# Influenza Virus Infection Modeling

A. Ambuehl – antonietta.ambuehl@dtc.ox.ac.uk
J. Leem – jinwoo.leem@dtc.ox.ac.uk
M. Lucken – malte.lucken@dtc.ox.ac.uk
W. Smith – william.smith@dtc.ox.ac.uk
O. Thomas – owen.thomas@dtc.ox.ac.uk

University of Oxford DTC
Rex Richards Building
South Parks Road
Oxford, OX1 3QZ, United Kingdom

January 18, 2013

**Abstract**

# Chapter 1

# Background and Aims

## 1.1 Biological Problem and Previous Work

The influenza virus is responsible for considerable public discomfort, and poses the risk of seasonal pandemic and a potentially major health hazard to the elderly or infirm. [13] For the purposes of treating and preventing viral infections, it would be desirable to generate a model which simulates the course of viral infection. Models that study the interplay between immune system components within an individual suffering from viral infection have previously been developed (for a review, see Perelson 2002 [11]), and have the potenital to capture multiple aspects of human epithelial immunology (*e.g.* interaction of cytokines, effector cells, antibodies, virions, *etc.*).
In particular, a model was devised in 2007 which encompassed many of the factors involved in human-virus immune response [1], such as:

- Antibodies and their affinity toward the circulating virus;
- Antigen presentation;
- Production and clearance of the virus, *etc.*.

This paper of Hancioglu *et al.* featured a continuum model based on a series of ordinary differential equations (ODEs). The model was used predict changes in the population of healthy and infected cells, and levels of the free virus over the course of infection.

In 2013, the results of the model have been reproduced by members of Group G by using a series of Matlab® and C++ code [4]. This computational model successfully reproduces cellular and viral dynamics in a hypothetical infection scenario, displaying good fidelity. In addition, the authors provide an open-source implementation of the model in MatLab®, featuring a graphic user interface (GUI) that enables users to control and systematically parse the model's 27 parameters, as well as facilitating the visualisation of multiple datasets.

## 1.2 Identification of areas for extension

Although the model is designed so that parameters can be tailored to each patient's immune capacity and each virus strain's virulence, it only reflects the natural biological

response (*i.e.* the adaptive and innate immune responses) to the virus. Moreover, the model was entirely deterministic: real biological processes tend to involve considerable noise, and several of the processes included in the system are thought to be intrinsically stochastic. Finally, we noted that the mode omitted certain interactions between immune system components. Our group felt that the model could be extended further by:

1. Considering the effects of anti-influenza drugs on the cellular and viral dynamics;
2. Converting the coupled ODE representation to include stochastic noise;
3. Including the influence of helper T-cells ($T_H$ cells), which help to form plasma cells.

In addition, we scrutinised the code of their implementation closely, and investigated ways in which it could be rewritten in a more efficient manner. Ultimately, these extensions culminated to a new model as shown in Figure 1.2:

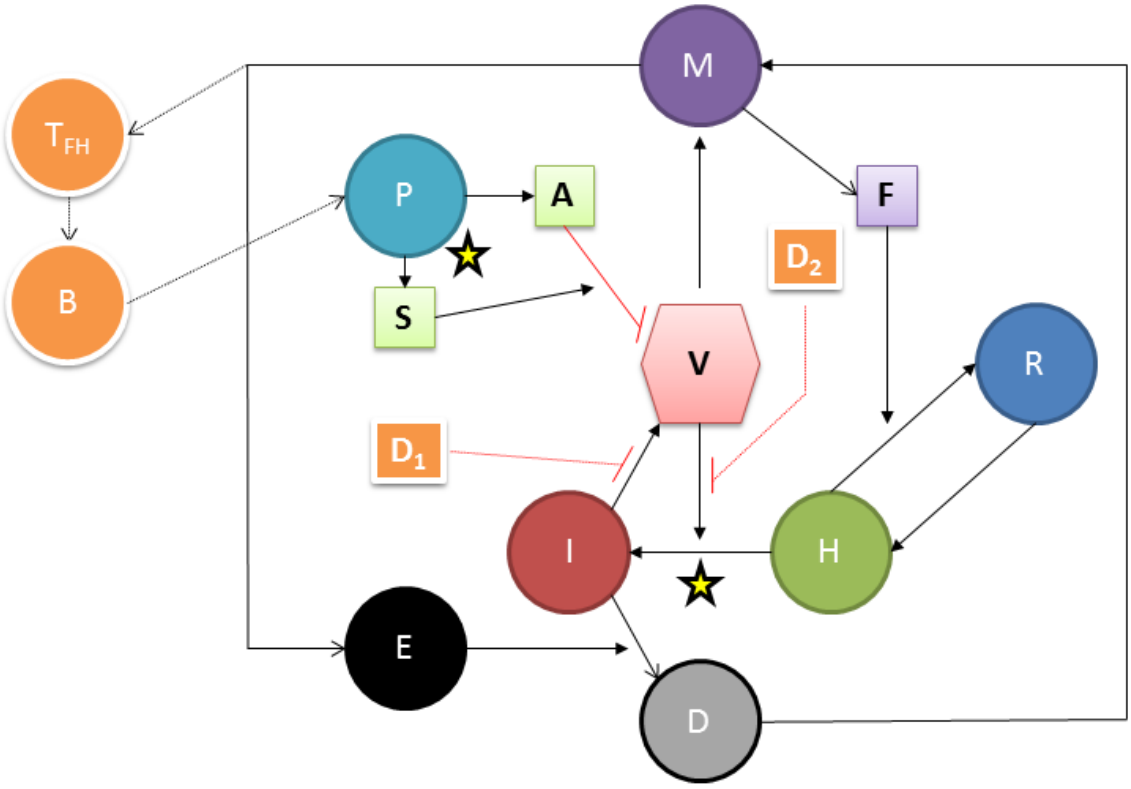


Figure 1.1: Each of the relevant compartments, as per the original model, are labelled in their respective abbreviations (*e.g.* Infected cells are represented by $I$ and antigen presenting cells (APCs) are represented by $M$, *etc.*). Our additions to the model are labelled as orange compartments with dotted lines for their relevant processes. These new additions include: drugs (neuraminidase inhibitors, $D_1$, and amantadine $D_2$), helper T cell:B cell dynamics ($T_{FH}$ and B) and introduction of stochasticity (labelled with yellow stars).

# Chapter 2

# Extension of the Model

## 2.1 Extension of the Model I – Modelling Antiviral therapy

To extend the model, we decided to investigate the effects of adding an antiviral agent to the system. Antivirals are drugs used to control viral infections both therapeutically and prophylactically. They operate by interfering with the virus copying sequence at one or more points in its replication cycle. For example, two common antiviral targets are:

1. **Viral M2 proton channels:** Disruption of viral unpackaging in host cytosol *via* the competitive inhibition of the viral M2 proton channel; [7]

2. **Viral Neuraminidase:** Prevention of viral budding *via* competitive inhibition of the neuraminidases responsible for severing newly-created virus particles from their host cells.[12]

Instances of antivirals exploiting the above mechanisms include Amantadine (trade name "Symmetrel") and Oseltamivir phosphate ("Tamiflu").

### 2.1.1 Including Drug Variables in Existing Model

Kinetic models have previously been used to investigate the influence of an antiviral drug on viral dynamics within infected individuals – such as the recent work of Smith *et al.* [13] who modelled drug influences in simple kinetic models. The influence of the 2 drug types was accommodated by changing terms in equations (1), (2) and (3), to redescribe the relationship between the variables $H$ (healthy cells), $I$ (Infected cells) and $V$ (Virus particles):

- $\gamma_{HV} VH \rightarrow \gamma_{HV}(1-\epsilon)VH$ in the case of Amantadinoid drugs;
- $\gamma_V I \rightarrow \gamma_V(1-\epsilon)I$ in the case of Neuroamidinase Inhibitors.

where $\epsilon$ represents the drug's efficacy. In their paper, Smith *et al.* assumed a drug concentration, and therefore an $\epsilon$ value, that was constant with respect to time – generally, this assumption would be inappropriate, as the amount of drug present in bodily tissues varies as it is degraded and excreted. We sought to extend the work from reference [13]

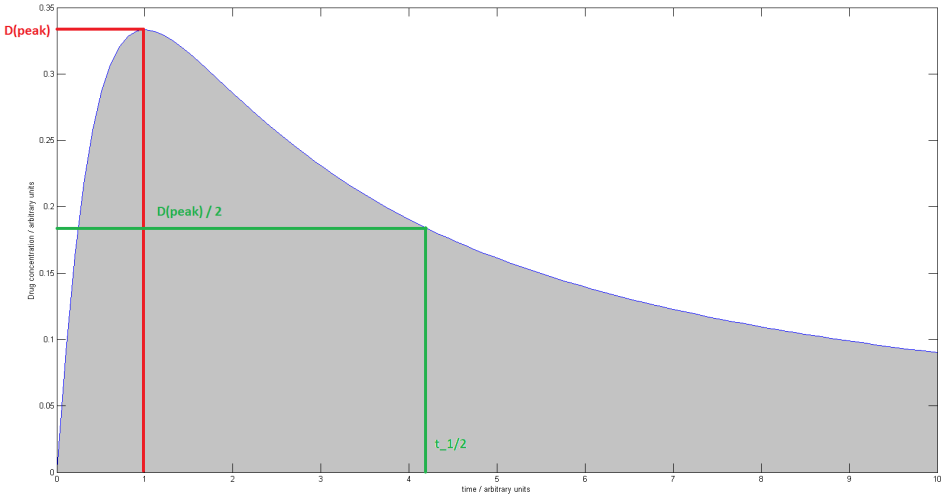


Figure 2.1: Plot of $\epsilon(t)$ against $t$ according to equation (2.1), for $c_1 = c_2 = c_3 = 1$.

by combining a more realistic, temporal drug model with the extended dynamic model of group G.[4]
To achieve this, we used a time-dependent expression for $\epsilon$ of the form

$$\epsilon(t) = \frac{c_1 t}{1 + c_2 t + c_3^2 t^2}. \tag{2.1}$$

Equation (2.1) seeks to emulate a realistic drug concentration profile in infected tissue.[1] A rapid initial surge in drug concentration following administration ($t = 0$), is followed by a slow, exponential purging of the drug from the tissues as it is excreted (figure 2.2). The parameters in equation 2.1 can be found by fitting to known experimental data pertaining to the drug's pharmacokinetic profile. Specifically, its peak time $t_{\text{peak}}$, when the full concentration $d_{\max}$ of the drug has been released, and its half life $t_{\text{half}}$, which is the time after administration at which the concentration of the drug has reached half its maximal value. The constants $c_1$, $c_2$ and $c_3$ relate to the experimental data $d_{\max}$, $t_{\text{peak}}$ and $t_{\text{half}}$ as follows:

$$c_3 = \frac{1}{t_{\text{peak}}} \tag{2.2}$$

$$c_2 = \frac{(c_3 t_{\text{half}})^2 - 4c_3 t_{\text{half}} + 1}{t_{\text{half}}} \tag{2.3}$$

$$c_1 = d_{\max}(2c_3 + c_2) \tag{2.4}$$

Also, in order to control the time of administration, a further parameter $t_{on}$ has to be introduced, and equation (2.1) changed to the form

$$\epsilon(n) = \begin{cases} 0 & \text{for} \quad 0 \le t \le t_{on} \\ \dfrac{c_1(t - t_{on})}{1 + c_2(t - t_{on}) + c_3^2(t - t_{on})^2} & \text{for} \quad t > t_{on} \end{cases} \tag{2.5}$$

[1] An alternative approach we considered was to adopt a conventional 2-compartment pharmacokinetic model (see reference [15] for an example), explicitly modelling drug concentrations as extra dynamic variables to be solved alongside the originals.

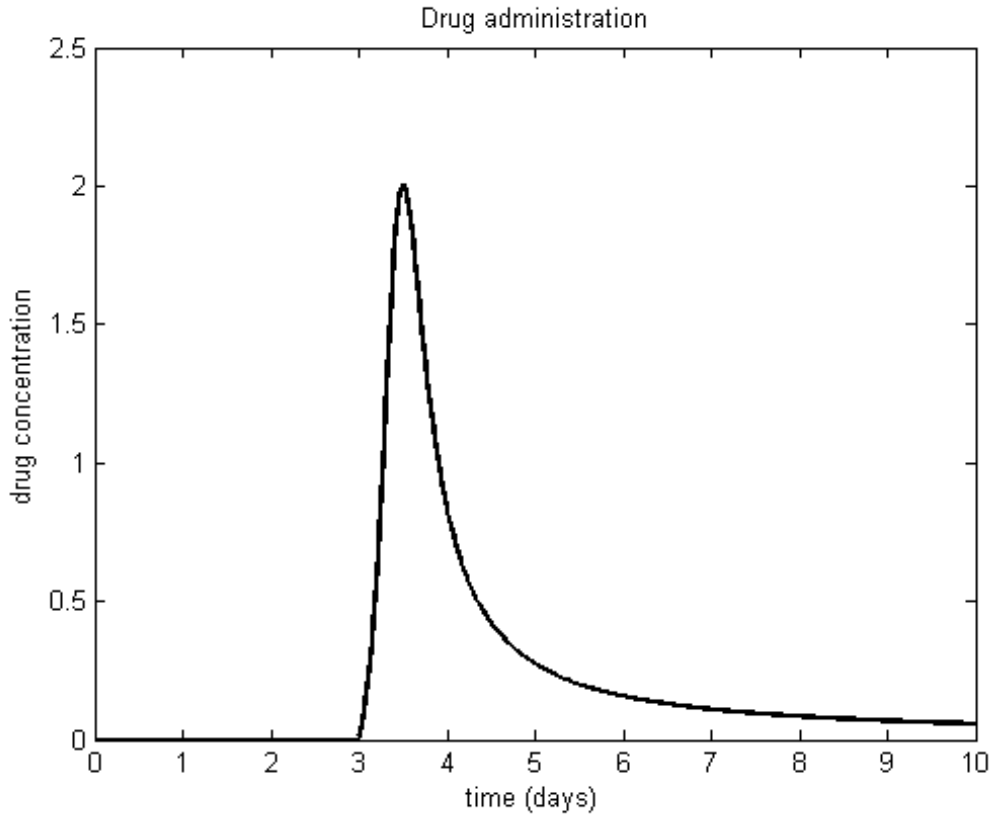


Figure 2.2: Plot of $\epsilon(t)$ against $t$ according to equation (2.5). The input parameters for this graph are: $d_{max} = 2$, $t_{on} = 3$, $t_{peak} = 0.5$, $t_{half} = 0.9$, which results in $c_1 = 1.42$, $c_2 = -3.29$, $c_3 = 2$.

Key questions include:

1. How does the time delay between infection onset and drug administration effect the treatment outcome?

2. Can polytherapy exhibit synergy? *I.e.* could 2 drugs working together ever achieve more than the sum of their individual effects?

### 2.1.2 Effects of adding VNI antivirals – some preliminary results

Adding a drug variable of the form of equation (2.1) in the GUI of group G enabled us to probe the qualitative effects of the administration of a hypothetical VNI antiviral to an infected patient.

Firstly, we examined the effect of changing the drug dose (by changing the value of parameter $d_{\max}$) on the viral population, as a function of time. Plotting the resulting curves of $V(t)$ against $t$ revealed some interesting behaviour (figure 2.3): as the drug dose was increased over the range $0.1 < d_{\max} < 1$, the onset time of the "viral surge" observed by Hancioglu *et al.* [1] increased,[2] accompanied by a marginal fall in severity

[2]We note that the intial decline in severity with increasing dosage is actually followed by a marginal

as measured by the maximum of $V(t)$. Then, when the dosage exceeded a critcal value of around $d_{\max} = 0.52$, the viral surge peaks vanished, implying that there was sufficient drug present at the correct time so as to completely inhibit the viral take-over.

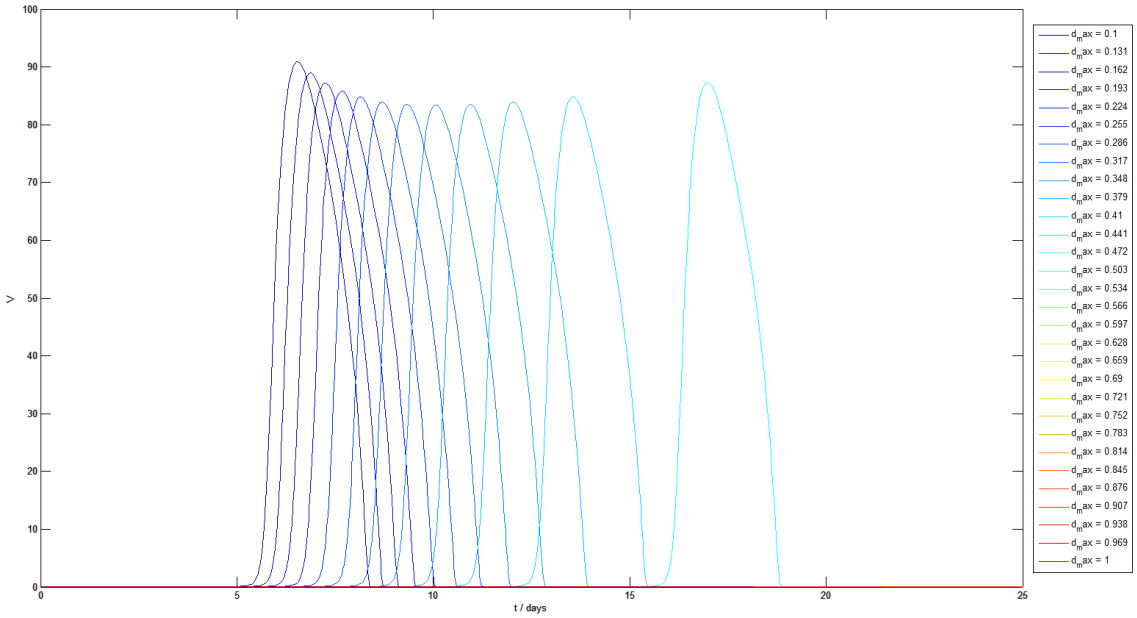


Figure 2.3: Plot of viral population $V$ against $t$ for 30 values of $d_{\max}$, given parameters $t_{\text{on}} = 0, t_{\text{peak}} = 3, t_{\text{half}} = 8$. The sudden disappearance of viral surges for $d \geq 0.543$ was unforseen.

Secondly, we investigated the effects of changing the time at which the drug was administered, for a fixed drug profile (figure 2.4), using a parameter sweep extension to group G's GUI. We observed that the time of administration, $t_{\text{on}}$, was of critical importance in determining the severity of the infection. If drugs are administered too late after onset of an infection, then the drug is ineffective with respect to virus concentration, as should be expected. The only influence a late administration can have is on the speed of viral decrease.

increase in same, for reasons as yet unknown to us.

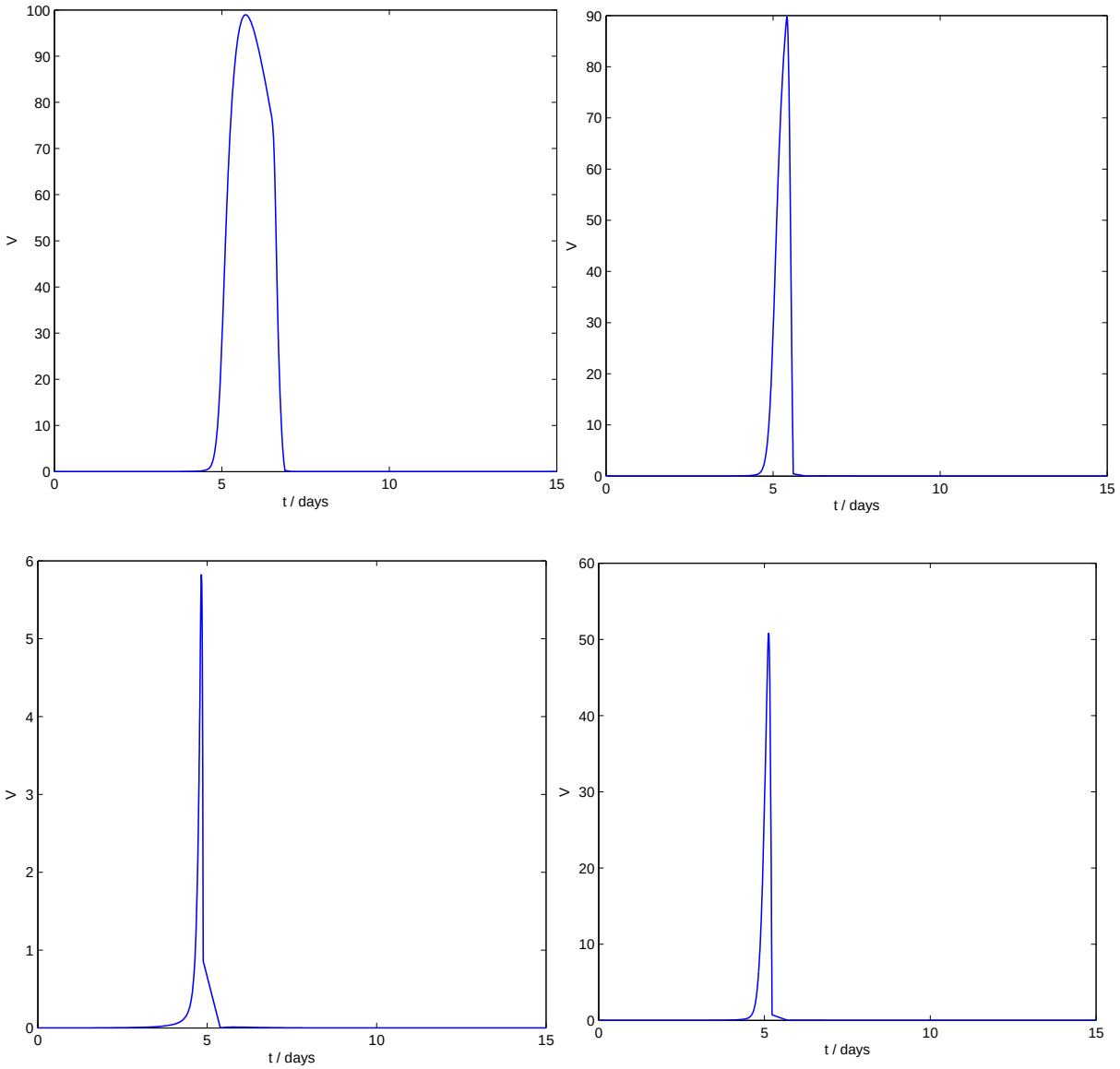


Figure 2.4: Consequences of altering the drug administration time for a given dose profile ( $d_{\max} = 4, t_{\text{peak}} = 0.14, t_{\text{half}} = 0.33$ ). Clockwise from top left: $t_{\text{on}} = 6.5,\ 5.4, 5.1,\ 4.8$. Note the change in scale along the y axis – administering the drug at times exactly coincidental with the viral surge can result in a 20-fold reduction in viral population.

We can infer from the parameter sweep shown in the previous plots, that the drug is most effective when administered around $t_{on} = 4.8$. We used this value of $t_{on}$ to check the effect of the maximum drug concentration $d_{max}$ on the virus concentration. We found that the concentration of viruses drop even for small values of $d_{max}$, but rises again if it has not reached a certain threshold, albeit not as high as without administering drugs.

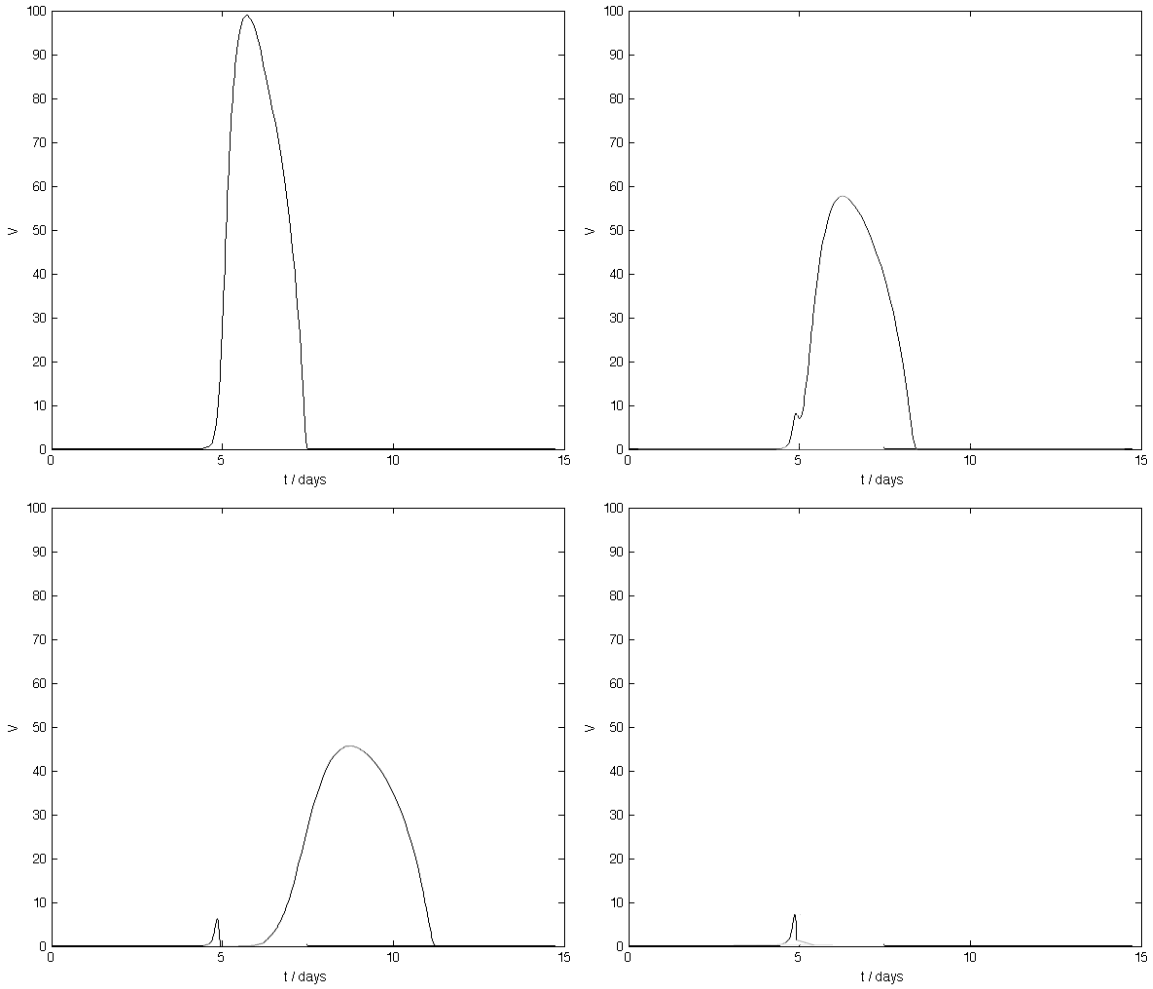


Figure 2.5: Plot of virus concentration versus time. The administration, peak and half-life times are held constant, just the maximum drug concentration is varied. The parameters are $t_{on} = 4.8$, $t_{peak} = 0.15$, $t_{half} = 0.4$, and $d_{max}$ varies in reading order from top left to bottom right through the values 0, 1, 2, 3. The maximum values of the virus concentrations drop from approx. 100, 60, 45 to 6 (again in reading order).

Generating representative plots of virus concentration versus time under drug administration poses the same problems as the original model. That is to say, that there is a far too large dependency on parameters that cannot (presently) be derived from experimental measurements. However, it is possible to infer certain dependencies on the parameters with the drug administration model as is.

## 2.2 $T_H$ cells and Plasma Cell Dynamics

In the original model, the interaction of $T_H1$ and $T_H2$ cells were omitted with almost no justification - this seems unusual, especially considering how critical $T_H$ cells are fundamental to the clonal expansion of B-cells and their transition into plasma cells. Interestingly, new research suggests that it's neither $T_H1$ or $T_H2$ cells; instead, another cell type, the follicular helper T-cell, $T_{FH}$, facilitates the switch [14]. Though it would be ideal to integrate $T_{FH}$:B-cell interaction into the model to enhance its biological relevance, their interaction has not yet been substantially modelled and our group felt it was precocious to propose an entirely new equation to model this interaction. On the other hand, the interaction of T-cells and B-cells has been modelled elsewhere [8] [9].

The change in T-cell population to antigen $k$ (*i.e.* $\frac{dT_k}{dt}$) is dependent on the T-cell death rate ($k_{DT}$), proliferation rate ($k_{PT}$), thymic output rate ($k_{ST}$) and the function $\alpha T$ which depends on $\pi_k$ (stimulatory signal for T-cell activation), $\eta_k$ (inhibitory signal for T-cell activation), $T_k$, and two constants $a_{T1}$ and $a_{T2}$, hence leading to the equation:

$$\frac{dT_k}{dt} = -k_{DT} \cdot T_k + k_{PT} \cdot \alpha_T(\pi_k, \eta_k, T_k) + k_{ST}; \tag{2.6}$$

$$\alpha_T(\pi_k, \eta_k, T_k) = exp\left[-\left(\frac{log(\eta_k) - a_{T1}}{a_{T2}}\right)^2\right] \cdot \pi_k \cdot T_k \tag{2.7}$$

The population of active B cells, $B_i$, is dependent on the T-cell population; like equation 2.6, $B_i$ depends on B-cell death rate ($k_{DB}$), proliferation rate ($k_{PB}$), and the function $\alpha B$ (which depends on $\sigma_i$ [B cell induction signal], $\tau_i$ [number of active T cells], $\beta_i$ [the number of induced B cells] and $B_i$), thus leading to:

$$\frac{dB_i}{dt} = -k_{DB} \cdot B_i + k_{PB} \cdot \alpha_B(\sigma_i, \tau_i, B_i); \tag{2.8}$$

$$\alpha_B(\sigma_i, \tau_i, B_i) = \frac{\tau_i \cdot \beta_i}{\tau_i + \beta_i}, \tag{2.9}$$

$$\beta_i = exp\left[-\left(\frac{log(\sigma_i) - b_1}{b_2}\right)^2\right] \cdot B_i \tag{2.10}$$

Surprisingly, for the purposes of analysis, the authors simplify both 2.6 and 2.8 assuming that there is at least one activated T-cell, thus reducing this system to:

$$\frac{dT_k}{dt} = (k_{PT} - k_{DT}) \cdot T_k \tag{2.11}$$

$$\frac{dB_i}{dt} = -k_{DB} \cdot B_i + k_{PB} \cdot \frac{T_k \cdot B_i}{T_k + B_i}; \tag{2.12}$$

Given this new simplified formula, we decided to assimilate these two equations into the existing computational model. Unfortunately, one of the biggest challenges was actually fitting the newly-defined $T_{FH}$ cells into the overall scheme in Figure 1.2. Considering that the mechanism of $T_{FH}$ cell differentiation remains undefined and the possibility that $T_{FH}$ cells are activated by B cells [3], the equations that we are using may be inaccurate. On the other hand, there is sufficient evidence to suggest that $T_{FH}$ cells are activated by

signals derived from APCs - thus, continuing with this notion, we suggest that equation 2.12 is scaled by the population of APCs, $M$.

$$\frac{dT_k}{dt} = (k_{PT} - k_{DT}) \cdot T_k \cdot M \tag{2.13}$$

Under the new scheme, naïve B-cell activation by $T_{FH}$ cells can be represented by equation 2.12. As for the subsequent formation of plasma cells from B cells, we can see the original ODE for the formation of plasma cells (where $b_{PM}$ and $\alpha P$ are constants representing plasma cell synthesis and death, respectively):

$$\frac{dP}{dt} = b_{PM} M P + \alpha_P (1 - P) \tag{2.14}$$

We decided to modify this equation slightly; we retained the constants as they represent properties intrinsic to the *plasma cell*. However, we changed the dependence of plasma cell synthesis to the population of active B cells, $B_i$, rather than M (*N.B.* To avoid confusion in nomenclature, $b_{PM}$ was renamed to $b_{PB}$). Therefore, we would attain:

$$\frac{dP}{dt} = b_{PB} B_i P + \alpha_P (1 - P) \tag{2.15}$$

In summary, to account for the role of helper T cells in plasma cell synthesis with the influenza virus (V) as the antigen, we propose the following system of ODEs:

$$\begin{aligned} \frac{dT_V}{dt} &= (k_{PT} - k_{DT}) \cdot T_V \cdot M \\ \frac{dB_i}{dt} &= -k_{DB} \cdot B_i + k_{PB} \cdot \frac{T_V \cdot B_i}{T_V + B_i} \\ \frac{dP}{dt} &= b_{PB} B_i P + \alpha_P (1 - P) \end{aligned}$$

Due to the addition of these new compartments, the inherited C++ mex function and its header, `mexodestiff.cpp` and `mexodestiff.hpp`, were re-adjusted. Much of the code was retained, with the exception that new variables were defined as necessary, and the Jacobian matrix in the header file was re-defined to accommodate for the two new compartments.

## 2.3 Introduction of Stochasticity

To improve the current computational model, we felt that giving the user the freedom to incorporate stochasticity into their analyses was essential. The idea was not to replace the code, but instead give the user a more realistic account of how cell populations and viral titre can vary over time. We decided that the two areas in which introduction of stochasticity were most biologiclaly justified were in virus production and development of antibody affinity. It is understood that there is a wide range of variability in virus production from infected cells [10], so we felt justified in adding a noise term which affects the total virus population V.

Similarly, the model already incorporates the affinity maturation of antibodies, but fails to model the fundamentally stochastic nature of the recombination process. In theory,

every antibody is constructed from a diverse genetic framework and every antibody has a different affinity for the virus - only the antibody that binds strongest is selected for further expansion in the geminal centres [5]. Consequently, we felt that another target for noise terms was the rate of change of variable S, in order to incorporate the iterative, random process of antibody selection.

There were two possible ways of approaching this problem; it is possible to introduce stochastic noise into either the cell populations or the dynamical rate paramters. Both were explored, and are discussed below.

### 2.3.1 Stochastic Populations

The study of stochastic differential equations (SDEs) is a widely studied topics in mathematics, with many resources for different algorithms and implementations [6]. The ideal is to solve a differential equation with a well-defined noise function adding a stochastic element over time:

$$\frac{\mathrm{d}X}{\mathrm{d}t}(t) = \mathrm{f}(X) + \text{“noise”} \tag{2.16}$$

In the above, $f(X,t)$ is the “drift function” of the system that would comprise a traditional ODE and “noise” is the component that contributes the stochastic element with time. It is possible to introduce a traditional finite-difference forward-regression approach, by implementing the Euler-Maruyama method for an SDE, such that:

$$X(t+\delta t) = X(t) + (\delta t)\,\mathrm{f}(X(t)) + \sqrt{\delta t}\,\mathrm{G}(X(t))\,\eta \quad \text{such that} \quad \eta \in N(0,1) \tag{2.17}$$

Here, $N(0,1)$ is a normal distribution, with a mean of zero and standard deviation of one and $G(X(t))$ is the “diffusion coefficient” for our system. Given access to the functions $f(x)$ and $G(X)$ and a normally distributed random number generator, it is possible to solve this iteratively, with each new timestep explicitly using the previous one. This is a simple but reliable implementation, which represents a good method for introducing background noise into the concentrations of variables in our system.

### 2.3.2 Stochastic Rates

The next approach, which is possibly more theoretically justifiable, is to allow the parameters of the ODE system to vary stochastically with each call of the function $f(X(T))$. This allows the code to isolate the mechanisms are contributing noise, rather than simply adding some arbitrary amount of stochasticity to the variable after each timestep. This method allows for the standard library of ODE solving regression algorithms, since it is the drift function that contributes noise rather than the finite different method itself. It is consequently slightly easier to implement in various languages.

### 2.3.3 Implementation in Matlab®

These two approaches were implemented in two different solver functions: `sdesolver()` and `sderatesolver()`, respectively. These resembled the native Matlab® solvers, but

with an extra input, describing the nature of the diffusion coefficient $G(X(t))$. `sdesolver()` is handed a two by ten dimensional matrix containing the vectors $a$ and $b$, used such that $G(X) = a.*X.^b$, elementwise. This gives a wide sampling of potential diffusion functions, allowing for easy sampling of interesting parameter spaces. The second function, `sderatesolver()`, demands two inputs, which specify the strength of the noise added to the creation rates of V and S, these being judged to be the processes most likely to realistically exhibit highly stochastic behaviour. A difficulty of implementation involved forcing all variables to remain non–negative: if either the numerical methods or stochastic noise push any variables below zero into an unphysical region of phase space, the variables would quickly and collectively diverge into positives or negative infinities. This was avoided with a simple zero check and modification.

### 2.3.4 Implementation in C with Mex files

The implementation of stochasticity in C was done reusing parts of Group G's C++ mex code and converting this to C syntax and adding a stochastic ODE solver after the Euler-Maruyama method described earlier. A 10 by 2 matrix with noise parameters a and b, representing the inputs in $G(x) = ax^b$ as described in the section above, was read in and used to calculate a noise factor, $G(x)$ from equation (2.17). In order to generate a Gaussian random variable, denoted $\eta$ in the aforementioned equation, a Box-Mueller transform [2] was used:

$$X_1 = \sqrt{-2\log(U_1)}\cos(2\pi U_2) \tag{2.18}$$

In this equation $U_1$ and $U_2$ denote uniformly distributed random numbers in the interval $[0; 1]$. The method used to create these uniformly distributed random numbers was: *rand()* % 1001/1000 The random number generator *rand()* was seeded by the CPU time at every loop iteration, thus the same seed was used for the $2N$ calls to *rand()* at every iteration, where $N$ is the number of equations in the system of coupled ODEs. This seeding method can be a cause for systematic error, due ot the pseudo-randomness of the *rand()* function in C. A further cause of systematic error is if RAND_MAX $\gg$ 1000 is not true. In this case there is a bias towards lower values. Furthermore, the uniform random variable only being sampled to 3 decimal places can cause errors in the distribution of the gaussian random variable.

The mex function was tested against the same code in matlab for different numbers of iterations of the main loop, dependent on start and end times and the timestep. The time comparison obtained from this can be seen in Table 2.1.
The mex solver in C exhibits some numerical instability, which occurs occasionally when the initial value for the Virus population, $V$, is at around 0.01 and the timestep is of a similar order of magnitude. This is an instability which is inherent in the method and its removal is out of the scope of this project.

Upon confirmation of the efficiency of the mex solver, this was then re-designed with the two additional compartments to generate a 12 by 2 matrix with appropriate noise parameters.

| Iteration | Matlab Code Time (s) | Mex Code Time (s) | Speed up factor |
|---|---|---|---|
| 1000 | 0.0016 | 0.0516 | 32.25 |
| 3000 | 0.0045 | 0.1550 | 34.44 |
| 10000 | 0.0147 | 0.5130 | 34.90 |
| 100000 | 0.1480 | 5.1298 | 34.66 |

Table 2.1: Comparison of Run times for Matlab and C implementation of the stochastic solver. The simulation end time was varied from 10 to 1000 with start time 0 and timestep 0.01. All other parameters used were given by the default values in the GUI.

## 2.4 Results

We observed several interesting phenomena over the course of exploring parameter and solver configurations for the SDEs. These can be loosely grouped into two classes, those of noise propagation and virus resurgence.

### 2.4.1 Noise propagation

An interesting question which we addressed is the extent to which noise introduced to one parameter will appear in the output of another parameter. Choosing S and V as being test cases, we added noise to each independently and both together, observing the extent to which this effect the behaviour of other variables. We found that noise added to V became manifest in slight perturbations to the response curves of H, I and M., although the curves were still qualitatively the same objects. Noise added to the variable S had no obvious effect: despite being relatively well–connected in the system of ODEs, the other variables were less effected by perturbations in S. This may be owing to the role of S as a logical variable, ranging from its steady states at zero and one; stochastic variation of this variable may be insignificant compared to the overall logical state of the efficacy.

### 2.4.2 Virus Resurgence

One dynamically significant effect of adding noise to either the virus population V or the rate of creation of V by infected cells is the occurrence of virus resurgence and infection. If the noise introduced to the rate of creation of V is significantly large then the virus population plotted against time will feature more than one spike, representing a reappearance of the virus population in the cell population. We speculate that this is the stochasticity pushing the depleted virus count above a threshold, sparking another wave of infection when the antibody count has decayed sufficiently. This new wave is accompanied by a corresponding rise in antibodies in response, hoping to combat the infection. It is interesting that the subsequent virus and antibody peaks are significantly lower than those of the first wave: this can be explained by reference to the antibody efficacy S. This grows every time a new infection appears, describing the cell population's capacity to "remember" how to combat the virus, making the virus less successful and demanding less antibody concentration for each wave. These effects can be observed in Figure 2.4.2.

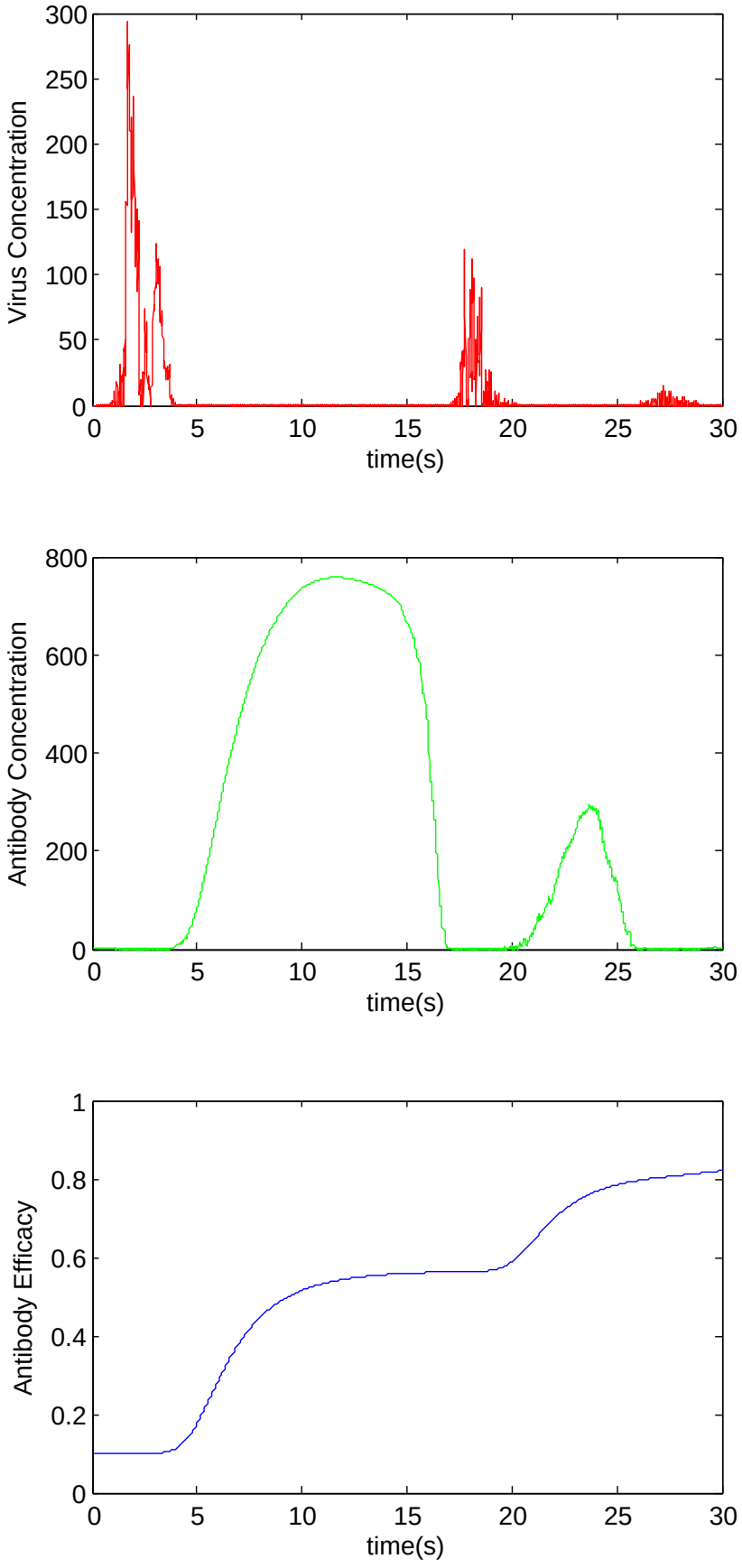


Figure 2.6: Outputs for V, A and S against time, using the default GUI parameter set and `sderatesolver()`, with `NoiseParameters.VNoise = 5` and `NoiseParameters.SNoise=0`

# Conclusions

To conclude, we have extended group G's implementation of Hancioglu's viral dynamics paper [1] in the following ways:

- Investigated the effects of adding noise to specific system variables using a Matlab®;
- Incorporated an analytical expression for the influence of an antiviral drug on the system;
- Modified the model to include further aspects of human immune system interactions;
- Implemented a fast stochastic solver in `C++`, interfaced with the orginal code using `Mex` functions.

# Bibliography


[1] David Swigon Baris Hancioglu and Gilles Clermont. A dynamical model of human immune response to influenza a virus infection. *Journal of Theoretical Biology*, 246:70–86, 2007.

[2] G.E.P. Box and M.E. Muller. A note on the generation of random normal deviates. *The Annals of Mathematical Statistics*, 29(2):610–611, 1958.

[3] Shane Crotty. Follicular helper cd4 t cells (tfh). *Annual Reviews of Immunology*, 29:621–663, 2011.

[4] N. Pearce L. Bowler F. Wolfreys M. Aldeghi I. Frost and P. Taylor. A dynamical model of human immune response to influenza a virus infection. 2013.

[5] Christine M. Grimaldi, Ruthmarie Hicks, and Betty Diamond. B cell selection and susceptibility to autoimmunity. *The Journal of Immunology*, 174(4):1775–1781, 2005.

[6] Desmond J. Higham. An algorithmic introduction to numerical simulation of stochastic differential equations. *SIAM Review*, 43(3):pp. 525–546, 2001.

[7] Jun Hu, Riqiang Fu, and Timothy A. Cross. The chemical and dynamical influence of the anti-viral drug amantadine on the m2 proton channel transmembrane domain. *Biophysical Journal*, 93(1):276 – 283, 2007.

[8] Jose Faro Jorge Carneiro, Antonio Coutinho and John Stewart. A model of the immune network with b-t cell co-operation i – prototypical structures and dynamics. *Journal of Theoretical Biology*, 182:513–529, 1996.

[9] Jose Faro Jorge Carneiro, Antonio Coutinho and John Stewart. A model of the immune network with b-t cell co-operation ii – the simulation of ontogenesis. *Journal of Theoretical Biology*, 182:531–547, 1996.

[10] Hugh Mitchell, Drew Levin, Stephanie Forrest, Catherine A. A. Beauchemin, Jennifer Tipper, Jennifer Knight, Nathaniel Donart, R. Colby Layton, John Pyles, Peng Gao, Kevin S. Harrod, Alan S. Perelson, and Frederick Koster. Higher level of replication efficiency of 2009 (h1n1) pandemic influenza virus than those of seasonal and avian strains: Kinetics from epithelial cell culture and computational modeling. *Journal of Virology*, 85(2):1125–1135, January 15, 2011.

[11] A.S. Perelson et al. Modelling viral and immune system dynamics. *Nature Reviews Immunology*, 2(1):28–36, 2002.

[12] Kanako Satoh, Ryouichi Nonaka, Akio Ogata, Dai Nakae, and Shin ichi Uehara. Effects of oseltamivir phosphate (tamiflu) and its metabolite (gs4071) on monoamine neurotransmission in the rat brain. *Biological and Pharmaceutical Bulletin*, 30(9):1816–1818, 2007.

[13] Amber M. Smith and Alan S. Perelson. Influenza a virus infection kinetics: quantitative data and models. *Wiley Interdisciplinary Reviews: Systems Biology and Medicine*, 3(4):429–445, 2011.

[14] K. Kai McKinstry Susan L. Swain and Tara M. Strutt. Expanding roles for cd4+ t cells in immunity to viruses. *Nature Reviews Immunology*, 12:136–148, 2012.

[15] Nikolaos M. Tsoukias and Steven C. George. A two-compartment model of pulmonary nitric oxide exchange dynamics. *Journal of Applied Physiology*, 85(2):653–666, 1998.