Macm 203 Assignment 4

April Nguyen

Table of Contents

Preamble

Here we consider 1D automata. There are a number of online references to Wolfram's enumeration of one-dimensional nearest neighbour automata. For example, see here:
https://mathworld.wolfram.com/ElementaryCellularAutomaton.html

In this assignment, you will write a Matlab code to construct a space-time pattern for one-dimensional nearest neighbour automata. and analyze an interesting case.

Questions.

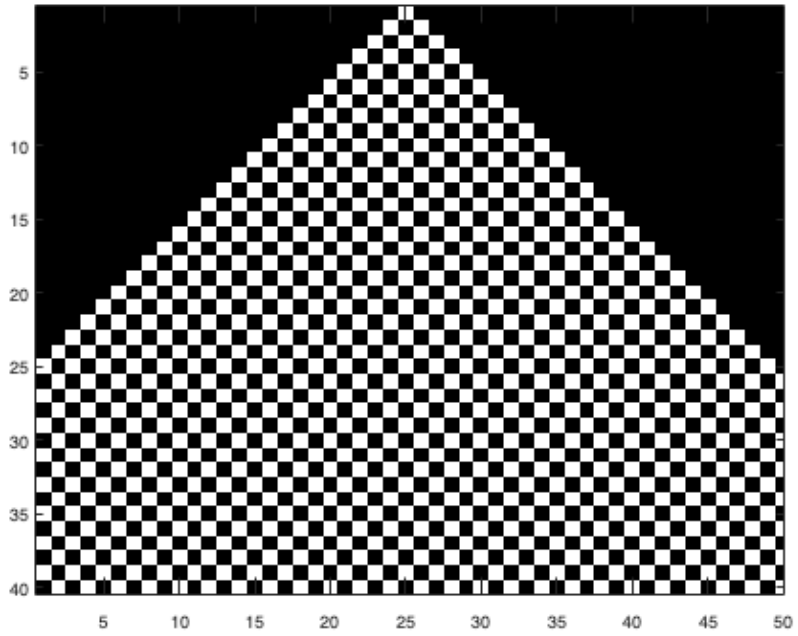
Part A: Matlab code for space time patterns

Write a code to generate space-time patterns for a one-dimensional nearest neighbour automaton. Your code should have the following features:

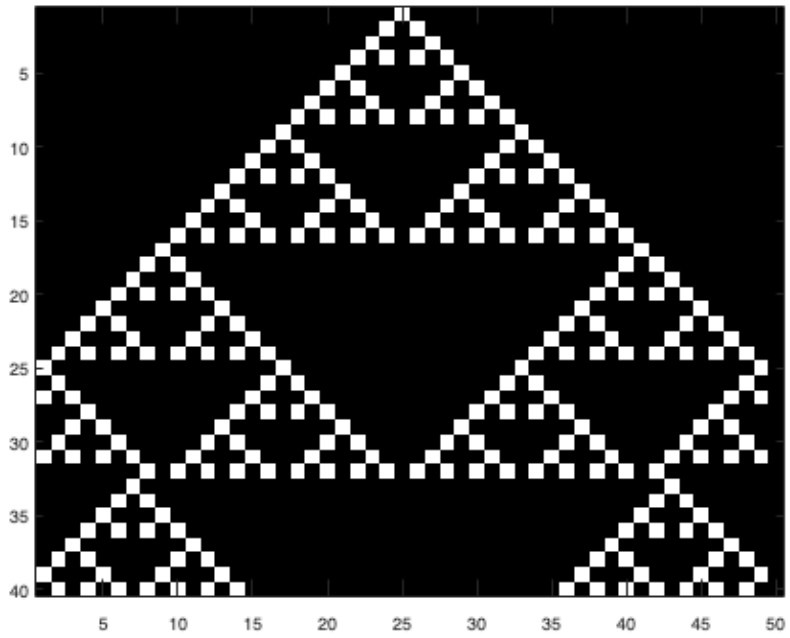
1. Impose periodic boundary conditions on a domain of width w.
2. Include an *anonymous function* which implements the rule. The rule will input an integer from 0 to 7, corresponding to a binary number that indexes the state of the local neighbourhood. The rule will output either 0 or 1, specifying the updated state of the neighbourhood.
3. Output a figure displaying a space-time pattern for the automaton for a given state.

```
% 1. Impose periodic boundary conditions on a domain of width w.
w = 50;             % domain has width w
l = 40;             % number of generations
right = [2:w 1];    % right neighbour
left = [w 1:w-1];   % left neighbour

% 2. Include an anonymous function which implements the rule (Wolfram Rule 50).
% The rule will input an integer from 0 to 7, corresponding to a binary number
% that indexes the state of the local neighbourhood. The rule will output either
% 0 or 1, specifying the updated state of the neighbourhood.
Rule50 = @(x) any(x==[1,4,5]); % where x represents the state of the neighbourhood
Rule90 = @(x) any(x==[1,3,4,6]);
Rule254 = @(x) any(x==[1,2,3,4,5,6,7]);

% 3. Output a figure displaying a space-time pattern for the automaton for
% a given initial state.
center = floor(w/2);
GRID50=zeros(l,w);
```

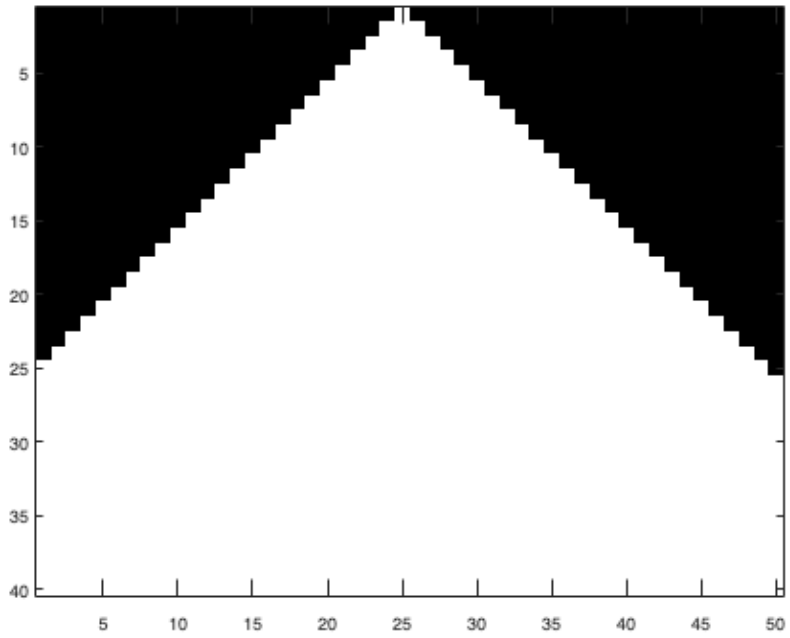
```
GRID50(1,center) = 1;
%GRID50(1,:) = randi([0,1],[1,w]);
colormap(gray(2));

for i=2:l
    for j=1:w
        state = GRID50(i-1,right(j))*(2^2) + GRID50(i-1,j)*2 + GRID50(i-1,left(j));
        GRID50(i,j) = Rule50(state);
        image(GRID50*2)
        pause(0.01);
    end
end

GRID90=zeros(l,w);
GRID90(1,center) = 1;
%GRID90(1,:) = randi([0,1],[1,w]);
colormap(gray(2));
```




```
for i=2:l
    for j=1:w
        state = GRID90(i-1,right(j))*(2^2) + GRID90(i-1,j)*2 + GRID90(i-1,left(j));
        GRID90(i,j) = Rule90(state);
        image(GRID90*2);
        pause(0.01);
    end
end

GRID254=zeros(l,w);
GRID254(1,center) = 1;
%GRID254(1,:) = randi([0,1],[1,w]);
colormap(gray(2));
```



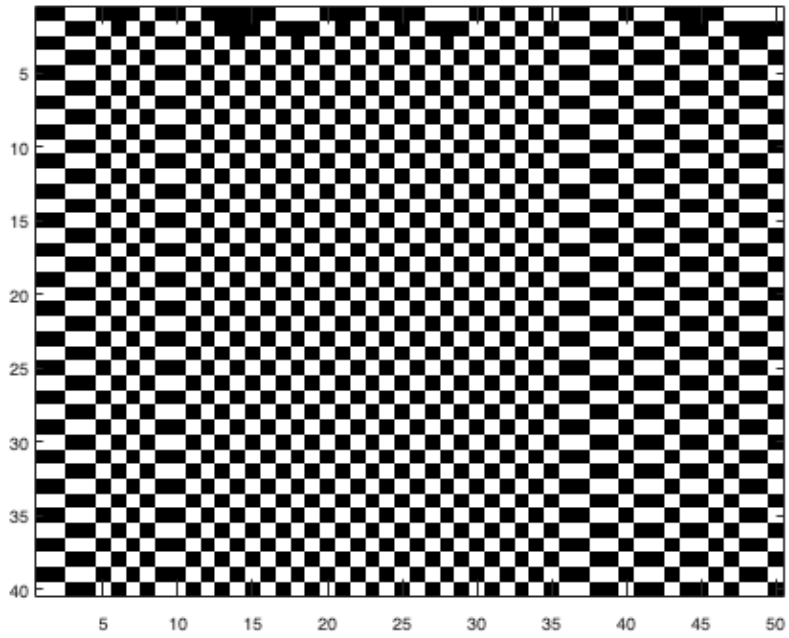

```
for i=2:l
    for j=1:w
        state = GRID254(i-1,right(j))*(2^2) + GRID254(i-1,j)*2 + GRID254(i-1,left(j));
        GRID254(i,j) = Rule254(state);
        image(GRID254*2);
        pause(0.01);
    end
end
```



Part B: Code Verification

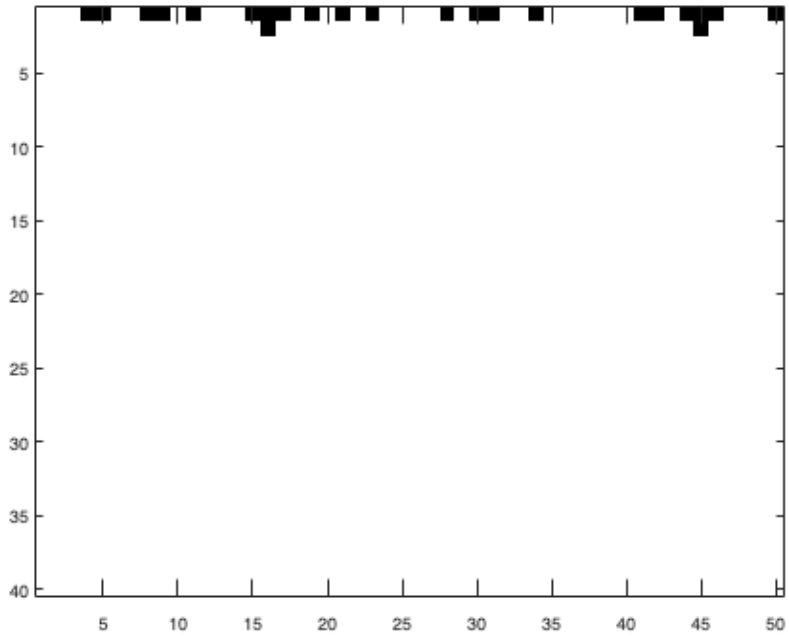
In class we considered Wolfram Rules 50, 90, and 254. Set up random initial conditions, evolve for a number of time steps and plot the results. For these 3 Wolfram Rules, do you see similar results to those demonstrated in class?

```
% Code from Part A
clf;
w = 50;
l = 40;
right = [2:w 1];
left = [w 1:w-1];

Rule50 = @(x) any(x==[1,4,5]);
Rule90 = @(x) any(x==[1,3,4,6]);
Rule254 = @(x) any(x==[1,2,3,4,5,6,7]);

center = floor(w/2);

GRID50=zeros(l,w);
%GRID50(1,center) = 1;
GRID50(1,:) = randi([0,1],[1,w]);
colormap(gray(2));

for i=2:l
    for j=1:w
        state = GRID50(i-1,right(j))*(2^2) + GRID50(i-1,j)*2 + GRID50(i-1,left(j));
        GRID50(i,j) = Rule50(state);
        image(GRID50*2)
        pause(0.01);
    end
end

GRID90=zeros(l,w);
%GRID90(1,center) = 1;
GRID90(1,:) = randi([0,1],[1,w]);
colormap(gray(2));
```




```
for i=2:l
    for j=1:w
        state = GRID90(i-1,right(j))*(2^2) + GRID90(i-1,j)*2 + GRID90(i-1,left(j));
        GRID90(i,j) = Rule90(state);
        image(GRID90*2);
        pause(0.01);
    end
end

GRID254=zeros(l,w);
%GRID254(1,center) = 1;
GRID254(1,:) = randi([0,1],[1,w]);
colormap(gray(2));
```


```
for i=2:l
    for j=1:w
        state = GRID254(i-1,right(j))*(2^2) + GRID254(i-1,j)*2 + GRID254(i-1,left(j));
        GRID254(i,j) = Rule254(state);
        image(GRID254*2);
        pause(0.01);
    end
end
```



Yes, with these 3 Wolfram Rules, there are similar results to those demonstrated in class.

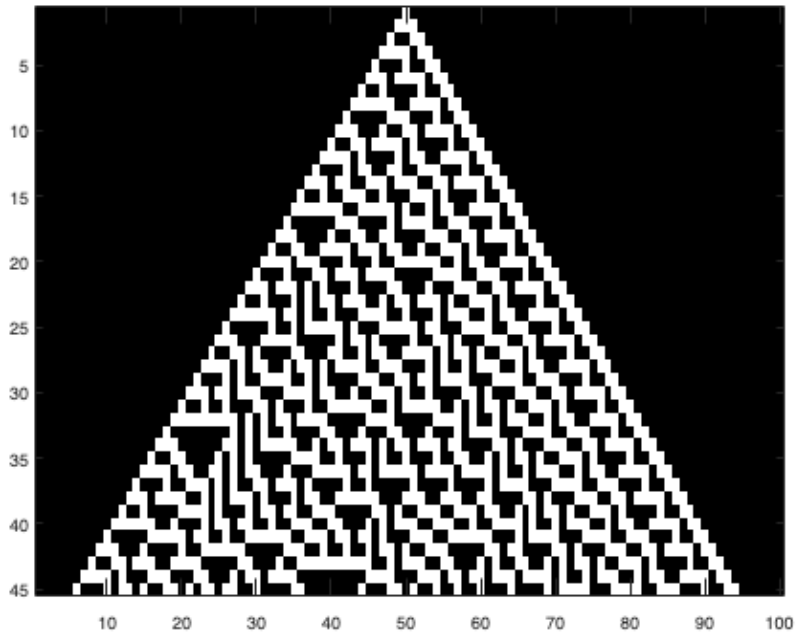
Part C: Wolfram Rule 30

In this problem choose a domain that is wide enough that the pattern does not interact with the boundary.

Start with a single black cell (state=1) in the centre column and apply Wolfram Rule 30. You might expect a simple pattern, but you instead see a complex evolving configuration. Provide a figure displaying such an evolution.

Does each colour of cell occur on average equally often in the centre column? Do not attempt a proof (unless you are curious and have spare time). Instead explore this question via computer experiments. Justify your answer.

```
w = 100;
l = 45;
right = [2:w 1];
left = [w 1:w-1];
Rule30 = @(x) any(x==[1,2,3,4]);
center = floor(w/2);

GRID30=zeros(l,w);
GRID30(1,center) = 1;
colormap(gray(2));

for i=2:l
    for j=1:w
        state = GRID30(i-1,right(j))*(2^2) + GRID30(i-1,j)*2 + GRID30(i-1,left(j));
        GRID30(i,j) = Rule30(state);
        image(GRID30*2)
        pause(0.01);
    end
end
```




```
% Experiments
w = 100;
l = 1000000;
right = [2:w 1];
```

```
left = [w 1:w-1];
Rule30 = @(x) any(x==[1,2,3,4]);
center = floor(w/2);

GRID30=zeros(l,w);
GRID30(1,center) = 1;

for i=2:l
    for j=1:w
        state = GRID30(i-1,right(j))*(2^2) + GRID30(i-1,j)*2 + GRID30(i-1,left(j));
        GRID30(i,j) = Rule30(state);
    end
end

centercol_1gen = GRID30(1,center);
ones_1gen = sum(centercol_1gen == 1) / 1
```

```
ones_1gen = 1
```

```
centercol_10gen = GRID30(1:10,center);
ones_10gen = sum(centercol_10gen == 1) / 10
```

```
ones_10gen = 0.7000
```

```
centercol_100gen = GRID30(1:100,center);
ones_100gen = sum(centercol_100gen == 1) / 100
```

```
ones_100gen = 0.5200
```

```
centercol_1000gen = GRID30(1:1000,center);
ones_1000gen = sum(centercol_1000gen == 1) / 1000
```

```
ones_1000gen = 0.5090
```

```
centercol_100000gen = GRID30(1:100000,center);
ones_100000gen = sum(centercol_100000gen == 1) / 100000
```

```
ones_100000gen = 0.5012
```

```
centercol_1000000gen = GRID30(1:1000000,center);
ones_1000000gen = sum(centercol_1000000gen == 1) / 1000000
```

```
ones_1000000gen = 0.4996
```

From these computations, it appears that as we increase the generations, each colour of cell appears to be occuring equally as often.