of points to be clustered
% L - landmarks
% S - nxp data matrix
% Returns:
% idxs - nx1 list of cluster ids for each point
% n_queries - scalar number of pairwise distances queried
% flag - boolean value, non-zero indicates failure to find a clustering.
    idxs = []; % Cluster ids for each point
    n_queries = 0;  % Number of pairwise distances calculated
    flag = 1;   % 0 = Success, 1 = Failure.
    n_samples = size(S, 1);
    n_landmarks = size(L,1);
    lm = zeros(n_samples, 1);  % Holds the index of the representative landmark for each data point
    items = zeros(n_landmarks, n_samples);  % data points in each ball about each landmark
    Clustered = zeros(n_samples, 1); % set of assigned points
    U = zeros(n_samples, n_samples); % set of sets of connected components in Gb
    n_components = 0;   % Tracks the number of connected components
```

```
  H = MinHeap(n_samples*n_landmarks);
  % Fill the heap with all the pairwise distances between landmarks and
  % samples
  for i = 1:n_samples
    for j = 1:n_landmarks
      l = L(j);
      dist = norm(S(l,:) - S(i,:));
      key = struct('val', dist, 'landmark_index', l, 's_index', i);
      n_queries = n_queries + 1;
      H.InsertKey(key);
    end
  end
  while H.Count() > 0
    x = H.ExtractMin();
    l_star = x.landmark_index;
    s_star = x.s_index;
    items(l_star, s_star) = 1;
    points_in_cur_ball = items(l_star, :);
    n_points_in_cur_ball = sum(points_in_cur_ball);
    %%% FILL IN HERE
          if(n_points_in_cur_ball>s_min)
        %[ U, n_components, lm ] = UpdateComponents( U, n_components, lm, l, s )
        [U,n_components,lm] = UpdateComponents(U,n_components,lm,l_star,s_star);
        Clustered(s_star)=1;
      end
      if (n_points_in_cur_ball==s_min)
        for i = 1:size(U,1)
          if sum(U(i,:)) == 0
            U(i,l_star) = 1;
            break;
          end
        end
         Clustered(l_star) = 1;
      lm(l_star) = l_star;
      n_components = n_components+1;
        for i =1:length(items(l_star,:))
          if(items(l_star,i)==1)
            [U,n_components,lm] = UpdateComponents(U,n_components,lm,l_star,i);
            Clustered(i)=1;
          end
        end
      end
    %%%
    num_clustered = sum(Clustered);
    if num_clustered >= n_prime && n_components == k
      [idxs, flag] = FormatClustering(U, items);
      break;
    end
  end
end
```

Q2.c.)



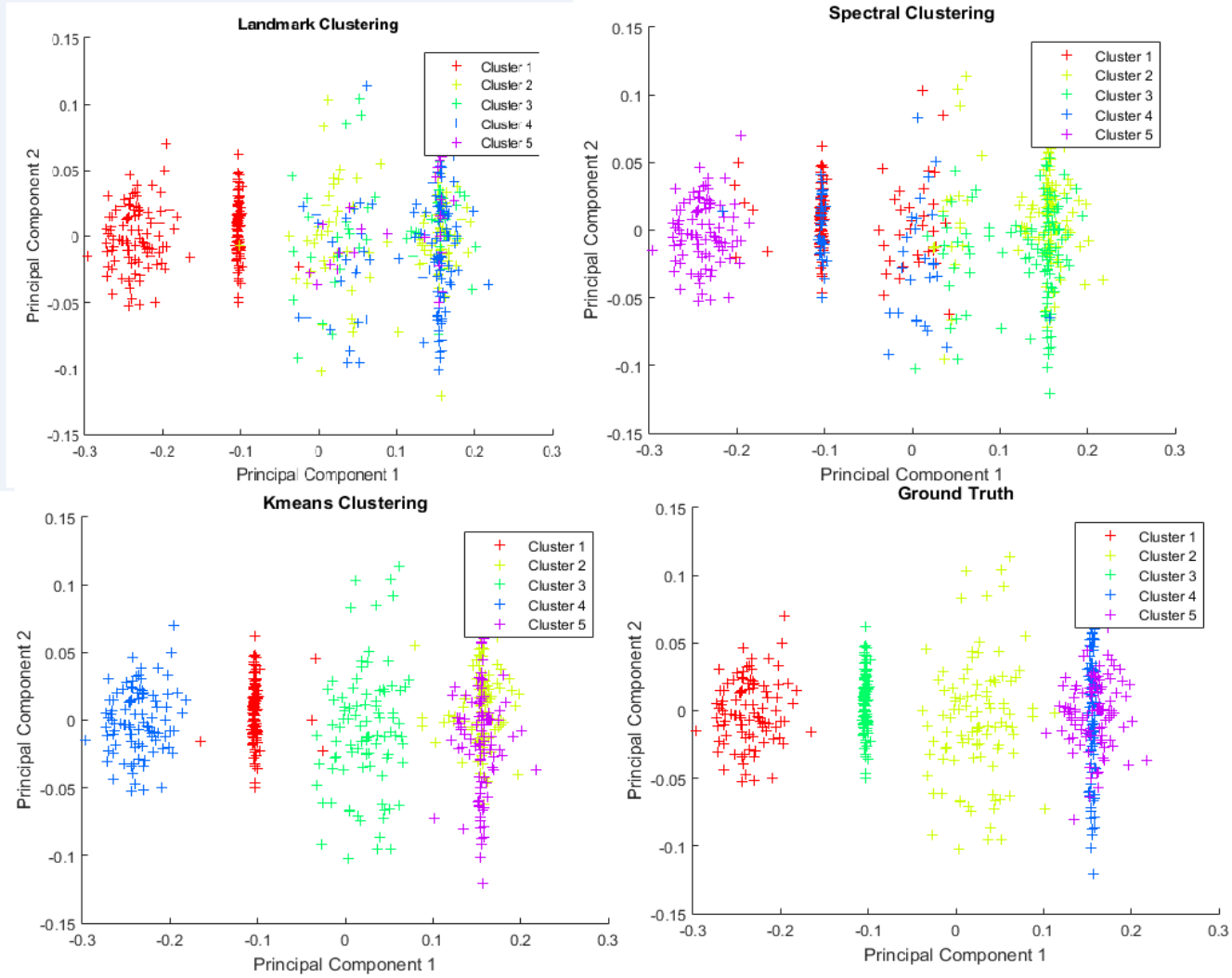
The decision boundary for Landmark based clustering closely resembles with Kmeans clustering. This is because Kmeans uses a median based approach which is similar to Landmark Clustering's median/mean based approach for clustering.

Q2.d)

Dataset size 3380
Landmark Clustering queries 378560

Q.3A)

K-means perform better because it closely resembles the ground truth.



Q3B.) (reference - http://www.cs.cmu.edu/~ninamf/papers/LandmarkClusteringMinSum.pdf)

(c; epsilon)-property is used to find the correctness of the algorithm. Landmark clustering algorithm uses the k-mean/median based object function. This is algorithms is efficient because it doesn't not require the full distance matrix which is infeasible in case of large datasets. Hence, the Landmark based clustering methodology works good in case of biological datasets (reference – Pg3 Last para http://www.cs.cmu.edu/~ninamf/papers/active-clustering-JMLR.pdf) .

Q3.C.)

In Landmark based clustering minimum number of points required to form a cluster for a Landmark. If this Smin is a very big value than more number of points will be required to form a cluster which may result in merging two or more clusters which are actually there in ground truth.

If the Smin values is small then as soon as the Smin value for a Landmark is reached it will try to expand to nearby outliers. This may result in clusters getting merged by expanding to nearby outliers. This results in two or more clusters hetting merged into one.