University of Amsterdam

Masters Thesis

Mathematically Modeling the Interactions Between Phages, Bacteria, and the Environment

Examiner:
Dr. Jaap Kaandorp
Author:
Victor Piaskowski
Supervisor:
Dr. Matti Gralka
Assessor:
Dr. Yuval Mulla

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computational Science

in the

Computational Science Lab
Informatics Institute

May 2025

Declaration of Authorship

I, Victor Piaskowski, declare that this thesis, entitled 'Mathematically Modeling the Interactions Between Phages, Bacteria, and the Environment' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at the University of Amsterdam.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date: May 29, 2025

"All models are wrong, but some are useful"

George E. P. Box

UNIVERSITY OF AMSTERDAM

Abstract

Faculty of Science
Informatics Institute

Master of Science in Computational Science

Mathematically Modeling the Interactions Between Phages, Bacteria, and the Environment

by Victor Piaskowski

Include your abstract here Abstracts must include sufficient information for reviewers to judge the nature and significance of the topic, the adequacy of the investigative strategy, the nature of the results, and the conclusions. The abstract should summarize the substantive results of the work and not merely list topics to be discussed. Length 200-400 words.

Acknowledgements

I would like to thank my parents for loving me despite some of my faults, and for supporting me through my Bachelor and Master studies even if they don't exactly know what I am studying. Without them, I wouldn't know where my life would be right now, and I certainly would be a different person if it were not for them.

Thank you to Dr. Matti Gralka for the weekly meetings and teaching me everything about phages and bacteria. Every meeting was always insightful, productive, and informative. I will forever be amazed at how he can remember which paper talks about which topic, and how he always had a paper for every topic.

Thank you to Sofia Blaszczyk for finding the Master thesis opening and suggesting that I email Dr. Gralka for an introductory meeting. She acted as my rubber duck programming buddy, and watched my cringe screen recordings that I sent her at 2am showcasing various demos of my project. If she wouldn't have found this opening, I wouldn't know what I would be doing as my thesis.

If I hadn't followed Dr. Rik Kaasschieter's and Dr. Martijn Anthonissen's courses "Introduction Computational Sciences" and "Numerical Linear Algebra" in my Bachelors, I would not have been interested in Computational Sciences. I would not have found the MSc Computational Sciences program, as Computational Sciences fits my interests and skill sets better than any other program I could have taken. For Rik and Martijn have forever altered my career trajectory.

Thank you to Sarah Flickinger for showing me the research that she has been doing in the lab. She allowed me to really connect my research and models to real life, reminding me that what I am doing has real life use cases than just a purely theoretical or programming challenge.

And finally, thank you to all of my friends for keeping me sane and helping me through both of my programs.

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

ABM	**A**gent **B**ased **M**odelling
ARD	**A**rms **R**ace **D**ynamic
BVP	**B**oundary **V**alue **P**roblem
CBASS	**C**yclic oligonucleotide-**B**ased **A**ntiphage **S**ignalling **S**ystems
CRISPR	**C**lustered **R**egularly **I**nterspaced **S**hort **P**alindromic **R**epeats
DDE	**D**elay **D**ifferential **E**quation
DNA	**D**eoxyribo**N**ucleic **A**cid
FSD	**F**luctuating **S**election **D**ynamics
GUI	**G**raphical **U**ser **I**nterface
IC	**I**nitial **C**ondition
IVA	**I**nitial **V**alue **A**nalysis
OD	**O**ptical **D**ensity
ODE	**O**rdinary **D**ifferential **E**quation
PA	**P**arameter **A**nalysis
PDE	**P**artial **D**ifferential **E**quation
PP	**P**hase **P**ortrait
RNA	**R**ibo**N**ucleic **A**cid
SIE	**S**uper**I**nfection **E**xclusion
SNP	**S**ingle **N**ucleotide **P**olymorphism
ST	**S**erial **T**ransfer
TAB	**T**ail **A**ssembly **B**locker
UA	**U**ltimate **A**nalysis
UvA	**U**niversitiet **v**an **A**msterdam

Chapter 1

Introduction

Phages are small viruses on the order of 27-190nm that infect and lyse (kill) specific bacteria, acting as nature's natural anti-microbial defense. Researchers are attempting to determine how phages can be used in various medical and industrial applications to control bacterial growth. However, researchers need to know how the interactions between phages and bacteria work in order to implement a robust method to control bacterial growth.

1.1 Biological Background

Phages are small viruses on the order of 27-190nm that infect and lyse (kill) specific bacteria. The phage cycle process starts with a phage coming into contact with a bacterium. Once it has identified an injection site, the phage can inject a strain of DNA into the bacteria. The DNA strand has two options: it can either merge into the bacterial DNA, allowing the phage's DNA strand to replicate alongside the bacteria as they reproduce. This process defines the Lysogenic cycle. After a set amount of time, the DNA of the phage can unmerge and hijack the DNA replicating mechanism, creating multiple copies of itself, using the transcription, translation, and replication process to create multiple copies of itself. The phages begin to self-assemble inside the bacteria until the bacteria is full of phages and explodes, the lysis stage, releasing the phages into the environment, ready to repeat the process again.

This process can be visualized in Figure 1.1.

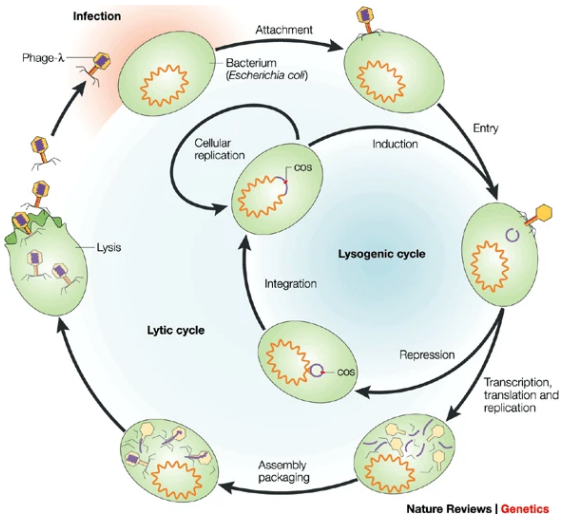


Figure 1.1: Life cycle of a phage, inside and outside a bacteria cell. Significant steps in the life cycle of a phage include the infection stage, integration, replication, and lysing process. Figure sourced from Campbell [1].

1.2 Phage Cocktails and Human Health

There is particular interest in phage applications in human and animal health, called phage cocktail therapy, due to phage cocktails not exhibiting side effects. Phage cocktails are a medicine that sick patients with bacterial diseases, such as *Escherichia coli* can use. A patient can swallow a pill filled with a range of different phages that target *E. coli*. The phages will target the specific *E. coli* bacteria, but it will not affect the other bacteria found in the gut of the human body and will not have any side effects on the body. There are about 100 trillion microbes across 5,000 different types of bacteria strains in the human gut. Antibiotics disrupt the intricate ecosystem of the gut microbiome, acting as a scorched-earth mechanism. Phages on the other hand specifically target a specific bacterial strain, acting as a sniper, with minimal to no effects to other bacterial strains, while antibiotics act as a bomb. A challenge that antibiotics face is that antibiotics create antibiotic resistant bacteria, making the antibiotic less effective in the future [9, 10]. There is however hope that phage resistant bacteria become more susceptible to antibiotics due to changes in the cell structure [11, 12]. Phage Therapy and Antibiotics in Appendix B: Industrial and Real Life Applications of Phages goes more in depth on how phages can be used in a healthcare setting.

1.3 Industrial Usage

Phages have many uses in an industrial setting. Similarly, phage therapies can be used as a preventative method, by preventing the spread of common bacteria in livestock by

dosing the animal feed with the phage pills. Farmers often raise livestock in tight spaces with a lack of sanitation facilities, increasing the risk of a disease spreading.
Phages can be used to control the growth of bacteria like *Salmonella* while producing food in a factory [6, 13]. Controlling Foodborne Bacteria in Appendix B: Industrial and Real Life Applications of Phages goes into more detail about using phages to control foodborne bacteria.

1.4 The Environment

In an ecosystem like the ocean, the gut, or in soil, there are thousands of different microbes all interacting with one another or the surrounding environment. The interactions are complex, with many factors affecting the growth of bacteria, fungi, phages, plants, animals, and more. Often, the interactions between agents in the environment are synergistic. When an animal dies, bacteria start to digest and decompose the animal into simpler chemicals like carbon and nitrogen that plants can use to grow, which is then eaten by other animals. Finally, phages can potentially be used to control *Cyanobacteria* (blue-green algae) blooms in the environment and affect other agents such as plankton in the environment [14]. *Cyanobacteria* cause damage to aquatic life by consuming resources and oxygen, starving aquatic life. *Cyanobacteria* can also affect human health by infecting drinking water. Having the ability to control *Cyanobacteria* growth can save nature and protect human health. With this, there is hope that water quality can be engineered without using harsh chemical processes what would otherwise pose environmental and health hazards [15].
More information about controlling *Cyanobacteria* can be read in Environmental Protection.

External factors, such as flooding, droughts, chemical spills, or introduction of new agents have a massive impact on the ecosystem. These events can add or remove resources from the system, change environmental parameters such as the surrounding temperature, introduce competition, or create an imbalance in the population by killing agents. These effects have a larger effect on the ecosystem and food chain as a whole as bacteria are one of the fundamental foundations for resource recycling.

1.5 Modelling Phages in a Complex Community

Not much is known about phages in large and complex communities between other phages, bacteria, resources, and the environment. There have been previous attempts

to model the complex dynamics of the populations between phages, bacteria, and resources, with the environment using Ordinary Differential Equations (ODE) and Delay Differential Equations (DDE). Not every interaction in the complex community can be identified, and if an interaction has been identified, the associated parameter values are unknown and need to be experimentally derived. Collecting interaction parameter values is an expensive and laborious task, as the data has to experimentally collected.

There are two main ways to model phage-bacteria dynamics: spatially and non-spatially. In a spatial model phages and bacteria can move through space and interact with their neighbors. Partial differential equations (PDE) and cellular agent-based models (ABM) have been used to model spatial interactions. Spatial models require special considerations, such as proximity to other agents. This creates areas of interaction and interest where agents are located, and areas of no interactions where there are no interactions. Spatial models lead to more computationally complex models, but result in more interesting results.

Whereas in non-spatial models such as ODEs and DDEs, the bacteria and phages are assumed to be in a well-mixed solution and no distinctions are made in regard to neighbors or distances to other agents. Interactions are simplified to a probabilistic approach, where a percentage p of agents interact with one another at time step t. Non non-spatial models are easier to develop, understand, and are more effective in modeling large populations, at the cost of losing spatial information.

For this thesis, the focus will be modelling resource, phage, and bacteria interactions using an ODE model. A phage-bacteria-resource system is described as an $p \times b \times r$ system, meaning p phages, b bacteria, Current modelling methods have mainly stayed with $1 \times 1 \times 1$ models, meaning 1 phage, 1 bacteria, and 1 resource. This thesis aims to develop a simulation framework that can model any $p \times b \times r$ system, where each agent can contain states (called hidden agents) that they can move to and from.

1.6 Thesis Project

The project is divided into three logical parts, with an optional fourth part. The first section is to create the network interaction. Here the user of the software can define the number of resources, phages, and bacteria, who interacts with who, and the strength and type of interactions. See Network Creation Tool for further information.
In Simulation Framework, the user uploads the network model and parameters and as output receives the time data and population data as an array.

Visualization Dashboard allows the user to interact with Network Creation Tool and Simulation Framework with a dashboard. The user can graphically edit the attribute values of the edges and nodes of the network, and the user can run more advanced visualizations, for example by changing a parameter value and seeing how that affects the population count. There are a few plots included out of the box that the user can test. The plots offered in part 3 offer interactivity like hiding and showing lines and dots, zooming in and out, and hovering over the lines and dots to show more details of the data.

Finally, the user can optionally run multiple simulations and download the data to their disk to create their own custom visualizations using Custom Visualizations and Analyses. The visualizations created in Visualization Dashboard can theoretically be recreated in Custom Visualizations and Analyses. The user can choose the same parameter values used for a specific plot in Visualization Dashboard, run the simulation (under the Ultimate Analysis section), download the data, and reimplement the graphs.

Chapter 2

Literature review

2.1 Methods of Modelling Phages and Bacteria

There are numerous ways to model the interactions between phages and bacteria. Models can be built at a molecular level, where the model simulates the mechanical and chemical behavior of a phage as it interacts with the surface of a bacterium using computational chemistry methods. On the other end of the spectrum, a different type of model can be built where populations of phages, bacteria, and resources can be modeled using Ordinary Differential Equations (ODEs) or Delay Differential Equations (DDEs). DDEs are similar to ODEs, except where when ODEs are calculating the values of the equations at time t using time $t-1$, DDEs can, but don't have to, use the value of the equation at time $t-\tau$, where $1 \leq \tau \leq t$. DDEs are a generalized version of ODEs and are significantly harder to analyze and find stability conditions than ODEs due to the dependence on the past [16].
One way to introduce DDE like behavior is to force agents to go through stages, causing a delay in other events. For example, in the paper Geng et al. [17], infected bacteria go through M stages of infection, before lysing. The more stages there are, the longer the delay in seeing a rise in phage population. By changing the value of τ in the model proposed by Geng et al. [17], the throughput of bacteria going from stage i to stage $i+1$ of infection increases, thus seeing a larger rise in phage population.

Each type of model has its pros and cons. With the molecular level model, the model is more complex and needs significantly more startup time, simulation time, and is in general much more complex. However, more information can be gained from the simulations and can guide research in creating phages for a certain type of bacteria. The ODE method is simpler and easier to set up, however it can only capture large population dynamics. Certain assumptions about the community interactions have to

be made. For example, ω percent of the bacteria population is washed out. The model can be made more complicated, by modelling each stage of the phage replication and lysis process, or instead of assuming exponential growth, there is a maximum carrying capacity of the population. The model can be further altered by using a normally distributed variable $\mathbf{N}(\mu = \omega, \sigma = 1)$ to account for noise when measuring the data. Ensuring the use of a seed value will ensure that each run of the model results in the same output.

2.1.1 Generalized Lotka-Volterra Model

The Lotka-Volterra model, a first-order non-linear differential model, captures the dynamics between predators and prey. Any population can be modelled as such:

$$\frac{dB_i}{dt} = B_i \left(\left(r_i + \sum_j^N \alpha_{ij} B_j \right) - m_i \right)$$

where r_i is reproduction rate, α_{ij} is the devour rate of B_i on B_j. If α_{ij} is negative, then B_i has a negative effect on B_j, otherwise B_i has a positive effect on B_j. m_i is the removal rate of B_i. The interactions can be seen in Figure 2.1a

2.1.2 Generalized Consumer-Resource Model

The generalized Consumer-Resource Model models the growth of a population and resource dynamics between a population of bacteria B_i and a resource R_i.

$$\frac{dB_i}{dt} = r_i B_i \left(\sum_\alpha \Delta w_{i\alpha} C_{i\alpha} R_\alpha \right) - m_i B_i \tag{2.1}$$

$$\frac{dR_\beta}{dt} = -\sum_i C_{i\beta} R_\beta B_i + \sum_{\alpha,i} D^i_{\beta\alpha} C_{i\alpha} R_\beta B_i \tag{2.2}$$

$$\Delta w_{i\alpha} = \sum_\beta D^i_{\beta\alpha} w_\beta$$

Equation (2.1) describes the growth of population B_i and Equation (2.2) describes the resource dynamics and metabolism of resource R_β. Resource R_α can become resource R_β at rate $R^i_{\beta\alpha}$. Bacteria B_i reproduces at rate r_i dependent on the concentration of resources $\sum_\alpha C_{i\alpha}$. Bacteria die out at rate m_i. For a visual, see Figure 2.1b

2.1.3 Trait-Based Model

The Trait-Based Model is a model that takes into account external factors such as the temperature or pH of the system and can be modeled as follows.

$$\frac{dB_i}{dt} = (r_i - m_i)\, B_i$$
$$r_i = \frac{r_{i\alpha}^{max} R_\alpha}{R_\alpha + K_{i\alpha}} e^{S_i(T - T_{ref_i})}$$

where r_i is influenced by the environment impact factor. S_i is the sensitivity to B_i to factor T, and with trade off if $r_i^{max} >$ mean r^{max} then $S_i >$ mean S. The larger $S1_i$ is, and the larger the difference T is from T_{ref_i}, the stronger the effect will have on the growth rate of r_i. r_i follows the Monod equation, with $r_{i\alpha}^{max}$ being the maximum growth rate of bacteria B_i, R_α is the resource concentration, and $K_{i\alpha}$ is the affinity constant. Figure 2.1c shows the agent interactions in detail.

2.1.4 Agent-Based Models

ABMs model the system through space and time. An $x \times y$ grid is created and split into smaller sub-cells containing resources and microbes. Each cell acts as its own tiny environment, where resources and microbes interact within the cell, but not with the neighboring cells. Resources diffuse through the system using a PDE solver for a Boundary Value Problem (BVP). Agents can move into neighboring grids with a probability p, where p can depend on any number of parameters such as resource density, microbe density, or stochastic chance.

ABMs are useful when simulating many individual elements interacting in a system. Chaotic or emergent behavior can arise from these interactions. Chaotic behavior refers to the irregular and unpredictable evolution of a system's behavior due to nonlinear equations, exhibiting sensitive dependence on the initial condition (IC) [18].

Emergent behavior is behavior that arises from the interactions of various agents in a system, that was not explicitly programmed into the system. The behavior can be beneficial, neutral, or harmful, but it can not be predicted until it arises, *if* it arises. Agents can have simple rules, but when interacting with other agents, behavior that hasn't been programmed can arise. Sometimes, people consider systems with emergent behaviors more complex than the sum of their parts.

$$\frac{\delta R_\alpha((x,y),t)}{\delta t} = \nabla \left[D\left(R_\alpha,(x,y)\right) \nabla R_\alpha\left((x,y),t\right) \right] \tag{2.3}$$

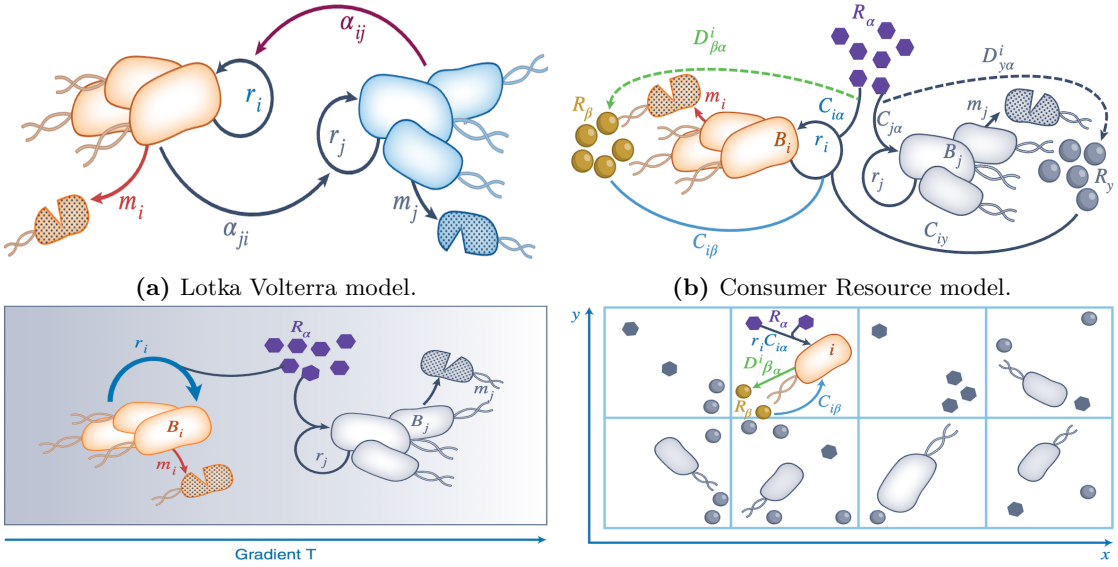
Equation (2.3) describes the diffusion of resource R_α through the matrix cells, dependent on the resource concentration R_α at cell location (x, y). The rules for cellular agents follow Equation (2.4).

$$\frac{di}{dt} = r_i \left(\sum_{\alpha} \Delta w_{i\alpha} C_{i\alpha} R_\alpha \right) \tag{2.4}$$

, where i is a bacterial agent, where if $0 \leq$ threshold $\leq \frac{di}{dt} \leq 1$, some threshold, $\frac{\frac{di}{dt}}{2}$ expands into the neighboring grid cell with probability p. Agent i consumes resources and converts them into new resource with Equation (2.5).

$$\frac{dR_\beta}{dt} = \sum_{i} C_{i\beta} R_\beta I + \sum_{\alpha,i} D^i_{\beta\alpha} C_{i\alpha} R_\alpha i \tag{2.5}$$

Figure 2.1d shows how the agents interact with other agents in their cell.



(a) Lotka Volterra model.

(b) Consumer Resource model.

(c) Trait Based model.

(d) Agent Based model.

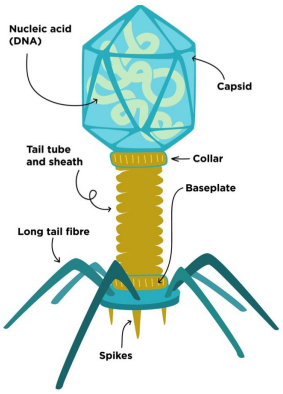
Figure 2.1: Different models and how the bacterial agents interact with itself, one another, resources and the environment. All figures sourced from van den Berg et al. [2]

2.2 Phage Biology

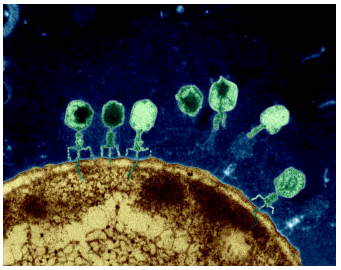
2.2.1 What Are Phages?

Phages are small bundles of proteins that contain viral DNA. Phages are made up of multiple parts built like LEGO to complete the task of infecting a bacterium. Figure 2.2a shows the body parts of a phage. The aim of the phage is to find a suitable bacterial host

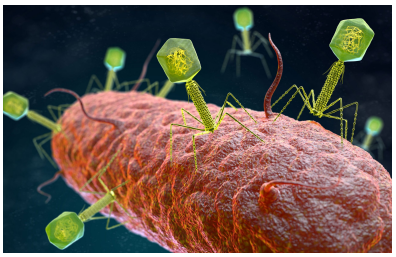
and infect the host with viral DNA. The DNA alters the host's metabolic pathways to its benefit and hijacks the cellular replication process to create new copies of the phage. Eventually, the cell lyses, releasing the newly created phages into the environment to infect more bacteria.



(a) Phage body structure.



(b) Phages infecting an *E. coli* bacteria.



(c) Artist representation of phages infecting a bacterium.

Figure 2.2: Parts of a phage, a real life picture of phages infecting an *E. coli* bacterium, and an artist's impression of phages infecting a bacterium.

2.2.2 How Does the Phage Cycle Work?

There are 3 main parts to the phage-bacteria host cycle, the infection stage, the lysogenic cycle, and the lytic cycle. In the infection stage, a phage floating through the environment detects and attaches to the surface of a bacteria cell. Once injected, the phage-cell pair can directly go into the lysogenic cycle or into the lytic cycle. Figure 1.1 shows a detailed overview of the phage cycle.

In the lysogenic cycle, the phage DNA injects integrates into the genome of the bacteria. As the bacteria undergoes cellular replication, the DNA of the phage will be copied with the cell. After a set amount of time, the phage DNA can cut itself from the genome and enters the lytic cycle.

In the lytic cycle, the phage hijacks the cellular process of the bacteria. The phage DNA hijacks the replication, transcription, and replication process of the cell, making more and more copies of phage. The phage parts build together to make a full part. Eventually the cell wall bursts releasing the phages into the environment ready to infect more bacteria.

2.2.2.1 Infection Stage

The infection stage is characterized as the searching for a bacterium, detection, and subsequent attachment and injection of DNA into the bacteria.

Detection and Attachment Phages float through the medium and by chance land on a bacteria. The phage detects the cell via phage receptor binding proteins located at the tip of the phage tail. Various inter-molecular forces such as hydrogen bonds help the phage detect and attach to the cell. The receptors are tuned to specific receptors found on the surface of the bacteria cell wall. Upon detection, conformational alterations in the phage's baseplate occur, causing changes in protein shapes, causing the sheath to contract and inject the viral DNA into the host. The successful binding and adsorption depends on the phage binding protein sensitivity, localization,and density of receptors. [19].

Phage DNA Injection The injection is triggered by the recognition between the phage's receptor-binding protein located at the tip of the tail and a specific receptor located on the surface of the bacteria. Once a suitable injection site has been identified, the phage injects the DNA into the cytoplasm of the cell. The specificity of recognition is directly related to the specificity of adsorption, which correlates to the structure of receptors located on the host's cell surface [19]. The injected DNA is called a plasmid, genetic structure usually in the shape of a circle that can replicate independently of chromosomes.

2.2.2.2 Lysogenic Cycle

The lysogenic cycle describes the process in which the viral DNA of the phage evades detection, integrates into the cell's DNA, replicates with the cell, and inducts from the DNA. Phages that have integrated into the host's DNA are called prophages.

Repression of DNA As phages are viruses, they need to evade viral detection methods such as Cyclic oligonucleotide-based anti-phage signalling systems (CBASS). CBASS triggers effector proteins that cause cell death, preventing phage replication and lysis [20]. Two big benefits of programmed cell death is that the cell death slows the growth of phages and the dead cells release resources into the environment, allowing other bacteria to recycle the resources and grow [21].
CRISPR-Cas is another method that bacteria can use to detect the presence of phage

DNA. CRISPR-Cas is an adaptive immune system in bacteria that defends against phages by acquiring foreign DNA sequences (spacers) into its CRISPR array, transcribing them into CRISPR RNAs (crRNAs), and using these crRNAs with Cas proteins to identify and degrade foreign DNA [22].

Phage DNA Integration Into Bacteria DNA The DNA of the phage is able to integrate into the bacteria's DNA. Prophages can alter the fitness of the cell, by changing metabolic routes and other cellular structures and functions to better survive under resource limitations or by increasing resistance against other phages. By altering the fitness of the cell, the prophage can wait until better conditions are met for a lytic approach to be favorable [21].

Cellular Replication The cell undergoes division multiple times, copying the prophage DNA into the cell copies. However prophages are still at risk of being discovered and excised by restriction enzymes [23].

Phage Induction Prophages induct (leave) from the bacteria DNA under specific conditions. The induction process starts with proteolytic cleavage and displacement of the phage repressor, which most of the time occurs upon activation of the SOS response following DNA damage [24]. Cell stressors such as DNA-damaging agents like UV light and antibiotics can jump-start the process to switch to the lytic cycle [19, 25].

2.2.2.3 Lytic Cycle

The lytic cycle describes the process in which the viral DNA hijacks the DNA replication process, assembles within the cell, and lyses the cell releasing the phages into the environment.

Hijacking DNA Replication Process The phage hijacks the cellular replication process to create the different proteins that make up the phage, like the legs, body, and head. Phenotypic reconfiguration of the host is frequently facilitated by auxiliary metabolic genes, which are genes initially sourced from host genomes but preserved and modified within viral genomes to channel energy and resources toward viral replication [21].

Assembly of Phage Parts Phage parts self-assemble by using various protein-protein and protein-nucleic interactions, along with other forms of interactions such as hydrogen bonding and hydrophobic/philic interactions [26].

Lysis of the Bacterial Cell Internal pressure buildup causes the cell wall to explode, releasing phages, resources, and other organic matter into the environment. Genetic material from one bacteria can be transferred to other bacterial cells via phages, driving bacterial evolution.

2.3 Bacterial Defense Against Phages

There is a constant battle between phages and bacteria. The bacteria don't want to be killed by the phages, so they adapt defenses such as thickening of the cell wall, or once the viral DNA has integrated with the bacteria's DNA, the bacteria will cut the viral DNA out of their DNA using CRISPR and restriction enzymes [27].

2.3.1 Mutations in Bacterial DNA (Genetic (Co-)Evolution)

As bacteria cells grow and divide, random point mutations can occur in the DNA. These mutations can affect phage defenses, like thickening the cell wall or removing a receptor, making it harder for the phages to detect and infect the cell. Mutations might not always work, or they can have the opposite effect. They can be partially effective if full effectiveness requires multiple steps to achieve, which can occasionally fail [28] or the mutation brings a cost to the bacteria cell by losing receptors on the cell wall.

Bacteria can horizontally transfer DNA to other bacteria on contact. There are three primary ways of this happening, which is visualized in Figure 2.3.

The first method is via conjugation, where a donor cell donates DNA fragments using a mechanism called the F-factor or plasmid with a pilus. The pilus acts as a tunnel between the donor cell and the recipient cell so that DNA can be transferred from the donor cell to the receiver cell. This method of sharing DNA can also have the unintended side effect where one bacteria will directly infect another bacteria by transferring phage DNA.

The second method, called transformation, occurs when a cell takes up released DNA fragments from the environment. Once inside the receiver cell, the donor DNA can integrate itself with the receiver DNA. There is also the chance that the bacteria picks up viral DNA [29].

The third method is via transduction. When a phage is assembling in the cell just before lysis, the phage can accidentally collect a piece of the host's DNA instead of its own DNA. The dying bacterium proceeds to lyse, releasing the phages. The phage with the now dead hosts DNA can infect the next bacteria, injecting the DNA strand of the now dead cell into the new host cell. The old bacterial DNA will proceed to integrate with the new host cells DNA [3, 30].

All methods provide a way for the cell to mutate and change the fitness value of the cell.

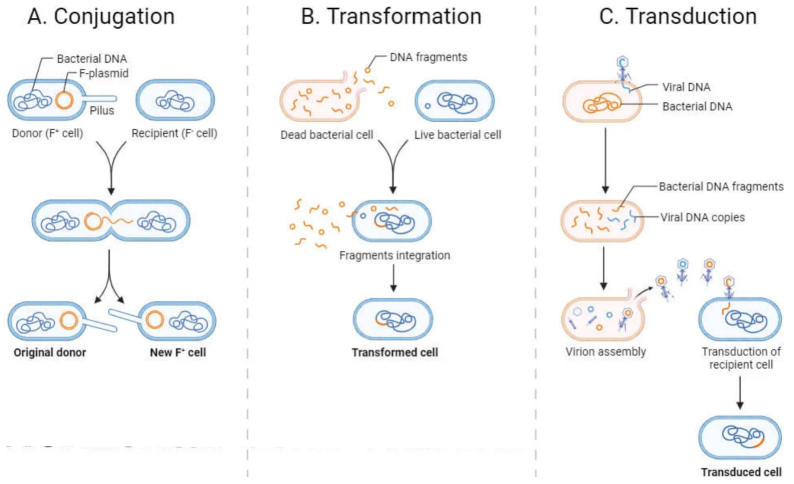


Figure 2.3: The three main ways that a (dead) bacterium can horizontally transfer DNA and genes over to another bacterium [3].

2.3.2 Phage Inactivation and Decoys

Bacteria can further protect themselves by producing decoys that the phage will attach to instead of themselves, inactivating the phage. Freshly lysed bacteria can still contain biomarkers that phages use to detect the bacteria, but upon injection, nothing happens as the cell doesn't function anymore. Bacteria can also produce proteolytic enzymes that will damage the proteins found in a phage [31].
Some bacteria can produce outer membrane vesicles that phages can absorb to, and later detach the vesicle with the phage [32]. The vesicle will proceed to float away with the attached phage, posing no risk to itself or to other bacteria. It is suspected that the impact of these vesicles acting as a sink is minor [33], but helpful nonetheless.

2.3.3 CRISPR-Cas Methods

CRISPR is a gene editing tool that cells can use to cut out specified/unwanted parts of a DNA strand. Researchers are commonly using CRISPR to genetically engineer plants and animals to have specific features. Strands of DNA can be selectively added or

removed from a DNA strand to achieve a better, more desired DNA strand. Specialized defenses in the bacteria can detect unwanted strands and remove the strand, acting as a line of defense against phages.

2.3.4 Phenotype Resistance

Although mutations can occur in bacterial DNA, not every mutation results in a distinct phenotypic change. However, there is still a chance that a mutation can change the phenotype representation. A mutation can cause the cell wall to thicken, making it harder for a phage to infect the cell. The bacterium can decrease the number of receptors that a phage can detect, making it harder for the phage to detect the bacterium.

Gupta et al. [34] found that some *Bacteroides fragilis* bacteria were able to evade phage infection. The presence of combinatorial phenotypic states where differential expression of protective mechanisms created rare super-resistant cells capable of withstanding phage attack. By acting together, these heterogeneously expressed anti-phage defense mechanisms created a phenotypic landscape where distinct protective combinations enabled the survival and re-growth of bacteria expressing these phenotypes without acquiring additional mutations.

2.3.5 Spatial Refuge/Biofilms

Usually bacteria and phages coexist in well mixed environments such as the ocean, however some environments offer natural structures for bacteria to hide behind. These structures can range from physical structure, like sediment in water to biochemical structures like biofilms, where the phages can't diffuse through the biofilm. In large enough quantities, bacteria and other microbial communities create biofilms, a layer of mucus containing various microbes. The thick mucus, microbes, and other spatial effects help protect the bacteria in the biofilm from external phages by making it hard for the phages to penetrate and diffuse through the mucus [35]. In the case of a lab experiment on an agar plate, bacteria protect one another by making it harder for the phages to diffuse through the system [36].

Phages can not swim and do not contain any parts that allow it to move under its own power. Movement is instead passive, relying on the environment to move through the environment, such as diffusion, changes in pressure or heat gradients [37]. The motion that phages exhibit is called Brownian motion, the seemingly random movement of small particles throughout a medium due to other microscopic particles interacting and bouncing off of one another [38]. Unlike phages, bacteria have the ability to actively

move through the environment, and they can use this to their advantage by crawling or swimming away if they detect a phage.

2.3.6 Other Methods of Defense

Other methods of defense include phage restriction by prokaryotic argonaute proteins, production of small molecules that block phage propagation, depletion of molecules essential for phage replication, systems that use small molecule signaling to activate immune effectors, retrons that involve reverse-transcription of non-coding RNAs, and more [39].

2.4 Phage Counter Defense Against Bacteria

With some of the defenses that bacteria have developed, phages are always mutating to counter their defenses. If phages don't adapt to the ever-changing bacterial defenses, the phages will die out due to their inability to infect and multiply. It essentially becomes a race to the bottom, seeing who can out-adapt the other. However, if the phages out-adapt the bacteria too much, the bacteria die out, then eventually the phages die out due to not having any bacteria left to infect.
This can be avoided if the phages can adapt to target a second strain of bacteria, but this is unlikely. On the other hand, if bacteria out-adapt the phages, that is no problem for the bacteria because they don't need the phages to survive, and can keep on growing, limited only by the available space and resources.

This is a problem intrinsic to predator-prey systems, namely that the predators are dependent on the prey. Once the prey disappear, the predators also disappear. If the prey population goes down, and as a result the predator population goes down and becomes extinct, the prey can come back without the threat of predators.

Phages face this exact same problem: the complete removal of either the bacteria or phages will lead to the removal of the phages from the system unless reintroduced.

2.4.1 Genetic Mutations

Mutations in viral DNA will affect how the phage body parts are designed and built. These mutations will affect external phage behavior such as how it detects a bacteria, as well as internal behavior such as evading detection and integrating with the cell's DNA.

The changes will lead to changes in overall phage fitness, ie the ability for the phage to infect, replicate, and lyse bacteria.

2.4.2 Viral Recombination

https://www.sciencedirect.com/science/article/pii/S1931312821004170 https://pmc.ncbi.nlm.nih.gov/a
Multiple phages can infect a cell and replicate itself using the cells internal replication process. Each phage has its own building blocks. Phage 1 could have long legs, a long neck, and a small head, while phage 2 can have long legs, a long neck, and a medium-sized head. When the phages are building copies of themselves, they could accidentally use the body parts of other phages. The primary method for proteins to bond with other proteins and molecules is via hydrogen bonds. These attractive forces hold proteins and other molecules in defined positions, and a change in molecule shape will change the bonds, which will force the other molecule to undergo changes in shape. If the proteins that build the subparts of each phage have similar chemical properties, they can be swapped between phages. This allows for biological diversity to spread throughout a phage population. Each phage body part can have unique characteristics such as better attachment rate, larger DNA storage capsule, or better probability of injection.

Coexistence between phages and bacteria via genetic co-evolution seems unlikely due to trade-offs imposed by the new mutations [40].

2.5 Phage Defense Against Phages

Some phages can employ defenses against other phages from infecting the bacterial cell ensuring the hots resources are all for itself.

2.5.1 Superinfection Exclusion

The act of preventing a secondary infection form a similar or closely related phage is called superinfection exclusion (SIE). [41]. There are various methods of preventing further infections.

2.5.2 Altering Cell Structure

The prophage can alter the surface receptors of the bacteria, making it harder for other phages to detect the bacteria, reducing the chance of attachment and injection by other

phages [42]. The prophage can hijack the internal metabolic pathway and cellular functions, affecting the genes that are expressed and transcribed to proteins.

2.5.3 Protein Creation

Other phages like the T4 phage can use proteins like the Spackle protein. The Spackle protein inhibits the lysozyme activity used in the process of DNA injection by other phages [42, 43]. Some prophages can encode proteins that will interfere with the replication process of other phages. For example, the SieA protein encoded by phage P22 blocks infection from other phages [44].
TAB (Tail Assembly Blocker) is an anti-phage defense mechanism encoded by a *Pseudomonas aeruginosa* prophage. While TAB permits the invading phage to replicate its genome, it inhibits the assembly of the phage tail, thereby preventing the production of infectious virions. The prophage that encodes TAB is not affected by this inhibition, as it also expresses a protein that neutralizes TAB's blocking activity. Although the host cell still undergoes lysis, no infectious phages are released.

2.5.4 Implications of Phage Against Phage Defense

SIE can affect the speed and development of phage and bacterial populations. A phage restricting other phages from infecting the bacteria creates a competitive environment and can out-compete and dominate the other population. This is commonly seen in wildlife populations, where invading species can out compete other species by consuming resources faster than local species, breeding at a faster rate than other species, and having no natural predators.

2.6 Bacteria and Phages in the Lab

Researchers around the world are running lab experiments to gain further knowledge of the interactions between phages and bacteria. The aim is to better understand how phages work and interact with bacteria at a molecular, host, and population level.

A researcher might run the experiment in a liquid medium containing water, carbon and nitrogen sources, and other chemicals such as anti-foaming or pH control chemicals. This liquid medium, often referred to as broth, allows for the cultivation of bacteria in a well-mixed environment, enabling researchers to monitor bacterial growth and phage infection dynamics over time. By adjusting parameters such as nutrient concentration, temperature, agitation speed, and pH, researchers can simulate different environmental

conditions and observe their effects on phage-bacteria interactions. Samples can be taken at various time points to measure bacterial density, phage titer, and resource depletion, providing quantitative data for model validation and hypothesis testing. If measured frequently enough, the researcher can get an ODE-like curve out, where each line shows the population levels at that time. Researchers can create a mathematical interpretation of the curves and run algorithms such as curve fitting and simulated annealing to find and tune the ODE model parameters. The tuned ODE parameter values tell the researcher the reaction rate speeds, the burst size of the cell, and cell latent period [17, 45]. Chemostats and batch cultures are commonly used setups, with chemostats allowing for continuous input of fresh medium and removal of waste, maintaining steady-state conditions ideal for studying long-term dynamics.

Petri dishes are another commonly used way to grow bacterial colonies. Agar, a jelly-like substance derived from seaweed, is commonly used as a solid growth medium in petri dishes. It provides a stable surface for bacteria to grow and form visible colonies. By adding nutrients and other supplements to the agar, researchers can tailor the medium to support the growth of specific bacterial strains or to test the effects of different environmental conditions. When phages are introduced to a bacterial lawn on agar, clear zones called plaques appear where phages have infected and lysed the bacteria, allowing for quantification and observation of phage activity. As a cell lyses, it releases phages into the surrounding. The phages can diffuse through the system, infecting neighboring cells. A small plaque of size 2-3mm can be created, where there are no bacteria.

Bacteria density can be measured optically using light. In the case of liquid cultures, as the bacteria grow and die, the solution will get more cloudy. By shining a light through a control vial with no bacteria growth and through a vial with bacteria growth, the change in light refraction and intensity can be measured. A researcher might also be interested in using a mass spectrometer to measure the density of phages and nutrients at specific time points.

With petri dishes, it is harder to measure the bacterial growth. Bacteria are usually mixed with phages in a heated liquid agar solution, and poured onto a petri dish. It might be possible to scrape the bacteria off of the dish into a liquid to measure the optical density (OD), but the results are not always consistent. A computer vision algorithm might be able to quantify the change in color on the petri dish, by comparing the photo of the bacterial lawn with a reference photo with no bacteria growth. Or it can compare the area of growth with area of no growth, where phages are present. However the results are sensitive to camera settings, such as exposure and sharpness. Lighting can have a big factor in the analysis such as if there are shadows from an object over the

plate, or if there is residual sunlight entering the room, making the room brighter or darker. It would be easier to measure the change in plaque size, assuming the camera and petri dish stay in the same position for every picture. Figure 2.4 shows a sample bacterial lawn with phage plaques. If one were to zoom in, Figure 2.2b shows stained phages infecting a bacterium.

Measuring OD is inaccurate and can only accurately measure up to an OD of 0.1. Even though using a special spectrophotometer allows consistent results, the results are dependent on the medium, the length of travel through the medium, bacteria size and density. THe device and measurements need to be calibrated to ensure proper results. Changing methods to using *cells/ml* over OD can be used to directly compare results across experiments, labs, and bacteria colonies [46].

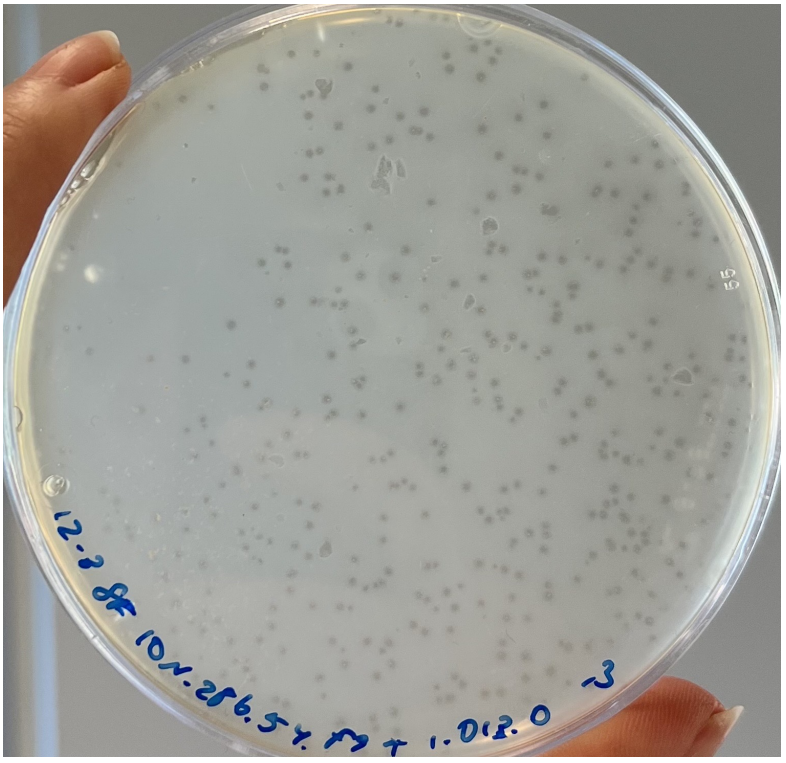


Figure 2.4: Bacteria lawn, the dots on the petri dish show no bacteria growth due to the presence of phages. Photo courtesy of S. Flickinger.

2.7 Software Mathematically Modelling Phages, Bacteria, and Resources

Some software programs modelling phage-bacteria-resource interactions already exists.

2.7.1 Cocktail

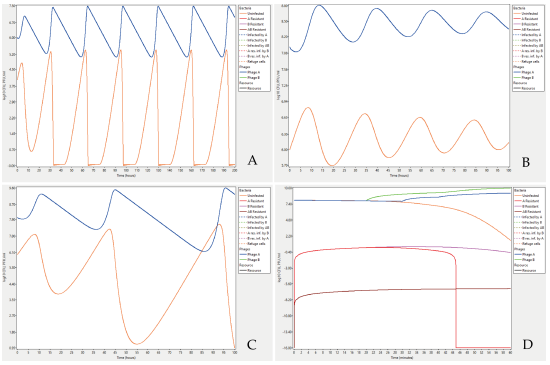
Cocktail was developed by Nilsson [4] to model phage-bacteria-resource kinetics in a chemostat. The model assumes there is one bacteria strain that can be infected by phage A and phage B, and by both phages at the same time, phage AB. The bacteria models resistance to phage A, B, and AB. The user controls the parameter values such as resistance rate to A, B, and AB, resource concentration and outflow, and phage adsorption rate. There is an also an option to periodically add more phages. Model settings, such as if the model is deterministic or stochastic, and the step size is also controllable [4]. After choosing the parameter values, an output is created, with four sample plots shown in Figure 2.5a

2.7.2 PhageDyn

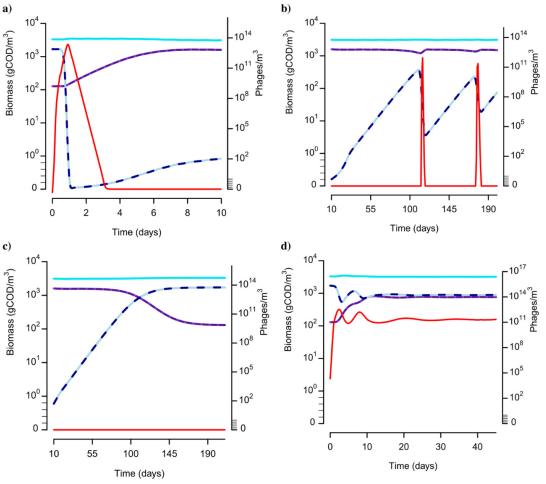
PhageDyn is a Java applet that interacts with existing files in GPS-X [47] to incorporate phage dynamics into models of industrial wastewater treatment plants [5]. The aim of PhageDyn is to model phage dynamics in multi-reactor models. Existing models are not applicable to a complex multi-reactor wastewater treatment plant model, which is why Krysiak-Baltyn et al. [5] decided to create PhageDyn to determine how phages can be used to reduce foaming caused by bacteria in wastewater treatment plants [48]. It should be noted that PhageDyn does not simulate phage dynamics on its own but rather manipulates existing files in GPS-X in order to incorporate phage dynamics into models of wastewater treatment plants. And in order for PhageDyn to work, an activated copy of GPS-X is required [5]. Figure 2.5b shows the output that PhageDyn provides.

2.7.3 Cocktail and PhageDyn Limitations

However there are limitations to Cocktail and PhageDyn. Cocktail can model up to a $2 \times 1 \times 1$ system, and is designed to model a chemostat. Constant resources are being added into a chemostat, with the constant removal of medium from the chemostat. Phages A and B can infect the bacteria, and the bacteria can gain resistance to phage A, B, and AB. The limitation of Cocktail is that the model can not easily be adapted. The ODE model accepts inputs from a hardcoded GUI frontend. So any changes to the frontend or to the ODE model will require changes to the ODE model and the frontend to accept the new inputs and outputs. The code for Cocktail is open source, so adding new buttons and changing the model should not pose a significant challenge, but still an undertaking.



(a) Figure A) *E. coli* infected with phage T4 in a chemostat exhibiting an oscillating growth behavior, following the model of Bohannan and Lenski [49]. Figure B) Oscillations of bacteria and phages can exist at higher titers, dependent on low resource concentration, following the model of Lenski [50]. Figure C) As the concentration of resources change, this results in increasing oscillations, but not going extinct. Figure D) A system modelling the interactions with phage A and B.



(b) **Purple** is heterotrophic biomass, **Blue** is foaming biomass, **Red** is phages, **Light Blue** is total suspended solids. Figure A) Biomass concentration immediately post phage dosing. Figure B) Biomass concentration with low phage concentration and maintain low concentration post spike in population count. Figure C) Biomass concentration when phages are extinct. Figure D) Biomass concentration with a less virulent and low adsorption rate phage, co-existence with biomass reached. A change in phage concentration shows a decrease in heterotrophic and foaming biomass [5].

Figure 2.5: Example output from Cocktail and PhageDyn respectively. For PhageDyn, concentration of heterotrophic biomass in an aerobic plug flow across four situations. See Nilsson [4] and Krysiak-Baltyn et al. [5] for more information on parameter values and supplementary resources.

PhageDyn works with GPS-X, a very niche wastewater treatment modelling software. PhageDyn is programmed for a very specific task with no flexibility in changing the model or inputs. To the best of my ability, I could not find a copy of PhageDyn or an acknowledgement that the code is open or closed source. When clicking on the link in the supplementary material of Krysiak-Baltyn et al. [5] to download a copy of the Java applet, the link returned a "URL Not Found" error.

Chapter 3

Methods

3.1 Project Overview

To help complete this Master thesis, I created various tools that would help create the final model outputs. The project is divided into four parts. The first part is the tool that a user can use to design and create the network of agent interactions. The second part is the simulation framework that handles the data and runs the ODE solving method. The third part is a dashboard that runs in the browser. The dashboard allows for a user to interact with the model, for example by changing parameter and environment values, and run some basic simulations and receive different plots as output. The final part allows the user to download the simulation data to create their own custom graphs and analyses.

A flowchart showing the user-system interactions can be seen in Appendix C: Flowchart of User and System Interactions.

3.2 The Golden Model

The default model, the "Golden model", sourced from Geng et al. [17], describes the interactions between Resources, Uninfected bacteria, Infected bacteria, and Phages. All plots and simulations will use this model by default, unless stated otherwise.

3.2.1 The Golden Model

where R is resources, U is uninfected bacteria, $I_{1,...,M}$ is the infected stage of the bacteria, and P is the phage population.

Equation 3.6 The golden model sourced from Geng et al. [17]. The text in red has been added to the model, adding (the wash-in) fresh resources (ω^i) and the removal (wash-out) of agents (ω^o). The washin is not dependent on the current resource population, as it is a constant rate being added. By default these values are 0. A summary of the parameters can be found at Table 8.1.

$$\frac{dR}{dt} = -e \cdot g(R, v, K) \cdot (U + \sum_{i=1}^{M} I_M) + w^i - w^o \cdot R \tag{3.1}$$

$$\frac{dU}{dt} = g(R, v, K) \cdot U - r \cdot U \cdot P - w^o \cdot U \tag{3.2}$$

$$\frac{dI_1}{dt} = r \cdot U \cdot P - \frac{M}{\tau} \cdot I_1 - w^o \cdot I_1 \tag{3.3}$$

$$\frac{dI_k}{dt} = \frac{M}{\tau}(I_{k-1} - I_k) - w^o \cdot I_k \text{ for } k = 2, \ldots, M \tag{3.4}$$

$$\frac{dP}{dt} = \beta \cdot \frac{M}{\tau} \cdot I_M - r \cdot (U + \sum_{i=1}^{M} I_M) \cdot P - w^o \cdot P \tag{3.5}$$

$$g(R, v, K) = \frac{v \cdot R}{R + K} \tag{3.6}$$

The model describes three biological processes, cell consumption of resources and growing, phage/cell encounters and infection, and cell lysis. The cell growth process is described by $g(R, v, K)$, the instantaneous growth rate dependent on the Monod equation, where v is the maximal growth rate and K is the Monod constant. The consumption rate of a resource by a bacteria is e.

Once infected by a phage, the bacteria goes from U to I_1. The bacteria goes through M stages of infection $I_1, \ldots, I_M$ before lysing, where the bacteria goes from state I_k to state I_{k+1} with equal transition rate $\frac{M}{\tau}$. The probability of a successful infection of a cell is r. After a bacteria lyses after stage I_M, β phages are released, the burst size of the phage.

However this model is specifically designed for a $1 \times 1 \times 1$ model. In order to adapt this model to fit an $p \times b \times r$ model, the model needs to be slightly adapted. There are other changes that can be made to the model, for example by adding a washin rate ω^i, where resources are constantly being introduced, and a washout rate ω^o where all agents are washed out at a constant rate. These changes are highlighted in Equation (3.6) in red.

3.2.2 The Adapted Golden Model

Equation (3.12) accounts for the interactions of multiple agents. Each agent is a sum of all interactions. For example, B_0 is independently infected by P_1 and P_2, thus the uninfected B_0 lose $r_{1,0} \cdot P_1 \cdot B_0 + r_{2,0} \cdot P_2 \cdot B_0$ uninfected bacteria. Likewise, if bacteria

B_1 is being infected by phage P_1 and P_2, the uninfected B_1 population is reduced by the sum of infections from P_1 and P_2.

Equation 3.12 Probability of phage infection $r_{p,b}$ is not to be confused with R_r, short for Resource r. The interactions are a sum of all interactions due to all interactions taking place at the same time.

$$\frac{dR_r}{dt} = -\sum_{b \in B} e_{b,r} \cdot g(R_r, v_{b,r}, K_{b,r}) \cdot (U_b + \sum_{i=1}^{M} I_{i_b}) + w_r^i - w^o \cdot R_r \tag{3.7}$$

$$\frac{dU_b}{dt} = U_b \cdot \sum_{r \in R} g(R_r, v_{b,r}, K_{b,r}) - U_b \cdot \sum_{p \in P} r_{p,b} \cdot P_p - w^o \cdot U_b \tag{3.8}$$

$$\frac{dI_{b_1}}{dt} = U_b \cdot \sum_{p \in P} r_{p,b} \cdot P_p - \frac{M}{\tau_b} \cdot I_{b_1} - w^o \cdot I_{b_1} \tag{3.9}$$

$$\frac{dI_{b_k}}{dt} = \frac{M}{\tau_b}(I_{b_{k-1}} - I_{b_k}) - w^o \cdot I_{b_k} \text{ for } k = 2, \ldots, M \tag{3.10}$$

$$\frac{dP_p}{dt} = \sum_{b \in B} \beta_{p,b} \cdot \frac{M}{\tau_b} \cdot I_{b_M} - r_{p,b} \cdot (U_b + \sum_{i=1}^{M} I_{b_i}) \cdot P_p - w^o \cdot P_p \tag{3.11}$$

$$g(R_r, v_{b,r}, K_{b,r}) = \frac{v_{b,r} \cdot R_r}{R_r + K_{b,r}} \tag{3.12}$$

.

3.2.3 Network Creation Tool

The network creation tool is the first step that a user needs to use, and it revolves around using a GUI tool built to create and edit the graph representation of the agent interaction. Numerous interactions occur between agents in a microbial environment. However, not every agent can and will interact with one another. Based on which agents interact with one another, a network topography representing the agent interactions and dynamics can be created.

Every node represents a unique agent, and each agent has their own intrinsic properties. The user can intuitively define agents, their interactions, environmental and model settings using the GUI tool. This tool allows users to quickly and intuitively define agents and their attributes, agent interactions and their attributes, environmental data, and model settings. An edge links two agents together if there is an arbitrary interaction occurring between the agents, with the properties exhibited in the interaction dependent on the interacting agents. Self interactions are allowed in the network. There is an environment node that is used to store global environmental data, such as the temperature and pH of the system. The settings node holds information such as simulation length, max time step, and the type of ODE solver to use. The tool provides functionalities for

adding, editing, and visualizing nodes and edges, as well as importing and exporting the network structure.

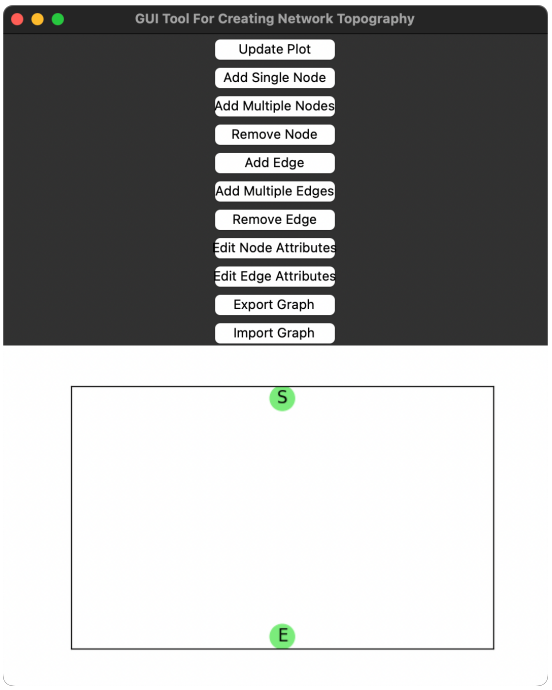
Once the user is happy with the graph shape, they can export the network representation for use in Simulation Framework, Visualization Dashboard, and Custom Visualizations and Analyses. The most important part is that the user defines the shape and the attributes of the network, as that can't be edited in part 2 onwards. It is possible go back to the network creation tool and upload the graph to the tool to be able to edit the network representation.
The user can edit the values of the attributes in Visualization Dashboard, so the parameter values do not have to be perfect. As such, the user does not need to keep on using the GUI tool to edit parameter values.

Figure 3.1a shows the layout of the GUI tool. Figure 3.1b shows an example network that can be created. There are numerous buttons that can be used to edit the graph, for example adding or removing nodes and edges. By default, an environment node holding parameters such as pH and temperature is added. A settings node is added as well, holding settings data to be used for the solver, like the type of solver or simulation length. Manually adding nodes and edges can get tedious and repetitive for large graphs, so the user can add multiple nodes and edges at the same time. Nothing can interact with the environment and setting node, as they are used to hold data about the environment and network solver. The user can alter the default attribute name and value by importing the GUI tool class and overriding the method implementation implementing the default names and values. The user can self-decide which parameter values to give, and if and how the parameter values are randomized.

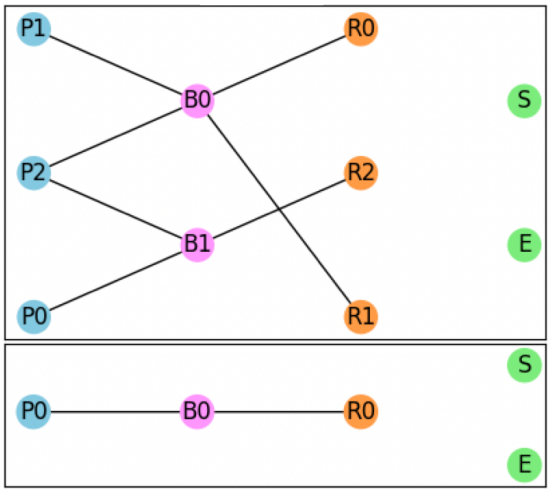
3.2.4 Simulation Framework

The user provides an ODE model and the network topography as input to the framework. The simulation framework deals with handling the input, output, collecting and storing of the simulation input and output. The framework uses SciPy's [51] *solve_ivp()* numerical solver [52] to simulate the provided ODE equations and calculate the population levels through time. The user receives two outputs from the framework. The first output is an array of time values that the solver used to calculate the population count. The second output is an array containing the population count at each time step for every agent.

In order to facilitate more complex model behavior, "hidden" agents can be added to the simulation, where the hidden agents represent states that a "visible" agent can be in. Such an example would be the distinction between uninfected and infected bacteria.



(a) The GUI tool as seen on the startup of the program.



(b) An arbitrary $3 \times 2 \times 3$ and $1 \times 1 \times 1$ network with each node representing a phage, bacteria, or resource, with arbitrary interactions occurring between them. Although not used here, edges between the same agent types and self loops are allowed. This network topography along with a $1 \times 1 \times 1$ network will be used in the Methods and Experiments and Results sections.

Each bacteria agent B_b would contain two hidden sub-agents, called sub-agent states, an uninfected state sub-agent U_b, and infected state sub-agent I_b. It is possible to further create sub-agents, by having a bacteria go through M stages of infection. So each infected agent would have sub-agent I_{b_k} where $1 \leq k \leq M$.

In the network model you would explicitly create a $3 \times 2 \times 3$ network, but when passing the network to the simulation framework, you would tell the framework to model the subagents. The user's ODE model has to correctly model each (hidden) agent and correctly handle the changes in states.

Even though the user might submit a $3 \times 2 \times 3$ model, if the user follows the uninfected and infected classification, where each bacterium goes through 4 stages, the ODE model and framework will explicitly be modelling 3 phages, 10 bacteria (2 uninfected + $2 \cdot 4 = 10$ infected bacteria), and 3 resources.

The user might also be interested in modelling a resource reservoir in a chemostat, where resources go from an external reservoir not accessible by the bacteria to the virtual chemostat ready to be consumed by the bacteria. 3 hidden resource agents would be added to the system, where the resources would model the transfer of resources from the reservoir to the simulation environment. The provided ODE model will have to correctly model and transfer the resources from $R_{r_{\text{reservoir}}}$ to $R_{r_{\text{chemostat}}}$, where $R_{r_{\text{reservoir}}}$ is unaccessible by the bacteria and $R_{r_{\text{chemostat}}}$ is accessible by the bacteria. It is then up

to the user to determine how the resources are transferred, if the resources are added at constant intervals or continuously.

3.2.5 Visualization Dashboard

The third part involves analyzing and visualizing the simulation results on an interactive Dash Plotly [53] dashboard. The user can use a dashboard built using Plotly Dash to interact with the solver and network. The user can interact with the solver and network by changing parameter, environment, and setting values on the fly. This allows the user to quickly change parameter values and test different situations. The dashboard includes various starter plots that allow the user to test the model. As output, the dashboard will show interactive plots so that the user can analyze the system.

The dashboard allows the user to interact with the network, the model, and some prebuilt visualizations, and is built into three logical sections. The first section allows for the user to edit the network parameters and setting values on the fly to quickly iterate through different conditions and to fine-tune parameter selection without having to rebuild the network using the GUI tool. The second section allows for the user to see how the population count evolves over time for a given IC and parameter values, allowing to quickly test the network input. The final section allows for the user to run more advanced analyses on the network, for example, by changing multiple parameter values and visualizing the output.

3.2.5.1 Editing Network and Parameter Values

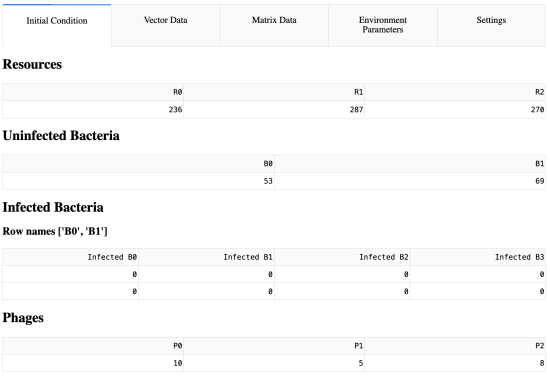
The editing network and parameter value contain five separate sections.

Initial Condition The IC settings panel (Figure 3.2a) allows for the user to edit the initial starting values of the agents. Each agent type has a table containing the initial population count. Extra hidden agents can be included. When a bacteria has been infected, the bacteria goes through multiple stages before lysing. Each bacteria agent starts out as uninfected, and once infected, the bacteria goes through 4 stages of infection before lysing as seen in Figure 3.2a.

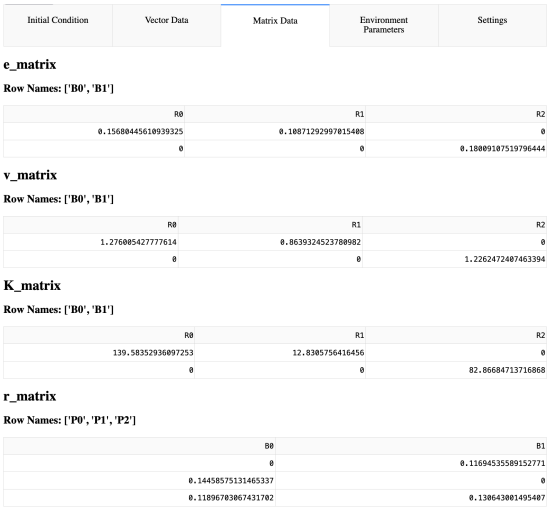
Vector Data Data that can be represented as a vector, for example the data attributed to an agent type have their own section, Figure 3.2c.

Matrix Data Data that is stored as a matrix, the data stored on edges between agents, is stored in the matrix tab (Figure 3.2b).

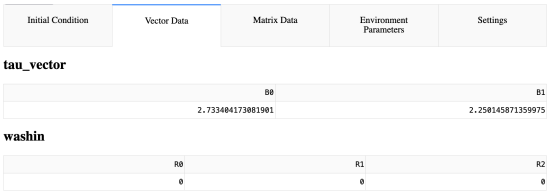
Environment and settings The environment data and settings data also have their own tab, Figure 3.2d and Figure 3.2e respectively.



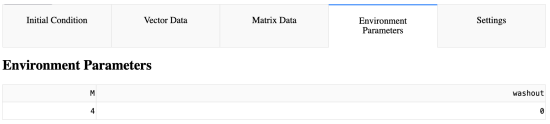
(a) The tab where the user can edit the ICs of the agents.



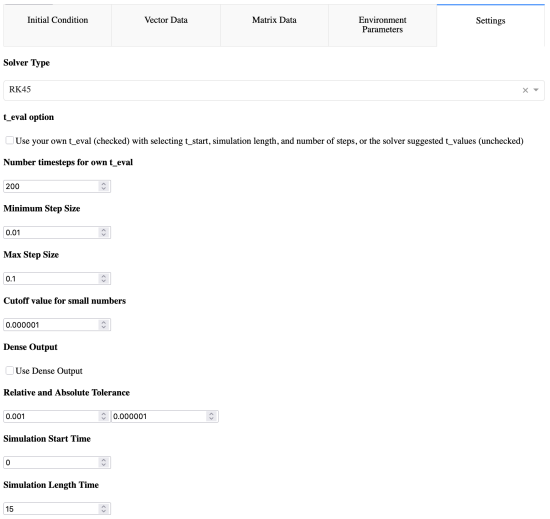
(b) The tab where the user can edit the matrix attribute values.



(c) The tab where the user can edit the vector attribute values.



(d) The tab where the user can edit the environment values.



(e) The tab where a user can edit the settings of the solver and simulation.

Figure 3.2: The tabs where the user can edit the various parameter values and control the simulation parameters

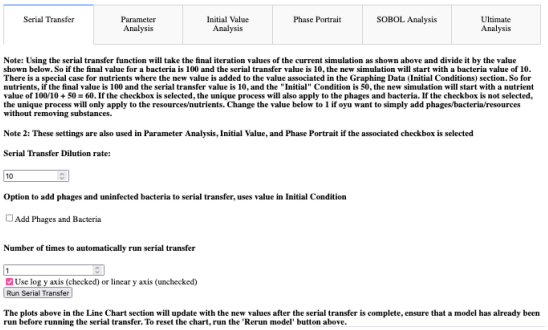
3.2.5.2 Visualization and Analysis

In the analysis section, the user can run different analysis methods to gain a greater understanding of the model. For simplicity, the visualizations only support a $1 \times 1 \times 1$ model, in order to make the analysis easier for the user, and to make it easier to analyze the visualization as the aim of the tool is to gain a deeper understanding of the interactions in a reduced complexity environment. These advanced visualizations were created with the mind of understanding a simple network. There are five different analysis and visualization methods, and one system where the user can run a large simulation on the whole network and receive an output file containing the raw simulation file data. The raw data is stored as a *parquet* file, a tabular-like data format, which when combined with Dask [54], allows for querying of the data similarly to Pandas. Parquet with Dask offers superior performance and data storage solutions that Pandas can't offer. Once queried, the user can create their own graphs and plots as they have access to the parameter values used and the raw simulation data.

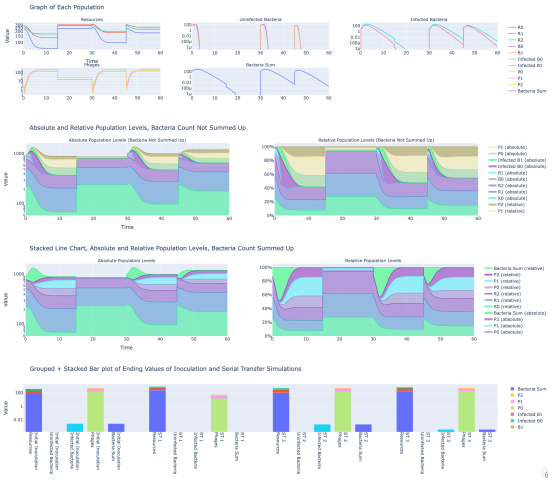
Serial Transfer Serial transfer (ST) is a method employed by bacteriologist where after a set amount of time, the bacteriologist pipettes a specified amount of media (for example 10ml of liquid) containing bacteria and resources, possibly with phages, and transfers the old media into a solution containing new media. At this stage, the bacteriologist can introduce new agents, or re-introduce agents if the agent population or concentration has died out. However, usually only resources are added during the transfer process. An example would be an experiment starts with 50ml of solution. The experiment runs for 24 hours before 5ml is removed. Researchers can run various tests, such as using optical density measurements to assess bacterial density in the solution or employing a mass spectrometer to determine the concentration of the resources. The 5ml is then re-added to a new solution of 45ml containing fresh resources. The effect that this has is it creates a sort of artificial stable point. As the bacteria grow, they consume the resources found in the solution. However eventually the resources run out, and the bacteria die out due to a lack of resources. By introducing new resources at set time intervals, the bacteria can regrow and exhibit a semi-stationary behavior.

The implementation of ST is slightly different. A user can select a number which will divide the population count of the agents by that number (Figure 3.3a). Then the program takes the IC values defined for the resources the IC in Section 3.2.5.1 and adds those values to the resources respectively. By selecting a checkbox, the values as defined in the IC box for phages and bacteria in Section 3.2.5.1 can optionally be added as well. As an example, if at the end of a simulation, there are 120 resources, 5000 bacteria, and 1000 phages remaining and the chosen ST value is 15, then the resource, bacteria, and

phage values would be decreased to 8, 333.33, and 66.66 respectively. Then, if the IC for the resources, bacteria, and phages in Section 3.2.5.1 are 500, 80, and 10 respectively, and the checkbox is unchecked, the new population count will be 508, 333.33 and 66.66 respectively. If the checkbox is checked, the new population count will be 508, 413.33, and 76.66 respectively. These new values would be used as the new starting IC for a new simulation, and the run results will be appended to the previous run. As output, new graphs are created showing the runs appended to one another, with an example output shown in Figure 3.3b.



(a) The section where the user can set up the ST.

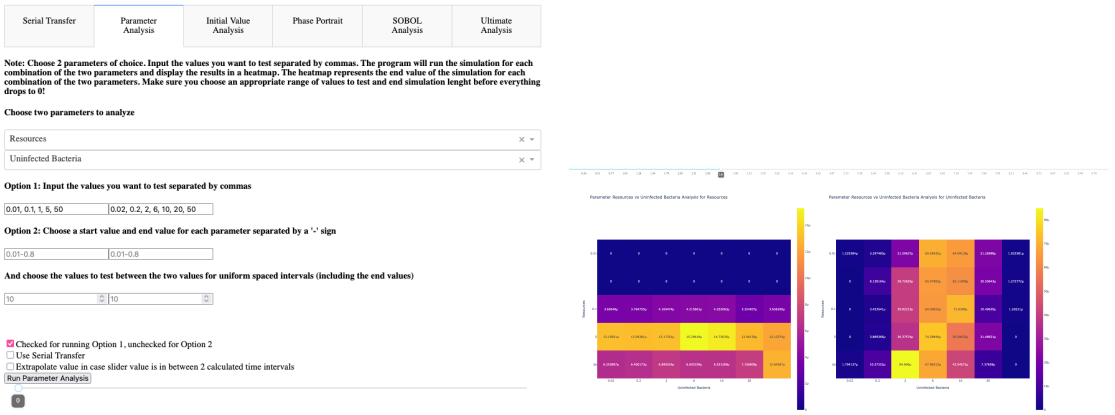


(b) The output plots of ST.

Figure 3.3: Serial Transfer

Parameter Analysis The parameter analysis (PA) settings tab as shown in Figure 3.4a allows the user to choose two parameters and individually run the model with the varying input values. The values that can be tested and changed include all IC values, vector and matrix data, and environmental data. As input, the user can select 2 parameters of choice. After the parameter name selection, the user can manually choose which parameter values they want to test or test a range of values equally spaced by selecting the number of values to test. Finally, the user can optionally run a ST, where the ST uses the settings found on the ST tab.

Figure 3.4b shows the heatmap that the user can expect, one heatmap for each agent type. Each heatmap cell represents the input of 2 unique parameter values, and shows the population count for that parameter run at the time shown in the slider. As the user slides the slider, the value inside the cell updates to correspond with the selected time. Note that the heatmap color range resets for each heatmap, so similar colors across heatmaps and across time will not correspond to the same values.



(a) The options available for the PP. **(b)** The output that the user can expect

Figure 3.4: Parameter Analysis

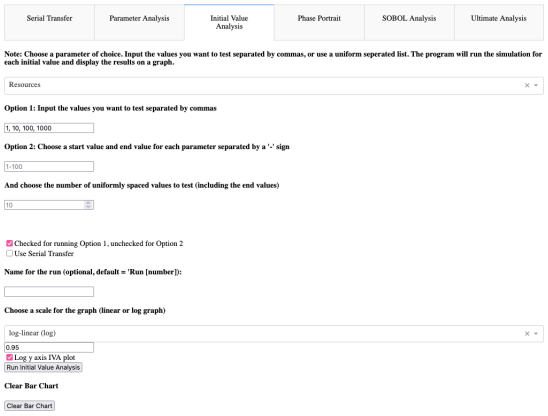
Initial Value Analysis The initial value analysis (IVA) settings tab as shown in Figure 3.5a allows the user to choose a single parameter and vary the value of that parameter, visualizing how a change in parameter value affects the population count of the agents.

Figure 3.5b shows the plots that the user receives. For each agent type, there are three plots made. The left plot shows the population count through time, one line for each parameter value submitted. The middle plot takes each run and calculates the "percentage from the max value" (default value of 0.95 $\rightarrow$ 95%) reached of the peak. This value is considered the time of peak, and is used to fix some issues that can arise where the population plateaus or only keeps on rising. The initial value is plotted on the x-axis, with the time at which the max value is reached on the y-axis. Using the plotted data, a linear or log fit can be created.

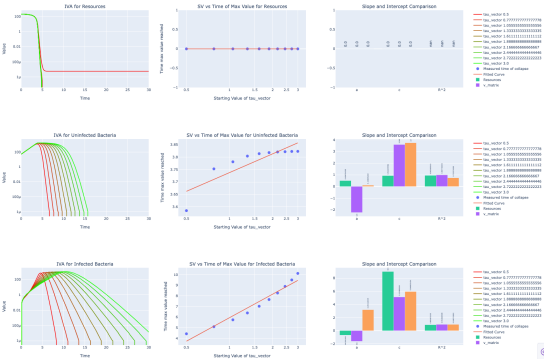
The R^2 value, or coefficient of determination, is calculated as $R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$ where y_i is the observed values, $\hat{y}_i$ is the predicted values, and $\bar{y}$ is the mean of the observed values.

Using this data can be useful for understanding how a change in parameter value affects the time at which the population count reaches a maximum. The slope, intercept and R^2 value is stored and saved in the third plot, a bar chart, with an editable name. For every re-run of the IVA, the slope, intercept and R^2 value is stored in the bar chart, allowing comparison of the slope-intercept data across different parameters.

Phase Portrait The phase portrait (PP) plot allows for the user to analyze how an agent population evolves with respect to the other agent population through time. PPs indicate how one population increases while the other decreases, and vice versa. Steady states can be identified and classified as either stable, unstable, or as saddle points. By



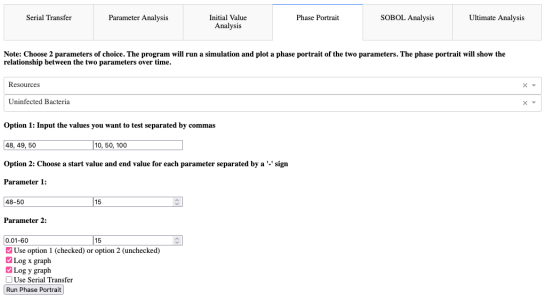
(a) The settings for the IVA tab.



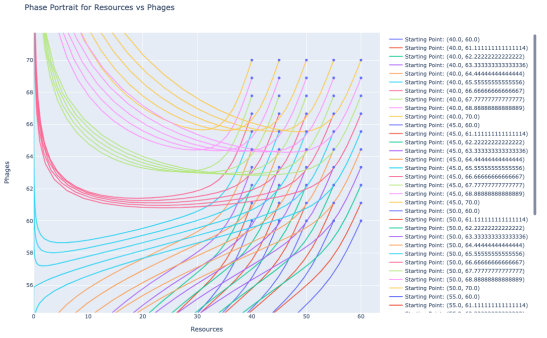
(b) An example IVA output.

Figure 3.5: The IVA settings and output.

comparing different starting points, it is possible to see if the system is chaotic or not. The setup for the PP can be seen in Figure 3.6a, and a sample output can be seen in Figure 3.6b.



(a) The user can select two starting values for the IC, but they can't choose vector, matrix, or environment settings due to the plot showing the development of agent populations against other agent populations. As typical, the user can select their own values or auto-generate values between two values, as well as use a ST option. There is also an option to take the logarithm of the x and/or y-axis.



(b) An example run of a PP.

Figure 3.6: PP settings and output.

SOBOL Sensitivity Analysis SOBOL analysis, a variance-based sensitivity analysis, is a method that allows a user to quantify how important an input parameter has on a measured aspect of the output by changing the parameter values of the model and measuring the change in model output. SOBOL quantifies how much variance in the output can be attributed to a specific parameter and can measure the effect of global/total (ST), first ($S1$), and second order sensitivity ($S2$). Global, also called total sensitivity, is the summation of all higher order interactions. First order Si, or local sensitivity, is the measurement of the effect that parameter i has on the variance of the output. Second order is the measurement of parameter i interacting with parameter

j, nad how the interaction attributes to the output variance. Etc for third order and higher. If $ST_i >> S1_i$, then parameter i depends on higher order interactions with other parameters, while when $ST_i \approx S1_i$, then i doesn't interact much with and depend on other parameters. It should be stated that $ST_i \geq S1_i$ and that ST_i can be greater than 1, while $S1_i <= 1$.

When a model is viewed as a black-box model, the model can be seen as a function $Y = f(X)$, where X is an input vector of d elements, and Y is a univariate model output. X is assumed to be independently and uniformly distributed within a hypercube $X_i \in [0, 1]$ for $i = 1, \ldots d$. The first order sensitivity measures the output variance of the main affect of parameter X_i. Measuring the effect of varying X_i averaged over other input parameters, and standardized to provide a fractional contribution to the overall output variance. The first order sensitivity is described as

$$S1_i = \frac{V_i}{Var(Y)}$$

where $V_i = Var_{X_i}(\mathbb{E}_{X_{\sim i}}[Y|X_i])$ and where $X_{\sim i}$ represents all the parameters that are not X_i. All parameters are summarized in Table 8.2

The second order index measures the impact of input X_i interacting with X_j. For many inputs, this becomes unwieldy to analyze. The global sensitivity is used to analyze the global sensitivity without evaluating $2^d - 1$ indices, and measures the contribution to the output variance of X_i, including all variance due to X_i's interaction with other variables.

$$S1_i = \frac{\mathbb{E}_{X_{\sim i}}[Var_{X_i}(Y|X_{\sim i}))}{Var(Y)} = 1 - \frac{Var_{X_i}(\mathbb{E}_{X_i}[Y|X_{\sim i}])}{Var(Y)}$$

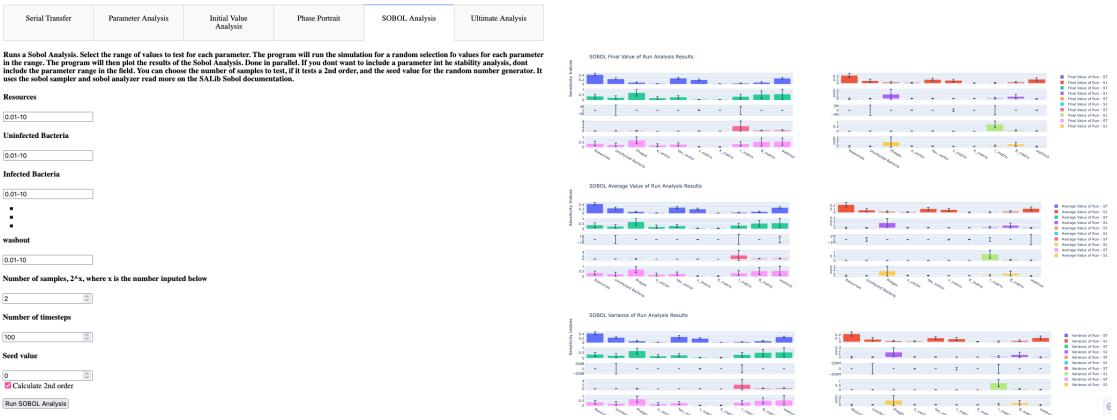
SOBOL can analyze various univariate outputs. This could be either the average value of an agent population, the variance in population count, the time at the peak of an agent count, the final population value, etc.
SOBOL accepts a list of parameter names and a list of range of values to sample from, which the user can input in the SOBOL settings tab, Figure 3.7a. If no values are added, the parameter is not included in the simulation and the default value is instead used. The user then needs to select the number of samples to run, using the formula 2^x, where x is the number they input, and 2^x is the number of samples that SOBOL will create and run. The larger x is, the more accurate the SOBOL analysis results will be, but the more simulations would need to be run.
Otherwise, if 2nd order is not chosen, the model is run $N(D+2)$ times. If the user wants to analyze the second order interactions, then the model will run the system $N(2D+2)$ times with the randomly sampled input values, where N is a multiple of 2, and D is the

number of parameters being tested. Due to the randomness of the sampling method, the user can, but does not need to, submit a seed value.

Three SOBOL analyses are included by default in the dashboard, as shown in Figure 3.7b. An analysis of the final value of the simulation, the average population count, and the variance in population count. The global and first sensitivity are shown next to one another, and each sub-row within a plot represents each agent type. The proportion of the global and local sensitivity can be seen for each agent type and each parameter.

It can be argued that the final, average, and variance value of the run is not a useful statistic to measure and run a SOBOL analysis on. One might give the reasoning that the population value at time t depends on the previous time step $t-1$. Thus the average and variance of the value is not completely random and is semi dependent on the previous value. Another argument is that the simple golden model doesn't exhibit complex behavior unlike the oscillating behavior exhibited in Figure 2.5.

Making a dashboard that can be used for different inputs is hard to make. Predicting the type of plots that a user might be interested in, and the type of behavior the user wants to analyze is impossible to predict. Therefore, three simple and easy to understand default SOBOL analysis methods are provided that aims to capture the simple dynamics of the system. Upon completion of a SOBOL analysis, the original simulation data is stored to the disk as a *.pickle* file so that the user can use the data and run their own SOBOL analyses.



(a) The SOBOL settings tab. **(b)** The output that can be expected from SOBOL.

Figure 3.7: SOBOL variance analysis settings and output.

Ultimate Analysis The Ultimate Analysis section does not produce any visualizations or analysis, but instead allows for the user to define which ICs and parameter values they want to run a simulation on. The solver will iterate over every single parameter input possibility and save the results in a *.parquet* file. Similarly settings in

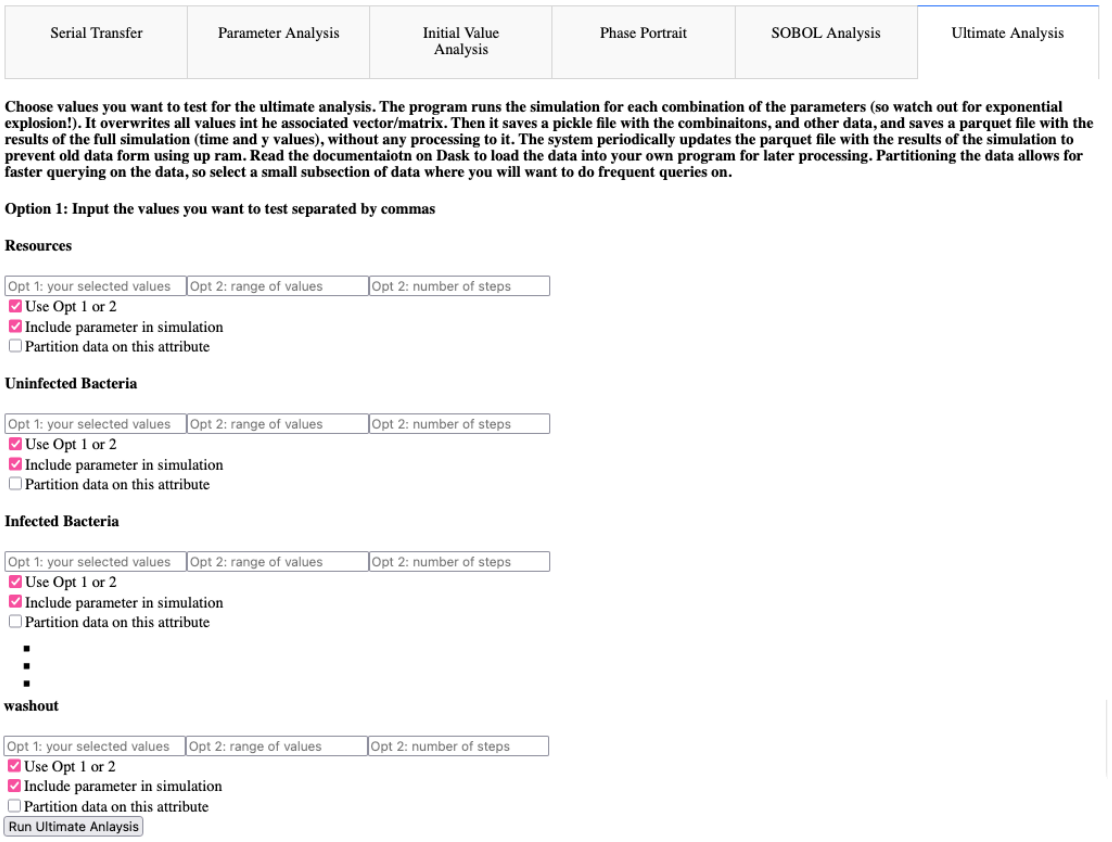


Figure 3.8: The ultimate analysis setup tab.

the other sections, the user can specify a start and end value, along with the number of values to generate evenly spaced within that range, including both the start and end values (Figure 3.8).
Using Dask and the saved *.parquet* file, the user can query for specific runs, for example runs where a parameter value was greater than 0.05, and use the simulation data to create their own plots.

3.2.6 Custom Visualizations and Analyses

The final part, an optional step, allows the user to define a number of parameters they want to simulate and download the simulation results. The user can use this data to create their own custom visualizations without having to rerun the simulations, especially if there are many simulations. The data can be further processed and visualized as the user wishes.

Depending on the provided model, different behavior might appear. As the dashboard can not create a graph for every situation, or be easily adapted to analyze every situation, Ultimate Analysis can be used to run and download the simulation data to the disk to later create your own custom visualizations. For example the agents in a model

can exhibit cyclic behavior. A custom visualization that the could be created for this cyclic model would be to perform a Fourier transformation on the curve to obtain the predominant frequencies. A change in parameter values would change the frequencies of the curve, so it would be easy to quantify how a change in parameter value affects the frequency output. If dampening occurs, then a change in amplitude can also be measured, and compared to the change in frequency, allowing the user to identify if the frequency-dampening relationship is correlated or not.

3.3 Software Used and Packages

The program was created exclusively in Python [55], and makes extensive usages of various packages, ranging from the standard scientific packages such as NumPy [56] and SciPy to more niche packages such as pickle and SALib [57, 58].

The graphical tool uses Tkinter acting as the front end, handling the user inputs, while NetworkX [59] stores the graph and contains the attribute data. The GUI tool also uses Matplotlib [60]to create the figure of the graph to display to the user in the GUI tool.

The simulation framework, the backend of the modelling, makes extensive usage of SciPy's *solve_ivp()* to create the ODE data. It also makes light usage of NetworkX to load the graph, as it initially takes a graph as an input, and light usage of NumPy to setup the parameters at startup.

The visualization part makes heavily usage of Dash and Plotly. Dash acts as the server and is used for displaying the HTML aspect of the frontend and dealing with any input and output. Upon choosing parameter values and clicking on "submit", Dash registers the activity and calls the function registered to the button, sending data such as parameter values and options like "log x-axis" form the frontend to the backend server. In the backend, the various inputs are handled, like changing the input string "0.05, 0.1, 0.15, 0.2" into an iterable list [0.05, 0.1, 0.15, 0.2] that the simulation framework can iterate over to vary the parameter value.

If there are many simulations to run through, in the case of Ultimate Analysis, an intermediate call to a parallel computing library Joblib is called, where Joblib parallelizes the for-loop to compute the simulations in parallel.

Ultimate analysis uses Pandas to write the data to a *.parquet* file. Pandas parquet offers efficient data compression, efficient memory usage and when combined with Dask, efficient querying functionalities in a Dataframe format that many data scientists would be familiar with.

SOBOL uses the SALib library to sample and analyze the parameter input. Both ultimate analysis and SOBOL save a *.pickle* file containing a dictionary with the parameter values tested, setting values, and other important information regarding the simulation.

Initial Value Analysis uses SciPy's *curve_fit()* function to curve fit the points in the middle plot (Figure 3.5b).

Other packages that are used include collections, copy, warnings, itertools, os, datetime, json, gc, and time.

Chapter 4

Experiments and Results

4.1 Graph Behavior

Table 4.1 elaborates how a change in parameter value changes the shape and population level of the agents. "A Good Curve", whose agent and parameter default values can be found in the first column of Table 4.1 and in Table 12.1 are used as a reference. Each parameter was individually changed to a higher and lower value from the reference value, and the changes were noted in Table 4.1. It should be noted that Table 4.1 provides illustrative examples rather than an exhaustive analysis. The observed behaviors reflect typical trends when varying each parameter in the indicated direction, but may not generalize to all possible values and cases, or the magnitude of the change. The results can also not be generalized to cases where two parameters values are changed at a time. Table 4.1 can be used in conjunction with SOBOL Sensitivity Analysis Results to gain a better idea of how sensitive the output is to that specific parameter.

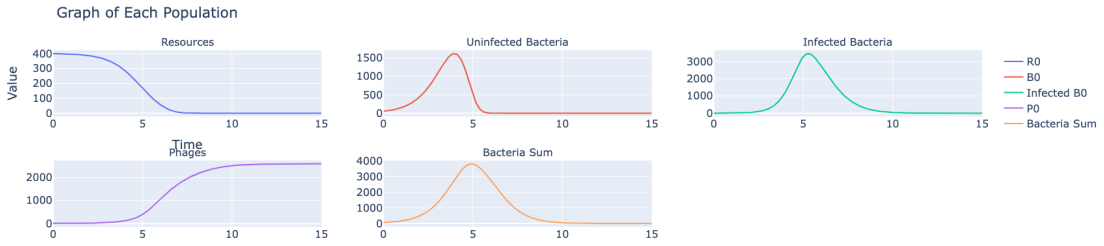
4.2 A Good Curve

Assuming a very simple model, with no washin or washout rate, a good bacteria growth curve looks like a mountain, with a clear rise, peak, and fall in population levels. For a given IC, the bacteria start to consume resources and replicate leading to exponential growth. The phages start to infect the bacteria and eventually the bacteria start to die, releasing new phages. The new phages infect more bacteria, putting pressure on the bacteria growth. Eventually, more bacteria are being infected than being created, causing the decline in bacteria population. Figure 4.1a shows an example of a good curve. Figure 4.1b is the same plot but with a logarithmic y-axis.

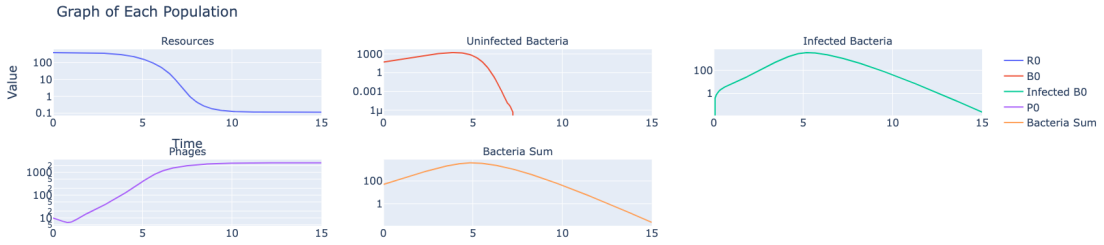
Parameter	Tested Value	Description of Behavior
R (400)	500	More uninfected and infected, slightly more phages
	300	Slightly less uninfected and infected, earlier resource depletion
U (50)	70	Slightly more phages and uninfected and infected bacteria
	30	Less uninfected and infected bacteria, slower resource depletion, not all resources used, slightly less phages
P (10)	20	Less resources consumed, less uninfected, bacteria peaks earlier, slightly less phages
	5	Resources consumed faster, more uninfected, infected, and phages
τ (2.14)	10	Faster resource depletion, faster bacteria peak, plateau, then fall in population. more uninfected and infected, less phages
	0.5	Barely any resource consumption, little bacteria growth and uninfected, more phages
ω^i (0)	15	Slightly more bacteria, resource replenish after bacteria die out
e (0.03)	0.1	Faster resource depletion, sharper decline in uninfected, less infected and phages
	0.01	Less resource consumption, slightly more bacteria
v (1.2)	1.8	More phages, significantly more bacteria, earlier and sharp peak in uninfected, ,
	1	Less phages and bacteria, less resource consumption, earlier bacteria peak
K (10)	100	Less resource consumption, less bacteria and phages, earlier bacteria peak
	1	Faster resource depletion and sudden stop instead of gradual slowdown, earlier bacteria peak
r (0.01)	0.1	Less consumption, less infected and phages, earlier peak in bacteria
	0.001	Faster resource consumption rate, more infected and phages, delay in bacteria peak, sharp bacteria peak, small plateau in bacteria count before drop
β (20)	50	More phages, earlier bacteria peak, less resources consumed, less bacteria
	10	Faster resource consumption, more uninfected, less phages, sharper bacteria peak
ω^o (0)	0.02	Faster resource depletion, more bacteria and sharper peak, later peak, and less phages.

Table 4.1: A table that compares how moving one individual parameter value up or down relative to the "A Good Curve" changes the general shape of the curve. This table is not meant to be exhaustive, cover edge cases, or extreme cases, or cover every exact detail and change in the population graph, but just to give an idea of how a change in parameter influences the graph shape, such as the rate of resource depletion, maximum number of bacteria and phages, and change in peak time. Reference parameter values are provided in the parentheses, from Table 12.1.

As the bacteria population grow, the resource consumption speeds up until there are trace amount of resources left at $t = 8$. The uninfected and infected bacteria exhibit exponential growth, peaking at 1617 at $t = 3.99$ and 3463 at $t = 5.27$ respectively. The delay in the uninfected to infected bacteria's peak is due to the infection stages and latent period of the phage infection. The bacteria sum do not have as stark of a peak in comparison to the uninfected and infected bacteria, due to the graph measuring all bacteria populations, but the peak of 3805 at $t = 4.89$ is still clear. The phages saw a significant increase in population count at around $t = 4$, coinciding with the peak in uninfected bacteria. At this point in time, the infection rate is larger than the bacteria replication rate, so the bacteria are starting to die out even though there are still sufficient resources remaining. At around $t = 4$ is when the the resource consumption rate inflects. The rate at which the resources are being consumed starts to slow down, showing a decreasing sigmoid shape. The total bacteria population reached a peak of 3805 at $t = 4.89$, a 76.1x increase in population count from the initial 50 starting uninfected bacteria. The phage population reached a peak of 2584 phages at $t = 15$, a 258.4x increase in population count.



(a) Linear y-axis for a "good" plot.



(b) Logarithmic y-axis for a "good" plot.

Figure 4.1: The log plot allows to see behavior happening at values near 0 and to plot data on a logarithmic scale. The parameters used for this plot can be found in Table 12.1.

4.3 SOBOL Sensitivity Analysis Results

It is important to understand how a change in parameter value affects the change in output of a model. Models will have parameters that are more important and have a larger effect on the model output than other parameters.

Figure 4.2a shows the impact that the parameter had on the final value of the population at $t = 15$. THe average value and variance of population value were intentionally left out of the analysis, despite being a part of the dashboard because the SOBOL sensitivity values are almost identical to the final sensitivity values. There were some very minor differences from bar to bar across plots, but the difference was imperceptible. Since the plots all look similar, only an analysis on the final value, Figure 4.2a will be done.

The parameters that were tested include all the parameters listed in the extended golden model, except for Uninfected Bacteria and M. Uninfected Bacteria was left out as it doesn't make sense to already add infected bacteria to the system M, the number of stages that the infection goes through, can not be tested as M hardcodes the number of infection stages that the bacteria has to go through. The hardcoding is done before the simulation framework starts. As such, it is not possible to change M.

4.3.1 Final Value Analysis

4.3.1.1 Resources

The ω^i/washin rate had the largest influence on the final, average and variance value. Without a washin rate, the resources will most likely have been consumed by the time the simulation ended at $t = 15$. The final values for Resources, Uninfected, Infected, and Phages would often be something similar to (0, 0, 0, 10000) at $t = 15$, where all the resources were consumed and the bacteria died out due to the phages, leaving only the phages remaining. The final value of the resources would often be 0, no matter what parameter values were used, with $\omega^i, \omega^o = 0$. With the addition of the washin, new resources were constantly being re-added. Once the bacteria died out, the resources could accumulate, with the accumulation dependent on the rate of the washin rate, hence why the washin rate has such a large impact on the final, average, and variance of population value for the resources. The final value would be dependent on when the bacteria died out, in turn allowing the resources to accumulate at a rate proportional to $\omega^i - \omega^0 \cdot R$. Resources were less dependent on higher order interactions, unlike the uninfected, infected, phages, and total bacteria sum.

4.3.1.2 Uninfected

The uninfected bacteria population sensitivities depend on many higher order interactions between the parameters as $ST_i >> S1_i$. The uninfected are highly dependent on β/B_matrix and initial phage population, as the initial phage population will determine how many bacteria become infected, and how quickly the phages can proliferate through the bacteria population. Surprisingly, r/r_matrix did not have as big of an influence on the uninfected as β did, even though the infection rate is dependent on r. The larger or smaller r is, the faster or slower the infection rate is. If r is really small, the infection rate would take forever, potentially allowing the bacteria to keep a stable population. r is equally as important at explaining the final value as τ/tau_vector, washin, e/e_matrix, and washout of sensitivity around 0.25.

4.3.1.3 Infected

Since $ST_i >> S1_i$ for the infected bacteria, where $S1_i \approx 0$ for nearly all of the parameters, the infected bacteria heavily depend on many interactions happening at the same time. This makes intuitive sense after looking at The Golden Model. The infected (and uninfected) bacteria directly interact with R, U, P, v, K, r, τ, and ω^o (M is not included as it was left out of the analysis). So due to the high coupling of parameters, the infected (and also the uninfected) have large global sensitivities compared to the local sensitivity.

However SOBOL had some difficulties assigning a good sensitivity score to each parameter for the ST and $S1$ tests as noticed by the slightly larger error bars in the infected than the uninfected or resources. This is most likely due to the infected bacteria going through multiple stages of infection, causing a delay and uncertain behavior in the final value, despite the ODE model being deterministic.

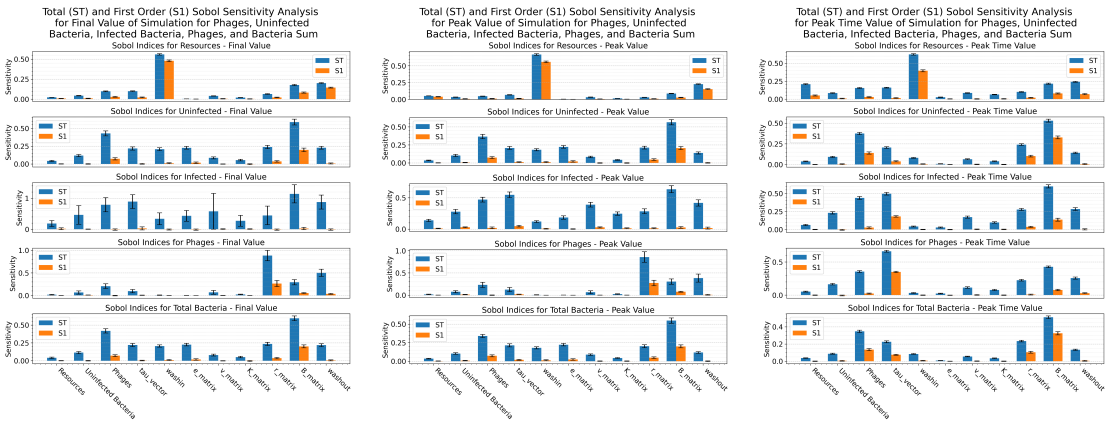
4.3.1.4 Phages

The most important factor for the final phage value is r, followed by β, ω^o and P. The r value allows the phages to infect the uninfected. When r is decreased, the final phage population is counterintuitively higher than when r is larger. The behavior is counterintuitive because one would expect that a higher infection rate would lead to more infections and thus more phages. With a higher r value, more phages are being removed from the phage population and infecting the bacteria. It can be seen as a way of "more phages are needed to infect a bacterium", therefore getting less phages out as a result as more phages are needed to infect a single bacteria.

Washout has a noticeable influence on the phage population, as not the phage population is being reduced at a rate proportional to the washout rate. The larger the washout rate, the larger the drawdown of phages. When the infected all die out, the phage population wont grow anymore. Given the phage population at that point in time, the phage removal rate is proportional to the washout rate.

4.3.1.5 Total Bacteria

Total bacteria is the sum of both uninfected and infected bacteria, so it makes sense for total bacteria to have similar values to uninfected and infected bacteria. Apparently the uninfected bacteria have a stronger influence on the output variance than the infected bacteria. The total bacteria sensitivities resemble the sensitivities of the uninfected bacteria more than the infected bacteria. It would have been expected for the total bacteria to resemble more of an average between the uninfected and infected.



(a) Final population value. **(b)** Peak population value. **(c)** Time at peak population.

Figure 4.2: SOBOL analyses for the average, peak, and peak time. The data was saved from the dashboard and plotted using Matplotlib. The average and variance analysis results were left out for nearly identical results to the final value. The values used for this SOBOL test can be found in Table 12.3. The data used in Figure 4.2a was used for Figure 4.2b and Figure 4.2c. The plot of the average and variance analysis can be found at Figure 13.1a and Figure 13.1b

4.3.2 Custom SOBOL Analysis - Peak Value and Peak Time

Due to the similarity of the final, average, and variance value as seen in Figure 4.2a, Figure 13.1a, and Figure 13.1b a custom SOBOL analysis that isn't included in the dashboard might result in a different SOBOL analysis result. To create the custom SOBOL analyses, the peak value and the time at the peak of the population is measured and analyzed. The peak is defined as the point where the population reaches 95% of its absolute maximum value. The time at peak is measured at the point in time that the

population reaches 95% of the maximum value. This removes unintended side effects of the simulation. For populations that are only increasing in value, this prevents the measured peak from bunching up at the end of the simulation, skewing the data. As the peak is defined at 95% of the absolute maximum value, populations that have a faster increase on population count at the end will have a time value closer towards the end of the simulation. For populations that reach a plateau, the 95% rule will push the peak time towards the beginning of the simulation, while still "respecting" the absolute final value since $95\% \approx 100\%$. The 95% rule can fail under certain situations, such as when there is cyclic behavior. See Why 95%? for a more detailed explanation on why the 95% rule is used.

The results of the SOBOL peak and time at peak analyses can be seen in Figure 4.2b and Figure 4.2c. Although some of the bars between the final, peak, and time at peak values are the same, some are different. But overall, similar values can be seen across the the final, peak, and time at peak analyses. The peak infected values are more certain compared to the final infected values, which could be due to the 95% rule removing some of the noise of the simulation. The time at peak values have less error compared to the final and peak value. This is due to the restricted range of values. The time at peak value can only fall somewhere between 0 and 15, the start and end values of the simulation respectively. The final and peak values can fall anywhere between 0 and any value, depending on the IC and how high the population can rise, and how fast the population can fall, *if* the population count falls.

4.3.3 SOBOL Analysis - Without Washin and Washout

In many of the plots, the washin and washout rate had a large influence on the final, peak, and time at peak value. Figure 13.2 ran the same input as Figure 4.2, but left the washin and washout rate out. The sensitivity plots for the final, average, variance, peak, and time at peak plots look different form one another. The final resource value depended heavily on the washin and washout rate, but without the washin and washout, the final resource depended heavily on the initial resource population. Since $S1 \approx ST$, the resource parameter does not depend on other higher order interactions.

The peak value for the resources without washin and washout only depended on the initial resource consumption. Since there was no washin, no resources could be added, so the peak for the resources was always at $t = 0$, and was dependent on the initial resource value. The time at peak value is always 0 as the resources are only being depleted, so no matter the change in parameter values, the parameter had no effect on the peak time, so SOBOL gives a value of 0 to every parameter for the resources. β

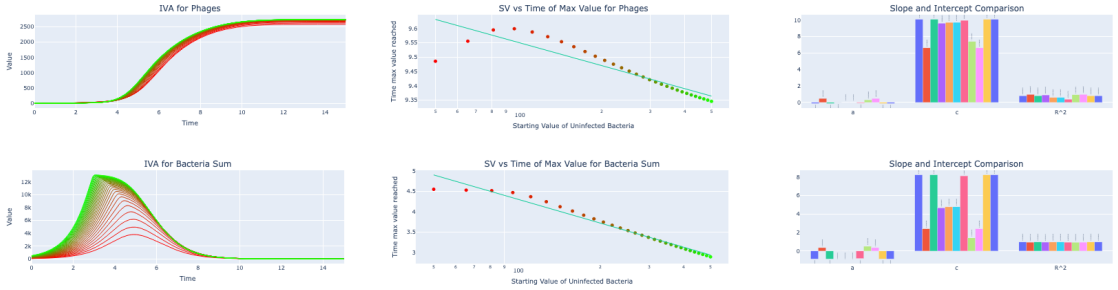
consistently had a large effect on the final, average, variance, peak, and time of peak value as

4.4 Initial Value Analysis Results

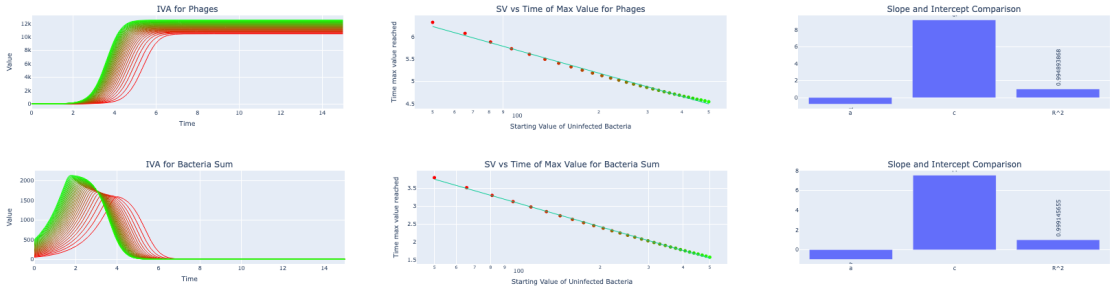
The IVA section of the dashboard allows the user to visualize how a change in parameter value affects the population growth of the agents.

In Figure 1 of Mulla et al. [45], they vary the initial concentration of bacteria and measure the time until bacterial collapse. The initial concentration value and time of collapse is plotted on the x and y-axis with a tight linear regression fit on the data. The observed logarithmic decrease suggests that the phage kinetics is adsorption-limited meaning that Figure 4.3b replicates this graph.

Figure 4.3a, even considered a "good" curve, shows interesting behavior that diverges from that of Figure 4.3b. Then though the behavior should be similar, changing other parameter values can alter how a parameter works. It would be expected that for10 initial uninfected bacteria and less the bacteria sum peak time would follow the linear regression line, but at around 100 uninfected bacteria and less, the peak curve goes horizontal. This suggests that something is restricting the bacterial growth.



(a) IVA for Table 12.1. Replicates Figure 1 of Mulla et al. [45].



(b) IVA for Table 12.2. For uninfected bacteria less than 100, the phage-bacteria interaction is resource limited. For 100 and higher, it is adsorption limited.

Figure 4.3: Varying initial Uninfected Bacteria concentration, from 50 to 500, with 30 unique values tested over two different instances of "good" curves. Even with two "good" curves, even varying the default parameter values a tiny bit can have a large influence on how changing the initial bacteria concentration can have an influence on the dynamics of the system, changing the behavior of the peak time. The default values for Figure a) and b) can be found at Table 12.1 and Table 12.2.

Chapter 5

Discussion

Chapter 6

Conclusion and future work

6.1 Conclusion

6.2 Future Work

Next steps would be to give the model to the lab technicians running lab experiments so that they can verify the results as seen in the output by comparing the lab results with the model output. With the lab results, the model can be adapted to better fit the lab results. This can be done by changing parameter values, or by changing the model equation. The user can decide to add the Monod microbial growth model to the growth of the bacteria, or adapt the Monod equation to being dependent on multiple sources. Using the model, the technicians can improve and validate their methods. If the empirical results significantly deviate from the model results, the technician can review to see if their method is good. They might have accidentally not added enough resources, or accidentally miscalculated the initial concentration of bacteria.

6.2.1 Other Models

"All models are wrong, but some are useful" — George E. P. Box

This quote could not be more true for modelling phages and bacteria. There are numerous considerations to account for, and there are numerous ways to go about the considerations. Each model has its pros and cons. Take the exponential population

growth model

$$\frac{dP}{dt} = rP \tag{6.1}$$

$$P(t) = P_0 e^{rt} \tag{6.2}$$

where $P(t)$ is the population at time t, P_0 is the initial population, and r is the growth rate. This model acts as a nice introduction to population modelling. It can accurately fit the exponential growth bacteria experience in a petri dish. However, this basic model does not account for a spatial and resource consumption. Eventually the bacteria run out of space and resources, and start to die out. A population can not grow exponentially forever, the resources can only support a maximum population, the carrying capacity. The model can be adapted to include a carrying capacity (the max population level that can be reached), where the new updated model is

$$\frac{dP}{dt} = rP(1 - \frac{P}{K}) \tag{6.3}$$

$$P = \frac{K}{1 + (\frac{K - N_0}{N_0})e^{-rt}} \tag{6.4}$$

where K is the carrying capacity. This adapted model, the logistic growth model better accounts for the eventual restriction of population growth.
Figure 6.1 shows how the carrying capacity has a large influence on the speed and growth trajectory of a population. The logistic curve initially follows the exponential curve before the maximum growth rate is reached and starts to slow down and taper off as the population asymptotically approaches the carrying capacity $K = 200$.

A further step would be to introduce competition between other bacteria. For example, a $p_{0,1} \cdot P_0 \cdot P_1$ term can be subtracted from the logistic growth curve. This term accounts for competition between Populace 0 and Populace 1, with $p_{0,1}$ being the interaction factor between P_0 and P_1. Assuming P_0 is being looked at and P_0 has a high value, if P_1 is high, then a lot of P_0 is going to die out due to the competition with P_1. If P_1 has a low population, then not many P_0 are going to die out due to less competition with P_1.

The model can be further extended by accounting for temperature, pH, more interactions between other agents, the constant addition and removal of other agents, and other considerations.

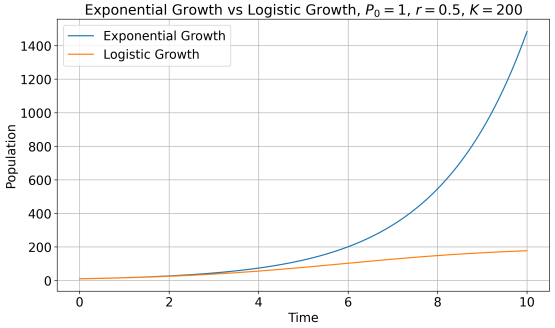


Figure 6.1: Exponential growth curve vs logistic growth

6.2.1.1 Spatial simulations

https://www.sciencedirect.com/science/article/abs/pii/S0022519318305368 The ODE models work very nicely when there is no consideration for space and 2D/3D-space dimensionality. Spatial models complicate the simulation, making it harder to analyze. Data collection and analysis becomes harder. Unique and novel analysis and visualization methods have to be created to be able to represent and visualize the data through space and time.

PDE PDE are the next logical step to add space to an ODE model. The general formula, as given by

$$\frac{\partial u}{\partial t} = D\nabla^2 u + f(u, x, y, \ldots, t)$$

where $u(x, y, \ldots, t)$ is the population density of interest, D is the diffusion constant, ∇^2 is the derivative of each spatial direction, and $f(x, y, \ldots, t)$ is the function encapsulating growth, death, and interactions dynamics.

Discretization The dimensions can be discretized into boxes of dimensions $\delta x, \delta y, \ldots$. This transforms the PDE into a system of difference equations, which can be solved numerically. For example, the Laplacian term $\nabla^2 u$ in 2D can be approximated using finite differences as:

$$\nabla^2 u \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\delta x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\delta y^2}$$

where $u_{i,j}$ represents the value of u at the grid point (i, j). This discretization allows the PDE to be solved iteratively over a grid, enabling spatial simulations of population dynamics. Each box can be represented by a matrix, and the population value can be displayed as a heatmap using visualization software.

Chapter 7

Ethics and Data Management

A new requirement for the thesis is that there must be a short section in which you reflect on the ethical aspects of your project. This requirement is related to one of the final objectives that a graduated student of the Master of Computational Science must meet: "The graduate of the program has insight into the social significance of Computational Science and the responsibilities of experts in this field within science and in society". You don't need to devote an entire chapter to this; a short section or paragraph is sufficient.

I acknowledge that the thesis adheres to the ethical code (https://student.uva.nl/en/topics/ethics-in-research) and research data management policies (https://rdm.uva.nl/en) of UvA and IvI.

The following table lists the data used in this thesis (including source codes). I confirm that the list is complete and the listed data are sufficient to reproduce the results of the thesis. If a prohibitive non-disclosure agreement is in effect at the time of submission "NDA" is written under "Availability" and "License" for the concerned data items.

Short description (max. 10 words)	**Availability (e.g., URL, DOI)**	**License (e.g., MIT, GPL, Creative Commons)**
Example dataset 1	<github url>or Figshare	GPL
Example source code	DOI (from Zenodo)	MIT
Example sensitive data	NDA	NDA

Chapter 8

Appendix A: Equation Parameters

Parameters used in the various equations.

8.1 Simple/Advanced Golden Model Parameters

8.2 SOBOL Parameters

8.3 Linear Regression Parameters

Variable	Name	Description
P/P_p	Phages agent	Phage population for phage p
U/U_b	Uninfected Bacteria agent	Uninfected population for bacteria b
I_i/I_{b_i}	Infected Bacteria agent	Infected population for bacteria b at stage $1, \ldots, i, \ldots, M$
B/B_b	Bacteria agent	Total bacteria population for bacteria b, assuming $B_b = U_b + \sum_{i=1}^{M} I_{b_i}$
R/R_r	Resource agent	Resource r concentration
$e/e_{b,r}$	Consumption rate	Rate at which resources are consumed
$\beta/\beta_{p,b}$	Burst size	Lytic burst size for phage p and bacteria b
$r/r_{p,b}$	Successful phage/cell encounter	Probability of a successful bacteria b infection from phage p
τ/τ_b	Latent period	Time it takes bacteria b to go through one infection stage
$v/v_{b,r}$	Maximal growth rate	Growth rate of bacteria b from resource r
$K/K_{b,r}$	Monod Constant	Monod constant for bacteria b's resource consumption rate dependent on resource r concentration
ω^i/ω^i_r	wash-in rate	Rate of resources r being added
ω^o/ω^o	wash-out rate	Rate of agents being removed, acts on all agents equally
M	Number of infection stages	Number of infection stages that a bacteria goes through, all bacteria agents have the same value for M
t	time	time value

Table 8.1: Golden model parameters with variables, names, and descriptions. Subscripts on parameters indicate relationships; for example, $e_{b,r}$ is nonzero if there is an edge connecting bacteria b to resource r in the network, zero otherwise.

Variable	Name	Description
Y	Univariate parameter output	univariate model output, such as mean μ or variance σ
X	Input vector	Vector of size d, input vector to f
i	Parameter input	Parameter i of input
X_i	Parameter value	Value of vector X at position $i = 1, \ldots, d$, the value of parameter i
d	Input size	Size of input vector X
$X_{\sim i}$	Parameter input	All values of X that are not X_i
f	Function f	Arbitrary black-box function describing model
N	Samples	Number of samples, power of 2, 2^x
D	Parameter input size	Number of parameters inputted into SOBOL, $d = \lvert X \rvert$
ST_i	Global sensitivity	Contribution of parameter X_i to output variance of Y due to interactions with other variables
$S1_i$	First order sensitivity	Contribution of X_i to output variance of Y

Table 8.2: SOBOL parameter symbols, name, and description.

Variable	**Name**	**Description**
a	Slope	Slope of the linear regression line
c	Intercept	y-intercept of linear regression line
R^2	Regression Coefficient	Coefficient of determination of linear regression fit, quality of regression
x_i	Data point	Data point on the x-axis
y_i	Actual Value	Actual value of data for a given x_i
$\hat{y}_i$	Predicted Value	Value predicted of equation for a given x_i
$\bar{y}$	Average Value	Average y value
n	Number of samples	Number of samples being tested

Table 8.3: Variable symbol, name, and description used for the linear regression.

Chapter 9

Appendix B: Industrial and Real Life Applications of Phages

Due to the nature of killing bacteria, there are numerous applications where a researcher or an organization might be interested in controlling bacterial populations.

A Food Safety Specialist might be interested in introducing a solution containing a high concentration of phages during food production to prevent the spread and growth of *Salmonella* or *E. coli* in the pet food. Alternatively, the Food Safety Specialist might want to promote beneficial bacteria like *Streptococcus thermophilus* used in the production of Emmental cheese, which heat would kill when the milk undergoes the pasteurization process.

A doctor might be interested in providing swallowable pills, more commonly known as phage cocktails, to a patient with a bacterial infection. There is evidence that phage-resistant bacteria are more susceptible to antibiotics, so the doctor might prescribe both medicines to effectively deal with the infection.

An Environmental Protection Officer might be interested to see how they can use phages to stop the spread of *Cyanobacteria* blooms in waterways, more commonly known as blue-green algae, a photosynthetic microscopic organism that is technically a type of bacteria. This would keep waterways safe for boating and swimming activity, aquatic life, and water consumption in farms, factories, and homes.

When there are a few known bacterial strains, a targeted concoction of phages can be used to control the bacterial population growth in any setting, either be it food, healthcare, or environmental. Phages offer properties of microbial control that other methods do not, making them an ideal candidate for some applications.

9.1 Controlling Foodborne Bacteria

Foodborne diseases are one of the primary ways for bacteria to spread to humans and animals. Some bacteria use the food as a vector to infect hosts, while some bacteria will deposit toxins on the food that is then ingested. If consumed in large enough quantities, or further produced in the host, the toxins can be fatal to the host.

Methods exist to control bacterial growth, for example by storing food below 5°C or above 60°C. Bacteria need moisture to grow, so starches like rice will have minimal bacterial growth. Bacteria prefer to live in slightly acidic to neutral pH environments, so having an environment that is extremely acidic like vinegar will prevent bacterial growth. The use of chemical antibacterial agents such as bleach is not desirable due to leaving chemicals on the food, which can be fatal if ingested. Physical agents like heat or radiation can kill bacteria, but at the cost of altering the food quality [61].

For example, *Streptococcus thermophilus* is one of three different bacteria strains used to create Emmental cheese. However, Emmental cheese does not use pasteurized milk, increasing the risk of *E. coli*. Emmental cheese producers can add phages that target *E. coli* to the milk during the production stage, while not affecting the bacteria used to produce the cheese.

9.1.1 Current Applications

Phage cocktails like SalmoFresh^{TM} have been proven to safely reduce *Salmonella* contamination in pet food and raw pet food ingredients [6], as well as in romaine lettuce and bean sprouts [7]. Pet food contains meat and vegetables, where vegetables grown in or on the ground are at risk of *Salmonella* due to contact with soil, manure, compost, and other agricultural runoff from neighboring farms [13]. Figure 9.1 and Figure 9.2 show how application of phages have reduced the count of *Salmonella* in ingredients used in pet food as well as romaine lettuce and bean sprouts. In Figure 9.1, each food group noticed at least a 68% reduction in CFU/g compared to the control when the 9×10^6 phage treatment was applied. There was at least an 80% reduction in CFU/g across all food groups when treated with a 9×10^6 or stronger phage solution. In Figure 9.2, the lettuce and bean sprouts noticed a reduction of at least 0.6 log CFU/mL in *Salmonella* count across all temperature ranges. The smallest reduction in bacteria count in lettuce was noticed at 1 hour at 2°C with an absolute reduction in 62.0% between the control and treatment, while the largest reduction in bacteria of 90.0% was found at 72 hours at 2°C. For the bean sprouts, the lowest reduction in phages was found at 1 hour at 2°C with a reduction of 78.1%, and the largest reduction was 90.0% at 25°C after 48

hours. Although these values are still high above food safe, the ability to reduce the *Salmonella* population by at least 62% and up to 90% at different temperatures and incubation periods is impressive and can prolong shelf life, especially for foods that do not have long shelf lives before spoiling due to bacteria. As such, phages can be shown to control the spread of *Salmonella* in food sources and extend the potential shelf life of certain foods.

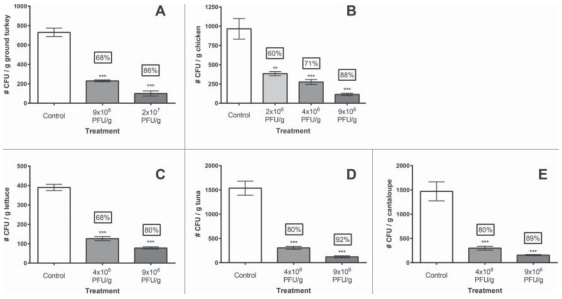


Figure 9.1: SalmoLyse® reduces Salmonella contamination on various food surfaces: Mean and standard error bars shown. Statistical analyses were carried out for each food group independently. Asterisks denote significant reduction from corresponding controls based on one-way ANOVA with Tukey's post-hoc tests for multiple corrections: ** denotes $p < 0.01$, while *** denotes $p < 0.001$ compared to the corresponding controls. There was significant reduction in Salmonella on all food surfaces with the addition of SalmoLyse® compared to the controls; the mean percent reductions from the control are noted in the boxes above treatment bars. CFU/g D colony forming units per gram. Each letter denotes a food group that was tested with SalmoLyse® and compared to a control: A= chicken; B= lettuce; C= tuna; D= cantaloupe; E= ground turkey. Plot sourced from Soffer et al. [6].

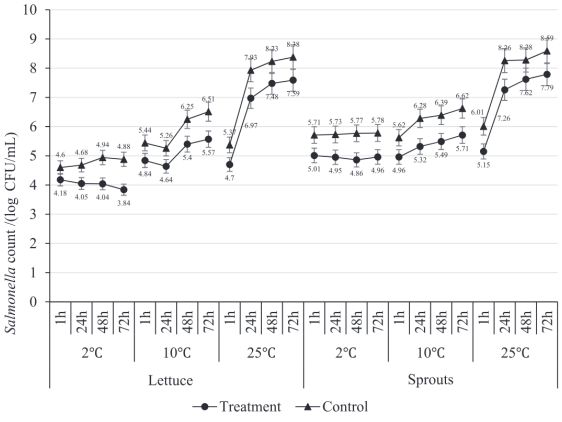


Figure 9.2: *Salmonella* count in a mixture of 5 *Salmonella* strains spot-inoculated (CFU/g) onto a) lettuce and b) sprouts after spraying with a mixture of bacteriophage (SalmoFresh™) relative to positive controls at 2, 10 and 25C and stored for 1, 24, 48 and 72 h. Plot sourced from Zhang et al. [7]

9.2 Phage Therapy and Antibiotics

Antibiotics are a common way to treat bacterial infections. However, antibiotics are not selective in the bacteria they kill, killing both harmful and beneficial bacteria. This can lead to the development of antibiotic-resistant bacteria, which makes it harder to combat that bacteria in the future. It has also been shown that antibiotics have a negative effect on the gut microbiome and brain development in mice. Phages are an alternative to antibiotics, as they are selective in the bacteria they kill and do not interact with cells or other important biological functions. The rise in antibiotic resistant bacteria can be attributed to the overuse and over-prescription of antibiotics and incorrect usage of antibiotics (for example prematurely stopping) [9]. These actions provide an evolutionary pressure on bacteria to mutate and gain resistance to the antibiotics. The phage therapy can contain any number of different phages that can target specific bacterial infections such as *Streptococcus pneumoniae* with minimal risk of side effects.

9.2.1 Current Applications: Bacterial Infection Control

One active area of research is the use of phages to control bacterial infections. Due to the specificity of phages, they can be used to target specific bacteria strains without affecting other beneficial bacteria. When sick with a bacterial infection, patients swallow antibiotic pills to help the body fight the infection. Antibiotics work by either interrupting intercellular processes like the synthesis of RNA [62], by disrupting the structural integrity of the cell wall [63], or by inhibiting protein synthesis [64].

However, antibiotics are not strain specific and indiscriminately kill gut and other bacteria. Common side effects of antibiotics, although usually not serious, include diarrhea, nausea, and headaches. It has also been shown that the effects of early-stage penicillin exposure in mice has found to have a long-lasting effect on the gut microbiome, frontal cortex gene expression, and amygdala gene expression [10]. Penicillin increases cytokine expression (small proteins used in cell signaling) in the frontal cortex of the brain, modifies the blood-brain barrier integrity, and alters behavior. The mice exhibited an increase in aggression and anxiety-like behavior [65]. Phages can be used as an alternative to antibiotics without the side effects and without affecting the gut biome.

With an increase in antibiotic usage, there has been an increase in antibiotic-resistant bacteria. The World Health Organization has stated that antibiotic resistance threatens the modern medicine and the sustainability of an effective, global public health response to the enduring threat from infectious diseases. Common infections, that previously

would have been easy to treat, are harder to treat, and can increase the risk of disease spread, severe illness, and death [66].

One area of research is exploring how bacteria can exchange traits such as phage resistance and antibiotic resistance. Some bacteria are multi-drug resistant, and don't react with the medicine anymore.

Laure and Ahn [11] showed evidence that *Salmonella Typhimurium* is more susceptible to ampicillin in the presence of phages, and phage-resistance can lead to reduced virulence and decreased antibiotic resistance.

Zhao et al. [12] showed that there exists an antagonist coevolution between the bacteria and phages, where the dynamics changed from an arms race dynamic (ARD) to a fluctuating selection dynamics (FSD). Due to phage selection and bacterial competition pressure, when the bacteria gained phage resistance, it lost antibiotic resistance. A genome analysis revealed mutations in the btuB gene of *Salmonella anatum*, with q higher mutation frequency in the ARD stage. A knockout experiment confirmed that the btuB gene is a receptor for the JNwz02 phage and resulted in reduced bacterial competitiveness. Further analysis detected multiple single nucleotide polymorphism (SNP) mutations in the phage-resistant strains. The SNPs potentially affected the membrane components, partially weakening the cell defense against antibiotics. These findings help advance our understanding of phage-host-antibiotics interactions and the impact of adaptations to antibiotic resistance. The research shows how phages can be used to reintroduce antibiotic susceptibility to previous insusceptible bacteria, preventing costly and lengthy research in new antibiotics [12].

Phage research is facing challenges due to bacterial strains evolving resistance to phages. Understanding the interplay between antibiotics and phages is essential for shaping future research [12].

9.3 Environmental Protection

Algae blooms, also called red tides, is the rapid spread of bacterial or algae organisms. Blooms are a growing environmental concern impacting water quality, aquatic ecosystems, and human health. These rapid increases in algae populations, often fueled by excess resources like nitrogen and phosphorus, can occur in freshwater, coastal, and marine environment.

Cyanobacteria blooms have major effects on the aquatic environment as well as human health. Cyanobacteria release nitrogen and phosphorous, which the bacteria use to grow

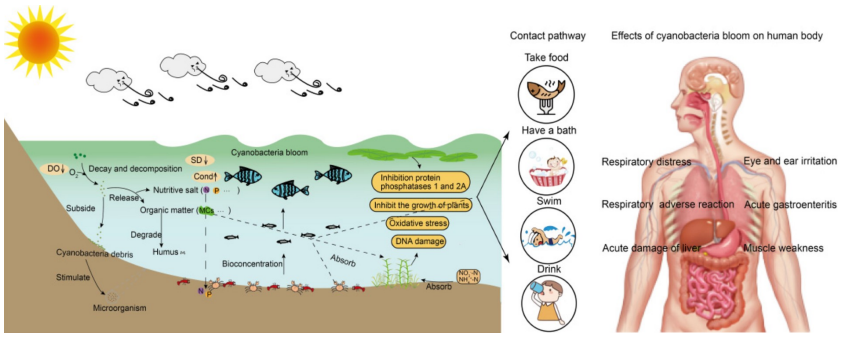


Figure 9.3: Cyanobacteria degradation cycle, main hazards of cyanobacteria bloom to water bodies, aquatic organisms, and the human body. (DO: dissolved oxygen; SD: water transparency; Cond: conductivity; N: nitrogen; P: phosphorus; MCs: microcystins). [8]

with oxygen, outpacing other aquatic growth, and killing aquatic marine life. Bacterial toxins can make their way into the food and water consumed by humans, causing muscle fatigue, respiratory issues, liver damage, and gastrointestinal issues [8]. Figure 9.3 shows the process of how cyanobacteria degrade and are absorbed into the environment, eventually making their way into the human body via various contact points.

9.3.1 Current Applications

There is interest in using phages to control cyanobacteria blooms. Phages can offer better and safer options than chemical options when trying to control bacterial blooms. Chemical options are indiscriminate, killing cyanobacteria, while also killing other beneficial bacteria and aquatic life, and can eventually seep into groundwater. Although not used to control bacteria blooms, some chemicals like PFAS, also called "Forever Chemicals", can last a long time in the environment and don't degrade and keep on negatively affecting the environment. Due to the specificity of phages, only the cyanobacteria will be targeted, and will not affect the surrounding environment.

Tucker and Pollard found that an isolated phage cocktail collected from Lake Baroon in Australia could decrease the abundance of *M. aeruginosa* by 95% within 6 days in a lab setting, before recovering within 3 weeks time [15].
There is evidence that phage-resistant bacteria can influence the population dynamics of other bacteria. It has been shown that the plankton level has been experimentally affected by the frequency of the phage-resistant *Nodularia* marine bacteria. Populations with high phage resistance ($> 50\%$) dominate the plankton communities despite a high phage count and eventually out compete other bacteria due to their slower loss in population count. Contrastingly, populations of bacteria with low phage resistance (between 0% and 5%) were lysed to extinction, releasing resources like nitrogen. This allows for other bacterial strains to absorb the resources and dominate the bacterial

community. Phages and the lysis of bacterial strains can have a dramatic effect on community dynamics and composition of other agents like phages, bacteria, and resources [14]. Phages have the potential to be used as a highly specific strategy for the control of cyanobacterial blooms, with minimal effects to the environment, and offer control of bacterial blooms, with limited impact to the environment. Usage should be relatively safe, novel, efficient, and sensitive.

However, there are issues with using phages to control bacterial blooms. Bacterial blooms can cover vast areas, or be in areas that would be hard to reach like marshlands, applying phages to combat the bloom might be infeasible. If the method of choice was to spray a solution of water containing phages, the solution needs to be shipped to the site and loaded onto special boats to spray the solution into the water, or the trucks need to drive along the shore and spray the solution into the water.

The phage density in the solution will have to be relatively high to quickly combat the bloom. These problems provide major logistical problems with creating the phages in a lab or factory, transporting the phages, and finally the administration of the phages to the waterways. Phages can only diffuse through the water, and can't actively swim, so they are dependent on the rate of diffusion and water currents. This will be difficult in marshlands, where the bacteria can "hide" in the grass and crevices created by aquatic life. If the bloom is in a high current area, like in a river or a bay, the water can wash the phages away.

Scientists have not yet fully understood the phage infection mechanism, and research into the artificial engineering of phages is limited, making it challenging to conduct studies in this area [67, 68].
Algae can produce toxins that threaten wildlife, contaminate drinking water, and disrupt local economies dependent on fishing and tourism. In the state of Florida, between the years 1995 and 2000, the restaurant and hotel industry lost an estimated $6.5 million to algae blooms. This accounts for about 25% of the average total monthly sales revenue in the region from June through October, the months that are most commonly affected by red tide[69]. During a red bloom event, hospital diagnoses in the county of Sarasota for pneumonia, gastrointestinal, and respiratory illness increased by 19%, 40% and 54% respectively [70, 71], with a respiratory illness visit costing between $0.5 and $4 million [72].

Chapter 10

Appendix C: Flowchart of User and System Interactions

Figure 10.1 shows how the user can interact with the system, the input and outputs for subsystems, and the systems working with one another. To read the flow chart, start from the top to the bottom. First the user creates a network using the GUI Network Creation Tool. After the graph is finished, the user provides an implementation of the network as an ODE model, using Python. Once finished, the user provides the network file and ODE model to the ODE solver. The solver uses information from the network file to determine the number of agents to create, parameter details (including names, values, and dimensions), and setting values. Then the user interacts with the Visualization Dashboard Tool, for example by clicking on buttons to run simulations, changing parameter values, (un)selecting checkboxes, and zooming in and out of plots, and hovering over plots to show data. Once a user has selected the parameter values, the parameter values are sent to the solver. The solver calculates the time and population values using the provided graph and ODE model and sends the data back to the Visualization Dashboard Tool, which then outputs the visualizations. If the user has run an ultimate analysis, then the user can query the saved data to make their own custom visualizations.

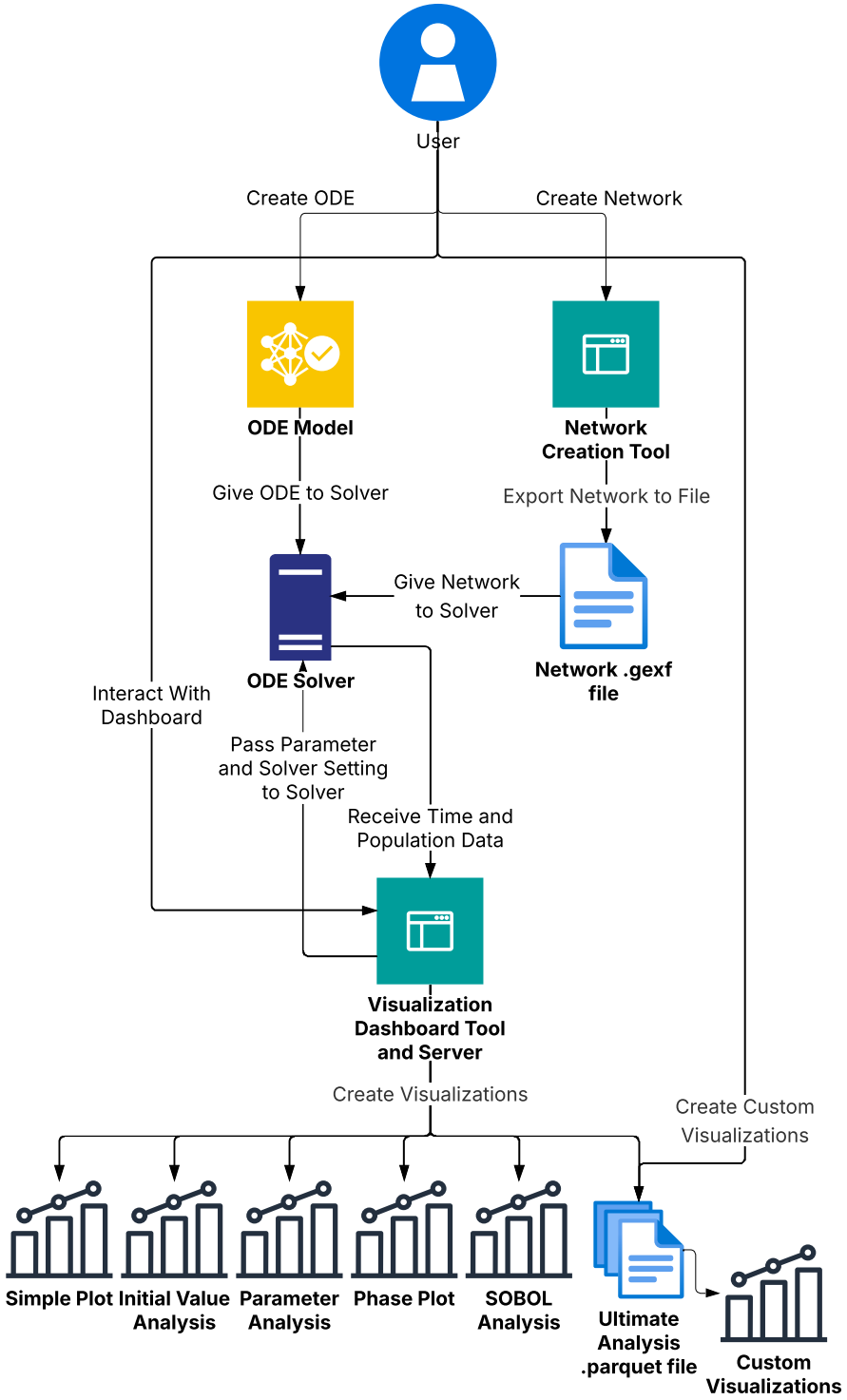


Figure 10.1: The flowchart of user and system interactions. Read from top to bottom.

Chapter 11

Appendix D: ODE Model Implementation

The code listed here is the implementation of Section 3.2.2.

```
import numpy as np
from Classes.Analysis import Analysis
from Classes.GraphMakerGUI import GraphMakerGUI
from Classes.Visualizer import Visualizer

# use base class Analysis to create a new class System
class System(Analysis):
    def __init__(self, graph_location):
        """The ODE representation of the Golden Model from 'Using
    population dynamics to count bacteriophages and their lysogens' from
    Yuncong Geng, Thu Vu Phuc Nguyen, Ehsan Homaee, and Ido Golding.
        Uses Classes.Analysis as a base class.
        Args:
            graph_location (str): location of the graph file
        """
        super().__init__(graph_location)

    def odesystem(self, t, Y, *params):
        """The sytem of ODEs that represent the Golden Model.
        Args:
            t (float): The time value that the solver is currently at.
            Y (np.array): The initial (for t=0)/current (for t>0)
    population values of the system. A 1D array of length equal to the
    number of variables in the system.
        """
        def g(R, v, K):
```

```
23          """Calculate the growth rate of the bacteria. The growth rate
     is a function of the concentration of the resource, the growth rate
    of the bacteria, and the carrying capacity of the bacteria.
24          Args:
25              R (float): Resource concentration
26              v (float): max rate of growth
27              K (float): carrying capacity
28          Returns:
29              float: The growth rate of the bacteria. v*R/(R+K)
30          """
31          return (R * v) / (R + K)
32      # Unpack the parameters. Need to get the graph network, the list
    of phage/bacteria/resource node names, value of M, the vectors (tau
    and washin), and the matrices (e, v, K, r, B), and the environment
    settings
33      graph_object, phage_nodes, bacteria_nodes, resource_nodes, M,
    tau_vector, washin_vector, e_matrix, v_matrix, K_matrix, r_matrix,
    B_matrix, environment = params
34      graph = graph_object.graph
35      Y = self.check_cutoff(Y) # check to see if any values are really
    small. If yes, set to 0
36      R, U, I, P = self.unflatten_initial_matrix(Y, [len(resource_nodes
    ), len(bacteria_nodes), (len(bacteria_nodes), M), len(phage_nodes)]) #
     Turn Y into the shape of the system.
37      new_R = np.zeros_like(R) # create fresh copies of the arrays to
    be updated.
38      new_U = np.zeros_like(U)
39      new_I = np.zeros_like(I)
40      new_P = np.zeros_like(P)
41
42      #update N vector
43      for resource in resource_nodes: # loop over the resource names
44          r_index = resource_nodes.index(resource) # get the index of
    the phage
45          washin = washin_vector[r_index]
46          sum = 0
47          for bacteria in bacteria_nodes: # loop over the bacteria
    names
48              b_index = bacteria_nodes.index(bacteria)
49              if graph.has_edge(bacteria, resource): # important to
    check if the edge between bacteria_b and resource_r exists.
50                  e = e_matrix[b_index, r_index] # get the associated
    values for this interaction
51                  v = v_matrix[b_index, r_index]
52                  K = K_matrix[b_index, r_index]
53                  U_b = U[b_index] # uninfected
54                  I_b = np.sum(I[b_index]) # sum of all infected
    bacteria b agents
```

```
55                  sum += e * g(R[r_index], v, K) * (U_b + I_b)
56          new_R[r_index] = -sum - R[r_index] * environment['washout'] +
     washin # calculate the new value of the resource.
57
58      # update U vector
59      for uninfected_bacteria in bacteria_nodes:
60          u_index = bacteria_nodes.index(uninfected_bacteria)
61          p_sum = 0
62          g_sum = 0
63          for resource in resource_nodes: # loop over the resource
    names
64              r_index = resource_nodes.index(resource) # get index of
    the resource
65              if graph.has_edge(uninfected_bacteria, resource):
66                  g_sum += g(R[r_index], v_matrix[u_index, r_index],
    K_matrix[u_index, r_index])
67          for phage in phage_nodes: # loop over the phage names
68              p_index = phage_nodes.index(phage) # get index of the
    phage
69              if graph.has_edge(phage, uninfected_bacteria): # check if
     the edge between phage_p and bacteria_b exists.
70                  p_sum += r_matrix[p_index, u_index] * P[p_index]
71          # update the uninfected bacteria
72          new_U[u_index] = U[u_index] * g_sum - U[u_index] * p_sum - U[
    u_index] * environment['washout']
73
74      # update I vector
75      for infected_bacteria in bacteria_nodes:
76          i_index = bacteria_nodes.index(infected_bacteria)
77          for k_index in range(0, M):
78              if k_index == 0: # if we are at the first stage of
    infection
79                  p_sum = 0
80                  for phage in phage_nodes: # loop over the phage names
81                      p_index = phage_nodes.index(phage) # get index of
     the phage
82                      if graph.has_edge(phage, infected_bacteria): #
    check if the edge between phage_p and bacteria_b exists.
83                          p_sum += r_matrix[p_index, i_index] * P[
    p_index]
84                  M_tau = 0 if tau_vector[i_index] == 0 else M /
    tau_vector[i_index]
85                  new_I[i_index, 0] = U[i_index] * p_sum - M_tau * I[
    i_index, 0] - environment['washout'] * U[i_index]
86              else: # if we are at the other stages of infection
87                  if(tau_vector[i_index] == 0): # prevent divide by 0
    error
88                      M_tau = 0
```

```
89                      else:
90                          m_tau = M / tau_vector[i_index] # get the value
    of M_tau
91                      right = I[i_index, k_index - 1] - I[i_index, k_index]
92                      new_I[i_index, k_index] = m_tau * right - environment
    ['washout'] * new_I[i_index, k_index]
93
94          # update P vector
95          for phage in phage_nodes: # loop over the phage names
96              p_index = phage_nodes.index(phage) # get index of the phage
97              left_sum = 0 # initialize sums
98              right_sum = 0
99              for infected_bacteria in bacteria_nodes: # loop over the
    bacteria names
100                 i_index = bacteria_nodes.index(infected_bacteria) # get
    index of the bacteria
101                 if graph.has_edge(phage, infected_bacteria): # check if
    the edge between phage_p and bacteria_b exists.
102                     if (tau_vector[i_index] == 0): # prevent divide by 0
    error
103                         M_tau = 0
104                     else:
105                         M_tau = M / tau_vector[i_index] # get the value
    of M_tau
106                     left_sum += B_matrix[p_index, i_index] * M_tau * I[
    i_index, -1]
107                     right_sum += r_matrix[p_index, i_index] * (U[i_index]
     + np.sum(I[i_index]))
108             # update the phage value
109             new_P[p_index] = left_sum - right_sum * P[p_index] -
    environment['washout'] * P[p_index]
110         # flatten the new updated initial conditions, undoes the
    flattening done by unflatten_initial_matrix().
111         flattened_y1 = self.flatten_lists_and_matrices(new_R, new_U,
    new_I, new_P)
112         return flattened_y1
113
114 # graph = GraphMakerGUI(seed=0) # create a new object using the GUI tool.
115 # system = System('simple_graph.gexf') # load the graph from the file.
116 system = System('a_good_curve.gexf') # load the graph from the file.
117 # system = System('a_good_curve_2.gexf') # load the graph from the file.
118 # system = System('complex_graph.gexf') # load the graph from the file.
119
120 phage_nodes = system.get_nodes_of_type('P') # get the phage nodes
121 bacteria_nodes = system.get_nodes_of_type('B') # get the bacteria nodes
122 resource_nodes = system.get_nodes_of_type('R') # get the resource nodes
123 environment_nodes = system.get_nodes_of_type('E') # get the environment
    nodes
```

```
124
125 # get the 'Initial_Condition' attribute values from the nodes. Saves as
        vector.
126 R0 = system.initialize_new_parameter_from_node("Initial_Concentration",
        resource_nodes)
127 U0 = system.initialize_new_parameter_from_node("Initial_Population",
        bacteria_nodes)
128 I0 = system.initialize_new_matrix(len(U0), system.M)
129 P0 = system.initialize_new_parameter_from_node("Initial_Population",
        phage_nodes)
130 # get the 'tau' and 'washin' values from the bacteria and resource nodes.
         Saves as vector
131 tau_vector = system.initialize_new_parameter_from_node('tau',
        bacteria_nodes)
132 washin = system.initialize_new_parameter_from_node('washin',
        resource_nodes)
133
134 # get the 'e', 'v', 'K', 'r', and 'B' values from the edges between the
        listed nodes. Saves as matrix.
135 e_matrix = system.initialize_new_parameter_from_edges('e', bacteria_nodes
        , resource_nodes)
136 v_matrix = system.initialize_new_parameter_from_edges('v', bacteria_nodes
        , resource_nodes)
137 K_matrix = system.initialize_new_parameter_from_edges('K', bacteria_nodes
        , resource_nodes)
138 r_matrix = system.initialize_new_parameter_from_edges('r', phage_nodes,
        bacteria_nodes)
139 B_matrix = system.initialize_new_parameter_from_edges('Burst_Size',
        phage_nodes, bacteria_nodes)
140
141 visualizer = Visualizer(system) # start the visualizer system.
142
143 # add the initial conditions to the visualizer, with a name, the initial
        conditions, and the node names.
144 visualizer.add_graph_data("Resources", R0, resource_nodes)
145 # create an uninfected 'hidden' agent
146 visualizer.add_graph_data("Uninfected Bacteria", U0, bacteria_nodes)
147 # create infected 'hidden' agent. provide row names, as well as column
        names.
148 visualizer.add_graph_data("Infected Bacteria", I0, row_names=
        bacteria_nodes, column_names=[f"Infected B{i}" for i in range(int(4))
        ], add_rows=4)
149 visualizer.add_graph_data("Phages", P0, phage_nodes)
150
151 # add the vector parameters to the visualizer, with a name, the parameter
         values, and the node names.
152 visualizer.add_non_graph_data_vector("tau_vector", tau_vector,
        bacteria_nodes)
```

```
153 visualizer.add_non_graph_data_vector("washin", washin, resource_nodes)
154
155 # add matrix parameters to the visualizer, with a name, the parameter
        values, and the node names.
156 visualizer.add_non_graph_data_matrix("e_matrix", e_matrix, bacteria_nodes
        , resource_nodes)
157 visualizer.add_non_graph_data_matrix("v_matrix", v_matrix, bacteria_nodes
        , resource_nodes)
158 visualizer.add_non_graph_data_matrix("K_matrix", K_matrix, bacteria_nodes
        , resource_nodes)
159 visualizer.add_non_graph_data_matrix("r_matrix", r_matrix, phage_nodes,
        bacteria_nodes)
160 visualizer.add_non_graph_data_matrix("B_matrix", B_matrix, phage_nodes,
        bacteria_nodes)
161
162 # optionally add other parameters to the visualizer, that will be passed
        to the ODE system.
163 visualizer.add_other_parameters(phage_nodes, bacteria_nodes,
        resource_nodes, int(system.M))
164
165 visualizer.run() # run the visualizer/dashboard
```

Chapter 12

Appendix E: Parameter Values Used

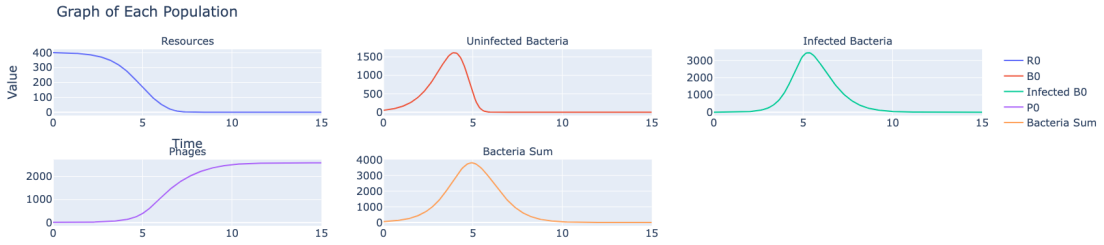
12.1 A Good Curve

IC **Resources**	**Uninfected Bacteria**			**Infected Bacteria**	**Phages**
400	50			(0, 0, 0, 0)	10
Vector Data $\boldsymbol{\tau}$	$\boldsymbol{\omega^i}$				
2.14	0				
Matrix Data $\boldsymbol{e}$	$\boldsymbol{v}$	$\boldsymbol{K}$	$\boldsymbol{r}$	$\boldsymbol{\beta}$	
0.03	1.2	10	0.01	20	
Environment Data $\boldsymbol{M}$		$\boldsymbol{\omega^o}$			
4		0			

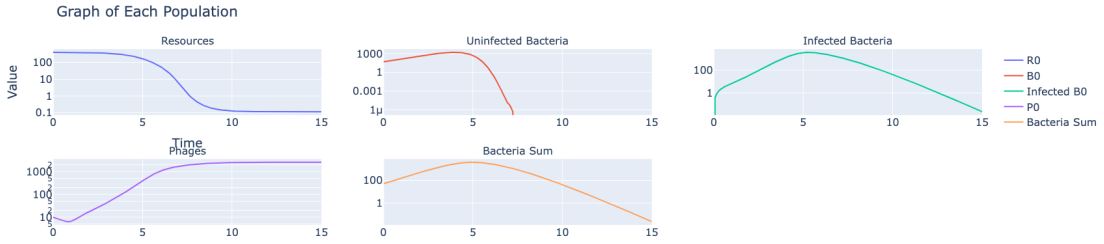
Table 12.1: The parameter values used for Figure 4.1.

12.2 A Good Curve 2

12.3 SOBOL Analysis



(a) Linear y-axis for a "good" plot.



(b) Logarithmic y-axis for a "good" plot.

Figure 12.1: The log plot allows to see behavior happening at values near 0 and to plot data on a logarithmic scale. The parameters used for this plot can be found in Table 12.1.

IC **Resources**	**Uninfected Bacteria**	**Infected Bacteria**	**Phages**
200	50	(0, 0, 0, 0)	10

Vector Data $\boldsymbol{\tau}$	$\boldsymbol{\omega^i}$
0.7	0

Matrix Data $\boldsymbol{e}$	$\boldsymbol{v}$	$\boldsymbol{K}$	$\boldsymbol{r}$	$\boldsymbol{\beta}$
0.12	1	10	0.001	10

Environment Data $\boldsymbol{M}$	$\boldsymbol{\omega^o}$
4	0

Table 12.2: Another set of "good" curves. The linear and logarithmic plot of this data can be seen in Figure 12.1

IC				
Resources	**Uninfected Bacteria**		**Phages**	
100-500	1-100		1-50	
Vector Data				
$\boldsymbol{\tau}$	$\boldsymbol{\omega^i}$			
0.5-3.5	0-100			
Matrix Data				
$\boldsymbol{e}$	$\boldsymbol{v}$	$\boldsymbol{K}$	$\boldsymbol{r}$	$\boldsymbol{\beta}$
0.05-0.25	0.8-1.9	10-250	0.001-0.2	0-100
Environment Data				
$\boldsymbol{\omega^o}$				
0-0.1				

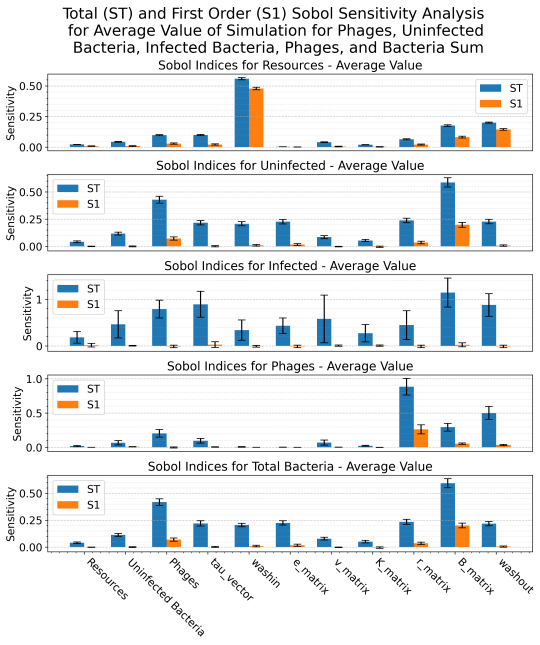
Other Data				
Seed Value	**2nd Order**	**Number Samples**	**Simulations Run**	**Simulation Lenght**
0	False	15	$2^{15}(11+2) = 425984$	15

Table 12.3: The parameter values used for the SOBOL sensitivity analysis in Figure 4.2, Figure 13.1 and Figure 13.2.

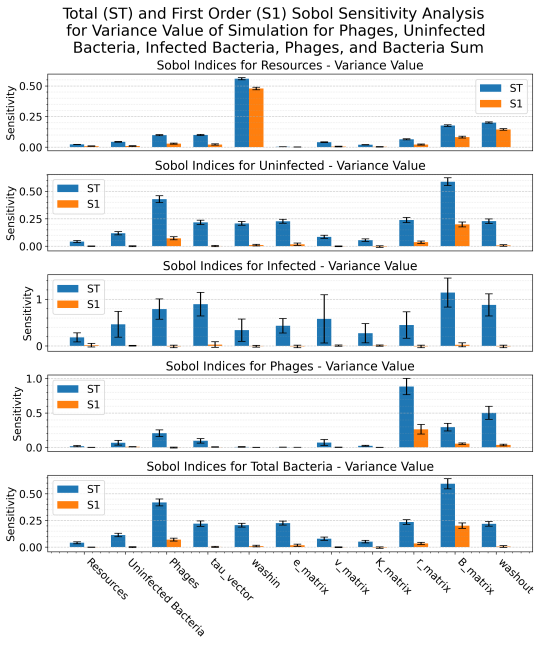
Chapter 13

Appendix F: Extra Plots and Figures

13.1 Extra SOBOL Analyses



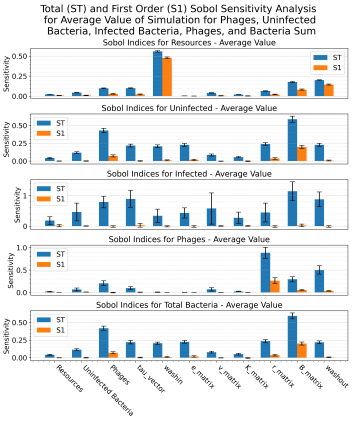
(a) The ST and $S1$ sensitivity for the average Resource, Uninfected, Infected, Phage, and Total Bacteria population.



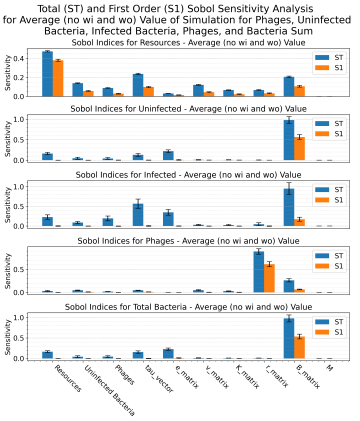
(b) The ST and $S1$ sensitivity for the variance of the Resource, Uninfected, Infected, Phage, and Total Bacteria population.

Figure 13.1: SOBOL analyses for the average, peak, and peak time. The data was saved from the dashboard and plotted using Matplotlib. The values used for this SOBOL test can be found in Table 12.3. The same data used in Figure 4.2 was used for Figure 13.1a and Figure 13.1b.

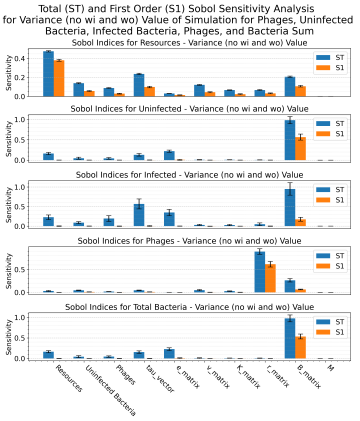
13.1.1 SOBOL Analysis Without Washin and Washout



(a) Final value, no washin and washout.

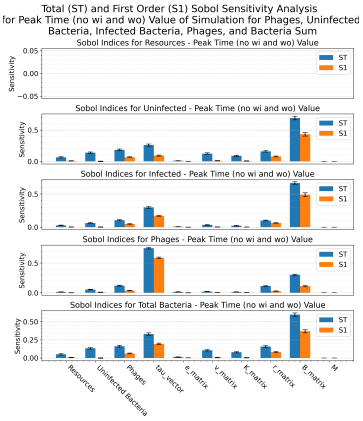


(b) Average value, no washin and washout.



(c) Variance of population value, no washin and washout.

(d) Peak population value, no washin and washout.



(e) Time of peak value, no washin and washout.

Figure 13.2: SOBOL analyses for the final, average, variance, peak, and peak time, without a washin and washout rate. The data was saved from the dashboard and plotted using Matplotlib. The values used for this SOBOL test can be found in Table 12.3, except washin and washout have been set to 0.

13.2 Why 95%?

The 95% rule helps in the IVA analysis. Due to the solver, when taking the absolute peak value, the same time value can occur. Or in an ever increasing value like phages, the peak values occur at the last time step of the simulation, or plateaus and doesn't grow anymore. However, as the parameter value is changing, each graph for every input change will change the growth rate of the agent, changing how fast the agent population grows.

Figure 13.3 shows how using the 95% rule vs the 100% rule for finding the max value reached helps smooth out computational errors from the ODE solver and smooths out the

shape. For the phages, using the 100% rule (Figure 13.3b) shows that the population peaked at the end of the simulation, $t = 15$, for all e values. However at $t = 15$, the population plateaued, as evident by the line graph. Plotting the same plot, but calculating the peak at 95% of the actual peak (Figure 13.3a) shows that the green line ($e = 0.25$) "reached" its peak at $t = 8.4$ before the red line ($e = 0.05$) at $t = 9.4$, a full unit of time after $e = 0.25$. The user can thus conclude that for this instance, larger e values will cause the phage population to reach its "peak" faster than smaller e values.

Figure 13.3c and Figure 13.3d likewise show how the 95% rule can improve analysis of the change in peak time. Figure 13.3d shows how apparently the peak is reached at set time values. Due to how *solve_ivp()* from SciPy works, it automatically chooses time values that it thinks would best capture the dynamics of the system without calculating too many steps. The user can control the step size by decreasing the absolute and relative error bounds, as well as by minimizing the time steps. The user can also provide their own time range with the number of steps to run, increasing the control of the time values chosen. It takes about 0.02321 seconds to run a simulation for 15 time units, where 200 time steps are selected and solved by the solver. Comparatively, a simulation with 1000 time units and 1000000 (a 5000x increase in samples) equidistant time samples takes about 1.71651 seconds to run, a 73.95562258x increase in time spent computing the simulation. The total time taken to run the whole method call, a call to the simple graph maker at the top of the dashboard took 1.76130 seconds vs 17.70634 seconds.

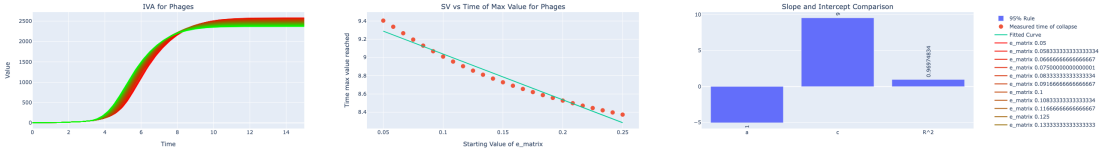
Alternatively instead of controlling the solver, the user can use the 95% rule. Although some accuracy is lost. Going from the 100% rule to the 95% rule, the solver still captures the peak values and the dynamics, but the accuracy is lost. The 100% rule shows that for $e = 0.25$, the time the uninfected population reached its peak occurred at $t = 3.2$. But for the 95% rule, the time at which the peak occurred at is at $t = 3.05$. The slope (the a value) and the intercept (the c value) are somewhat similar, with very high and similar R^2 values (0.97), suggesting a good linear fit of the data.

Figure 13.3e shows how the by increasing the time sampling to more fine grained results in a more accurate graph. Instead of having the solver choose the time values to test, 1000 equidistant time values were selected between 0 and 15. The solver can more accurately calculate the population values and calculate the proper peak time. Comparing the 100% rule without the custom time values with the 100% rule with the custom time values shows the same time values were calculated. in both, the $e = 0.25$ resulted in a time of peak at 3.2 and for $e = 0.05$, the time of peak occurred at $t = 3.95$. This is in stark comparison to the 95% rule vs 100% rule without the custom time, showing a difference of 0.15 time units. The custom time values also preserved the shape of the

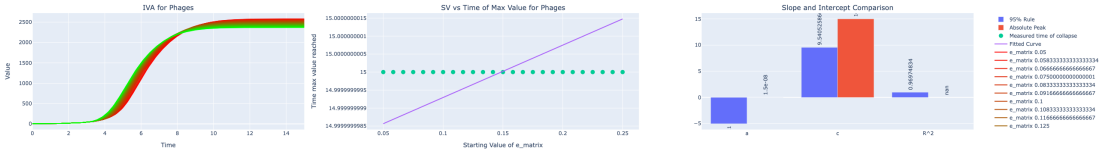
curve e-value vs time curve, being almost identical to that of the 95% rule as seen in Figure 13.3c and Figure 13.3e.

Another issue that arises with the custom time is that it doesn't solve the issue seen with the phages, where the time of peak is at $t = 15$.

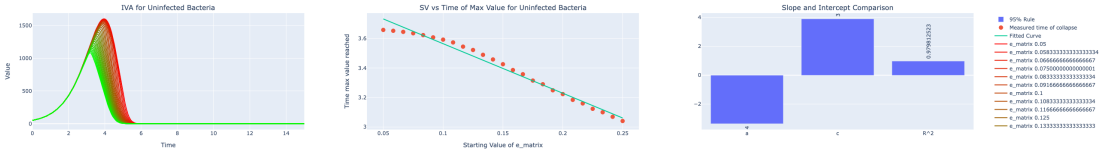
The user can control the % rule with a value input on the dashboard. They can select to use the 95% rule, or 100% rule, or even 83% rule if they want by changing the value they use. The user can use their own custom time values, to ensure that they get high quality curves.



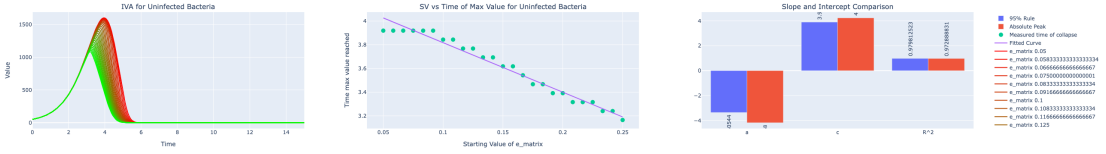
(a) IVA for phage population, 95% rule



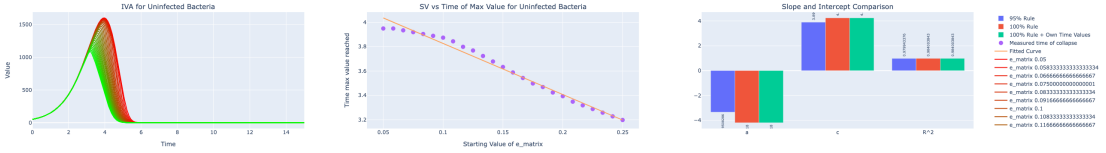
(b) IVA for phage population, 100% rule.



(c) IVA for uninfected population, 95% rule.



(d) IVA for uninfected population, 100% rule.



(e) IVA for uninfected population, 100% rule, 1000 equidistant time steps.

Figure 13.3: Testing the 95% rule vs the 100% rule, where the time at the absolute peak is taken and plotted in the second plot. A comparison of phages and uninfected bacteria is shown. Verification of the graph shape between the 95% rule graph and a frequent time step with 100% rule can be seen between c) and e). The e value is changed, ranging from 0.05 to 0.25.

Bibliography

[1] Allan Campbell. The future of bacteriophage biology. *Nature Reviews Genetics*, 4 (6):471–477, June 2003. ISSN 1471-0056, 1471-0064. doi: 10.1038/nrg1089. URL https://www.nature.com/articles/nrg1089.

[2] Naomi Iris van den Berg, Daniel Machado, Sophia Santos, Isabel Rocha, Jeremy Chacón, William Harcombe, Sara Mitri, and Kiran R. Patil. Ecological modelling approaches for predicting emergent properties in microbial communities. *Nature Ecology & Evolution*, 6(7):855–865, July 2022. ISSN 2397-334X. doi: 10.1038/s41559-022-01746-7.

[3] Sanju Tamang. Horizontal Gene Transfer in Prokaryotes and Eukaryotes, September 2023. URL https://microbenotes.com/horizontal-gene-transfer-prokaryotes-eukaryotes/.

[4] Anders S. Nilsson. Cocktail, a Computer Program for Modelling Bacteriophage Infection Kinetics. *Viruses*, 14(11):2483, November 2022. ISSN 1999-4915. doi: 10.3390/v14112483. URL https://www.mdpi.com/1999-4915/14/11/2483.

[5] Konrad Krysiak-Baltyn, Gregory J. O. Martin, Anthony D. Stickland, Peter J. Scales, and Sally L. Gras. Simulation of phage dynamics in multi-reactor models of complex wastewater treatment systems. *Biochemical Engineering Journal*, 122: 91–102, June 2017. ISSN 1369-703X. doi: 10.1016/j.bej.2016.10.011. URL https://www.sciencedirect.com/science/article/pii/S1369703X16302728.

[6] Nitzan Soffer, Tamar Abuladze, Joelle Woolston, Manrong Li, Leigh Farris Hanna, Serena Heyse, Duane Charbonneau, and Alexander Sulakvelidze. Bacteriophages safely reduce Salmonella contamination in pet food and raw pet food ingredients. *Bacteriophage*, 6(3):e1220347, July 2016. ISSN null. doi: 10.1080/21597081.2016.1220347. URL https://doi.org/10.1080/21597081.2016.1220347.

[7] Xuan Zhang, Yan Dong Niu, Yuchen Nan, Kim Stanford, Rick Holley, Tim McAllister, and Claudia Narváez-Bravo. SalmoFresh™ effectiveness in controlling Salmonella on romaine lettuce, mung bean sprouts and seeds. *International*

Journal of Food Microbiology, 305:108250, September 2019. ISSN 0168-1605. doi: 10.1016/j.ijfoodmicro.2019.108250. URL https://www.sciencedirect.com/science/article/pii/S0168160519301709.

[8] Weizhen Zhang, Jing Liu, Yunxing Xiao, Yumiao Zhang, Yangjinzhi Yu, Zheng Zheng, Yafeng Liu, and Qi Li. The Impact of Cyanobacteria Blooms on the Aquatic Environment and Human Health. *Toxins*, 14(10):658, September 2022. ISSN 2072-6651. doi: 10.3390/toxins14100658. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9611879/.

[9] Stephen Odonkor and Kennedy Addo. Bacteria Resistance to Antibiotics: Recent Trends and Challenges. *International Journal of Biological & Medical Research*, pages 1204–1210, January 2011.

[10] Angelina Volkova, Kelly Ruggles, Anjelique Schulfer, Zhan Gao, Stephen D. Ginsberg, and Martin J. Blaser. Effects of early-life penicillin exposure on the gut microbiome and frontal cortex and amygdala gene expression. *iScience*, 24(7): 102797, July 2021. ISSN 2589-0042. doi: 10.1016/j.isci.2021.102797. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8324854/.

[11] Nana Nguefang Laure and Juhee Ahn. Phage resistance-mediated trade-offs with antibiotic resistance in Salmonella Typhimurium. *Microbial Pathogenesis*, 171: 105732, October 2022. ISSN 08824010. doi: 10.1016/j.micpath.2022.105732. URL https://linkinghub.elsevier.com/retrieve/pii/S088240102200345X.

[12] Yuanyang Zhao, Mei Shu, Ling Zhang, Chan Zhong, Ningbo Liao, and Guoping Wu. Phage-driven coevolution reveals trade-off between antibiotic and phage resistance in Salmonella anatum. *ISME Communications*, 4(1):ycae039, January 2024. ISSN 2730-6151. doi: 10.1093/ismeco/ycae039. URL https://doi.org/10.1093/ismeco/ycae039.

[13] Beata Kowalska. Fresh vegetables and fruit as a source of Salmonella bacteria. *Annals of agricultural and environmental medicine: AAEM*, 30(1):9–14, March 2023. ISSN 1898-2263. doi: 10.26444/aaem/156765.

[14] Sebastián Coloma, Ursula Gaedke, Kaarina Sivonen, and Teppo Hiltunen. Frequency of virus-resistant hosts determines experimental community dynamics. *Ecology*, 100(1):e02554, 2019. ISSN 1939-9170. doi: 10.1002/ecy.2554. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/ecy.2554.

[15] Stephen Tucker and Peter Pollard. Identification of Cyanophage Ma-LBP and Infection of the Cyanobacterium Microcystis aeruginosa from an Australian Subtropical

Lake by the Virus. *Applied and Environmental Microbiology*, 71(2):629–635, February 2005. doi: 10.1128/AEM.71.2.629-635.2005. URL https://journals.asm.org/doi/10.1128/aem.71.2.629-635.2005.

[16] Zongcheng Li. Exploring complicated behaviors of a delay differential equation. *Mathematical Modelling and Control*, 3(mmc-03-01-001):1–6, 2023. ISSN 2767-8946. doi: 10.3934/mmc.2023001. URL http://www.aimspress.com/article/doi/10.3934/mmc.2023001.

[17] Yuncong Geng, Thu Vu Phuc Nguyen, Ehsan Homaee, and Ido Golding. Using bacterial population dynamics to count phages and their lysogens. *Nature Communications*, 15(1):7814, September 2024. ISSN 2041-1723. doi: 10.1038/s41467-024-51913-6. URL https://www.nature.com/articles/s41467-024-51913-6.

[18] Meyers Robert A. Encyclopedia of Physical Science and Technology. URL http://www.sciencedirect.com:5070/referencework/9780122274107/encyclopedia-of-physical-science-and-technology.

[19] Edel Stone, Katrina Campbell, Irene Grant, and Olivia McAuliffe. Understanding and Exploiting Phage–Host Interactions. *Viruses*, 11(6):567, June 2019. ISSN 1999-4915. doi: 10.3390/v11060567. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6630733/.

[20] Dalton V. Banh, Cameron G. Roberts, Adrian Morales-Amador, Brandon A. Berryhill, Waqas Chaudhry, Bruce R. Levin, Sean F. Brady, and Luciano A. Marraffini. Bacterial cGAS senses a viral RNA to initiate immunity. *Nature*, 623(7989):1001–1008, November 2023. ISSN 1476-4687. doi: 10.1038/s41586-023-06743-9. URL https://www.nature.com/articles/s41586-023-06743-9.

[21] Joanna Warwick-Dugdale, Holger H. Buchholz, Michael J. Allen, and Ben Temperton. Host-hijacking and planktonic piracy: How phages command the microbial high seas. *Virology Journal*, 16(1):1–13, December 2019. ISSN 1743-422X. doi: 10.1186/s12985-019-1120-1. URL https://virologyj.biomedcentral.com/articles/10.1186/s12985-019-1120-1.

[22] Asaf Levy, Moran G. Goren, Ido Yosef, Oren Auster, Miriam Manor, Gil Amitai, Rotem Edgar, Udi Qimron, and Rotem Sorek. CRISPR adaptation biases explain preference for acquisition of foreign DNA. *Nature*, 520(7548):505–510, April 2015. ISSN 1476-4687. doi: 10.1038/nature14302. URL https://www.nature.com/articles/nature14302.

[23] P. M. Sharp. Molecular evolution of bacteriophages: Evidence of selection against the recognition sites of host restriction enzymes. *Molecular Biology and Evolution*, 3(1):75–83, January 1986. ISSN 0737-4038. doi: 10.1093/oxfordjournals.molbev.a040377.

[24] Matthew K Waldor and David I Friedman. Phage regulatory circuits and virulence gene expression. *Current Opinion in Microbiology*, 8(4):459–465, August 2005. ISSN 1369-5274. doi: 10.1016/j.mib.2005.06.001. URL `https://www.sciencedirect.com/science/article/pii/S1369527405000755`.

[25] Louis-Charles Fortier and Ognjen Sekulovic. Importance of prophages to evolution and virulence of bacterial pathogens. *Virulence*, 4(5):354–365, July 2013. ISSN 2150-5594. doi: 10.4161/viru.24498. URL `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3714127/`.

[26] Anastasia A. Aksyuk and Michael G. Rossmann. Bacteriophage Assembly. *Viruses*, 3(3):172–203, February 2011. ISSN 1999-4915. doi: 10.3390/v3030172. URL `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3185693/`.

[27] Claudia Igler. Phenotypic flux: The role of physiology in explaining the conundrum of bacterial persistence amid phage attack. *Virus Evolution*, 8(2):veac086, July 2022. ISSN 2057-1577. doi: 10.1093/ve/veac086. URL `https://doi.org/10.1093/ve/veac086`.

[28] Richard E Lenski. TWO-STEP RESISTANCE BY ESCHERICHIA COLI B TO BACTERIOPHAGE T2. *Genetics*, 107(1):1–7, May 1984. ISSN 1943-2631. doi: 10.1093/genetics/107.1.1. URL `https://doi.org/10.1093/genetics/107.1.1`.

[29] Inês Chen, Peter J. Christie, and David Dubnau. The Ins and Outs of DNA Transfer in Bacteria. *Science (New York, N.Y.)*, 310(5753):1456–1460, December 2005. ISSN 0036-8075. doi: 10.1126/science.1114021.

[30] Laura M. Kasman and La Donna Porter. Bacteriophages. In *StatPearls*. StatPearls Publishing, Treasure Island (FL), 2025.

[31] Demeng Tan, Sine Lo Svenningsen, and Mathias Middelboe. Quorum Sensing Determines the Choice of Antiphage Defense Strategy in Vibrio anguillarum. *mBio*, 6(3):10.1128/mbio.00627–15, June 2015. doi: 10.1128/mbio.00627-15. URL `https://journals.asm.org/doi/10.1128/mbio.00627-15`.

[32] Avinoam Rabinovitch, Ira Aviram, and Arieh Zaritsky. Bacterial debris—an ecological mechanism for coexistence of bacteria and their viruses. *Journal of Theoretical Biology*, 224(3):377–383, October 2003. ISSN 0022-5193. doi: 10.

1016/S0022-5193(03)00174-7. URL https://www.sciencedirect.com/science/article/pii/S0022519303001747.

[33] James J. Bull, Kelly A. Christensen, Carly Scott, Benjamin R. Jack, Cameron J. Crandall, and Stephen M. Krone. Phage-Bacterial Dynamics with Spatial Structure: Self Organization around Phage Sinks Can Promote Increased Cell Densities. *Antibiotics*, 7(1):8, March 2018. ISSN 2079-6382. doi: 10.3390/antibiotics7010008. URL https://www.mdpi.com/2079-6382/7/1/8.

[34] Anika Gupta, Norma Morella, Dmitry Sutormin, Naisi Li, Karl Gaisser, Alexander Robertson, Yaroslav Ispolatov, Georg Seelig, Neelendu Dey, and Anna Kuchina. Combinatorial phenotypic landscape enables bacterial resistance to phage infection, January 2025.

[35] Stephen T. Abedon. Phage "delay" towards enhancing bacterial escape from biofilms: A more comprehensive way of viewing resistance to bacteriophages. *AIMS Microbiology*, 3(microbiol-03-00186):186–226, 2017. ISSN 2471-1888. doi: 10.3934/microbiol.2017.2.186. URL http://www.aimspress.com/article/doi/10.3934/microbiol.2017.2.186.

[36] Rasmus Skytte Eriksen, Sine L. Svenningsen, Kim Sneppen, and Namiko Mitarai. A growing microcolony can survive and support persistent propagation of virulent phages. *Proceedings of the National Academy of Sciences*, 115(2):337–342, January 2018. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1708954115. URL https://pnas.org/doi/full/10.1073/pnas.1708954115.

[37] Christoph Lohrmann, Christian Holm, and Sujit S. Datta. Influence of bacterial swimming and hydrodynamics on attachment of phages. *Soft Matter*, 20(24):4795–4805, June 2024. ISSN 1744-6848. doi: 10.1039/D4SM00060A. URL https://pubs.rsc.org/en/content/articlelanding/2024/sm/d4sm00060a.

[38] S. Moineau. Bacteriophage. In Stanley Maloy and Kelly Hughes, editors, *Brenner's Encyclopedia of Genetics (Second Edition)*, pages 280–283. Academic Press, San Diego, January 2013. ISBN 978-0-08-096156-9. doi: 10.1016/B978-0-12-374984-0.00131-5. URL https://www.sciencedirect.com/science/article/pii/B9780123749840001315.

[39] Avigail Stokar-Avihail, Taya Fedorenko, Jens Hör, Jeremy Garb, Azita Leavitt, Adi Millman, Gabriela Shulman, Nicole Wojtania, Sarah Melamed, Gil Amitai, and Rotem Sorek. Discovery of phage determinants that confer sensitivity to bacterial immune systems. *Cell*, 186(9):1863–1876.e16, April 2023. ISSN 0092-8674. doi: 10.1016/j.cell.2023.02.029.

[40] J. J. Bull. Optimality models of phage life history and parallels in disease evolution. *Journal of Theoretical Biology*, 241(4):928–938, August 2006. ISSN 0022-5193. doi: 10.1016/j.jtbi.2006.01.027. URL https://www.sciencedirect.com/science/article/pii/S0022519306000415.

[41] Pramalkumar H. Patel, Véronique L. Taylor, Chi Zhang, Landon J. Getz, Alexa D. Fitzpatrick, Alan R. Davidson, and Karen L. Maxwell. Anti-phage defence through inhibition of virion assembly. *Nature Communications*, 15(1):1644, February 2024. ISSN 2041-1723. doi: 10.1038/s41467-024-45892-x. URL https://www.nature.com/articles/s41467-024-45892-x.

[42] Michael J. Bucher and Daniel M. Czyż. Phage against the Machine: The SIE-ence of Superinfection Exclusion. *Viruses*, 16(9):1348, August 2024. ISSN 1999-4915. doi: 10.3390/v16091348.

[43] Shuji Kanamaru, Kazuya Uchida, Mai Nemoto, Alec Fraser, Fumio Arisaka, and Petr G. Leiman. Structure and Function of the T4 Spackle Protein Gp61.3. *Viruses*, 12(10):1070, September 2020. ISSN 1999-4915. doi: 10.3390/v12101070. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7650644/.

[44] Justin C. Leavitt, Brianna M. Woodbury, Eddie B. Gilcrease, Charles M. Bridges, Carolyn M. Teschke, and Sherwood R. Casjens. Bacteriophage P22 SieA-mediated superinfection exclusion. *mBio*, 15(2):e02169–23, January 2024. doi: 10.1128/mbio.02169-23. URL https://journals.asm.org/doi/10.1128/mbio.02169-23.

[45] Yuval Mulla, Janina Müller, Denny Trimcev, and Tobias Bollenbach. Extreme diversity of phage amplification rates and phage-antibiotic interactions revealed by PHORCE, December 2024.

[46] Portia Mira, Pamela Yeh, and Barry G. Hall. Estimating microbial population data from optical density. *PLOS ONE*, 17(10):e0276040, October 2022. ISSN 1932-6203. doi: 10.1371/journal.pone.0276040.

[47] Advanced Wastewater Modelling | GPS-X - Hydromantis. URL https://www.hydromantis.com/GPSX-innovative.html.

[48] Jacqueline Heard, Emma Harvey, Bruce B. Johnson, John D. Wells, and Michael J. Angove. The effect of filamentous bacteria on foam production and stability. *Colloids and Surfaces. B, Biointerfaces*, 63(1):21–26, May 2008. ISSN 0927-7765. doi: 10.1016/j.colsurfb.2007.10.011.

[49] Brendan J. M. Bohannan and Richard E. Lenski. Effect of Resource Enrichment on a Chemostat Community of Bacteria and Bacteriophage. *Ecology*, 78(8):2303–2315, 1997. ISSN 1939-9170. doi: 10.1890/0012-9658(1997)078[2303:EOREOA]2.0.

CO;2. URL https://onlinelibrary.wiley.com/doi/abs/10.1890/0012-9658%281997%29078%5B2303%3AEOREOA%5D2.0.CO%3B2.

[50] Richard E. Lenski. Dynamics of Interactions between Bacteria and Virulent Bacteriophage. In K. C. Marshall, editor, *Advances in Microbial Ecology*, pages 1–44. Springer US, Boston, MA, 1988. ISBN 978-1-4684-5409-3. doi: 10.1007/978-1-4684-5409-3_1. URL https://doi.org/10.1007/978-1-4684-5409-3_1.

[51] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, and Paul van Mulbregt. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3):261–272, March 2020. ISSN 1548-7105. doi: 10.1038/s41592-019-0686-2.

[52] J. R. Dormand and P. J. Prince. A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26, March 1980. ISSN 0377-0427. doi: 10.1016/0771-050X(80)90013-3.

[53] Dash Documentation & User Guide | Plotly. https://dash.plotly.com/.

[54] Dask Development Team. *Dask: Library for dynamic task scheduling*, 2016. URL http://dask.pydata.org.

[55] Python Software Foundation. Python programming language, version 3.11.6, 2024. URL https://www.python.org. Accessed: 2025-05-19.

[56] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

[57] Takuya Iwanaga, William Usher, and Jonathan Herman. Toward SALib 2.0: Advancing the accessibility and interpretability of global sensitivity analyses. *Socio-Environmental Systems Modelling*, 4:18155–18155, May 2022. ISSN 2663-3027. doi: 10.18174/sesmo.18155.

[58] Jon Herman and Will Usher. SALib: An open-source Python library for Sensitivity Analysis. *Journal of Open Source Software*, 2(9):97, January 2017. ISSN 2475-9066. doi: 10.21105/joss.00097.

[59] Aric Hagberg, Pieter J. Swart, and Daniel A. Schult. Exploring network structure, dynamics, and function using NetworkX. Technical Report LA-UR-08-05495; LA-UR-08-5495, Los Alamos National Laboratory (LANL), Los Alamos, NM (United States), January 2008.

[60] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

[61] Lars Fieseler and Steven Hagens. Food Safety. In David R. Harper, Stephen T. Abedon, Benjamin H. Burrowes, and Malcolm L. McConville, editors, *Bacteriophages: Biology, Technology, Therapy*, pages 857–890. Springer International Publishing, Cham, 2021. ISBN 978-3-319-41986-2. doi: 10.1007/978-3-319-41986-2_29. URL `https://doi.org/10.1007/978-3-319-41986-2_29`.

[62] Heinz G. Floss and Tin-Wein Yu. RifamycinMode of Action, Resistance, and Biosynthesis. *Chemical Reviews*, 105(2):621–632, February 2005. ISSN 0009-2665. doi: 10.1021/cr030112j. URL `https://doi.org/10.1021/cr030112j`.

[63] A. Tomasz. The Mechanism of the Irreversible Antimicrobial Effects of Penicillins: How the Beta-Lactam Antibiotics Kill and Lyse Bacteria. *Annual Review of Microbiology*, 33(Volume 33, 1979):113–137, October 1979. ISSN 0066-4227, 1545-3251. doi: 10.1146/annurev.mi.33.100179.000553. URL `https://www.annualreviews.org/content/journals/10.1146/annurev.mi.33.100179.000553`.

[64] Sergei B. Vakulenko and Shahriar Mobashery. Versatility of Aminoglycosides and Prospects for Their Future. *Clinical Microbiology Reviews*, 16(3):430–450, July 2003. doi: 10.1128/cmr.16.3.430-450.2003. URL `https://journals.asm.org/doi/10.1128/cmr.16.3.430-450.2003`.

[65] Sophie Leclercq, Firoz M. Mian, Andrew M. Stanisz, Laure B. Bindels, Emmanuel Cambier, Hila Ben-Amram, Omry Koren, Paul Forsythe, and John Bienenstock. Low-dose penicillin in early life induces long-term changes in murine gut microbiota, brain cytokines and behavior. *Nature Communications*, 8:15062, April 2017. ISSN

2041-1723. doi: 10.1038/ncomms15062. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5382287/.

[66] Global action plan on antimicrobial resistance. URL https://www.who.int/publications/i/item/9789241509763.

[67] Christopher R. Grasso, Kaytee L. Pokrzywinski, Christopher Waechter, Taylor Rycroft, Yanyan Zhang, Alyssa Aligata, Michael Kramer, and Anisha Lamsal. A Review of Cyanophage–Host Relationships: Highlighting Cyanophages as a Potential Cyanobacteria Control Strategy. *Toxins*, 14(6):385, June 2022. ISSN 2072-6651. doi: 10.3390/toxins14060385. URL https://www.mdpi.com/2072-6651/14/6/385.

[68] Katelyn M. McKindles, Makayla A. Manes, Jonathan R. DeMarco, Andrew McClure, R. Michael McKay, Timothy W. Davis, and George S. Bullerjahn. Dissolved Microcystin Release Coincident with Lysis of a Bloom Dominated by Microcystis spp. in Western Lake Erie Attributed to a Novel Cyanophage. *Applied and Environmental Microbiology*, 86(22):e01397–20, October 2020. doi: 10.1128/AEM.01397-20.

[69] (PDF) Economic Impacts of Red Tide Events on Restaurant Sales. URL https://www.researchgate.net/publication/23515658_Economic_Impacts_of_Red_Tide_Events_on_Restaurant_Sales.

[70] Yung Sung Cheng, Yue Zhou, Clinton M. Irvin, Richard H. Pierce, Jerome Naar, Lorraine C. Backer, Lora E. Fleming, Barbara Kirkpatrick, and Dan G. Baden. Characterization of Marine Aerosol for Assessment of Human Exposure to Brevetoxins. *Environmental Health Perspectives*, 113(5):638–643, May 2005. ISSN 0091-6765. doi: 10.1289/ehp.7496. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1257561/.

[71] Barbara Kirkpatrick, Judy A Bean, Lora E Fleming, Gary Kirkpatrick, Lynne Grief, Kate Nierenberg, Andrew Reich, Sharon Watkins, and Jerome Naar. Gastrointestinal Emergency Room Admissions and Florida Red Tide Blooms. *Harmful algae*, 9(1):82–86, January 2010. ISSN 1568-9883. doi: 10.1016/j.hal.2009.08.005. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2786186/.

[72] Porter Hoagland, Di Jin, Lara Y. Polansky, Barbara Kirkpatrick, Gary Kirkpatrick, Lora E. Fleming, Andrew Reich, Sharon M. Watkins, Steven G. Ullmann, and Lorraine C. Backer. The costs of respiratory illnesses arising from Florida gulf coast Karenia brevis blooms. *Environmental Health Perspectives*, 117(8):1239–1243, August 2009. ISSN 1552-9924. doi: 10.1289/ehp.0900645.