VG101 FA2021 lab2: Conway's Game of Life

In memory of British Mathematician John Horton Conway, who died of COVID-19 in 2020.

Author: Shuyu Wu. Please email wushuyu2002@sjtu.edu.cn if you find any error or ambiguity in this file.

1. background

The game of life, invented by British Mathematician John Horton Conway, is also called "Cellular automaton". It is a game with no player required. The game rule is listed above:

Suppose there's an infinite two-dimensional orthogonal grid. In each grid there's a cell, either dead or alive. At each step of the game, some alive cell will die, and some dead cell will be alive. The rules are:

1. If an alive cell has less than 2 alive cell around, or more than 3 alive cell around, it will die in the next step.
2. If a dead cell has exactly 3 alive cell around, it will be alive in the next step.
3. In other cases, the state of the cell remain still. For example, an alive cell with 2 or 3 alive cell around will keep alive in the next step, and a dead cell with 0-2, 4-8 alive cell around will remain dead.

"The cell around" means the eight cells which enclose the center cell.

Let's see an example. Let "0" to represent the dead cell and "1" to represent the alive cell.

initial state:

```
0    0    0    0    0
0    0    1    0    0
0    1    0    0    0
0    0    0    1    0
0    0    0    0    0
```

after one step:

```
0    0    0    0    0
0    0    0    0    0
0    0    1    0    0
0    0    0    0    0
0    0    0    0    0
```

All the previously alive cells died because these cells have at most 1 cell around. But one dead cell becomes alive because there's three alive cells around it in the last step.

2. tasks

In this lab, you need to use matlab to simulate the game of life. You need to use a **finite** rectangle to substitute the infinite two-dimensional orthogonal grid.

Please follow the direction below to finish this lab. Codes that does not follow instructions will get deductions.

step 1: initialization

There should be three elements **for user to input**: a, b and p. "a" and "b" stands for the total size, meaning that the rectangle's size (i.e. , the total cells, both alive and dead) is a*b. "p" is the proportion of the alive cell at the beginning. For example, if a=10, b=15, p=0.3, then the rectangle is 10*15 size and the initial alive cell is 10*15*0.3=45, and the 45 alive cells are distributed **randomly** in the grid.

Use a matrix with elements 1 and 0 to represent the grid with cells. 1 stands for alive cell and 0 stand for dead cell.

Please write a function with prototype

```
function board=init_board()
```

and put the function in a file "init_board.m".

After the initialization part, the game board is set up.

The effect of the function should be as followed:

```
>> init_board
row=8 % '8' here is user input
column=8 % '8' here is user input
init probability of the life, in [0.1,0.5]: 0.3 % '0.3' here is user input

ans =

     1     0     0     0     1     0     1     0
     0     0     0     0     0     0     0     0
     0     1     1     0     1     0     1     1
     0     0     0     1     0     0     0     0
     0     0     0     1     0     0     1     1
     0     0     1     0     1     1     0     0
     0     0     1     0     0     1     0     0
     0     0     0     1     0     1     0     0
```

step 2: update

This part can update the game board. Please write a function with prototype

```
function board=update_board(in_board)
```

Which input the current game board and return the game board of the next step. The effect of the function should be as followed:

```
>> board

board =

     0     0     0     0     0
     0     0     1     0     0
     0     1     0     0     0
     0     0     0     1     0
```

```
     0     0     0     0     0

>> update_board(board)

ans =

     0     0     0     0     0
     0     0     0     0     0
     0     0     1     0     0
     0     0     0     0     0
     0     0     0     0     0
```

step 3: view

You need to turn the 0-1 matrix into concrete picture that can display these cells. You can use circle or rectangle to represent the alive cells, and the dead cells can be simply left blank.

You may use `scatter` function to draw points.

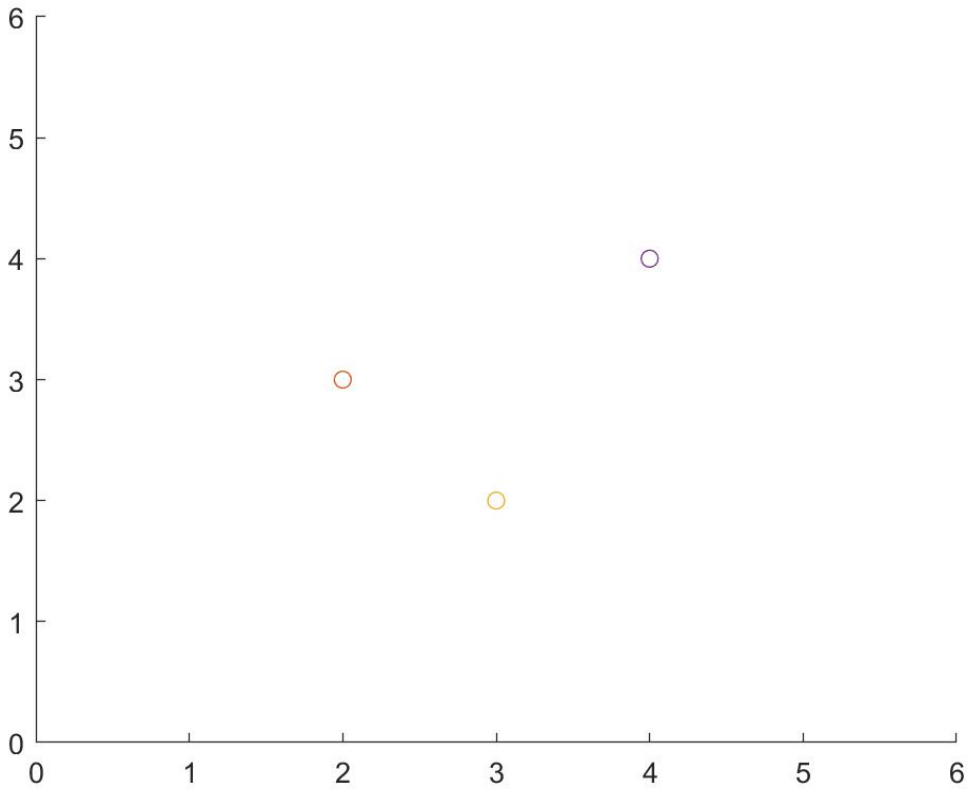
In this part, you only need to draw the picture with respect to a fixed matrix.

In this part, please write a function with prototype

```
function view_board(in_board)
```

This kind of picture is good enough:

```
>> view_board(board)
```



step 4: put them together

With the 3 functions above, you can write a matlab script to put them together in order to simulate the Cellular automaton.

Each step should last 0.5 second, and the game should process automatically.

In the recitation class in week 3, we'll see a short demo of the lab.

other requirements

1. You should let p in the range of [0.1, 0.5]. If a user gives a number less than 0.1, you should fix p at 0.1 and give out a message. If the users give a number more than 0.5, you should fix p at 0.5 and give out a message.

```
>> init_board
row=6
column=6
init probability of the life, in [0.1,0.5]: 0
too small, p fixed at 0.1

ans =

     1     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     1     0     0
     0     0     0     0     0     1
     0     0     0     0     0     0
     0     0     0     0     0     0
```

2. You can use `pause` function to pause 0.5 second after each step to let people have enough time to see the current situation.
3. The axis should be fixed when displaying. You can use `axis` function.

3. tips

1. You can refer to search engine or wikipedia to know more about the game of life.
2. Actually, you can type `life` directly in matlab to see a demo of the game of life made by MathWorks, Inc. However, the source code may not help you in this lab.
3. Please use `help` document in matlab to get help about the functions you're not familiar with. For example, `help scatter`, `help axis` . You can also use Google/Bing/Baidu (before asking TA for help).
4. Press Ctrl-C in command line to interrupt the execution of your code.

4. Rubric

Init function: 15 pts

Update function: 25 pts

view function: 20 pts

put them together: 25 pts

fulfill other requirements and get ideal result: 15 pts

testcase

$0 < a, b \leq 30, p \in \mathbb{R}$

deduction

Do not have 4 files: -50 pts (**You should follow the instructions and have 4 files, i.e., 3 functions and one script, in total.**)

Use global variables: -10 pts. (We do not need any global variable in this lab.)

5. submission

Your solution should be written into four different MATLAB files. When you come to TA during the lab, show these files to TA and run it to show the output according to TA's instructions. We will grade according to both your output, oral presentation and your code.

After the lab session is over, remember to compress your 4 files into a single zip file in the following name format:
lab2-[Your Last Name]-[Your First Name]-[Your SJTU ID].zip,
for example, lab2-Wu-Shuyu-520370910097.zip
and submit it to corresponding homework page on canvas.