

**RMIT University**  
**School of Engineering**  
**EEET2248 – Electrical Engineering Analysis Lectorial Report**  
**Lecturer: Dr. Katrina Neville**  
**Student Names: Lakshay Dawar; Aiden Contini; Chris Iannazzone**  
**Student Numbers: 3712960; 3780445; 3785872**  
**Lectorial Session: Monday 10:30am      Submission Due Date:**  
**June 7th**

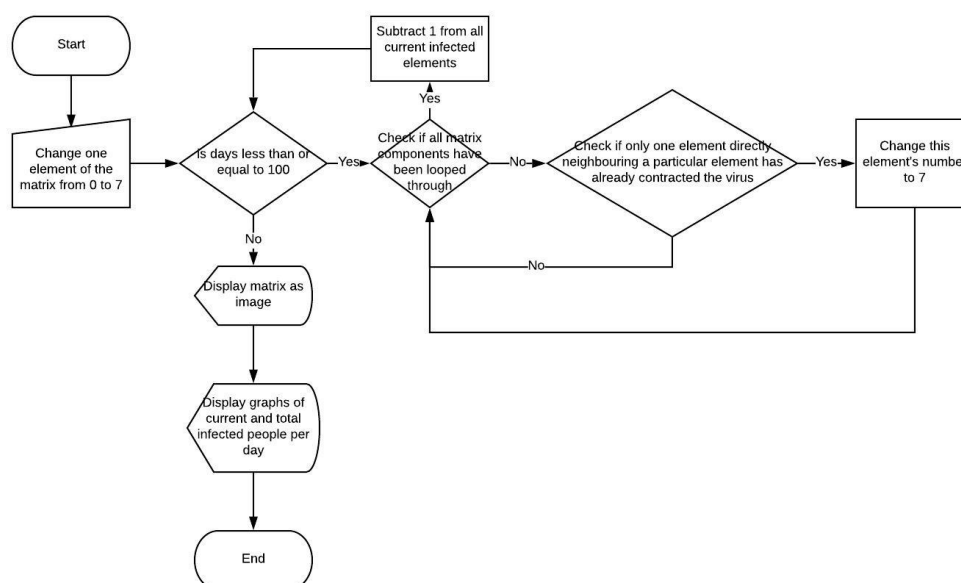
## Part 1: Zombie apocalypse task

### Problem solving steps

This problem involved students being required to create a fractal pattern through use of loops, over a period of 100 days, to simulate a zombie apocalypse. In order to achieve this, students were given guidelines on what the fractal pattern should look like, as well as dimensions that the matrix should be (how many people should be in the population). The amount of days for which the apocalypse could run for is variable, however 100 days was given as the default.

The logic for forming the pattern was also given, whereby the concept was that if an individual array element had no more than one infected element neighbouring it, then it would also be infected. Through the introduction of safety, this included the rule that each person in the array would become free of the virus after 7 days. There was also an option to perform the zombie apocalypse with or without immunity, and in this case it would mean that once a person contracted the virus, they were then immune to getting it again event after they had become free of it.

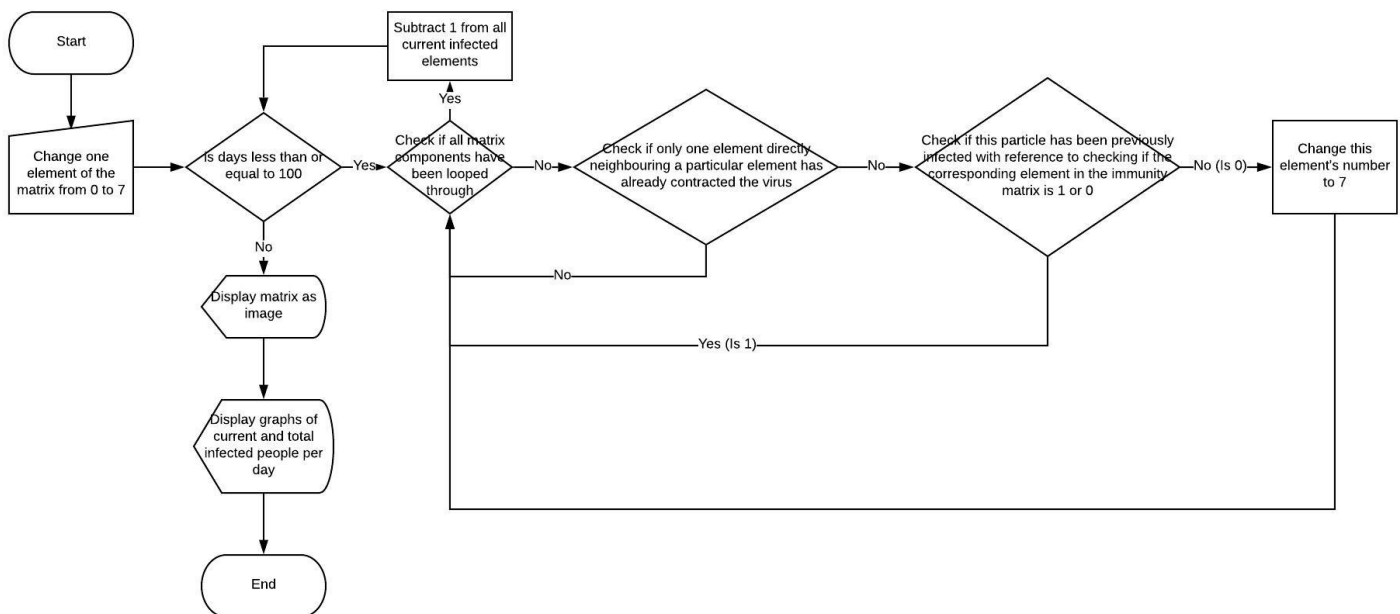
In order to complete the problem, the requirement was to output an animated fractal pattern which shows the spread of the virus progressively over the number of days specified. After this animation runs, two graphs should also be outputted, showing the number of current infected people on each day, as well as showing the number of newly infected people each day. These graphs should give an overall view of the spread of the virus over the period of time specified. There should also be an option at the start to be able to choose whether to view the simulation with or without the immunity rule, and hence the same outputs should be present for both versions of the simulation.



**Figure 1.1** - Flow chart of script logic without immunity

Above is a flow chart (figure 1.1) which describes the logic involved in completing the fractal pattern without immunity. Figure 1.2 shows a similar flow chart, however this one includes the logic for immunity to occur.

**Figure 1.2 - Flow chart of script logic with immunity**



### Design algorithms:

#### Without immunity:

Without immunity, the flow chart above shows that the random placement of the infected patient is selected, and this matrix is then passed through to the “fractal” function. The fractal function then loops through the matrix and returns a matrix containing the array elements of which positions in the main matrix need to be infected. The main script then uses this array, and infects these elements accordingly. As infection is done by setting the array elements from 0 (not infected) to 7 (first day of infection), the last part of the script then subtracts 1 from all infected elements, to ensure that after 7 days, any infected person becomes free of the virus (back to 0). This then loops over for however many days specified, and finally outputs the graphical results (which have been kept neatly in an array which updates on each iteration of the loop).

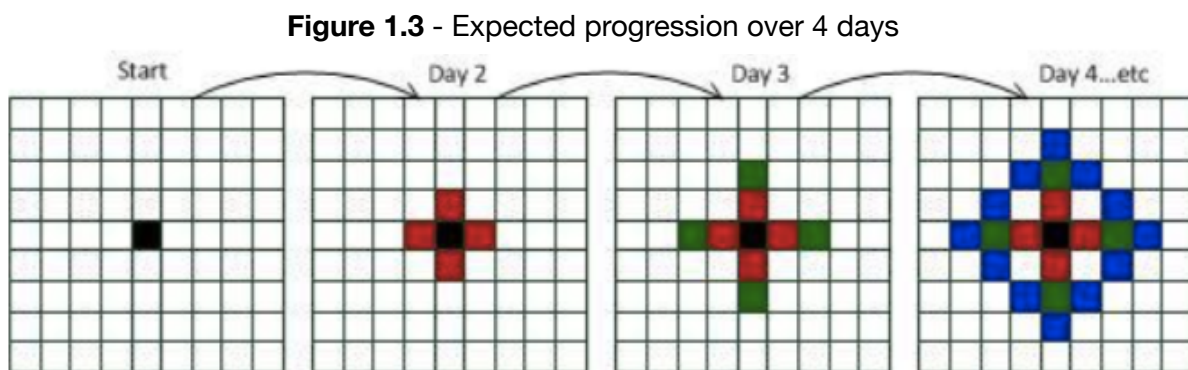
#### With immunity:

With immunity, the flow chart works very similar, however there is also an “immunity matrix” which must also be checked before any elements are changed. This is done by passing the initial matrix of elements to be changed into a second function, which then checks this against the immunity matrix, and creates a new array containing only the coordinates which need to be changed after considering the immune elements. This ensures that a matrix is kept up to date with the elements which have already been

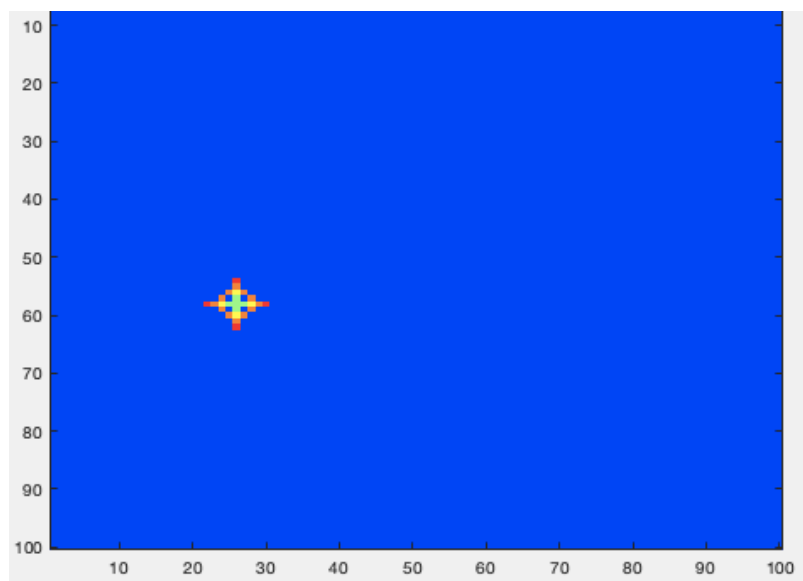
infected, so no person contracts the virus twice. After returning these coordinates, the relevant elements are changed to 7, and the script proceeds as above.

### Software solution and testing:

Figure 1.3 displayed the pattern given by the electoral task instructions on the expected layout of the fractal pattern over the first 4 days. Comparisons to day 4 of the Matlab output in figure 1.4 provides evidence for this to be working as expected, showing the first zombie pattern for a 4 day iteration.

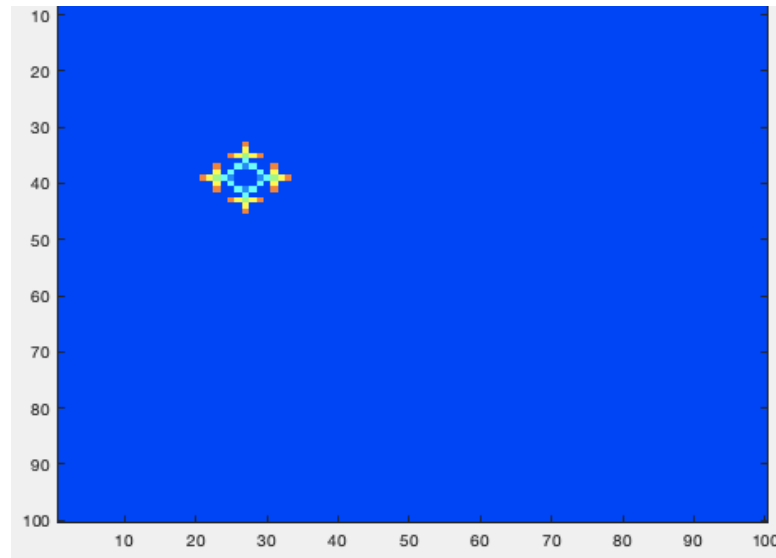


**Figure 1.4 - Non-immunity output of virus progression after 4 days**



The next part of testing involved testing the functionality of a person becoming virus free after 7 days. This was done by ensuring that the centre of the fractal pattern (the point where the fractal began) becomes the colour blue after 7 iterations. Figure 1.5 provides the evidence for this occurrence, as it can be seen that the very middle of the fractal in figure 1.5 has in fact become blue.

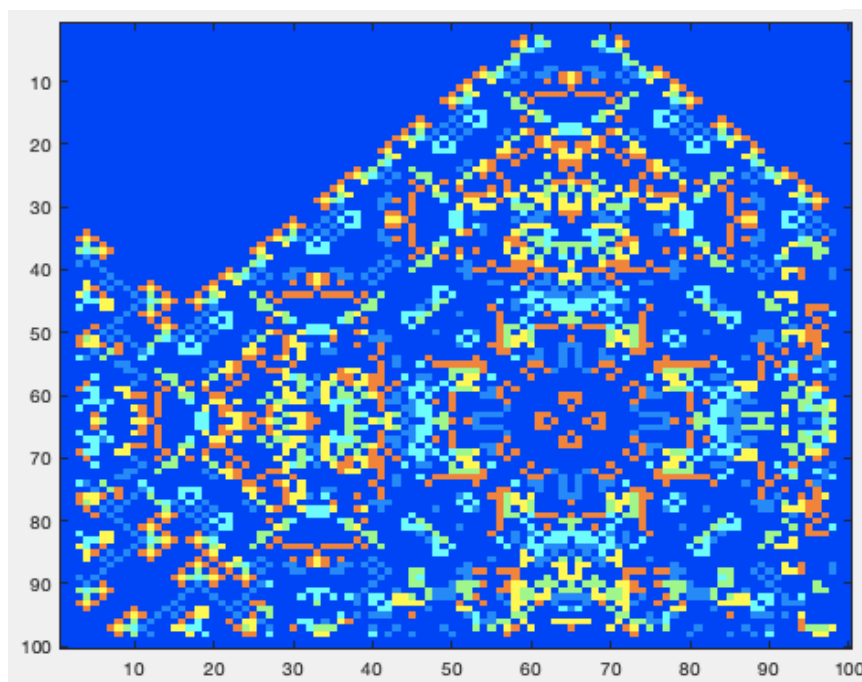
**Figure 1.5** - Non-immunity output of virus progression after 7 days



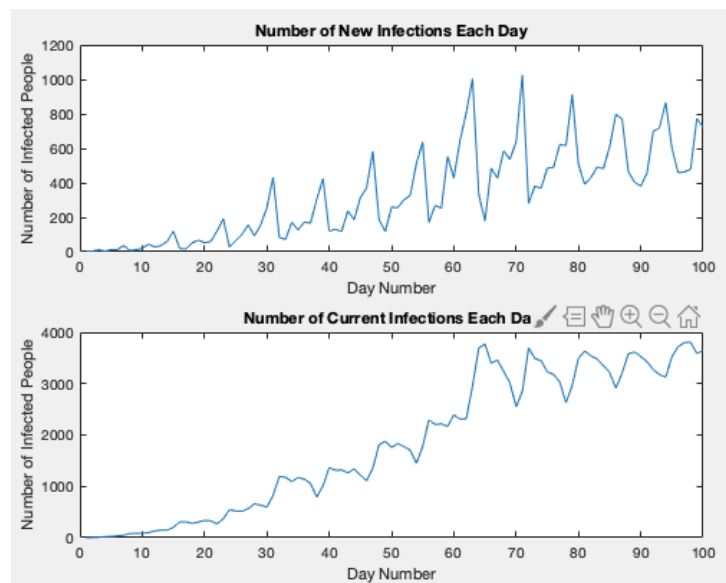
The next stage of testing involved ensuring the fractal was able to work over a period of 100 days as expected. In order to achieve this, the loop was set to 100, and the resulting outputted pattern was recorded in figure 1.6. As figure 1.6 reported no crashes, the fractal pattern was seen to work over a period of 100 days effectively.

Figure 1.7 shows the graphical results of the virus infection shown in figure 1.6. It can be seen from this that the number of current infections begins at 1 on day 1, and gradually climbs towards 4000 over the period of 100 days. Similarly, on day 2, the number of new infections recorded is 4, and this number of new infections climbs steadily as there is no immunity included in these iterations of the script.

**Figure 1.6** - Non-immunity output of virus progression after 100 days

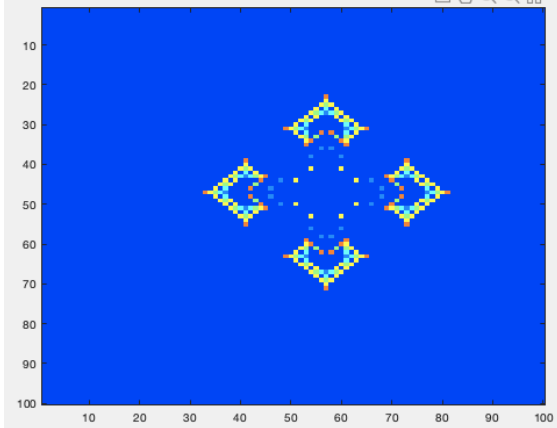
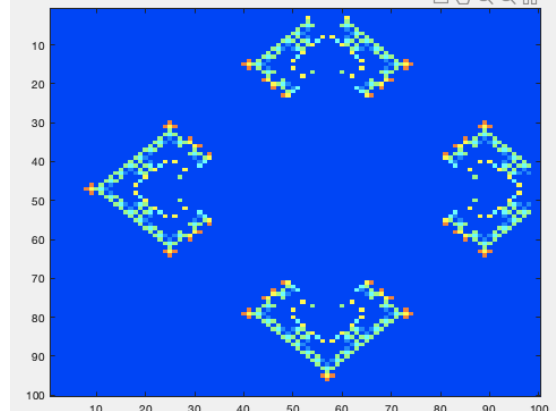


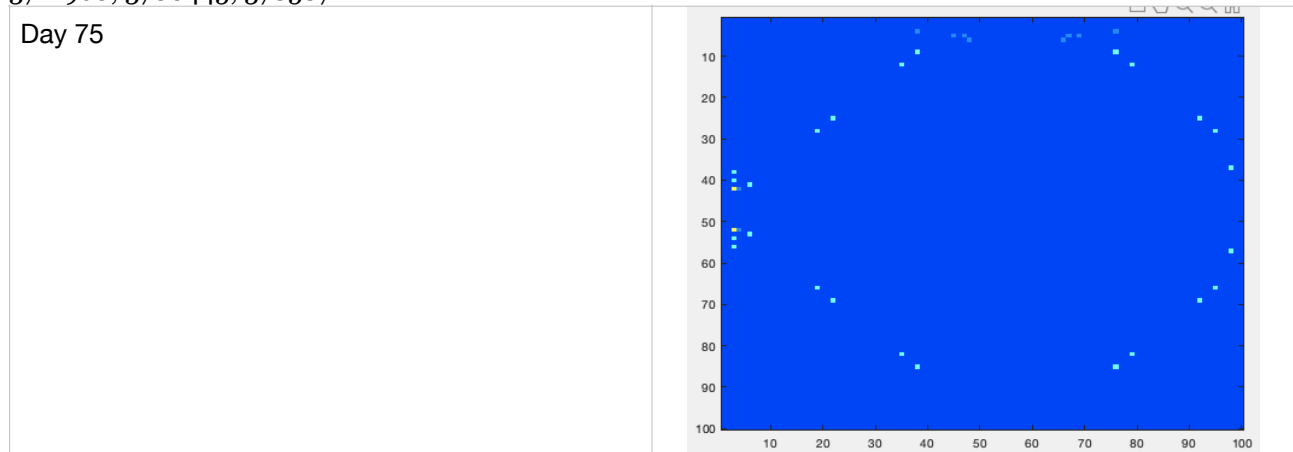
**Figure 1.7** - Non-immunity output of virus progression after 100 days



The final part of testing involved including the occurrence of immunity, and ensuring that this worked correctly. In testing this, the table in figure 1.8 displays the fractal pattern at days 25, 50 and 75, in order to emphasise that the same person has not been reinfected with the virus twice through the whole 100 day cycle (as the virus is completely cured before day 100, proof of day 100 was not included).

**Figure 1.8** - Immunity output of virus progression after given days on left

Day Number	Virus Spread
Day 25	
Day 50	



Finally, figure 1.9 provides the graphical output of this virus spread over the 100 days given in figure 1.8. It can be seen from this graph that the number of new infections each day is significantly lower than that of the virus spread with no immunity. It can also be seen that at day 70, both the number of new infections and current infections drops to almost 0, meaning that by day 70 the virus is almost completely cured from the population. Due to the presence of immunity, these results were expected, and hence the program can be seen to be functioning accurately as expected.

**Figure 1.9** - Immunity output of virus progression after 100 days

