元胞自动机与 MATLAB

引言

元胞自动机（CA）是一种用来仿真局部规则和局部联系的方法。典型的元胞自动机是定义在网格上的，每一个点上的网格代表一个元胞与一种有限的状态。变化规则适用于每一个元胞并且同时进行。典型的变化规则，决定于元胞的状态，以及其（ 4 或 8 ）邻居的状态。元胞自动机已被应用于物理模拟，生物模拟等领域。本文就一些有趣的规则，考虑如何编写有效的 MATLAB 的程序来实现这些元胞自动机。

MATLAB 的编程考虑

元胞自动机需要考虑到下列因素，下面分别说明如何用 MATLAB 实现这些部分。并以 Conway 的生命游戏机的程序为例，说明怎样实现一个元胞自动机。

- 矩阵和图像可以相互转化，所以矩阵的显示是可以真接实现的。如果矩阵 cells 的所有元素只包含两种状态且矩阵 Z 含有零，那么用 image 函数来显示 cat 命令建的 RGB 图像，并且能够返回句柄。

```
imh = image(cat(3,cells,z,z));
set(imh, 'erasemode', 'none')
axis equal
axis tight
```

- 矩阵和图像可以相互转化，所以初始条件可以是矩阵，也可以是图形。以下代码生成一个零矩阵，初始化元胞状态为零，然后使得中心十字形的元胞状态= 1。

```
z = zeros(n,n);
cells = z;
cells(n/2,.25*n:.75*n) = 1;
cells(.25*n:.75*n,n/2) = 1;
```

- Matlab 的代码应尽量简洁以减小运算量。以下程序计算了最近邻居总和，并按照 CA 规则进行了计算。本段 Matlab 代码非常灵活的表示了相邻邻居。

```
x = 2:n-1;
y = 2:n-1;
sum(x,y) = cells(x,y-1) + cells(x,y+1) + ...
                cells(x-1, y) + cells(x+1,y) + ...
                cells(x-1,y-1) + cells(x-1,y+1) + ...
                cells(x+1,y-1) + cells(x+1,y+1);
cells = (sum==3) | (sum==2 & cells);
```

- 加入一个简单的图形用户界面是很容易的。在下面这个例子中，应用了三个按钮和一个文本框。三个按钮，作用分别是运行，停止，程序退出按钮。文框是用来显示的仿真运算的次数。

```
%build the GUI
%define the plot button
plotbutton=uicontrol('style','pushbutton',...
        'string','Run', ...
        'fontsize',12, ...
        'position',[100,400,50,20], ...
        'callback', 'run=1;');

%define the stop button
erasebutton=uicontrol('style','pushbutton',...
        'string','Stop', ...
        'fontsize',12, ...
        'position',[200,400,50,20], ...
        'callback','freeze=1;');

%define the Quit button
quitbutton=uicontrol('style','pushbutton',...
        'string','Quit', ...
        'fontsize',12, ...
        'position',[300,400,50,20], ...
        'callback','stop=1;close;');

number = uicontrol('style','text', ...
        'string','1', ...
        'fontsize',12, ...
        'position',[20,400,50,20]);
```

经过对控件（和 CA）初始化，程序进入一个循环，该循环测试由回调函数的每个按钮控制的变量。刚开始运行时，只在嵌套的 while 循环和 if 语句中运行。直到退出按钮按下时，循环停止。另外两个按钮按下时执行相应的 if 语句。

```
stop= 0; %wait for a quit button push
run = 0; %wait for a draw
freeze = 0; %wait for a freeze
while (stop==0)
         if (run==1)
             %nearest neighbor sum
             sum(x,y) = cells(x,y-1) + cells(x,y+1) + ...
                         cells(x-1, y) + cells(x+1,y) + ...
                         cells(x-1,y-1) + cells(x-1,y+1) + ...
                         cells(3:n,y-1) + cells(x+1,y+1);
             % The CA rule
             cells = (sum==3) | (sum==2 & cells);
             %draw the new image
             set(imh, 'cdata', cat(3,cells,z,z) )
             %update the step number diaplay
             stepnumber = 1 + str2num(get(number,'string'));
             set(number,'string',num2str(stepnumber))
        end
        if (freeze==1)
            run = 0;
            freeze = 0;
end
        drawnow    %need this in the loop for controls to work
end
```

例子

1 .Conway 的生命游戏机。

规则是：

- 对周围的 8 个近邻的元胞状态求和
- 如果总和为 2 的话，则下一时刻的状态不改变
- 如果总和为 3 ，则下一时刻的状态为 1
- 否则状态= 0

核心代码：

```
x = 2:n-1;
y = 2:n-1;
   %nearest neighbor sum
sum(x,y) = cells(x,y-1) + cells(x,y+1) + ...
                cells(x-1, y) + cells(x+1,y) + ...
                cells(x-1,y-1) + cells(x-1,y+1) + ...
                cells(3:n,y-1) + cells(x+1,y+1);
% The CA rule
cells = (sum==3) | (sum==2 & cells);
```

2 .表面张力

规则是：

- 对周围的 8 近邻的元胞以及自身的状态求和
- 如果总和< 4 或= 5 ，下一时刻的状态= 0
- 否则状态= 1

核心代码：

```
x = 2:n-1;
y = 2:n-1;
%nearest neighbor sum
   sum(x,y) = cells(x,y-1) + cells(x,y+1) + ...
                cells(x-1, y) + cells(x+1,y) + ...
                cells(x-1,y-1) + cells(x-1,y+1) + ...
                cells(3:n,y-1) + cells(x+1,y+1)+...
                cells(x,y);
            % The CA rule
            cells = ~((sum< 4) | (sum==5));
```

3 .渗流集群

规则：

- 对周围相邻的 8 邻居求和（元胞只有两种状态，0 或 1 ）。元胞也有一个单独的状态参量（所谓'记录' ）记录它们之前是否有非零状态的邻居。
- 在 0 与 1 之间产生一个随机数 r 。
- 如果总和> 0 （至少一个邻居）并且 r >阈值，或者元胞从未有过一个邻居，则元胞= 1 。
- 如果总和> 0 则设置"记录"的标志，记录这些元胞有一个非零的邻居。

核心代码：

```
sum(2:a-1,2:b-1) = cells(2:a-1,1:b-2) + cells(2:a-1,3:b) + ...
                        cells(1:a-2, 2:b-1) + cells(3:a,2:b-1) + ...
                        cells(1:a-2,1:b-2) + cells(1:a-2,3:b) + ...
```

```
                cells(3:a,1:b-2) + cells(3:a,3:b);

    pick = rand(a,b);
    cells = cells   | ((sum>=1) & (pick>=threshold) & (visit==0))   ;
    visit = (sum>=1) ;
```

变量 a 和 b 是图像的尺寸。最初的图形是由图形操作决定的。以下程序设定坐标系为一个固定的尺寸，在坐标系里写入文本，然后获得并返回坐标内的内容，并用 getframe 函数把它们写入一个矩阵

```
ax = axes('units','pixels','position',[1 1 500 400],'color','k');
text('units', 'pixels', 'position', [130,255,0],...
    'string','MCM','color','w','fontname','helvetica','fontsize',100)
text('units', 'pixels', 'position', [10,120,0],...
'string','Cellular Automata','color','w','fontname','helvetica','fontsize',50)
initial = getframe(gca);

[a,b,c]=size(initial.cdata);
z=zeros(a,b);
cells = double(initial.cdata(:,:,1)==255);
visit = z ;
sum = z;
```

经过几十个时间间隔（从 MCM Cellular Automata 这个图像开始），我们可以得到以下的图像。

4.激发介质（ BZ reaction or heart）

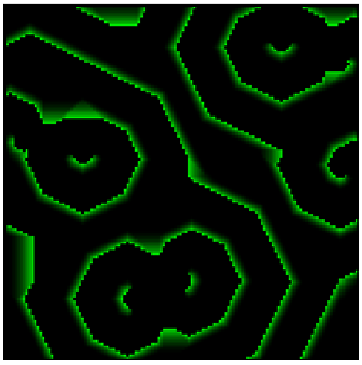
规则：

- 元胞有 10 个不同的状态。状态 0 是体眠。1-5 为活跃状态，、6-9 为是极活跃状态。
- 计算每一个处于活跃的状态的元胞近邻的 8 个元胞。
- 如果和大于或等于 3 （至少有三个活跃的邻居），则下一时刻该元胞= 1 。
- 不需要其它输入，1 至 9 种状态依次出现。如果该时刻状态= 1 那么下一时刻状态= 2 。如果该时刻状态= 2 ，然后下一时刻状态= 3 ，对于其它的状态依次类推，直到第 9 种状态。如果状态= 9 ，然后下一状态= 0 并且元胞回到休息状态。

核心代码：

```
x = [2:n-1];
y = [2:n-1];

 sum(x,y) = ((cells(x,y-1)> 0)&(cells(x,y-1)< t)) + ((cells(x,y+1)> 0)&(cells(x,y+1)< t)) + ...
    ((cells(x-1, y)> 0)&(cells(x-1, y)< t)) + ((cells(x+1,y)> 0)&(cells(x+1,y)< t)) + ...
    ((cells(x-1,y-1)> 0)&(cells(x-1,y-1)< t)) + ((cells(x-1,y+1)> 0)&(cells(x-1,y+1)< t)) + ...
    ((cells(x+1,y-1)> 0)&(cells(x+1,y-1)< t)) + ((cells(x+1,y+1)> 0)&(cells(x+1,y+1)< t));

    cells = ((cells==0) & (sum>=t1)) + ...
            2*(cells==1) + ...
            3*(cells==2) + ...
            4*(cells==3) + ...
            5*(cells==4) + ...
            6*(cells==5) +...
            7*(cells==6) +...
            8*(cells==7) +...
            9*(cells==8) +...
            0*(cells==9);
```

一个 CA 初始图形经过螺旋的变化，得到下图。



5 .森林火灾

规则：

- 元胞有 3 个不同的状态。状态为 0 是空位，状态= 1 是燃烧着的树木，状态 = 2 是树木。
- 如果 4 个邻居中有一个或一个以上的是燃烧着的并且自身是树木（状态为 2 ），那么该元胞下一时刻的状态是燃烧（状态为 1 ）。
- 森林元胞（状态为 2 ）以一个低概率（例如 0.000005 ）开始烧（因为闪电）。
- 一个燃烧着的元胞（状态为 1 ）在下一时时刻变成空位的（状态为 0 ）。
- 空元胞以一个低概率（例如 0.01 ）变为森林以模拟生长。
- 出于矩阵边界连接的考虑，如果左边界开始着火，火势将向右蔓延，右边界同理。同样适用于顶部和底部。

核心代码：

```
sum = (veg(1:n,[n 1:n-1])==1) + (veg(1:n,[2:n 1])==1) + ...
        (veg([n 1:n-1], 1:n)==1) + (veg([2:n 1],1:n)==1) ;

veg = ...
      2*(veg==2) - ((veg==2) & (sum> 0 | (rand(n,n)< Plightning))) + ...
      2*((veg==0) & rand(n,n)< Pgrowth) ;
```

注意环形连接是由序标实现的。

6 .气体动力学

这个 CA （以及接下来的两个 CA）是用来模拟粒子运动的。此元胞自动机需要一种不同类型的元胞的邻居。此元胞的邻居时刻变化，因此某一个方向运动趋势，将继续在同一个方向。换言之，此规则保存势头，这是基础的动力仿真。这种邻居通常被称为 margolis 邻居并且这种邻居通常由重叠的 2x2 块的元胞构成。在下面的表格中，偶数步长时左上方 4 元胞为邻居关系，奇数步长时右下的 4 元胞为邻居关系。某一特定元胞在每一个时间步长都有 3 个邻居，但是具体的元胞构成了邻居的旋转和反复。

偶	偶	
偶	元胞	奇
	奇	奇

规则：

- 此规则叫作 HPP-气体规则。
- 每个元胞有 2 种状态。状态= 0 是空的，状态= 1 代表粒子。
- 在任何一个时间步长，假设粒子是刚刚进入 2x2 的网格块。它将通过其网格块的中心到达对角的网格中，所以在任何时间步长，每一个元胞与该元胞对角对元胞交换的内容。如下所示，左边显示出来的元胞结构经过一个时间步长变为右边的结构。以下是六种不同的情况，所有所有的元胞都遵循相同的转动规则。下文还将考虑两种特殊情况，即粒子-粒子碰撞和粒子-墙碰撞。

0	0	⇒	0	0
0	0		0	0

1	0	⇒	0	0
0	0		0	1

1	0	⇒	1	0
0	1		0	1

1	0	⇒	0	1
1	0		0	1

1	1	⇒	0	1
1	0		1	1

1	1	⇒	1	1
1	1		1	1

- 为了实现粒子碰撞过程（保证动量和能量守恒），对于两个处于对角线上的粒子，他们相互撞击后偏转 90 度。在一个时间步长里使其从一个对角转成另一个对角。你可以逆时针旋转这四个元胞来实现这个过程。则第三规则可以表示为：

1	0	⇒	0	1
0	1		1	0

- 粒子撞击墙壁时，简单地使其离开且状态不变。这就引起反射现象。

核心代码：

```
p=mod(i,2); %margolis neighborhood, where i is the time step
        %upper left cell update
        xind = [1+p:2:nx-2+p];
        yind = [1+p:2:ny-2+p];

        %See if exactly one diagonal is ones
        %only (at most) one of the following can be true!
        diag1(xind,yind) = (sand(xind,yind)==1) & (sand(xind+1,yind+1)==1) & ...
                (sand(xind+1,yind)==0) & (sand(xind,yind+1)==0);

        diag2(xind,yind) = (sand(xind+1,yind)==1) & (sand(xind,yind+1)==1) & ...
                (sand(xind,yind)==0) & (sand(xind+1,yind+1)==0);

        %The diagonals both not occupied by two particles
        and12(xind,yind) = (diag1(xind,yind)==0) & (diag2(xind,yind)==0);

        %One diagonal is occupied by two particles
        or12(xind,yind)    = diag1(xind,yind) | diag2(xind,yind);

        %for every gas particle see if it near the boundary
        sums(xind,yind) = gnd(xind,yind) | gnd(xind+1,yind) | ...
                                        gnd(xind,yind+1) | gnd(xind+1,yind+1) ;

        % cell layout:
        % x,y        x+1,y
        % x,y+1    x+1,y+1
        %If (no walls) and (diagonals are both not occupied)
        %then there is no collision, so move opposite cell to current cell
        %If (no walls) and (only one diagonal is occupied)
        %then there is a collision so move ccw cell to the current cell
        %If (a wall)
        %then don't change the cell (causes a reflection)
        sandNew(xind,yind) = ...
                (and12(xind,yind)    & ~sums(xind,yind) & sand(xind+1,yind+1)) + ...
                (or12(xind,yind) & ~sums(xind,yind) & sand(xind,yind+1)) + ...
                (sums(xind,yind) & sand(xind,yind));

        sandNew(xind+1,yind) = ...
                (and12(xind,yind)    & ~sums(xind,yind) & sand(xind,yind+1)) + ...
                (or12(xind,yind) & ~sums(xind,yind) & sand(xind,yind))+ ...
                (sums(xind,yind) & sand(xind+1,yind));

        sandNew(xind,yind+1) = ...
                (and12(xind,yind)    & ~sums(xind,yind) & sand(xind+1,yind)) + ...
                (or12(xind,yind) & ~sums(xind,yind) & sand(xind+1,yind+1))+ ...
                (sums(xind,yind) & sand(xind,yind+1));

         sandNew(xind+1,yind+1) = ...
                (and12(xind,yind)    & ~sums(xind,yind) & sand(xind,yind)) + ...
                (or12(xind,yind) & ~sums(xind,yind) & sand(xind+1,yind))+ ...
                (sums(xind,yind) & sand(xind+1,yind+1));

sand = sandNew;
```

8.扩散限制聚集

这个系统是模拟粘性颗粒的聚集，形成分形结构。质点以一个类似于例 6 中的 HPP-气体规则发生运动 。不同的是粒子在一些高密度（但看不见）的液体周围被假定是弹跳的。效果是每一个粒子在每个时间步长在随机的方向上运动。换言之，每一个时间步长是一个碰撞的过程。这个仿真矩阵的中心确定了在一个固定生长颗粒。任何弥散粒子触及它就会被它粘住，并成为一个不能移动的，有粘性颗粒。

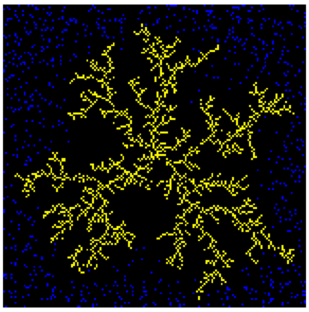
规则：

- 使用 Margolus 型邻居。在每一个时间步，等概率地顺时针或逆时针旋转 4 个元胞。旋转使速度随机化。
- 在移动后，如果八个最近的邻居有一个或一个以上元胞是固定的粘性颗粒，则下时刻该元胞将被冻结，并且使之有粘性。

核心代码：

```
p=mod(i,2); %margolis neighborhood

    %upper left cell update
    xind = [1+p:2:nx-2+p];
    yind = [1+p:2:ny-2+p];
    %random velocity choice
    vary = rand(nx,ny)< .5 ;
    vary1 = 1-vary;

    %diffusion rule -- margolus neighborhood
    %rotate the 4 cells to randomize velocity
    sandNew(xind,yind) = ...
        vary(xind,yind).*sand(xind+1,yind) + ... %cw
        vary1(xind,yind).*sand(xind,yind+1) ;    %ccw

    sandNew(xind+1,yind) = ...
        vary(xind,yind).*sand(xind+1,yind+1) + ...
        vary1(xind,yind).*sand(xind,yind) ;

    sandNew(xind,yind+1) = ...
        vary(xind,yind).*sand(xind,yind) + ...
        vary1(xind,yind).*sand(xind+1,yind+1) ;

     sandNew(xind+1,yind+1) = ...
        vary(xind,yind).*sand(xind,yind+1) + ...
        vary1(xind,yind).*sand(xind+1,yind) ;

    sand = sandNew;

    %for every sand grain see if it near the fixed, sticky cluster
    sum(2:nx-1,2:ny-1) = gnd(2:nx-1,1:ny-2) + gnd(2:nx-1,3:ny) + ...
                         gnd(1:nx-2, 2:ny-1) + gnd(3:nx,2:ny-1) + ...
                         gnd(1:nx-2,1:ny-2) + gnd(1:nx-2,3:ny) + ...
                         gnd(3:nx,1:ny-2) + gnd(3:nx,3:ny);

    %add to the cluster
    gnd = ((sum> 0) & (sand==1)) | gnd ;
    %and eliminate the moving particle
    sand(find(gnd==1)) = 0;
```

以下经过很多时间步长后固定集聚后的图像显示。



9 .砂堆规则

一堆沙子的横截面，可以使用 Margolus 型邻居仿真，但运动规则不同。

规则：

- 元胞有 2 个状态。状态= 0 是空的，状态= 1 代表沙子。
- 在任何时间步长，一个粒子，可以在 2x2 块中向着底部运动。可能运动如下所示。墙壁和底部将阻止粒子继续运动。
- 为了让该运动略有随机性，我亦补充说一项规则，有时处于下落状态的两个元胞还旋转，直到所有的动作都完成。

0	0	⇒	0	0
0	0		0	0

1	0	⇒	0	0
0	0		1	0

0	1	⇒	0	0
0	0		0	1

1	0	⇒	0	0
1	0		1	1

0	1	⇒	0	0
0	1		1	1

1	0	⇒	0	0
0	1		1	1

0	1	⇒	0	0
1	0		1	1

1	1	⇒	1	0
1	0		1	1

1	1	⇒	0	1
0	1		1	1

核心代码：

```
p=mod(i,2); %margolis neighborhood
sand(nx/2,ny/2) = 1; %add a grain at the top

%upper left cell update
xind = [1+p:2:nx-2+p];
yind = [1+p:2:ny-2+p];
%randomize the flow -- 10% of the time
vary = rand(nx,ny)< .9 ;
vary1 = 1-vary;

sandNew(xind,yind) = ...
    gnd(xind,yind).*sand(xind,yind) + ...
    (1-gnd(xind,yind)).*sand(xind,yind).*sand(xind,yind+1) .* ...
        (sand(xind+1,yind+1)+(1-sand(xind+1,yind+1)).*sand(xind+1,yind));
```

```
        sandNew(xind+1,yind) = ...
            gnd(xind+1,yind).*sand(xind+1,yind) + ...
            (1-gnd(xind+1,yind)).*sand(xind+1,yind).*sand(xind+1,yind+1) .* ...
                (sand(xind,yind+1)+(1-sand(xind,yind+1)).*sand(xind,yind));

        sandNew(xind,yind+1) = ...
            sand(xind,yind+1) + ...
            (1-sand(xind,yind+1)) .* ...
            (   sand(xind,yind).*(1-gnd(xind,yind)) + ...
         (1-sand(xind,yind)).*sand(xind+1,yind).*(1-gnd(xind+1,yind)).*sand(xind+1,yind+1));

         sandNew(xind+1,yind+1) = ...
            sand(xind+1,yind+1) + ...
            (1-sand(xind+1,yind+1)) .* ...
            (   sand(xind+1,yind).*(1-gnd(xind+1,yind)) + ...
            (1-sand(xind+1,yind)).*sand(xind,yind).*(1-gnd(xind,yind)).*sand(xind,yind+1));

        %scramble the sites to make it look better
        temp1 = sandNew(xind,yind+1).*vary(xind,yind+1) + ...
            sandNew(xind+1,yind+1).*vary1(xind,yind+1);

        temp2 = sandNew(xind+1,yind+1).*vary(xind,yind+1) + ...
            sandNew(xind,yind+1).*vary1(xind,yind+1);
        sandNew(xind,yind+1) = temp1;
        sandNew(xind+1,yind+1) = temp2;

    sand = sandNew;
```

参考文献

[1] Cellular Automata Modeling of Physical Systems

[2] Cellular Automata Machines by Tommaso Toffoli and Norman Margolus, MIT Press, 1987.

[3] Cellular Automata Modeling of Physical Systems by Bastien Chopard and Michel Droz, Cambridge University Press, 1998.

本文由一英文网页翻译而来，原文见：
http://instruct1.cit.cornell.edu/courses/bionb441/CA/

本文中的程序：

1、Conway’s life

```
%Conway's life with GUI
clf
clear all
%===========================
%build the GUI
%define the plot button
plotbutton=uicontrol('style','pushbutton',...
      'string','Run', ...
      'fontsize',12, ...
      'position',[100,400,50,20], ...
      'callback', 'run=1;');

%define the stop button
erasebutton=uicontrol('style','pushbutton',...
      'string','Stop', ...
      'fontsize',12, ...
      'position',[200,400,50,20], ...
      'callback','freeze=1;');

%define the Quit button
quitbutton=uicontrol('style','pushbutton',...
      'string','Quit', ...
      'fontsize',12, ...
      'position',[300,400,50,20], ...
      'callback','stop=1;close;');

number = uicontrol('style','text', ...
        'string','1', ...
      'fontsize',12, ...
      'position',[20,400,50,20]);
%=============================
%CA setup
n=128;
%initialize the arrays
z = zeros(n,n);
cells = z;
sum = z;
%set a few cells to one
cells(n/2,.25*n:.75*n) = 1;
cells(.25*n:.75*n,n/2) = 1;
```

```
%cells(.5*n-1,.5*n-1)=1;
%cells(.5*n-2,.5*n-2)=1;
%cells(.5*n-3,.5*n-3)=1;
cells = (rand(n,n))<.5 ;
%how long for each case to stability or simple oscillators

%build an image and display it
imh = image(cat(3,cells,z,z));
set(imh, 'erasemode', 'none')
axis equal
axis tight

%index definition for cell update
x = 2:n-1;
y = 2:n-1;

%Main event loop
stop= 0; %wait for a quit button push
run = 0; %wait for a draw
freeze = 0; %wait for a freeze

while (stop==0)

    if (run==1)
        %nearest neighbor sum
        sum(x,y) = cells(x,y-1) + cells(x,y+1) + ...
            cells(x-1, y) + cells(x+1,y) + ...
            cells(x-1,y-1) + cells(x-1,y+1) + ...
            cells(3:n,y-1) + cells(x+1,y+1);
        % The CA rule
        cells = (sum==3) | (sum==2 & cells);
        %draw the new image
        set(imh, 'cdata', cat(3,cells,z,z) )
        %update the step number diaplay
        stepnumber = 1 + str2num(get(number,'string'));
        set(number,'string',num2str(stepnumber))
    end

    if (freeze==1)
        run = 0;
        freeze = 0;
    end
    drawnow    %need this in the loop for controls to work
end
```

2、Surface Tension

```
%Conway's life with GUI
clf; clear all
%============================================
%build the GUI
%define the plot button
plotbutton=uicontrol('style','pushbutton',...
    'string','Run', ...
    'fontsize',12, ...
    'position',[100,400,50,20], ...
    'callback', 'run=1;');

%define the stop button
erasebutton=uicontrol('style','pushbutton',...
    'string','Stop', ...
    'fontsize',12, ...
    'position',[200,400,50,20], ...
    'callback','freeze=1;');

%define the Quit button
quitbutton=uicontrol('style','pushbutton',...
    'string','Quit', ...
    'fontsize',12, ...
    'position',[300,400,50,20], ...
    'callback','stop=1;close;');

number = uicontrol('style','text', ...
      'string','1', ...
    'fontsize',12, ...
    'position',[20,400,50,20]);
%============================================
%CA setup
n=128;
%initialize the arrays
z = zeros(n,n);
cells = z;
sum = z;
%set a few cells to one
cells(n/2,.25*n:.75*n) = 1;
cells(.25*n:.75*n,n/2) = 1;

%cells(.5*n-1,.5*n-1)=1;
%cells(.5*n-2,.5*n-2)=1;
%cells(.5*n-3,.5*n-3)=1;
```

```
cells = (rand(n,n))<.5 ;
%how long for each case to stability or simple oscillators

%build an image and display it
imh = image(cat(3,cells,z,z));
set(imh, 'erasemode', 'none')
axis equal
axis tight

%index definition for cell update
x = 2:n-1;
y = 2:n-1;

%Main event loop
stop= 0; %wait for a quit button push
run = 0; %wait for a draw
freeze = 0; %wait for a freeze

while (stop==0)

    if (run==1)
        %nearest neighbor sum
        sum(x,y) = cells(x,y-1) + cells(x,y+1) + ...
            cells(x-1, y) + cells(x+1,y) + ...
            cells(x-1,y-1) + cells(x-1,y+1) + ...
            cells(3:n,y-1) + cells(x+1,y+1)+...
            cells(x,y);
        % The CA rule
        cells = ~((sum<4) | (sum==5));
        %draw the new image
        set(imh, 'cdata', cat(3,cells,z,z) )
        %update the step number diaplay
        stepnumber = 1 + str2num(get(number,'string'));
        set(number,'string',num2str(stepnumber))
    end

    if (freeze==1)
        run = 0;
        freeze = 0;
    end

    drawnow    %need this in the loop for controls to work

end
```

3、Percolation Cluster

```
% Percolation Cluster
clf
clear all
threshold = .63;
%
ax = axes('units','pixels','position',[1 1 500 400],'color','k');
text('units', 'pixels', 'position', [50,255,0],...
    'string','BioNB','color','w','fontname','helvetica','fontsize',100)
text('units', 'pixels', 'position', [120,120,0],...
    'string','441','color','w','fontname','helvetica','fontsize',100)
initial = getframe(gca);

[a,b,c]=size(initial.cdata);
z=zeros(a,b);
cells = double(initial.cdata(:,:,1)==255);
visit = z ;
sum = z;

imh = image(cat(3,z,cells,z));
set(imh, 'erasemode', 'none')

%return

for i=1:100
    sum(2:a-1,2:b-1) = cells(2:a-1,1:b-2) + cells(2:a-1,3:b) + ...
                        cells(1:a-2, 2:b-1) + cells(3:a,2:b-1) + ...
                        cells(1:a-2,1:b-2) + cells(1:a-2,3:b) + ...
                        cells(3:a,1:b-2) + cells(3:a,3:b);

    pick =   rand(a,b);
    %edges only
    %cells = (cells & (sum<8)) | ((sum>=1) & (pick>=threshold) & (visit==0))   ;
    cells = cells   | ((sum>=1) & (pick>=threshold) & (visit==0))   ;
    visit = (sum>=1) ;%& (pick<threshold)   ;

    set(imh, 'cdata', cat(3,z,cells,z) )
    drawnow
end

return
figure(2)
image(cat(3,z,cells,z))
```

4、Excitable media (BZ reaction or heart)

```
%CA driver
%excitable media
clf; clear all
n=128;

z=zeros(n,n);
cells=z;

cells = (rand(n,n))<.1 ;
%cells(n/2,n*.25:n*.75) = 1;
%cells(n*.25:n*.75,n/2) = 1;
sum=z;

imh = image(cat(3,cells,z,z));
set(imh, 'erasemode', 'none')
axis equal
axis tight

x = [2:n-1]; y = [2:n-1];

t = 6; % center value=6; 7 makes fast pattern; 5 analiating waves
t1 = 3; % center value=3
for i=1:1200

    sum(x,y) = ((cells(x,y-1)>0)&(cells(x,y-1)<t)) + ((cells(x,y+1)>0)&(cells(x,y+1)<t)) + ...
        ((cells(x-1, y)>0)&(cells(x-1, y)<t)) + ((cells(x+1,y)>0)&(cells(x+1,y)<t)) + ...
        ((cells(x-1,y-1)>0)&(cells(x-1,y-1)<t)) + ((cells(x-1,y+1)>0)&(cells(x-1,y+1)<t)) + ...
        ((cells(x+1,y-1)>0)&(cells(x+1,y-1)<t)) + ((cells(x+1,y+1)>0)&(cells(x+1,y+1)<t));

    cells = ((cells==0) & (sum>=t1)) + ...
            2*(cells==1) + ...
            3*(cells==2) + ...
            4*(cells==3) + ...
            5*(cells==4) + ...
            6*(cells==5) +...
            7*(cells==6) +...
            8*(cells==7) +...
            9*(cells==8) +...
            0*(cells==9);
    set(imh, 'cdata', cat(3,z,cells/10,z) )
    drawnow
    end
```

5、Forest Fire

```
%CA driver
%
%forest fire

clf
clear all

n=100;

Plightning = .000005;
Pgrowth = .01; %.01

z=zeros(n,n);
o=ones(n,n);
veg=z;
sum=z;


imh = image(cat(3,z,veg*.02,z));
set(imh, 'erasemode', 'none')
axis equal
axis tight

% burning -> empty
% green -> burning if one neigbor burning or with prob=f (lightning)
% empty -> green with prob=p (growth)
% veg = {empty=0 burning=1 green=2}
for i=1:3000
    %nearby fires?

     sum = (veg(1:n,[n 1:n-1])==1) + (veg(1:n,[2:n 1])==1) + ...
           (veg([n 1:n-1], 1:n)==1) + (veg([2:n 1],1:n)==1) ;

    veg = ...
         2*(veg==2) - ((veg==2) & (sum>0 | (rand(n,n)<Plightning))) + ...
         2*((veg==0) & rand(n,n)<Pgrowth) ;

    set(imh, 'cdata', cat(3,(veg==1),(veg==2),z) )
    drawnow
end
```

6、Gas dynamics

```
%CA driver
%HPP-gas

clear all
clf

nx=52; %must be divisible by 4
ny=100;

z=zeros(nx,ny);
o=ones(nx,ny);
sand = z ;
sandNew = z;
gnd = z ;
diag1 = z;
diag2 = z;
and12 = z;
or12 = z;
sums = z;
orsum = z;

gnd(1:nx,ny-3)=1 ; % right ground line
gnd(1:nx,3)=1 ; % left ground line
gnd(nx/4:nx/2-2,ny/2)=1; %the hole line
gnd(nx/2+2:nx,ny/2)=1; %the hole line
gnd(nx/4, 1:ny) = 1; %top line
gnd(3*nx/4, 1:ny) = 1 ;%bottom line

%fill the left side
r = rand(nx,ny);
sand(nx/4+1:3*nx/4-1, 4:ny/2-1) = r(nx/4+1:3*nx/4-1, 4:ny/2-1)<0.3;
%sand(nx/4+1:3*nx/4-1, ny*.75:ny-4) = r(nx/4+1:3*nx/4-1, ny*.75:ny-4)<0.75;
%sand(nx/2,ny/2) = 1;
%sand(nx/2+1,ny/2+1) = 1;

imh = image(cat(3,z,sand,gnd));
set(imh, 'erasemode', 'none')
axis equal
axis tight
```

```
for i=1:1000
    p=mod(i,2); %margolis neighborhood

    %upper left cell update
    xind = [1+p:2:nx-2+p];
    yind = [1+p:2:ny-2+p];

    %See if exactly one diagonal is ones
    %only (at most) one of the following can be true!
    diag1(xind,yind) = (sand(xind,yind)==1) & (sand(xind+1,yind+1)==1) & ...
        (sand(xind+1,yind)==0) & (sand(xind,yind+1)==0);

    diag2(xind,yind) = (sand(xind+1,yind)==1) & (sand(xind,yind+1)==1) & ...
        (sand(xind,yind)==0) & (sand(xind+1,yind+1)==0);

    %The diagonals both not occupied by two particles
    and12(xind,yind) = (diag1(xind,yind)==0) & (diag2(xind,yind)==0);

    %One diagonal is occupied by two particles
    or12(xind,yind)    = diag1(xind,yind) | diag2(xind,yind);

    %for every gas particle see if it near the boundary
    sums(xind,yind) = gnd(xind,yind) | gnd(xind+1,yind) | ...
                            gnd(xind,yind+1) | gnd(xind+1,yind+1) ;

    % cell layout:
    % x,y       x+1,y
    % x,y+1    x+1,y+1
    %If (no walls) and (diagonals are both not occupied)
    %then there is no collision, so move opposite cell to current cell
    %If (no walls) and (only one diagonal is occupied)
    %then there is a collision so move ccw cell to the current cell
    %If (a wall)
    %then don't change the cell (causes a reflection)
    sandNew(xind,yind) = ...
        (and12(xind,yind)    & ~sums(xind,yind) & sand(xind+1,yind+1)) + ...
        (or12(xind,yind) & ~sums(xind,yind) & sand(xind,yind+1)) + ...
        (sums(xind,yind) & sand(xind,yind));

    sandNew(xind+1,yind) = ...
        (and12(xind,yind)    & ~sums(xind,yind) & sand(xind,yind+1)) + ...
        (or12(xind,yind) & ~sums(xind,yind) & sand(xind,yind))+ ...
        (sums(xind,yind) & sand(xind+1,yind));
```

```
        sandNew(xind,yind+1) = ...
            (and12(xind,yind)    & ~sums(xind,yind) & sand(xind+1,yind)) + ...
            (or12(xind,yind) & ~sums(xind,yind) & sand(xind+1,yind+1))+ ...
            (sums(xind,yind) & sand(xind,yind+1));

         sandNew(xind+1,yind+1) = ...
            (and12(xind,yind)    & ~sums(xind,yind) & sand(xind,yind)) + ...
            (or12(xind,yind) & ~sums(xind,yind) & sand(xind+1,yind))+ ...
            (sums(xind,yind) & sand(xind+1,yind+1));

        sand = sandNew;

        set(imh, 'cdata', cat(3,z,sand,gnd) )
        drawnow
end
```

8、Diffusion limited aggregation

```
%diffusion + dla

clear all
clf

nx=200; %must be divisible by 4
ny=200;

z=zeros(nx,ny);
o=ones(nx,ny);
sand = z ;
sandNew = z;
sum = z;
gnd = z;
gnd(nx/2,ny/2) = 1 ;

sand = rand(nx,ny)<.1;

imh = image(cat(3,z,sand,gnd));
set(imh, 'erasemode', 'none')
axis equal
axis tight

for i=1:10000
        p=mod(i,2); %margolis neighborhood
```

```
    %upper left cell update
    xind = [1+p:2:nx-2+p];
    yind = [1+p:2:ny-2+p];
    %random velocity choice
    vary = rand(nx,ny)<.5 ;
    vary1 = 1-vary;

    %diffusion rule -- margolus neighborhood
    %rotate the 4 cells to randomize velocity
    sandNew(xind,yind) = ...
        vary(xind,yind).*sand(xind+1,yind) + ... %cw
        vary1(xind,yind).*sand(xind,yind+1) ;     %ccw

    sandNew(xind+1,yind) = ...
        vary(xind,yind).*sand(xind+1,yind+1) + ...
        vary1(xind,yind).*sand(xind,yind) ;

    sandNew(xind,yind+1) = ...
        vary(xind,yind).*sand(xind,yind) + ...
        vary1(xind,yind).*sand(xind+1,yind+1) ;

     sandNew(xind+1,yind+1) = ...
        vary(xind,yind).*sand(xind,yind+1) + ...
        vary1(xind,yind).*sand(xind+1,yind) ;

    sand = sandNew;

    %for every sand grain see if it near the fixed, sticky cluster
    sum(2:nx-1,2:ny-1) = gnd(2:nx-1,1:ny-2) + gnd(2:nx-1,3:ny) + ...
                         gnd(1:nx-2, 2:ny-1) + gnd(3:nx,2:ny-1) + ...
                         gnd(1:nx-2,1:ny-2) + gnd(1:nx-2,3:ny) + ...
                         gnd(3:nx,1:ny-2) + gnd(3:nx,3:ny);

    %add to the cluster
    gnd = ((sum>0) & (sand==1)) | gnd ;
    %and eliminate the moving particle
    sand(find(gnd==1)) = 0;

    set(imh, 'cdata', cat(3,gnd,gnd,(sand==1)) );
    drawnow
end
```

9、Sand pile

```
%sand pile
clear all
clf

nx=52; %must be divisible by 4
ny=100;

Pbridge = .05;

z=zeros(nx,ny);
o=ones(nx,ny);
sand = z ;
sandNew = z;
gnd = z ;
gnd(1:nx,ny-3)=1 ; % the ground line
gnd(nx/4:nx/2+4,ny-15)=1; %the hole line
gnd(nx/2+6:nx,ny-15)=1; %the hole line
gnd(nx/4, ny-15:ny) = 1; %side line
gnd(3*nx/4, 1:ny) = 1 ;

imh = image(cat(3,z',sand',gnd'));
set(imh, 'erasemode', 'none')
axis equal
axis tight


for i=1:1000
    p=mod(i,2); %margolis neighborhood
    sand(nx/2,ny/2) = 1; %add a grain at the top

    %upper left cell update
    xind = [1+p:2:nx-2+p];
    yind = [1+p:2:ny-2+p];
    vary = rand(nx,ny)<.95 ;
    vary1 = 1-vary;

    sandNew(xind,yind) = ...
        gnd(xind,yind).*sand(xind,yind) + ...
        (1-gnd(xind,yind)).*sand(xind,yind).*sand(xind,yind+1) .* ...
            (sand(xind+1,yind+1)+(1-sand(xind+1,yind+1)).*sand(xind+1,yind));
```

```
    sandNew(xind+1,yind) = ...
        gnd(xind+1,yind).*sand(xind+1,yind) + ...
        (1-gnd(xind+1,yind)).*sand(xind+1,yind).*sand(xind+1,yind+1) .* ...
            (sand(xind,yind+1)+(1-sand(xind,yind+1)).*sand(xind,yind));

    sandNew(xind,yind+1) = ...
        sand(xind,yind+1) + ...
        (1-sand(xind,yind+1)) .* ...
        (    sand(xind,yind).*(1-gnd(xind,yind)) + ...

(1-sand(xind,yind)).*sand(xind+1,yind).*(1-gnd(xind+1,yind)).*sand(xind+1,yind+1));

     sandNew(xind+1,yind+1) = ...
        sand(xind+1,yind+1) + ...
        (1-sand(xind+1,yind+1)) .* ...
        (    sand(xind+1,yind).*(1-gnd(xind+1,yind)) + ...
           (1-sand(xind+1,yind)).*sand(xind,yind).*(1-gnd(xind,yind)).*sand(xind,yind+1));

    %scramble the sites to make it look better
    temp1 = sandNew(xind,yind+1).*vary(xind,yind+1) + ...
        sandNew(xind+1,yind+1).*vary1(xind,yind+1);

    temp2 = sandNew(xind+1,yind+1).*vary(xind,yind+1) + ...
        sandNew(xind,yind+1).*vary1(xind,yind+1);
    sandNew(xind,yind+1) = temp1;
    sandNew(xind+1,yind+1) = temp2;

    sand = sandNew;
    set(imh, 'cdata', cat(3,z',sand',gnd') )
    drawnow
end
```