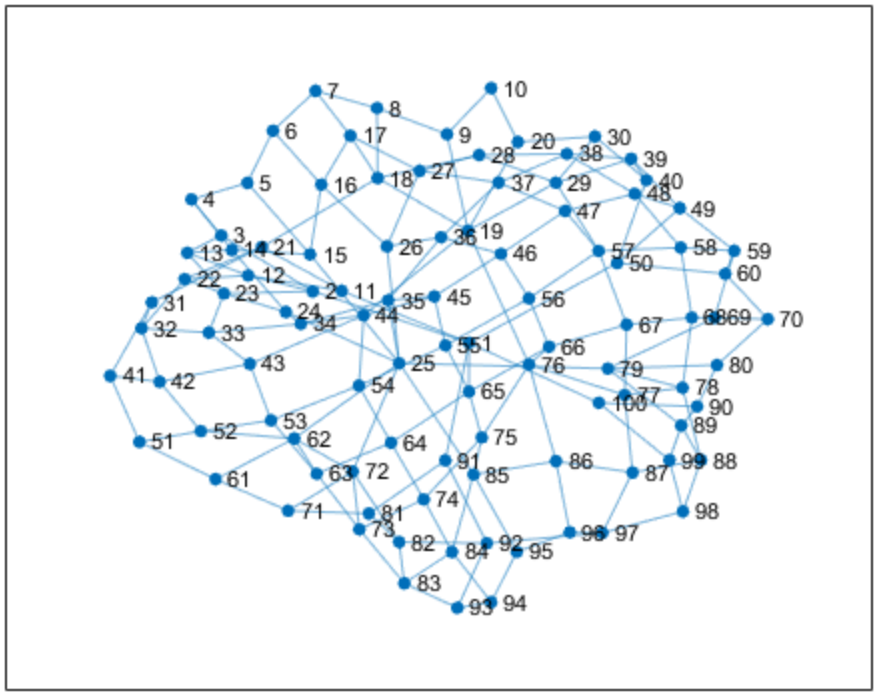
base script for virality example

Table of Contents

create 2-D 10x10 lattice network

```
n = 12;
A = delsq(numgrid('S',n));
G = graph(A,'omitselfloops');

% implement small-world topography, add some edges to G
% there is really good transportation, so many populations are
 connected.
G = addedge(G, 25, 79, 1);
G = addedge(G, 25, 62, 1);
G = addedge(G, 18, 21, 1);
G = addedge(G, 2, 23, 1);
G = addedge(G, 92, 97, 1);
G = addedge(G, 29, 57, 1);
G = addedge(G, 1, 65, 1);
G = addedge(G, 1, 91, 1);
G = addedge(G, 1, 50, 1);
G = addedge(G, 1, 100, 1);
G = addedge(G, 19, 76, 1);
G = addedge(G, 19, 44, 1);
G = addedge(G, 25, 85, 1);
G = addedge(G, 25, 72, 1);

figure();
plot(G);


% extract the adjacency matrix
ad = adjacency(G);

% establish the dimensions
n_dim = length(ad);
```



now we create a state transition matrix for our linear dynamical model

```
% set a self-decay factor
decayf = 0.9;                                   % originally 0.5

% scale strength of nodal interactions
Aff = ad*0.20;                                  % originally 0.16

% create the final dynamical matrix as sum of self-decay and nodal
% interactions
Aff = Aff + (eye(n_dim).*decayf);
```

simulate the network

```
% set the length of the simulation
Tf = 90;

% initialize an array to store the state of the network at each time
 point
x_out = zeros(n_dim,Tf);

% random positive initial conditions
% x_init = randn(n_dim,1).^2;
```

```
% zero initial condition
x_init = zeros(n_dim,1);

x_out(:,1) = x_init;

control_out = zeros(n_dim,Tf);
```

model some disturbance.

In this example, we will consider two short bursts of excitation applied at t = 25 (to node 25) and 50 (to node 45)

```
w_dist = zeros(n_dim, Tf);
w_dist(25,25) = 15;
w_dist(45,50) = 15;
```

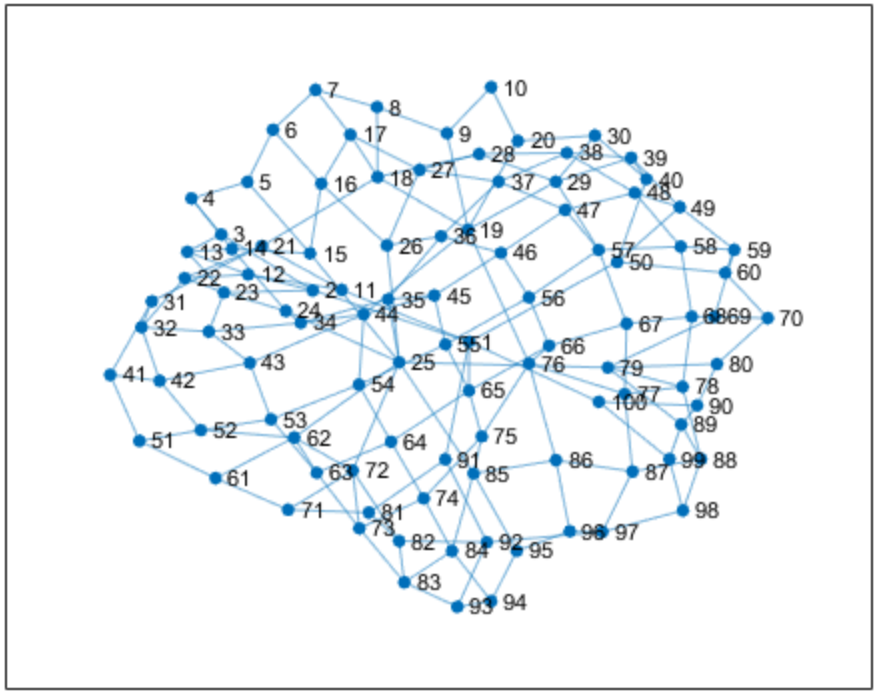
simulate

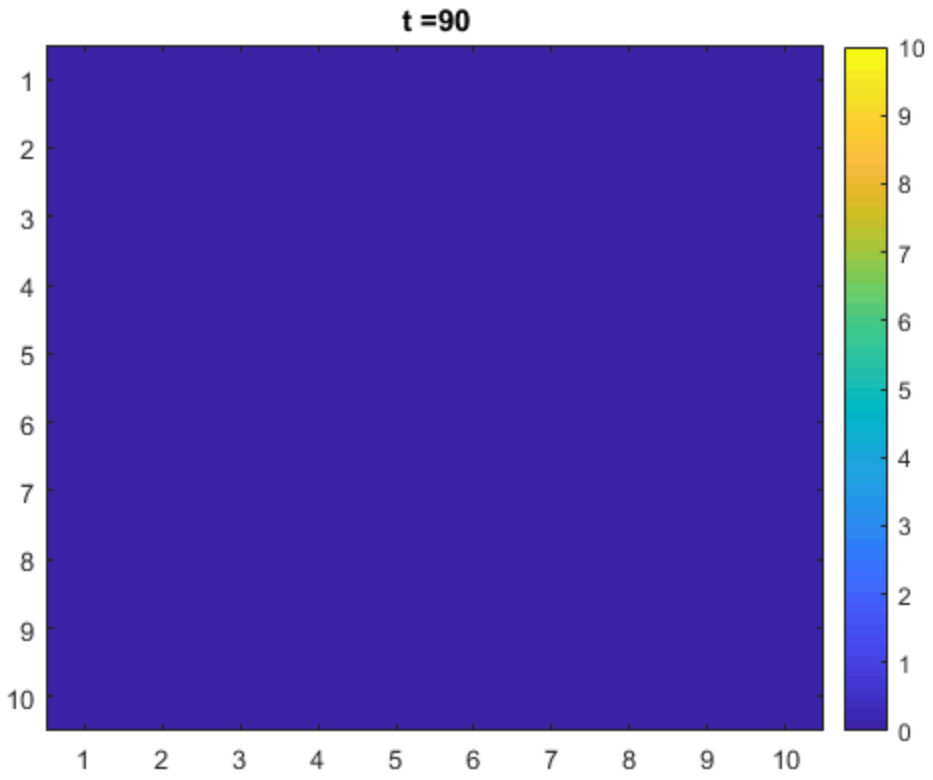
```
for ind = 2:Tf
    % we were originally inputting x_out(:,ind-1)
    % we changed it to x_out(:,1:ind-1)
    control_vector = find_cvec(x_out(:,1:ind-1));

    % without control
    % x_out(:,ind) = Aff*x_out(:,ind-1) + w_dist(:,ind);

    % with control
      x_out(:,ind) = Aff*x_out(:,ind-1) + w_dist(:,ind) +
 control_vector;
      control_out(:,ind) = control_vector;

      % keep x_out positive
      x_out = max(x_out, 0);

end

% anything over 10 is saturated
x_out = min(x_out,10);
```

plot a movie of the network state evolution

```
figure(2);
for ind = 1:Tf

    F = reshape(x_out(:,ind),[10 10]);
    F = min(F,10);
    figure(2);
    imagesc(F,[0 10]);
    title(strcat('t = ',num2str(ind)));
    colorbar;
```

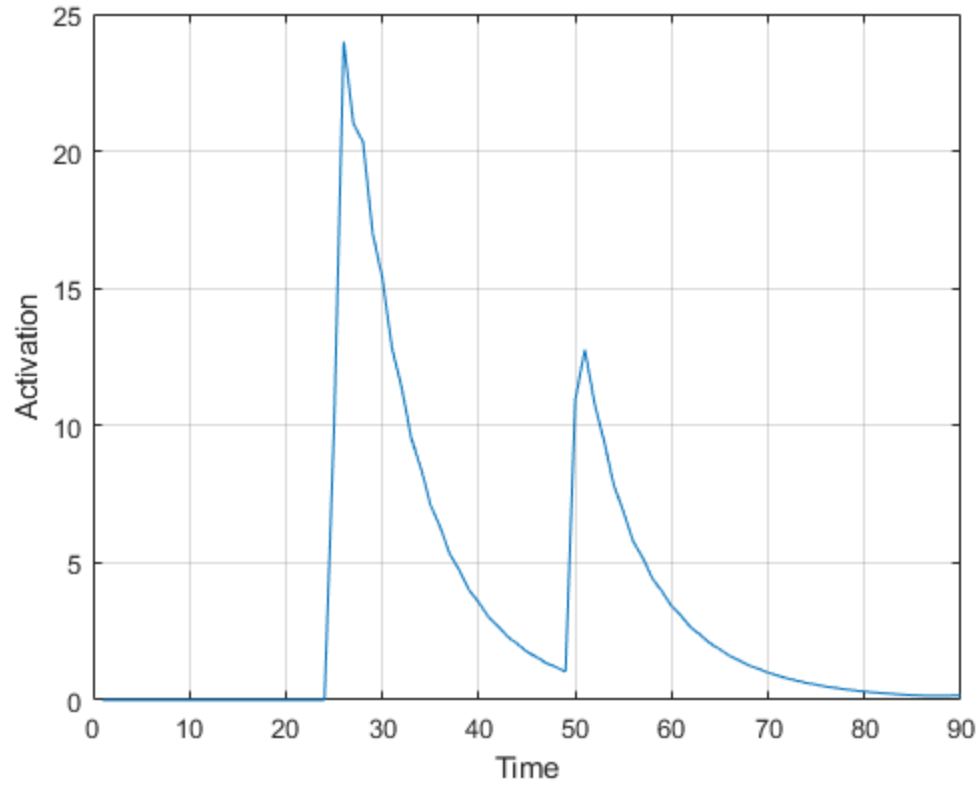
```
    pause(0.1);
end
```

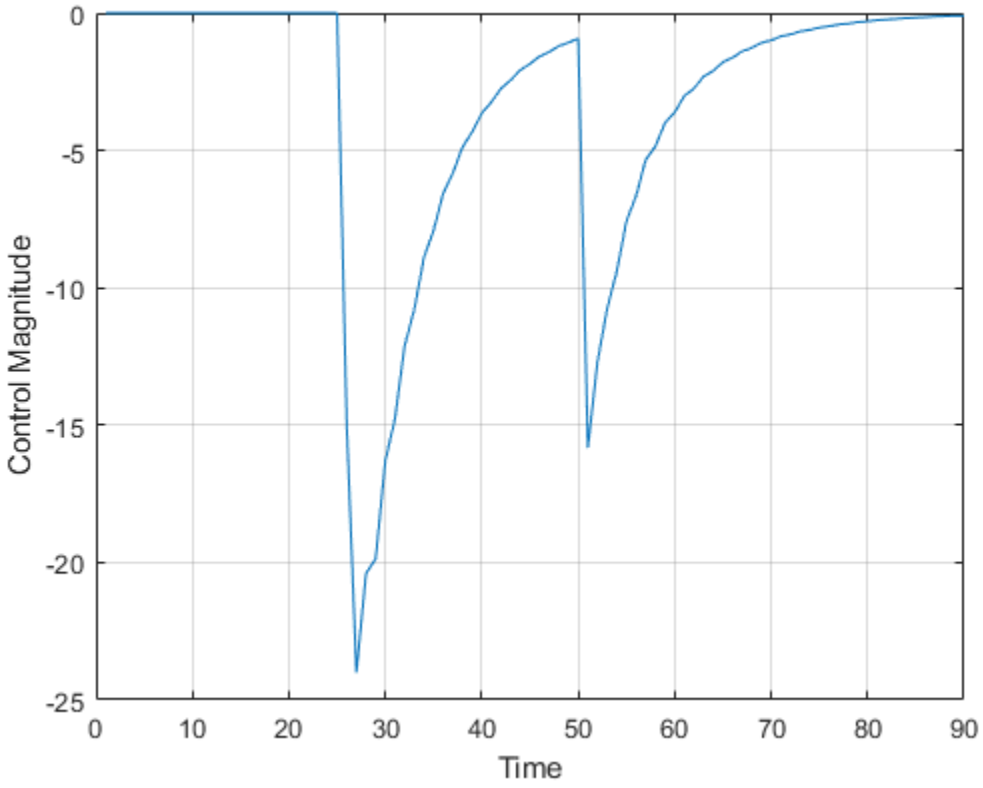

7
8
10
6
17
9
20
30
28
38
39
5
16
18
27
37
29
40
48
4
47
49
3
19
13
14
21
15
26
36
46
57
58
59
50
60
22
12
2
11
35
45
56
31
23
24
44
67
69
70
32
33
34
55
66
41
42
43
25
76
79
80
54
65
77
78
100
90
53
89
51
52
62
64
75
91
86
99
88
63
72
85
87
61
74
71
81
98
73
82
84
92
98
97
95
83
93
94



Plot the total activation of the network as a function of time

```
figure(3);
plot(sum(x_out));
xlabel('Time');
ylabel('Activation');
grid on;

figure(4);
plot(sum(control_out));
xlabel('Time');
ylabel('Control Magnitude');
grid on;
```


Activation
Time
0
5
10
15
20
25
0
10
20
30
40
50
60
70
80
90


Control Magnitude
Time
0
-5
-10
-15
-20
-25
0
10
20
30
40
50
60
70
80
90

prototype for function that returns a "control vector"

```
% parameter x_out is the column data for the current day, along with
 the
% column data for every previous day that has happened so far
 function control_vector = find_cvec(x_out);

    % make control vector 100x1 vector
    control_vector = zeros([length(x_out),1]);

    % check if number of columns is only 1
    cols = size(x_out, 2);
    if(cols == 1)
        % do nothing, leave control_vector as all zeros
    else
        for i = 1:100
            % check difference between each row of the
            % last column with the row in the previous column
            % i.e. each row of "cols" - each row of "cols-1"
            if(x_out(i,cols) - x_out(i,cols-1) > 0.001)
                control_vector(i) = -1*x_out(i, cols);
            end
        end
    end
 end
```

Published with MATLAB® R2019a