TRANSLATION OF THE CODE

Everything is green are comments, titles & explanations. They also have '%' at the beginning of the sentence. The rest is code.

```
clear; close all; clc; %clears workspace & command window & closes all figs


N = 10e2;    % Our population size aka the Agents (dots) in the graph


% Initialize Agents : we tell the program to create a graph with the following:
for k=1:N
    agents(k).pos = rand(1,2)*4; % This is the position of our dots which is random
    agents(k).infect = 0; % The number of dots infected which at the beginning is 0
    agents(k).infectday = 1; % The day which the dots can start infecting (i think????)
    agents(k).symptoms = true;% Dots that if infected show symptoms(IMPORTANT FOR FACTOR QUARANTINE)
end

%% Tick Days
% 0 = S, 1 = I, 2 = R
% Parameters : this is what we can modify depending on the simulation that is carried on
initially_infected_percentage = 0.05;  % Only 0.05% of our population is infected initially
recoverytimemean = 12; % Average days to get recovered
recoverytimestd = 1.5; %1.5*2  NOT SURE
infectradius = 0.1;  % The radius of the dots where they can infect other dots.

maxinfectchance = 0.7;  % The probability of getting infected when being in the infect radius
f = @(x)(maxinfectchance/ infectradius)*x; % Linear function to have the probability of getting infected
increase the closer the dot is to the infectious radius
```

```
quarantine = true;
central_hub = true;
isodays = 2; % When infected, the days before they are put into quarantine
percentagesymptoms = 0.8; % Probability of infected dots that show symptoms (symptomatic)
probability_central_hub = 0.02; % Probability of a dot going to the central hub



different_speeds = true;  %dots move at different speeds: to simulate different types of people (some might go to more places, see more people, etc)
percentage_fast=0.3; % 30% of the population goes faster
percentage_slow=0.7;

percentage_social_distancing=1; % percentage of people social distancing
social_distancing =true;
social_distance = 0.05; %how much social distancing is

timesteps_per_day = 4; %how many time the dots move in a day




dayz = 180; %The days the simulation goes through

i0 = randi(N,N*initially_infected_percentage,1);  % choose 5 random initially-infected agents.
for k = 1:length(i0)
    agents(i0(k)).infect = 1; % The Random agents are labeled as infected
end

i1 = randi(N, N-N*percentagesymptoms, 1); % chooses randomly out of the infected who is asymptomatic (WE USE THIS TO ONLY SEND THE SYMPTOMATIC ONES TO QUARANTINE)

for g = 1:length(i1)
```

```
    agents(i1(g)).symptoms = false;
end

totalinfect = zeros(dayz,1); % Creating array to store the infection numbers for each day
totalrecover = totalinfect; % Not sure why this is chosen. Should take into account recovery time.
totalsus = totalinfect; % total suspectable

h = figure(1); %Matlab creates Figure 1 (simulation)
clf;  % Clears current figure

plt = 1;     % set plot flag

for T = 1:dayz % this loop makes the code start at 0 days and continue until the end of the simulation (in this case 180 days)
    daily_infections=0;

    for T2 = 1:timesteps_per_day % this loop makes the dots take 4 steps each day
        set(h, 'Visible', 'off');



        index_of_infected = find([agents.infect]==1); % Returns indice of infected agents

        if isempty(index_of_infected)      % stop when no more infected agents
            break
        end

        if plt
            clf; hold on;
        end

        for k=1:N % This checks every dots position to see if they're in the central hub. If they are, they get sent to a random location
```

```
        if central_hub == true && 3>= agents(k).pos(1) >= 2.5 && 3>= agents(k).pos(2) >= 2.5
            agents(k).pos = rand(1,2)*4;
        end
        % random walk agents
        th = 2*pi*rand; % angle
         % Different loops for the differents speeds
        if different_speeds ==true && rand(1,1) < percentage_fast
            r = 0.8*rand; % distance
              agents(k).pos = [agents(k).pos] + r*[cos(th) sin(th)];
        end

        if different_speeds ==true && rand(1,1) < percentage_slow
            r = 0.075*rand;
            agents(k).pos = [agents(k).pos] + r*[cos(th) sin(th)];
        end

         if different_speeds ==false
            r = 0.075*rand;
            agents(k).pos = [agents(k).pos] + r*[cos(th) sin(th)];
        end

         % to keep the dots in the box
        if agents(k).pos(1) > 4
            agents(k).pos(1) = 5 - agents(k).pos(1); % If an agent has a position > 1 (outside box) the
position is changed to 2 - the position
        elseif agents(k).pos(1) < 0
            agents(k).pos(1) = abs(agents(k).pos(1)); % if pos<0 absolute value is taken

        end

        % Same for other coordinate (remember that the position of the dots has two coordinates x,y)
        if agents(k).pos(2) > 4
            agents(k).pos(2) = 5 - agents(k).pos(2);
```

```
            elseif agents(k).pos(2) < 0
                agents(k).pos(2) = abs(agents(k).pos(2));
            end

            % Send random dots (within a probability) to the central hub

            if central_hub == true && rand(1,1) < probability_central_hub
                agents(k).pos = 2.5 + rand(1,2)*0.5;
            end


            % social distancing
            if social_distancing ==true && rand(1,1) < percentage_social_distancing
                for j = 1:N           % iterating over all the 'i+1' particle which is interacting with the
other
                 d = distance(agents(k), agents(j));  % calculating the diameter of 2 particles and then
calculating the distance of separation between them
                    if d < social_distance
                        Mx = (agents(k).pos(1) + agents(j).pos(1))/2;
                        My = (agents(k).pos(2) + agents(j).pos(2))/2;
                        M = [Mx, My];
                        r = d/2;
                        th = atan((agents(k).pos(2)-agents(j).pos(2)/(agents(k).pos(1)-agents(j).pos(1))));
                        agents(k).pos = [M] + r*[cos(th) sin(th)];
                    end

                end
            end
```

```
            if agents(k).infect == 1    % infected agents
             % Loop for people getting sent to quarantine
                 if quarantine == true && T >= agents(k).infectday + isodays && agents(k).symptoms == true
                    agents(k).pos = 4.1+ rand(1,2)*0.9;
                 end
                % recover. they are sent somewhere random in the box
                if T - agents(k).infectday > ... %T is current day infect day, is the day they got infected
                        recoverytimemean + recoverytimestd*randn
                    agents(k).infect = 2;
                    agents(k).pos = rand(1,2)*4;
                end

            end

            if agents(k).infect == 0    % susceptible agents

                % infected by neighbors
                for j = 1:length(index_of_infected) %ii = array with indices of infected agents

                    if distance(agents(k), agents(j)) < infectradius ... %Norm returns the distance between
the two agents
                            && rand < f(distance(agents(k), agents(j)))  % random value is below the infection
chance
                        agents(k).infect = 1;
                        daily_infections = daily_infections + 1;
                        agents(k).infectday = T; % Setting infect day to the current day
                        break
                    end
                end

            end

            % plot positions
```

```
        if plt %Adding dot in correct color to plot
            if agents(k).infect == 1 && quarantine == true && T >= agents(k).infectday + isodays &&
agents(k).symptoms == true %this is so when dots are in quarantine they dont have the circle around them

                   plot(agents(k).pos(1),agents(k).pos(2),'r.','MarkerFaceColor','r');
                    %infected in quarantine

            elseif  agents(k).infect == 1 %&& quarantine == false || agents(k).infect == 1 && quarantine
== true && T < agents(k).infectday + isodays

                plot(agents(k).pos(1),agents(k).pos(2),'r.','MarkerFaceColor','r');
                         %infected
                theta = 0 : 0.01 : 2*pi;
                radius=infectradius;
                xCenter = agents(k).pos(1);
                yCenter =agents(k).pos(2);
                 thisX = radius * cos(theta) + xCenter;
                thisY = radius * sin(theta) + yCenter;

                   % Plot circles around the center
                plot(thisX, thisY, 'r-', 'LineWidth', 0.1);

            elseif agents(k).infect == 0
                plot(agents(k).pos(1),agents(k).pos(2),'b.','MarkerFaceColor','b');
                % susceptible


            elseif agents(k).infect == 2

                plot(agents(k).pos(1),agents(k).pos(2),'g.','MarkerFaceColor','g');
                % recovered
         end
       end
```

```
end




axis square; axis([0 5 0 5]); % define axis limit


% box around community

x=[0 4 4 0 0];
y=[0 0 4 4 0];

plot(x,y,'k', 'Linewidth', 2);
axis square; axis([0 5 0 5]); % define axis limit


% box around quarantine zone

x=[4 5 5 4 4];
y=[4 4 5 5 4];

plot(x,y,'g', 'Linewidth', 2.5);
axis square; axis([0 5 0 5]); % define axis limit
text(4.0,5.2,'Quarantine Zone');

% box around central hub

x=[2.5 3 3 2.5 2.5];
y=[2.5 2.5 3 3 2.5];
```

```
plot(x,y,'k', 'Linewidth', 1);
axis square; axis([0 5 0 5]); % define axis lim

%legend for agents
x=[0.2 1.6 1.6 0.2 0.2];
y=[4.1 4.1 4.9 4.9 4.1];
plot(x,y,'k', 'Linewidth', 0.5);
axis square; axis([0 5 0 5]); % define axis limit


text(0.7,4.7,'Susceptible');
plot(0.5,4.7, 'b.');

text(0.7,4.5, 'Infected');
plot(0.5,4.5, 'r.');

text(0.7, 4.3, 'Removed');
plot(0.5, 4.3, 'g.');
%legend for R_0

if T2 == 1
     R_0 = (daily_infections/ recoverytimemean);
end

legend({'R_0 =  '+ string(R_0)}, 'location', 'southeast');


if plt
    set(h, 'Visible', 'on');
    drawnow % Updates the graph immediately
end
```

```
    end
    % Updating numbers
    %tf = isinteger(int8(T/2));
    %if tf ==
            totalsus(T) = sum([agents.infect]==0);
        totalinfect(T) = sum([agents.infect]==1);
        totalrecover(T) = sum([agents.infect]==2);
    %end
    disp(T)

end

%% Plot SIR totals (graph)

% trim
totalinfect = totalinfect(1:T-1);
totalrecover = totalrecover(1:T-1);
totalsus = totalsus(1:T-1); %ceil(T/2)

figure(2); clf;
h = area([totalinfect, totalrecover, totalsus]);
xlabel('Days Since Patient 0','fontsize',14);
ylabel('Population','fontsize',14)

h(1).FaceColor = 'r';
h(2).FaceColor = 'g';
h(3).FaceColor = 'b';

legend('Infected','Removed','Susceptible','location','northwest');
axis tight;
grid on;
set(gca,'fontsize',12);
```

```
return
```