```
clc;clear;

%===========Algorithm Parameters==============

               ALG_NUM = 2;

%===========Set Parameters====================
n = 2; %number of dimensions
Cv = .01;

num_nodes = 30; %Randomly generate nodes
grid_size = 25; %25x25
r = 17; %set communication range
r_s = 5*ones(num_nodes,1); %sensing range
scale = [.5 -6 6]; %for survaillance region [scale factor, range_min, range_max
    ]
meshdim = scale(2):scale(1):scale(3);

%generate scalar field with gaussian mixture
dim = [grid_size grid_size]; %dimensions
theta = [20 50 35 40]; %weights of the distributions
corr = .1333*ones(4,1); %correlations
var = [2.25 2.25; 1.25 1.25; 1.25 1.25; 1.25 1.25]; %variances
mu = [2 2; 1 .5; 4.3 3.5; 3 -3]; %means
F = generate_scalar_field(dim, scale, theta, corr, var, mu);

%compute graph
nodes = (scale(3)-scale(2))*rand(num_nodes, n) + scale(2); %Randomly generate
    initial positions of MSN
[Nei_agent, A] = findneighbors(nodes, n, r);

plot_graph(true, num_nodes, nodes, Nei_agent)
plot_graph(false, num_nodes, nodes, Nei_agent)

%compute variance vector
Var = zeros(num_nodes,grid_size,grid_size);
q_ave = mean(nodes);
for k = 1:num_nodes
    for i = 1:grid_size
        for j = 1:grid_size
            Var(k,i,j) = (norm(nodes(k,:) - [meshdim(i) meshdim(j)])^2 + Cv) /
                (r_s(k)^2); %variance matrix
        end
    end
end

%set initial measurement matrix
nodes_va = zeros(num_nodes,grid_size,grid_size);
for k = 1:num_nodes
    for i = 1:grid_size
        for j = 1:grid_size
            if norm(nodes(k,:) - [meshdim(i) meshdim(j)]) <= r_s(k) %acts as
                observance matrix
                nodes_va(k,i,j) = F(i,j) + normrnd(0,Var(k,i,j)); %sets noisy
                    measurement
            else
```

```
%========================================================
%Anthony Bugatto
%CS 455: Mobile Sensor Netorks
%Project 1: Flocking
%========================================================

                    clc,clear
                    close all

%================ALGORITHM NUMBER======================

                    ALG_NUM = 5;

%=============PARAMETER OF SIMULATION==================

d = 15;%Set desired distance among sensor nodes
k_scale = 1.2;%Set the scale of MSN
r = k_scale*d; %Set the active range
r_prime = .22*k_scale*r; %Set the active range of beta agent
epsilon = 0.1; %Set a constant for sigma norm
num_nodes = 10; %Set number of sensor nodes
n=2; %Set number of dimensions

F = 50; %actual value
Cv = .01;

    grid_size = 20; %20x20
    r = 17; %set communication range
    r_s = 17*ones(num_nodes,1); %measurement radius
    num_nodes = 10; %Randomly generate nodes
    grid_size = 4; %4x4
    r = 2; %set communication range
    r_s = 3*ones(num_nodes,1); %measurement radius

%Place F Value on Graph
subplot(1,3,1);
plot(.5*grid_size, .5*grid_size, 's', 'LineWidth' ,.1 , 'MarkerEdgeColor', 'r',
    'MarkerFaceColor', 'r', 'MarkerSize', 10)
hold on

nodes = grid_size.*rand(num_nodes, n) + grid_size.*repmat([0 1], num_nodes, 1);
    %Randomly generate initial positions of MSN
p_nodes = zeros(num_nodes,n); %Set initial velocties of MSN
delta_t_update = 0.08; %Set time step
t = 0:delta_t_update:7;% Set simulation time

if ALG_NUM == 5 %=================SET OBSTACLES=========================
    obstacles = [50, 100; 150 80; 200, 230; 280 150]; %set positions of
        obstacles
    Rk = [20; 10; 15; 8]; %Radii of obstacles
    num_obstacles = size(obstacles,1); %Find number of obstacles
end

if ALG_NUM ~= 1 %=========SET A STATIC TARGET===============
    qt1 = [150 150]; %Set position of the static target (gamma agent)
    pt1= [0 0]; %Set initial velocity of the target
```

```
%===========Algorithm Parameters==============

               ALG_NUM = 2;

%===========Set Parameters====================
n = 2; %number of dimensions
Cv = .01;

num_nodes = 30; %Randomly generate nodes
grid_size = 25; %25x25
r = 17; %set communication range
r_s = 5*ones(num_nodes,1); %sensing range
scale = [.5 -6 6]; %for survaillance region [scale factor, range_min, range_max
    ]
meshdim = scale(2):scale(1):scale(3);

%generate scalar field with gaussian mixture
dim = [grid_size grid_size]; %dimensions
theta = [20 50 35 40]; %weights of the distributions
corr = .1333*ones(4,1); %correlations
var = [2.25 2.25; 1.25 1.25; 1.25 1.25; 1.25 1.25]; %variances
mu = [2 2; 1 .5; 4.3 3.5; 3 -3]; %means
F = generate_scalar_field(dim, scale, theta, corr, var, mu);

%compute graph
nodes = (scale(3)-scale(2))*rand(num_nodes, n) + scale(2); %Randomly generate
    initial positions of MSN
[Nei_agent, A] = findneighbors(nodes, n, r);

plot_graph(true, num_nodes, nodes, Nei_agent)
plot_graph(false, num_nodes, nodes, Nei_agent)

%compute variance vector\
Var = zeros(num_nodes,grid_size,grid_size);
q_ave = mean(nodes);
for k = 1:num_nodes
    for i = 1:grid_size
        for j = 1:grid_size
            Var(k,i,j) = (norm(nodes(k,:) - [meshdim(i) meshdim(j)])^2 + Cv) /
                (r_s(k)^2); %variance matrix
        end
    end
end

%set initial measurement matrix
nodes_va = zeros(num_nodes,grid_size,grid_size);
for k = 1:num_nodes
    for i = 1:grid_size
        for j = 1:grid_size
            if norm(nodes(k,:) - [meshdim(i) meshdim(j)]) <= r_s(k) %acts as
                observance matrix
                nodes_va(k,i,j) = F(i,j) + .001*normrnd(0,Var(i)); %sets noisy
                    measurement
            else
                nodes_va(k,i,j) = 0; %sets to zero
            end
        end
    end
end
nodes_va0 = nodes_va; %save the initial measurement

%Implement Algorithm
iteration = 2;
nodes_va_next = zeros(num_nodes,grid_size,grid_size);
nodes_va_f = zeros(num_nodes,grid_size,grid_size);
history{1} = nodes_va0;
while(1) %loop until convergance (then break)
    for k = i:grid_size     %loop through grid
        for l = 1:grid_size %... and then do same operation as p1
            for i = 1:num_nodes %sum the neighbor weights
                %compute Wii
                if ALG_NUM == 1 %weighted alg2
                    Wii = weighted_design2(i ,i ,Var(:,k,l) ,Nei_agent ,Cv ,r ,
                        r_s);
                elseif ALG_NUM == 2 %metropolis
                    Wii = weighted_metropolis(i, i, Nei_agent);
                end

                %compute sum of Wij_k*X_j_k
                sum = 0;
                for j = 1:size(Nei_agent{i},1) %iterates through neighbors
                    if ALG_NUM == 1 %weighted alg2
                        sum = sum + (nodes_va(Nei_agent{i}(j),k,l) *
                            weighted_design2(i ,Nei_agent{i}(j) ,Var(:,k,l) ,
                            Nei_agent ,Cv ,r ,r_s));
                    elseif ALG_NUM == 2 %metropolis
                        sum = sum + (nodes_va(Nei_agent{i}(j),k,l) *
                            weighted_metropolis(i, Nei_agent{i}(j), Nei_agent))
                            ;
                    end
                end

                %get estimates X_i_k
                nodes_va_next(i,k,l) = Wii*nodes_va(i,k,l) + sum;
            end
        end
    end

    nodes_va = nodes_va_next; %iterate estimates
    history{iteration} = nodes_va;
    %plot_error(F, nodes_va_next, nodes_va_f, scale);

    error = zeros(size(meshdim,2), size(meshdim,2));
    final = zeros(size(meshdim,2), size(meshdim,2));
    for k = 1:size(meshdim,2)
        for l = 1:size(meshdim,2)
            final(k,l) = nodes_va_next(1,k,l);
            error(k,l) = F(k,l) - final(k,l);
        end
    end

    subplot(2,3,4), %new grid
    pcolor(meshdim, meshdim, -final)
    hold on

    subplot(2,3,5), %new mesh
    surf(meshdim, meshdim, -final)
    hold on

    subplot(2,3,3), %stem error
    stem(error(:))
    hold on

    subplot(2,3,6), %mesh error
    surf(meshdim, meshdim, error)
    hold on
end
```

```
%===========Algorithm Parameters==============

               ALG_NUM = 2;

               TEST = true;

%===========Set Parameters====================
n = 2; %number of dimensions
F = 50; %actual value
Cv = .01;

if ALG_NUM == 5 || ALG_NUM == 6
    num_nodes = 50; %Randomly generate nodes
    grid_size = 20; %20x20
    r = 17; %set communication range
    r_s = 17*ones(num_nodes,1); %measurement radius
else
    num_nodes = 20; %Randomly generate nodes
    grid_size = 4; %4x4
    r = 2; %set communication range
    r_s = 3*ones(num_nodes,1); %measurement radius
end

%Place F Value on Graph
subplot(1,3,1);
plot(.5*grid_size, .5*grid_size, 's', 'LineWidth' ,.1 , 'MarkerEdgeColor', 'r',
    'MarkerFaceColor', 'r', 'MarkerSize', 10)
hold on

%compute graph
nodes = grid_size.*rand(num_nodes, n); %Randomly generate initial positions of
    MSN
[Nei_agent, A] = findneighbors(nodes, n, r);
subplot(1,3,1), plot(nodes(:,1), nodes(:,2), 'o', 'LineWidth' ,.2 ,
    'MarkerEdgeColor', 'b', 'MarkerFaceColor', 'm', 'MarkerSize', 5)
hold on

for node_i = 1:num_nodes %graph lines between the nodes
    tmp=nodes(Nei_agent{node_i},:);
    for j = 1:size(nodes(Nei_agent{node_i},1))
        line([nodes(node_i,1),tmp(j,1)],[nodes(node_i,2),tmp(j,2)])
    end
end

%compute grid average
x_ave = 0;
y_ave = 0;
for i = 1:num_nodes
    x_ave = x_ave + nodes(i,1);
    y_ave = y_ave + nodes(i,2);
end
q_ave = (1/num_nodes)*[x_ave y_ave];

%Compute Variance Matrix (works for all cases in static system
Var = zeros(num_nodes,1);
for i = 1:num_nodes
```

```
%===========Algorithm Parameters==============

               ALG_NUM = 2;

%===========Set Parameters====================
n = 2; %number of dimensions
Cv = .01;

num_nodes = 30; %Randomly generate nodes
grid_size = 25; %25x25
r = 17; %set communication range
r_s = 5*ones(num_nodes,1); %sensing range
scale = [.5 -6 6]; %for survaillance region [scale factor, range_min, range_max
    ]
meshdim = scale(2):scale(1):scale(3);

%generate scalar field with gaussian mixture
dim = [grid_size grid_size]; %dimensions
theta = [20 50 35 40]; %weights of the distributions
corr = .1333*ones(4,1); %correlations
var = [2.25 2.25; 1.25 1.25; 1.25 1.25; 1.25 1.25]; %variances
mu = [2 2; 1 .5; 4.3 3.5; 3 -3]; %means
F = generate_scalar_field(dim, scale, theta, corr, var, mu);

%compute graph
nodes = (scale(3)-scale(2))*rand(num_nodes, n) + scale(2); %Randomly generate
    initial positions of MSN
[Nei_agent, A] = findneighbors(nodes, n, r);

plot_graph(true, num_nodes, nodes, Nei_agent)
plot_graph(false, num_nodes, nodes, Nei_agent)

%compute variance vector\
Var = zeros(num_nodes,grid_size,grid_size);
q_ave = mean(nodes);
for k = 1:num_nodes
    for i = 1:grid_size
        for j = 1:grid_size
            Var(k,i,j) = (norm(nodes(k,:) - [meshdim(i) meshdim(j)])^2 + Cv) /
                (r_s(k)^2); %variance matrix
        end
    end
end

%set initial measurement matrix
nodes_va = zeros(num_nodes,grid_size,grid_size);
for k = 1:num_nodes
    for i = 1:grid_size
        for j = 1:grid_size
            if norm(nodes(k,:) - [meshdim(i) meshdim(j)]) <= r_s(k) %acts as
                observance matrix
                nodes_va(k,i,j) = F(i,j) + .001*normrnd(0,Var(i)); %sets noisy
                    measurement
            else
                nodes_va(k,i,j) = 0; %sets to zero
            end
        end
    end
end
nodes_va0 = nodes_va; %save the initial measurement

%Implement Algorithm
iteration = 2;
nodes_va_next = zeros(num_nodes,grid_size,grid_size);
nodes_va_f = zeros(num_nodes,grid_size,grid_size);
history{1} = nodes_va0;
while(1) %loop until convergance (then break)
    for k = i:grid_size     %loop through grid
        for l = 1:grid_size %... and then do same operation as p1
            for i = 1:num_nodes %sum the neighbor weights
                %compute Wii
                if ALG_NUM == 1 %weighted alg2
                    Wii = weighted_design2(i ,i ,Var(:,k,l) ,Nei_agent ,Cv ,r ,
                        r_s);
                elseif ALG_NUM == 2 %metropolis
                    Wii = weighted_metropolis(i, i, Nei_agent);
                end

                %compute sum of Wij_k*X_j_k
                sum = 0;
                for j = 1:size(Nei_agent{i},1) %iterates through neighbors
                    if ALG_NUM == 1 %weighted alg2
                        sum = sum + (nodes_va(Nei_agent{i}(j),k,l) *
                            weighted_design2(i ,Nei_agent{i}(j) ,Var(:,k,l) ,
                            Nei_agent ,Cv ,r ,r_s));
                    elseif ALG_NUM == 2 %metropolis
                        sum = sum + (nodes_va(Nei_agent{i}(j),k,l) *
                            weighted_metropolis(i, Nei_agent{i}(j), Nei_agent))
                            ;
                    end
                end

                %get estimates X_i_k
                nodes_va_next(i,k,l) = Wii*nodes_va(i,k,l) + sum;
            end
        end
    end

    nodes_va = nodes_va_next; %iterate estimates
    history{iteration} = nodes_va;
    %plot_error(F, nodes_va_next, nodes_va_f, scale);

    error = zeros(size(meshdim,2), size(meshdim,2));
    final = zeros(size(meshdim,2), size(meshdim,2));
    for k = 1:size(meshdim,2)
        for l = 1:size(meshdim,2)
            final(k,l) = nodes_va_next(1,k,l);
            error(k,l) = F(k,l) - final(k,l);
        end
    end

    subplot(2,3,4), %new grid
    pcolor(meshdim, meshdim, -final)
    hold on

    subplot(2,3,5), %new mesh
    surf(meshdim, meshdim, -final)
    hold on

    subplot(2,3,3), %stem error
    stem(error(:))
    hold on

    subplot(2,3,6), %mesh error
    surf(meshdim, meshdim, error)
    hold on
end
```

```
%=========================================================
%Anthony Bugatto
%CS 455: Mobile Sensor Netorks
%Project 1: Flocking
%=========================================================

                    clc,clear
                    close all

%================ALGORITHM NUMBER=======================

                    ALG_NUM = 5;

%=============PARAMETER OF SIMULATION===================

d = 15;%Set desired distance among sensor nodes
k_scale = 1.2;%Set the scale of MSN
r = k_scale*d; %Set the active range
r_prime = .22*k_scale*r; %Set the active range of beta agent
epsilon = 0.1; %Set a constant for sigma norm
num_nodes = 10; %Set number of sensor nodes
n=2; %Set number of dimensions

F = 50; %actual value
Cv = .01;

    grid_size = 20; %20x20
    r = 17; %set communication range
    r_s = 17*ones(num_nodes,1); %measurement radius
    num_nodes = 10; %Randomly generate nodes
    grid_size = 4; %4x4
    r = 2; %set communication range
    r_s = 3*ones(num_nodes,1); %measurement radius

%Place F Value on Graph
subplot(1,3,1);
plot(.5*grid_size, .5*grid_size, 's', 'LineWidth' ,.1 , 'MarkerEdgeColor', 'r',
    'MarkerFaceColor', 'r', 'MarkerSize', 10)
hold on

nodes = grid_size.*rand(num_nodes, n) + grid_size.*repmat([0 1], num_nodes, 1);
    %Randomly generate initial positions of MSN
p_nodes = zeros(num_nodes,n); %Set initial velocties of MSN
delta_t_update = 0.08; %Set time step
t = 0:delta_t_update:7;% Set simulation time

if ALG_NUM == 5 %=================SET OBSTACLES=========================
    obstacles = [50, 100; 150 80; 200, 230; 280 150]; %set positions of
        obstacles
    Rk = [20; 10; 15; 8]; %Radii of obstacles
    num_obstacles = size(obstacles,1); %Find number of obstacles
end

if ALG_NUM ~= 1 %=========SET A STATIC TARGET===============
    qt1 = [150 150]; %Set position of the static target (gamma agent)
    pt1= [0 0]; %Set initial velocity of the target
```

```
end

nodes_old = nodes; %KEEP previous positions of MSN
q_mean = zeros(size(t,2), n); %Save positions of COM (Center of Mass)
p_mean = zeros(size(t,2), n); %Save velocities of COM (Center of Mass)
Connectivity = []; %save connectivity of MSN
q_nodes_all = cell(size(t,2), num_nodes); %creates cell array to store history
    of system pos
p_nodes_all = cell(size(t,2), num_nodes); % -   -   -   -   -   -   -   -   -
    -   -   -vel

%compute grid average
x_ave = 0;
y_ave = 0;
for i = 1:num_nodes
    x_ave = x_ave + nodes(i,1);
    y_ave = y_ave + nodes(i,2);
end
q_ave = (1/num_nodes)*[x_ave y_ave];

%Compute Variance Matrix (works for all cases in static system
Var = zeros(num_nodes,1);
for i = 1:num_nodes
    diff = norm(nodes(i,:) - q_ave);
    Var(i) = (diff^2 + Cv) / (r_s(i)^2);
end

%initial value
nodes_va = F.*ones(num_nodes,1) + normrnd(0,Var(:)); %Add measurement for each
    node: yi= theta + v_i
nodes_va0 = nodes_va; %save the initial measurement

%Define History
history = []; %add nodes_va_next after each iteration
history(1,:) = nodes_va0; %each column is a history vector

nFrames = 20; %set number of frames for the movie
mov(1:nFrames) = struct('cdata', [],'colormap', []); %Preallocate movie
    structure
iteration = 2;
nodes_ave = zeros(num_nodes,1);
nodes_wht = zeros(num_nodes,1);
nodes_avef = zeros(num_nodes,1);
nodes_whtf = zeros(num_nodes,1);
for iteration = 1:length(t)
            %Line Trajectory of a moving target
            qt_x1 = t(iteration);
            qt_y1 = t(iteration);

            %compute position of target
            qt1(iteration,:) = [qt_x1, qt_y1];

            %compute velocities of target
            pt1(iteration,:) = (qt1(iteration,:) - qt1(iteration-1,:)) /
                delta_t_update;
```

```
        plot(qt1(:,1),qt1(:,2),'ro','LineWidth',2,'MarkerEdgeColor','r',
            'MarkerFaceColor','r', 'MarkerSize',4.2)
        hold on

    %=====================(LOOP) UPDATE PROCESS for MSN=====================

        [Nei_agent, A] = findneighbors1(nodes, r);
        [Ui] = inputcontrol_Algorithm2(ALG_NUM, num_nodes, nodes, Nei_agent, n,
            epsilon, r, d, qt1, pt1, p_nodes);

    p_nodes = (nodes - nodes_old)/delta_t_update; %COMPUTE velocities of sensor
        nodes
    p_nodes_all{iteration} = p_nodes; %SAVE VELOCITY OF ALL NODES
    nodes_old = nodes;
    nodes = nodes_old + p_nodes*delta_t_update + .5*Ui*delta_t_update*
        delta_t_update;
    q_mean(iteration,:) = mean(nodes); %Compute position of COM of MSN

    for l = 1:num_nodes %sum the neighbor weights
        %compute Wii
        Wii = weighted_design2(l ,l ,Var ,Nei_agent ,Cv ,r ,r_s);

        %compute sum of Wij*X_j
        sum1 = 0;
        sum2 = 0;
        for m = 1:size(Nei_agent{l},1) %iterates through neighbors
            sum1 = sum1 + (nodes_va(Nei_agent{l}(m)) * weighted_design2(i ,
                Nei_agent{l}(m) ,Var ,Nei_agent ,Cv ,r ,r_s));
            sum2 = sum2 + (nodes_va(Nei_agent{l}(m)) * weighted_metropolis(i,
                Nei_agent{l}(m), Nei_agent));
        end

        %get estimates X_i
        nodes_ave(l) = Wii*nodes_va(l) + sum1;
        nodes_wht(l) = Wii*nodes_va(l) + sum2;
    end

    if ALG_NUM ~= 1
        plot(q_mean(:,1),q_mean(:,2),'ro','LineWidth',2,'MarkerEdgeColor','k',
            'MarkerFaceColor','k','MarkerSize',4.2)
        hold on
    end

    p_mean(iteration,:) = mean(p_nodes); %Compute velocity of COM of MSN
    q_nodes_all{iteration} = nodes;
    Connectivity(iteration)= (1 / num_nodes) * rank(A);

    if ALG_NUM == 5 %Draw obstacles
         phi = 0:.1:2*pi;
         for k = 1:num_obstacles
            X = Rk(k)*cos(phi);
            Y = Rk(k)*sin(phi);
            plot(X+obstacles(k,1),Y+obstacles(k,2),'r',nodes(:,1),nodes(:,2),
                'g>')
            fill(X+obstacles(k,1),Y+obstacles(k,2),'r')
            axis([0 250 0 80]);
```

```
                hold on
             end
        end

        %================== PLOT and LINK SENSOR TOGETHER ================
        plot(nodes(:,1),nodes(:,2), '.')
        hold on
        plot(nodes(:,1),nodes(:,2), 'k>','LineWidth',.2,'MarkerEdgeColor','k',
            'MarkerFaceColor','k','MarkerSize',5)
        hold off

        for node_i = 1:num_nodes
            tmp=nodes(Nei_agent{node_i},:);
            for j = 1:size(nodes(Nei_agent{node_i},1))
                line([nodes(node_i,1),tmp(j,1)],[nodes(node_i,2),tmp(j,2)])
            end
        end

        convergance = false;
        conv_val = nodes_va_next(1);
        for i = 2:num_nodes %check if converged within .001%
            if (conv_val - nodes_va_next(i)) < .001 %breaks if not in conscensus
                if i == 10
                    convergance = true;
                    nodes_va_f = conv_val; %set final converged estimate
                end
            end
        end

        if convergance == true
            break
        end

        history(iteration,:) = nodes_va_next; %add estimate to history
        nodes_va = nodes_va_next; %iterate estimates
        mov(iteration) = getframe;
        hold off
    end



    %===========================VIDEO SIMULATION=============================
    %{
    v = VideoWriter('flocking.avi', 'MPEG-4'); %Make movie
    open(v)
    writeVideo(v,mov);
    %}
    %=========================PLOT VELOCITY OF MSN===========================

    p_each_nodes = [];
    for i = 2:size(t,2) %iterates through the timesteps for the history cell matrix
        tmp7 = p_nodes_all{i};
        for j = 1:num_nodes
            if j == 1 %Plot velociy of sensor node 1; you can change this number to
                plot for other nodes
                p_each_nodes(i) = norm(tmp7(j,:));
                figure(3), plot(p_each_nodes, 'b')
                hold on
            end
        end
    end

    figure(4), plot(Connectivity)
    grid on

    %========================PLOT TRAJECTORY OF SENSOR NODES===============

    for i = 2:length(q_nodes_all)
        tmp8 = q_nodes_all{i};
        figure(5), plot(tmp8(:,1), tmp8(:,2), 'k.')
        hold on
    end

    hold on
    plot(nodes(:,1), nodes(:,2), 'm>', 'LineWidth', .2, 'MarkerEdgeColor', 'm',
        'MarkerFaceColor', 'm', 'MarkerSize', 5)

    %========================PLOT TRAJECTORY OF COM AND TARGET===============

    figure(6), plot(q_mean(:,1), q_mean(:,2),'k.')
    hold on

    if ALG_NUM ~= 1 || ALG_NUM ~= 2
        plot(qt1(:,1), qt1(:,2), 'r.')
    end
```

```
        diff = norm(nodes(i,:) - q_ave);
        Var(i) = (diff^2 + Cv) / (r_s(i)^2);
    end

    %initial value
    nodes_va = F.*ones(num_nodes,1) + normrnd(0,Var(:)); %Add measurement for each
        node: yi= theta + v_i
    nodes_va0 = nodes_va; %save the initial measurement

    %Define History
    history = []; %add nodes_va_next after each iteration
    history(1,:) = nodes_va0; %each column is a history vector

    %Implement Algorithm
    iteration = 2;
    nodes_va_next = zeros(num_nodes,1);
    nodes_va_f = zeros(num_nodes,1);
    while(1) %loop until convergance (then break)
        for i = 1:num_nodes %sum the neighbor weights
            %compute Wii
            if ALG_NUM == 1 %weighted alg1
                Wii = weighted_design1(i ,i ,Var ,Nei_agent ,Cv ,r ,r_s);
            elseif ALG_NUM == 2 %weighted alg2
                Wii = weighted_design2(i ,i ,Var ,Nei_agent ,Cv ,r ,r_s);
            elseif ALG_NUM == 3 || ALG_NUM == 5 %metropolis
                Wii = weighted_metropolis(i, i, Nei_agent);
            elseif ALG_NUM == 4 || ALG_NUM == 6 %max degree
                Wii = weighted_max_degree(i, i, Nei_agent);
            end

            %compute sum of Wij*X_j
            sum = 0;
            for j = 1:size(Nei_agent{i},1) %iterates through neighbors
                if ALG_NUM == 1 %weighted alg1 -> (Nei_agent{i}(j) is index of j
                    node)
                    sum = sum + (nodes_va(Nei_agent{i}(j)) * weighted_design1(i ,
                        Nei_agent{i}(j) ,Var ,Nei_agent ,Cv ,r ,r_s));
                elseif ALG_NUM == 2 %weighted alg2
                    sum = sum + (nodes_va(Nei_agent{i}(j)) * weighted_design2(i ,
                        Nei_agent{i}(j) ,Var ,Nei_agent ,Cv ,r ,r_s));
                elseif ALG_NUM == 3 || ALG_NUM == 5 %metropolis
                    sum = sum + (nodes_va(Nei_agent{i}(j)) * weighted_metropolis(i,
                        Nei_agent{i}(j), Nei_agent));
                elseif ALG_NUM == 4 || ALG_NUM == 6 %max degree
                    sum = sum + (nodes_va(Nei_agent{i}(j)) * weighted_max_degree(i,
                        Nei_agent{i}(j), Nei_agent));
                end
            end

            %get estimates X_i
            nodes_va_next(i) = Wii*nodes_va(i) + sum;
        end

        convergance = false;
        conv_val = nodes_va_next(1);
        for i = 2:num_nodes %check if converged within .001%
```

```
            if (conv_val - nodes_va_next(i)) < .001 %breaks if not in conscensus
                if i == 10
                    convergance = true;
                    nodes_va_f = conv_val; %set final converged estimate
                end
            end
        end

        if convergance == true
            break
        end

        history(iteration,:) = nodes_va_next; %add estimate to history
        nodes_va = nodes_va_next; %iterate estimates
        iteration = iteration + 1;

        if TEST == true
            plot_data(TEST, num_nodes, history, nodes_va_next, nodes_va_f,
                nodes_va0)
        end
    end

    if TEST == false
        plot_data(TEST, num_nodes, history, nodes_va_next, nodes_va_f, nodes_va0);
    end
```

```
                nodes_va(k,i,j) = 0; %sets to zero
            end
        end
    end
end
nodes_va0 = nodes_va; %save the initial measurement

%Implement Algorithm
iteration = 2;
nodes_va_next = zeros(num_nodes,grid_size,grid_size);
nodes_va_f = zeros(num_nodes,grid_size,grid_size);
history = {};
history{1} = nodes_va0;

weights = zeros(25,25);
while(iteration ~= 1000) %loop until convergance (then break)
    for k = i:grid_size     %loop through grid
        for l = 1:grid_size %... and then do same operation as p1
            for i = 1:num_nodes %sum the neighbor weights
                %compute Wii
                if ALG_NUM == 1 %weighted alg2
                    Wii = weighted_design2(i ,i ,Var(:,k,l) ,Nei_agent ,Cv ,r ,
                        r_s);
                elseif ALG_NUM == 2 %metropolis
                    Wii = weighted_metropolis(i, i, Nei_agent);
                end

                %compute sum of Wij_k*X_j_k
                sum = 0;
                for j = 1:size(Nei_agent{i},1) %iterates through neighbors
                    if ALG_NUM == 1 %weighted alg2
                        sum = sum + (nodes_va(Nei_agent{i}(j),k,l) *
                            weighted_design2(i ,Nei_agent{i}(j) ,Var(:,k,l) ,
                            Nei_agent ,Cv ,r ,r_s));
                    elseif ALG_NUM == 2 %metropolis
                        sum = sum + (nodes_va(Nei_agent{i}(j),k,l) *
                            weighted_metropolis(i, Nei_agent{i}(j), Nei_agent))
                            ;
                        weights(k,l) = weights(k,l) + weighted_metropolis(i,
                            Nei_agent{i}(j), Nei_agent);
                    end
                end

                %get estimates X_i_k
                nodes_va_next(i,k,l) = Wii*nodes_va(i,k,l) + sum;
            end
        end
    end

    nodes_va = nodes_va_next; %iterate estimates
    history{iteration} = nodes_va;

    meshdim = scale(2):scale(1):scale(3); %mesh dimensions
    error = zeros(size(meshdim,2), size(meshdim,2));
    final = zeros(size(meshdim,2), size(meshdim,2));
    for k = 1:size(meshdim,2)
```

```
            for l = 1:size(meshdim,2)
                %final(k,l) = nodes_va0(1,k,l);
                final(k,l) = nodes_va_next(1,k,l); %assume one node is correct
                error(k,l) = F(k,l) - final(k,l);
            end
        end

        hold on
        subplot(2,3,4) %new grid
        pcolor(meshdim, meshdim, -final)
        hold off

        hold on
        subplot(2,3,5) %new mesh
        surf(meshdim, meshdim, -final)
        hold off

        hold on
        subplot(2,3,3) %stem error
        stem(-error(:))
        hold off

        hold on
        subplot(2,3,6) %mesh error
        surf(meshdim, meshdim, -error)
        hold off

        iteration = iteration + 1
    end
```