REFERENCES

1. White S. H., Martin del Rey A. and Rodriguez Sanchez G. "Modeling epidemics using cellular automata." Applied Mathematics and Computation. 186(March 2007): 193-202.
2. de Vries G., Hillen T., Lewis M., Schõnfisch B. and Muller J. A Course in Mathematical Biology: Quantitative Modeling with Mathematical and Computational. Philadelphia : Society for Industrial and Applied Mathematics, 2006.
3. Fuentes M.A. and Kuperman M.N. "Cellular automata and epidemiological models with spatial dependence." Physica A. 267(May 1999): 471-486.
4. Hu R. and Ruan X. "Differential equation and cellular automata model." Proceedings of IEEE International Conference on Robotics, Intelligent Systems and Signal Processing. IEEE. 2(October 2003): 1047-1051.
5. Kulenovic M.R.S. and Merino O. Discrete Dynamical Systems and Difference Equations with Mathematica. Boca Raton : Chapman and Hall/CRC, 2002.
6. Chokprasit S., Kittivachpokawat K., Nantawannakorn T. , Moore E. J. and Khumsup P. "Teaching modeling of disease transmission using mathematical software." Proceedings of the Thailand International Conference on 21st Century Information Technology in Mathematics Education. ICITME2006. (2006): 214-224.
7. May R.M. "Simple mathematical models with very complicated dynamics." Nature. 261(June 1976): 459-467.
8. Brauer F. "Modeling Influenza: Pandemics and Seasonal Epidemics." In Brauer F., van den Driessche, P. and Wu, J. (Eds.). Mathematical Epidemiology (Lecture Notes in Mathematics). Berlin : Springer-Verlag. (April 2008): 321-344.

9. Awachai P., Jumpen W., Wiwatanapataphee B. and Wu H. Y. "Spreading Dynamics of SEIR Model on Complex Population Network." เอกสารการประชุมวิชาการการนำเสนอผลงานวิจัย ระดับบัณฑิตศึกษา ครั้งที่ 2. กรุงเทพฯ : สำนักบริหารวิชาการ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. (2009) : 134-139.

10. Kermack W.O. and McKendrick A.G. "Contributions to the mathematical theory of epidemics." Proc. Roy. Soc. Lond. 115(August 1927): 700–721.

11. Jones B.D. and Sleeman D.S. Differential Equations and Mathematical Biology. Boca Raton : Chapman & Hall/CRC, 2005.

12. Schneckenreither G., Popper N., Zauner G., Breitenecker F. "Modelling SIR type epidemics by ODEs, PDEs, difference equations and cellular automata – A comparative study." Simulation Modelling Practice and Theory. 16(September 2008): 1014-1023.

13. Thomas G. B. and Finney R. L. Calculus and Analytic Geometry. 8th ed. Reading, Mass. : Addison-Wesley, 1992.

14. Lam C-Y. Applied Numerical Methods for Partial Differential Equations. New York : Prentice-Hall, 1994.

15. Sae-jie W., Bunwong K. and Moore, E.J. "Qualitative Behavior of SIS Epidemic Model on Time Scales". Proceedings of the 4th International Conference on Applied Mathematics, Simulation, Modelling. (2010): 159-164.

16. Sae-jie W., Bunwong K. and Moore, E.J. "The Effect of Time Scales on SIS Epidemic Model". WSEAS Transactions on Mathematics. (2010): 757-767.

APPENDIX A

Finite-Difference Methods

A.1 Forward finite-difference approximation for time derivative

The Taylor series expanded about $u(t)$ is shown in (A-1) - (A-2),

$$u(t+\Delta t) = u(t) + \Delta t \frac{\partial u}{\partial t} + \frac{(\Delta t)^2}{2!} \frac{\partial^2 u}{\partial t^2} + \dots, \qquad \text{(A-1)}$$

$$\frac{\partial u}{\partial t} = \frac{u(t+\Delta t) - u(t)}{\Delta t} + O(\Delta t). \qquad \text{(A-2)}$$

The forward difference of $\partial u / \partial t$ at position indices *i,j* at time index *n* becomes

$$\left(\frac{\partial u}{\partial t}\right)_{i,j}^{n} = \frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\Delta t}. \qquad \text{(A-3)}$$

A.2 Finite-difference approximation for Laplacian for von Neumann neighborhood

For a function $u(x,y)$, we use the Taylor series about x_i at $(x_i + h)$ and $(x_i - h)$. The expansions are the following,

$$u(x+h,y) = u(x,y) + h\frac{\partial u(x,y)}{\partial x} + \frac{h^2}{2!}\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{h^3}{3!}\frac{\partial^3 u(x,y)}{\partial x^3} + \dots, \qquad \text{(A-4)}$$

$$u(x-h,y) = u(x,y) - h\frac{\partial u(x,y)}{\partial x} + \frac{h^2}{2!}\frac{\partial^2 u(x,y)}{\partial x^2} - \frac{h^3}{3!}\frac{\partial^3 u(x,y)}{\partial x^3} + \dots. \qquad \text{(A-5)}$$

where *h* is a grid size, or step size, that is sufficiently small for the second-order approximation to be a good approximation.

Using the subscript notation i,j in *x*-position and *y*-position, we can write equations (A-4) and (A-5) as

$$u_{i+1,j} = u_{i,j} + h\frac{\partial u_{i,j}}{\partial x} + \frac{h^2}{2!}\frac{\partial^2 u_{i,j}}{\partial x^2} + \frac{h^3}{3!}\frac{\partial^3 u_{i,j}}{\partial x^3} + \dots, \qquad \text{(A-6)}$$

$$u_{i-1,j} = u_{i,j} - h\frac{\partial u_{i,j}}{\partial x} + \frac{h^2}{2!}\frac{\partial^2 u_{i,j}}{\partial x^2} - \frac{h^3}{3!}\frac{\partial^3 u_{i,j}}{\partial x^3} + \dots. \qquad \text{(A-7)}$$

If we take (A-6) + (A-7) and rearrange, we get the central difference for the second order derivative with an error $O(h^2)$

$$\frac{\partial^2 u}{\partial x^2} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} \qquad \text{(A-8)}$$

Similarly, for the y-derivative, we obtain

$$\frac{\partial^2 u}{\partial y^2} = \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{k^2} \tag{A-9}$$

Therefore, the approximation of the Laplace equation for a von Neumann neighborhood is the following

$$\begin{aligned}\nabla^2 u(x,y) &= \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \\ &= \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{k^2}\end{aligned} \tag{A-10}$$

A.3 Finite-difference approximation for Laplacian for Moore neighborhood

For a function $u(x,y)$, we use the Taylor series about x_i and y_j at (x_i+h, y_j), (x_i-h, y_j), (x_i, y_j+k), (x_i, y_j-k), (x_i+h, y_j+k), (x_i+h, y_j-k), (x_i-h, y_j+k) and (x_i-h, y_j-k). The approximation for the first four points is already shown in section A.2 for the von Neumann neighborhood. The expansions for the second four points are as follows:

$$\begin{aligned}u(x+h, y+k) &= u(x,y) + h\frac{\partial u(x,y)}{\partial x} + k\frac{\partial u(x,y)}{\partial y} \\ &+ \frac{1}{2!}\left(h^2\frac{\partial^2 u(x,y)}{\partial x^2} + 2hk\frac{\partial^2 u(x,y)}{\partial x \partial y} + k^2\frac{\partial^2 u(x,y)}{\partial y^2}\right) + \ldots,\end{aligned} \tag{A-11}$$

$$\begin{aligned}u(x+h, y-k) &= u(x,y) + h\frac{\partial u(x,y)}{\partial x} - k\frac{\partial u(x,y)}{\partial y} \\ &+ \frac{1}{2!}\left(h^2\frac{\partial^2 u(x,y)}{\partial x^2} - 2hk\frac{\partial^2 u(x,y)}{\partial x \partial y} + k^2\frac{\partial^2 u(x,y)}{\partial y^2}\right) + \ldots,\end{aligned} \tag{A-12}$$

$$\begin{aligned}u(x-h, y+k) &= u(x,y) - h\frac{\partial u(x,y)}{\partial x} + k\frac{\partial u(x,y)}{\partial y} \\ &+ \frac{1}{2!}\left(h^2\frac{\partial^2 u(x,y)}{\partial x^2} - 2hk\frac{\partial^2 u(x,y)}{\partial x \partial y} + k^2\frac{\partial^2 u(x,y)}{\partial y^2}\right) + \ldots,\end{aligned} \tag{A-13}$$

$$\begin{aligned}u(x-h, y-k) &= u(x,y) - h\frac{\partial u(x,y)}{\partial x} - k\frac{\partial u(x,y)}{\partial y} \\ &+ \frac{1}{2!}\left(h^2\frac{\partial^2 u(x,y)}{\partial x^2} + 2hk\frac{\partial^2 u(x,y)}{\partial x \partial y} + k^2\frac{\partial^2 u(x,y)}{\partial y^2}\right) + \ldots.\end{aligned} \tag{A-14}$$

For a square grid, we assume $x = y = \varepsilon$. Using the two subscript notation i, j in *x*-position and *y*-position, we obtain the sum of (A.11) + (A.12) + (A.13) + (A.14) and (A.8) and (A.9), we get the central difference for the second order derivative with an error $O(h^2)$ for the Moore neighborhood as:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{1}{3}\left(\frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} + u_{i+1,j+1} + u_{i+1,j-1} + u_{i-1,j+1} + u_{i-1,j-1} - 8u_{i,j}}{\varepsilon^2}\right)$$

(A-15)

Therefore, the approximation of the Laplace equation with Moore neighborhood is the following

$$\begin{aligned}\nabla u^2 &= \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \\ &= \frac{1}{3}\left(\frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} + u_{i+1,j+1} + u_{i+1,j-1} + u_{i-1,j+1} + u_{i-1,j-1} - 8u_{i,j}}{\varepsilon^2}\right)\end{aligned}$$

(A-16)

APPENDIX B

Computer Program for Simulation of Discretization of SIR-PDE Epidemics Model

```
function [Stot,Itot,Rtot] = SIR_FDE(x,y,epsilon,del_t,I_start,alpha,beta,K,tmax)
% approximate SIR_PDE model by finite-difference
% input:      x         : a length of x
%             y         : a length of y
%          epsilon      : a grid size
%           del_t       : a step of time
%          I_start      : a number of infected
%           alpha       : an infection rate in the same cell
%           beta        : a recovery rate
%             K         : an movement rate
%           tmax        : a max time
% output:   Stot        : total number of susceptible individual
%           Itot        : total number of infected individual
%           Rtot        : total number of recovered individual
% size of matrix
row_a = x/epsilon;
col_b = y/epsilon;
% a start cell
start_cell = [row_a/2,col_b/2];
% a start matrix of S, I, R
Sin = ones(row_a,col_b);
Iin = zeros(row_a,col_b);
Rin = zeros(row_a,col_b);
% total number of population
N = Sin+Iin+Rin;
% number of S and I at a start cell
Iin(start_cell(1),start_cell(2)) = I_start;
Sin(start_cell(1),start_cell(2)) = 1-Iin(start_cell(1),start_cell(2));
% start to compute for S, I, R
S0 = Sin;
I0 = Iin;
R0 = N-Sin-Iin;
```

```
DS0 = round(100*S0)/100;
DI0 = round(100*I0)/100;
DR0 = N-S0-I0;
[nrow,ncol] = size(Sin);
irow = 2:nrow-1;
icol = 2:ncol-1;
[IROW,ICOL] = meshgrid(irow,icol);
Stot = zeros(1,tmax);
Itot = zeros(1,tmax);
Rtot = zeros(1,tmax);
k=1;
for n=1:tmax
   interact_S = infect_S(S0);
   interact_I = infect_I(I0);
   interact_R = infect_R(R0);
  if n==1 || 5*fix(n/5)==n
   figure()
   clf reset
   imshow(I0(2:nrow-1,2:ncol-1),'InitialMagnification','fit')
   splot(k) = getframe;
   k=k+1;
  end
% main model
  S01 = ((1-(4*K*del_t/epsilon^2))*S0)+((K*del_t/epsilon^2)*interact_S)-
(alpha*del_t*S0.*I0);
  I01 = ((1-(4*K*del_t/epsilon^2)-
(beta*del_t))*I0)+((K*del_t/epsilon^2)*interact_I)+(alpha*del_t*S0.*I0);
  R01 = ((1-(4*K*del_t/epsilon^2))
*R0)+((K*del_t/epsilon^2)*interact_R)+((beta*del_t)*I0);
% check for value of S, I, R
  S01 = (S01 >= 0).*S01;
  I01 = (I01 >= 0).*I01;
  S0N = (S01 > N).*N;
  S0o = (S01 <= N).*S01;
  S01 = S0N+S0o;
  I0N = (I01 > N).*N;
  I0o = (I01<=N).*I01;
  I01 = I0N+I0o;
  DS0 = S01;
  DI0 = I01;
  DR0 = N-DS0-DI0;
  Stot(n+1) = sum(sum(DS0(irow,icol)));
  Itot(n+1) = sum(sum(DI0(irow,icol)));
  Rtot(n+1) = sum(sum(DR0(irow,icol)));
  S0=S01;
  I0=I01;
  R0=R01;
end
% plot graph
figure()
movie(splot,1,4);
figure()
plot(2:tmax,Stot(2:tmax),'b-',2:tmax,Itot(2:tmax),'r-',2:tmax,Rtot(2:tmax),'k-');
xlabel('time')
ylabel('population')
legend('Stot','Itot','Rtot',0)
title(sprintf('\n alpha = %0.5g, beta = %0.5g,K= %0.5g',alpha,beta,K));
```

APPENDIX C

Computer Program for Simulation of SIR-CA Epidemics Model

```
function [splot,Stot,Itot,Rtot] =
SIR_CA(row_a,col_b,start_cell,I_start,infec_rate,recover_rate,neighborhood_type,v_
c,v_m,tmax)
% simulate SIR-CA model
% input:      row_a           : number of row
%             col_b           : number of column
%             start_cell      : a start cell
%             I_start         : number of infected at the beginning
%             infec_rate      : an infection rate
%             recover_rate    : a recovery rate
%             neighborhood_type    : a type of the neighborhood,
%                                    1 for von Neumann,
%                                    2 for Moore
%             v_c             : connection factor
%             v_m             : movement factor
%             tmax            : a max time
% output:     splot           : simulation of the model
%             Stot            : total number of susceptible individual in the lattice
%             Itot            : total number of infected individual in the lattice
%             Rtot            : total number of recovered individual in the lattice
% check for the neighborhood
if neighborhood_type == 1
  c = [0 v_c 0;v_c 0 v_c;0 v_c 0];
  m = [0 v_m 0;v_m 0 v_m;0 v_m 0];
else
  c = [v_c v_c v_c;v_c 0 v_c;v_c v_c v_c];
  m = [v_m v_m v_m;v_m 0 v_m;v_m v_m v_m];
end
% define the parameters
alpha = infec_rate;
beta = recover_rate;
mu = c.*m.* alpha;
```

```
% define the lattice
Sin = ones(row_a+2,col_b+2);
Iin = zeros(row_a+2,col_b+2);
Rin = zeros(row_a+2,col_b+2);
Iin(start_cell(1)+1,start_cell(2)+1) = I_start;
Sin(start_cell(1)+1,start_cell(2)+1) = 1-Iin(start_cell(1),start_cell(2));
% Total number of people
N = Sin+Iin+Rin;
% Start state of each individual
S0 = Sin;
I0 = Iin;
R0 = N-Sin-Iin;
DS0 = round(100*S0)/100;
DI0 = round(100*I0)/100;
DR0 = N-S0-I0;
[nrow,ncol] = size(Sin);
irow = 2:nrow-1;
icol = 2:ncol-1;
[IROW,ICOL] = meshgrid(irow,icol);
S1=zeros(1,tmax); I1=zeros(1,tmax); R1=zeros(1,tmax);
 S1(1) = DS0(start_cell(1)+1,start_cell(2)+1);
I1(1) = DI0(start_cell(1)+1,start_cell(2)+1);
R1(1) = DR0(start_cell(1)+1,start_cell(2)+1);
Stot = zeros(1,tmax); Itot = zeros(1,tmax); Rtot = zeros(1,tmax);
% Compute a value of each state
k=1;
for n=1:tmax
   interact = infect_int(mu,I0,S0);
   if n==1 || 5*fix(n/5)==n
   figure()
   clf reset
   imshow(I0(2:nrow-1,2:ncol-1),'InitialMagnification','fit')
```

```
splot(k) = getframe;
k=k+1;
end
S01 = S0-(alpha *S0.*I0)-interact;
I01 = (1-beta)*I0+(alpha*S0.*I0)+interact;
R01 = R0+(beta*I0);
S01 = (S01 >= 0).*S01;
I01 = (I01 >= 0).*I01;
S0N = (S01 > N).*N;
S0o = (S01 <= N).*S01;
S01 = S0N+S0o;
I0N = (I01 > N).*N;
I0o = (I01<=N).*I01;
I01 = I0N+I0o;
DS0 = round(100*S01)/100;
DI0 = round(100*I01)/100;
DR0 = N-(DS0+DI0);
S1(n+1) = DS0(26,26);
I1(n+1) = DI0(26,26);
R1(n+1) = DR0(26,26);
Stot(n+1) = sum(sum(DS0(irow,icol)));
Itot(n+1) = sum(sum(DI0(irow,icol)));
Rtot(n+1) = sum(sum(DR0(irow,icol)));
S0=S01; I0=I01; R0=R01;
S01 = S0-(alpha*S0.*I0)-interact;
I01 = (1-beta)*I0+(alpha*S0.*I0)+interact;
R01 = R0+(beta*I0);
S01 = (S01 >=0).*S01;
I01 = (I01>=0).*I01;
S0N = (S01 > N).*N;
S0o = (S01<=N).*S01;
S01 = S0N+S0o;
```

```
  I0N = (I01 > N).*N;
  I0o = (I01<=N).*I01;
  I01 = I0N+I0o;
  DS0 = round(100*S01)/100;
  DI0 = round(100*I01)/100;
  DR0 = N-DS0-DI0;
  S1(n+1) = DS0(start_cell(1)+1,start_cell(2)+1);
  I1(n+1) = DI0(start_cell(1)+1,start_cell(2)+1);
  R1(n+1) = DR0(start_cell(1)+1,start_cell(2)+1);
  Stot(n+1) = sum(sum(DS0(irow,icol)));
  Itot(n+1) = sum(sum(DI0(irow,icol)));
  Rtot(n+1) = sum(sum(DR0(irow,icol)));
  S0=S01; I0=I01; R0=R01;
end
% Plot graph
figure()
movie(splot,1,4);
figure()
plot(1:tmax,S1,'b-',1:tmax,I1,'r-',1:tmax,R1,'k-');
xlabel('time')
ylabel('population')
legend('S1','I1','R1',0)
title(sprintf('\n Start cell with alpha= %0.5g, epsilon = %0.5g',alpha,beta));
figure()
figure()
plot(2:tmax,Stot(2:tmax),'b-',2:tmax,Itot(2:tmax),'r-',2:tmax,Rtot(2:tmax),'k-');
xlabel('time')
ylabel('population')
legend('S','I','R',0)
title(sprintf('\n alpha= %0.5g, beta = %0.5g',alpha,beta));
```

APPENDIX D

Computer Program for Simulation of SIS-CA Epidemics Model

```
function [splot,S1,I1,S2,I2,Stot,Itot] =
SIS_CA(row_a,col_b,start_cell,I_start,v,epsilon,v_c,v_m,neighborhood_type,nmax)
% simulate SIR-CA model
% input:      row_a       : number of row
%             col_b       : number of column
%             start_cell  : a start cell
%             I_start     : number of infected at the beginning
%             v           : an infection rate
%             epsilon     : a recovery rate
%             v_c         : connection factor
%             v_m         : movement factor
%             nmax        : a max time
% output:     splot       : simulation of the model
%             S1          : total number of susceptible individual in a start cell
%             I1          : total number of infected individual in a start cell
%             S2          : total number of susceptible individual in non-start cell
%             I2          : total number of infected individual in non-start cell
%             Stot        : total number of susceptible individual in the lattice
%             Itot        : total number of infected individual in the lattice
 % Check for the neighborhood
if neighborhood_type == 1
   c = [0 v_c 0;v_c 0 v_c;0 v_c 0]; m = [0 v_m 0;v_m 0 v_m;0 v_m 0];
else
   c = [v_c v_c v_c;v_c 0 v_c;v_c v_c v_c];
   m = [v_m v_m v_m;v_m 0 v_m;v_m v_m v_m];
end
 % Define the parameters
Sin = ones(row_a+2,col_b+2); Iin = zeros(row_a+2,col_b+2);
mu = c.*m.*v;
Iin(start_cell(1),start_cell(2)) = I_start;
Sin(start_cell(1),start_cell(2)) = 1-Iin(start_cell(1),start_cell(2));
% Total number of people
```

```
N=Sin+Iin;
% Start state of each individual
I0 = Iin; S0 = N-Iin; DI0 = round(100*I0)/100; DS0  = N-I0;
[nrow,ncol] = size(Sin);
irow = 2:nrow-1; icol = 2:ncol-1;
[IROW,ICOL] = meshgrid(irow,icol);
S1 = zeros(1,nmax); I1 = zeros(1,nmax); S2 = zeros(1,nmax); I2 = zeros(1,nmax);
S1(1) = DS0(start_cell(1),start_cell(2)); I1(1) = DI0(start_cell(1),start_cell(2));
S2(1) = DS0(start_cell(1)+1,start_cell(2)+1);
I2(1) = DI0(start_cell(1)+1,start_cell(2)+1);
Stot = zeros(1,nmax); Itot = zeros(1,nmax);
% Compute a value of each state
k=1;
for n=1:nmax-1
   interact = infect_int(mu,I0,S0);
  if n==1 || 5*fix(n/5)==n
   figure()
   clf reset
   imshow(I0(2:nrow-1,2:ncol-1),'InitialMagnification','fit')
   splot(k) = getframe; k=k+1;
  end
  I01 = (1-epsilon)*I0+(v*S0.*I0)+interact;
  S01 = N-I01; S01 = (S01 >=0).*S01; I01 = (I01>=0).*I01;
  S0N = (S01 > N).*N; S0o = (S01<=N).*S01;
  S01 = S0N+S0o; I0N = (I01 > N).*N;
  I0o = (I01<=N).*I01; I01 = I0N+I0o;
  DI0 = round(100*I01)/100; DS0 = N-DI0;
  S1(n+1) = DS0(26,26); I1(n+1) = DI0(26,26);
  S2(n+1) = DS0(27,27); I2(n+1) = DI0(27,27);
  Stot(n+1) = sum(sum(DS0(irow,icol))); Itot(n+1) = sum(sum(DI0(irow,icol)));
  S0=S01; I0=I01;
end
```

```
% Plot graph
figure()
movie(splot,2,4);
figure()
plot(1:nmax,S1,'b-',1:nmax,I1,'r-');
xlabel('time')
ylabel('population')
legend('S','I',0)
title(sprintf('\n Start cell with alpha= %0.5g, beta= %0.5g',v,epsilon));
figure()
plot(1:nmax,S2,'b-',1:nmax,I2,'r-');
xlabel('time')
ylabel('population')
legend('S','I',0)
title(sprintf('\n Non-start cell with alpha= %0.5g, beta= %0.5g',v,epsilon));
if neighborhood_type == 1
figure()
plot(2:nmax,Stot(2:nmax),'b-',2:nmax,Itot(2:nmax),'r-');
xlabel('time')
ylabel('population')
legend('S','I',0)
title(sprintf('\n von Neumann neighborhood with alpha= %0.5g, beta=
%0.5g',v,epsilon));
else
   figure()
   plot(2:nmax,Stot(2:nmax),'b-',2:nmax,Itot(2:nmax),'r-');
   xlabel('time')
   ylabel('population')
   legend('S','I',0)
   title(sprintf('\n Moore neighborhood with alpha= %0.5g, beta= %0.5g',v,epsilon));
end
```

BIOGRAPHY

Name	:	Miss Antiga Prinyanilakul
Thesis Title	:	Modeling Spatial Spread of Epidemics Using Cellular Automata
Major Field	:	Applied Mathematics

Biography

Date of Birth	:	27 October 1985
Education	:	2004-2007 B.Sc. in Applied Mathematics, King Mongkut's University of Technology North Bangkok
		2008-2010 M.Sc. in Applied Mathematics, King Mongkut's University of Technology North Bangkok
Address	:	1669/961 Pinjaroen 2, Songprapa Road, Seekun, Donmuang, Bangkok, Thailand 10210

Publication :

[1] A. Prinyanilakul, E.J.Moore and U. Phalavonk, *Modeling spatial spread of epidemics using cellular automata*, Proceedings of the 15th Annual Meeting in Mathematics (AMM2010), 2010, Bangkok: KMUTNB, pp. 9-20.