***Project Report***

**Introduction**

The aim of the project is to produce a program that runs a simulation of an epidemic, for example, Coronavirus or Spanish flu. This being achieved using ordinary differential equations which define the rates of change within the population of the differing states found through an epidemic, such as infection to the point of hospitalisation, or death. The rate of transition through these stages are determined by user-inputted constants, relating to various properties of a virus/infection. The program is based upon the SIR epidemic model, which functions using 3 states (susceptible, infected, or recovered).

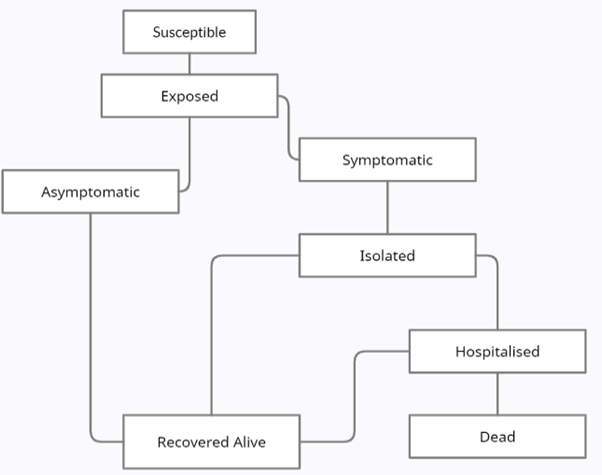
Our project aims to build upon said model and provide the user with a more comprehensive result. The program runs two versions of the simulation, Basic and Advanced, where the Basic simulation uses the same states as the SIR model, and the advanced uses 8 states, heavily focusing on the different stages of infection. This being to reflect the reality of the ongoing Coronavirus pandemic, as knowing which state of infection an individual is currently experiencing has been key to containing the virus.

Figure : Flowchart showing the state transition of modelled persons.

The flowchart above shows the state transitions for the advanced simulation. The assumption has been made that all transitions must follow as such. This allows the user to have significantly more control with the simulation. In turn, allowing for the creation of models of diseases with vastly differing traits.

**Analysis**

**Simulation Functions**

Basic SIR Model:

Using a classification system to allocate a “status” to the members of the subject population to represent what state they are in with respect to the virus. We can form equations for the rate of change of the number of people in each “status” which will depend on a variety of variables relating to the transmission of pathogens. When combined, A system of the equations for each state will give a concise and accurate representation of the virus’ progress

The model can be found at:

<https://www.maa.org/press/periodicals/loci/joma/the-sir-model-for-spread-of-disease-the-differential-equation-model>

Advanced SIR Model:

Using the same principals upon which the basic SIR model is built upon, we developed a new system of DEs to allow more states to be analysed in the simulation results. (See Fig 1). This required us to define a new set of constants which determine the transition rate from different states.

These can be found below:

------- Constants/Coefficients ---------

Format: Description of Constants (Dictionary Key) = variable name for use in functions

Variables involved in Susceptible to exposed

( Contacts = contacts sufficient to spread virus):

• Asymptomatic Number of Contacts per day

("AsympC") = a

• Symptomatic Number of Contacts per day

("SympC") = b

• Isolated Number of contacts per day

("IsoC") = c

• Hospitalized Number of contacts per day

("HosC") = f

Exposed to Asymptomatic / Symptomatic :

• Fraction of Exposed Population that become

asymptomatic infected per day

("EAsymp") = g

• Fraction of Exposed Population that become

symptomatic infected per day

("ESymp") = j

• Sum of g , j = Fraction of Exposed Population

that become infected each day

= k

Symptomatic to Isolation :

• Fraction of Symptomatic Population that become isolated

per day

("SympIso") = L

Isolation to Hospital :

• Fraction of Isolated population that become hospitalized

per day

("IsoHos") = m

Hospital to Dead:

• Fraction of Hospitalized population that die per day

("HosD") = p

Recovery Constants

• Fraction of Asymptomatic Population that recover each day

("AsympR") = q

• Fraction of isolated population that recover each day

("IsoR") = u

• Fraction of Hospitalized Population that recover each day

("HosR") = v

In order to incorporate these systems into our program, we decided to define two Classes. One for the basic model and one for the advanced. We then defined the system of equations for the model in a function named *deriv* which returned the system of equations in their time derivative form. The values for constants are class attributes and therefore the *deriv* function did not require these to be defined when constructing the system. Then using an ODE integration method from scipy library ( integrate class) the system of equations were integrated over the time period of the simulation using pre determined initial values. The value of the number of people in each state at each time interval is recorded and stored in a 1D array as a class attribute of the simulation. Each state has its own 1D array.

**Input Handler:**

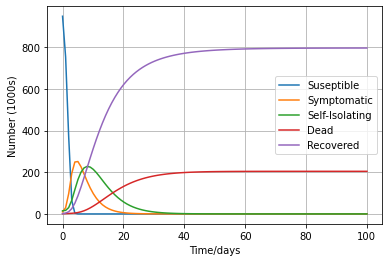
To obtain user defined inputs for the constants and initial state values, we used a command line prompt input which simultaneously checked any necessary conditions on the inputs.

Once the user has given all the necessary inputs for their chosen simulation version, the values are packed into an array. This array is then subsequently unpacked in other modules for use of the values within (e.g. in Simulator and WriteInputsToText))

**Plotting Module:**

First the formatting of the GUI is inputted (lines 26-27) since it will be the same for both potting functions. The basicplot() function is introduced on line 30, calling the BasicSimulation class as its argument. This is because this function is using the arrays created by said class to plot a graph.

Then the GUI options are set for the basic plot (line 36) and is launched (line 37). Once the user has selected their desired arrays, the function then plots these arrays onto the graph (lines 38-44).

****Then the function adds a legend to make the graph readable (line 46). The second function (line 50) has the same setup as the first, however calls the AdvancedSimuation class instead. Then the options for the advanced plot are set and the GUI is launched.

The user then choses their arrays and then these arrays are then plotted in the same way to the first function and a legend is added.

For example, in order to produce this graph, the user will chose to have an advanced graph. Then, when they are presented with the GUI, select the options: Susceptible, Symptomatic, Self-Isolating, Dead and Recovered. Then the code will plot these lines on the graph for the user to see.

**Conclusion:**

Our simulation achieves the task of creating a sufficient and customisable model for the spread of an infection in a population.

Further work could have been done to improve user quality of life, regarding their altering of constants. Currently no option is available to enter values for only specific constants, one must either enter values for all or use all the default values. A menu which enabled the user to jump to the desired constants and alter only the ones they wanted, could ease this issue.

We could have fixed the need to enter the inputs as decimals to prevent type error. If we did, the user could have input their desired inputs as fractions for increased usability.

Other models of simulations were considered, however, for the purposes of this task an expanded SIR model was chosen. We made this decision for ease of user customisability of the constants that determine the simulation.

Our project focused on the transitioning of states of members of the test population. In term assumptions were made to make this more readable. For example, a person (in the simulation) is unable to transition from the symptomatic state to the dead state, without first moving to an isolated and then Hospitalised state. This is even though in reality this is entirely possible. The user is able to account for this, however, through use of averaging out certain rate of state change constants to mimic the impossible state transitions.