Aaron Kuo
UID: 305-170-284
CEE / MAE M20
July 31st 2020

HOMEWORK 05

1 The Shared Birthday Problem

1.1 Introduction

The objective of this problem is to develop a script that solves a Monte Carlo simulation to determine how many have to be added into a group before two share the same birthday week. This code will run this simulation 10,000 times to calculate and print the average amount of people as well as plot a histogram of the results.

1.2 Model and Theory

This script lacks additional functions and equations to work, rather it uses simple logic. Arrays for the amount of days in a year and the number of trials to be done are first created. Randomized days are inserted into the group using the `randi` function and having it set to 365. This provides us with randomized integers from 1 to 365 representing the possible birthdays (excluding leap year.) To solve if the birthdays are in the same week, simple arithmetic is used to subtract the absolute value of each new birthdate and all the birthdates that are already in the group. If the difference is less than 7, then we know the birthdays are only 7 days apart and the trial ends.

1.3 Methods and Pseudo-code

The calculation process goes as follows:

1. Set up zero arrays for number of trials and days in a year
2. For loop begins for 10,000 trials
    a. For loop begins for 53 random dates
        i. (53 is chosen since there are only 52 weeks in a year, it is impossible to pick more than 53 dates and not having 2 be on the same week or have a 7 day difference)
        ii. Calculations begin

```
if (abs(days(b) - days(k)) < 7)
trialSpace(trials) = k;
end
if (abs(days(b) - 365 - days(k)) < 7) || (abs(days(k)- 365 -
days(b)) < 7)
trialSpace(trials) = k;
end
```

        iii. The above code is used to calculate if two dates are 7 days apart
        iv. Loop ends when two dates are a week apart
    b. Loop ends when all 10,000 trials are finished.

1.4 Calculations and Results

When the function is run, the following output is displayed.

```
Median Number of People =  7
```

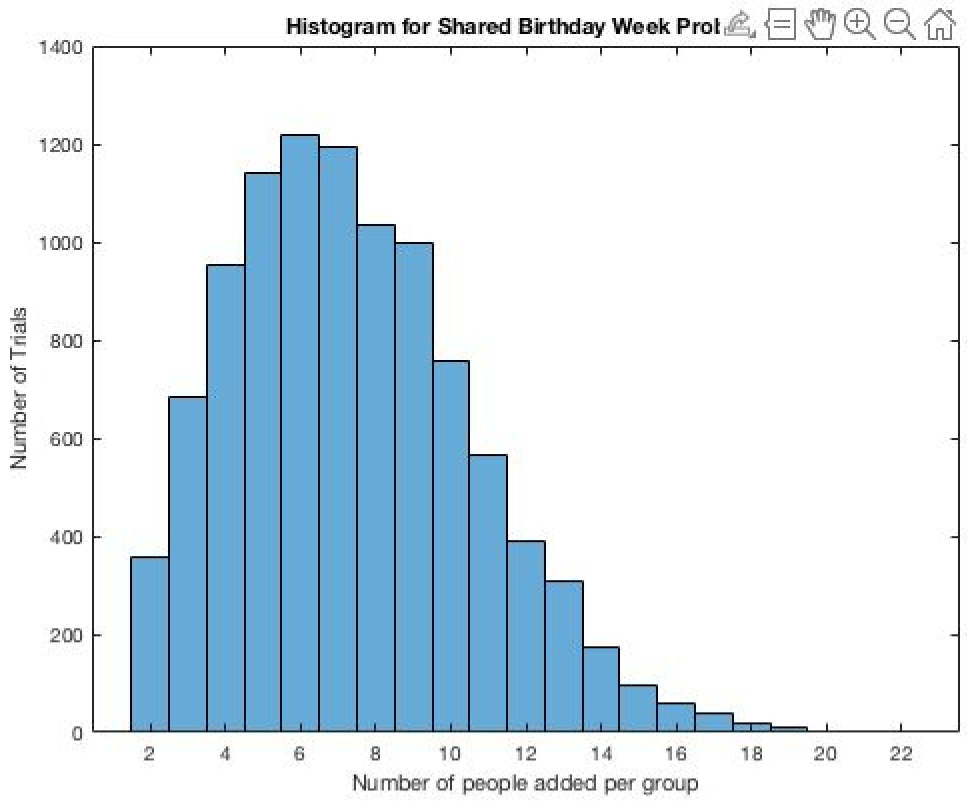

Figure 1: Histogram Plot displaying results of the average number of people added to a group before two people have matching birthday weeks.

1.5 Discussions and Conclusions

- In regards to the median, the number seems to be a bit too low for my personal belief. I expected the median to be much higher since it seems very unlikely that just in 7 people, it is more likely for two of them to have birthdays seven days apart.
- I would expect this value to increase or decrease drastically if the fact that birthdays have different probabilities were taken into account. This is very hard to verify since it seems unlikely someone is actively updating the statistics for how many people are born on each day and are still present. Patterns will emerge if this was possible and different days with higher probability will appear more often. Based on these higher probability days, they can either be 7 days apart, for example if July 1st and July 3rd have very high probability, which will in turn decrease the average number. On the other hand, they can be very far apart, if all the high probability days are 7 days apart, say every high probability day is at least 8 days apart, then the median would increase.

2. Random Walk Collisions

2.1 Introduction

The objective of this problem is to model the movements and collision of two particles on an 11 x 11 grid. The particles are stationed opposite of each other along the wall of the grid along the x-axis. Each particle is allowed to move one step in the up, down, left, right directions or not move at all. The script wants to determine after how many steps will the particles collide. Two scenarios are present in this problem, one where only one particle is moving, and one where both particles are moving. There will be 5000 trials per scenario and the user inputs which scenario is run. The average number of steps taken before collision will be calculated and printed.

2.2 Model and Theory

The user-made DrawRect function is a function created to color in the path of each particle's movement. The credit has to be given to Professor Gao for including this function in his examples for Lecture 5A. The colors of each path are adjustable in this function and it will be used at every step of each particles' movement to denote where it has traveled to.

The user made moveX and moveY function practically have the same function, just differ in the axis of movement. Both functions need the random variable `randomMove` to function. This variable is a randomized value from 0 to 1. If the value is between 0.2 - 0.4, or 0.6 - 0.8, then the moveX function is activated. If the value is between 0.2 - 0.4, then the function moves the particle once to the right. If the particle is at the right wall already, the function will know not to move the particle at all and let it remain. This is the basic concept for all the other values of moveX and moveY. The chart below will simplify the two functions.

Randomized value of randomMove	Action
0 - 0.2	Move Up
0.2 - 0.4	Move Right
0.4 - 0.6	Move Down
0.6 - 0.8	Move Left
0.8 - 1	No Movement

The script is able to detect a collision by checking at every interval if the coordinates of the two particles equal each other. If not, the particles keep moving until the coordinates match.

2.3 Methods and Pseudo-code

```
1. Input which scenario to run, 1 or 2
    a. Scenario 1: One particle moving, one stationary
        i. Set up grid specifications
        ii. Set up trial and move arrays
        iii. Set initial positions of both particles
            1. Moving particle at (-5,0)
            2. Stationary particle at (5,0)
        iv. For loop begins for total amount of trials
        v. While loop begins and continues as long as there are no collisions and the number of steps is below 1000
            1. Random value randomMove is determined
            2. Particle moves in a direction based on the value
            3. Check if particle has same coordinates
            4. If particles have the same coordinates, this indicates a collision and the while loop is ended.
        vi. Number of moves is counted and saved in trial array
```

vii. For loop is stopped after all trials are finished

viii. Calculates and prints out average number of steps before collision

b. Scenario 2: Both particles are moving

  i. Set up grid specifications

  ii. Set up trial and move arrays

  iii. Set initial positions of both particles

    1. Moving particle A at (-5,0)
    2. Moving particle B at (5,0)

  iv. For loop begins for total amount of trials

    1. Random values randomMoveA and randomValueB are determined for each particle
    2. Particles move in a direction based on the value
    3. Check if particles have the same coordinates
    4. If particles have the same coordinates, this indicates a collision and the while loop is ended.

  v. Number of steps is counted and saved in trial array

  vi. For loop is stopped after all trials are finished

  vii. Calculates and prints out average number of steps before collision

2.4 Calculations and Results

If scenario 1 is chosen, the resulting output from the script is:

```
Scenario 1: One particle moving, one fixed
Scenario 2: Both particles are moving

Which scenario would you like to run? 1 or 2?
1

Median = 96.00
```

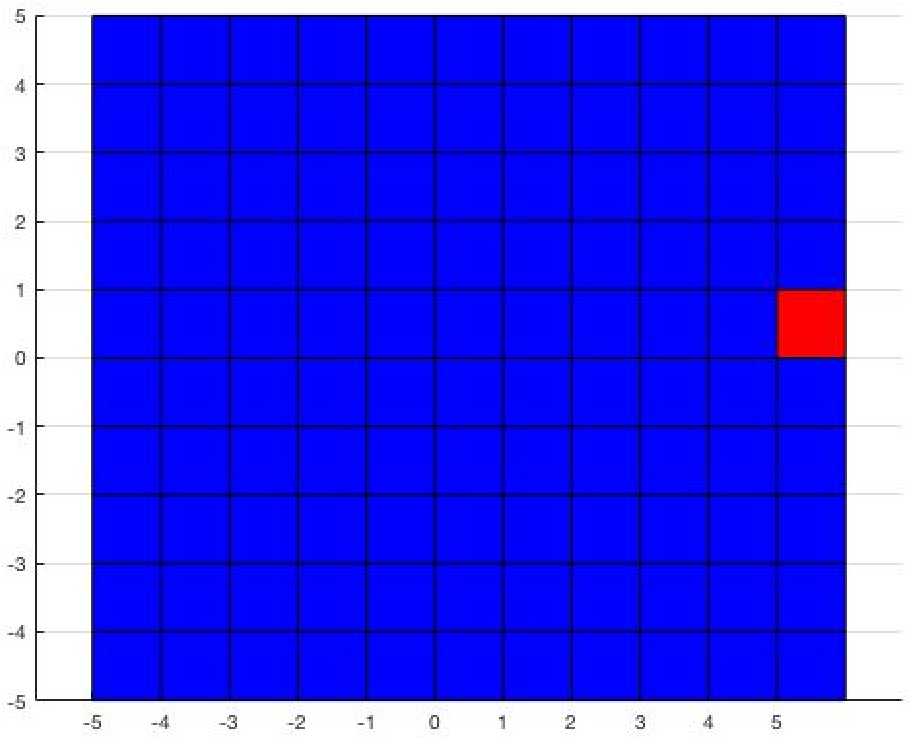

Figure 2: Example of a grid where one particle is moving (blue) and the other is stationary (red) after trial is complete

If scenario 2 is chosen, the resulting output from the script is:

```
Scenario 1: One particle moving, one fixed
Scenario 2: Both particles are moving

Which scenario would you like to run? 1 or 2?
2

Median = 67.00
```

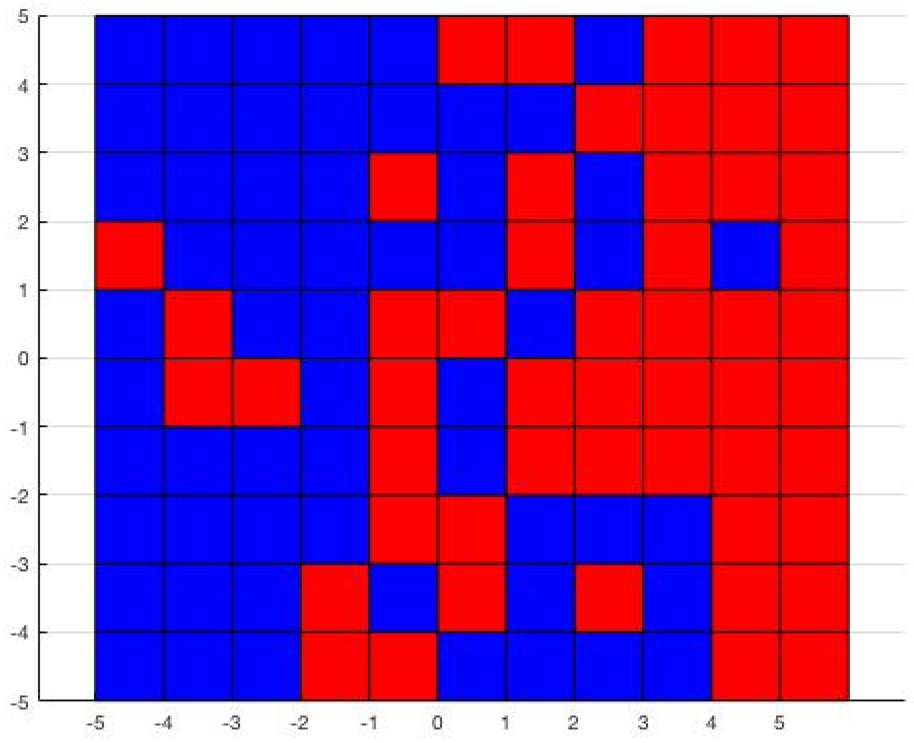

Figure 3: Example of a grid where both particles are moving

2.5 Discussions and Conclusions

Judging on how MATLAB processed this second problem, this seems to be an inefficient method at receiving the approximate answer. 5000 trials was nearly impossible for the computer to process so I had to reduce the amount of trials to 500 to prevent the computer from crashing. This will be an external factor that skews the results of my data,

- Based on the results of both trials, it is comparatively better for both parties to be moving while separated from one another. The average number of steps needed if only one party was moving was 96 steps in a trial of 500, while this value was only 67 when both parties were moving. Of course, these results are only useful in this ideal situation as there are many more factors in a real life scenario. Although actual values are not present, the amount of time needed to process the second situation was faster than the first, supporting that both parties moving are more efficient.
- If the starting location were changed, say if both parties started at (-3,0) and (3,0) respectively, the results are below.

  `Median = 53.00` (Result for Situation 1)

  `Median = 81.00` (Result for Situation 2)
- In this case, the situation switches where both parties moving is a more inefficient method than one staying put. This may be because it is easier for a moving particle to cover the surrounding proximity and collide if the other particle was in close vicinity.