Grant Davis

CPE 470

06/08/2022

Project 3. Consensus Filters for a Multi-robot System Report

This report shows the results of the convergence of the consensus filter with a weighted average. The weights are implemented as weight design 1 and weight design 2. The convergence of the consensus filter with an average consensus with both max-degree and metropolis weights. Each weight design is a factor in the weighted average consensus algorithm, which outputs future node measurement predictions of the cell. The sensors and cells are both times fixed in this report, meaning both the nodes and the environment are unmoving and the neighbors are fixed. A 3D map of the consensus with a weighted average is built using a network of 30 nodes.

Weighted Average Consensus

The weighted average consensus is a nodal network, that uses the weights of its nodes with an associated cell measurement to produce a consensus of the nodes, of the measurement. Shown in Fig. 1 is the nodal network that consists of 10 nodes on a 1x1 grid. To accommodate the close proximity, the sensing range for each node has been scaled to tenth place. Changing the sensing range will the number of neighbors each node has. Its' sensing range is used to compute the design factors for the weight designs 1 and 2. It's also used to compute the noise variance of the model.

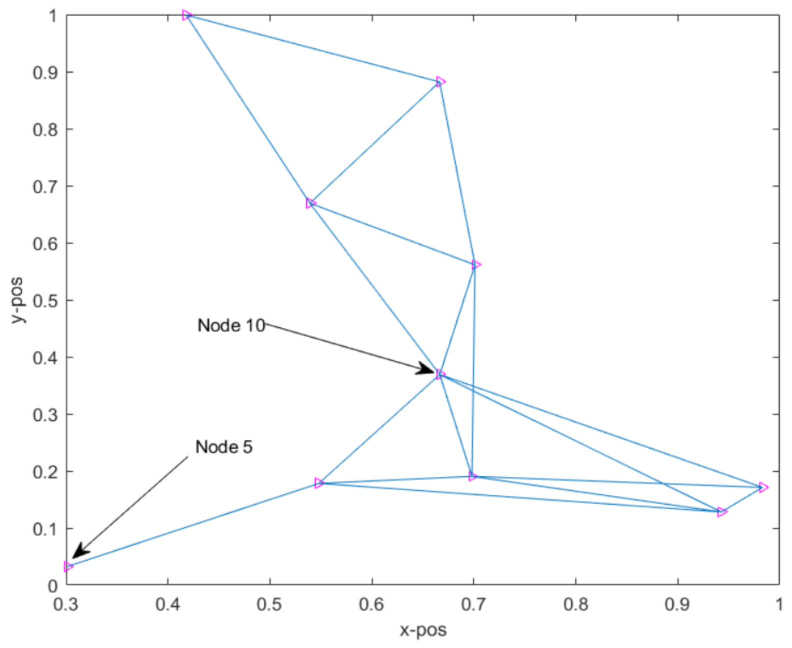

Fig. 1: Ten node network for the weight designs.

Weight Design 1

During every iteration, the first weight design converges linearly. Over time, the convergence becomes logarithmic over many iterations. This design is obtained by solving and setting all the vertex and edge weights to one. The weights of each node's neighbors are calculated using the distance of each node from the average node and the sensing range. As shown in Fig. 2, the nodes with the most neighbors, took the most time to converge. The node that took the least time to converge also had the least amount of neighbors. The node with the most neighbors took the longest to converge. As shown in Fig. 3, the initial estimate of node 10 was the furthest from the final estimate.

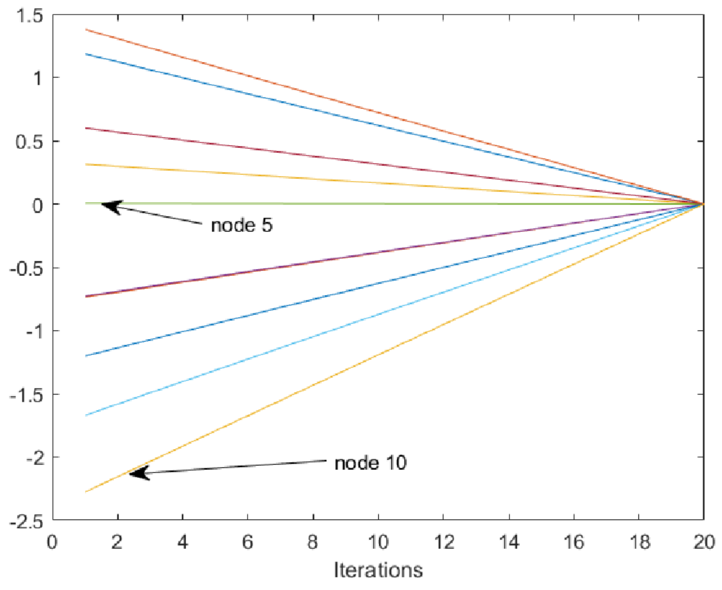

Fig. 2: Five and ten have the smallest and largest neighbors

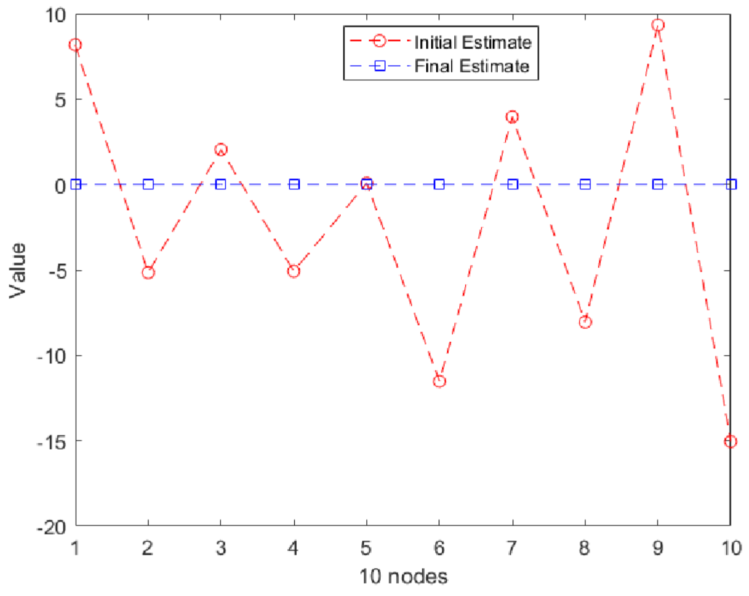

Fig. 3: Biggest error range of all the designs.

Weight Design 2

Similar to the first design, weight design 2 uses the weights of the nodes and the sensing range to generate weights. First, the weights of each node are calculated using a design factor, the sensing range, and the noise variance. Next, the weights are used along with the number of neighbors each node has to calculate the remaining weights. In Fig. 4. the consensus algorithm begins to fail. The convergences change directions until the last iteration where the final and initial estimates are equal.

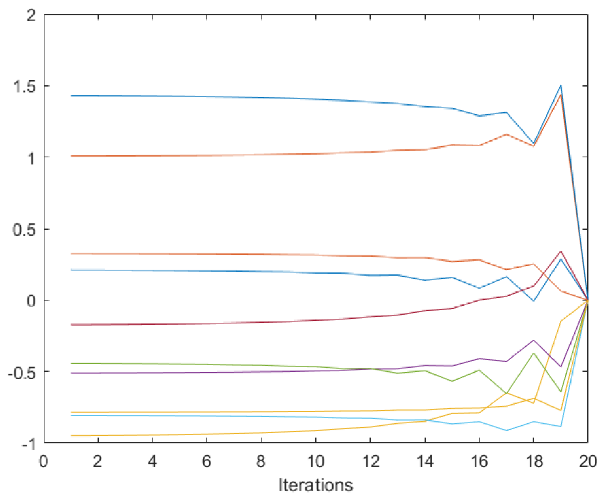

Fig. 4: Weighted Consensus Algorithm cannot adequately find the measurement.

The performance of the consensus algorithm could be partially analyzed by comparing the initial measurements of each design. For design 1, the initial estimate errors were off by around five each time. The initial estimates for design 2 show improvement by ranging within two as shown in Fig. 5. This could indicate that Weight Design 2 performs better at first.

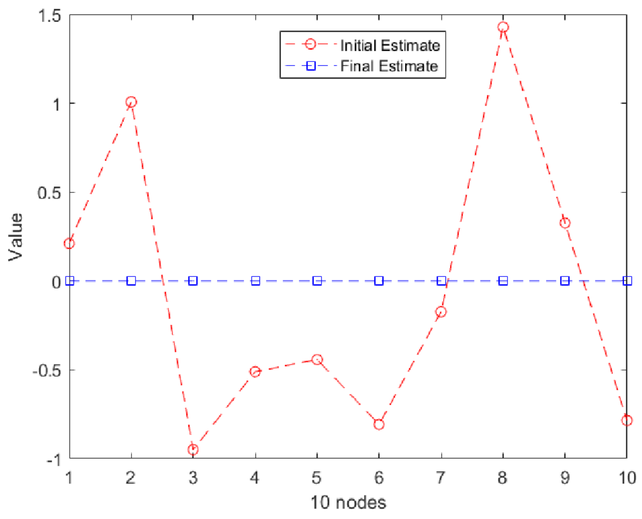

Fig. 5: Initial measurements are not exact to the final, but they are close.

Average Consensus

The average consensus convergence also uses a nodal network, similar to the weighted average. However, this one does not include noise variance or design factors. Two designs are used, the metropolis design and the max-degree design.

Metropolis Design

The Metropolis Design is the first of the average consensus designs. Instead, for all neighbors of each node, the weight is calculated using the max number between the number of neighbors between the node and the number of neighbors of the node's neighbors. The sum of the weights for all the nodes should sum up to 1. The convergence is shown in Fig. 6.

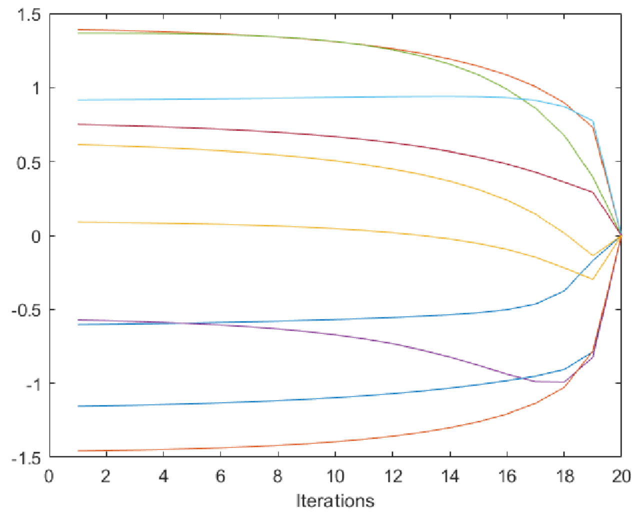

Fig. 6: Minimal convergence for each node until the final iterations.

Similar to weight design 2, there is difficulty for nodes to converge. The total weights of the nodes equal one, as expected. In Fig. 7, the estimate range is close to the actual estimate.

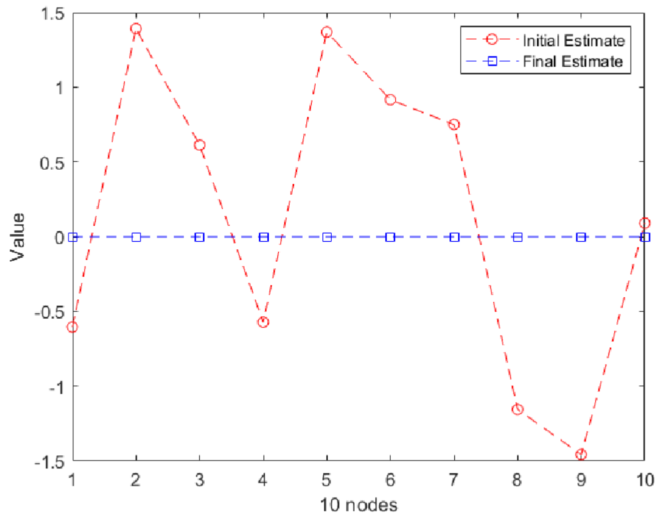

Fig. 7: The estimated range is similar to weight design 2 and max-degree design.

Maximum-Degree Design

The Maximum-Degree Design is the simplest of the weight designs used in the experiment. The weights depend entirely on the number of nodes used to model the system and the number of neighbors each node has. Fig. 8 shows the convergence of the maximum-degree design.

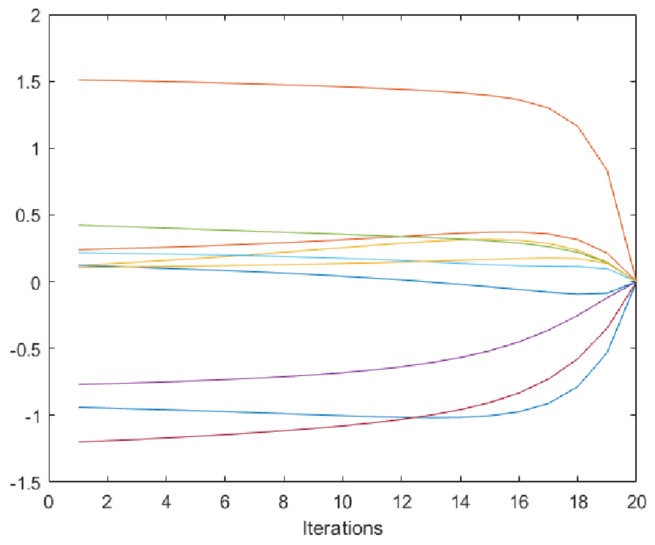

Fig. 8: Similar to the metropolis design but the curvature has increased to the convergence.

In order to optimize performance, the maximum-degree design should only be used when the number of nodes isn't too big. As the number of nodes increases, the size of the weights decreases. In the case of a larger node network, designs like Weight Design 2 should be used instead. The initial estimates using the maximum-degree function proved optimal outperforming the other designs as shown in Fig. 9. This may be due to the smaller size of the system used in the experiment.

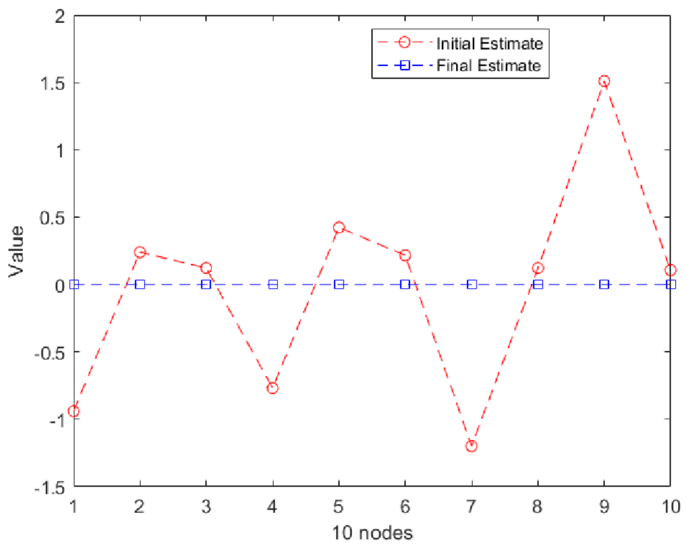

Fig. 9: Node ten, with the most neighbors, is the best initial estimate.

Network Convergence of 50 nodes

A network of 50 nodes performs just as well as a network of 10 nodes. The node with the least amount of neighbors in both cases converged the fastest in weight design. The node with the most neighbors had the closest initial estimate to the final estimate using the maximum-degree design. Fig. 10 shows the generated network of 50 nodes. While Fig. 11 shows the convergence of the network using weight design 1.

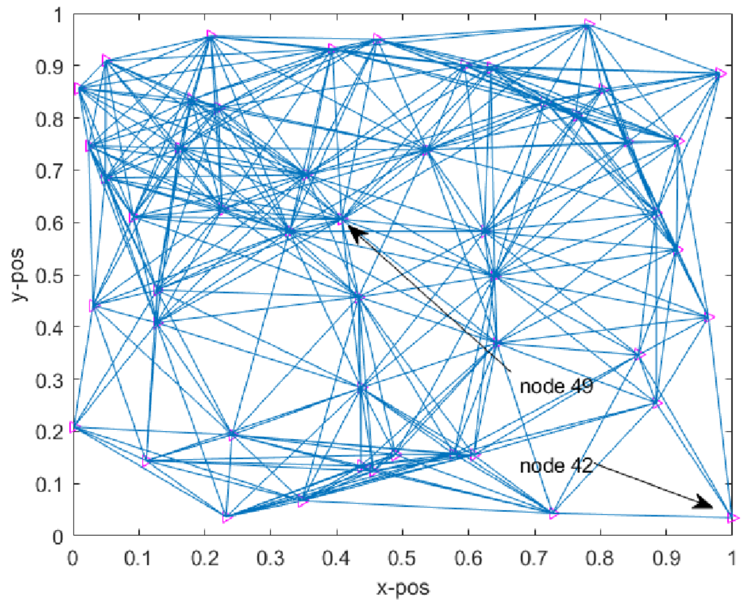

Fig. 10: Node 49 has 22 neighbors while node 42 has only six neighbors.

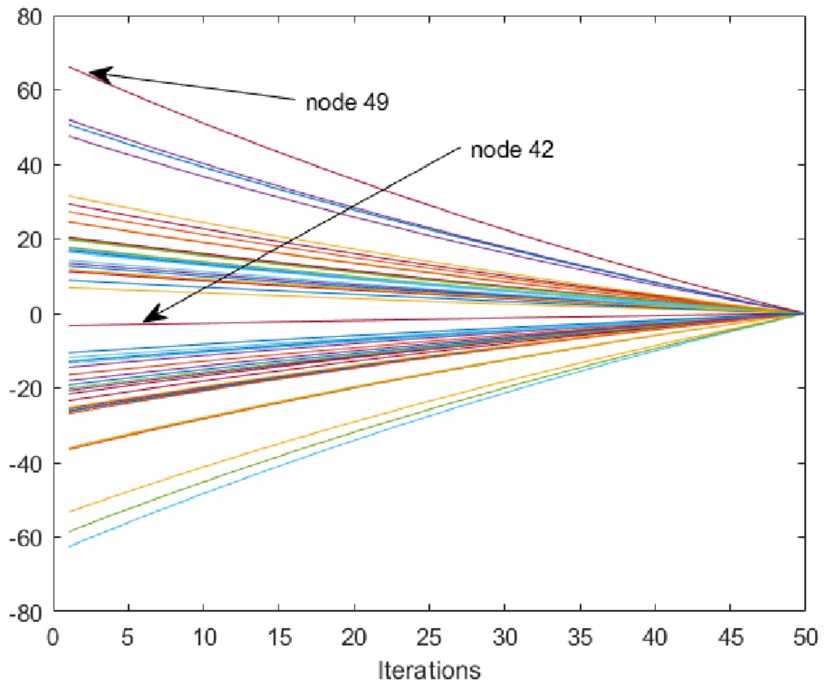

Fig. 11: Fifty nodes behave similarly to the ten nodes during convergence.

Weighted Average Consensus 3D map

Using a multi-cell field of 625 cells, a node network of 30 nodes is used for creating the map. The weighted average consensus is found using weight designs 1 and 2 and the average consensus should be found using the metropolis and maximum-degree designs. In Fig. 12 scalar map of the field is generated.

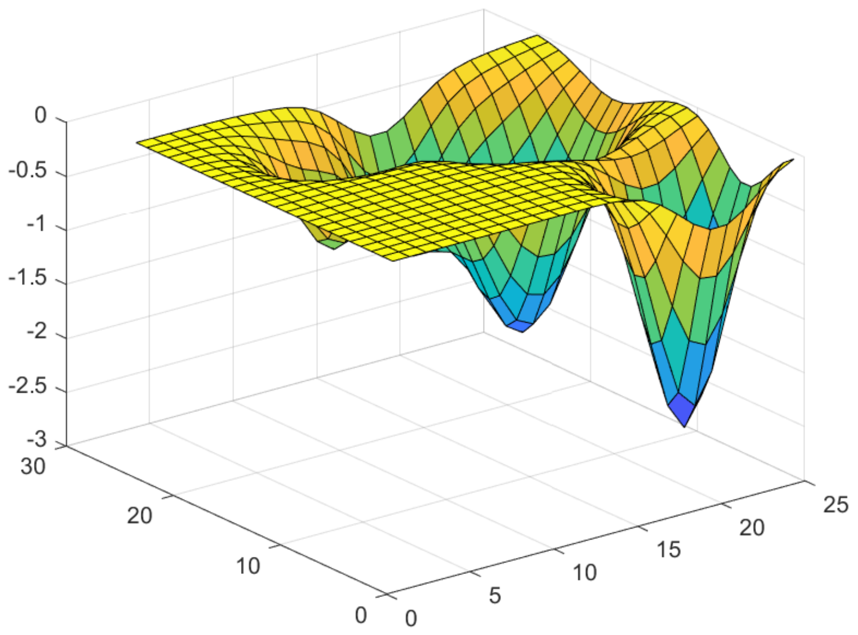

Fig. 12: 3D Scalar Map of Weighted Average Consensus

Appendix: Software Source Code

1. **main.m**

```
%Grant Davis
%CPE470
%Project 3: Consensus Filters for a Multi-robot System

clc
close all
clear
%code by Jim La
num_nodes = 50; %number of nodes
r = .4; %communication range <==========================Change r here
n = 2; % number of dimensions
delta_t_update = .008;
nodes = rand(num_nodes, n); %uncomment to generate new node set
%save('Nodes.txt', 'nodes');
%nodes = importdata('Nodes.txt');
F = 50;
%nodes = nodes * 4;
%Add measurement for each node
M = F * ones(num_nodes, 1) + 1 * randn(num_nodes, 1);
m_i = M; % Save initial measurement
[Nei_agent, A] = findneighbors(nodes, r, n, delta_t_update);
cv = .01;
W = zeros(num_nodes, num_nodes);%Weights initialized to 0
% generate design factor
c1_w = ((2 * cv) / (r * r * (num_nodes - 1))).* rand(1,1);
c2_w = (cv / (r * r)) .*rand(1,1);
%generate V
V = zeros(num_nodes, 1);
q_bar = mean(nodes);
%Q = transpose(q_bar);
for i = 1:num_nodes
  e_d = norm(nodes(i,:) - q_bar);
  V(i) = (e_d + cv) / (r * r);
end
%Select Weight Design
str = 'CPE470 Project 3:';
str = [str newline '1. Weight Design 1'];
str = [str newline  '2. Weight Design 2'];
str = [str newline '3. Metropolis Design'];
str = [str newline '4. Maximum Degree Design'];
str = [str newline 'Choice : '];
choice = input(str);
switch choice
  case 1 % Weight Design 1 set L to 500
      for i = 1:num_nodes % for each node
          for j = 1:num_nodes %for each weight
              for k = 1:size(Nei_agent{i})%for each neighbor
                  TEST = Nei_agent{i}(k);
                  if j == TEST
                      if i ~= j
                          W(i,j) = c1_w / (V(i) + V(j));
                      end
                  end
              end
          end
          for j = 1:num_nodes % for each weight where i equals j
              if i == j
                  W(i,j) = 1 - sum(W(i));
              end
          end
      end
  case 2 % Weight Design 2 set r to .5 and L to 50
      for i = 1:num_nodes %for each node
           W(i,i) = c2_w / V(i);
      end
      for i = 1:num_nodes % for each node
          for j = 1:num_nodes %for each weight
              m = size(Nei_agent{i});
              for k = 1:size(Nei_agent{i})%for each neighbor
                  TEST = Nei_agent{i}(k);
                  if j == TEST
                      if i ~= j
                          W(i,j) = (1 - W(i,i)) / m(:,1);
                      end
                  end
              end
          end
```

```
    end
case 3 % Metropolis Design set L to 15
    for i = 1:num_nodes %for each node
        for j = 1:num_nodes %for each weight
            for k = 1:size(Nei_agent{i})%for each neighbor
                TEST = Nei_agent{i}(k);
                if j == TEST
                    m = size(Nei_agent{i});
                    n = size(Nei_agent{j});
                    W(i,j) = 1 / (max(m(:,1), n(:,1)) + 1);
                end
            end
        end
        for j = 1:num_nodes %for each weight
            if i == j

                W(i,j) = 1 - sum(W(i,Nei_agent{i}));
            end
        end
    end
case 4 % Maximum Degree Design set L to 50
    for i = 1:num_nodes %for each node
        for j = 1:num_nodes%for each weight
            if i == j
                m = size(Nei_agent{i});
                W(i,j) = 1 - (m(:,1) / num_nodes);
            end
        end
        for j = 1:num_nodes %for each weight
            for k = 1:size(Nei_agent{i})%for each neighbor
                TEST = Nei_agent{i}(k);
                if j == TEST
                    W(i,j) = 1 /num_nodes;
                end
            end
        end
    end
otherwise % Terminate program
        disp('No Design selected\n');
        quit;
```

```
end
L = 100;%<=======================================Change L here
Val = zeros(9,1);
X = zeros(L, num_nodes);
X(1,:) = m_i;% first iteration
for j = 2:L %Weighted Average Consensus
  for i = 1:num_nodes
     temp1 = transpose(X((j - 1), Nei_agent{i}));
     temp2 = W(i,Nei_agent{i});
     Val = temp2 * temp1;
     X(j,i) = W(i,i) * X((j - 1),i) + Val;
  end
end
j = L;
C = zeros(L,num_nodes);
for i = 1:L %calculating convergence C
  for k = 1:num_nodes
     C(i,k) = (X(j,k) - X(1,k));
     if floor(k/2) == k/2
        C(i,k) = C(i,k) * -1;%flip every other node
     end
  end
  j = j - 1;
end
%plotting node network
figure(1), plot(nodes(:,1), nodes(:,2), 'm>', 'Linewidth', .2, ...
               'MarkerEdgeColor', 'm', ...
               'MarkerSize',5)
hold on
for i = 1:num_nodes
  %Line the neighbors together
  tmp = nodes(Nei_agent{i},:);
  for j = 1:size(nodes(Nei_agent{i},1))
     line([nodes(i,1), tmp(j,1)],[nodes(i,2),tmp(j,2)])
  end
end
%plotting Convergence
figure(2), plot(C)
%plotting final vs. intial measurements
figure(3), plot(C(1,:), '--ro')
```

```
hold on
plot(C(L,:), '--bs')
```

2. **findneighbors.m**

```
function [Nei_agent, A] = findneighbors(nodes,r,n, delta_t_update)
% Inputs: %Positions of nodes (robots/sensors),
        %Active sensing range (r)
        %Number of dimensions (n), 2D or 3D
        %Time step (delta_t_update)


% Outputs: %Indices of neighbors (Nei_agent)
         %Adjacency matrix (A)

%*******Find neighbors ***********************
num_nodes = size(nodes,1);
dif = cell(num_nodes,1); % save the difference between each alpha agent and all other nodes
                    % each element of cell is a matrix(size:num_nodes x n)
distance_alpha = zeros(num_nodes,num_nodes); % save the distance (norm) between each agent
and all other nodes
                          % each column for one node
Nei_agent = cell(num_nodes,1); %Save the indices of neighbors of each agent
                         %each element of cell is a matrix (maximum size num_nodes x 1)

for i = 1:num_nodes
  dif{i} = repmat(nodes(i,:),num_nodes,1)- nodes;
  tmp = dif{i}; %recall cell i th of dif
  for j = 1:num_nodes
      d_tmp(j,:) = norm(tmp(j,:)); %compute distance between each alpha agent and all other
nodes
  end
  distance_alpha(i,:)= d_tmp;
end
for k = 1:num_nodes
  Nei_agent{k} = find(distance_alpha(:,k)<r & distance_alpha(:,k)~=0); %find the neighbors of
agent i
end
%++++++++++++++++++BUILDING THE ADJACENCY
MATRIX+++++++++++++++++++++
A = [];
for i = 1:num_nodes
  Nei_node = nodes(Nei_agent{i},:); %temporary recall neighbors of agent jth
  for j = 1:num_nodes

      dist_2nodes= norm(nodes(j,:)-nodes(i,:));
      if dist_2nodes==0;
         A(i,j) = 0;
      elseif dist_2nodes<r & dist_2nodes~=0;
         A(i,j) = 1;
      else dist_2nodes>r;
         A(i,j) = 0;
      end
  end

end
```

3. Scalar_field_generation.m

```
% For CPE 470/670
% By Dr. Hung (Jim) La @ 2022
%====PARAMETER OF GAUSSIAN MODEL (MULTIVARIATE NORMAL
DISTRIBUTION)===
x_neg = -6;
x_pos = 6;
y_neg = -6;
y_pos = 6;
%Gaussian distribution 0
mu_x1 = 3;%2
mu_y1 = 2;%0
mu1 = [mu_x1 mu_y1];
variance_x1 = 2.25;
variance_y1 = 2.25;
rho1 = 0.1333; %correlation between 2 variabes (#1)
covariance_xy1 = rho1*sqrt(variance_x1*variance_y1);
%Sigma = 0.25*[.25 .3; .3 1];
Sigma1 = [variance_x1 covariance_xy1; covariance_xy1 variance_y1];
mu_x2 = 1;%0;
mu_y2 = 4.5;%2
mu2 = [mu_x2 mu_y2];
variance_x2 = 1.25;
```

```
variance_y2 = 1.25;
rho2 = 0.1333; %correlation between 2 variabes (#1)
covariance_xy2 = rho2*sqrt(variance_x2*variance_y2);
%Sigma = 0.25*[.25 .3; .3 1];
Sigma2 = [variance_x2 covariance_xy2; covariance_xy2 variance_y2];
mu_x3 = -2;%0;
mu_y3 = 3;%2
mu3 = [mu_x3 mu_y3];
variance_x3 = 1.25;
variance_y3 = 1.25;
rho3 = 0.1333; %correlation between 2 variabes (#1)
covariance_xy3 = rho3*sqrt(variance_x3*variance_y3);
%Sigma = 0.25*[.25 .3; .3 1];
Sigma3 = [variance_x3 covariance_xy3; covariance_xy3 variance_y3];
mu_x4 = 4;%0;
mu_y4 = -4;%2
mu4 = [mu_x4 mu_y4];
variance_x4 = 1.25;
variance_y4 = 1.25;
rho4 = 0.1333; %correlation between 2 variabes (#1)
covariance_xy4 = rho4*sqrt(variance_x4*variance_y4);
%Sigma = 0.25*[.25 .3; .3 1];
Sigma4 = [variance_x4 covariance_xy4; covariance_xy4 variance_y4];
%Devide the Region F into cells
scal = 0.5;%0.1;
x = x_neg:scal:x_pos; %x dimension of the survellance Region
y = y_neg:scal:y_pos; %y dimension of the survellance Region
[X1,X2] = meshgrid(x,y);
Theta = [30 10 8 20]; %The true constant vector
Phi = [mvnpdf([X1(:) X2(:)],mu1,Sigma1) mvnpdf([X1(:) X2(:)],mu2,Sigma2) mvnpdf([X1(:)
X2(:)],mu3,Sigma3) mvnpdf([X1(:) X2(:)],mu4,Sigma4)]; %The distribution of the environment
F1 = Theta*Phi';
num_cells = length(F1);
F = reshape(F1,length(y),length(x)); %Find cell's value (y by x matrix)
%F =[F2(2,:); F2(1,:)];
figure(1), surf(-F) %Plot the true map
```

Execution Commands

To execute the program using provided source code, must use MATLAB environment with all files in the same directory (main.m, Scalar_field_generation.m, Scalar_field_data.txt, and findneighbors.m). Open MATLAB and run main.m, this will prompt you to enter a choice into the command window. Choose your desired option from the UI. Next, with MATLAB open still run Scalar_field_generation.m.