INTRODUCTION TO FOCUS AREAS WS 22/23

Report: Complex Systems

Abhinav Mishra, Jule Brenningmeyer, Maike Herkenrath*

Correspondence:
abhinav.mishra@fu-berlin.de
Department of Mathematics and Computer Science, Freie Universität-Institute of Computer Science, Takustraße 9, 14195 Berlin, Germany
Full list of author information is available at the end of the article
*Group 7

Abstract

We know perfectly well that, an ODE might be wrong, and hence, we consider *stochastic dynamics* to be the **ground truth**, relatively. The modelling, simulation, and inference of a well-mixed dynamical system are part and parcel for the study of complex systems.

We implemented, and estimated the unknown parameters for within-host dynamical system of viral infection in the human body, using both, built-in (*continuous* and *deterministic*), and user-defined (*discrete*) optimization functions, i.e. *curve_fit*, *solve_ivp* (ODE-solver), and *SSA* algorithm in *python*. The distribution of optimized parameters played a key role in parameter identification, and the time-series formalism for infection probability.

Overall, 25 hours were spent on this project.

Keywords: ODE simulation; Stochastic; Dynamical systems; Chemical reactions

Scientific Background

Viral infections can spread differently strong and infect thereby more or less healthy cells. To get a deeper understanding of the dynamic of a virus infection, it can be modeled by using reaction rates, a stoichiometric matrix, reaction parameters and initial states. A possible way of generating the model is to classify the cells in 2 groups: *uninfected target cells*, and *infected cells*. For each group, a variable is generated and an additional one for the free virus. [1][2]

The human body produces cells, which can be infected by the virus. Hence, the virus RNA is transferred into the human cell, which leads to the production of more virus by the infected cell. After producing the virus, the cell releases the virus to the surrounding. To prevent the virus from spreading, humans have an immune system, which initiates the apoptosis of infected cells. Also non infected cells can initiate apoptosis under different circumstances. For each of these processes, a rate is needed to create a model. [1][2]

Homework 1

Simulation, Optimization, Programming

Goal

1 Implement the viral infection model model in such as way, that the unknown parameters are arguments that you pass to your reaction rate function, right-hand-side function (set of ODEs), as well as to an ODE-solver. Set up the ODE-solver, such that the simulation results corresponding to the provided data.

2 Estimate all unknown parameters of the aforementioned model using the data provided by you.
3 Perform the parameter estimation 30 times with random start parameters. Collect the inferred 'optimal parameters' and make a boxplot.

Methods

The program was written in *python*, a high-level programming language, which focuses on readability. It is equipped with many libraries that extend the capabilities of the language [3]. The libraries *numpy*, *pandas*, *seaborn*, *matplotlib*, *scipy* and *sklearn* were used. *numpy* is a library for scientific computing, providing support for multidimensional arrays and more [4]. The library pandas offers the function *DataFrame*, which was used to generate a dataframe from a dictionary [5]. *matplotlib* is an extensive library for the creation of visualisations in Python [6]. The library *seaborn* is based on *matplotlib* and was used to visualize data [7]. *scipy* is a library which offers functions, which can be used for scientific computing [8]. The *MinMaxScaler* of the open source library *sklearn*, for predictive data analysis, was used for nomraliztion [9].

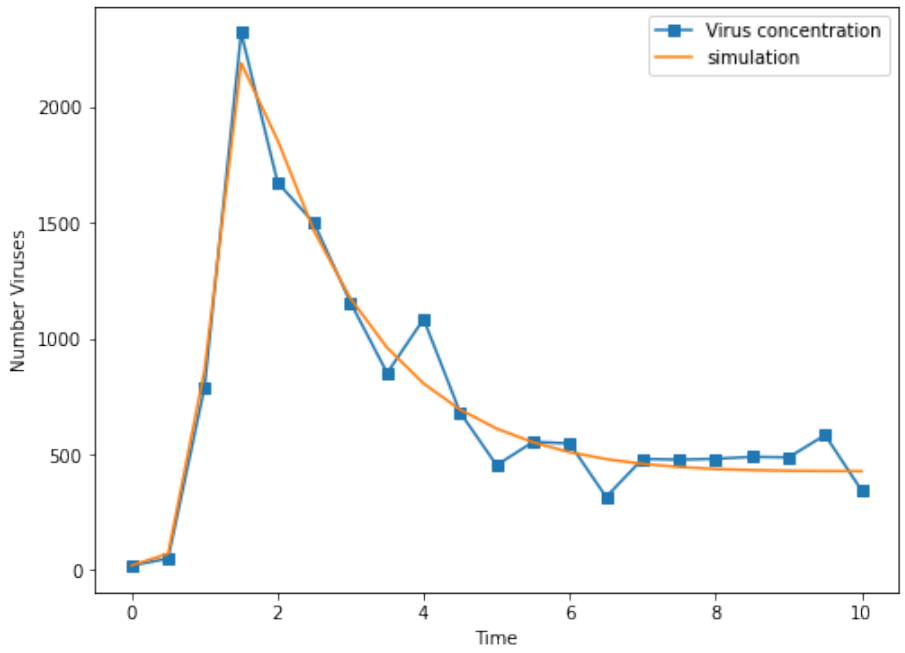

Figure 1 Viral concentrations data (blue squares), as well as model prediction with estimated parameters c_2, c_3, c_4, c_5 (orange line)

Results & discussion

The function *curve_fit* from *scipy* was used to calculate the missing parameters. Therefore the initial guesses $[0.1, 1, 5, 1]$ were used for the parameters. The following values for the parameters resulted from the calculation: $c_2 = 0.00, c_3 = 0.65, c_4 = 42.73, c_5 = 14.16$. These values were used to predict the number of viruses and compared in a plot with the original data of the viral concentration (Figure 1). The

curves of the original data and the prediction are mostly overlapping and therefore, it can be concluded that the model prediction with the before estimated parameters is a good approximation of real virus concentration. After tweaking the *curve_fit*

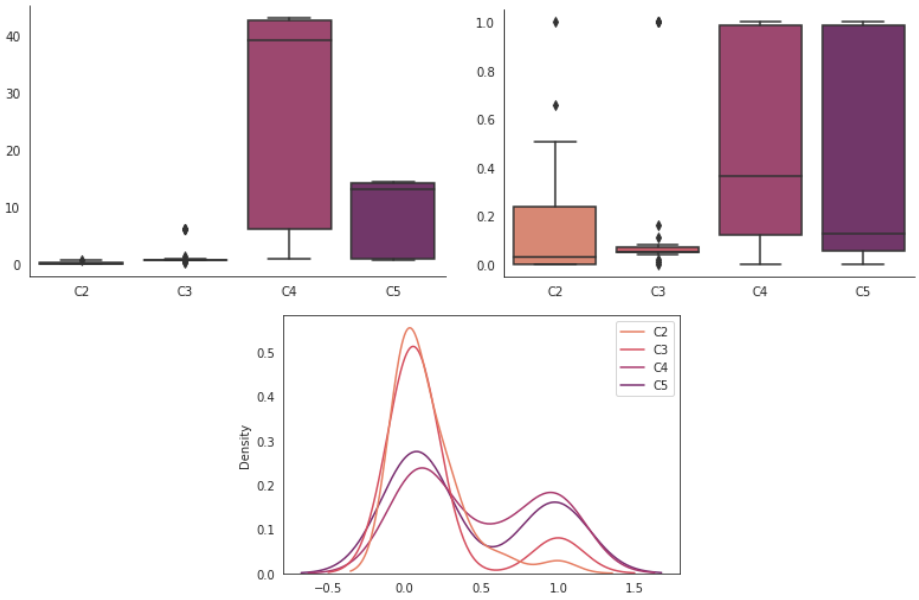

Figure 2 Boxplots for standard (top-left), and normalized (top-right) values, and density plot for normalized values of estimated parameters c_2, c_3, c_4, c_5 (bottom) . Normalization was done using MinMaxScaler from library sklearn

function with random start parameters, we collected the inferred parameters after 30 simulations, and plotted them in a boxplot with and without normalization (Figure 2), where each boxplot depicts the distribution of obtained parameter values for $c_2, c_3, c_4, \& c_5$. Normalization was done using *MinMaxScaler* from library *sklearn* [10].

The parameters c_2 and c_3 have outliers before and after the normalization, while c_5's median is highly skewed to the minima after the normalization. The leak of outliers of the normalized boxplot of c_4 indicates a normal distribution and the reasonable central median is an indication for a symmetrical distribution. In conclusion, out of the four parameters the results for the parameter c_4 are more equally distributed. However, the inter-quartile ranges of c_4 and c_5 are large, which means that we have a larger range in which the true value of the parameters probably lies compared to c_2 and c_3. Therefore, these two parameters can be estimated the least exact and consequently the least well.

Based on the evidence above, c_4 or c_5 should be the parameter to be determined. Hence c_5 is less equally distributed than c_4, the results of c_5 presumably give a better tendency of the true value. For that reason c_4 should be determined in a further experiment. After the biological experiments, the same should be fixed resulting in $c_4 = 10$. Therefore, due to parameter adjustment in the refactored arguments, the remaining parameters $c_2 = 0.02, c_3 = 2.20$ and $c_5 = 0.82$ estimation was satisfactory compared to the previous one as it reduced parameter uncertainty/error in solving ODE, and simulating 30 times (Figure 3). As *curve_fit* used non-linear least

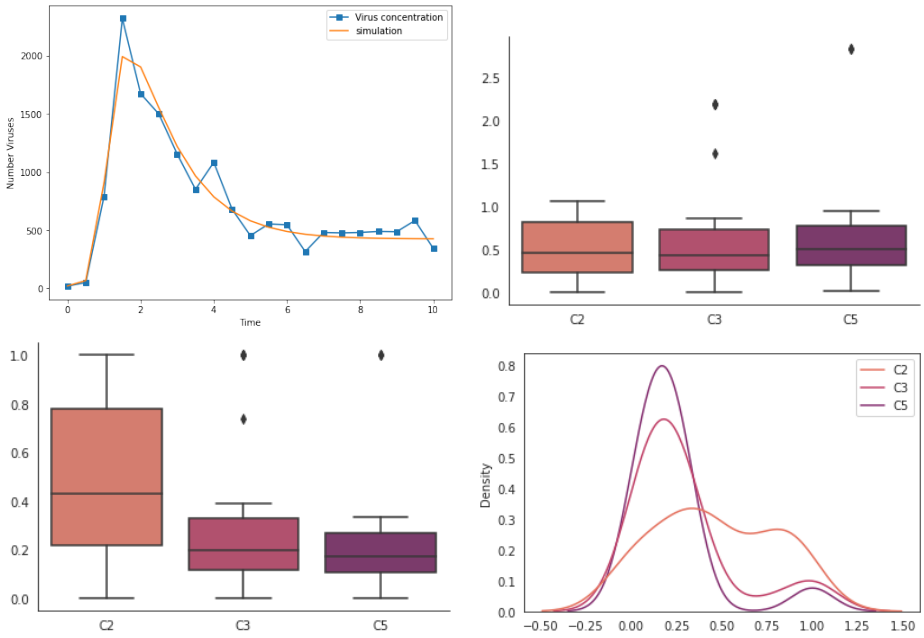

Figure 3 Viral concentrations data (blue squares), as well as model prediction with estimated parameters c_2, c_3, c_5 (orange line)(top-left). The boxplots for standard (top-right), and normalized (bottom-left) values, and density plot (bottom-right) for normalized values of estimated parameters c_2, c_3, c_4, c_5. Normalization was done using MinMaxScaler from library sklearn

squares (also known as *Levenberg-Marquardt* method) to fit a function, and refining parameters after each iteration, so the covariance matrix, and it's inverse solution becomes more efficient, and faster with less residual error.

Curve-fitting is to get the values for a dataset through which a given set of explanatory variables can actually depict another variable. This is the reason that altered our ability to estimate the remaining three parameters, most likely.

Homework 2

Prediction, Programming, Analysis

Goal

1 Perform stochastic simulations with the *SSA* algorithm to study how the infection probability depends on the number of viruses that an individual is exposed with, using the model from *Homework 1*.

2 Calculate the *Exposure* - *Infection* Probability based on 300 stochastic simulations.

3 Plot the infection probability (*y-axis*) as a function of the viral exposure (*x-axis*).

Methods

For the five values $X_2 \in [1, 2, 3, 4, 5]$ the SSA algorithm was executed 300 times and from the resulting trajectories infection probabilities were computed for each

of the values of X_2. We used the code for the SSA algorithm from previous assignments with some adjustments. For example, the stop criterion was not a given time threshold, but dependent on the values of X_1 and X_2.

Results & Discussion

For this task, we used the provided parameters $k_0 = 100, k_1 = 0.1, c_2 = 0.01, c_3 = 1.0, c_4 = 10, c_5 = 2$ that made it possible for us to be able to work on the task without depending on the results for the previous one, primarily. Especially, since we always got different results for task 1 based on the machine used to run the code, the python version used or even the scipy version that was installed. It was a problem to find the right balance between deterministic, and stochastic.

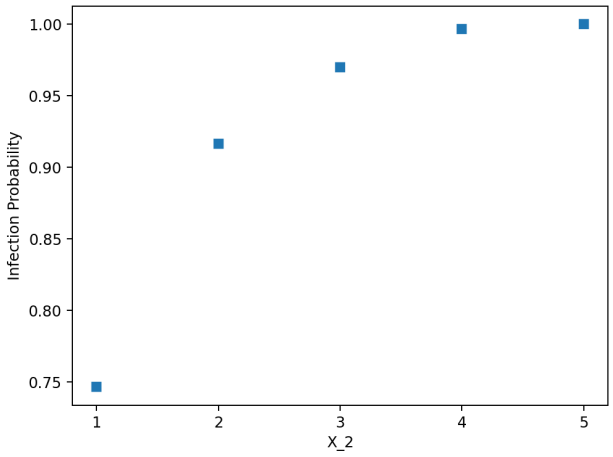

Figure 4 Infection probability (y-axis) as a function of the viral exposure (x-axis)

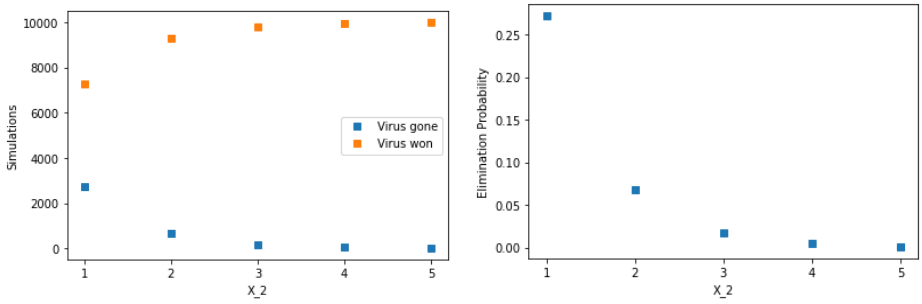

Figure 5 Viral trend after 10,000 simulations (left), and Elimination probability as a function of the viral exposure (x-axis) (right)

For the criteria to determine whether the virus has been eliminated, we concluded that an infection is eliminated when there are no more infected cells and also there are no more free virus in the system. Thus, the values $X_1 = 0$ and $X_2 = 0$ have to be achieved to be counted as an eliminated infection. We can see an increasing upward trend in the infection probability plot as a function of the viral exposure (Figure 4), so the elimination probability must have a downward trend (hyperbolic).

Furthermore, we can extrapolate the elimination probability after exposure with a single virus.

If the infection probability after exposure with a single virus $P_{inf}(X_2(1)) = 0.72$ then, the elimination probability after exposure with a single virus $P_{elim}(X_2(1)) = 1 - 0.72 = 0.28$ (Figure 5). After 10,000 simulations, we came up with the following formula that estimates the elimination probability after exposure with n viruses, based on the elimination probability ($P_{elim}(X_2(1)) = 0.28$) after exposure with a single virus:

$$P_{elim} \propto 0.28 * n^{-1}$$

Abbreviations
ODE: Ordinary differential equations, **SSA**: Stochastic simulation algorithm, **RNA**: Ribonucleic acid

Competing interests
The authors declare that they have no competing interests.

Author's Contributions (Group 7)
1 **Abhinav Mishra** edited, and wrote parts in the report. He also worked on the implementation of homework 1.
2 **Jule Brenningmeyer** wrote the section scientific background and parts of homework 1. She worked on the code for homework 1.
3 **Maike Herkenrath** worked on the code for homework 1 and 2. She also added parts in the report.

References

1. Boianelli, A., Nguyen, V.K., Ebensen, T., Schulze, K., Wilk, E., Sharma, N., Stegemann-Koniszewski, S., Bruder, D., Toapanta, F.R., Guzmán, C.A., *et al.*: Modeling influenza virus infection: a roadmap for influenza research. Viruses **7**(10), 5274–5304 (2015)
2. von Kleist, M.: 4. homework & report (complex systems block) introduction to focus areas ws 2022/23 (2022)
3. Welcome to python.org. doi:https://www.python.org/
4. Numpy. doi:https://numpy.org/
5. pandas.dataframe - pandas 1.5.2 documentation. doi:https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html
6. Matplotlip - visualisation with python. doi:https://matplotlib.org/
7. seaborn: statistical data visualization- seaborn 0.12.2. doi:https://seaborn.pydata.org/
8. Scipy. doi:https://scipy.org/
9. scikit-learn: machine learning in python. doi:https://scikit-learn.org/stable/
10. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)