

RMIT University

School of Engineering

EEET2248 – Electrical Engineering Analysis

Group Lectorial Task 2

Birthday Paradox

Lecturer: Dr. Katrina Neville

Student Names: Nathan Williams, Jake Weiner, Brendon Vo, Thư Nguyễnn, James Howard

Student Number: s3707244, s3720634, s3717676, s3654756, s3287047

Submission Due Date: 1st June 2018

**Problem Statement**

Create a program that outputs a pseudo probability of two people in a room having matching birthdays with a varying number of participants based on the results of a simulation. Output these results graphically featuring a line of best fit.

**Input Data**

* Days in the year
* Number of people
* Number of simulations
* Formula: In(|f(x) – 1|) = - αx^2 + In(|A|)
* Value of A
* Value of α

**Output Data**

* Plot displaying probability vs number of people
* Curve of best fit
* Probability of having a matching birthday
* Linearized values for probability

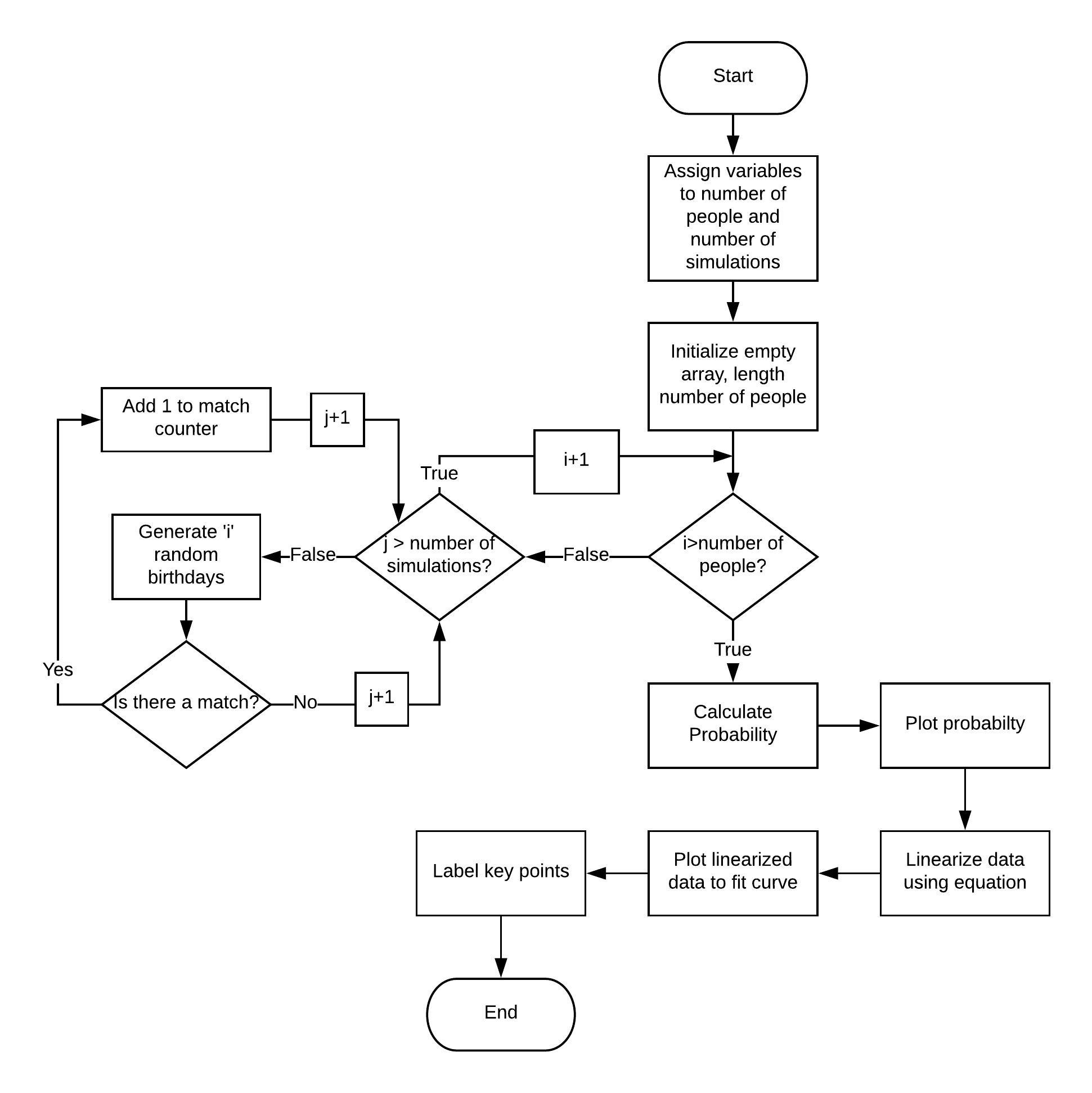
**Design**

Figure 1: Flowchart showing how the program solves the problem.

**Design Algorithm**

Our program design was based on maximising efficiency and creating easily interpretable results. First our program begins by clearing variables and CLI. Then variables are set for the max number of people to simulate and the amount of simulations to run per scenario (i.e. per number of people). Then an array of zeros “matchArray” is initialised of length equal to the numPeople variable just set. Then we have a for loop that iterates through to numSims that is nested in a loop running to numPeople. Inside the innermost loop an array “birthdays” of length equal to the counter is generated that is filled with numbers between 1-365 representing a day of the year (i.e. a birthday). Then the length of the birthdays array is compared to the length of the output of the unique function for the birthdays array. If they’re not equal this means there is a matching birthday “in the room” and the matchArray is incremented at the index corresponding to the amount of people in that simulation.

After the loops have resolved an array probability is initialised to equal the matchArray divided by numSims. This is plotted against an x vector from 1 to numPeople. A line of best fit is found by using the formula supplied to scale the probability from 0 to the initial instance of 100% probability. Once scaled, polyfit is used to find coefficients, these are put back into the equation and plotted.

Then features such as text are added to the plot to highlight data points where probability initially reaches 50% and 99% respectively. Our initial design used horizontal lines across the graph to mark this however using text and markers appears much more user friendly. A feature that could be added in future would be to output the amount of matches in each simulation and provide other extra data for the user to interpret.

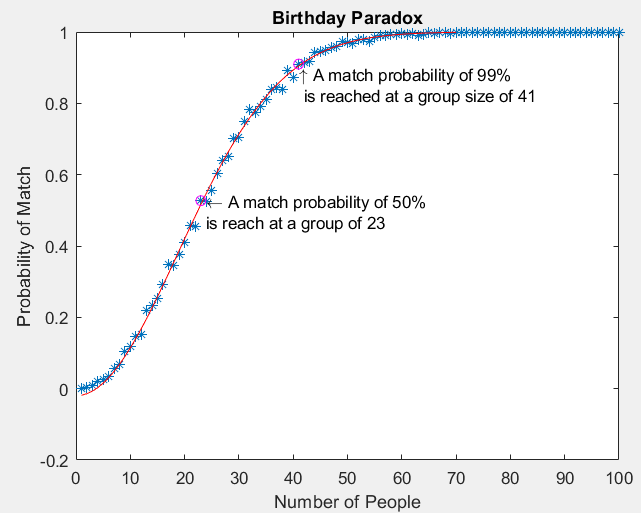
**Solution Testing**

Figure 2: Graph when program runs with 1000 simulations.

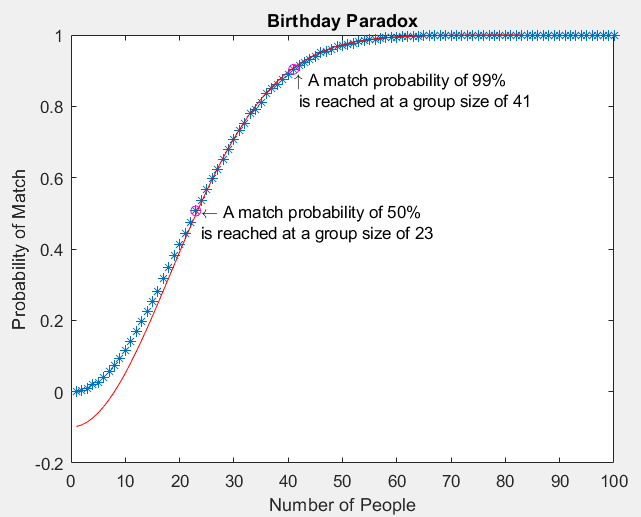
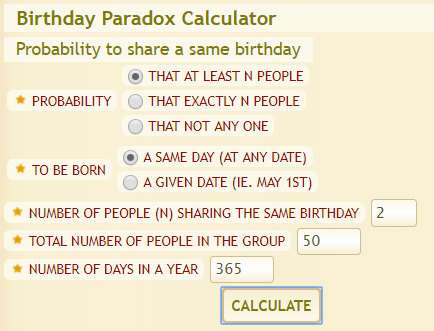


Figure 3: Graph when program runs with 30000 simulations.

There is a marginal amount of variation among the results each time the program is run. This is because the numbers are randomly generated producing slightly different results each simulation. However, as the number of simulations increase, the results become more consistent. Figure 2 shows the program running with 1000 simulations and Figure 3 shows the program running with 30000 simulations. Compare the line of best fits relevant to the data points. The points and curve of best fit on Figure 2 aren’t as smooth as they are on Figure 3, which shows evidence that the program is more accurate the higher the amount of simulation. Both figures also show an arrow with text next to it that indicate which group size the probability first reaches 50% and 99%. Both Figure 2 and Figure 3 shows the probability first hits 50% at the group size 23 people and hit probability 99% at the group size 41 people. This shows that choosing 1000 simulations is a very well estimated amount of simulations to use since it has the same result for 30000 simulations, which is expected to have a higher level of accuracy than 1000 simulations.

To test for the solution of the program, our program results can be compared to an existing program from the internet. The program can give the probability of having a birthday match in a specified group of people. This can be compared to the result the command ‘matchArray(x)/1000’, x being number of people in the group. The answer to the command will be compared to the answer on the existing program online, however the probabilities won’t be completely accurate due to the MATLAB program generating different birthdays each run.



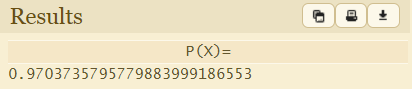


Figure 5: Online program showing the probability of at least 2 people having a birthday match out of a group of 50 people.

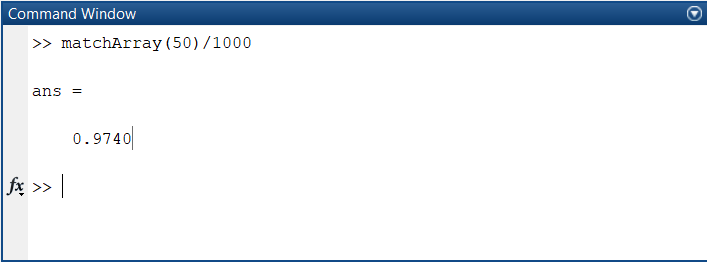


Figure 6: MATLAB program showing the results of the command for 50 people.

The two results as expected aren’t completely accurate since the MATLAB probability is the result of a simulation. Compared to the online probability application [1] which outputs a result based on mathematical probability. Considering that the two probabilities still managed to be very similar, 0.974 compared to 0.9703, shows evidence that the MATLAB program is accurate.

**Reference List**

[1] "DCODE group". (2018, 30/05/2018). *Birthday Probabilities*. Available: <https://www.dcode.fr/birthday-problem>

**Code Appendix**

%%variables

numPeople = 100;

numSims = 1000;

%%script

matchArray = zeros(1, numPeople);

for i = 1:1:numPeople

for j = 1:1:numSims

birthdays = randi(365, 1, i);

if (length(birthdays) ~= length(unique(birthdays)))

matchArray(1, i) = matchArray(1, i) + 1;

end

end

end

x = 1:1:numPeople;

probability = (matchArray/numSims);

figure (1)

plot(x, probability)

xlabel('Number of People')

ylabel('Probability of Match')

title('Birthday Paradox)

hold on

f\_lim = find(probability == 1);

x = 1:1:f\_lim(1) - 1;

f = probability(1:f\_lim(1) - 1);

f\_scaled = log(abs(f - 1));

coeffs = polyfit((x.^2), f\_scaled, 1);

t = 1 - exp(coeffs(2)) \* exp(coeffs(1) \* x.^2);

plot(x, t, 'r')

b = probability >= 0.5;

g50 = (length(probability)) - length(probability(b)) + 1;

c = probability >= 0.9;

g99 = (length(probability)) - length(probability(c)) + 1;

txt50 = sprintf('\n \\leftarrow A match probability of 50%%\n is reached at a group size of %1.0f', g50);

text(g50, probability(g50), txt50)

plot(g50, probability(g50), 'mo')

txt99 = sprintf('\n\n\\uparrow A match probability of 99%%\n is reached at a group size of %1.0f' , g99);

plot(g99,probability(g99), 'mo')

text(g99,probability(g99), txt99)

hold off