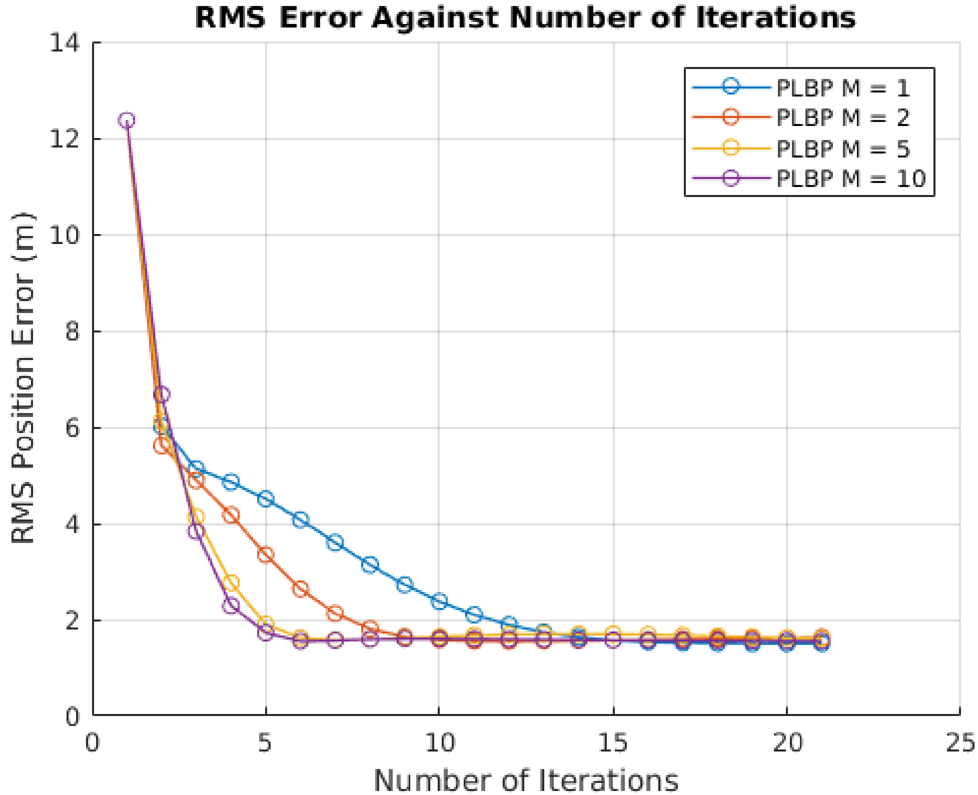
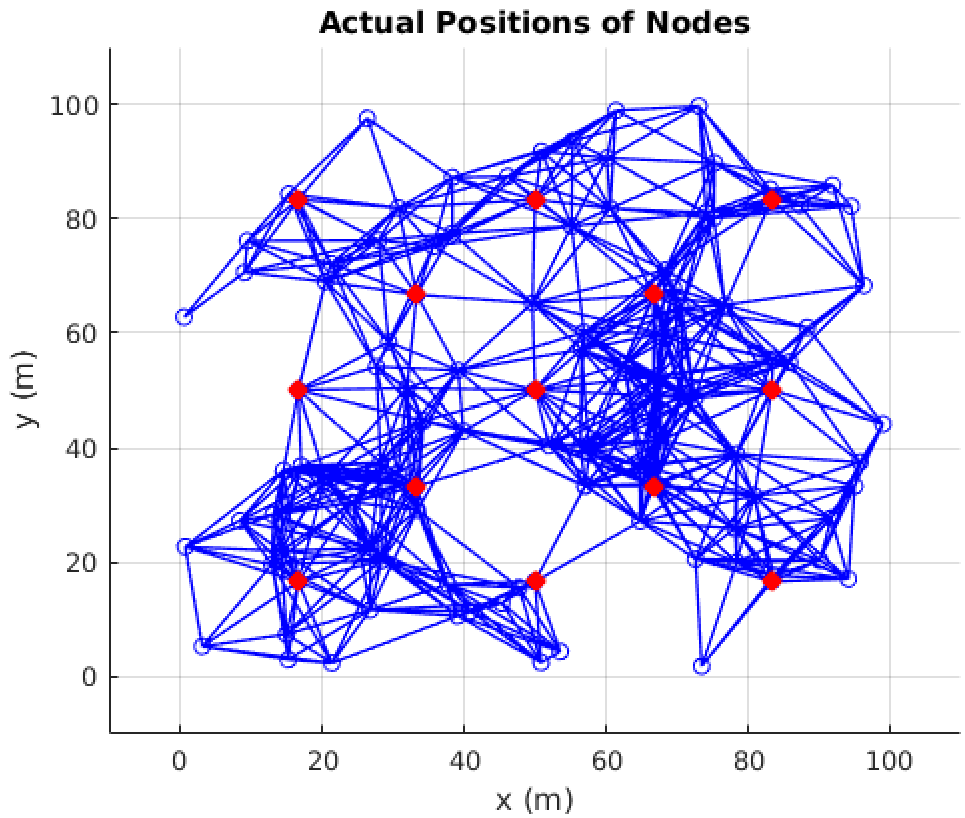
Main PLBP Algorithm

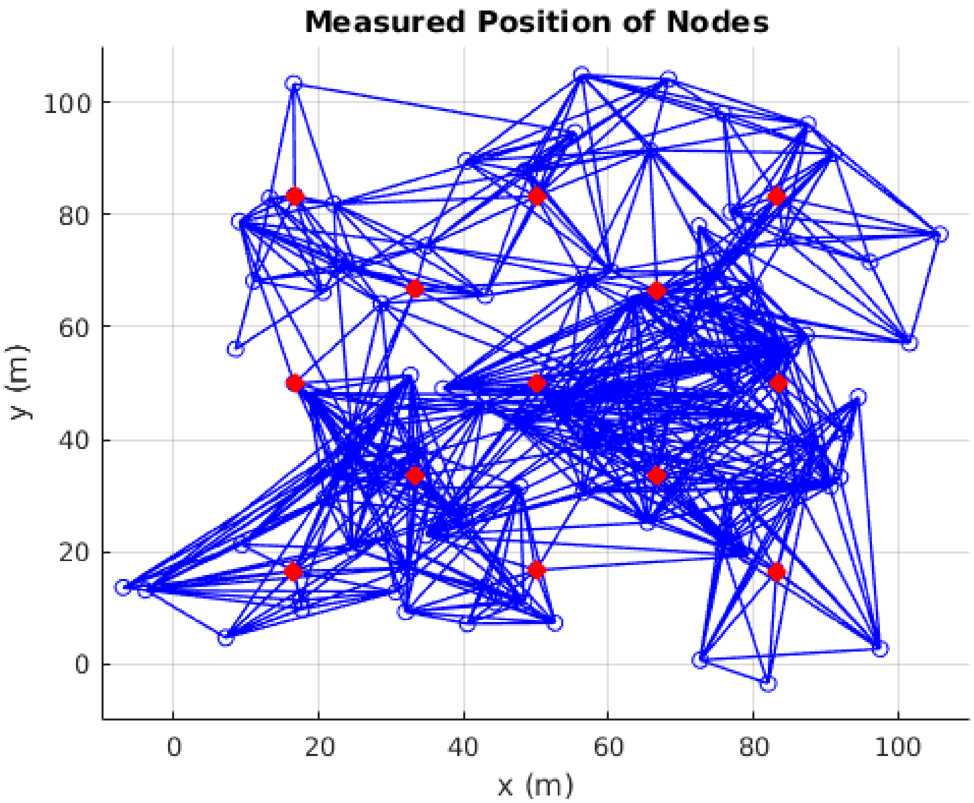
Node (1 to 100 - Normal Nodes) and (101 to 113 - Anchor Nodes) ------

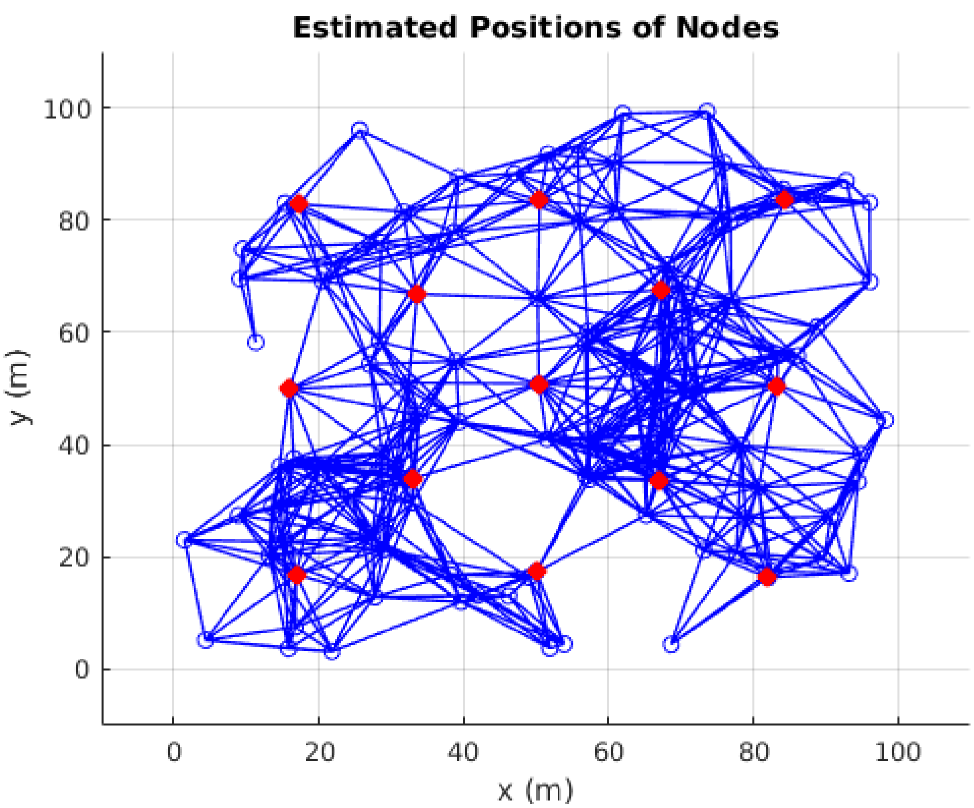
```
clear;

% Load the generated data --------
load data.mat

% Setting up variance for different measured data
 ---------------------
for i=1:100
  P(:,:,i) = 100.*eye(2);
end

for i=101:113
  P(:,:,i) = 0.01.*eye(2);
end

R = 1;
J = 20;
%---------------------------------------------------------------------

% Run four iterations for 1 PLBP, 2 PLBP, 5 PLBP and 10 PLBP, where M
 PLBP means M BP iterations.
for M=[1 2 5 10]
  u = x_observed;
  W = P;

  A(:,:,113,113) = zeros(1,4);
  b = zeros(113,113);
  sigma = zeros(113,113);
  Error = x_actual - u;
  RMSE = sqrt(sum(sum(Error.*Error))/113);

  % No of iteration for PLBP i.e. J times
 -------------------------------
  for k=1:J
    waitbar(k/20)

    % Run SLR Algorithm for i,j edges
 --------------------------------------
    for i=1:113
      for j=1:113
        if E(i,j)&&(i~=j)
          ul = transpose([u(i,:), u(j,:)]);
          Wl = [W(:,:,i),zeros(2,2);zeros(2,2),W(:,:,j)];
          [A(:,:,i,j), b(i,j), sigma(i,j)] = doSLR(ul, Wl);
        end
      end
```

```
    end
    %
-----------------------------------------------------------------------

    % Run BP for M times for every nodes from 1 to 113
----------------------
    for m=1:M
      for r=1:113
        [u(r,:), W(:,:,r)] = doBP(A, b, sigma, u, W, r, E, h_observed,
R);
      end
    end
    %
----------------------------------------------------------------------

    Error = x_actual - u;
    RMSE(:,k+1) = sqrt(sum(sum(Error.*Error))/113);
  end

  hold on;
  plot(1:21,RMSE(:,1:21),'o-', 'LineWidth', 1);
end


legend('PLBP M = 1', 'PLBP M = 2', 'PLBP M = 5', 'PLBP M = 10')

title('RMS Error Against Number of Iterations');
xlabel('Number of Iterations')
ylabel('RMS Position Error (m)')
grid on;

figure(2)
plotGraph(x_actual,E)
title('Actual Positions of Nodes');
xlim([-10 110])
ylim([-10 110])
xlabel('x (m)')
ylabel('y (m)')

figure(3)
plotGraph(x_observed,E)
title('Measured Position of Nodes');
xlim([-10 110])
ylim([-10 110])
xlabel('x (m)')
ylabel('y (m)')

figure(4)
plotGraph(u,E)
title('Estimated Positions of Nodes');
xlim([-10 110])
ylim([-10 110])
xlabel('x (m)')
ylabel('y (m)')
```


RMS Error Against Number of Iterations
PLBP M = 1
PLBP M = 2
PLBP M = 5
PLBP M = 10
RMS Position Error (m)
Number of Iterations


Actual Positions of Nodes
y (m)
x (m)


Measured Position of Nodes
y (m)
x (m)


Estimated Positions of Nodes
y (m)
x (m)

Published with MATLAB® R2017a

This function is to generate the required positions of nodes to setup the model.

Output: data.mat = matlab workspace file which will contain the node positions x_actual - Actual position of nodes x_observed - Measured position of nodes i.e. a gaussian noise is added to actual positions h_actual, h_observed - Message (Distance) between two nodes, actual and with noise(measure) respectively E - Matrix giving information about which node can communicate with each other ie. distance between them is within 20m E(i,j) = 0 for no communication between i and j nodes, E(i,j) = 1 for communication between i and j nodes.

```
%---------------------------------------------------------------------------------

clear;

Xa =
 [16.666,50,83.333,33.333,66.666,16.666,50,83.333,33.33,66.66,16.666,50,83.33];
Ya =
 [16.666,16.666,16.666,33.333,33.333,50,50,50,66.66,66.66,83.333,83.33,83.33];

x_actual = zeros(113, 2);
x_actual(1:100,:) = 100.*rand(100,2);
x_actual(101:113,:) = [transpose(Xa), transpose(Ya)];

x_observed = zeros(113, 2);
x_observed(1:100,:) = x_actual(1:100,:) + 10.0.*randn(100, 2);
x_observed(101:113,:) = x_actual(101:113,:) + 0.1.*randn(13, 2);

h_actual = zeros(113,113);
E = zeros(113,113);
for i = 1:113
  for j = 1:113
    h_actual(i,j) = norm(x_actual(i,:) - x_actual(j,:));
    if h_actual(i,j) <= 20.0
      E(i,j) = 1;
    end
  end
end

h_observed = h_actual + 1.*randn(113, 113);

save data.mat
```

Published with MATLAB® R2017a

SLR based on Sigma Points

Inputs: ul, Wl - mean and Covariance matrix for two nodes which is to be linearised Outputs: A, b, sigma - Linearisation Parameters. A = [A1 A2]

```
function [A, b, sigma] = doSLR(ul, Wl)
  N = 4;
  X = zeros(4,9);

  % Sigma Points and Corresponding Weight Generation
 --------------------------
  X(:,1) = ul;
  w1 = 1/3;
  wo = (1-w1)./(2.*N);
  T = chol(Wl);
  f = (N/(1-w1))^(1/2);

  % Approximating Linearisation based on the sigma points selected
 above -------
  for i=2:5
    X(:,i) = ul + f.*(T(i-1,:)');
    X(:,i+N) = ul - f.*(T(i-1,:)');
  end


  Z = sqrt((X(1,:) - X(3,:)).^2 + (X(2,:) - X(4,:)).^2);
  z = w1.*Z(:,1) + wo.*sum(Z(:,2:9));

  shi = w1.*(X(:,1) - ul).*(Z(:,1) - z);
  for j=2:9
    shi = shi + wo.*(X(:,j) - ul).*(Z(:,j) - z);
  end

  phi = w1.*(Z(:,1) - z).*(Z(:,1) - z);
  for j=2:9
    phi = phi + w1.*(Z(:,j) - z).*(Z(:,j) - z);
  end

  A = (shi')*(Wl^(-1));
  b = z - A*ul;
  sigma = phi - A*Wl*(A');

end
```

```
Not enough input arguments.

Error in doSLR (line 10)
  X(:,1) = ul;
```

Published with MATLAB® R2017a

Function to run belief propagation on a node 'k'

Inputs: A, b, sigma - Linearisation parameters obtained from SLR u, W - Old mean and variance of node 'k' E - matrix containing info about the edges which can communicate z - message(distance) matrix between two nodes R - Variation of measured message 'z' Outputs: ui, Wi - updated mean and variance for node 'k'

```
%-----------------------------------------------------------------------------

function [ui, Wi] = doBP(A, b, sigma, u, W, k, E, z, R)

  % Kalman update for all neighbouring nodes.
  for p=1:113
    if (E(p,k)&&(p~=k))

      alpha = z(p,k) - A(:,1:2,p,k)*(transpose(u(p,:))) - b(p,k);
      H = A(:,3:4,p,k);
      T = R + sigma(p,k) +
 A(:,1:2,p,k)*W(:,:,k)*transpose(A(:,1:2,p,k));

      ze = H*(u(k,:)');
      S = H*W(:,:,k)*(H') + T;
      shi = W(:,:,k)*(H');
      a = u(k,:)' + shi*(S^(-1))*(alpha - ze);
      Ae = W(:,:,k) - shi*(S^(-1))*(shi');

      u(k,:) = a';
      W(:,:,k) = Ae;
    end
  end
  ui = u(k,:);
  Wi = W(:,:,k);
end
```

```
Not enough input arguments.

Error in doBP (line 14)
    if (E(p,k)&&(p~=k))
```

Published with MATLAB® R2017a

To plot the graphs with all nodes and vertices --------

Inputs: V - Nodes/ Vertex E - edges

```
function plotGraph(V, E)
  for i = 1:113
    for j = 1:113
      if E(i,j)
        hold on;
        plot([V(i,1) V(j,1)], [V(i,2) V(j,2)], 'o-b', 'LineWidth', 1);
      end
    end
  end
  plot(V(101:113,1), V(101:113,2), 'xr', 'LineWidth', 5);
  grid on
end
```

```
Not enough input arguments.

Error in plotGraph (line 8)
      if E(i,j)
```

Published with MATLAB® R2017a