A Project Report

On

In-host viral dynamic modeling of SARS-CoV-2 infection

BY

VIDULA RAJAIN

ABSTRACT

In this paper, we investigate mathematical models to understand the behaviour of SARS-Cov-2 infection in human host and validate the models using reported viral load data. It is a type of coronavirus that causes respiratory illness. It was first detected in Wuhan, China and is now a global pandemic. It is dangerous for some people since it can cause severe lung damage, though most people are asymptomatic.

The virus is transmitted through respiratory droplets, it then settles in the nasal passages and starts replicating throughout the body. If the virus enters the lungs, it can invade the tiny air sacs (alveoli) and cause inflammation in the lungs which can cause breathing problems. It has been observed that initially the viral load increases rapidly till it reaches a peak, followed by a plateau phase possibly generated by lymphocytes as a secondary target of infection. In the last stage, viral load declines due to the emergence of adaptive immune responses.

Using viral load data obtained from a paper reporting the clinical trials conducted in Germany, we would do curve fitting to obtain the value of the unknown parameters used in the mathematical model. We would then see the effect of treatment on the viral dynamics and compare the effect of drugs with different efficacies.

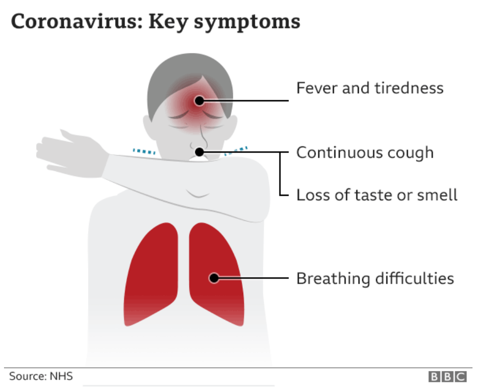
Contents

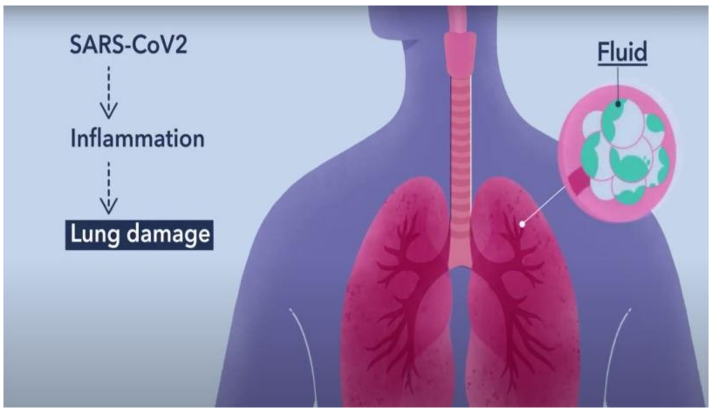
INTRODUCTION

This study is about the virus that completely changed our lives. The virus that united the whole world in the fight against it and the one that put so many countries under lockdown. SARS-CoV-2 was first detected in December 2019, in Wuhan, China and it is now a global pandemic.

It is crucial to understand how a virus reacts inside the host body to be able to come up with treatment plans. Mathematical models help us in predicting the way a virus would react. In this study, we would use mathematical models to observe the characteristic of SARS-Cov-2 virus. The viral load data used in this paper was from patients in Germany (paper given in reference). Viral RNA levels of all patients were collected every day using RT-PCR test. Then the models are fitted to this data to observe the interaction between viral replication and host immune response.

The virus is transmitted through direct contact with respiratory droplets of an infected person (e.g., Sneezing & coughing). If the virus enters the lungs, it can invade the tiny air sacs (alveoli) and cause pneumonia. At this point, the lungs will become inflamed, blood oxygen level would drop and it would become harder to breath. The inflammation is due to the body's immune response to the infection.





Coronavirus targets type 2 pneumocytes. These cells are responsible for transporting fluid out of the airspace(lungs). When cells are infected, the body's immune response is triggered which results in accumulation of fluid in the air space and inflammation. Because pneumocytes are then not able to perform their function efficiently, this leads to even more fluid accumulation in the lungs. Due to this accumulated fluid in the airspace, alveoli are not open for gas exchange and the host has breathing problems. If the inflammation keeps increasing, it might cause lung damage and the patient may require the use of life support systems.

MATHEMATICAL MODELS

A BASIC VIRAL DYNAMIC MODEL: -

Introduction

This is a very basic model that was studied to get a basic understanding of the virus. It only includes target cells, infected cells and the virus.

The ODEs for the basic model are: -

$dT/dt = -\beta V\ T$ (1)

$dI/dt = \beta V\ T - \delta I$ (2)

$dV/dt = pI - cV$ (3)

Where,
T: Target Cells
V: Virus
I: Infected cells
β: Viral Infection rate
δ: Death rate
p: rate of production of virus per infected cell
c: rate at which the virus is cleared

(1) Tells the rate at which the target cells, pneumocytes are infected. It depicts the amount of uninfected target cells left after the virus infected them at a mass action rate $\beta V\ T$.

(2) This equation depicts the infected cells. The left-over infected cells would be the total infected cells subtracted by the infected cells that die. We have assumed the death rate to be a constant for this model, 2 cells per day.

(3) Shows the amount of virus left. The infected cells produce more virus at a rate p and the virus is cleared out at a rate c.

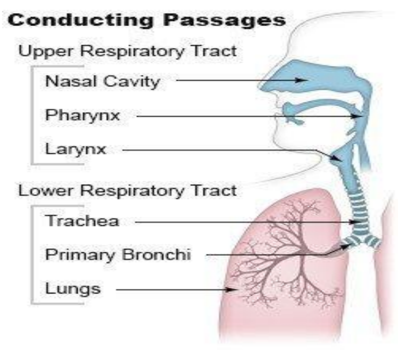
Data used for this Model: -

Data used from Table S2. Best fits of model (1) to the URT viral load data in patients from Germany. Patients chosen: #1 and #8

Patient	β (ml/virus/day)	p (day^-1)	c (day^-1)	V (0) RNA/ml
#1	0.01	50	0.93	0.0001
#8	0.00091	2	0.6	0.0001

Other data from the research article: -
δ= 2 cells /day
T (0) =60000 (Given number of target cells initially)
I (0) = 0 (Initially there would be no infected cells)

We have used Upper Respiratory Tract (URT) data instead of LRT since it is easier to obtain samples from URT. The most common test done for COVID-19 is the RT-PCR test, this gives the URT data since sample is collected from the nasal cavity or the throat. Collecting samples from the LRT is very difficult and less common.



Using ode45 in MATLAB to solve the odes, the following figures were obtained: -

Figure obtained for patient #1: - (Code in Appendix A)

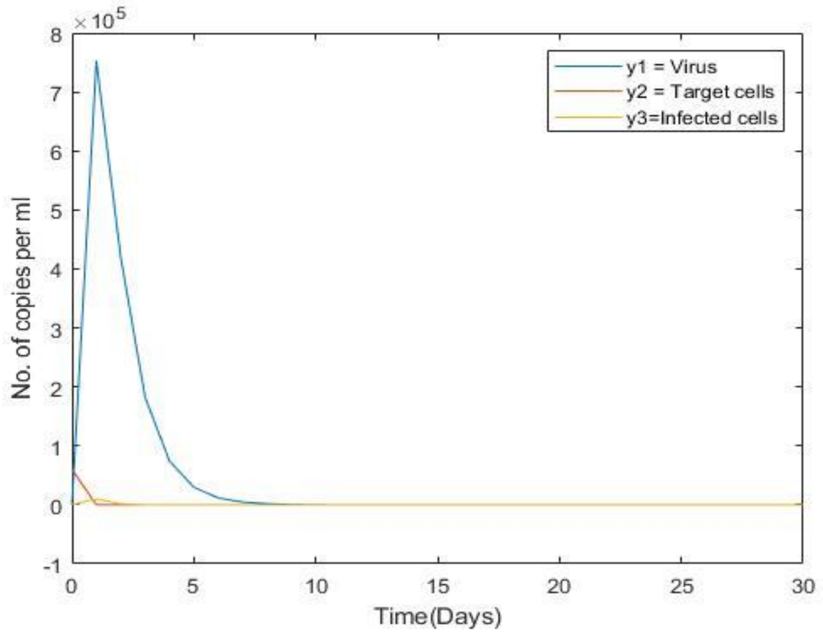
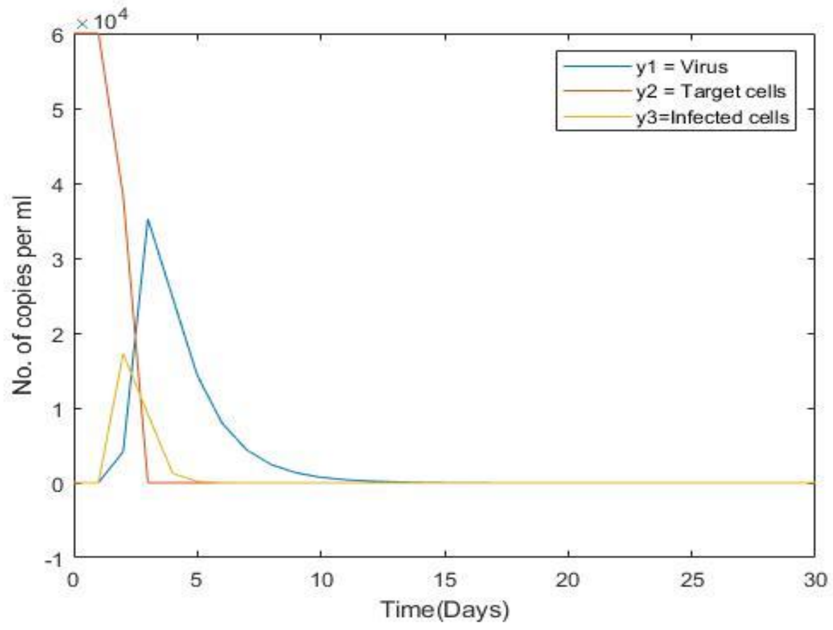


Figure obtained for patient #8: - (Code in Appendix A)



THE MODEL WITH A SECONDARY TARGET OF INFECTION

Introduction

This represents the viral dynamics more accurately, by taking into account some more parameters: -

1. A secondary target of infection (Lymphocytes)
2. The death rate is a function instead of a constant
3. Parameters that account for the adaptive immune response

The ODEs for this model are: -

$dT_1/dt = -\beta V\, T_1$ (1)

$dT_2/dt = \lambda - \beta V\, T_2$ (2)

$dI/dt = \beta V\, (T_1 + T_2) - [\delta\,(t) + \omega T_2]\, I$ (3)

$dV/dt = pI - cV$ (4)

DEATH RATE: - (5)

$\delta\,(t) = \delta_o$ for $t < \mu$

$\delta\,(t) = \delta_o e$^$(\sigma(t-\mu))$ for $t \geq \mu$

Where,

T_1: concentration of pneumocytes
T_2: concentration of lymphocytes
V: Virus
I: Infected cells
β: Viral Infection rate
p: rate of production of virus per infected cell
c: rate at which the virus is cleared
λ: Rate of recruitment of lymphocytes
δ_o: Base death rate
σ: Parameter in the killing by adaptive immune response
μ: Time of emergence of adaptive immune response

Explanation: -

(1) $dT_1/dt = -\beta V\, T_1$: Rate at which pneumocytes are infected to generate infected cells.
This depicts the amount of pneumocytes left after the virus infects them.

(2) $dT_2/dt = \lambda - \beta V\, T_2$: Rate of recruitment of lymphocytes- rate of infection of lymphocytes.
This depicts the number of recruited lymphocytes left on the infection site after the virus infects some of them at a mass action rate of β.

(3) $dI/dt = \beta V\, (T_1 + T_2) - [\delta\,(t) + \omega T_2]\, I$: Total rate of infection- death rates of infected cells, where $\omega T_2 I$ is death by innate immune response.
This depicts the number of infected cells left after some of them die.

(4) $dV/dt = pI - cV$: rate of production – rate of clearance. This depicts the amount of virus left over taking into account new virus produced from the infected cells and the clearance of virus cells at a constant rate.

(5) The death rate is constant up to a certain time (assumed to me 2 cells/day). After day μ, the adaptive response kicks in and the death rate becomes an exponential function.

Lymphocytes (The secondary target cells)
These were taken as the secondary target since they were detected in a large number of infected patients. They are present in white blood cells. They are recruited to the infection site due to the lung inflammatory response. We presume the recruitment rate to be constant since the turnover of uninfected lymphocytes is slow and they are usually constant in the body of a healthy person.

Immune responses

Innate immune responses

The innate immune responses are the first line of defense against invading pathogens. Innate immune responses rely on the body's ability to recognize conserved features of pathogens that are not present in the uninfected host.

For the model we presumed the concentration of lymphocytes to be proportional to innate immune response. Therefore, $\omega T_2 I$ is clearance of infected cells by innate immune response.

Adaptive immune responses

Adaptive immunity is an immunity that occurs after exposure to an antigen either from a pathogen or a vaccination. This part of the immune system is activated when the innate immune response is insufficient to control an infection.

In the model it activates after day μ and the death rate of infected cells increases exponentially ($\delta(t) = \delta_o e$^$(\sigma(t-\mu))$).

Data used for this Model: -

Data used from Table 3: Best fits of model (3) to the URT viral load data in patients from Germany. Patients chosen: #1 and #8

Patient	**β (ml/virus/day)**	**p (day^-1)**	**c (day^-1)**	**ω (ml/cell/day)**	**σ (day−1)**	**V (0) RNA/ml**	**μ**
#1	1.1×10^{-5}	940	50	1.9×10^{-4}	0.13	0.0001	10
#8	7×10^{-6}	3100	118	1.5×10^{-4}	0.07	0.0001	6

Other data from the research article: -

δ_o= 2 cells /day
Initial concentration of pneumocytes, T_1 (0) =60000
Initial concentration of lymphocytes, T_2 (0) =0
$\lambda=10^4$ per day
I (0) = 0 (Initially there would be no infected cells)

Using ode45 in MATLAB to solve the odes, the following figures were obtained: -

Figure obtained for patient #1: - Code in Appendix B

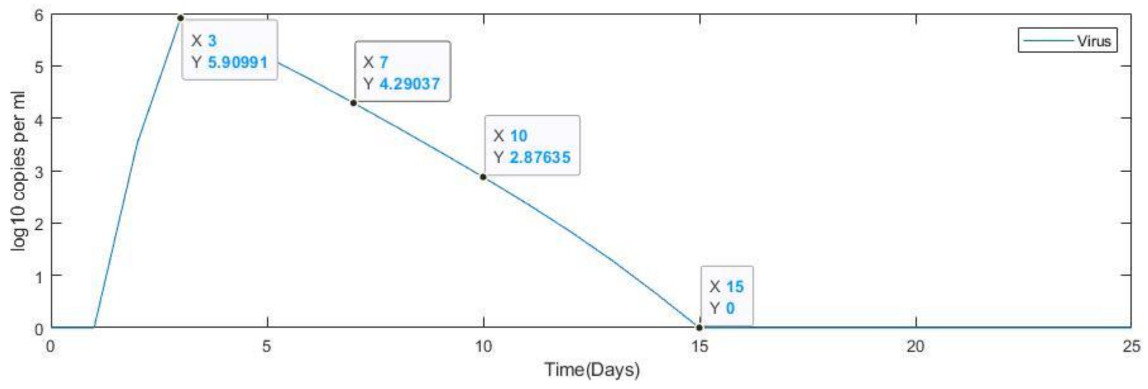
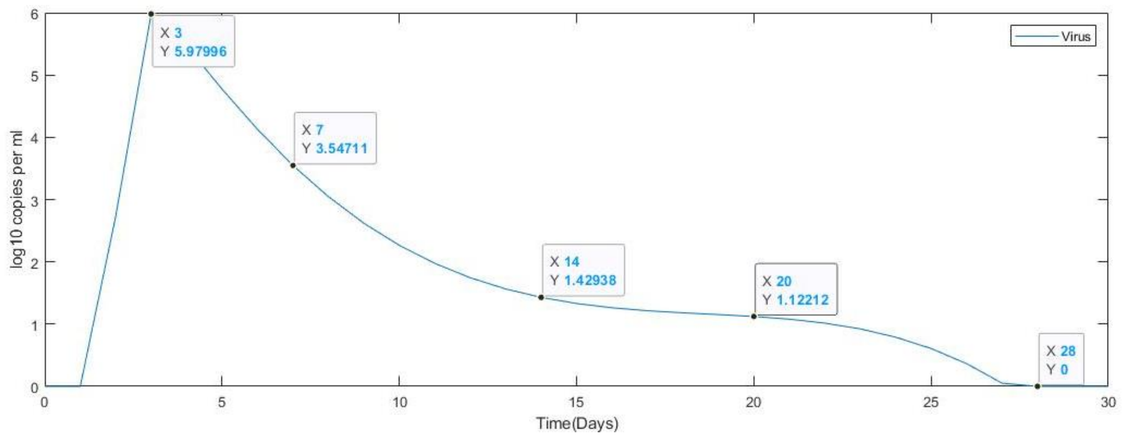


Figure obtained for patient #8: - Code in Appendix B



Therefore, to conclude ode45 to solve the ODEs representing the model since they give an accurate representation of the observed viral trends.

There are still a few drawbacks to using this model. We have assumed that the virus infects both the targets cells at the same rate, this may not be the case in real life. To make this more realistic we can take two different kinds of infected cells that were generated from the two target cells respectively, with different virus production and clearance rate. We have also not taken into account the natural death rate of the target cells. Many more parameters can be added to this model, but it would make solving it very difficult. Therefore, we try our best to minimise the number of parameters.

The next steps in this project would be to do curve fitting of the given viral load data to obtain the values of the unknown parameters of the model 3. If successful, we can then use this model for other data sets as well.

CURVE FITTING FOR THE MODEL WITH A SECONDARY TARGET OF INFECTION

In this process, the viral load data collected from each patient every day, is plotted and using the mathematical model and different initial guesses for the six parameters to be fitted, the final value of the parameters is to be found out.

Getting proper curve fitting is a tedious task and various methods were tried out before we got accurate fittings for all patients.

The following steps were carried out:-

1. Extracting Viral Load data to be used for the fitting. (This was done for 8 patients from the reference paper)
2. Curve fitting using lsqcurvefit in MATLAB. This did not give accurate results.
3. Curve fitting after generating random initial values for the parameters to be fitted, using fmincon in MATLAB and applying non linear constraints.
4. Curve fitting after data preprocessing: Replacing outliers using filloutliers in MATLAB. Fitting without outliers was giving incorrect fitting.

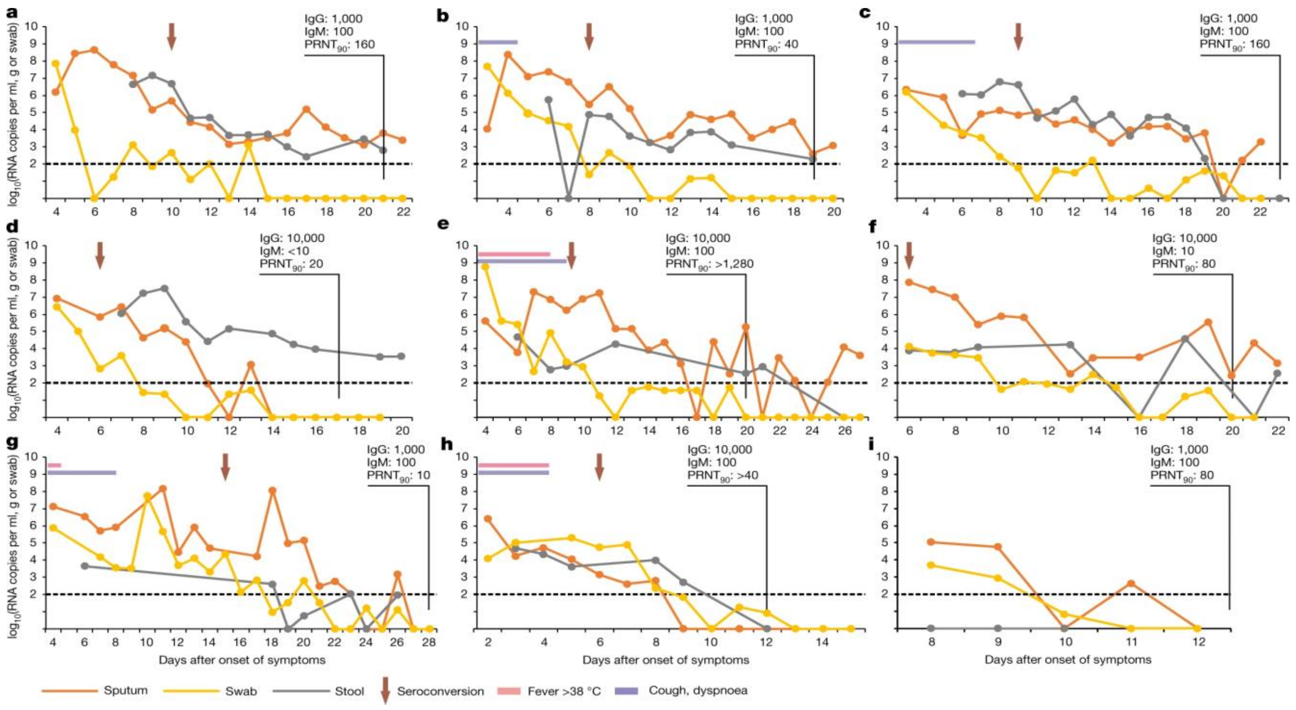
(Each task has been explained in details later)

Parameters to be fitted are:-

1. β: Viral Infection rate
2. p: rate of production of virus per infected cell
3. c: rate at which the virus is cleared
4. σ: Parameter in the killing by adaptive immune response (determines how fast the death rate increases)
5. μ: Time of emergence of adaptive immune response
6. ω: Killing rate by innate immune response

Viral load data

The viral load data was obtained from a research paper which included clinical observations of patients in Germany. RT-PCR test was used and the viral load kinetics was plotted as shown:-



Since in this project we were interested in the URT data, we extracted that data for all patients from the graphs for the swab data (yellow). Extraction was done using an online graph reader.

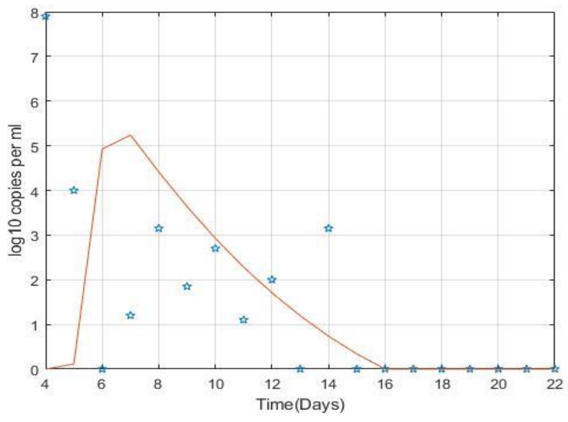
Curve fitting using lsqcurvefit in MATLAB

Lsqcurvefit is an inbuilt function in MATLAB that solves non-linerar curve fitting (data-fitting) problems with least-squares method. Steps carried out for the curve fitting :-

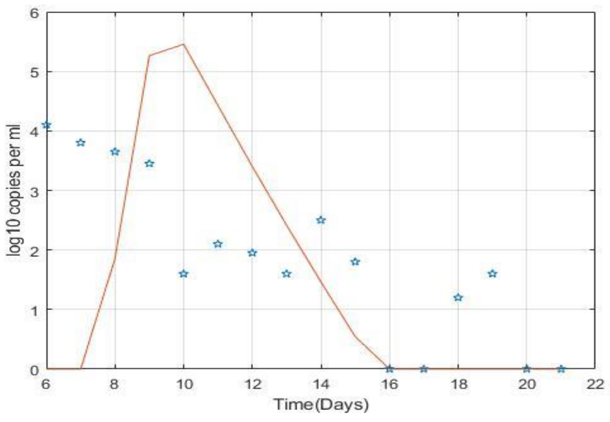
- The extracted viral load data was placed into the code, we would use these data points to do the curve fitting in MATLAB using lsqcurvefit.
- Initial guesses for the six parameters to be fitted are assumed and then the output is observed.
- We first use the parameter values given in the reference paper itself to check the output and then make slight changes to the initial guess and observe the changes in the output.
- 4 different initial parameter guesses for each patient were taken.

Conclusion:-

The fitted parameter values returned by the code depend mainly upon the initial guesses used. No consistency could be seen in the outputs. Therefore, it was concluded that lsqcurvefit is not a good option to do the curve fitting. (Code in Appendix C)



Patient 1



Patient 8

Curve fitting using fmincon with Non linear constraints and randomly generated initial values of the parameters

Fmincon is a nonlinear programming solver that minimizes the objective function bound to linear or non linear constraints. Steps carried out:-

- Random variables were generated for each of the 6 parameters under appropriate upper & lower bounds. (60 for each parameter)

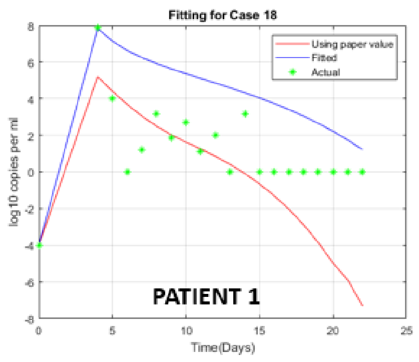
S.no	Parameter	Highest Value	Lowest Value	Upper Bound	Lower Bound
1	Beta	4.5×10−5	9.9×10−7	10^-4	10^-8
2	Sigma	0.13	0	0.5	0
3	Meu	15	6	15	6
4	Omega	3×10−3	1.5×10−4	10^-2	10^-5
5	p	1.08×10^4	8.8×10	10^5	10
6	c	118	3.5	300	0.5

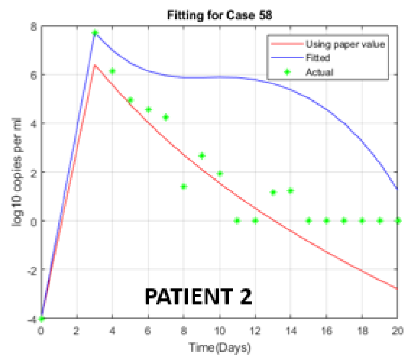
- The non linear constraint of y(4)>0.01 or log10(y(4))>-2 was applied while performing the fitting. (here y(4) is the Virus values) { Under the assumption that the virus cannot be detected below this level while testing }
- 61 cases of fitting was done for all the 8 patients.
  Where, Case 1:- using the parameter values given in the reference paper as the initial guess and rest 60 cases:- from random variable generation
- The output for each patient given was stored in excel and the error values for each case was compared with the error of the paper by calculating the percentage reduction.

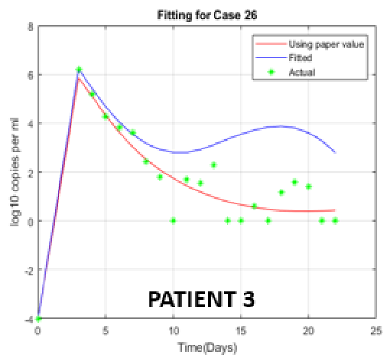
For each patient, no. of cases better than the paper:-

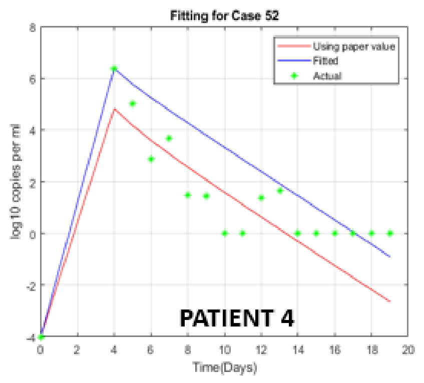
Patient no	No. of cases with error less than that of the paper (out of 61)	Percentage of cases better than the paper
1	52	85.25%
2	52	85.25%
3	15	24.59%
4	54	88.52%
7	57	93.44%
8	8	13.11%
10	51	83.61%
14	59	96.72%

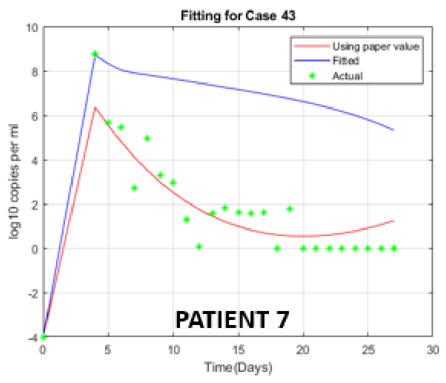
Best fitted graphs obtained for all the patients

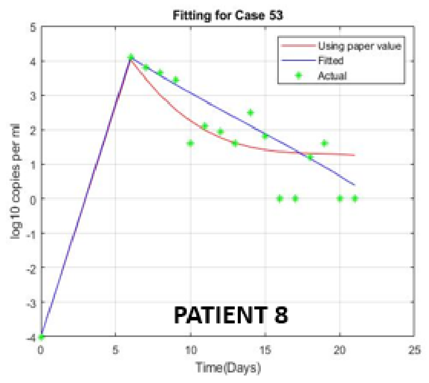


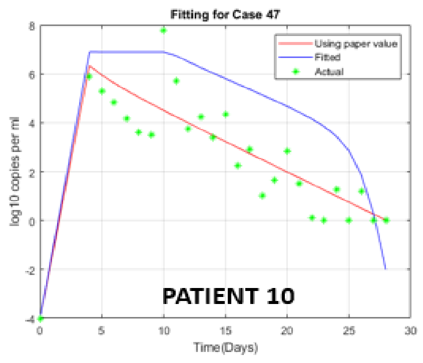


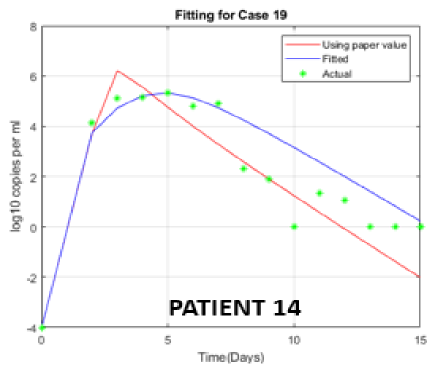












It can be observed that the curve is not passing through most data points for most patients. This is because of the presence of outliers. So, we then looked into methods to detect and replace outliers.

Curve fitting after data pre-processing

Since we were getting incorrect curve fitting due to the outliers, an inbuilt function in MATALB, filloutliers was used to detect & replace them. filloutliers(A,fillmethod,findmethod) detects outliers in A using the method specified in 'findmethod' and replaces them according to the method in 'fillmethod'.

We tested two kinds of methods to detect the outliers:-

1. **Quartiles**
   Outliers are defined as elements more than 1.5 interquartile ranges above the upper quartile (75 percent) or below the lower quartile (25 percent). This method is useful when the data is not normally distributed.

2. **Threshold [10 90]**
   A threshold of [10 90] defines outliers as points below the 10th percentile and above the 90th percentile.

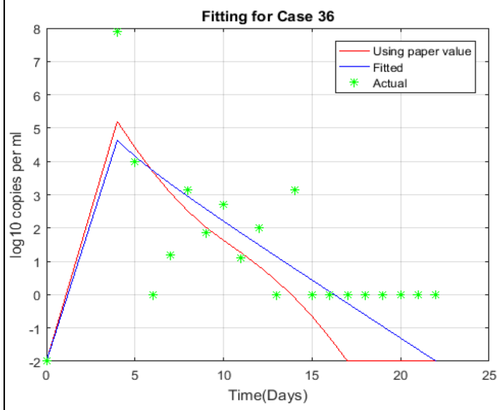
Upon running 60 cases for both of these for all the patients, it was concluded that Quartiles gave better results (based on error and visual representation).

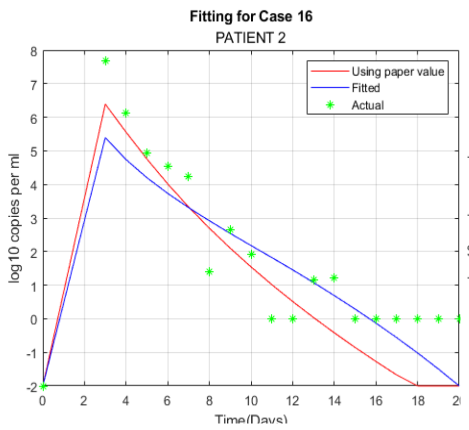
'clip' method was used to replace the outliers.

It fills with the lower threshold value for elements smaller than the lower threshold determined by findmethod. Fills with the upper threshold value for elements larger than the upper threshold determined by findmethod. (Code in Appendix D)

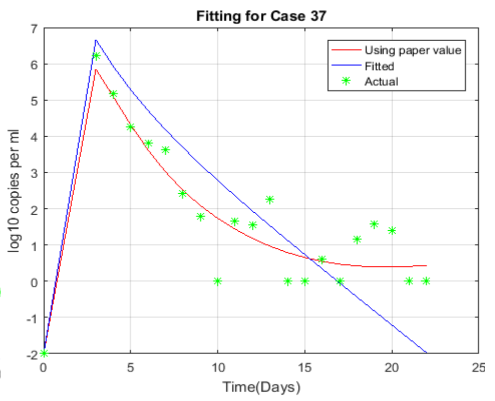
Best fitted graphs for all patients using Quartiles as outliers detection method

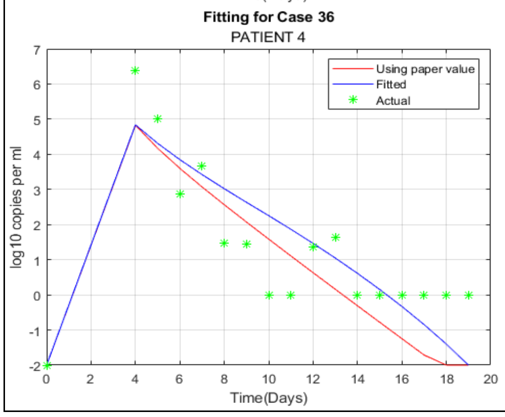
PATIENT 1

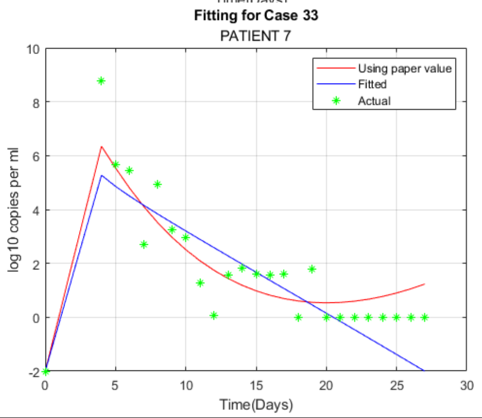


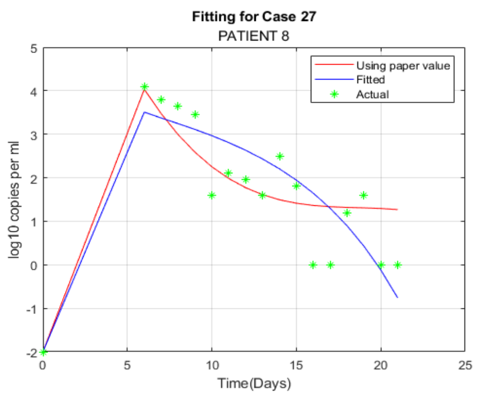


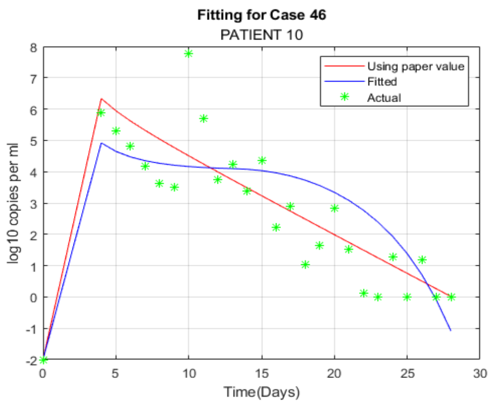
PATIENT 3

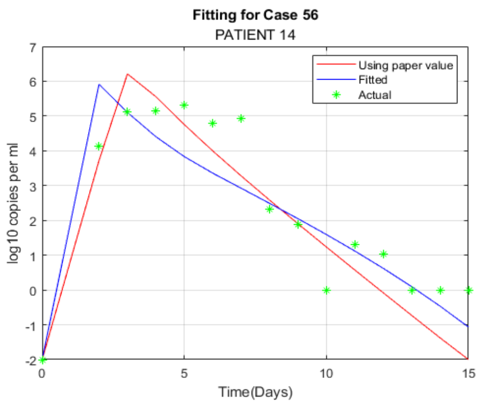












Parameter Values obtained for each patient after fitting

Patient	β (ml/virus/day)	σ (day−1)	μ (days)	ω (ml/cell/day)	p (day^-1)	c (day^-1)
1	0.0000165	0.0380706	14	0.0091126	14476.42	31.16571
2	1.01008E-05	0.1821148	10	0.01	78925.114	80.38738
3	0.000000299	0.02929	12	0.0011458	100000	26.77858
4	0.00000893	0.2386817	10	0.01	67283.691	60.80413
7	6.59E-06	0.0729415	6	0.0095497	29980.462	21.31287
8	0.0000786	0.2099361	6	0.0046554	1922.0378	32.17586
10	0.0000187	0.2685314	13	0.01	19134.965	35.65845
14	0.0000172	0.1666006	6	0.01	100000	171.9629

(Code in Appendix D)

Predicting Viral Dynamics under Treatment

We compared the effect on viral dynamics when treatment was started on different days after the symptoms started showing. We also observed the effect on viral dynamics upon using drugs with different efficacies. This drug would have two effects. Firstly, we would factor in its ability to eliminate the virus. Secondly, we would factor in it’s ability to prevent the target cells, pneunocytes from becoming effected. For these, we would make a few changes to the equations of the model used before.

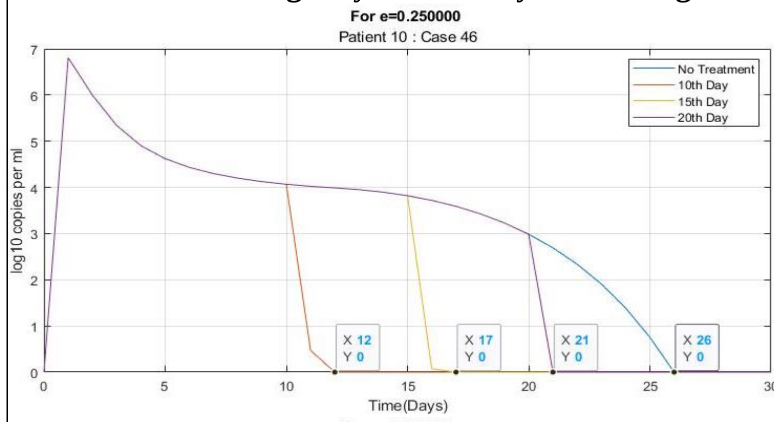
Modifications made to the equations of the model with secondary target of infection:-

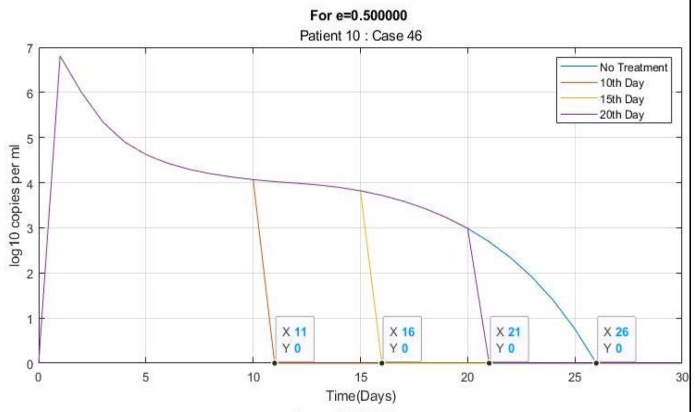
1. The 4th equation would be modified as follows for the days after the treatment is started:- dV/dt =((1-e)* pI) – cV, where e is the efficacy of the drug to eliminate virus particles

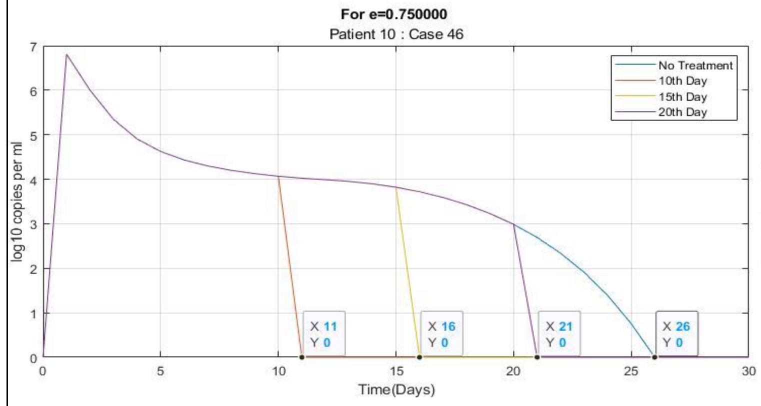
2. The 1st equation would be modified as follows for the days after the treatment is started:- $dT_1/dt = -\beta V\ T_1$(1-N), where N represents the efficacy of the drug to reduce the infection rate of pneumocytes ,i.e, preventing T_1 from becoming V.

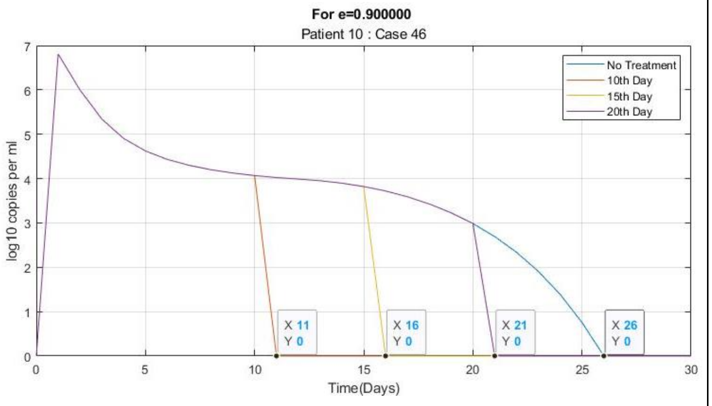
3. The third equation is modified as follows for the days after the treatment is started to account for the modified T_1:- $dI/dt = \beta V(T_1(1\text{-N}) + T_2) - [\delta (t) + \omega T_2]I$ (Code in Appendix E)

Considering only the efficacy of the drug to eliminate virus particles



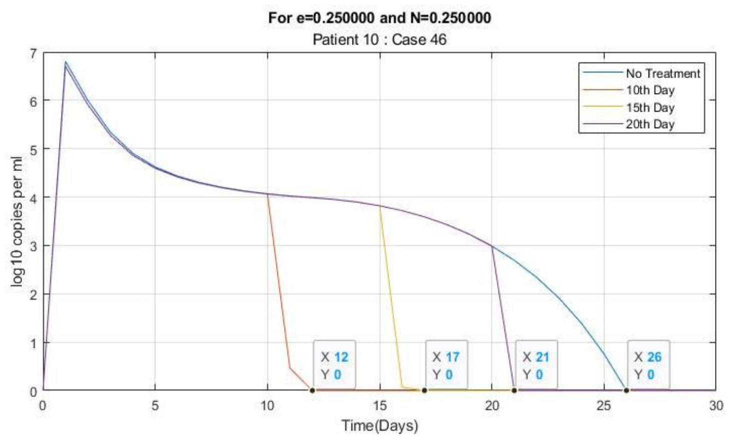


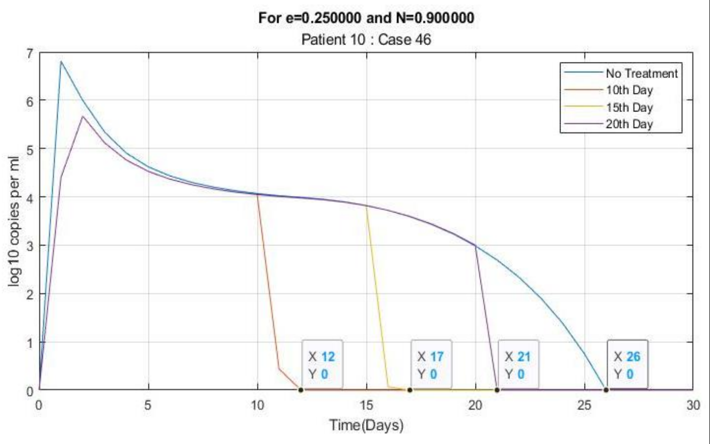


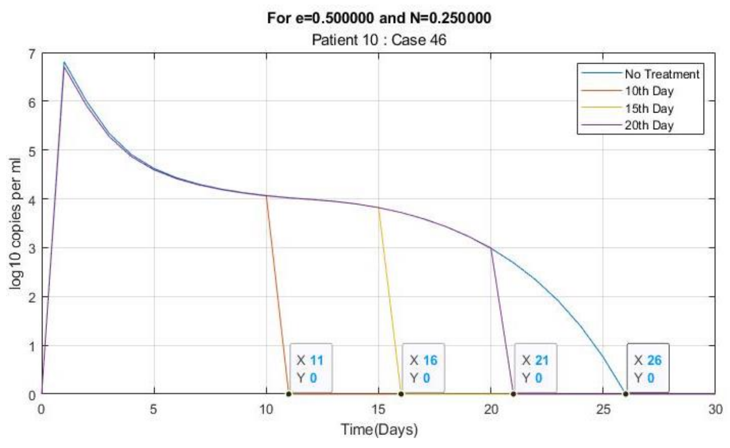


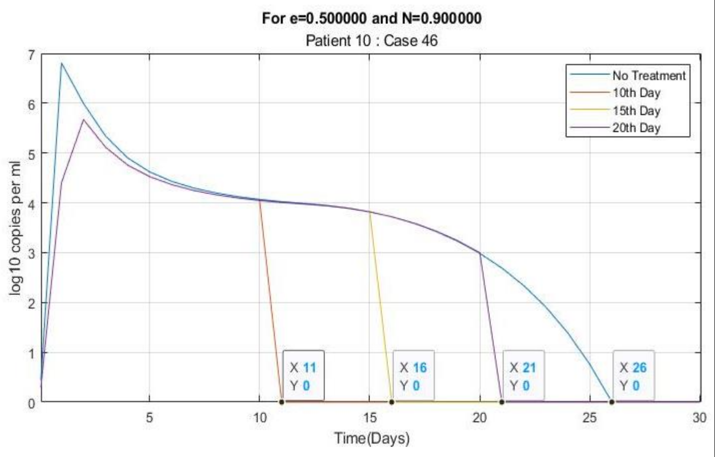
It can be observed that a drug with a higher efficacy would eliminate the virus faster.Starting the treatment earlier would eliminate the virus earlier for all cases.

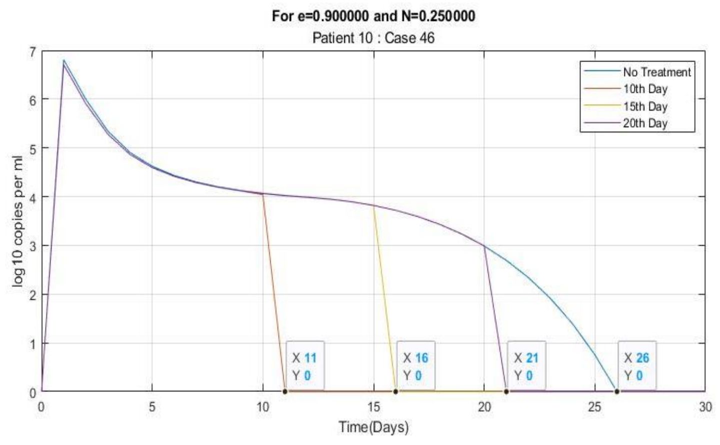
Considering both the effects of the drug

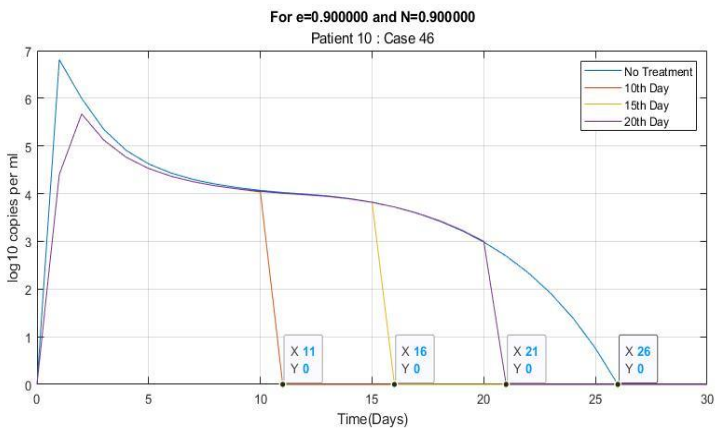












It can be observed that as the efficacy of the drug to reduce the infection rate of pneumocytes(preventing T_1 from becoming V) increases, we see a change in the shape of the curve. The peak of the Virus decreases, i.e., overall quantity of virus decreases. (Code in Appendix E)

Therefore, it can be concluded that higher efficacy drugs are better for eliminating the virus faster as well as decreasing the quantity of the virus. Starting the treatment earlier would eliminate the virus earlier.

References

(1) Sunpeng Wang,Yang Pan,Quanyi Wang, Hongyu Miao,Ashley N.Brown,Libin Rong, Modeling the viral dynamics of SARS-CoV-2 infection, (2020), https://doi.org/10.1016/j.mbs.2020.108438 .

(2) R. Wölfel, V.M. Corman, W. Guggemos, M. Seilmaier, S. Zange, M.A. Müller,et al., Virological assessment of hospitalized patients with COVID-2019, Nature (2020) http://dx.doi.org/10.1038/s41586-020-2196-x .

(3) https://in.mathworks.com/help/matlab/ref/ode45.html

(4) http://www.graphreader.com/ (Used to extract Data from the graphs)

(5) https://in.mathworks.com/help/optim/ug/lsqcurvefit.html

(6) https://in.mathworks.com/help/optim/ug/fmincon.html

(7) https://in.mathworks.com/help/matlab/ref/rmoutliers.html

(8) https://in.mathworks.com/help/matlab/data_analysis/data-smoothing-and-outlier-detection.html

(9) https://www.ruf.rice.edu/~bioslabs/tools/data_analysis/errors_curvefits.html

(10) https://statisticsbyjim.com/basics/outliers/

(11) https://www.who.int/emergencies/diseases/novel-coronavirus-2019

(12) https://www.fda.gov/media/136151/download#:~:text=The%20COVID%2D19%20RT%2DPCR%20Test%20is%20a%20real%2D,tract%20aspirates%2C%20bronchoalveolar%20lavage%2C%20and

APPENDIX (CODES)

Appendix A: Solving the Basic Viral Dynamic Model

Using ode45 to Solve the Basic Viral Dynamic Model

MATLAB CODES: -

Definition of the function: - This code would be the same for both the patients

```
%Declaration of the ODEs
function dy=temp(t,y)

global B D p c
dy=zeros(length(y),1);

dy(1)=p*y(3)-c*y(1);
dy(2)=(-1*B)*y(1)*y(2);
dy(3)=B*y(1)*y(2)-D*y(3);

    %y(1): Virus(V)
    %y(2): Target Cells(T)
    %y(3): Infected Cells(I)


    End
```

Code to solve the equations using ode45:-

This would be slightly different for both the patients because of the change in values of β,p and c.

Patient #1 :-

```
%Solving the ODEs using RK-Method

close all
clear all
global B D p c

B=0.01;
D=2;
p=50;
c=0.93;

tend=30; %30 days
tspan=0:tend;
y0=[0.0001 60000 0];
```

```
[t y]=ode45('temp',tspan,y0);
plot(t,y)
xlabel('Time(Days)')
ylabel('No. of copies per ml')
legend({'y1 = Virus','y2 = Target cells', 'y3=Infected
cells'},'Location','northeast')
```

Patient #8:-

```
%Solving the ODEs using RK-Method

close all
clear all
global B D p c

B=0.00091;
D=2;
p=2;
c=0.6;

tend=30; %30 days
tspan=0:tend;
y0=[0.0001 60000 0];
[t y]=ode45('temp',tspan,y0);
plot(t,y)
xlabel('Time(Days)')
ylabel('No. of copies per ml')
legend({'y1 = Virus','y2 = Target cells', 'y3=Infected
cells'},'Location','northeast')
```

Appendix B: Solving the Model with a Secondary Target of Infection

Using ode45 to solve the model with a secondary target of infection

MATLAB CODES:-

Function defining the ODEs :- This code would be the same for both the patients

```
 function dy=virus(t,y)

global B p c D w sig U L
dy=zeros(length(y),1);

dy(1)=(-1*B)*y(4)*y(1);
dy(2)=L-(B*y(4)*y(2));
dy(3)=B*y(4)*(y(1)+y(2))-(death(t)+(w*y(2)))*y(3);
dy(4)=p*y(3)-c*y(4);

     %y(1): T1
     %y(2): T2
     %y(3): I
     %y(4): V

     end
```

Function defining the death rate:-

```
function f=death(t)
global D sig U
f=D*exp(sig*(t-U));
end
```

Code to solve the equations using ode45:-

This would be slightly different for both the patients because of the change in values of β,p and c.

Patient #1 :-

```
%Solving the ODEs using RK-Method
close all
clear all
global B p c D w sig U L

B=1.1*10^(-5);
D=2; %given
p=940;
c=50;
w= 1.9*10^(-4);
sig=0.13;
```

```
U=10;
L=10000


tend=25; %25 days
tspan=0:tend;
y0=[60000 0 0 0.0001];
[t y]=ode45('virus',tspan,y0);
Y=log10(y)
Y(Y<0)=0

plot(t,Y)
xlabel('Time(Days)')
ylabel('log10 copies per ml')
legend({'T1 = Conc of Pneumocytes','T2 = conc of Lymphocytes', 'I=Infected cells','V=Virus'},'Location','northeast')
```

Patient #8:-

```
%Solving the ODEs using RK-Method
close all
clear all
global B p c D w sig U L

B=7*10^(-6);
D=2; %given
p=3100;
c=118;
w= 1.5*10^(-4);
sig=0.07;
U=6;
L=10000

tend=30; %30 days
tspan=0:tend;
y0=[60000 0 0 0.0001];
[t y]=ode45('virus',tspan,y0);

Y=log10(y)
Y(Y<0)=0

plot(t,Y)
xlabel('Time(Days)')
ylabel('log10 copies per ml')
legend({'T1 = Conc of Pneumocytes','T2 = conc of Lymphocytes', 'I=Infected cells','V=Virus'},'Location','northeast')
```

Appendix C: Curve Fitting with Lsqcurvefit

Curve fitting of the model with a secondary target of infection with lsqcurvefit

Function Defining the odes:-

```
function dy=odes(t,y,b)

%b is the array of constants that need to be fitted
B=b(1);
p=b(2);
c=b(3);
sigma=b(4);
U=b(5);
w=b(6);

      dy=zeros(length(y),1);

      dy(1)=(-1*B)*y(4)*y(1);
      dy(2)=10000-(B*y(4)*y(2));
      dy(3)=B*y(4)*(y(1)+y(2))-(die(t,b)+(w*y(2)))*y(3);
      dy(4)=p*y(3)-c*y(4);

      %y(1): T1
      %y(2): T2
      %y(3): I
      %y(4): V

      %b(1)=B(Beta)
      % b(2)=p; b(3)=c
      %b(4)=sigma
      %b(5)=U(mu)
      %b(6)=w(omega)
      End
```

Function used to do fitting :-

```
function X= odefit(b,t)
tend=30; %30 days
tspan=0:tend;
 y0=[60000 0 0 0.0001];
[t y]=ode45(@(t,y) odes(t,y,b),t,y0);
X=y(:,4);
End
```

Main code :-

```
clc
clear
%PATIENT 8

observations= [6    12589.25412
7   6309.573445
8   4466.835922
9   2818.382931
10  39.81071706
11  125.8925412
12  89.12509381
13  39.81071706
14  316.227766
15  63.09573445
16  1
17  1
18  15.84893192
19  39.81071706
20  1
21  1
];

t_obs = observations(:,1);
v1_obs = observations(:,2);
%parameters to be fitted
beta=7*10^(-6);
pp=3100;
cc=118;
omega= 1.5*10^(-4);
sigma=0.07;
mu=6;
%Initial concentrations
t1=60000;
t2=0;
I=0;
V=0.0001;
%Initial conditions
X0 = [beta;pp;cc;omega;sigma;mu];
lb=[0;0;0;0;0;0];
% %durataion of the experiment
 tstart = 4;
 tend = 22; %days
tspan = [tstart tend];

B = lsqcurvefit(@(b,t)odefit(b,t), X0, t_obs, v1_obs,lb);

figure
plot(observations(:,1), observations(:,2:end),'p')
hold on
plot(observations(:,1), odefit(B,observations(:,2)))
xlabel('Time(Days)')
ylabel('number of copies per ml')
hold off
grid
```

Appendix D: Final Curve Fitting using Fmincon

This was the Final code used to do curve fitting for the model with a secondary target of infection.

Defining the mathematical model

```
function dy = model(t,y,b)
%% This is the function that is used in ode45

% b(1) : beta
% b(2) : sigma
% b(3) : meu
% b(4) : omega
% b(5) : p
% b(6) : c

      dy(1) = -b(1)*y(4)*y(1); % T1

      dy(2) = 10^4 - b(1)*y(4)*y(2); % T2

      % This is the part euivalent to death function
      if(t >= b(3))
          dy(3) = b(1)*y(4)*(y(1) + y(2)) - (2*exp(b(2)*(t-b(3))) +
b(4)*y(2))*y(3); % I
      else
          dy(3) = b(1)*y(4)*(y(1) + y(2)) - (2 + b(4)*y(2))*y(3); % I
      end
      %

      dy(4) = b(5)*y(3) - b(6)*y(4); % V

      dy = dy';

      End
```

Non-linear Constraints

```
function [c,ceq] = constraints(x,limit,t_actual,y0,v_actual)

[~, conc] = ode45(@(t,y) model(t,y,x),t_actual,y0);
 v_ode = conc(2:end,4);

 c= ((-1)*v_ode)+limit;
ceq=[];
%

      end

```

Fitting Function

```
function sol = fitness_function(x,y0,t_actual,v_actual)
[~, conc] = ode45(@(t,y) model(t,y,x),t_actual,y0);
v_ode = conc(:,4);
sol =sum((v_ode - v_actual).^2);

end
```

Main

```
%%
%% CURVE FITTING BY GENERATING RANDOM VARIABLES AS INITIAL GUESSES FOR PATIENT 14
% Threshold

%%
%% This is the main function


subtit= sprintf('PATIENT 14');


    %%
    %% Parameters to be fitted

    % %Patient 1
    % beta = 1.1*10^(-5);
    % p = 940;
    % c = 50;
    % omega = 1.9*10^(-4);
    % sigma = 0.13;
    % meu = 10;

    % %Patient 2
    % beta = 2.2*10^(-6);
    % p = 4.8*10^(3);
    % c = 35;
    % omega = 3*10^(-4);
    % sigma = 0.01;
    % meu = 8;

    % %Patient 3
    % beta = 9.8*10^(-6);
    % p = 1.3*10^(3);
    % c = 58;
    % omega = 2*10^(-4);
    % sigma = 0.04;
    % meu = 9;


    % %Patient 4
    % beta = 4.5*10^(-5);
    % p = 88;
    % c = 5;
    % omega = 0.001;
    % sigma = 0.001;
    % meu = 6;


    % %Patient 7
    % beta = 9.9*10^(-7);
    % p =  1.08*10^(4);
    % c = 48;
    % omega = 2.1*10^(-4);
    % sigma = 10^(-4);
    % meu = 9;


    % %Patient 8
    % beta = 7*10^(-6);
    % p = 3100;
    % c = 118;
    % omega = 1.5*10^(-4);
    % sigma = 0.07;
    % meu = 6;


    % %Patient 10
    % beta = 1.8*10^(-6);
    % p = 4.9*10^(3);
    % c = 3.5;
    % omega = 3*10^(-3);
    % sigma = 0;
    % meu = 15;


    %Patient 14
    beta = 5.7*10^(-6);
    p = 800;
    c = 10;
    omega = 5.3*10^(-4);
    sigma = 0;
    meu = 6;


    %%
    %% The vector b stores all fitting parameters
    b(1) = beta;
    b(2) = sigma;
    b(3) = meu;
    b(4) = omega;
    b(5) = p;
    b(6) = c;
    %%
    %% Initial Conditions
    T1 = 60000;
```

```
T2 = 0;
I = 0;
V = 0.0001;

y0 = [T1;T2;I;V]; % Storing Initial Conditions
%%
%% Solving system of ODEs using ode45
t_start = 0;
t_end = 30;
t_span = [t_start:t_end];
[time, conc] = ode45(@(t,y) model(t,y,b),t_span,y0);
%%
%% Viral Load Data

% %Patient 1
% observations=[
% 4 79432823.4724
% 5 10000.0000
% 6 1.0000
% 7 15.8489
% 8 1412.5375
% 9 70.7946
% 10    501.1872
% 11    12.5893
% 12    100.0000
% 13    1.0000
% 14    1412.5375
% 15    1.0000
% 16    1.0000
% 17    1.0000
% 18    1.0000
% 19    1.0000
%  20   1.0000
%  21   1.0000
%  22   1.0000
% ];




% % Patient 2
% observations=[
%      3 49203953.57
% 4 1324341.535
% 5 86297.85478
% 6 35563.13186
% 7 17021.58508
% 8 25.70395783
% 9 457.0881896
% 10    83.75292821
% 11    1
% 12    1
% 13    14.25607594
% 14    16.51961798
% 15    1
% 16    1
%  17   1
% 18    1
% 19    1
% 20    1
% ];


% % %Patient 3
% observations=[
%     3 1640589.773
% 4 151008.0154
% 5 18323.14422
% 6 6683.439176
% 7 4216.965034
% 8 269.1534804
% 9 62.08690342
% 10    1
% 11    47.09773264
% 12    35.80964371
% 13    186.6379691
% 14    1
% 15    1
% 16    3.962780343
% 17    1
% 18    14.28893959
% 19    39.26449354
% 20    24.77422058
% 21    1
% 22    1
% ];

%
% %Patient 4
% observations=[
%     4 2371373.706
% 5 104712.8548
% 6 739.6052751
% 7 4623.810214
% 8 29.78516429
% 9 27.16439269
% 10    1
% 11    1
% 12    22.64644308
% 13    42.95364268
% 14    1
% 15    1
% 16    1
% 17    1
% 18    1
% 19    1
% ];


% %Patient 7
% observations=[
%     4 582103217.8
% 5 453941.6167
% 6 287078.0582
% 7 511.6818355
% 8 87096.359
% 9 1849.268619
% 10    887.156012
% 11    18.83649089
% 12    1.202264435
% 13    35.80964371
% 14    68.07693587
% 15    39.26449354
% 16    35.80964371
% 17    39.26449354
% 18    1
% 19    62.08690342
% 20    1
% 21    1
% 22    1
% 23    1
% 24    1
% 25    1
% 26    1
% 27    1
% ];


% %Patient 8
% observations= [
% 6 12589.25412
% 7 6309.573445
% 8 4466.835922
% 9 2818.382931
% 10    39.81071706
% 11    125.8925412
% 12    89.12509381
% 13    39.81071706
% 14    316.227766
% 15    63.09573445
% 16    1
% 17    1
% 18    15.84893192
% 19    39.81071706
% 20    1
% 21    1
% ];


% %Patient 10
% observations=[
%     4 787045.7897
% 5 199067.3339
% 6 66221.65037
% 7 15240.52754
% 8 4216.965034
% 9 3206.269325
% 10    58748935.25
% 11    497737.085
% 12    5559.042573
% 13    16710.90614
% 14    2432.204009
% 15    22029.26463
% 16    170.2158508
% 17    809.0958992
% 18    10.86425624
% 19    42.95364268
% 20    674.5280277
% 21    32.65878322
% 22    1.318256739
% 23    1
% 24    18.83649089
% 25    1
% 26    15.6675107
% 27    1
% 28    1
% ];


%Patient 14
observations=[
    2   13182.56739
3   131825.6739
4   144543.9771
5   208929.6131
6   63095.73445
7   83176.37711
8   208.9296131
9   75.8577575
10  1
11  20.89296131
12  10.96478196
13  1
14  1
15  1
];

%%
%%

t_actual = observations(:,1);
v_actua = [V;observations(:,2)];


%Case 1
%v_actual = filloutliers(v_actua,'clip','quartiles');
```

```

 %Case 2
threshold=[10 90];

 v_actual = filloutliers(v_actua,'clip','percentiles',threshold);



% count=0;
%
% for j=1:length(v_actual)
%     if(v_actual(j)~= v_actua(j))
%         count=count+1;
%     end
% end
%
% count





[time, conc_paper] = ode45(@(t,y) model(t,y,b),[0;t_actual],y0);

sol_rmse_paper = sum((conc_paper(:,4) - v_actual).^2);


er= sprintf('Error from the paper = %d \n',sol_rmse_paper);
disp(er);

options.Display = 'iter';
options.Algorithm = 'sqp';

%%
%%

%%Values of the parameters from the paper

% %Patient 1
% b0 = [1.1*10^(-5);0.13;10;1.9*10^(-4);940;50];

% %Patient 2
% b0 = [2.2*10^(-6);0.01;8;3*10^(-4);4.8*10^(3);35];

% %Patient 3
% b0 = [9.8*10^(-6);0.04;9;2*10^(-4);1.3*10^(3);58];
%
% %Patient 4
% b0 = [4.5*10^(-5);0.001;6;0.001;88;5];

% %Patient 7
% b0 = [9.9*10^(-7);0.0001;9;2.1*10^(-4);1.08*10^(4);48];

% %Patient 8
% b0 = [7*10^(-6);0.07;6;1.5*10^(-4);3100;118];

% %Patient 10
% b0 = [1.8*10^(-6);0;15;3*10^(-3);4.9*10^(3);3.5];

%Patient 14
b0 = [5.7*10^(-6);0;6;5.3*10^(-4);800;10];




disp('Initial Values for Case 1:-');
Xi = sprintf('Beta: %d \nSigma: %d \nMeu: %d \nOmega: %d \np: %d \nc: %d\n\n',b0(1),b0(2),b0(3),b0(4),b0(5),b0(6));
disp(Xi);

%%
%%
%% Ode Fitting

% This limit signifies the inequality where all the log10 values
%must be greater than or equal to -2
limit=0.01;


[x,sol_rmse_fitted] = fmincon(@(x)fitness_function(x,y0,[0;t_actual],v_actual),b0,[],[],[],[],[10^(-8);0;6;10^(-5);10;0.5],[0.0001;0.5;15;0.01;10^5;300],@(x)constraints(x,limit,[0;t_actual],y0,v_actual),options);
[time, conc_fitted] = ode45(@(t,y) model(t,y,x),[0;t_actual],y0); % We need it for the plot function

%%
disp('Final Value for Case 1 :-');
Xf = sprintf('Beta: %d \nSigma: %d \nMeu: %d \nOmega: %d \np: %d \nc: %d\n\n',x(1),x(2),x(3),x(4),x(5),x(6));
disp(Xf);

err0= sprintf('Error for Case 1 = %d ',sol_rmse_fitted);
disp(err0);

%Storing the Values properly in cells so that data can be observed in Excel
%easily
result{1,1}='Error';
result{1,2} ='Beta(initial)';
result{1,3} ='Beta(final)';
result{1,4} ='Sigma(initial)';
result{1,5} ='Sigma(final)';
result{1,6} = 'Meu(initial)';
result{1,7} = 'Meu(Final)';
result{1,8} = 'Omega(Initial)';
```

```
result{1,9} = 'Omega(Final)';
result{1,10} ='p (Initial)';
result{1,11} = 'p (Final)';
result{1,12} = 'c(Initial)';
result{1,13} = 'c(Final)';
result{1,14} = 'V values';

%Fitting data obtained for Case 1( using the actual value of parameters
%from the paper)
result{2,1} = sol_rmse_fitted; %Error
result{2,2} = b0(1);
result{2,3} = x(1);
result{2,4} = b0(2);
result{2,5} = x(2);
result{2,6} = b0(3);
result{2,7} = x(3);
result{2,8} = b0(4);
result{2,9} = x(4);
result{2,10} = b0(5);
result{2,11} = x(5);
result{2,12} = b0(6);
result{2,13} = x(6);
result{2,14} = conc_fitted(:,4); %V values

%%
%%

%Plotting the graph for Case 1

figure
xa=log10(conc_paper(:,4));
xa(xa<-2)=-2;
plot(time,xa,'r-')
hold on

xx=log10(conc_fitted(:,4));
xx(xx<-2)=-2;
xy=log10(v_actua);
xy(xy<-2)=-2;
plot(time,xx,'b-')
hold on
plot(time,xy,'g*')

title('Fitting for Case 1')
subtitle(subtit)

xlabel('Time(Days)')
ylabel('log10 copies per ml')
legend('Using paper value','Fitted','Actual')

hold off
grid

```

```


%%
%% Generating random Variables for the initial guesses

%bet= 10^(-8)+(0.0001-10^(-8)).*rand(1,60);
% sigm= 0+(0.5-0).*rand(1,60);
% me= randi([6 15],1,60); %We can only have integer values here
% omeg= 10^(-5)+(0.01-10^(-5)).*rand(1,60);
% pp= 10+(10^5-10).*rand(1,60);
% cc= 0.5+(300-0.5).*rand(1,60);
%
% rands=[ bet; sigm; me; omeg; pp; cc;];

%%
%% Fitting for different Initial Guesses ( Plots given at the end)

for r=1:60
    b0=rands(:,r);
    z=r+2;

i= sprintf('Initial Values for Case %d :-',r+1);
disp(i);
X0 = sprintf('Beta: %d \nSigma: %d \nMeu: %d \nOmega: %d \np: %d \nc: %d\n\n',b0(1),b0(2),b0(3),b0(4),b0(5),b0(6));
disp(X0);

[x,sol_rmse_fitted] = fmincon(@(x)fitness_function(x,y0,[0;t_actual],v_actual),b0,[],[],[],[],[10^(-8);0;6;10^(-5);10;0.5],[0.0001;0.5;15;0.01;10^5;300],@(x)constraints(x,limit,[0;t_actual],y0,v_actual),options);

[time, conc_fitted] = ode45(@(t,y) model(t,y,x),[0;t_actual],y0); % We need it for the plot function

f= sprintf('Final Values for Case %d :-',r+1);
disp(f);
X1 = sprintf('Beta: %d \nSigma: %d \nMeu: %d \nOmega: %d \np: %d \nc: %d\n\n',x(1),x(2),x(3),x(4),x(5),x(6));
disp(X1);

err= sprintf('Error for Case %d = %d \n\n',r+1,sol_rmse_fitted);
disp(err);

%Storing the data obtained
result{z,1} = sol_rmse_fitted;
result{z,2} = b0(1);
result{z,3} = x(1);
result{z,4} = b0(2);
result{z,5} = x(2);
result{z,6} = b0(3);
result{z,7} = x(3);
result{z,8} = b0(4);
```

```
result{z,9} = x(4);
result{z,10} = b0(5);
result{z,11} = x(5);
result{z,12} = b0(6);
result{z,13} = x(6);
result{z,14} = conc_fitted(:,4);

t= sprintf('Fitting for Case %d',r+1);

%%

% Plots for each case

figure
xa=log10(conc_paper(:,4));
xa(xa<-2)=-2;
plot(time,xa,'r-')
hold on

xx=log10(conc_fitted(:,4));
xx(xx<-2)=-2;
xy=log10(v_actua);
xy(xy<-2)=-2;
plot(time,xx,'b-')
hold on
plot(time,xy,'g*')
title(t)
subtitle(subtit)
xlabel('Time(Days)')
ylabel('log10 copies per ml')
legend('Using paper value','Fitted','Actual')
hold off

grid

end

%%
```

Appendix E: Viral Dynamics Under Treatment

CODE FOR VIRAL DYNAMICS UNDER TREATMENT

Original Model Function

```matlab
function dy = model(t,y,b)
%% This is the function that is used in ode45

% b(1) : beta
% b(2) : sigma
% b(3) : meu
% b(4) : omega
% b(5) : p
% b(6) : c

    dy(1) = -b(1)*y(4)*y(1); % T1

    dy(2) = 10^4 - b(1)*y(4)*y(2); % T2

    % This is the part euivalent to die.m
    if(t >= b(3))
        dy(3) = b(1)*y(4)*(y(1) + y(2)) - (2*exp(b(2)*(t-b(3))) + b(4)*y(2))*y(3); % I
    else
        dy(3) = b(1)*y(4)*(y(1) + y(2)) - (2 + b(4)*y(2))*y(3); % I
    end
    %

    dy(4) = b(5)*y(3) - b(6)*y(4); % V

    dy = dy';

    end

```

Modified Model Function :-

```matlab
function dy = model_with_drug(t,y,b,e,day,N)
%% This is the function that is used in ode45



% e is the efficacy to eliminate the virus
% N represents the efficacy of the drug to block the infection ( Prevent T1 from
% becoming V)

    % b(1) : beta
    % b(2) : sigma
    % b(3) : meu
    % b(4) : omega
    % b(5) : p
    % b(6) : c

    if( t<day)

    dy(1) = -b(1)*y(4)*y(1); % T1

    dy(2) = 10^4 - b(1)*y(4)*y(2); % T2

    % This is the part euivalent to die.m
    if(t >= b(3))
        dy(3) = b(1)*y(4)*((y(1)) + y(2)) - (2*exp(b(2)*(t-b(3))) + b(4)*y(2))*y(3); % I
    else
        dy(3) = b(1)*y(4)*((y(1)) + y(2)) - (2 + b(4)*y(2))*y(3); % I
    end
    %

        dy(4) = b(5)*y(3) - b(6)*y(4); % V

    else


     dy(1) = -b(1)*y(4)*y(1)*(1-N); % T1
        dy(2) = 10^4 - b(1)*y(4)*y(2); % T2

        if(t >= b(3))
        dy(3) = b(1)*y(4)*((y(1)) + y(2)) - (2*exp(b(2)*(t-b(3))) + b(4)*y(2))*y(3); % I
    else
        dy(3) = b(1)*y(4)*((y(1)) + y(2)) - (2 + b(4)*y(2))*y(3); % I
        end
    dy(4) = (1-e)*b(5)*y(3) - b(6)*y(4);

    end



    dy = dy';

    end


```

Main :-

```
clc;
clear;

s='Patient 10 : Case 46';

efficacy=[0.50 0.95];

%Different efficacies of antiviral drug
drugs=[0.25 0.5 0.75 0.9];

      e=0.25;

      %Days on which the treatment was started
      days=[10 15 20];

      %%
      %%

      %Parameter Values for the patient
      beta = 0.0000187 ;
      p = 19134.96455;
      c = 35.65844795;
      omega = 0.01;
      sigma = 0.268531406;
      meu = 12;


      %Vector storing the parameter values
      b(1) = beta;
      b(2) = sigma;
      b(3) = meu;
      b(4) = omega;
      b(5) = p;
      b(6) = c;

      %%
      %% Initial Conditions
      T1 = 60000;
      T2 = 0;
      I = 0;
      V = 0.0001;

      y0 = [T1;T2;I;V]; % Storing Initial Conditions

      %%
      %% Solving system of ODEs using ode45
      t_start = 0;
      t_end = 30;
      t_span = [t_start:t_end];

      %Running the loop for different efficacies
```

```matlab


    for j=1:length(drugs)

        N=drugs(j); %Efficacy

    %Without Treatment
    [time, vir] = ode45(@(t,y) model(t,y,b),t_span,y0);

    figure
    xa=log10(vir(:,4));
    xa(xa<0)=0;
    plot(time,xa)
    hold on

    %%
    %%

    %Running the loop for different days on which the treatment was started
    for i=1:3
        day=days(i); %different days on which the treatment was started

    %with Treatment
    [time, vir_d] = ode45(@(t,y) model_with_drug(t,y,b,e,day,N),t_span,y0);

    xx=log10(vir_d(:,4));
    xx(xx<0)=0;
    plot(time,xx)
    hold on


    end

    %%
    %%

    t= sprintf('For e=%f and N=%f',e,N);

    title(t)
    subtitle(s)
    xlabel('Time(Days)')
    ylabel('log10 copies per ml')
    legend('No Treatment ','10th Day','15th Day','20th Day')
    hold off
    grid

    end


```