MATLAB PROJECT - PHASE ONE

A Multi-Scale Model of Influenza

Authors:
Michael Boemo
Lukas Hutter
Beth McMillan
Noemi Picco
Alex Saunders
Huw Colin York

January 14, 2013

1 Introduction

Influenza A is a virus that affects various species, including humans. Human infection with Influenza A is characterised by the sudden onset of a high temperature, nasal congestion, cough, headache and loss of appetite [1]. Influenza epidemics are relatively common and have high mortality rates - for example, in 1918-19 the so-called "Spanish Flu" caused approximately 50 million deaths worldwide [2].

Influenza is has an RNA genome and is enclosed inside a membrane envelope [3]. Different strains of influenza are identified by the glycoproteins embedded in the outer membrane (hemagglutinin (H) and neuraminidase (N)) which each have several subtypes. The strain of swine flu that caused outbreaks across the world in 2009 was identified as H1N1 [4].

This report aims to document how the model and results of the Hancioglu *et al.* were recreated using the computer algebra system, MatLab. The report also outlines developments made to the model used by Hancioglu *et al.*.

1.1 Infection

The epithelial cells of the respiratory tract are the target of the Influenza A virus. The virus infiltrates host cells by using hemagglutinin to bind to cell surface receptors. After using new virus particles are created by replication by the host machinery, they are released from the cell by the action of neuraminidase, killing the cell and allowing the viral particles to infect other cells.

1.2 Antigen-presenting cells

Certain types of cell (including macrophages, B-cells and epithelial cells) can present viral antigens on their surface. Antigen-presenting cells take up proteins from virus particles and lysed infected cells and present them as part of a major histocompatibility complex.

1.3 Adaptive immunity

Antigen-presenting cells cause virus-specific plasma cells to proliferate. These cells produce specific antibodies to parts of the virus. Antibodies to the hemagglutinin in the viral coat prevent the virus from infecting cells, and antibodies that bind to neuraminidase prevent it from promoting release of viral particles from infected cells.

1.4 Interferon

Interferons α and β confer resistance to the cold virus to healthy cells, and cause the body to raise its temperature. Cells that have been infected by the virus, and antigen-presenting cells, will begin to secrete interferons α and β, thus preventing the virus from spreading further.

1.5 Effector cells

Another method used by the innate immune system to prevent the spread of the virus is the destruction of infected cells by cytotoxic T-cells and natural killer cells. These cells are stimulated by the antigen-presenting cells

2 Hancioglu *et al.*, 2007 model

$$\frac{dV}{dt} = \gamma_V I - \gamma_{VA} SAV - \gamma_{VH} HV - \alpha_V V - \frac{a_{V1} V}{1 + a_{V2} V} \tag{1}$$

V = viral load; γ_V = rate constant for production of virus by infected cells; I = infected cells; $\gamma_V A$ = rate constant for antibodies binding to virus particles; S = compatibility between antibodies and virus strain; A = antibodies; γ_{VH} = rate constant for virus entering infected cells; H = healthy cells; α_V, a_{v1}, a_{v2} = rate constants for physical removal of virus particles.

The change in viral load per epithelial cell is increased by the production of virus particles by infected cells. Virus particles are removed by antibody inactivation, by infecting healthy cells and by non-specific removal and degradation.

$$\frac{dH}{dt} = b_{HD} D(H + R) + a_R R - \gamma_{HV} VH - b_{HF} FH \tag{2}$$

b_{HD} = rate constant for epithelial cell division; D = dead cells; R = resistant cells; a_R = rate constant for cells losing resistance γ_{HV} = rate constant for infection; b_{HF} = rate constant for creation of resistant cells; F = interferon concentration;

Healthy cell proportion is increased by proliferation of healthy and resistant cells, as well as loss of resistance by resistant cells, and decreased by infection and resistance.

$$\frac{dI}{dt} = \gamma_{HV} VH - b_{IE} EI - a_I I \tag{3}$$

b_{IE} = rate constant for damage to infected cells by cytotoxic T cells; E = effector cells; a_I = rate constant for cytopathicity caused by the virus.

The proportion of infected cells is increased by infection of healthy cells, and decreased by destruction by effector cells and cytotoxic effects caused by the virus.

$$\frac{dM}{dt} = (b_{MD}D + b_{MV}V)(1 - M) - a_M M \tag{4}$$

M = antigen presenting cells; b_{MD} = rate constant for cells to take up and present antigens from dead cells; b_{MV} = rate constant for cells to take up and present antigens from virus particles; a_M = rate cosntant for antigen presenting cells ceasing to present antigens.

Antigen-producing cells take up viral proteins from dead cells and from virus particles. Over time, they naturally lose their antigen-presenting state.

$$\frac{dF}{dt} = b_F M + c_F I - b_{FH} HF - a_F F \tag{5}$$

b_F = rate of interferon production per antigen-presenting cell; c_F = rate of interferon production per infected cell; b_{FH} = rate constant for interferon binding to epithelial cells; a_F = rate constant for interferon decay;

Interferon is produced by antigen-presenting cells and by infected cells. It is lost by binding to healthy cells and to natural decay.

$$\frac{dR}{dt} = b_{HF} FH - a_R R \tag{6}$$

Resistant cells are produced when interferon binds to healthy cells. Cells lose their resistant state over time.

$$\frac{dE}{dt} = b_{EM} ME - b_{EI} IE + a_E(1 - E) \tag{7}$$

b_{EM} = rate constant for stimulation of effector cells by antigen-producing cells; b_{EI} = rate constant for lysis of infected cells; a_E = rate constant for natural death of effector cells.

Effector cells are stimulated by antigen-presenting cells. They are depleted after interactions with infected cells and are maintained at numbers within strict bounds in the body by homeostasis.

$$\frac{dP}{dt} = b_{PM} MP + a_P(1 - P) \tag{8}$$

P = antibody-producing plasma cells; b_{PM} = rate constant for production of plasma cells; a_P = rate constant for the natural death of plasma cells.

The number of antibody-producing plasma cells depends on their stimulation by antigen-presenting cells, and is maintained within strict bounds in the body by homeostasis.

$$\frac{dA}{dt} = b_A P - \gamma_{AV} SAV - a_A A \tag{9}$$

b_A = rate of antibody production per plasma cell; γ_{AV} = rate constant for antibody binding to virus particles; a_A = rate constant for the natural destruction of antibodies;

Antibody concentration depends on the production of antibodies by the antibody-producing plasma cells, the binding of antibodies to virus particles and the natural decay of antibodies in the body.

$$\frac{dS}{dt} = rP(1 - S) \tag{10}$$

r = rate constant for S

S is a measure of the immune memory of the host. Hosts with high values of S produce antibodies with a high binding affinity to the virus particles. The value of S depends on the production of antibodies by the plasma cells.

$$D = 1 - H - R - I \tag{11}$$

The dead cells are the cells in the system that are neither healthy, resistant or infected.

2.1 Scope and Impact of the Hancioglu Model

While influenza at population level has been extensively studied [5], there are only about a dozen non-linear models for simulating a host's immune response to an infection with influenza A, among which the Hancioglu model is generally referred to as one of the more complex. [6, 7, 8, 9]

It is discussed in a variety of papers concerned with influenza epidemiology, as well as studies of inflammation[10, 11, 12, 13, 14] and constitutes the foundation for a modelling study on antigenic sin [15].

In a 2011 review [16] however the paper is criticised because of over-parametrisation and consequentially its lack of parameter identifyability and limited possibilities for model validation. Among the non-linear intra-host models for influenza A infection, the much more tractable, data -driven model [17] seems to be much more popular and is in fact used in various multi-scale approaches targeted at modelling population dynamics of influenza based on a intra-host model [18].

Albeit being explicitly defined as an objective of the Hancioglu model, its implementation in a spatial population model, as presented in this report, constitutes a new approach.

2.2 Reproducing the Hancioglu paper's result

As a starting point the model in Hancioglu *et al.* was reproduced, merely based on equations and parameters given in the paper, [2]. The system of

ODEs given by equations (1), (2), (3), (4), (5), (6), (7), (8), (9), (10) can be readily solved using the Matlab routine `ode45`. The values for parameters and initial conditions were set to be the same that the authors used. The function `plot_graphs_paper` allows to recreate and plot the results shown in Figure 2 of [2]. In Figure 1 results of our own simulations are shown.

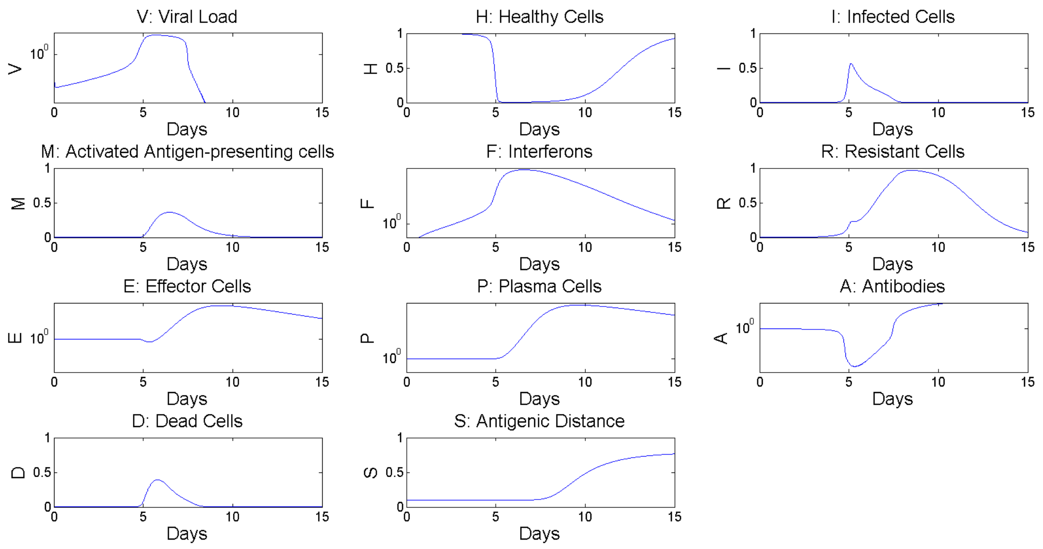


Figure 1: Results of simulations for the ODE system in Hancioglu 2007, [2].

3 Multiscale Model

Starting from the model of the immune response of individuals, it is possible to investigate the problem at a population level. The system of ODEs previously introduced describes the evolution of different cell types and components of the immune system. The next step is to integrate it into a higher scale model describing the geographic spread of the disease over time. We have approached this problem in two ways and have created two separate models: one continuous and one discrete.

4 Continuous Model

In this Section, we motivate our discrete model with a continuous one. We first make two strong assumptions: the duration of the disease is quite short, and there is no immunity (individuals who come in contact with infected individuals will themselves be infected). In this case, the spread of an infectious disease is a wave. Certain members of a population become

infected, and will pass the disease to those around them. Hence, the wave begins at a source (a "patient zero") and ripples outward from this location. Because the duration of the disease is short, individuals near the source will recover or die as the wave of disease spreads. This idea is applicable to infectious diseases such as influenza: Infected patients either recover or die, and the duration of the disease is fairly short.
A wave can be modelled by the *wave equation*:

$$\frac{\partial^2 v}{\partial t^2} = c^2 \nabla^2 v$$

where $v(x, y, t)$ is the displacement in spacetime and c is the speed of the wave. It is important to note that c may be either a constant or a function. Here, we set c as a constant and v as the viral load of individuals at position (x, y) and time t.
Previously, we have assumed that there is no immunity. That is, healthy individuals that come into contact with infected individuals are guaranteed to become infected. Fix t and consider a heatmap representing infection on the surface $v(x, y, t)$. Plotting the heatmap on v, we would expect something similar to Figure 2. Here, we see that the number of infected individuals (coloured from blue to red) correlates perfectly with v.

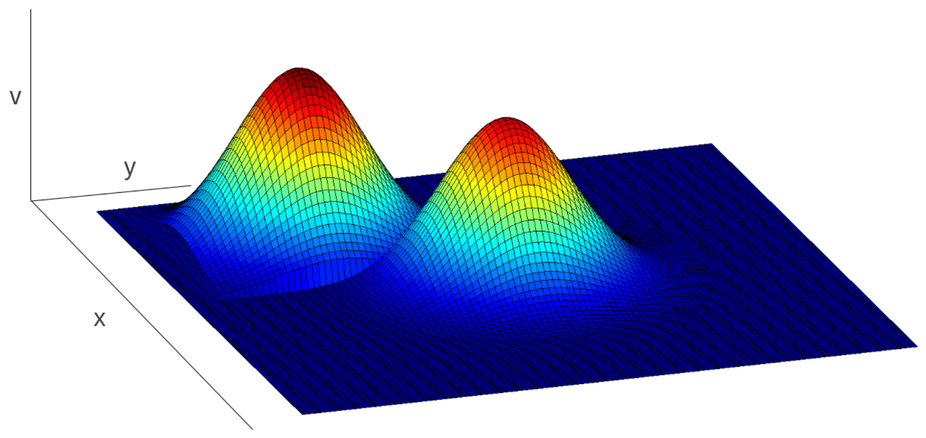


Figure 2: Infection colored from zero (blue) to high (red) at a fixed time t with two sources. Here, we see infection is high (orange-red) when viral load v is high.

If we consider immunity and resistance, we get a weaker correlation between infection and v. While the two are correlated, there are also regions where there is a high viral load and a relatively small number of infected individuals. In this case, we expect to see a plot similar to Figure 3.

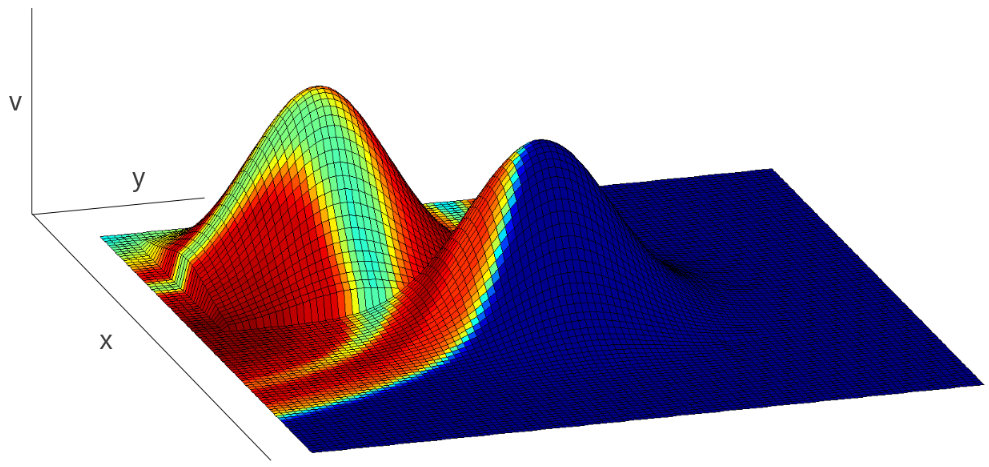


Figure 3: Infection at a fixed time t with two sources, allowing the possibility of resistance and immunity. Note how there are regions of medium to low infection (yellow-green) even when v is large.

While we neglect resistance and immunity in this model (hence producing plots similar to Figure 2) we recognise that their inclusion is an exciting possibility for expansion and improvement. The phenomenon of infectious diseases can certainly be modelled by solving the wave equation with suitable boundary conditions. In lieu of this approach, we have created a discrete model in order to add more adjustable parameters and make a more customisable, intuitive, and flexible user interface.

5 Discrete Model

The continuous model studies state variables of individuals (e.g. the viral burden V) as spatial distributions of their intensities. At each time t, the value of $V(x, y, t)$ represents the density distribution over space domain (x, y). In order to achieve an individual representation of the state variables, it is possible to introduce a discrete version of the coupled multiscale model. In order to do this, the space is discretised and each individual is considered to occupy a cell of the resulting grid. Hence $V(x_i, y_i, t)$ will represent the value of the state variable V for the individual at position (x_i, y_i) at time t. We consider a population of 100 individuals, distributed over a spatial region implemented as a square grid. Each individual is assigned a set of variables describing his state:

```
variable.V(x,y,t)
variable.H(x,y,t)
variable.I(x,y,t)
variable.M(x,y,t)
variable.F(x,y,t)
variable.R(x,y,t)
variable.E(x,y,t)
variable.P(x,y,t)
variable.A(x,y,t)
variable.S(x,y,t)
 healthy(x,y,t)
```

Where `x` and `y` represent his spatial position in the grid, and `t` the time at which the individual's state is considered.
The `variable`'s fields correspond to those previously introduced for the ODE system in [2]. `healthy` is a boolean state variable (1 if the individual is considered infected ($V > 0$)).
The undelying algorithm is made of two main parts:

1. For each individual:
   - Solve the ODE system to update `variable.[](x,y,t)`
   - Update `healthy(x,y,t)`
2. Interface with the population-scale model:
   - Evaluate infection that infected individuals will spread to their neighbours.
   - For each individual update `variable.V(x,y,t)`: these will provide initial values for `V` in the ODE system in 1 for time `t+1` .
3. Increment time, go back to 1.

The viral burden passed by the infected individuals to the eight neighbors is evaluated as the eighth part of his viral burden `V`.

5.1 Initial conditions and Parameters

`Variable.V` is initialised to different values depending on the health of infected condition of the individual.
The user will be able to set the initial infected individuals.
The function

```
[variable healthy] = Coupled_Multiscale(
gridDim_input, coordinates, gammaV_input, endTime_input )
```

receives in input:

- the matrix `coordinates`. This is a $(2 \times N)$ matrix, where N is the number of initial infected individuals, whose x and y coordinates are on the first and second row of such matrix respectively.
- `gridDim`, will be set to 10 (allowing a simulation on 100 individuals).
- `gammaV_input`, sets the viral burden of the disease, allowing to simulate different degrees in the disease strength.
- `endTime_input`, sets the time interval the simulation will be run on.

All the other parameters required by the ODE system will be set to default values (see the script `SetParams`).

5.2 Optimisation

Once the outline of the code had been written and had been shown to reproduce results from the Hancioglu *et al.* paper, optimisation of the code was considerd.
Vectorisation and preallocation of variables `Coupled_Multiscale_vectorisation.m` produced barely noticable results (column 1, Table 1) as the vast majority of time (> 98 %) was being spent in solving the ODE system for each cell at each time point. The time could be reduced by only considering cells where the virus load was no longer zero by using an if statement. The computation time was futher reduced by using the *find* command to find all the cells where the virus load was above 0.0001 `Coupled_Multiscale_using_find.m`. This was an arbitrary small, non-zero number chosen to stop the inclusion of cells where the virus load was very small but non-zero - changing this value may affect the result however we postulated this affect would be minimal. This did not affect the time taken for each time step (column 2, Table 1).

In order to reduce the time to run a simulation, parallelisation was considered `Coupled_Multiscale_parallelised.m`. The different timesteps could not be parallelised as each one was dependent on the previous one however each cell in the grid of people was independent of one another hence a parfor loop could be implemented here. Early attempts to use a parfor loop were unsuccessful as an error regarding the variable containing the structure could not be recognised by the parfor function. It was found that parfor could not handle structures being passed in. In order to use parfor, the structure was transferred to a temporary vector containing the ten values from the structure (*V, H, M, I* etc). This would require copying the data to the temporary vector and back on each iteration however this overhead was likely to be small than the gain in being able to use multiple processors. Future work could be to remove the structure from the code and replace it as an array of size ten. This would reduce the copying required throughout the program however the structure was left in for human-readability reasons.

Time Step	Vectorisation	Using *find* function	Parallelisation
1	0.0757	0.0403	0.0293
2	0.1467	0.0997	0.1012
3	0.1121	0.2225	0.2273
4	0.0664	0.3738	0.3822
5	0.0547	0.5768	0.6032
6	0.1585	1.0853	1.1026
7	0.4691	2.3268	6.1066
8	2.3300	5.9191	6.1066
9	215.2908	583.6394	590.1720

Table 1: Time taken (in seconds) for each time step using the three optimised methods.

The code produced where the parfor is implemented does not contain the correct features for using the GUI due to time constraints however it would be easy to implement the changes at a later date. Using the three codes from optimisation, the time of completion for each timestep is shown below (Table 1).

5.3 Fast version

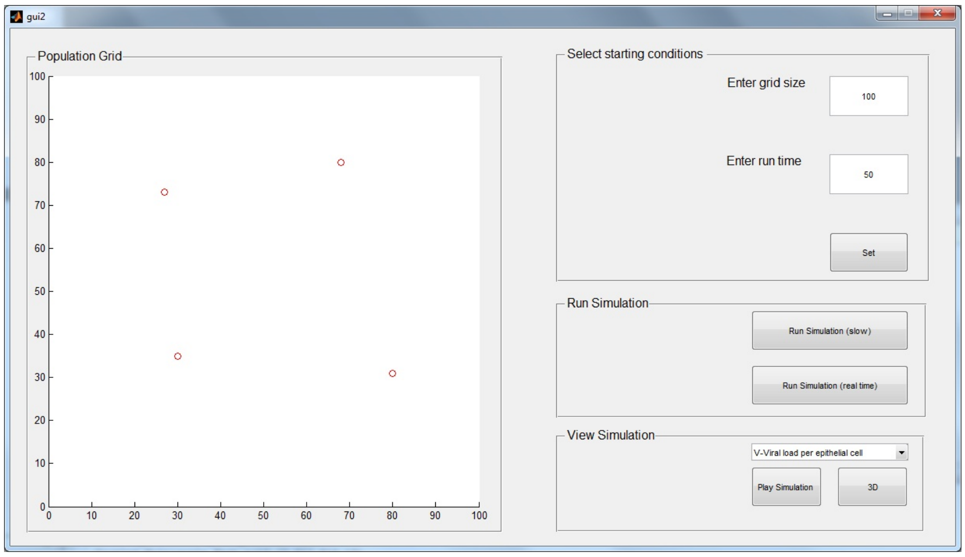
In an effort to improve the speed of the population simulation at the expense of a certain amount of tunability, we have created a faster version. In this faster version, the system of ODEs is dispensed with altogether. Instead, a vector containing the viral load (`variable.V` as in the original simulation) of an individual over time is used for every individual. This modification decreases the time taken to run the simulation by several orders of magnitude.

This vector was calculated from our implementation of the Hancioglu 2007 model over 50 time steps using one set of parameters. In order to run this faster simulation with different parameter values, it would be necessary to first create the vector of viral load values by running the `ODE_Function` script after modifying the `SetParams` file to include the desired values.

Unlike in the slower version of the simulation, the viral load administered to each individual when they are infected is the same, reducing the variability in infection across the population.

6 GUI Interface

A graphical user interface was implemented to allow the user to select both the population size and the duration of the simulation. An axis, displaying the population grid allows the user to select the location of the initial infected individual, or individuals. There is no limit on the number of individuals



Enter grid size 100

Enter run time 50

V-Viral load per epithelial cell

Figure 4: Screen capture of the graphical interface showing the selection of the gird size (100) and run time (50). Four individuals have been selected as infected on the axis

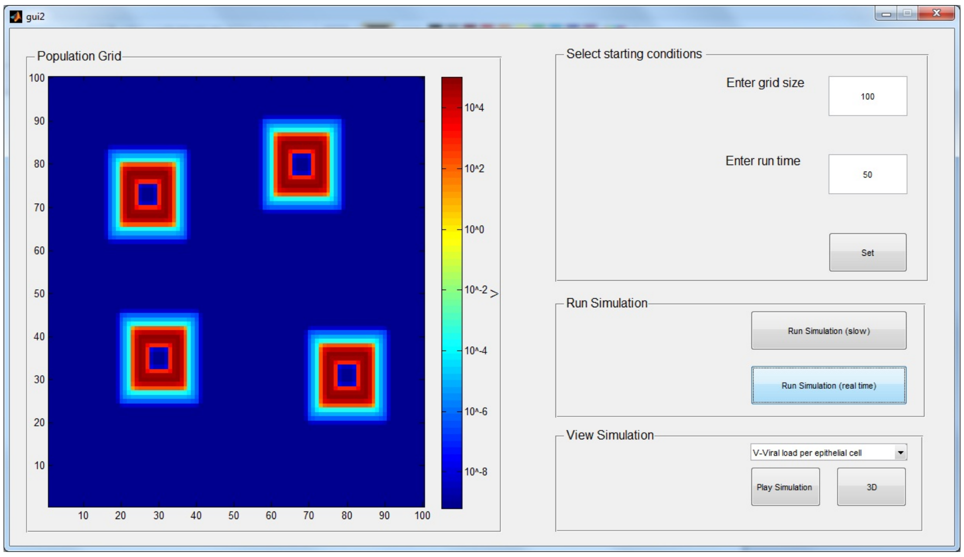
the user is able to select (see Figure 4). Once these initial parameters have been defined the user can choose to run simulation the simulation in two modes: Run Simulation (Realtime) and Run Simulation (slow).

6.1 Run Simulation (Realtime)

On clicking the button 'Run Simulation (Realtime)' the program runs `Coupled_Multiscale_Fast.m`. The simplification of `Coupled_Multiscale_Fast.m` permits the user to observe the simulation in real time. The figure used to select individuals now updates to display a heat map of the viral load (V) of each individual at each time step, allowing the user to observe the spread of the virus throughout the grid (see Figure 5). On completing, the user can then review the simulation in 2D (button 'Play simulation') or 3D where the z axis represents V and the colour maps represents the infected individuals (button '3D')(See Figure 6).

6.2 Run Simulation (Slow)

On clicking the button 'Run Simulation (Slow)' the program runs `Coupled_Multiscale.m`. The speed of the simulation prohibits a real time display therefor the GUI displays a waitbar displaying the progress of the simulation by time step. On completing the user is able to again view the simulation in both 2D and 3D.



Enter grid size: 100

Enter run time: 50

V-Viral load per epithelial cell

Figure 5: Screen capture of the graphical interface showing the progression of the viral load (V) throughout the population grid. The colour map is displayed on a log scale

A Glossary of terms

- **Adaptive immunity** - The mechanism by which an organism recognises, remembers and destroys specific pathogens.
- **Antibody** - A molecule of the immune system that binds to and immobilises specific threats.
- **Antigen** - A molecule that binds to an antibody.
- **B-cell** - A white blood cell that produces antibodies.
- **Cytotoxic T-cell** - A white blood cell that destroys infected cells.
- **Epithelial cell** - A type of cell that lives at the interface between tissue and body cavities.
- **Glycoprotein** - Proteins that are attached to carbohdyrates.
- **Innate immunity** - A broad non-specific immune response to a pathogen.
- **Interferon** - A type of protein that triggers the immune system.
- **Lysis** - Splitting of a cell.
- **Macrophage** - A type of immune cell that infiltrates tissues

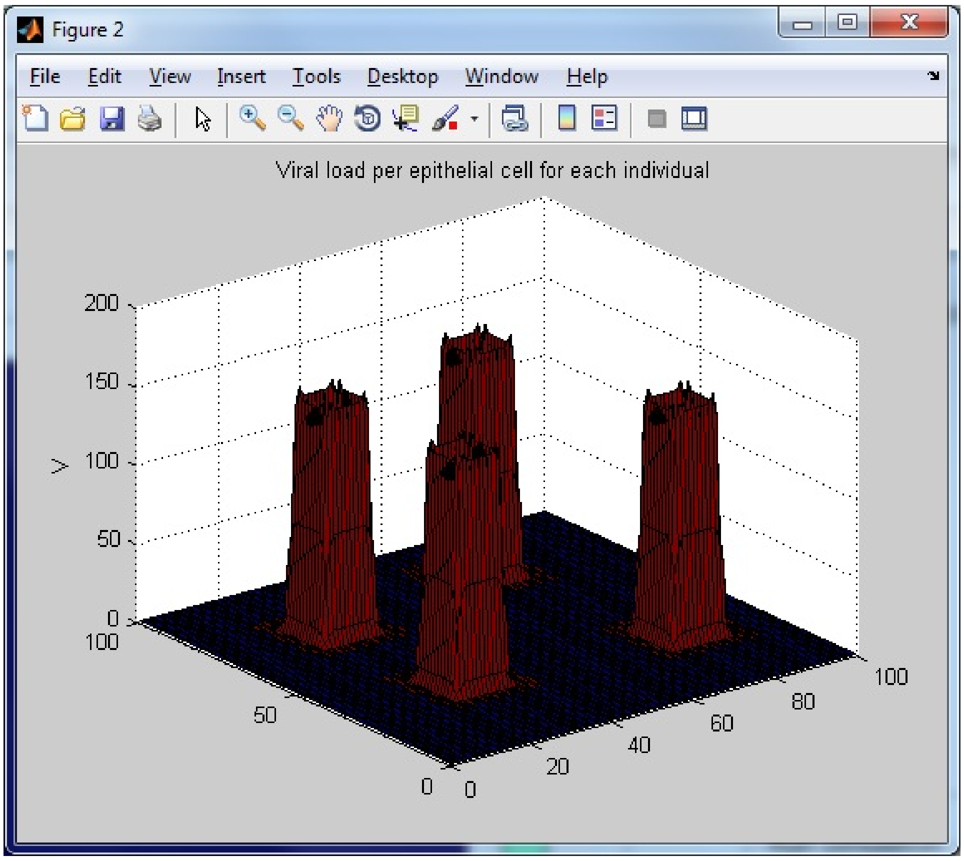


Figure 2
File Edit View Insert Tools Desktop Window Help
Viral load per epithelial cell for each individual
V
200
150
100
50
0
100
50
0
0
20
40
60
80
100

Figure 6: Screen capture of the 3D representation of the viral progression. Blue regions correspond to individuals free from infection and red regions contain infected individuals. The Z axis corresponds to the viral load V. The colour map is displayed on a log scale

- **Major histocompatibility complex** - A cell surface molecule that interacts with white blood cells
- **Natural killer cell** - A white blood cell that kills infected or stressed cells
- **Respiratory tract** - The nose, throat and lungs.
- **RNA** - Ribonucleic Acid. A type of molecule that carries genetic information.
- **Virus particle** - The intact virus, including the envelope, genome and associated proteins.

References

[1] A. Monto, S. Gravenstein, M. Elliott, M. Colopy, and J. Schweinle, "Clinical Signs and Symptoms Predicting Influenza Infection," *Archives of Internal Medicine*, vol. 160, pp. 3243–7, 2000.

[2] B. Hancioglu, D. Swigon, and G. Clermont, "A Dynamical Model of Human Immune Response to Influenza A Virus Infection," *Journal of Theoretical Biology*, vol. 246, pp. 70–86, 2007.

[3] N. Campbell, J. Reece, L. Mitchell, and M. Taylor, *Biology Concepts and Connections.* 2003.

[4] N. S.-O. I. A. H. V. I. Team, "Emergence of a Novel Swine-Origin Influenza A (H1N1) Virus in Humans," *The New England Journal of Medicine*, vol. 360, no. 25, pp. 2605–15, 2009.

[5] R. A. Saenz, M. Quinlivan, D. Elton, S. Macrae, A. S. Blunden, J. A. Mumford, J. M. Daly, P. Digard, A. Cullinane, B. T. Grenfell, J. W. McCauley, J. L. N. Wood, and J. R. Gog, "Dynamics of influenza virus infection and pathology," *Journal of virology*, vol. 84, no. 8, pp. 3974–3983, 2010.

[6] S. Daun and G. Clermont, "In silico modeling in infectious disease," *Drug Discovery Today: Disease Models*, vol. 4, no. 3, pp. 117–122, 2007.

[7] E. L. Haseltine, V. Lam, J. Yin, and J. B. Rawlings, "Image-guided modeling of virus growth and spread," *Bulletin of mathematical biology*, vol. 70, no. 6, pp. 1730–1748, 2008.

[8] C. A. A. Beauchemin and A. Handel, "A review of mathematical models of influenza a infections within a host or cell culture: Lessons learned

and challenges ahead," *BMC Public Health*, vol. 11, no. SUPPL. 1, 2011.

[9] M. Delitala, P. Pucci, and M. C. Salvatori, "From methods of the mathematical kinetic theory for active particles to modeling virus mutations," *Mathematical Models and Methods in Applied Sciences*, vol. 21, no. SUPPL. 1, pp. 843–870, 2011.

[10] A. C. Paulo, M. Correia-Neves, T. Domingos, A. G. Murta, and J. Pedrosa, "Influenza infectious dose may explain the high mortality of the second and third wave of 1918 1919 influenza pandemic," *PLoS ONE*, vol. 5, no. 7, 2010.

[11] S. C. Chen, C. P. Chio, L. J. Jou, and C. M. Liao, "Viral kinetics and exhaled droplet size affect indoor transmission dynamics of influenza infection," *Indoor air*, vol. 19, no. 5, pp. 401–413, 2009.

[12] J. D. Mathews, J. M. Chesson, J. M. Mccaw, and J. Mcvernon, "Understanding influenza transmission, immunity and pandemic threats," *Influenza and other Respiratory Viruses*, vol. 3, no. 4, pp. 143–149, 2009.

[13] Y. Vodovotz, G. Constantine, J. Faeder, Q. Mi, J. Rubin, J. Bartels, J. Sarkar, R. H. Squires Jr., D. O. Okonkwo, J. Gerlach, R. Zamora, S. Luckhart, B. Ermentrout, and G. An, "Translational systems approaches to the biology of inflammation and healing," *Immunopharmacology and immunotoxicology*, vol. 32, no. 2, pp. 181–195, 2010.

[14] S. C. Chen, S. H. You, C. Y. Liu, C. P. Chio, and C. M. Liao, "Using experimental human influenza infections to validate a viral dynamic model and the implications for prediction," *Epidemiology and infection*, vol. 140, no. 9, pp. 1557–1568, 2012.

[15] K. Pan, "Understanding original antigenic sin in influenza with a dynamical system," *PLoS ONE*, vol. 6, no. 8, 2011.

[16] H. Miao, X. Xia, A. S. Perelson, and H. Wu, "On identifiability of nonlinear ode models and applications in viral dynamics," *SIAM Review*, vol. 53, no. 1, pp. 3–39, 2011.

[17] P. Baccam, C. Beauchemin, C. A. Macken, F. G. Hayden, and A. S. Perelson, "Kinetics of influenza a virus infection in humans," *Journal of virology*, vol. 80, no. 15, pp. 7590–7599, 2006.

[18] L. Canini and F. Carrat, "Population modeling of influenza a/h1n1 virus kinetics and symptom dynamics," *Journal of virology*, vol. 85, no. 6, pp. 2764–2770, 2011.