Quantitatively Modelling the Transition from Non-Genetic to Genetic Antimicrobial Resistance

by

Joshua David Guthrie

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Physics
University of Alberta

Abstract

Antimicrobial resistance is a growing global health crisis that requires an interdisciplinary approach to better understand and combat. Physics-based modelling has provided valuable insights into the underlying mechanisms of antimicrobial resistance and the quantitative behaviour of resistant cell populations. Antimicrobial resistance can arise through genetic mutations and from non-genetic mechanisms that arise due to underlying stochastic physical processes which occur during gene expression. Non-genetic resistance results in phenotypic heterogeneity in genetically identical cell populations, which enables a fraction of the population to survive during drug treatment. Although genetic and non-genetic mechanisms have been investigated individually, it remains unclear how non-genetic resistance alters the evolution of antimicrobial resistance. In this thesis, physics-based methods are used to develop and analyze a phenomenological model of cell population dynamics to investigate the evolution of antimicrobial resistance during drug treatment. The model quantifies the transition from reversible non-genetic resistance to permanent genetic resistance and is used to make quantitative predictions about antimicrobial resistance evolution in cell populations. A deterministic framework that utilizes population growth equations is used to investigate regimes where stochastic fluctuations in population dynamics are negligible and standard numerical methods are used to solve these equations. In regimes where stochastic fluctuations play an important role in the population dynamics leading to antimicrobial resistance, the model is reformulated in a stochastic framework as a set of continuous-time stochastic processes and simulated using Monte Carlo methods. The structure and parameter values of the model are guided by experimental studies of non-genetic and genetic antimicrobial resistance in fungi, with parameter scans being used to investigate changes in subpopulation fitness and their effect on the survival and evolution of the population. The research presented in this thesis supports previous experimental and qualitative findings of antimicrobial resistance evolution by showing quantitatively that non-genetic resistance aids the survival of cell populations undergoing drug treatment and increases the probability of an antimicrobial resistance mutation appearing in the population. Additionally, a novel hypothesis about the evolution of antimicrobial resistance in cell populations is suggested, namely that increased survival due to non-genetically resistant cells results in intraspecific competition between subpopulations and hinders the evolution of antimicrobial resistance. The research presented in this thesis has advanced our understanding of antimicrobial resistance by providing a quantitative framework for investigating the transition from non-genetic to genetic resistance.

Preface

The work presented in Chapter 2 was previously published in the journal *Physical Biology* (Joshua D Guthrie and Daniel A Charlebois, "Non-genetic resistance facilitates survival while hindering the evolution of drug resistance due to intraspecific competition", *Physical Biology* 19 066002, 2022) [1]. I presented early findings of the work at the 2022 *APS March Meeting*, with my presentation titled "Phenotypic Heterogeneity Facilitates Survival While Hindering the Evolution of Drug Resistance Due to Intraspecific Competition". In addition to the work presented in this thesis, I co-authored a textbook chapter on applying machine learning to the study of antimicrobial resistance and drug development ("Machine Learning for Antimicrobial Resistance Research and Drug Development", Shamanth A. Shankarnarayan, Joshua D. Guthrie and Daniel A. Charlebois, The Global Antimicrobial Resistance Epidemic - Innovative Approaches and Cutting-Edge Solutions, *IntechOpen*, 2022) [2].

To my parents, Dave and Tammy, for their unconditional support and encouragement.

"Joy in looking and comprehending is nature's most beautiful gift."

-Albert Einstein

Acknowledgements

I would like to thank my supervisor, Dr. Daniel Charlebois, for his teaching, guidance, encouragement, and support over the last few years. I would also like to thank all members of the Charlebois Laboratory for insightful discussions and feedback on my research, especially Harold Flohr for reviewing Section 1.4 of my thesis. I extend my thanks to the members of my supervisory committee, Dr. Daniel Charlebois, Dr. Richard Marchand, and Dr. Michael Woodside, for their helpful feedback on my research. Additionally, I would like to thank Daniel Aldrich for making the *LaTeX* template I used to organize my thesis openly available to University of Alberta graduate students. The University of Alberta Department of Physics and the Charlebois Laboratory provided financial support throughout my program. I was also supported financially in part by an Alberta Graduate Excellence Scholarship Indigenous, and I would like to extend my thanks to the Government of Alberta for their support of Indigenous graduate students through these awards. Lastly, I would like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) for supporting me as an undergraduate through multiple Undergraduate Student Research Awards, enabling me to work on various projects in physics that introduced me to scientific research.

Table of Contents

List of Tables

List of Figures

List of Symbols

CV	Standard Deviation Divided by the Mean (Coefficient of Variation).
G	Genetically Resistant Cell Subpopulation.
N	Non-Genetically Resistant Cell Subpopulation.
P_G	Probability of Genetic Resistance Emerging Before Population Extinction.
$P_{\tau_{fix}}$	Genetic Fixation Probability.
P_τ	Genetic First-Appearance Probability.
S	Drug Susceptible Cell Subpopulation.
T	Total Cell Population $S + N + G$.
δ_i	Death Rate of Subpopulation i.
τ	Genetic First-Appearance Time.
τ_{fix}	Genetic Fixation Time.
k_i'	Scaled Birth Rate $k_i z(T)$.
k_i	Maximum Birth Rate of Subpopulation i.
$r_{i,j}$	Switching Rate from Subpopulation j to i.
t	Time.
$z(T)$	Inhibition Function.

The symbols above are defined as used in Chapter 2.

Abbreviations

AMR Antimicrobial Resistance.

CME Chemical Master Equation.

CV Coefficient of Variation.

DNA Deoxyribonucleic Acid.

GPF Grand Probability Function.

mRNA Messenger RNA.

ODE Ordinary Differential Equation.

RNA Ribonucleic Acid.

SSA Stochastic Simulation Algorithm.

Chapter 1

Introduction

"Everything should be made as simple as possible, but no simpler."

- Albert Einstein

1.1 Antimicrobial Resistance

1.1.1 A Global Health Crisis

Antimicrobial resistance (AMR), defined as a decrease in the sensitivity of microbes to antimicrobial drugs, has resulted in a growing global health crisis [3, 4]. AMR occurs when microbes (such as bacteria, fungi, and viruses) no longer respond to antimicrobial drugs [3, 4], which have played a significant role in the development of modern medicine [3–5]. It has been estimated that AMR will result in more than 10 million deaths per year by 2050 if preventative measures are not developed [4]. Additionally, AMR places a significant burden on healthcare systems and has major economic and social implications [6, 7], with an estimated loss of global capital between $300 billion to $1 trillion by 2050 [7] and a significant increase in poverty, especially in under-developed countries and among marginalized groups [7, 8]. In Canada, healthcare costs related to AMR are currently estimated to exceed $1 billion per year and are projected to reach $120 billion per year by 2050 [8]. The medical, societal, and economic burdens that AMR causes has led to an extensive global research effort to better understand the underlying mechanisms and evolution of AMR, with contributions from many disciplines. Physics has played an important role in understanding AMR, with biophysicists providing a robust understanding of the underlying molecular mechanisms responsible for AMR [9], along with providing quantitative frameworks for investigating AMR phenomenon, such as AMR evolution in cellular populations [10, 11].

1.1.2 Taking a Physics-Based Approach to Study the Evolution of Antimicrobial Resistance

Biophysics research is particularly interested in understanding biological phenomena through the application of physical theories and methodology. The physics methodology of producing mathematical models to predict the behaviour of complex systems that can be tested experimentally has led to a robust, quantitative understanding of many biological mechanisms on a variety of scales, from the molecular level of biological molecules [9, 12, 13] to the macroscopic scale of interacting organisms [11, 14, 15]. Biophysical investigations have provided invaluable insights into the molecular mechanisms of AMR, especially through the application of statistical mechanics and other physical theories to the study of gene expression and genetic networks [9, 16–18], and in understanding the behaviour of larger-scale biological systems through the application of mathematical modelling and computational methods commonly used in physics [10, 11, 19, 20]. The latter is the approach taken for the research presented in this thesis, where we have aimed to better understand the evolutionary dynamics of AMR at the cell population level using predictive modelling and computational methods commonly used in physics.

The evolution of AMR due to genetic mutations within cell populations is a well-understood phenomenon, with a long history of major advancements using both experimental and theoretical frameworks [21–23]. The development of AMR due to non-genetic mechanisms is increasingly understood, although not to the extent of genetic mutations in terms of cell population evolution [10, 20]. While these mechanisms have been studied individually, it remains unclear how non-genetic mechanisms may alter the course of AMR evolution at the cell population level [10, 20]. Understanding AMR evolution in heterogeneous cell populations is crucial for developing strategies for treating drug-resistant infections, as infections occur due to pathogenic microbes that grow alongside and interact with other cells, forming large populations of cells where the collective behaviour of the population can have a significant impact on the severity of infections and the evolution of AMR [24, 25]. Additionally, heterogeneous cell populations exhibit large-scale behaviour that would not be evident by studying AMR at the single-cell level or in homogeneous populations, such as impacting the overall evolution of AMR through competition between susceptible and resistant cells [24–26].

Developing a quantitative framework that can be used to predict the evolution of AMR in cell populations that exhibit genetic and non-genetic AMR is the main goal of the research presented in this thesis, the results of which are presented in Chapter 2. By taking a physics-based approach to studying AMR through the development of quantitative frameworks for investigating the evolution of AMR in heterogeneous cell populations researchers may be able to predict the development of AMR and describe quantitatively how AMR evolves over time [10, 11, 19, 20]. Quantitative models of AMR evolution also have the potential to guide medical treatments of drug-resistant infections and diseases in humans by providing a predictive method that can be used to alter treatment strategies [11, 19, 27], a topic that is discussed further in Chapter 3 of this thesis.

In the remainder of this chapter, I give an overview of gene expression, including stochasticity in gene expression that occurs due to underlying physical processes and which plays a significant role in the development of non-genetic AMR. This is followed by an introduction to mechanisms responsible for cells becoming drug-resistant, which is important for understanding the cell population-level characteristics that arise from them. These mechanisms are then related to biological evolution at the cell population level. Next, deterministic and stochastic models of cell population dynamics, along with the computational methods required to simulate them, are discussed. Chapter 2, which was previously published in the journal *Physical Biology* [1], presents a quantitative model to investigate the evolutionary dynamics of a cell population under the effects of antimicrobial drugs. The model is based on the AMR literature, which was used to tune the overall mathematical structure and parameter values of the model, with a particular focus on yeast due to their extensive use as model organisms [28–30]. Additionally, the mathematical formulation of the model was inspired by compartmental models that have been successfully applied in other fields, such as infectious disease modelling in epidemiology [31]. The model, which incorporates cell population-level characteristics of AMR that arise due to the mechanisms discussed in this chapter, is expressed both as a deterministic set of population growth equations and as a discrete system of stochastic processes. Computational methods are used to simulate the model in each of these frameworks. The resulting simulations are then analyzed to make novel predictions about the effects of non-genetic resistance on the evolution of AMR and to support previous experimental and qualitative findings. Chapter 3 concludes the thesis with a summary of the research, a discussion of the limitations of the model presented in Chapter 2, and my thoughts on future directions for the field.

1.2 Gene Expression

1.2.1 Genetic Information and Gene Expression

At the molecular level, cells are made up of molecules that are responsible for carrying out different biological processes within the cell [32]. The nucleic acids deoxyribonucleic acid (DNA) and ribonucleic acid (RNA) are responsible for storing and encoding information within cells and are of particular importance for the replication and use of genetic information [32, 33]. Genetic information contains the instructions (the genetic code) required to create the biological molecules that carry out functions within cells [32–34]. This genetic code is separated into genes, which are sequences of nucleotides that encode the information required to translate DNA into RNA [32]. RNA can carry out specific processes in the cell or be used as a template for the creation of proteins (in the latter case, the RNA is referred to as messenger RNA or mRNA) [32]. Proteins are complex biological macromolecules that carry out a vast number of essential processes within cells, including (but not limited to) catalyzing biochemical reactions, providing cellular structure, transporting molecules, responding to stimuli, and replicating DNA [32]. Gene expression, which is the process of translating mRNA into specific proteins from genes encoded in DNA (represented by a promoter, the regulatory region of DNA that precedes the gene) through the transcription of DNA into mRNA (also referred to as the "central dogma" of molecular biology [33]) [32]. This process is shown visually in Figure 1.1 as a phenomenological model of gene expression [16, 35].

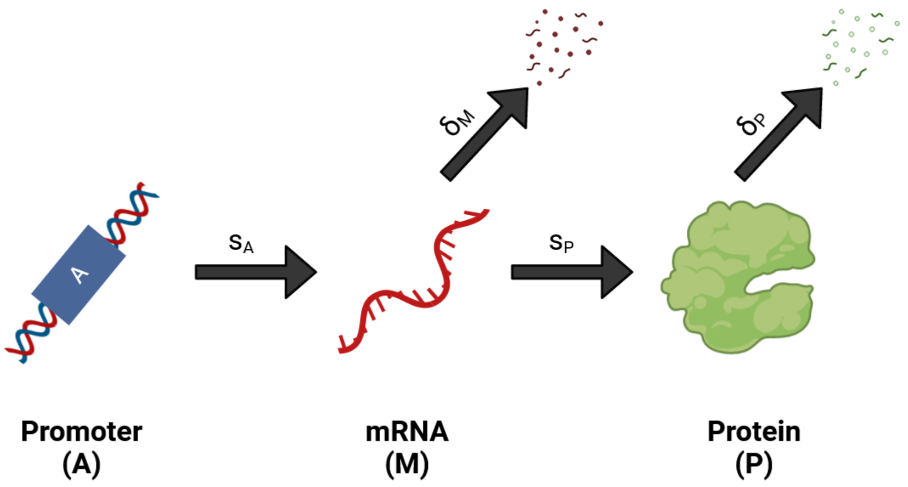

Figure 1.1: A phenomenological model of gene expression [35]. A gene encoded in DNA, represented by its promoter A, is expressed through transcription into mRNA (M) at a rate s_A, which is then translated into a protein (P) at a rate s_P. mRNA and proteins degrade at rates δ_M and δ_P, respectively. Figure inspired by Figure 1 in [16] and Figure 1A in [35].

Cells that contain the same genetic information are said to have the same "genotype”, while cells that possess the same observable physical and biochemical characteristics are said to have the same “phenotype” [36]. Changes to the genetic code, for example through genetic mutations, can result in different phenotypes [37]. Having the same genotype does not necessarily produce the same phenotype - gene expression levels among genetically identical cells can vary due to intrinsic stochastic effects and extrinsic environmental conditions, leading to phenotypic variations among genetically identical cells [16, 38–40]. The intrinsic gene expression “noise” that arises due to the underlying physical processes of gene expression is described further in Section 1.2.4. Figure 1.2 illustrates the relationship between gene expression noise and protein levels, showing how differences in protein levels can lead to phenotypic variations in genetically identical cells [16, 38].

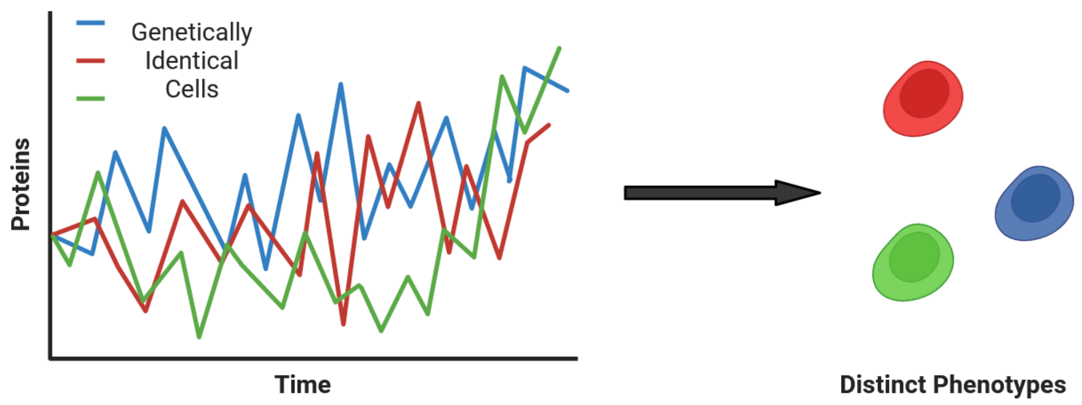

Figure 1.2: Gene expression (Figure 1.1) occurs due to discrete biochemical reactions which are intrinsically stochastic, leading to variations in gene expression among genetically identical cells [16, 38]. Stochastic variations in protein levels due to gene expression noise can result in phenotypic variation among genetically identical cells. Figure inspired by Figure 1 in [38].

For a more in-depth review of the information covered in this section, see standard biology textbooks such as [32]. For similar information placed into the context of physical biology, see textbooks such as [41].

1.2.2 Two-Stage Model of Gene Expression

The process of gene expression illustrated in Figure 1.1 can be modelled as a biochemical reaction system that determines the molecular levels of mRNA and proteins within the system [35]. A common formulation of this model is referred to as the "two-stage" model of gene expression [35]. Although this model simplifies the complex biochemical reactions involved in gene expression [35], it provides a reasonable approximation in some cases and can be useful for investigating important dynamics involved in the expression of genes [16]. The reactions of this system can be described by the following set of reaction equations [42]:

$$A \xrightarrow{s_A} A + M \tag{1.1}$$

$$M \xrightarrow{s_P} M + P \tag{1.2}$$

$$M \xrightarrow{\delta_M} \varnothing \tag{1.3}$$

$$P \xrightarrow{\delta_P} \varnothing \tag{1.4}$$

where A represents the promoter of a given gene, M denotes mRNA, and P denotes proteins [35, 42]. Equations 1.1 and 1.2 represent transcription and translation with reaction rates s_A and s_P, respectively, and Equations 1.3 and 1.4 respectively represent the degradation of M and P with reaction rates δ_M and δ_P [35, 42]. In this model, a single promoter A is considered, while the number of M and P molecules change when reactions occur [35, 42]. These equations are first-order chemical reactions and a deterministic approximation of the system can be obtained by applying the empirical law of mass action [43], resulting in a set of coupled ordinary differential equations [42]:

$$\frac{d[M]}{dt} = s_A - \delta_M[M] \tag{1.5}$$

$$\frac{d[P]}{dt} = s_P[M] - \delta_P[P] \tag{1.6}$$

where $[M]$ and $[P]$ denote the concentrations of mRNA and protein, respectively. The term s_A denotes the rate of mRNA production, which is constant due to only a single promoter (A) being present [35, 42]. $s_P[M]$ defines the rate of protein synthesis, while $\delta_M[M]$ and $\delta_P[P]$ are the rates of mRNA and protein dilution and degradation, respectively [35, 42]. Although these equations can provide an accurate approximation when molecular populations are large and stochastic fluctuations are negligible, they fail to capture the underlying stochastic behaviour of the reaction system when molecular populations are small and stochastic fluctuations may significantly affect the dynamics of the system, which is often the case for gene expression [16]. The deterministic formulation is therefore insufficient to describe gene expression noise and the system must be formulated using stochastic chemical kinetics.

1.2.3 Stochastic Chemical Kinetics

Gene expression relies on biochemical reactions where the number of molecules involved in these reactions varies by discrete integer amounts and these molecules undergo physical interactions with each other and their environment [16, 35, 42, 44]. Biochemical systems are generally considered to be in thermal equilibrium (but not chemical equilibrium), resulting in a random distribution of molecules throughout the volume that contains them [42, 45, 46]. In the formulation of stochastic chemical kinetics, the reaction environment is assumed to contain underlying fluids and is well-stirred, which results in random movement fluctuations of the reacting molecules due to Brownian motion [44, 47]. Brownian motion arises from random non-reactive collisions with other molecules in the system, in particular the fluid molecules which have random motions due to thermal fluctuations and non-reactive collisions with other molecules [44, 47]. Reactant molecules must collide with each other for a chemical reaction to occur and the physical interactions in the system lead to random collisions [45–47]. These random collisions result in stochastic timing of reaction events and, therefore, stochastic molecular levels [45–47]. Stochastic chemical kinetics is formulated by determining the statistical behaviour of these collisions through the application of statistical mechanics to the reaction system and can be used to describe the stochastic nature of the system [45–47]. This is achieved using a finite differential-difference equation that describes the time evolution of the reactant species in the system [44, 45]. For chemical reactions, this equation is referred to as the chemical master equation (CME) and its solution is referred to as the grand probability function (GPF) [42, 44, 45, 47].

A reaction system contained within a volume V can be defined by N reaction species $\{S_1, ..., S_N\}$ and M possible reactions $\{R_1, ..., R_M\}$ which occur with reaction constants $\{c_1, ..., c_M\}$ [47]. The state of the system can be described by a vector of independent integer variables $\boldsymbol{X(t)} = \boldsymbol{x} = [X_1, ..., X_N]$ containing the number of molecules X_i of each species S_i at some time t [47]. Each reaction R_j can change the system, and state change vectors $\boldsymbol{v}_j$ can be defined which contain the change in population numbers of each population S_i for every reaction R_j [47]. Propensity functions a_j for each reaction R_j, where $a_j dt$ describes the probability that a reaction R_j will occur in the infinitesimal time interval dt (the fundamental premise of stochastic chemical kinetics [47]), can be defined as $a_j = c_j h_j$ where h_j is the number of distinct molecular reaction combinations for R_j in the system at time t and c_j is defined as $c_j dt$ being the average probability that a particular combination of reactants of R_j will occur in the infinitesimal time interval dt [45, 47]. The mathematical form of the reaction constants c_j and combinations h_j are defined based on the underlying physics of the system (generally through the application of statistical mechanics [47]) and change based on the order of the reactions that they describe, but for first-order reactions h_j is always equal to the population size of the reactant species of R_j and c_j is equivalent to the empirical reaction rates defined in the deterministic formulation of chemical kinetics described by the law of mass action [45, 47]. For a reaction system defined in this way, the GPF takes the form $p(\boldsymbol{x}, t)$, which is a probability density function of the time-dependent variable $\boldsymbol{x} = \boldsymbol{X}(t)$ and describes the probability that the system will consist of X_1 molecules of S_1, X_2 molecules of S_2, ..., and X_N molecules of S_N at time t in the volume V of the system [45]. The CME can then be defined as [47]

$$\frac{\partial}{\partial t} p(\boldsymbol{x}, t) = \sum_{j=1}^{M} [a_j(\boldsymbol{x} - \boldsymbol{v}_j) p(\boldsymbol{x} - \boldsymbol{v}_j, t) - p(\boldsymbol{x}, t) a_j(\boldsymbol{x})] \tag{1.7}$$

which analytically describes the time evolution of the GPF and the stochastic nature of chemical reaction systems (see [47] for a full derivation of the CME based on the microphysical foundations of chemical kinetics). To write the CME more concisely for particular systems, the step operator $\boldsymbol{E}_{X_i}^k$ can be defined as [42]

$$\boldsymbol{E}_{X_i}^k f(X_i, X_l) = f(X_i + k, X_l) \tag{1.8}$$

which describes the removal or addition of k molecules from species i when a given reaction occurs [42, 48].

The CME can be written as a set of ordinary differential equations in t, but the number of equations required to describe a given reaction system is comparable to the number of combinations of molecules in the system, making analytical solutions of the CME generally impossible in biologically realistic scenarios and the use of standard numerical methods difficult [45–47]. Although approximation methods exist to investigate the CME analytically, such as using the Fokker-Planck equation to approximate the CME [49], Monte Carlo methods are the standard approach for investigating stochastic systems described by the CME [45–47]. Monte Carlo methods for simulating stochastic reaction systems are discussed further in Section 1.4.4.

1.2.4 Gene Expression Noise

With respect to gene expression, noise refers to the stochastic variations of mRNA and protein levels among genetically identical cells in the same environment [16, 42]. These fluctuations arise due to the underlying stochastic biochemical processes that occur during transcription and translation [16, 42]. This in turn can lead to stochastic phenotypic variations among genetically identical cells (or "phenotypic heterogeneity") [16, 38, 50]. A common quantitative measure of noise in gene expression is the coefficient of variation (CV), which is defined as the standard deviation σ divided by the mean and describes the relative deviation from the average of a distribution [16, 42]. Another common measure is the fano factor (or noise strength), defined as the variance σ^2 divided by the mean, which can provide additional information that might be missed when using the CV due to the inverse square root scaling of noise that is often present in stochastic systems (e.g. in "translational bursting", where the amplitude of protein fluctuations depends on the number of proteins produced per mRNA [16]) [16, 42, 51]. Through the application of stochastic chemical kinetics, statistical properties of mRNA and protein levels can be investigated [35].

The two-stage model of gene expression discussed in Section 1.2.2 is relatively simple (as it contains only a small number of populations and reactions) and analytic methods have been used to investigate it [35]. Of particular interest are the moments of the GPF for the system with respect to each molecular species, with the first moment corresponding to the average number of molecules $\langle n_i(t)\rangle$ of species i and the second moment describing the variance $\sigma_i^2(t) = \langle n_i^2\rangle - \langle n_i\rangle^2$ [35, 42]. These moments can be used to quantify the noise of the system in terms of mRNA and protein population levels. The GPF for this system is defined as $p(M, P, t)$ and by applying Equation 1.7 (using the step operator defined in Equation 1.8) to the reactions described by 1.1-1.4, the CME of the system can be written as [35, 42]

$$\begin{aligned}\frac{dp(M,P,t)}{dt} &= s_A(\boldsymbol{E}_M^{-1} - 1)p(M,P,t) + s_P(\boldsymbol{E}_P^{-1} - 1)Mp(M,P,t) \\ &+\delta_M(\boldsymbol{E}_M^{1} - 1)Mp(M,P,t) + \delta_P(\boldsymbol{E}_P^{1} - 1)Pp(M,P,t)\end{aligned} \tag{1.9}$$

which defines the time evolution of the GPF that describes the stochastic mRNA and protein levels [35].

To investigate the statistics of the system when it has reached a steady state (when the rate of change of each population is zero, denoted M^s and P^s), analytical methods have been applied to determine the first and second moments of the GPF with respect to M^s and P^s [35]. The first moment results in the average steady-state population numbers for M^s and P^s, giving [35, 42]

$$\langle M^s \rangle = \frac{s_A}{\delta_M} \tag{1.10}$$

$$\langle P^s \rangle = \frac{s_A s_P}{\delta_M \delta_P} = \langle M^s \rangle \frac{s_P}{\delta_P} \tag{1.11}$$

which are the same values that would be found by setting the deterministic rate equations (Equations 1.5 and 1.6) to zero and solving for M and P, showing that the deterministic rate equations can approximate averages of the GPF [16, 35, 42]. M production and decay is fully described by Poisson statistics and therefore the steady-state variance of M^s is equivalent to its mean (i.e, $(\sigma_M^s)^2 = \langle M^s \rangle$) [35, 42]. The variance of P^s is determined by calculating the second moment of the GPF with respect to P^s and results in [35]

$$(\sigma_P^s)^2 = \langle P^s \rangle (1 + \frac{s_P}{\delta_M}) \tag{1.12}$$

The means and variances with respect to M^s and P^s can be used to quantify the noise of the steady-state system. Defining the steady state CV for each population i as CV_i^s, we have

$$CV_M^s = \frac{\sigma_M^s}{\langle M^s \rangle} = \frac{1}{\sqrt{\langle M^s \rangle}} \tag{1.13}$$

and

$$CV_P^s = \frac{\sigma_P^s}{\langle P^s \rangle} = \sqrt{\frac{1 + \frac{s_P}{\delta_M}}{\langle P^s \rangle}} \tag{1.14}$$

which shows that the steady-state noise of the mRNA and protein distributions fall off to the inverse square of the population sizes, a result that is common for stochastic processes [16]. This implies that noise becomes negligible when molecular reactant population sizes are large (enabling the use of deterministic rate equations) but can be significant when population sizes are small (which is often the case during gene expression) [16, 47]. In the latter case, the use of stochastic chemical kinetics is required to accurately model the dynamics of the system. For models more complicated than the simple two-stage model, Monte Carlo methods are generally required to simulate trajectories of the GPF [45, 47].

The macroscopic phenotypic implications of the molecular level processes that result in gene expression noise have been studied, showing that noise in gene expression can result in phenotypic variances among members of genetically identical cellular populations [16, 38]. Gene expression noise has also been shown to be modulated by gene expression networks, which can change gene expression levels and alter biological functions [10, 17]. This enables the use of synthetic biology methods to investigate the behaviour and resulting biological functions of gene expression noise [10, 11]. Additionally, phenotypic characteristics that arise due to gene expression noise can be heritable (although not as stable as genetic mutations) in some cases [20], in part due to epigenetic mechanisms (heritable changes in gene expression not caused by alterations of the genetic code, e.g. through DNA methylation and chromatin remodeling [52]) [20]. This has important implications for non-genetic mechanisms of AMR, which are discussed further in Section 1.3.3.

Genetic information and gene expression play crucial roles in the development of AMR in all cells [20]. The genetic and non-genetic mechanisms responsible for AMR development, and how they relate to biological evolution at the level of cell populations, are discussed in the next section.

1.3 Antimicrobial Drugs, Mechanisms of Antimicrobial Resistance, and Antimicrobial Resistance Evolution in Cell Populations

1.3.1 Antimicrobial Drugs and Resistance

In the study of AMR, bacteria and fungi are of particular interest as they are responsible for a large number of infections and diseases in humans [3, 4, 53]. Infections and diseases caused by bacteria and fungi are generally treated using antibiotic and antifungal drugs, respectively, which are compounds that either inhibit cell growth (static drug) or kill cells (cidal drug) [54, 55]. Antimicrobial drugs target specific biological mechanisms that are responsible for the growth and survival of cells, which can vary among different species [54–56].

Fungi, especially the genera *Candida*, *Aspergillus* and *Cryptococcus*, are of particular interest to AMR researchers as fungal infections due to certain species, such as *Candida albicans* and *Cryptococcus neoformans*, can become pathogenic in humans, causing serious, life-threatening illnesses [53, 57]. Additionally, fungi species often display resistance to the limited number of currently available antifungal drugs [57], with *Candida auris* infections being resistant to all classes of antifungal drugs in some cases, making it a pathogen of significant importance for global healthcare [58–61]. The seriousness of infections due to *Candida* species and other genera of pathogenic fungi, and the difficulty in treating them due to AMR, has made studying antifungal AMR a highly active area of research [23]. Along with the significant threats that certain yeast species pose to human health, yeast cells, such as *Saccharomyces cerevisiae*, have a long history of being studied as model organisms to investigate the biological functions of eukaryotic cells [28–30], which in turn has led to significant advancements in our understanding of the underlying biological mechanisms of AMR [56, 62]. This has provided researchers with a large amount of information to guide the formulation of mathematical models to further investigate AMR. In the research presented in Chapter 2, the formulation and parameters of the developed mathematical model were guided by previous studies that have utilized model organisms to study cell population-level characteristics of AMR that arise due to underlying physical and biochemical mechanisms.

In evolutionary terms, microbes in a population of cells that exhibit resistance to a drug can be said to have a higher “fitness” in the drug environment [63]. For the research presented in this thesis, we define fitness as the rate of growth of a population in a given environment [63, 64]. Cells that have higher fitness will have a selective advantage relative to the rest of the population, leading them to become dominant within the population [10, 22]. AMR can arise due to genetic mechanisms, which involve changes to the genetic information of cells and result in phenotypes with selective advantages [10, 22, 23], and non-genetic mechanisms, which can produce heritable (and reversible) resistance to a drug without changing the underlying genetic code of the organism [20]. The exact biological mechanisms responsible for the development of AMR within cells can vary greatly among cells of different biological kingdoms (for example, between bacteria and fungi) due to differences in cell structure and the overall biological functions of their constituents [23, 56, 65], but there are resistance mechanisms that are present in all cells, such as AMR that arises due to genetic mutation and non-genetic gene expression noise [20].

1.3.2 Genetic Mechanisms of Antimicrobial Resistance

The main driving force of genetic AMR is evolution by natural selection due to beneficial genetic mutations that provide a selective advantage to the phenotypes the mutation produces, enabling them to preferentially survive and replicate within a population [10, 20, 22, 66]. With DNA being relatively stable, and the presence of biomolecular mechanisms designed to repair damaged DNA [32], genetic information is generally replicated accurately to produce offspring with the same genetic code as the parent cells [32, 67, 68]. Even so, occasional permanent changes to the DNA of a cell can occur for a variety of reasons (such as errors in the DNA error-checking process or the effects of chemicals and radiation that can alter DNA), leading to a mutation in the genetic code [66–68]. These mutations are often small, and may only affect a single gene, but any change to the genetic code and genes of an organism may result in differences in cellular function [66]. In the context of AMR, if these mutations affect the genes responsible for AMR beneficially they may provide the resultant mutant cells with an evolutionary advantage relative to cells that are susceptible to the antimicrobial drugs [10, 22, 23, 66].

Although genetic mutations are the main driving force for the development of long-term AMR, beneficial genetic mutations are rare [68, 69]. For cells that have already acquired a beneficial mutation, this enables them to maintain and pass on the mutation to their offspring (with a low possibility of the mutated genes being altered by further mutations), eventually leading to the phenotype created by the mutation becoming dominant within the population due to natural selection [10, 20]. Mutation rates vary but are generally estimated to be on the order of 10^{-10} mutations per base pair per generation [68, 69]. Beneficial mutations are even more scarce, given that very few mutations will result in a beneficial phenotypic difference [70]. Experimental AMR studies have established that the rate of a cell acquiring a beneficial AMR mutation is generally on the order of 10^{-6} per hour [71].

Given the small mutation rates required to produce beneficial genetic mutations that result in AMR, researchers have questioned the ability of purely genetic mechanisms to explain the development of AMR [20]. For large populations consisting of many cells that survive over long periods of time, the probability of a beneficial genetic mutation appearing somewhere in the population increases drastically, given that a cell population can consist of thousands to billions of individual cells, each of which having long lifespans ranging from days to years [72, 73]. But for small populations where individual cells do not survive long (which is often the case when cells are exposed to an antimicrobial drug), the possibility of a beneficial genetic mutation occurring before the population goes extinct is low [10]. This has led researchers to postulate that non-genetic mechanisms play a critical role in the development of AMR [10, 20].

1.3.3 Non-Genetic Mechanisms of Antimicrobial Resistance

Over the past few decades, the importance of non-genetic mechanisms to the development of AMR has been increasingly investigated [10, 20]. Non-genetic AMR is defined as resistance that arises without modifications to the genetic code of the cell [20]. Of particular importance is the phenotypic heterogeneity caused by gene expression noise [16, 38], as the processes of transcription and translation are common to all cells [32, 33]. As described in Section 1.2.4, the processes of transcription and translation that result in the expression of genes which encode specific biological functions within cells are not deterministic, mainly due to stochastic biochemical reactions [16]. This can result in a large variance in gene expression levels, resulting in phenotypic differences among genetically identical cells (Figure 1.2) [16, 38, 74].

Non-genetic phenotypic heterogeneity among genetically identical cells has been increasingly shown to play an important role in the development of AMR [10, 20]. Populations of genetically identical cells that exhibit a large amount of phenotypic heterogeneity can produce cells with phenotypes that are better suited to survive and grow under drug exposure [10, 20], as illustrated in Figure 1.3. Additionally, it has been shown that heritable AMR can potentially arise independently of mutation due to non-genetic mechanisms [75]. Experimental studies using synthetic gene networks (or "circuits") that control gene expression levels [10] have quantified the rates at which non-genetic AMR arises and the stability of non-genetically resistant phenotypes over multiple generations in cellar populations [63, 76]. Along with phenotypic heterogeneity arising due to gene expression noise, other mechanisms of non-genetic AMR have been investigated, such as "heteroresistance" in yeast [56] and "persistence" in bacteria [77]. These studies have illustrated the importance of considering non-genetic mechanisms alongside genetic mechanisms when investigating AMR.

The model presented in Chapter 2 incorporates non-genetic AMR by modelling cell population level characteristics that arise due to underlying physical mechanisms that take place at the molecular level and lead to phenotypic heterogeneity among genetically identical cells in the same environment [10, 16, 20, 74].

1.3.4 Evolution of Antimicrobial Resistance in Cell Populations

Studying the evolution of AMR within cellular populations is of particular importance for understanding how cell populations, especially those that are pathogenic to humans, develop and maintain AMR through time [20]. In the context of this

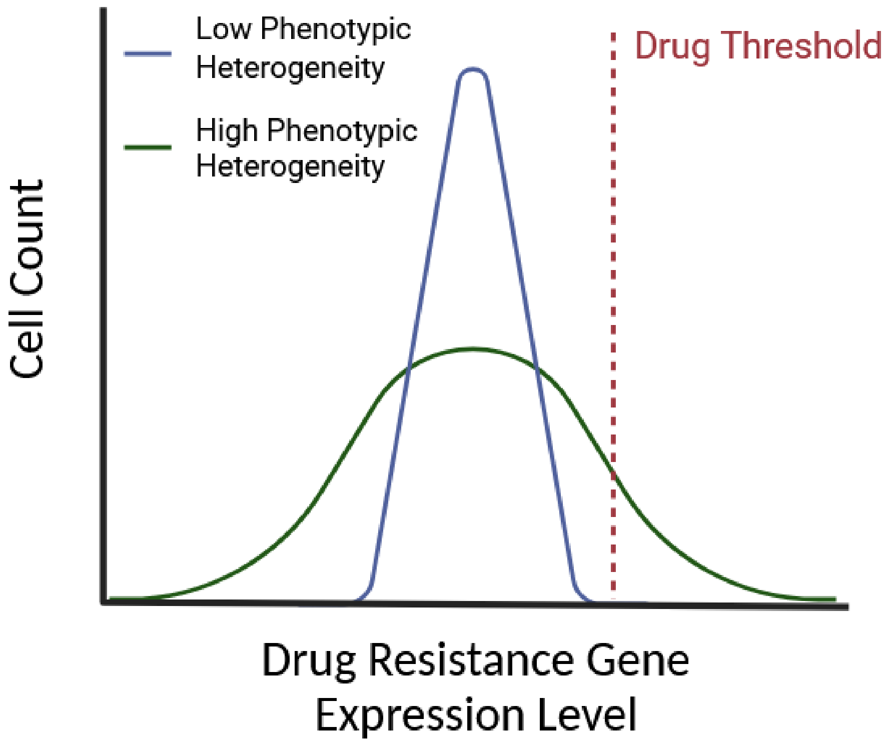

Figure 1.3: High non-genetic phenotypic heterogeneity can lead to the survival of cells which express genes that aid in AMR in populations of genetically identical cells [10]. Cells left of the drug threshold succumb to the effects of the drug. Figure inspired by Figure 1 in [10] and Figure 1 in [71].

thesis, the evolution of a cell population is defined as changes in the characteristics of the population that are maintained over time [78]. With respect to AMR, this often corresponds to a genetic mutation that offers resistance to a drug (and therefore a selective advantage) becoming dominant in the cell population [10], i.e., the phenotype produced by the genetic mutation becomes the phenotype that makes up the majority of the population [78]. Figure 1.4(A) highlights this scenario, where a drug-resistant mutation appears at some point in time (the "first appearance time" of the genetic mutation) in a genetically identical population of cells susceptible to a given drug. Over time, the phenotype created by the genetic mutation will become the dominant phenotype in the population due to the selective advantage that the genetic mutation creates. The time at which this phenotype becomes overwhelmingly dominant in the total population can be defined as the “fixation time”, which can be used as a quantitative measure of genetic evolution in cell populations [78, 79]. Phenotypes that are resistant to a given drug due to non-genetic mechanisms also provide a selective advantage when the population is exposed to a drug, but these phenotypes are not sustained after the drug is removed due to the stochastic nature of these phenotypes [10], which is illustrated in Figure 1.4(B).

Although the effects of genetic mutations and non-genetic phenotypic variation on cell populations are increasingly well understood individually, the effects that non-genetic AMR has on the evolution of AMR within populations of cells is an open question [10, 11, 20]. Figure 1.4(C) provides a visual representation of this problem. Qualitative reasoning in the literature generally indicates that non-genetic resistance is beneficial to the genetic evolution of AMR mutations by suggesting that the extended survival and growth time frames provided by non-genetically resistant phenotypes enable drug-resistant mutations to occur and fixate within cell populations [20, 80, 81]. Many qualitative and experimental investigations of this sort are given in the literature, but there is a lack of studies that have investigated the quantitative implications of non-genetically resistant phenotypes being present in a cell population that can develop AMR through beneficial mutations [10]. The work presented in Chapter 2 aims to fill this gap by investigating the evolutionary dynamics of such a system through the use of quantitative modelling. Although the structure of the model in Chapter 2 was motivated by AMR characteristics observed in yeast, it can in theory be modified to account for AMR characteristics specific to other organisms (e.g. bacteria). The mathematical and computational methods used to develop, solve, and analyze the model presented in Chapter 2 are discussed in the remainder of this chapter.

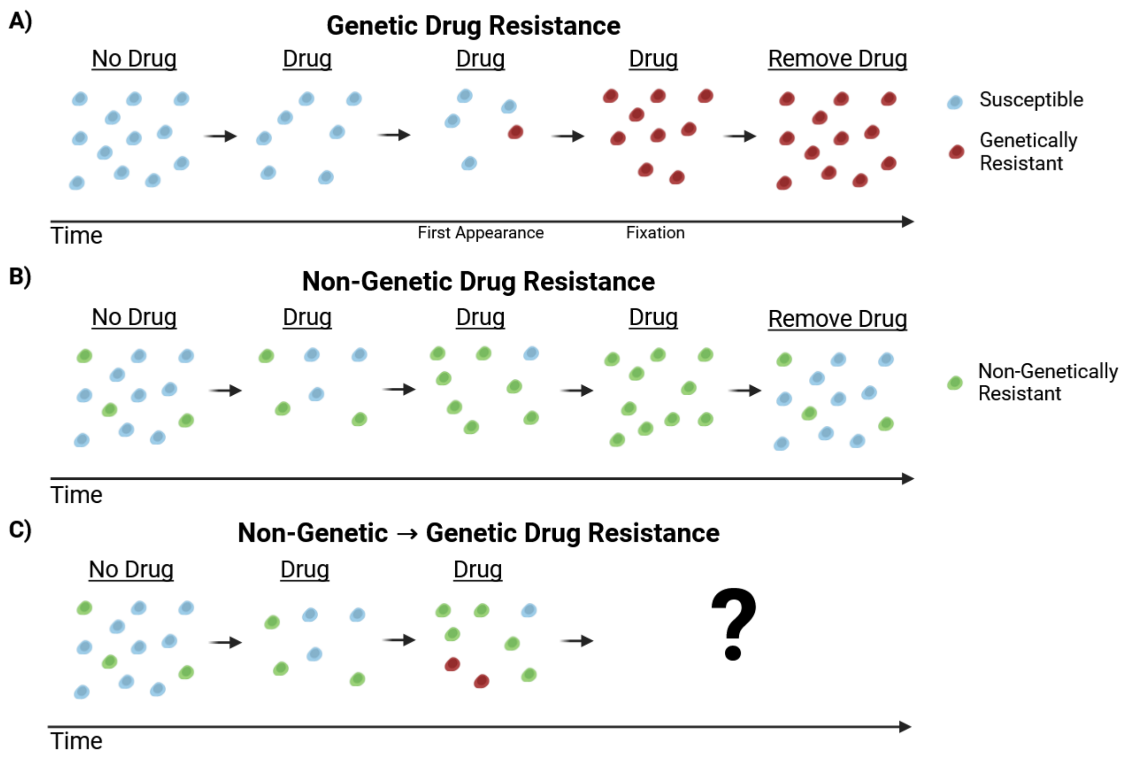

Figure 1.4: Schematic showing the time evolution of different cell populations when exposed to a drug. Figure inspired by Figure 1 in [10]. A) A genetically identical susceptible population undergoes drug treatment. A beneficial genetic mutation arises, providing a selective advantage to the phenotype produced allowing it to survive and fixate in the population. When the drug is removed, the phenotype produced by the genetic mutation remains the dominant phenotype in the population. B) A genetically identical susceptible population containing phenotypic heterogeneity is exposed to a drug. Phenotypes that are non-genetically resistant survive and propagate while some of the remaining susceptible cells acquire non-genetic resistance, eventually resulting in a full population of non-genetically resistant cells. Once the drug is removed, the population returns to the original distribution of phenotypes due to the reversibility of non-genetic resistance. C) A cell population under similar conditions that contains all three phenotypes. The overall evolution of cell populations containing susceptible, non-genetically resistant, and genetically resistant cells is an active area of study and the work presented in Chapter 2 proposes a model to investigate this scenario.

1.4 Quantitative Modelling of Antimicrobial Resistance at the Cell Population Level

1.4.1 Cell Population Dynamics

Cells are seldom isolated entities - they generally grow within large populations where individual cells affect the growth, survival, functions, and overall dynamics of other cells in the population [64]. Although the behaviour of individual cells and their interactions with other cells are complex, mathematical modelling can be used to model the collective behaviour of all cells within a population to determine high-level characteristics of cell population dynamics and evolution [64].

Cell populations can be homogeneous, where each individual shares the same genetic makeup and phenotypic characteristics as all other individuals in the population, or heterogeneous, where different subpopulations of cells having different phenotypes make up the overall population [64]. As discussed in the previous section, genetically identical cells can result in heterogeneous populations due to significant variation in gene expression due to stochastic and epigenetic mechanisms [20]. Many mathematical models have been proposed to investigate the time evolution of both idealistic homogeneous populations and heterogeneous populations made up of many subpopulations that interact and compete with each other [24, 25, 64]. With these models, the time evolution of cell populations can be quantified, providing an efficient and robust means to carry out investigations that would otherwise be extremely time-consuming and costly to study in a biological lab [64]. Models of cell population dynamics can also be used to guide experimental studies by producing novel hypotheses that can be tested experimentally, by applying mathematical models to analyze experimental data and determine certain aspects of the system, or by reducing the space of potential experiments needed to investigate a given system [10, 11, 64]. Along with aiding biological experiments, quantitative mathematical modelling can be used to guide clinical treatments by providing healthcare researchers with a means to better understand and predict the behaviour of cell populations responsible for certain infections and diseases [10, 11, 19].

Cell population dynamics models generally fall into one of two categories: 1) deterministic models that approximate cell populations as a continuous variable through time, and 2) stochastic models that take into account the discrete nature of cells within a population to account for stochastic fluctuations that may affect the overall population dynamics through time [64]. By applying these models in the regimes where their underlying assumptions are satisfied, characteristics of the time evolution of cell populations can be quantitatively investigated. Characteristics of the biological mechanisms of AMR can be incorporated into population-level models to investigate the time evolution of the population when the cells are exposed to a drug environment [10, 11]. In the following sections, I will discuss deterministic and stochastic modelling approaches commonly used for studying cell population dynamics, with an emphasis on the particular methods used to produce the results in Chapter 2.

1.4.2 Deterministic Modelling of Cell Population Dynamics

The most common deterministic models of cell population dynamics are population growth models that take the form of an ordinary differential equation (ODE) that describes the rate of change of a cell population where the solution provides the trajectory of the population size through time [64]. For heterogeneous populations containing multiple subpopulations, this framework is expanded into a set of coupled ODEs where each differential equation describes the dynamics of an individual subpopulation [64]. If we consider a cell population consisting of three distinct subpopulations, similar to the model presented in Chapter 2, we can model the growth of the system with the following set of coupled ODEs:

$$\frac{dx}{dt} = f(t, x, y, z) \tag{1.15}$$

$$\frac{dy}{dt} = g(t, x, y, z) \tag{1.16}$$

$$\frac{dz}{dt} = h(t, x, y, z) \tag{1.17}$$

where $x(t)$, $y(t)$ and $z(t)$ are the solutions to each equation and describe the number of cells in each subpopulation as functions of time [64]. The rate of change of the total population $x + y + z$ can be modelled by combining these equations:

$$\frac{d(x + y + z)}{dt} = \frac{dx}{dt} + \frac{dy}{dt} + \frac{dz}{dt} = f(t, x, y, z) + g(t, x, y, z) + h(t, x, y, z) \tag{1.18}$$

The functions $f(t, x, y, z)$, $g(t, x, y, z)$ and $h(t, x, y, z)$ describe the rate of change of each subpopulation and depend on the solutions $x(t)$, $y(t)$ and $z(t)$. Multiple mathematical models have been developed to define these functions based on the observed biological behaviour of cell populations [64]. Population growth models also closely resemble the mathematical rate equations that describe chemical reactions and follow mass action kinetics [43], enabling cell population growth to be modelled as a set of "reactions" [64], which will be discussed further in Sections 1.4.4-1.4.5.

Without any external restraints, such as constricted space or lack of nutrients, cell populations grow exponentially [64]. If we consider a cell population N, we can model exponential growth with the ODE

$$\frac{dN}{dt} = rN \tag{1.19}$$

where r is the population growth rate [64]. The growth rate r is often used as a measure of biological fitness and models how well a given population is adapted to the growth environment, as higher growth rates lead to more efficient population growth [63, 64]. This simple ODE can be solved analytically, giving the solution

$$N(t) = N_0 e^{rt} \tag{1.20}$$

where N_0 is the initial size of the population, showing that the population size grows exponentially with a growth rate r [64]. This equation can accurately model cell population growth when there are few external constraints on the system but fails to accurately model the full dynamics of realistic cell population growth where space and energy sources are limited [64]. For cell populations growing in a large volume with plentiful resources, exponential growth can accurately model cell population dynamics until the populations become large enough to be limited by the environment, in which case the population growth saturates until a steady concentration is reached [64, 82]. This is more accurately modelled by modifying the exponential growth model with a phenomenological inhibition factor $z(N)$ that ranges from 1 to 0 to scale the growth rate as the population N increases, giving the updated equation

$$\frac{dN}{dt} = z(N)rN = r'N \tag{1.21}$$

where $r' = z(N)r$ is the effective growth rate of the population [64]. This equation satisfies the following limit:

$$\lim_{N\to K} \frac{dN}{dt} = \lim_{N\to K} z(N)rN = 0 \tag{1.22}$$

where K is the "carrying capacity" of the system and is defined as the concentration at which the population is no longer able to grow any larger [64, 82]. A model of this form separates the population dynamics into three phases: (1) the exponential phase, which can accurately be described by the exponential growth model and occurs during early growth where $z(N) \approx 1$, (2) a saturation phase where the growth rate decreases, corresponding to $0 < z(N) < 1$, and (3) the stationary phase where the population no longer grows, corresponding to $z(N) \approx 0$ [64]. The time spent in each phase and how the system transitions between each phase depends on the form of the inhibition function $z(N)$. In the case where there are multiple subpopulations within the total population such that $N = x + y + z + ...$, the inhibition function is applied to each of the individual growth rates of the subpopulations [64]. If we define $N = x + y + z$ and again consider the three population models described in Equations 1.15-1.17 and use Equation 1.21 to model the growth of each subpopulation using rates r_x, r_y and r_z, we have

$$\frac{dx}{dt} = z(N)r_x x \tag{1.23}$$

$$\frac{dy}{dt} = z(N) r_y y \tag{1.24}$$

$$\frac{dz}{dt} = z(N) r_z z \tag{1.25}$$

for each of the subpopulations and

$$\frac{dN}{dt} = \frac{d(x+y+z)}{dt} = \frac{dx}{dt} + \frac{dy}{dt} + \frac{dz}{dt} = z(N)r_x x + z(N)r_y y + z(N)r_z z = z(N)(r_x x + r_y y + r_z z) \tag{1.26}$$

for the total population. The inhibition function $z(N)$ is used to model the effects that the environment (which consists of resources, chemicals, available space, and other cells) has on the growth of the population, leading to an effective growth rate of $r' = z(T)r$ where r is the maximum exponential growth rate seen when $z(T) = 1$ [64]. Many models of cell population growth have been developed that use different inhibition functions [64], such as the logistic growth model where $z(N) = 1 - N/K$ [64, 82, 83], Monad kinetics where $z(N) = \frac{N}{K_N + N}$ and K_N defines the concentration of N where the inhibition function is 0.5 [64, 84], Allee models that define a critical population size N_c required for growth and use an inhibition function of $z(N) = (1 - \frac{N}{K})(\frac{N}{N_c} - 1)$ [64, 85–87], and the more general Baranyi model where the inhibition function can also be scaled by a time-dependent adjustment function $\alpha(t)$ that ranges from 0 to 1 to model the "lag" or "adaptation" phase of early cell growth making the total inhibition function $\alpha(t)z(N)$ where $z(N)$ can be any unitless inhibition function that ranges from 1 to 0 [64, 88, 89], along with others [64].

The choice of inhibition function generally depends on the empirically observed growth curve characteristics of the system being modelled [64]. In the work presented in the following chapter, a Hill-type inhibition function defined by [90]

$$z(N) = \frac{h^n}{h^n + N^n} \tag{1.27}$$

where h is the point at which $z(N)$ is at its half maximum, and n is the Hill coefficient that defines the overall steepness of the inhibition function [90], is used with a general Baranyi model to investigate cell population dynamics in a drug environment. The choice of using a Hill function is based on observations of cell population growth curves while cells are growing in a drug environment, as the shapes of these curves are better modelled by an adjustable inhibition function (using the parameters h and n) than the standard logistic inhibition function [90]. It also enables intraspecific competition (competition between members of the same species for limited resources [25, 26]) to be modelled. An example Baranyi-Hill model, along with exponential and logistic growth models, are shown in Figure 1.5.

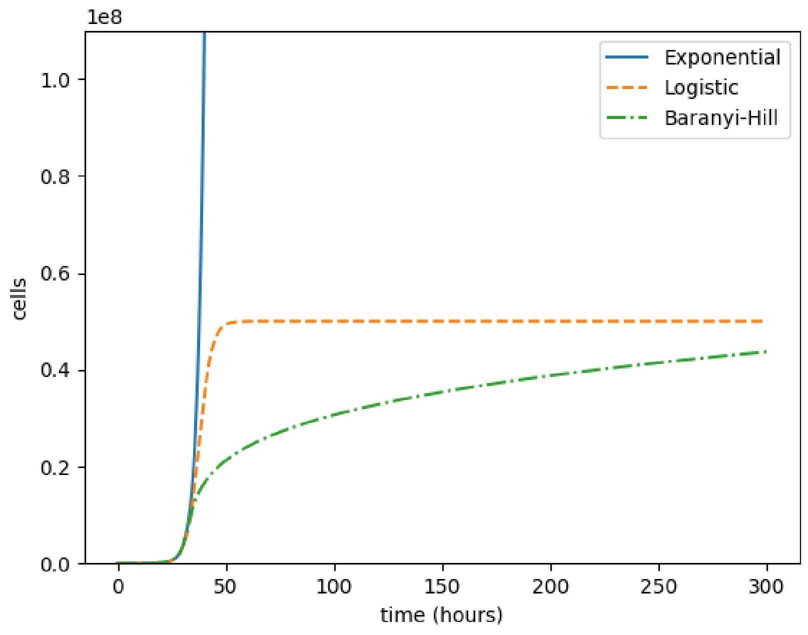

Figure 1.5: Numerical solutions of growth curves produced by exponential, logistic, and Baranyi-Hill models of cell population dynamics. Each model here has the same maximum growth rate (r) value. The exponential model creates indefinite exponential growth of the population, while the logistic model remains nearly exponential until the carrying capacity is reached. The Baranyi-Hill model saturates over a longer timescale, which can be more representative of growth in a drug environment than the standard logistic growth model [90].

A few adjustments to the equations described above are required to formulate the model presented in Chapter 2. These adjustments add additional terms to the equations based on mass action kinetics, by considering additional characteristics of the subpopulations (such as cell death and phenotypic switching) as "reactions" that occur at a given rate [43, 64]. Firstly, the above models assume that the cells survive indefinitely (i.e. once a cell is present in the population, it will remain in the population as $t \to \infty$). This assumption may be reasonable for short time scales, but for the longer time scales required to investigate biological evolution cell death must be considered. This can be obtained by adding a negative cell death term for each of the subpopulations, which depends on a death rate δ_i and the size of the given population i, to model the death of the cells in each subpopulation either due to natural means, such as aging [73], or due to external factors, such as the application of a cidal drug [54, 55]. This can be done by adding a death term $-\delta_i i$, where δ_i is the death rate of population i, to the ODE for each subpopulation, making the equations

$$\frac{dx}{dt} = r'_x x - \delta_x x \tag{1.28}$$

$$\frac{dy}{dt} = r'_y y - \delta_y y \tag{1.29}$$

$$\frac{dz}{dt} = r'_z z - \delta_z z \tag{1.30}$$

The coupled ODE model can also be adjusted to account for interactions between the subpopulations. For the model presented in Chapter 2, we are particularly interested in "switching" between subpopulations, corresponding to cells in one subpopulation changing their phenotype to that of another subpopulation (either through mutations or non-genetic mechanisms). Switching can be modelled by incorporating terms into the equations that model cells leaving one subpopulation and entering another [90]. These terms have the same mathematical form but opposite signs, with the negative term being added to the equation for the subpopulation that has cells switching to another type (i.e. leaving the subpopulation, hence the negative sign), while the positive term is added to the subpopulation that is being switched into [64]. This is modelled by a first-order reaction $j \xrightarrow{r_{i,j}} i$ between subpopulations i and j [64] and corresponding terms are added to the equations based on mass action kinetics [43, 64]. As an example using the model described above, we can consider switching that occurs between each of the three subpopulations (i.e. x can switch into y and z, y can switch into x and z, and z can switch into x and y). We can assume that switching occurs exponentially with some rate $r_{i,j}$ where i corresponds to the subpopulation that cells are switching into and j corresponds to the subpopulation of cells that are undergoing switching. Applying mass action kinetics, this produces the model

$$\frac{dx}{dt} = r'_x x + r_{x,y} y + r_{x,z} z - r_{y,x} x - r_{z,x} x - \delta_x x \tag{1.31}$$

$$\frac{dy}{dt} = r'_y y + r_{y,x} x + r_{y,z} z - r_{x,y} y - r_{z,y} y - \delta_y y \tag{1.32}$$

$$\frac{dz}{dt} = r'_z z + r_{z,x} x + r_{z,y} y - r_{x,z} z - r_{y,z} z - \delta_z z \tag{1.33}$$

Since the switching terms cancel out when solving for the total population growth (due to the opposite signs), switching does not affect the growth of the overall population but can drastically change the dynamics of the individual subpopulation.

The model shown in equations 1.31-1.33 is the general form of the model presented in Chapter 2. The model incorporates inhibited exponential cell growth due to the environment (which includes intraspecific competition between the subpopulations), cell death, and phenotypic switching between subpopulations (due to genetic mutations and non-genetic phenotypic heterogeneity). This model can be adjusted to account for certain biological conditions by removing transitions between specific subpopulations (for example, letting x switch into y but not the reverse) that are not biologically relevant and by tuning the numerical values of the parameters based on experimental results.

1.4.3 Numerical Solutions of Coupled ODEs

To solve the system of coupled ODEs presented in the next chapter, MATLAB's ode45 solver was used [91, 92]. ode45 is the recommended ODE solver in MATLAB for systems of non-stiff equations and is based on an explicit Runge-Kutta formula, the Dormand-Prince pair, which is a single-step method where the solution at a given time depends on the solution at the previous time step [92, 93]. Explicit Runge-Kutta methods consist of algorithms that solve ODEs by estimating the solution at each time step using a weighted average over slope estimations at different points along the step interval (generally the middle, end, and multiple mid-points), calculated using the ODE and the solution at the previous time point [93, 94]. The Dormand-Prince pair is an adaptive method that tunes the parameters of the algorithm to minimize truncation errors while maintaining computational efficiency [92–94].

A standard initial value problem can be defined as:

$$\frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0 \tag{1.34}$$

where t is time, y is the unknown solution, $f(t, y)$ is a function of t and y, and y_0 is the solution for y at the initial time t_0. To numerically solve an initial value problem of this form, the Dormund-Prince pair uses a fifth-order Runge-Kutta formula. Using a time step Δt, a fifth-order Runge-Kutta method provides a numerical solution at the next time point $t_{n+1} = t_n + \Delta t$ using the formula [93, 94]

$$y_{n+1} = y_n + \sum_{i=1}^{6} b_i k_i \tag{1.35}$$

where the sum is a weighted average over slope estimations k_i using weights b_i. The slope estimations are calculated using the following equations [93, 94]

$$k_1 = \Delta t f(t_n, y_n) \tag{1.36}$$

$$k_2 = \Delta t f(t_n + c_2 \Delta t, y_n + a_{21} k_1) \tag{1.37}$$

$$k_3 = \Delta t f(t_n + c_3 \Delta t, y_n + a_{31} k_1 + a_{32} k_2) \tag{1.38}$$

$$k_4 = \Delta t f(t_n + c_4 \Delta t, y_n + a_{41} k_1 + a_{42} k_2 + a_{43} k_3) \tag{1.39}$$

$$k_5 = \Delta t f(t_n + c_5 \Delta t, y_n + a_{51} k_1 + ... + a_{54} k_4) \tag{1.40}$$

$$k_6 = \Delta t f(t_n + c_6 \Delta t, y_n + a_{61} k_1 + ... + a_{65} k_5) \tag{1.41}$$

where c_i and a_{ij} are constants used to optimize the algorithm and are defined, along with the weights b_i, in the original Dormund-Prince paper [93].

For an initial value problem consisting of a system of ODEs, the slope estimations are calculated for each ODE to numerically solve the system [94]. If we consider a system of three coupled ODEs (similar to the model presented in the following chapter) given by the initial value problem

$$\frac{dx}{dt} = f(t,x,y,z), \quad x(t_0) = x_0 \tag{1.42}$$

$$\frac{dy}{dt} = g(t,x,y,z), \quad y(t_0) = y_0 \tag{1.43}$$

$$\frac{dz}{dt} = h(t,x,y,z), \quad z(t_0) = z_0 \tag{1.44}$$

the fifth-order Runge-Kutta solution with Dormund-Prince parameters is then

$$x_{n+1} = x_n + \sum_{i=1}^{6} b_i k_i \tag{1.45}$$

$$y_{n+1} = y_n + \sum_{i=1}^{6} b_i l_i \tag{1.46}$$

$$z_{n+1} = z_n + \sum_{i=1}^{6} b_i m_i \tag{1.47}$$

where k_i, l_i and m_i are the slope estimations for each equation given by [94]

$$k_1 = \Delta t f(t_n, y_n) \tag{1.48}$$

$$l_1 = \Delta t g(t_n, y_n) \tag{1.49}$$

$$m_1 = \Delta t h(t_n, y_n) \tag{1.50}$$

$$k_2 = \Delta t f(t_n + c_2\Delta t, x_n + a_{21}k_1, y_n + a_{21}l_1, z_n + a_{21}h_1) \tag{1.51}$$

$$l_2 = \Delta t g(t_n + c_2\Delta t, x_n + a_{21}k_1, y_n + a_{21}l_1, z_n + a_{21}h_1) \tag{1.52}$$

$$m_2 = \Delta t h(t_n + c_2\Delta t, x_n + a_{21}k_1, y_n + a_{21}l_1, z_n + a_{21}h_1) \tag{1.53}$$

$$\ldots$$

$$k_6 = \Delta t f(t_n + c_6\Delta t, x_n + a_{61}k_1 + \ldots + a_{65}k_5, y_n + a_{61}l_1 + \ldots + a_{65}l_5, z_n + a_{61}m_1 + \ldots + a_{65}m_5) \tag{1.54}$$

$$l_6 = \Delta t g(t_n + c_6\Delta t, x_n + a_{61}k_1 + ... + a_{65}k_5, y_n + a_{61}l_1 + ... + a_{65}l_5, z_n + a_{61}m_1 + ... + a_{65}m_5) \tag{1.55}$$

$$m_6 = \Delta t h(t_n + c_6\Delta t, x_n + a_{61}k_1 + ... + a_{65}k_5, y_n + a_{61}l_1 + ... + a_{65}l_5, z_n + a_{61}m_1 + ... + a_{65}m_5) \tag{1.56}$$

1.4.4 Stochastic Modelling of Cell Population Dynamics Using Reaction Systems and SSA

Although deterministic population models are useful, they fail to capture the true stochastic behaviour of cell population dynamics that arise due to the discrete nature of cell populations [16]. Stochastic effects become particularly important at low cell numbers, where stochastic fluctuations can greatly affect the resulting dynamics of the system [45–47, 64]. For example, stochastic fluctuations in the early stages of cell population growth in stressful environments may lead to the extinction of the population, as fluctuations in the growth and death of cells can lead to all viable cells dying before full population growth is reached, a phenomenon investigated in Chapter 2. Therefore, a stochastic modelling framework is required to investigate cell population dynamics in low cell number regimes. Many stochastic models exist, but the most commonly used framework to study stochastic cell population dynamics is modelling the system as a set of stochastic processes (or "reactions") and applying Gillespie's stochastic simulation algorithm (SSA) [45, 46]. The SSA was originally developed to produce exact realizations of the GPF that solves the CME for molecular reaction systems but has since found success in modelling other stochastic systems that can be modelled as a set of discrete events defined as continuous-time Markov processes [46, 47, 95, 96].

The SSA is a Monte Carlo method that produces single stochastic realizations (trajectories) from the GPF that solves the CME of a given reaction system where the reactions are modelled as discrete stochastic events that occur in real-time [45–47]. A reaction system is defined as a set of reaction channels $\{R_1, ..., R_M\}$ where M

is the number of reactions [47]. Each reaction R_i is described by reactants $\{S_1, ..., S_N\}$ where N is the number of reactant species, the reaction products, and a reaction constant c_i [47]. Reactions can be of the following types [45]:

$$* \xrightarrow{c} products \tag{1.57}$$

$$S_i \xrightarrow{c} products \tag{1.58}$$

$$S_i + S_j \xrightarrow{c} products \tag{1.59}$$

$$2S_i \xrightarrow{c} products \tag{1.60}$$

$$S_i + S_j + S_k \xrightarrow{c} products \tag{1.61}$$

$$S_i + 2S_k \xrightarrow{c} products \tag{1.62}$$

$$3S_i \xrightarrow{c} products \tag{1.63}$$

where Equation 1.57 is a zeroth-order (or "external source") reaction which does not depend on the reactants $\{S_1, ..., S_N\}$ [45]. Equation 1.58 is a first-order reaction [45] and is of particular importance for the work presented in Chapter 2, where our model is expressed as a set of first-order reactions. The mathematical form of the reaction constants c_j depends on the order of the reaction channel and for first-order reactions it is exactly the exponential growth rate (also referred to as the reaction rate) used in the deterministic rate equations described in Section 1.4.2 (i.e., $c_i = r_i$) [45]. For higher-order reactions c_j depends on additional factors, such as the volume occupied by the reactants and underlying physical interactions between reactants, and generally requires the application of statistical mechanics to determine [45, 47].

With the set of reaction channels $\{R_1, ..., R_M\}$, reaction constants $\{c_1, ..., c_M\}$, reactants $\{S_1, ..., S_N\}$ and products, the SSA can be used to compute exact individual trajectories of the reaction system through time [45–47]. Single trajectories provide limited information, but distributions can be created by computing a large number of trajectories to estimate the overall GPF and to analyze the statistical behaviour of the system [45–47]. The number of trajectories required to create accurate distributions depends on the system but can be found by adding trajectories to the distribution until the overall distribution is relatively unchanged by additional realizations (often requiring thousands of trajectories) [45, 46].

The SSA computes trajectories by using randomly generated numbers from the uniform distribution to determine the time to the next reaction event and which reaction will occur [45–47]. To do this, the propensity functions for each reaction, which describe the probability per unit time of a reaction R_i occurring in the current state of the system $\boldsymbol{X}(t) = \boldsymbol{x}$ at a given time t, are required [45, 47]. $\boldsymbol{x}$ is defined by a vector containing the number of reactants in each reactant species population at time t:

$$\boldsymbol{x} = [S_1(t),\ ..., S_N(t)] \tag{1.64}$$

where $S_j(t)$ corresponds to the total number of reactants within the population of the reactant species S_j at time t [47]. Product (or state-change) vectors $\boldsymbol{v}_i = [v_{1i}, ..., v_{Ni}]$ can be defined for each reaction channel R_i and contain the change in each population S_j when reaction R_i occurs [47]. For first-order reactions, the propensity function $a_i(\boldsymbol{x})$ at time t is equal to the reaction rate multiplied by the total size of the reactant species population at time t [45, 47]. Therefore, for a reaction R_i with a reaction rate k_i and reactant S_j, the propensity function of R_i when the system is in the state $\boldsymbol{x}$ at time t is given by [45, 47]:

$$a_i(\boldsymbol{x}) = k_i S_j(t) \tag{1.65}$$

To calculate the time to the next reaction and which reaction will occur, the summation of all propensity functions $\{a_1(\boldsymbol{x}),\ ...,\ a_M(\boldsymbol{x})\}$ is needed [45, 47]. Defining the summation of propensity functions as $a_0(\boldsymbol{x})$, we have [47]:

$$a_0(\boldsymbol{x}) = \sum_{n=1}^{M} a_n(\boldsymbol{x}) \tag{1.66}$$

Using two uniform random numbers r_1 and r_2, the time to the next reaction τ and which reaction will occur, symbolized by its index i, are then determined using the following equations [45, 47]:

$$\tau = \frac{1}{a_0(\boldsymbol{x})} ln\left(\frac{1}{r_1}\right) \tag{1.67}$$

$$i = smallest\ value\ that\ satisfies\ \sum_{i'=1}^{i} a_{i'}(\boldsymbol{x}) \ > \ r_2 a_0(\boldsymbol{x}) \tag{1.68}$$

These equations are derived analytically by considering the joint probability distribution $p(\tau,\, i \,|\, \boldsymbol{x},\, t)$ which is defined as [45, 47]:

$$p(\tau,\, i \,|\, \boldsymbol{x},\, t) d\tau = a_j(\boldsymbol{x}) e^{-a_0(\boldsymbol{x})\tau} d\tau \tag{1.69}$$

and describes the probability that the reaction R_i will occur in the time interval $[t+\tau,\ t+\tau+d\tau)$ when the system is in the state $\boldsymbol{x}$ (see [45, 47] for the full derivation of $p(\tau,\, i \,|\, \boldsymbol{x},\, t)$).

The algorithm is implemented through the following steps [45, 47]:

1. Set the initial values for the time and state vector. Create state-change vectors for each reaction containing the change in population size produced by the reaction products.

2. Generate two uniform random numbers. Evaluate all propensity functions and their sum for the given state.

3. Calculate the time to the next reaction and which reaction will occur. Propagate the time forward using the calculated time to the next reaction and update the state vector using the state-change vector for the determined reaction.

4. Record the new time and state.

5. Repeat steps (2)-(4) until the desired time is reached.

This process can then be repeated for the desired number of trajectories. The pseudocode I used to implement the SSA to calculate a large number of trajectories for the results presented in Chapter 2 is shown in Algorithm 1.

Algorithm 1 Pseudocode implementation of the SSA for multiple trajectories. The implementation is based on the original formulation of the Gillespie SSA [45]. The C code I wrote to implement the algorithm and produce the results in Chapter 2 is shown in Appendix A.2.

```
Define initial time $t = t_0$, initial state vector $\boldsymbol{x} = \boldsymbol{x_0}$, and initial trajectory number
$T = 1$. Define state-change vectors $\boldsymbol{v}_i$ that contain the products for each reaction $R_i$.
Define the final time of each trajectory as $t_{final}$ and the total number of trajectories
as $T_{end}$.

while $T \leq T_{end}$ do
    Set $t = t_0$ and $\boldsymbol{x} = \boldsymbol{x_0}$.
    while $t \leq t_{final}$ do
        Generate two uniform random numbers $r_1$ and $r_2$.
        Calculate each $a_i(\boldsymbol{x})$ using equation (1.65).
        Calculate $a_0(\boldsymbol{x})$ using equation (1.66).
        Calculate $\tau$ and $i$ using equations (1.67) and (1.68).
        Set $t = t + \tau$ and $\boldsymbol{x} = \boldsymbol{x} + \boldsymbol{v}_i$.
        Record the new state $(\boldsymbol{x}, t)$.
    end while
    Store each $(\boldsymbol{x}, t)$ calculated for the trajectory.
    Set $T = T + 1$.
end while
```

Distributions over the trajectories calculated using the SSA simulate the GPF of the reaction system and can be analyzed to determine the statistical characteristics of the system [45, 47]. For example, the fixation time of a genetic mutation can be calculated for each trajectory to create a fixation probability distribution over time to investigate the evolutionary dynamics of a cell population, which is investigated in Chapter 2. Additionally, cell population survival distributions can be created by analyzing the number of trajectories that die out (equivalent to all populations in the state vector reaching zero), another case that is investigated in Chapter 2.

Multiple formulations of the SSA have been developed, with the version I have presented here and used to produce the results in Chapter 2 being referred to as the "direct method", which is the original formulation of the SSA presented by Gillespie [45] and does not make any approximating assumptions as it simulates *every* reaction that occurs in the system [45, 47]. The limitation of doing this is that it is computationally expensive [47]. Accelerated SSA methods have been produced, such as the Gibson-Bruck procedure [97] and tau leaping [47, 98], which use simplifying assumptions to reduce the computational cost of the algorithm, such as grouping multiple reactions over a predefined time step in the case of tau leaping [47, 98]. The computational efficiency of the direct method was sufficient to produce the results presented in Chapter 2 due to the relatively small size of the reaction system modelled, but accelerated methods are generally required for large systems where the direct method is no longer feasible [47].

1.4.5 Relating Deterministic Population Growth Models to Stochastic Reaction Systems

An interesting aspect of the deterministic population growth models presented in Section 1.4.2 is that their mathematical form closely resembles the reaction rate equations used to model chemical reaction systems under conditions where a deterministic approach can be applied as an accurate approximation (generally when the populations of each reactant species are large) [43, 45–47]. Additionally, each term in the ODEs has the same form as the propensity functions for first-order reactions described in Equation 1.65. Mathematically, reactions are continuous-time Markov processes, where each reaction changes the state of the system based on a random variable from the exponential distribution [47]. Cell populations are made up of discrete quantities that undergo stochastic processes as they evolve through time and the underlying dynamics of the population growth models presented in Section 1.4.2 can be thought of as arising due to a set of continuous-time Markov chain processes (or "reactions") [46, 64].

Continuing with the three-subpopulation example presented in Section 1.4.2, the system (Equations 1.31-1.33) can be translated into a set of reactions of the following form:

$$x \xrightarrow{r'_x} 2x \tag{1.70}$$

$$y \xrightarrow{r'_y} 2y \tag{1.71}$$

$$z \xrightarrow{r'_z} 2z \tag{1.72}$$

$$x \xrightarrow{\delta_x} \varnothing \tag{1.73}$$

$$y \xrightarrow{\delta_y} \varnothing \tag{1.74}$$

$$z \xrightarrow{\delta_z} \varnothing \tag{1.75}$$

$$x \xrightarrow{r_{y,x}} y \tag{1.76}$$

$$x \xrightarrow{r_{z,x}} z \tag{1.77}$$

$$y \xrightarrow{r_{x,y}} x \tag{1.78}$$

$$y \xrightarrow{r_{z,y}} z \tag{1.79}$$

$$z \xrightarrow{r_{x,z}} x \tag{1.80}$$

$$z \xrightarrow{r_{y,z}} y \tag{1.81}$$

where the exponential rates now become reaction constants and can be thought of as the probability per unit time that the given reaction will occur [47]. Equations 1.70-

1.72 model cell division, 1.73-1.75 model cell death, and 1.76-1.81 model switching between subpopulations. As these are first-order reactions, the propensity function of each reaction follows Equation 1.65 [45, 47]. Using the terminology of the previous section, the set of coupled ODEs used to describe the system can now be rewritten as a vector equation of the following form [47]:

$$\frac{d\boldsymbol{x}}{dt} = \sum_{i=1}^{M} \boldsymbol{v}_i a_i(\boldsymbol{x}) \tag{1.82}$$

where $\boldsymbol{x} = [x, y, z]$, $a_i(\boldsymbol{x})$ are the propensity functions for each reaction, and $\boldsymbol{v}_i$ are the state change vectors corresponding to the numerical change in each subpopulation when the reaction i occurs [47].

In the large number limit, trajectories produced by the SSA to simulate a given reaction system generally converge to the deterministic rate equations described above [47]. To show why this is the case, it is useful to discuss the Langevin equation for continuous-time Markov processes, which can be used to approximate the time evolution of a reaction system when certain conditions are met [47, 99]. For a given reaction system, a macroscopically infinitesimal time increment dt can be defined if the following two conditions are satisfied: (1) During dt, the propensity functions $a_i(\boldsymbol{x} = \boldsymbol{X}(t))$ of all reactions R_i do not change significantly, and (2) each reaction in the system occurs many times during dt [47, 99]. If these two conditions are met, the stochastic behaviour of the reaction system can be approximated using the Langevin equation [47]:

$$\boldsymbol{X}(t + dt) = \boldsymbol{X}(t) + \sum_{i=1}^{M} \boldsymbol{v}_i a_i(\boldsymbol{X}(t))dt + \sum_{i=1}^{M} \boldsymbol{v}_i \sqrt{a_i(\boldsymbol{X}(t))} N_i(t)\sqrt{dt} \tag{1.83}$$

where $N_i(t)$ are M are independent, uncorrelated, normal random variables with means of 0 and variances of 1 (for the full derivation of the Langevin equation with respect to the SSA, see [47, 99]) [47]. Equation 1.83 holds only if dt is small enough to satisfy condition (1) while also large enough to satisfy condition (2), which is generally possible when population sizes are large [47, 99]. The second term on the right side of equation 1.83 is deterministic and proportional to dt, while the third term is stochastic and proportional to $\sqrt{dt}$. The deterministic term in equation 1.83 scales linearly with the propensity functions $a_i(\boldsymbol{X}(t))$ while the stochastic term grows to the power of 1/2 of the propensity functions. As is clear from the propensity functions of first-order reactions (equation 1.65), the propensity functions grow proportionally with the size of the populations. Relative to the deterministic component, the stochastic component therefore scales to the power of $-1/2$ of the system size, implying that stochastic fluctuations in the reaction system scale with the inverse square root of the size of the system (a result that is common for stochastic processes [16]) [47]. For large population sizes, this means that the stochastic term essentially vanishes and the Langevin equation can be approximated as [47]

$$\boldsymbol{X}(t+dt) = \boldsymbol{X}(t) + \sum_{i=1}^{M} \boldsymbol{v}_i a_i(\boldsymbol{X}(t))dt \tag{1.84}$$

or

$$\boldsymbol{X}(t+dt) - \boldsymbol{X}(t) = \sum_{i=1}^{M} \boldsymbol{v}_i a_i(\boldsymbol{X}(t))dt \tag{1.85}$$

which is a function containing differentials and can be rewritten as

$$\frac{d\boldsymbol{X}(t)}{dt} = \sum_{i=1}^{M} \boldsymbol{v}_i a_i(\boldsymbol{X}(t)) \tag{1.86}$$

which is exactly the deterministic rate equation (a set of coupled ODEs) for a system where the dynamics can be described as a set of reactions with propensity functions $a_i(\boldsymbol{X}(t))$ shown in Equation 1.82 [47]. Therefore, when population sizes are large, the trajectories produced using the SSA can be accurately approximated using the deterministic rate equations described in Section 1.4.2. To show this convergence visually, Figure 1.6 provides example SSA and population growth equation simulations to a simple reaction system containing two subpopulations of cells (based on the model presented in Chapter 2). As can be seen, the SSA trajectories converge to the deterministic rate equation solutions as the size of each subpopulation becomes large, illustrating that the deterministic growth equations can provide an accurate approximation of growth dynamics for large populations [47, 64].

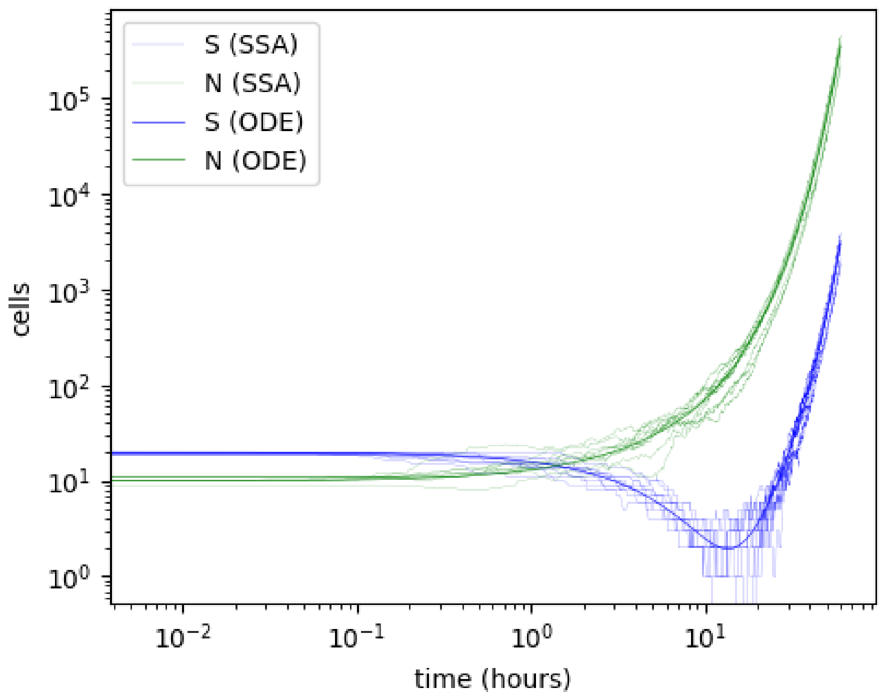

Figure 1.6: A system containing two cell populations (S and N) modelled by stochastic reactions and the corresponding deterministic population growth equations. The system incorporates Baranyi-Hill growth, switching, and death, with all parameters set to the same numerical values for each case. Ten SSA trajectories are shown by faint lines, while the bold lines show numerical solutions to the population growth ODEs. The SSA trajectories converge to the numerical solutions of the deterministic population growth ODEs as the population sizes become large, with the largest fluctuations occurring when subpopulation sizes are small. The model used in this plot is based on the model presented in Chapter 2 and uses low initial subpopulation sizes to highlight the stochastic fluctuations.

The relationship between SSA and deterministic rate equations is important for the work presented in Chapter 2, where we present a phenomenological model of cell population dynamics to investigate the evolution of AMR. A deterministic framework consisting of coupled population growth equations is used when population sizes are large and stochastic fluctuations are negligible. A stochastic framework is used to investigate the statistical properties of the system when population levels are low and the system cannot be modelled accurately using deterministic population growth equations. To do this, the model is reformulated as a set of first-order reactions and the SSA is used to simulate a large number of trajectories to create distributions and investigate the statistical properties of the system.

Chapter 2

Non-Genetic Resistance Facilitates Survival While Hindering the Evolution of Drug Resistance Due to Intraspecific Competition

The work presented in this chapter was previously published in the journal *Physical Biology* [1]. All codes written to produce the results are given in the Appendix of this thesis.

Physical Biology

OPEN ACCESS

RECEIVED
29 July 2022

ACCEPTED FOR PUBLICATION
23 August 2022

PUBLISHED
8 September 2022

PAPER

Non-genetic resistance facilitates survival while hindering the evolution of drug resistance due to intraspecific competition

Joshua D Guthrie[1] and Daniel A Charlebois[1,2,*]

[1] Department of Physics, University of Alberta, 11455 Saskatchewan Drive NW, Edmonton, Alberta, Canada
[2] Department of Biological Sciences, University of Alberta, 11455 Saskatchewan Drive NW, Edmonton, Alberta, Canada
[*] Author to whom any correspondence should be addressed.

E-mail: dcharleb@ualberta.ca

Keywords: antimicrobial resistance, deterministic models, population dynamics, stochastic simulations, fluctuating drug conditions, resistance evolution

Abstract

Rising rates of resistance to antimicrobial drugs threaten the effective treatment of infections across the globe. Drug resistance has been established to emerge from non-genetic mechanisms as well as from genetic mechanisms. However, it is still unclear how non-genetic resistance affects the evolution of genetic drug resistance. We develop deterministic and stochastic population models that incorporate resource competition to quantitatively investigate the transition from non-genetic to genetic resistance during the exposure to static and cidal drugs. We find that non-genetic resistance facilitates the survival of cell populations during drug treatment while hindering the development of genetic resistance due to competition between the non-genetically and genetically resistant subpopulations. Non-genetic resistance in the presence of subpopulation competition increases the fixation times of drug resistance mutations, while increasing the probability of mutation before population extinction during cidal drug treatment. Intense intraspecific competition during drug treatment leads to extinction of susceptible and non-genetically resistant subpopulations. Alternating between drug and no drug conditions results in oscillatory population dynamics, increased resistance mutation fixation timescales, and reduced population survival. These findings advance our fundamental understanding of the evolution of resistance and may guide novel treatment strategies for patients with drug-resistant infections.

1. Introduction

Antimicrobial (drug) resistance occurs when bacteria, viruses, fungi, and parasites no longer respond to drug therapy, making infections difficult or impossible to treat, which increases the risk of disease transmission, severe illness, and death [1]. The evolution of genetic drug resistance is known to arise from the natural selection of mutations or resistance genes that provide microbes with the ability to survive and proliferate during treatment [2]. It has also been shown that non-genetic mechanisms promote microbial phenotypic diversification and survival strategies in selective drug environments [3]. Phenotypic heterogeneity has important implications for drug resistance [4, 5], with heritable resistance potentially arising independently of genetic mechanisms [6]. The stochastic or 'noisy' expression of genes [7, 8] introduces phenotypic heterogeneity among genetically identical cells in the same drug environment, which can result in the fractional killing of clonal microbial populations [3, 9], as well as chemotherapy resistance in cancer [10]. This stochasticity is due in part to the inherently random nature of the biochemical reactions involved in the transcription and translation of genetic material, and can lead to the emergence of phenotypically distinct subpopulations within an actively replicating clonal cell population [8, 11]. Another form of non-genetic drug resistance called 'tolerance' occurs in fungi, in which a slow-growing subpopulation of cells (that are genetically identical to susceptible cells) emerges during antifungal drug treatment [12]; related phenomena occur in bacteria [13, 14] and cancer [15].

Non-genetic drug resistance has been proposed to promote the development of genetic drug resistance [4, 5, 10, 12, 16]. This process may be enhanced by the interaction between non-genetic and genetic mechanisms inside the cell [5]. For instance, promoter mutations can alter the expression noise levels of drug resistance genes [9], genetic network architecture can modulate gene expression noise to enhance drug resistance [17–19], and stress response genes can evolve elevated transcriptional variability through natural selection [20, 21]. Non-genetic mechanisms can facilitate the generation of genetic diversity by increasing the expression of key regulators involved in DNA replication, recombination, or repair [22, 23], as well as by enhancing the adaptive value of beneficial mutations during drug treatment [24] and promoting the fixation of favorable gene expression altering mutations [25]. Non-genetic phenotypic variability can impact cellular populations by providing a link between micro-scale dynamics (such as stochasticity at the molecular level) and macro-scale biological phenomena (including the fate of interacting cell populations) [26]. Such noise in biological systems may facilitate the adaptation to environmental stress by allowing distinct, co-existing cellular states in a population to find the best adaptive solution from multiple starting points [27]. However, there are conflicting views on how phenotypic heterogeneity may facilitate adaptive evolution [28] and the transition from non-genetic to genetic drug resistance remains to be quantified [5, 12].

Fungal pathogens are among the leading causes of infectious disease mortality, which is expected to accelerate due to a variety of factors including climate change [29]. Particularly concerning is the emergence of multidrug resistant yeast pathogens around the globe [30]. Mathematical models and synthetic gene networks (or 'circuits') are being used to experimentally investigate drug resistance in yeast [31]. In particular, synthetic gene circuits have been designed to mimic network motifs that occur naturally, such as positive feedback loops, to study non-genetic resistance in the budding yeast *Saccharomyces cerevisiae* [18, 32, 33]. This positive feedback has been shown to confer yeast cells with a heritable, non-genetically drug-resistant phenotype for up to 283 h before switching back to the drug-susceptible phenotype [32]. These experimental studies reveal important insights into fungal drug resistance and provide parameters for our quantitative models.

Selective pressures during infection can lead to cooperation and competition within microbial communities [34], and these interactions can have implications for disease outcomes [35]. Competition within a microbial community composed of the same species becomes relevant when resources such as nutrients or space become limiting, such as at high population density. Intraspecific competition results from 'exploitation competition', which involves the relatively more efficient use of a limiting resource or from 'interference competition', which results from the production of toxic substances that impair the growth or survival of competitors [36]. Intraspecific competition leads to logistic growth, whereby population growth is exponential when population size and resource competition are low, followed by a progressively reduced growth rate as the population size increases toward the carrying capacity of the micro-environment [37]. Phenotypic heterogeneity can promote interactions among subpopulations as well as the division of labour between individual cells, providing clonal microbial populations with new functionalities [38]. Importantly, the evolutionary effects of intraspecific competition have not been investigated in the context of resource competition between non-genetically and genetically resistant subpopulations in microbial populations undergoing drug treatment.

In this study, we investigate the transition from non-genetic to genetic resistance during static drug (drugs that stop or slow cell growth) and cidal drug (drugs that kill cells) treatment in the presence of resource competition using deterministic and stochastic population models [37]. Overall, we find that non-genetic resistance facilitates the survival of cell populations undergoing drug treatment, while hindering the fixation of genetic mutations due to competition effects between the non-genetically and genetically resistant subpopulations.

2. Methods

2.1. Deterministic population model

The deterministic population model describes changes in cellular subpopulation concentrations over time during drug treatment. Three different subpopulations comprising the total population T are described in this model: a susceptible subpopulation S, a non-genetically resistant subpopulation N, and a genetically resistant subpopulation G. Cells may switch between the S and N subpopulations, and cells in the N subpopulation can mutate into the G subpopulation (figure 1). The mathematical model is described by a set of coupled ordinary differential equations (ODEs):

$$\frac{dS}{dt} = Sk'_S + Nr_{S,N} - Sr_{N,S} - S\delta_S \quad (1)$$

$$\frac{dN}{dt} = Nk'_N + Sr_{N,S} - Nr_{G,N} - Nr_{S,N} - N\delta_N \quad (2)$$

$$\frac{dG}{dt} = Gk'_G + Nr_{G,N} - G\delta_G, \quad (3)$$

where $r_{S,N}$ is the switching rate from N to S, $r_{N,S}$ is the switching rate from S to N, $r_{G,N}$ is the mutation rate from N to G, and δ_S, δ_N, δ_G are the death rates of S, N, and G, respectively. k'_S, k'_N, and k'_G describe the birth rate of each subpopulation in the presence of a drug and resource competition, and are described by equation (6). There is no mutational pathway from S to G, as we are considering drugs that completely arrest growth and division ($k'_S = 0$) and therefore genetic mutation due to DNA replication errors in the S subpopulation does not occur [39]. Appendix D considers cases where S is allowed to grow ($k'_S \neq 0$), which allows mutation from S to G ($r_{G,S} \neq 0$), and shows that the qualitative trends seen in our main results hold for this scenario. Unless otherwise indicated, we assume that N has partial, temporary resistance (i.e., $0 < \delta_N < \delta_S$) and that genetic mutation provides complete, permanent resistance to the drug. To model static and cidal drug treatments of varying strengths, we respectively varied the birth and death rates in equations (1)–(3).

Summing equations (1)–(3) under the above assumptions yields the following equation for the concentration of the total population:

$$T = S + N + G \tag{4}$$

as well as an equation for the growth rate of the total population:

$$\frac{\mathrm{d}T}{\mathrm{d}t} = Sk'_S + Nk'_N + Gk'_G - S\delta_S - N\delta_N - G\delta_G. \tag{5}$$

Resource competition between the subpopulations was modeled by scaling k_S, k_N, and k_G by a Baranyi-Hill type function, which depends on T and results in logistic growth [37]. The Baranyi model accurately describes the transition from lag-phase to exponential growth that occurs during the adaptation to antimicrobial drugs [37, 40]. For subpopulation i (where, $i \in \{S, N, G\}$), this is given by:

$$k'_i = k_i z(T) = k_i \left(\frac{h^n}{h^n + T^n} \right), \tag{6}$$

where k_i is the maximum birth rate for subpopulation i (which leads to exponential growth in the absence of competition for limited resources), n is the Hill coefficient, and h is the point at which the competition function $z(T)$ is half of its maximum value.

The growth dynamics of S, N, and G were obtained by solving the deterministic model, starting from initial population sizes S_i, N_i, and G_i and numerically integrating equations (1)–(3) over a total time t_{tot} using a time step Δt. This numerical integration was performed using the *ode45* ODE solver, which is based on an explicit Runge–Kutta method, in MATLAB [41]. The fixation time τ_{fix} was used as a quantitative measure of how long it takes for G to become dominant in the population [42] and was defined as the time it takes for G to comprise 95% of the total population.

2.2. Stochastic population model

Next, we developed a stochastic population model corresponding to the deterministic population model to study the effects of non-genetic resistance on the evolution of genetic resistance in low cell number regimes. Low numbers of infectious cells can occur at the onset of infection and during the final stages of drug treatment, and is the regime where stochastic fluctuations are expected to have a significant effect on population dynamics. Accordingly, equations (1)–(3) were translated into the following set of reactions:

$$S \xrightarrow{k'_S} 2S \tag{7}$$

$$N \xrightarrow{k'_N} 2N \tag{8}$$

$$G \xrightarrow{k'_G} 2G \tag{9}$$

$$S \xrightarrow{r_{N,S}} N \tag{10}$$

$$N \xrightarrow{r_{S,N}} S \tag{11}$$

$$N \xrightarrow{r_{G,N}} G \tag{12}$$

$$S \xrightarrow{\delta_S} \varnothing \tag{13}$$

$$N \xrightarrow{\delta_N} \varnothing \tag{14}$$

$$G \xrightarrow{\delta_G} \varnothing \tag{15}$$

equations (7)–(15) were simulated using the Gillespie stochastic simulation algorithm [43, 44].

To quantify the effect of non-genetic drug resistance on the evolution of genetic drug resistance in the stochastic population model, we obtained the first-appearance time (P_τ) and fixation time ($P_{\tau_{\text{fix}}}$) distributions of G. For parameter regimes where population extinction could occur during the cidal drug treatment simulations (i.e., when S and N go extinct before G appears), we determined the effect that the death rate of N had on the probability of G emerging (P_G) before population extinction. This was determined from the number of population extinction events that occurred over a large number of simulations for different values of the death rate for N.

While the deterministic population model (equations (1)–(3)) was suitable for investigating large population dynamics under drug treatment, the corresponding stochastic population model (equations (7)–(15)) was necessary to accurately quantify fixation time and mutation first-appearance time distributions and extinction events for cidal drug treatment scenarios, where the total population size becomes small enough to result in extinction events (no drug resistance mutation appears before S and N reach zero). When starting with no pre-existing mutations ($G_i = 0$), the fixation times calculated using the deterministic model of cidal drug treatment tended to be underestimated compared to those calculated using averages over exact stochastic simulations. When starting with a pre-existing mutation ($G_i = 1$), the average fixation times using the stochastic model converged to the those found using the deterministic model (appendix H). This

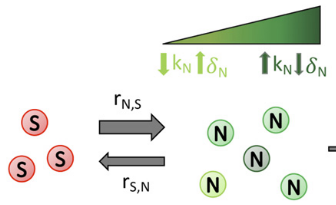

Figure 1. Schematic depicting the transitions between susceptible, non-genetically resistant, and genetically resistant subpopulations in a cell population undergoing drug treatment. Cells with the drug susceptible (S) phenotype can switch to non-genetically drug-resistant (N) phenotype and vice versa (at rates $r_{N,S}$ and $r_{S,N}$, respectively). The degree of susceptibility or resistance of S, N, and G to a drug can be varied between simulations and is dependent on the treatment scenario being investigated. For example, a higher birth rate (k_N) or a lower death rate (δ_N) results in the degree of transient drug resistance of N cells being higher (represented by dark green cells), whereas a lower k_N or a higher δ_N results in a lower level of transient drug resistance (represented by light green cells), as described overall by the fitness (N subpopulation growth rate) in the presence of a static or cidal drug, respectively; this is similar, but not shown in the schematic for clarity, for S and G cells. Cells from the N subpopulation can mutate (at a rate $r_{G,N}$) to become permanently genetically drug-resistant (G) cells.

highlights the importance of stochastic modeling when considering low numbers of infectious cells with no pre-existing drug resistance mutations.

We focus on the cidal drug treatment scenario for stochastic simulations for constant and fluctuating drug conditions, as the corresponding static drug stochastic simulations took prohibitively long to simulate due the lack of cell death, which resulted in larger subpopulation/population sizes, and correspondingly stochastic fluctuations were not expected to have much effect on the population dynamics.

3. Results and discussion

The parameters for the deterministic and stochastic population models are provided in appendix A (table A1) and the simulation codes are freely available at: https://github.com/CharleboisLab/S-N-G.

3.1. Deterministic population and evolutionary dynamics under static drug exposure

We began by numerically solving the deterministic population model to generate the time series subpopulation concentrations to investigate the relative fitness effects of the non-genetically and genetically resistant subpopulations on the evolution of drug resistance during static drug treatment. We model the effects of a static drug by setting the birth rate of the susceptible cells S to zero ($k_S = 0$) and setting all death rates to a natural basal death rate, which was based on the chronological life span of *S. cerevisiae* (table A1) [45].

The concentration of S initially decreases after the application of the static drug as a result of cells switching from S to N and then increases logistically due to switching from N to S, before falling toward extinction due to resource competition with the G subpopulation (figure 2(A)). The growth of N follows a logistic-type curve before also falling toward extinction (figure 2(B)). Overall, the growth of S and N (figures 2(A) and (B)), along with the growth of total population (figure 2(D)), increase as the fitness of N increases (modeled for static drug treatment by increasing the birth rate of N).

The concentration of G (figure 2(C)) and the fraction of G in the total population (figure 2(E)) reveal that an increase in the fitness of N slows the expansion of G. This can be attributed to resource competition between the subpopulations, as a higher total population size reduces the growth rate of G (equation (6)).

The growth rate of the total population over time increases before sharply decreasing after it reaches a maximum (figure 2(F)). This is a result of the growth of the population beginning to slow down as it increases in size ($\mathrm{d}T/\mathrm{d}t \to 0$ as $T \to \infty$), which is expected for logistic-type growth [37]. Despite the decrease in the expansion of G, the total population growth rate increases as the fitness of N increases (figure 2(F)). Thus, increasing the fitness of N in the static drug environment enhances the growth of the population, while at the same time hindering the expansion of G. Additionally, intense resource competition drives the S and N subpopulations to extinction at longer timescales.

Then, we quantified how the fitness of the N and G subpopulations affect the fixation of the mutated G subpopulation. As expected, an increase in k_G relative to k_N shortens the fixation time of G in the population (figure 3(A)). Importantly, increasing k_N relative to k_G lengthens the fixation time of G (figure 3(A)), due to competition decreasing the fraction of G in the total population (figure 2(E)).

Overall, the trends in the static drug environment were similar for a wide range of mutation rates (figure F5), though there were some qualitative differences for variations in the S–N switching rates (figures F1 and F2) (appendix F).

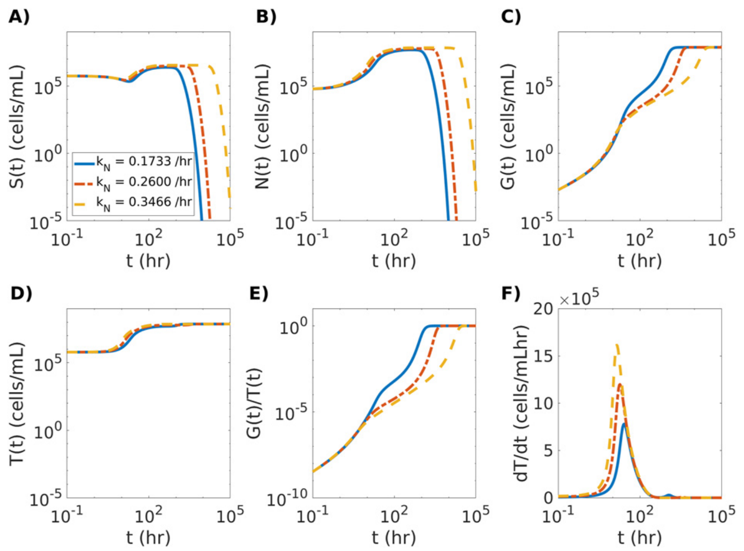

Figure 2. Growth of the genetically resistant subpopulation is hindered by an increase in birth rate of the non-genetically resistant subpopulation during static drug treatment. (A) The growth curve of the susceptible (S) subpopulation. (B) The growth curve of the non-genetically resistant (N) subpopulation. (C) The growth curve of the genetically resistant (G) subpopulation. (D) The growth curve of the total population (T). (E) The fraction of G in the total population (T). (F) The rate of change in the size of T ($\mathrm{d}T/\mathrm{d}t$) as a function of time (t). Each coloured line represents a different numerical simulation corresponding to the growth value of N (k_N) shown in the legend in (A), with the solid blue line representing the lowest level of N fitness ($k_N = 0.1733$ h^{-1}), the red dash-dotted line an intermediate level of N fitness ($k_N = 0.2600$ h^{-1}), and the yellow dashed line the highest level of N fitness ($k_N = 0.3466$ h^{-1}) relative to the fitness of G ($k_G = 0.3466$ h^{-1}) when exposed to a static drug.

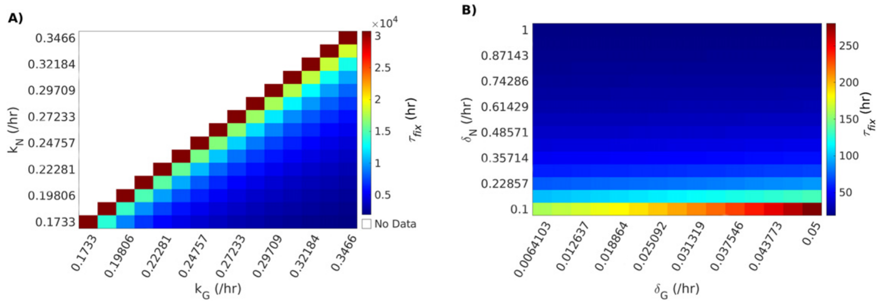

Figure 3. Drug resistance of the non-genetic subpopulation slows the evolution of the genetically resistant subpopulation during drug exposure. (A) Heat map shows the effect of the birth rates of the non-genetically resistant (N) and genetically resistant (G) subpopulations (k_N and k_G, respectively) on the fixation time (τ_{fix}) of the genetically resistant subpopulation (G) during static drug treatment. (B) Heat map shows the effect of the death rates of N and G (δ_N and δ_G, respectively) on the τ_{fix} of G during cidal drug treatment. For this case, we set δ_S was set to 1.0 h^{-1} and k_N and k_G to 0.3466 h^{-1}. Each bin in (A) and (B) corresponds to a simulation for a particular combination of k_N and k_G or δ_N and δ_G, respectively. The colour bar gives τ_{fix} in hours. As k_N is less than or equal to k_G, numerical simulation data does not appear in the upper diagonal of the heat map in (A).

3.2. Deterministic population and evolutionary dynamics under cidal drug exposure

Next, we investigated how the relative fitness of the non-genetically resistant and genetically resistant subpopulations affected the evolutionary dynamics of the population under cidal drug treatment.

The concentration of S quickly declines after exposure to the cidal drug, with phenotype switching from N providing temporary survival before dying off (figure 4(A)). The N subpopulation shows temporary growth for lower values of δ_N before dying off, while higher values of δ_N produce flat growth curves before going extinct due to drug treatment and subpopulation competition (figure 4(B)). Lower δ_N values (higher N fitness) prolong the temporary survival of S and N compared to higher δ_N values (figures 4(A) and (B)). The logistic growth of G also changes due to the fitness of N, with the higher δ_N (lower N fitness)

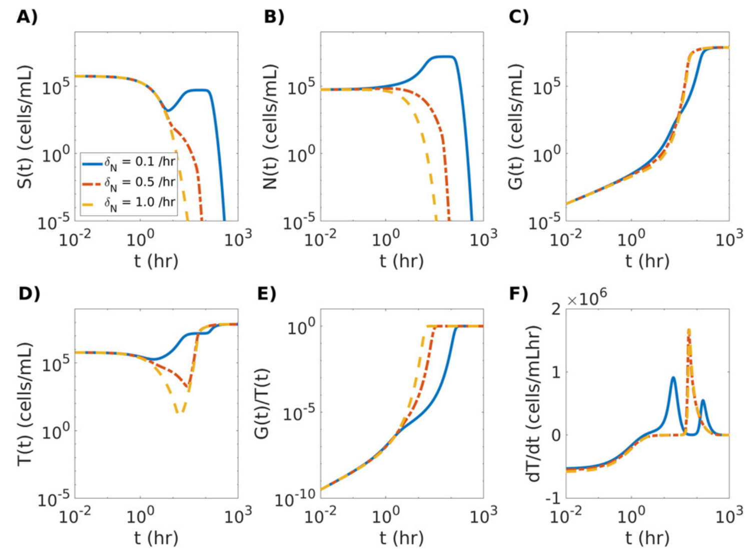

Figure 4. The fraction of the genetically resistant cells in the population increases with a decrease in fitness of the non-genetically resistant subpopulation in a cidal drug environment due to intraspecific competition. (A) The growth curve of the drug susceptible (S) subpopulation. (B) The growth curve of the non-genetically drug resistant (N) subpopulation. (C) The growth curve of the genetically drug resistant (G) subpopulation. (D) The growth curve of the total population (T). (E) The fraction of G in the total population (T). (F) The rate of change in the size of T ($\mathrm{d}T/\mathrm{d}t$) as a function of time (t). Each coloured line represents a different numerical simulation corresponding to the death rates of N shown in the legend in (A), with the solid blue line representing the highest level of N fitness (death rate of 0.1 h^{-1}), the red dash-dotted line an intermediate level of N fitness (death rate of 0.5 h^{-1}), and the yellow dashed line the lowest level N fitness (death rate of 1.0 h^{-1}) relative to the fitness of G (unaffected by the drug) during cidal drug treatment. S was given a death rate of 1.0 h^{-1} for these simulations, and the birth rate of both N and G was set to 0.3466 h^{-1}.

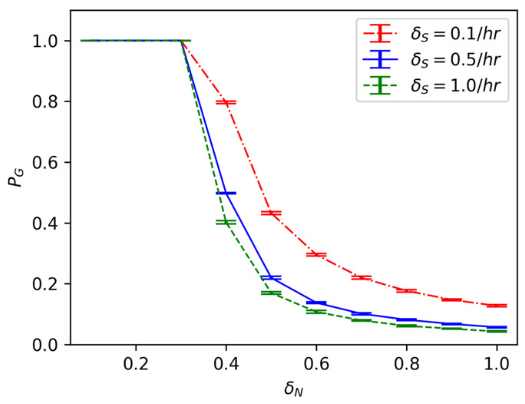

Figure 5. Probability of genetic drug resistance emerging before population extinction increases with the fitness of the non-genetically resistant subpopulation during cidal drug treatment. The appearance probability of the genetically drug resistant subpopulation (P_G) is shown as a function of decreasing N fitness (death rate; δ_N). Each coloured line represents a different strength of the cidal drug on the susceptible population S, with the red dashed-dotted line representing the lowest strength ($\delta_S = 0.1\ \mathrm{h}^{-1}$), the blue line an intermediate strength ($\delta_S = 0.5$), and the green dashed-dotted line the highest strength ($\delta_S = 1.0\ \mathrm{h}^{-1}$). The birth rates of N and G were set to $k_N = k_G = 0.3466\ \mathrm{h}^{-1}$. Each data point is an average over ten realizations of 10 000 simulations. Error bars show the standard deviation.

values producing a sharper increase toward saturation (figure 4(C)). Increasing the fitness of N decreases the fraction of G in the total population, as lower δ_N values result in more N cells and consequently lower G/T values (figure 4(E)).

The number of cells in the total population initially declines (negative growth rate; figure 4(D)) be fore stabilizing at zero population growth (figure 4(F)). Then the population growth rate increases as the genetically drug resistant G subpopulation

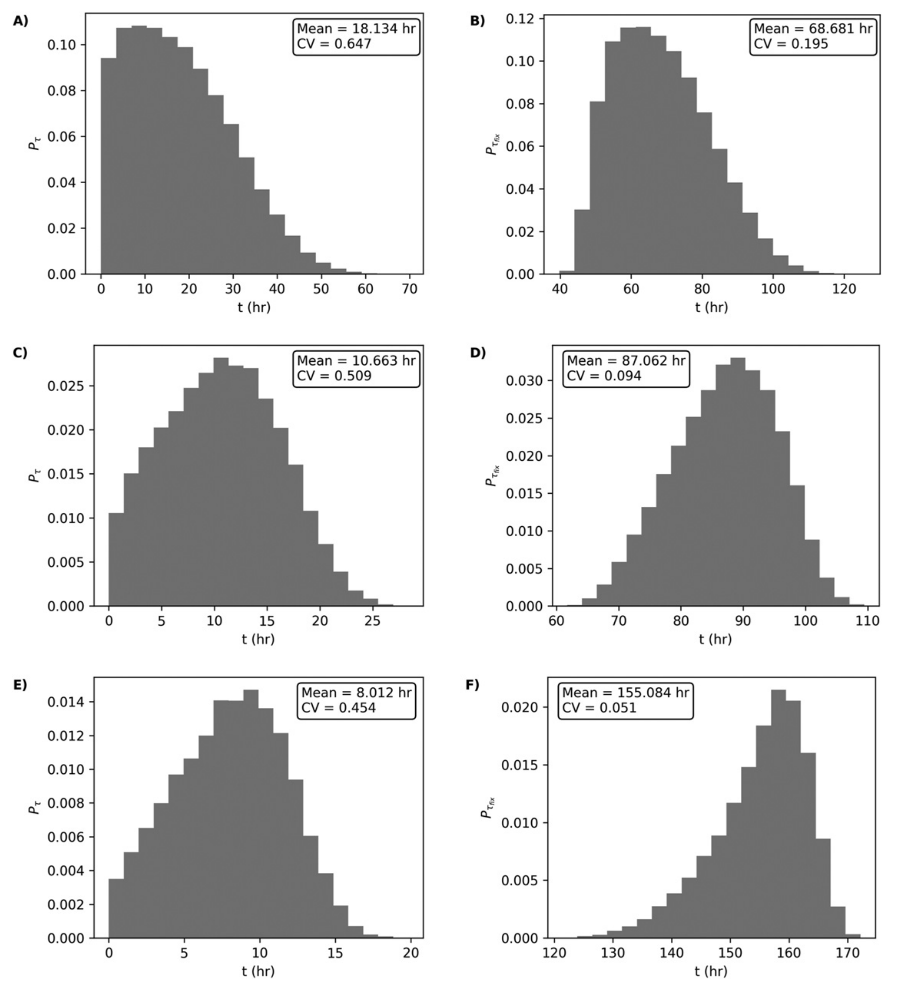

Figure 6. Drug resistance mutation first-appearance time and fixation time distributions during cidal drug treatment. (A) First-appearance time (P_τ) and (B) fixation time ($P_{\tau_{\mathrm{fix}}}$) distributions for the genetically resistant subpopulation (G) for a low level of non-genetically resistant subpopulation (N) fitness (death rate; $\delta_N = 0.3\ \mathrm{h}^{-1}$). For (A) and (B), histograms show results for 14 625 stochastic simulations. (C) P_τ and (D) $P_{\tau_{\mathrm{fix}}}$ distributions for G for an intermediate level of N fitness ($\delta_N = 0.2\ \mathrm{h}^{-1}$). For (C) and (D), histograms show results for 30 391 SSA simulations. (E) P_τ and (F) $P_{\tau_{\mathrm{fix}}}$ distributions for G for a high level of N fitness ($\delta_N = 0.1\ \mathrm{h}^{-1}$). For (E) and (F), histograms show results for 100 000 stochastic simulations. The mean and the CV for each distribution is provided in the top corners of each plot. The death rate of S was set to $\delta_S = 1.0\ \mathrm{h}^{-1}$ and the birth rates of N and G were set to $k_N = k_G = 0.3466\ \mathrm{h}^{-1}$ for these simulations.

expands to take over the population. This is followed by a decrease in the population growth rate, as the total population size moves toward saturation (zero population growth) after the maximum population growth rate is reached. Interestingly, when the fitness of N was high, there was a subsequent resurgence in the population growth rate before the population finally saturates. The first peak in the population growth rate is a due increasing S and N concentrations, and the second peak is due to a subsequent rising in G subpopulation concentration. When G is considered to be partially resistant (i.e. $\delta_G > 1/156\ \mathrm{h}^{-1}$), cidal drug treatment can either drive the total population to extinction or result in non-zero steady-state S, N, and G subpopulation concentrations (figure E1).

As for the static drug treatment, increasing the fitness of N hinders the fixation time of the genetically resistant subpopulation during cidal drug treatment (figure 3(B)). Similar population dynamics were observed when N was given a smaller birth rate ($k_N = 0.1733\ \mathrm{h}^{-1}$) compared to G (figure D1). Overall, the findings for cidal drug treatment were qualitatively similar for a wide range of parameters (figures C1, F3, F4, and F6) (appendix F).

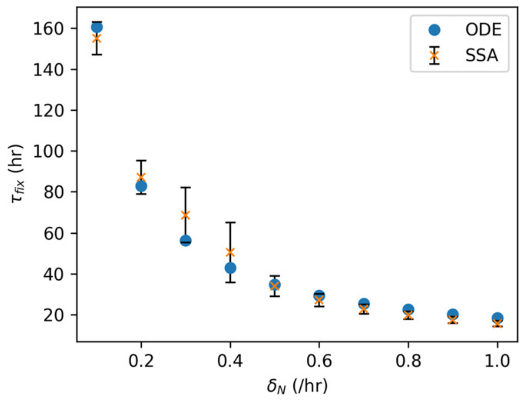

Figure 7. Fixation time of a genetic resistance mutation as a function of non-genetic drug resistance. Shown are the numerical solutions of the deterministic ODE model (blue dots) and SSA results (orange crosses) for the corresponding stochastic model. The death rate of the susceptible population (S) was set to $\delta_S = 1.0\ \mathrm{h}^{-1}$ and the birth rates of N and G were set to $k_N = k_G = 0.3466\ \mathrm{h}^{-1}$ for these simulations. Error bars on the SSA results denote the standard deviation.

3.3. Stochastic evolutionary dynamics during cidal drug treatment

We simulated the stochastic population model (equations (7)–(15)) translated from the deterministic population model (equations (1)–(3)) to investigate the transition from non-genetic to genetic drug resistance in cell populations moving toward extinction during cidal drug treatment. This is important as fluctuations in small subpopulation sizes may impact the evolutionary dynamics of the population. Given that S and N are killed to differing degrees by the cidal drug, and that the evolution of G depends directly on N, we hypothesized that stochastic fluctuations in the size of N could lead to the survival or extinction of the total population. To quantify the population and evolutionary drug resistance dynamics in this regime, we determined how the fitness of N affects the probability of population extinction (which occurs when S and N go extinct before G emerges; figure G1), along with the first-appearance and fixation times of a genetic mutation, which are bound to occur in our model once G is present in the population.

Decreasing the fitness of N (increasing δ_N) decreased the likelihood of G appearing and rescuing the cell population from extinction during cidal drug treatment (figure 5). This is in qualitative agreement with a previous study that found that increasing the fluctuation relaxation time of a drug resistance gene increased the probability of acquiring a drug resistance mutation [6]. When N had high fitness in the cidal drug environment ($\delta_N = 0.1\ \mathrm{h}^{-1}$ to $\delta_N = 0.3\ \mathrm{h}^{-1}$) the total population never went extinct. The extinction probability increased exponentially as the fitness of N decreased, with the population going extinct between 57% and 83% (depending on the value of δ_S) of the time for moderate fitness ($\delta_N = 0.5\ \mathrm{h}^{-1}$), and between 87% and 96% (depending on the value of δ_S) of the time for low fitness ($\delta_N = 1.0\ \mathrm{h}^{-1}$). These results show that the presence of non-genetic resistance enhances population survival when there are no pre-existing drug resistance mutations prior to drug exposure, and that non-genetic resistance increases the chance that a mutation will occur by providing a drug-exposed population with more time before extinction. As expected, G always appeared in the population when the strength of the cidal drug was low ($\delta_N < 0.3\ \mathrm{h}^{-1}$), and conversely, the population almost always went extinct before G could emerge when the strength of the cidal drug was high ($\delta_N = 1.0\ \mathrm{h}^{-1}$) (figure 5).

When the fitness of N increased in the cidal drug environment so did the means of the mutation first-appearance time and fixation time distributions (figure 6). This indicates that the presence of non-genetic drug resistance slows the evolution of genetic drug resistance, in agreement with the results obtained from the deterministic population model (figure 3(B)). While the coefficient of variation (CV; defined as the standard deviation divided by the mean) of the first-appearance time distributions (figures 6(A), (C) and (E)) was only marginally dependent on the fitness of N, the CV of the fixation time distributions (figures 6(B), (D) and (F)) increased approximately three fold as the fitness of N increased from low to high. Therefore, the presence of increased non-genetic drug resistance is predicted to not only slow down genetic drug resistance, but also to increase the uncertainty in its evolution.

A comparison of the fixation times found using the deterministic ODE model and the mean fixation times calculated over many stochastic simulation algorithm (SSA) simulations for various δ_N values

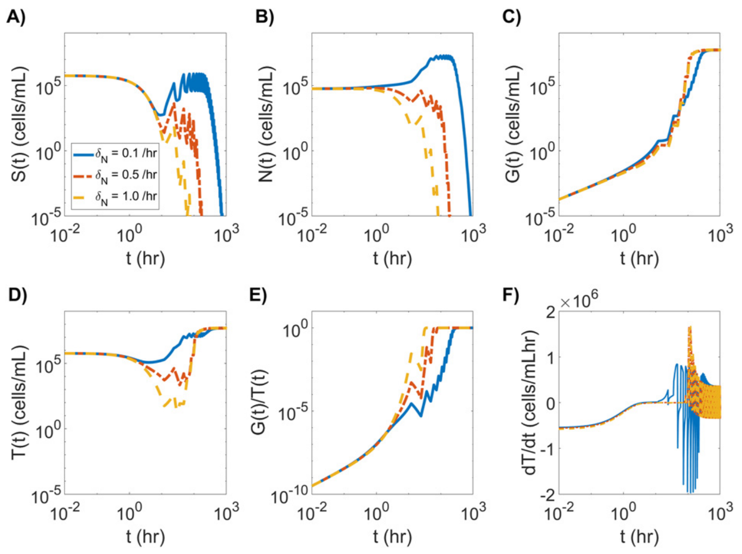

Figure 8. Oscillatory subpopulation concentration and growth rate dynamics in alternating drug-no drug conditions. The fraction of the genetically resistant cells in the population increases with the death rate of the non-genetically resistant subpopulation in a fluctuating cidal drug environment (12 h alternating drug-no drug intervals, starting with the drug applied). (A) The growth curve of the drug susceptible (S) subpopulation. (B) The growth curve of the non-genetically drug resistant (N) subpopulation. (C) The growth curve of the genetically drug resistant (G) subpopulation. (D) The growth curve of the total population (T). (E) The fraction of G in the total population (T). (F) The rate of change in the size of T ($\mathrm{d}T/\mathrm{d}t$) as a function of time (t). Each coloured line represents a different numerical simulation corresponding to the death rates of N shown in the legend in (A), with the solid blue line representing the highest level of N fitness (death rate of 0.1 h^{-1}), the red dash-dotted line an intermediate level of N fitness (death rate of 0.5 h^{-1}), and the yellow dashed line the lowest level N fitness (death rate of 1.0 h^{-1}) relative to the fitness of G (unaffected by the drug) during cidal drug treatment. S was given a death rate of 1.0 h^{-1} for these simulations, and the birth rate of both N and G was set to 0.3466 h^{-1}.

is shown in figure 7. As expected, the mean fixation times calculated from the stochastic simulations generally match those found using the deterministic model. When modeling a pre-existing mutation ($G_i = 1$), which removes the stochasticity in the first appearance time of the mutant subpopulation, the mean values of the stochastic simulation results converge with those found using the deterministic model (figure H1). Importantly, both deterministic and stochastic models show that decreasing the fitness of N increases the speed of genetic fixation (this also holds for pre-existing mutation scenarios; figure H1).

3.4. Evolutionary dynamics during fluctuating cidal drug treatment

To investigate evolutionary drug resistance dynamics in fluctuating drug treatment scenarios, we assigned resistant subpopulations a fitness cost when the drug was removed. This was done by reducing k_N to 0.2600 h^{-1} and k_G to 0.1733 h^{-1}, while allowing S to grow with $k_S = 0.3466$ h^{-1} in the no-drug environment (which also opened the mutational pathway from S to G at a rate $r_{G,S}$); the parameters were the same as before in the intervals where the cidal drug was applied (table A1).

We first performed deterministic simulations where drug application intervals ranged from 6 to 48 h, followed by no-drug intervals of the same duration. The fraction of G in the total population changes relative to the fitness of N in a similar way as the constant drug environment, showing a hindrance of the fixation of G with increasing N fitness (figures 8 and 9). Alternating between drug and no-drug environment resulted in oscillations in the subpopulation concentrations and population growth rate (figure 8). Interestingly, when N had a high level of resistance to cidal drug treatment, the population growth rate oscillated dramatically before saturating, rather than the isolated population growth rate surge and resurgence peaks that occurred prior to saturation in the constant cidal drug environment simulations (figure 8(F)). Increasing the period of the drug–no drug fluctuations also lengthens the fixation time scales of G (figure 9) compared to the constant cidal drug treatment scenario (figure 3(B)). Thus, fluctuating the drug condition, along with the presence of non-genetic drug resistance, can length the onset of permanent genetic drug resistance.

Next, we performed stochastic simulations to determine the probability of genetic drug resistance appearing before population extinction for 12, 24, and 48 h fluctuations in cidal drug treatment.

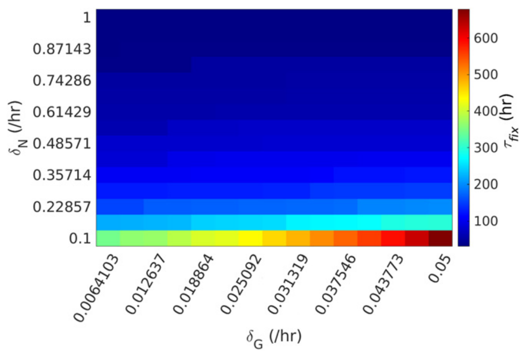

Figure 9. Drug resistance of the non-genetic subpopulation slows the evolution of the genetically drug resistant subpopulation during fluctuating cidal drug exposure. Heat map shows the effect of the death rates δ_N and δ_G on the τ_{fix} of G during cidal drug treatment for 12 h drug–no drug fluctuations time, starting with drug applied. δ_S was set to 1.0 h^{-1}, as it was assumed that S would experience the least amount of resistance to the drug compared to N and G, and the trends seen here were found to hold for fluctuation times of 6–48 h (data not shown). Each bin corresponds to a simulation for a particular combination of δ_N and δ_G. The colour bar gives τ_{fix} in hours.

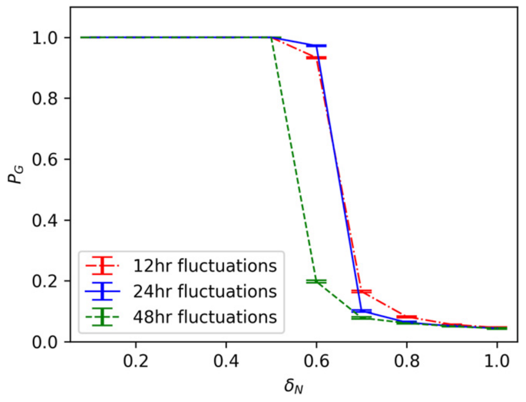

Figure 10. Probability of genetic drug resistance emerging before population extinction decreases at longer cidal drug–no cidal drug fluctuation intervals. The appearance probability of the genetically drug resistant subpopulation (P_G) is shown as a function of N fitness (death rate; δ_N). Each coloured line represents a different fluctuation time, with the red dashed-dotted line representing 12 h fluctuations, the blue line 24 h fluctuations, and the green dashed-dotted line 48 h fluctuations (each starting with drug applied). Each data point is an average over ten realizations of 10 000 simulations. Error bars show the standard deviation.

Importantly, increasing the fluctuation timescale of the drug to 48 h decreases the probability of genetic drug resistance emerging at intermediate levels of δ_N, compared to 12 and 24 h drug–no drug fluctuation intervals (figure 10). As in the constant cidal drug scenario, decreasing the fitness of the non-genetically resistant subpopulation lowers the chance of G appearing before population extinction.

Overall, these results suggests that alternating drug conditions can result in oscillatory population dynamics, and that increasing the drug–no drug timescales can increase resistance mutation fixation times and decrease population survival.

4. Conclusion

We found using deterministic and stochastic population models that while non-genetic resistance enhances population survival, a slower rate of genetic resistance evolution emerges from resource competition between these subpopulations during constant and fluctuating drug treatments. Specifically, increasing the fitness of the non-genetically resistant subpopulation (which allows the population to survive initial drug exposure) exponentially increased the chance of a genetically resistant subpopulation appearing and rescuing the total population from extinction during treatment at intermediate cidal drug strength. However, increasing the fitness of the non-genetically resistant subpopulation, along with fluctuating the drug condition, slowed down the fixation time of the genetic drug resistance mutation due to subpopulation competition effects, when no pre-existing mutations were present in the population. Incorporating pre-existing mutations into the model reduced the timescale of fixation, as it removed the growth delay resulting from the time taken by non-genetically resistant cells to mutate into genetically resistant cells. For the stochastic simulations, the presence of pre-existing mutations reduced the stochasticity of the first-appearance time of the genetically resistant subpopulation and protected against population extinction. Corresponding experimental investigations could be performed using microbes harbouring inducible synthetic gene circuits to control the fraction of non-genetically resistant cells in the population in combination with DNA sequencing to track the appearance time and frequency of drug resistance mutations [31–33].

High levels of competition drove the competing susceptible and non-genetically resistant subpopulations extinct in static and cidal drug treatment scenarios, which opens the possibility of incorporating competition and resource limitation strategies into antimicrobial therapies. These predictions could be tested experimentally, for instance through competition assays [46] in which synthetic gene circuits tune the initial fractions of susceptible and non-genetically drug resistant cells in the population [31].

Alternating drug–no drug conditions generated oscillatory population dynamics, and increasing the drug–no drug fluctuation timescale resulted in lengthened resistance mutation fixation times and a sharper population survival-extinction ‘phase transition’. It will be important to investigate the effects of non-genetic resistance on the development of genetic resistance in more complex cell models [47] and further in the context of fluctuating environments, which may be governed by environment-sensing genetic networks [48], along with cellular trade-offs that may occur in drug environments [49]. Microfluidic devices could be used to experimentally study the effects of fluctuating drug stress at the single-cell level [50, 51]. Fluctuating environmental stressors have been shown to facilitate ‘bet-hedging’ in cell populations [52, 53], whereby some cells adopt a non-growing, stress-resistant phenotype to increase the long-term fitness of the population. This could be modelled using stochastic hybrid processes [54], for instance by using an stochastic ON–OFF switch coupled to a system of ODEs describing subpopulation dynamics in the presence of a drug. Furthermore, the first-appearance time, fixation time, and extinction events could be described analytically in future studies using a first-passage time framework [6, 55].

Overall, our quantitative model generated robust and novel predictions on the evolution of drug resistance, and revealed that the interplay between transient non-genetic drug resistance and permanent genetic resistance may be more complex than previously thought. Specifically, in addition to enhancing the survival of a drug-exposed microbial population in constant drug conditions [4, 5, 10, 16, 24], our findings demonstrate that transient non-genetic resistance may hinder the evolution of permanent genetic resistance in constant and fluctuating drug conditions. As drug exposure is generally a form of selective pressure, the results of this study are also anticipated to be useful for understanding the evolutionary dynamics of other stress-resistant microbial populations.

A complete understanding of the drug resistance process, including the interplay between non-genetic and genetic forms of drug resistance, will be important for mitigating the socio-economic costs of antimicrobial resistance [5]. The resistance mutation appearance probabilities and first-appearance time distributions determined using stochastic population models may prove useful for guiding drug therapies. Quantitative distributions such as these may someday serve as a way to avoid drug failure resulting from the transition from non-genetic to genetic resistance by indicating the timescale at which a clinician should substitute or combine drugs during treatment to avoid the selection of resistance-conferring mutations [56, 57]. Fluctuations in mutation first-appearance times are also important, as they can be the difference between the eradication or the establishment of drug-resistant infection during drug therapy. Finally, drug resistant infections may one day be overcome by novel strategies that enhance competition between non-genetically and genetically resistant pathogens during treatment. One potential strategy is to revert an infectious population of cells from being drug-resistant to drug-sensitive by periodically fluctuating the drug environment. Specifically, by temporarily removing or substituting the drug, the fitness costs associated with subpopulation competition considered in our study, along with the previously established costs of non-genetic resistance [9, 32] and resistance mutations [58], could be exploited such that drug-susceptible cells dominate in the population. Overall, improving our quantitative understanding of how non-genetic and genetic mechanisms interact will advance our fundamental understanding of drug resistance evolution and may lead to more effective treatments for patients with drug-resistant infections.

Acknowledgements

DC was financially supported by the Government of Canada's New Frontiers in Research Fund—Exploration program (2019-01208), a Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant (RGPIN-2020-04007), an NSERC Discovery Launch Supplement (DGECR-2020-00197), and the University of Alberta. JG was supported by a 2021 NSERC Undergraduate Student Research Award.

Author contributions

DC conceived, designed, and supervised the research. DC developed the mathematical models. JG carried out the numerical and stochastic simulations. JG and DC analysed the data. DC and JG wrote the article.

Conflict of interest

The authors have no conflicts of interest to declare.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://github.com/CharleboisLab/S-N-G.

Appendix A. Parameters

The parameter values used in our study (table A1) were based on experimental studies on the budding yeast *S. cerevisiae* [32, 33, 45, 59, 60]. Additionally, some parameters were scanned over a range of values to account for parameter uncertainty, to probe for parameter regime specific model behaviour, and to test the robustness of the simulation results.

Drug susceptible cells S formed the majority of the total initial population (T_i) and non-genetically drug resistant cells N formed 1%–10% of T_i [19]. We assumed there were no pre-existing genetic drug resistance mutations ($G_i = 0$) for all numerical and stochastic simulations in the main text. Numerical simulation of the deterministic population started with $S_i = 5.5 \times 10^5$ cells ml^{-1} and $N_i = 5.5 \times 10^4$ cells ml^{-1}, which are common initial concentrations for 'log-phase' laboratory experiments [60].

Depending on the concentration of the drug being considered, constant birth rates k_N and k_G (which model the fitness of the given subpopulation in the presence of a static drug where death rates are unaffected) were assigned values between 0.1733 h^{-1} and 0.3466 h^{-1}, based on birth rates measured in standard yeast cell culture experiments [59]. For cidal drugs, we modeled fitness by setting the birth rates k_N and k_G equal and scanning over N and G death rates (δ_N and δ_G) at a constant S death rate (δ_S); the trends also held for lower k_N and δ_S values (see appendix C and appendix D). The parameter ranges in table A1 were the basis of parameter scans that were used to predict how the relative fitness between N and G will affect population dynamics and the evolution of genetic drug resistance in our study.

To model partial drug resistance due to gene-expression noise, it was assumed that $k_N \leqslant k_G$ for static drugs and $\delta_G \leqslant \delta_N$ for cidal drugs, with genetic drug resistance mutations providing the greatest level of resistance. Genetic mutations were also assumed to be permanent in our simulations.

The switching rates between S and N are based on experimental estimates [32] and ranged between $r_{S,N} = 0.0035$ h^{-1} and $r_{N,S} = 0.0625$ h^{-1}. The mutation rate from N to G was based on a previous modeling-experimental study [19], which ranged from 10^{-6} to 10^{-7} per cell division, and was assigned a value of $r_{G,N} = 0.3 \times 10^{-6}$ h^{-1} in our simulations.

The deterministic and stochastic population models provide a means to simulate the effects of non-genetic resistance on the evolution of drug resistance. The effects of static drugs (which reduce cell growth but do not kill cells) were modeled by arresting the birth of S ($k_S = 0$ and $r_{G,S} = 0$) and varying the birth rates of N and G (k_N and k_G, respectively) and by setting the death rates to a low natural basal death rate ($\delta_S = \delta_N = \delta_G = 1/156$ h^{-1}). Cidal drugs (which eventually kill all non-genetically resistant cells) were modeled by adding non-zero death rates for S and N (δ_S and δ_N, respectively). Unless otherwise stated, we also modeled partial drug resistance of N by setting $\delta_N < \delta_S$. Overall, varying key parameters specified in table A1 did not yield qualitatively different results.

The parameter values given in table A1 were also used for the stochastic simulations, as all the corresponding reactions in the stochastic population model were of zeroth-order or first-order [37]. Note that in the stochastic population model the values of S_i, N_i, and G_i are exact numbers of cells ($S_i = 5.5 \times 10^5$ cells and $N_i = 5.5 \times 10^4$ cells), which we used as the initial conditions to investigate the stochastic transition from non-genetic to genetic resistance as the number of cells in the population approached zero (extinction) due to cidal drug treatment. The S to N (and vice-versa) phenotype switching rates ($r_{N,S}$ and $r_{S,N}$, respectively), the N to G mutation rate ($r_{G,N}$), and the birth and death rates (k_i and δ_i, respectively, where $i \in \{S, N, G\}$) are all given as probability per unit time in the stochastic population model.

Table A1. Parameters used for numerically simulating the deterministic population dynamics model in static and cidal drug conditions. A horizontal line in the 'cidal drug value' column indicates that the value is the same as the corresponding value in the 'static drug value' column. A blank entry in the 'units' column indicates no units and 'see text' in the 'reference' indicates that the justification for the parameter value is given in the main text or appendices.

Parameter	Static drug value	Cidal drug value	Units	Reference
S_i	5.5×10^3 to 5.5×10^8	—	Cells ml^{-1}	[60]
N_i	5.5×10^2 to 1.1×10^8	—	Cells ml^{-1}	[9, 19]
G_i	0	—	Cells ml^{-1}	[39]
k_S	0	—	h^{-1}	[39]
k_N	0.1733–0.3466	—	h^{-1}	[59]
k_G	0.1733–0.3466	—	h^{-1}	[59]
$r_{S,N}$	0.0035	—	h^{-1}	[32, 33]
$r_{N,S}$	0.0625	—	h^{-1}	[32, 33]
$r_{G,S}$	0	—	h^{-1}	[39]
$r_{G,N}$	$10^{-6}/3$	—	h^{-1}	[19]
δ_S	1/156	0.1–1.0	h^{-1}	[45]
δ_N	1/156	0.1–1.0	h^{-1}	[45]
δ_G	1/156	1/156–0.05	h^{-1}	[45]
h	1×10^7	—	Cells ml^{-1}	See text
n	2	—		[59]

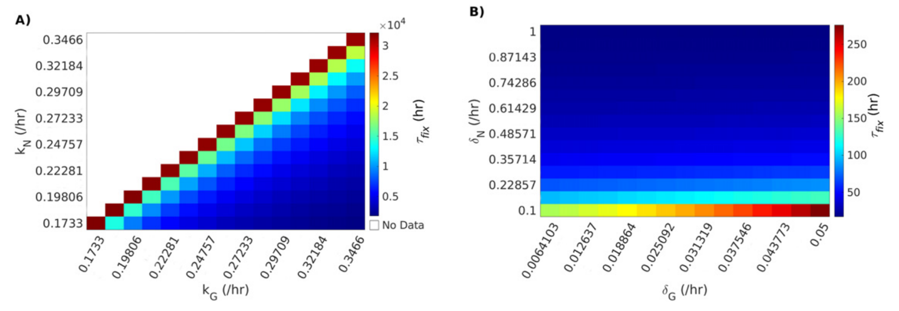

Figure B1. Simulations with a small S birth rate of $k_S = 0.01$ h^{-1} and a mutational pathway from S to G (with a mutation rate $r_{G,S}$ equal to $r_{G,N}$ used previously). (A) Fixation time heatmap for static drug case. (B) Fixation time heatmap for cidal drug case (with $\delta_S = 1.0$ h^{-1}).

Appendix B. Small S birth rate and mutational pathway from S to G

To test the scenario where drugs hinder but do not completely arrest growth of the susceptible cells S, we ran simulations with a small birth rate k_S. We also considered an active mutational pathway from S to G (with the mutation rate $r_{G,S}$ being equal to the $r_{G,N}$ mutation rate used in the main text). For all values of k_S tested (ranging from 1% to 50% the growth rates used in the main text), the main conclusion that non-genetic resistance hinders the fixation of resistance mutations held for static and cidal drug scenarios. A representative case for $k_S = 0.01$ h^{-1} is provided for the static drug case in figure B1(A) and for the cidal case in figure B1(B). Specifically, there were small quantitative differences in fixation times, but the qualitative relationship between fixation time and k_N or δ_N held.

Appendix C. Different cidal drug strengths

Figure C1 demonstrates that the qualitative results discussed in the main text hold for higher levels of S fitness (death rate $\delta_S = 0.5$ h^{-1} and $\delta_S = 0.1$ h^{-1}) in the cidal drug environment. Figure C1(A) shows the cidal drug fixation heatmap using the deterministic ODE model for an intermediate S fitness of $\delta_S = 0.5$ h^{-1} and figure C1(B) shows the same case for a high S fitness of $\delta_S = 0.1$ h^{-1}. The finding that the fitness of the S subpopulation had little effect on the fixation of G in the cidal drug environment can likely be attributed to the fact that $k_S = 0$ makes the

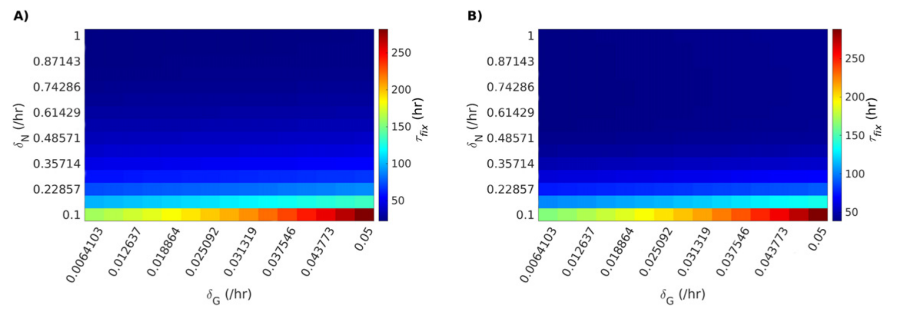

Figure C1. The results in the main text regarding N fitness and G fixation are shown to hold for higher levels of S fitness than those considered in the cidal drug environment in the main text ($\delta_S = 1.0\ \mathrm{h}^{-1}$; figure 3(B)). (A) Fixation time heatmap for intermediate S fitness ($\delta_S = 0.5\ \mathrm{h}^{-1}$). (B) Fixation time heatmap for higher S fitness ($\delta_S = 0.1\ \mathrm{h}^{-1}$).

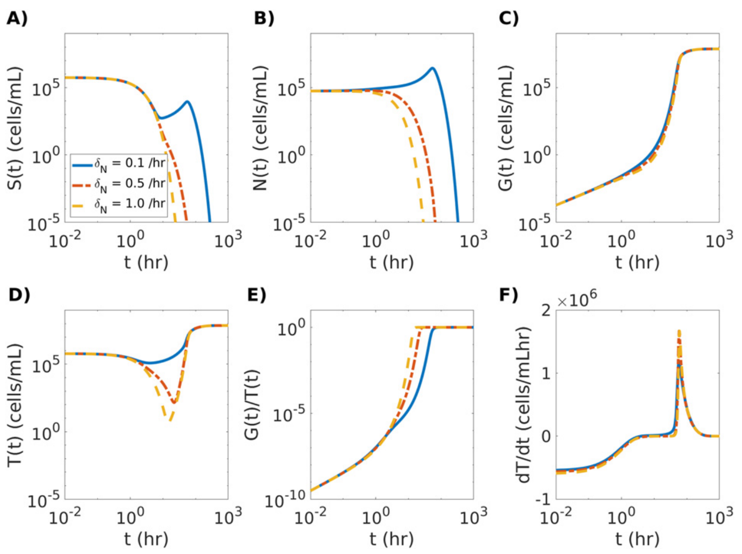

Figure D1. Growth of the genetically resistant subpopulation is hindered by the growth of the non-genetically resistant subpopulation with a low birth rate in cidal drug environment. (A) The growth curve of the drug susceptible (S) subpopulation. (B) The growth curve of the non-genetically drug resistant (N) subpopulation. (C) The growth curve of the genetically drug resistant (G) subpopulation. (D) The growth curve of the total population (T). (E) The fraction of $G(t)$ in the total population T. (F) The rate of change in the size of T ($\mathrm{d}T/\mathrm{d}t$) as a function of time. Each coloured line represents a different numerical simulation corresponding to the death rates of N shown in the legend in (A), with the solid blue line representing the highest level of N fitness (death rate of $0.1\ \mathrm{h}^{-1}$), the red dash-dotted line an intermediate level of N fitness (death rate of $0.5\ \mathrm{h}^{-1}$), and the yellow dashed line the lowest level N fitness (death rate of $1.0\ \mathrm{h}^{-1}$) relative to the fitness of G (unaffected by the drug) during cidal drug treatment. S was given a death rate of $\delta_S = 1.0\ \mathrm{h}^{-1}$ for these simulations, and the birth rate of G was set to $k_N = 0.3466\ \mathrm{h}^{-1}$.

S population significantly less fit relative to N and G subpopulations regardless of the cidal drug strength.

Appendix D. Cidal drug simulations with low N birth rate

Here we consider a cidal drug scenario where N has a low birth rate relative to G by setting $k_N = 0.1733\ \mathrm{h}^{-1}$, while k_G remained at $0.3466\ \mathrm{h}^{-1}$. Subpopulation trajectories, the fraction of genetically drug resistant cells in the population G/T, and the population rate of change $\mathrm{d}T/\mathrm{d}t$ for this case are shown in figure D1. A fixation time heat map for this case is shown in figure D2. The qualitative trends and conclusions made in the main text hold for this scenario.

Appendix E. Partially drug-susceptible genetic mutant

Figure E1 shows a cidal drug case where G is assigned a death rate of $\delta_G = 0.5\ \mathrm{h}^{-1}$ and hence is not fully resistant to the drug. The time series show three

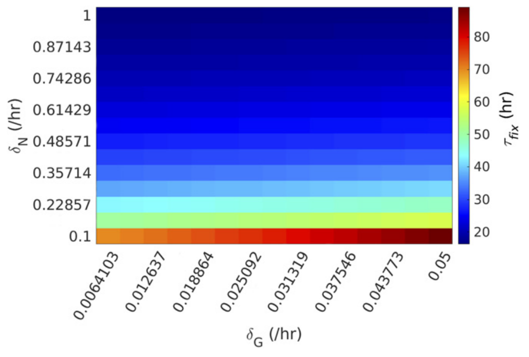

Figure D2. Drug resistance of the non-genetic subpopulation slows the development of the genetically drug resistant subpopulation in a cidal drug environment with N having a low birth rate relative to G ($k_N = 0.1733$ h^{-1}). Heat map shows the effect of the death rates of the non-genetically resistant (δ_N) and genetically resistant (δ_G) subpopulations on the fixation time (τ_{fix}) of G under cidal drug treatment. Each bin corresponds to a simulation for a combination of δ_N and δ_G parameter values. The colour map show the fixation time in hours.

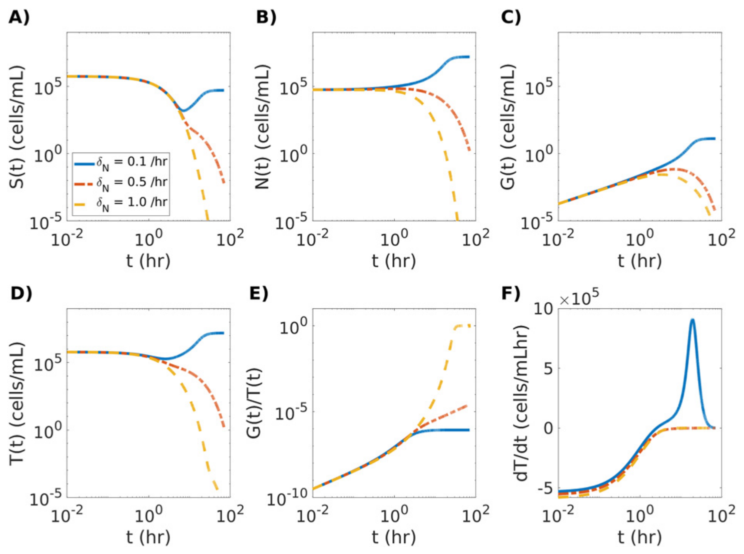

Figure E1. Partially resistant genetic mutation results in either total population extinction or non-zero steady-state population size during cidal drug treatment ($\delta_G = 0.5$ h^{-1}). (A) The growth curve of the drug susceptible (S) subpopulation. (B) The growth curve of the non-genetically drug resistant (N) subpopulation. (C) The growth curve of the genetically partially drug resistant (G) subpopulation. (D) The growth curve of the total population (T). (E) The fraction of $G(t)$ in T. (F) The rate of change in the size of T ($\mathrm{d}T/\mathrm{d}t$) as a function of time. Each coloured line represents a different numerical simulation using the death rate values shown in the legend in (A), with the solid blue line representing the highest level of N fitness ($\delta_N = 0.1$ h^{-1}), the red dash-dotted line an intermediate level of N fitness ($\delta_N = 0.5$ h^{-1}), and the yellow dashed line the lowest level of N fitness ($\delta_N = 1.0$ h^{-1}) relative to the intermediate fitness level of G ($\delta_G = 0.5$ h^{-1}).

different cases. The first case is where N has a relatively high-fitness death rate, resulting in the total population moving toward a non-zero steady state over time (solid blue lines in figure E1). The second case captures a scenario where N and G have the same intermediate fitness death rate, resulting in extinction of the total population (dash-dotted red lines in figure E1). The last case shows a situation where N has a low fitness death rate that is greater than the death rate of G, which also results in total population extinction (dashed yellow lines in figure E1). Consistent with our main results, this case shows that the fitness of N hinders the evolution of G, even when G is partly susceptible to the drug.

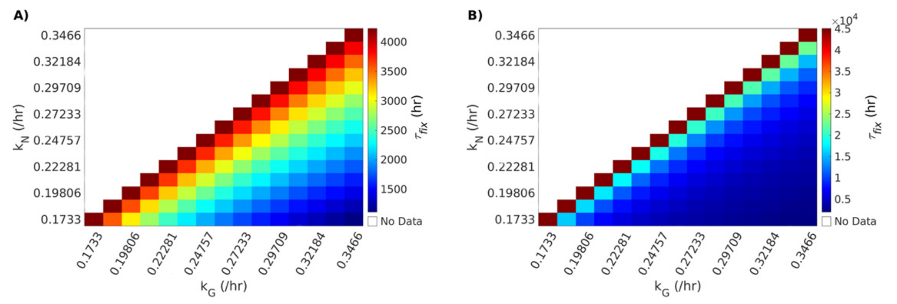

Figure F1. Genetic fixation results in the static drug environment for different values of switching rate $r_{N,S}$. (A) $r_{N,S} = 0.001\ \text{h}^{-1}$. (B) $r_{N,S} = 0.1\ \text{h}^{-1}$.

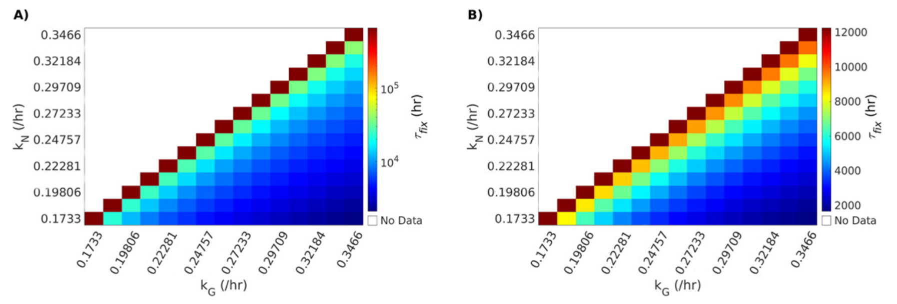

Figure F2. Genetic fixation results in the static drug environment for different values of switching rate $r_{S,N}$. (A) $r_{S,N} = 0.0001\ \text{h}^{-1}$. Due to the large range of values (ranging from 2080 h to over 10^6 hr) this color map was plotted on a log scale. (B) $r_{S,N} = 0.01\ \text{h}^{-1}$.

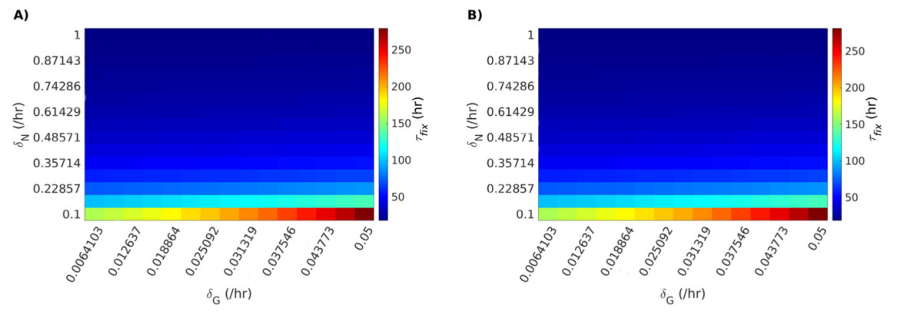

Figure F3. Genetic fixation results in the cidal drug environment for different values of switching rate $r_{N,S}$. (A) $r_{N,S} = 0.001\ \text{h}^{-1}$. (B) $r_{N,S} = 0.1\ \text{h}^{-1}$.

Appendix F. Parameter scans of switching and mutation rates

To further test the robustness of our main findings, we performed order-of-magnitude parameter scans of the switching rates $r_{N,S}$ and $r_{S,N}$ (figures F1–F4), as well as the mutation rate $r_{G,N}$ (figures F5 and F6), for the static and cidal drug scenarios.

Specifically, when the switching rate from S to N ($r_{N,S}$) was decreased by an order of magnitude it reduced the timescale (i.e., slightly decreased the lower bound of τ_{fix} and drastically decreased the upper bound of τ_{fix}) over which G fixed in the population (figure F1(A)); (2) when the switching rate from N to S ($r_{S,N}$) was decreased by an order of magnitude it increased the timescale (i.e., slightly decreased the lower bound of τ_{fix} and drastically increased the upper bound of τ_{fix}) over which G evolved in the population (figure F2(A)); and (3) when the switching rate from N to S ($r_{S,N}$) was increased by an order

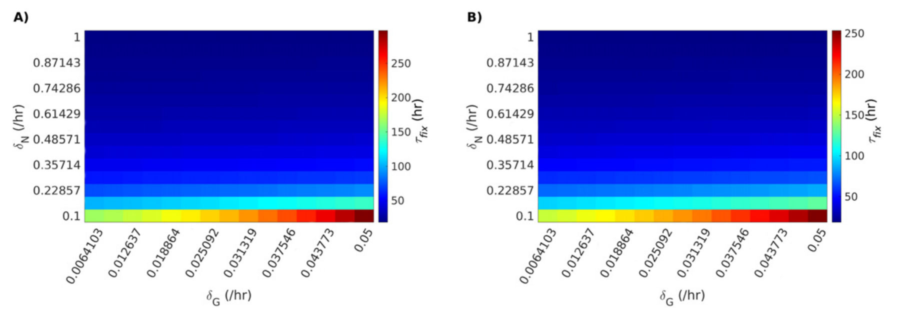

Figure F4. Genetic fixation results in the cidal drug environment for different values of switching rate $r_{S,N}$. (A) $r_{S,N} = 0.0001\ \mathrm{h}^{-1}$. (B) $r_{S,N} = 0.01\ \mathrm{h}^{-1}$.

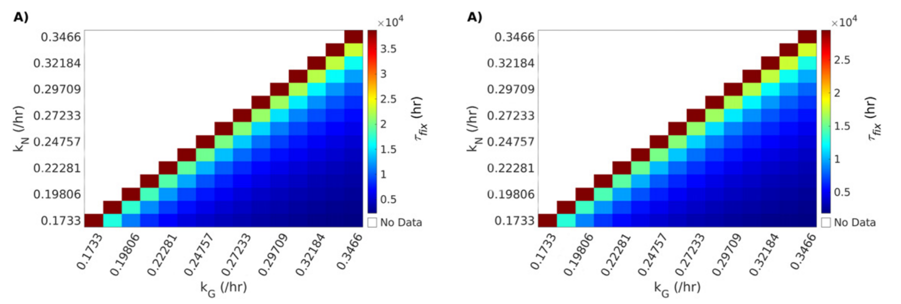

Figure F5. Genetic fixation results in the static drug environment for different values of mutation rate $r_{G,N}$. (A) $r_{G,N} = 10^{-7}/4$ h (B) $r_{G,N} = 10^{-6}/2$ h.

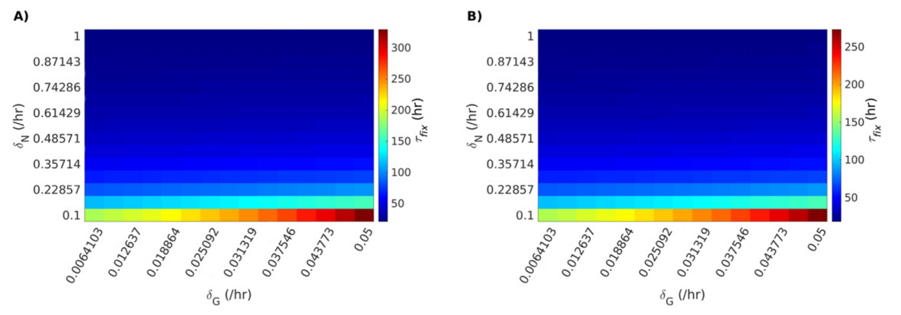

Figure F6. Genetic fixation results in the cidal drug environment for different values of mutation rate $r_{G,N}$. (A) $r_{G,N} = 10^{-7}/4$ h (B) $r_{G,N} = 10^{-6}/2$ h.

of magnitude, it decreased the timescale (i.e., slightly decreased the lower bound of τ_{fix} and drastically decreased the upper bound of τ_{fix}) over which G fixed in the population (figure F2(B)). One might expect that any change that leads to an increase in N would also increase G (because N mutates to G) and therefore would decrease the fixation of time of G, however, this was not the case and the evolution of G depended on the interactions between the competing subpopulations.

There were no important differences were observed for parameter scans of the N to G mutation rate (figures F5 and F6).

Overall, these results demonstrate that the conclusions made in the main text hold for a wide range of parameters, though changing the switching rates

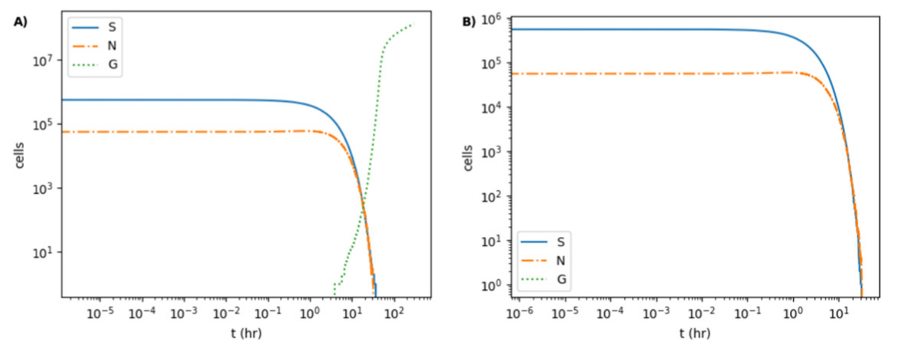

Figure G1. Cell population survival and extinction during cidal drug treatment. (A) A representative population non-extinction case where G appears before S and N go extinct. (B) A representative population extinction case where S and N go extinct before G appears in the population.

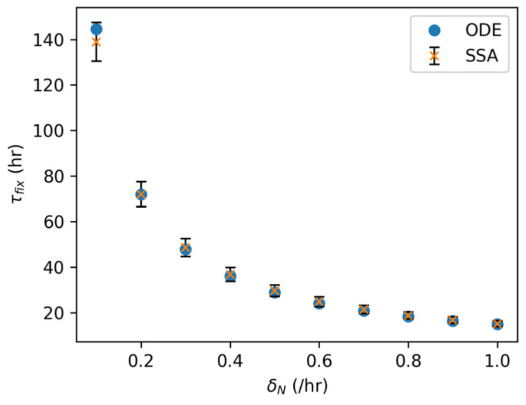

Figure H1. Genetic subpopulation G fixation time comparison for deterministic ODE and SSA simulations with a pre-existing mutation ($G_i = 1$). Genetic fixation time τ_{fix} is shown as a function of non-genetic subpopulation N fitness (δ_N) in the cidal drug environment. The death rate of the susceptible population S was set to $\delta_S = 1.0\ \text{h}^{-1}$ and the birth rates of N and G were set to $k_N = k_G = 0.3466\ \text{h}^{-1}$ for these simulations. Error bars on the SSA results show the standard deviation.

qualitatively affects the evolution of drug-resistant mutants in a way that is not entirely obvious and that depends on the underlying population dynamics.

Appendix G. Survival and extinction scenarios

Representative survival and extinction trajectories from the stochastic simulations are shown in figure G1. Figure G1(A) shows a case where the cell population survives cidal drug treatment due to G emerging before S and N go extinct. G subsequently takes over the population in this scenario. In contrast, figure G1(B) shows a case where the cell population goes extinct due to S and N reaching zero cells before G appears.

Appendix H. Pre-existing mutation for stochastic simulations

To show that the discrepancies seen in figure 7 were due to differences between the modeling approaches, i.e., SSA models extinction due to stochasticity in τ_{app}, simulations using a pre-existing mutation ($G_i = 1$) were made. Figure H1 shows the comparison of genetic fixation times calculated using simulations from the deterministic ODE and SSA modeling approaches, similar to figure 7 in the main text. As seen, removing the possibility of extinction by removing the stochastic nature of the first appearance time of G results in the SSA results converging to those found using the deterministic ODE model.

Appendix I. Varying initial population sizes

Starting the deterministic simulations with a low or high initial population size ($T_i \approx 10^3$ or $T_i \approx 10^8$) or with varying amounts of initial non-genetically resistant cells (N_i making up 1%–20% of T_i) were found to have little effect on the fixation time of the genetically resistant population, as the population dynamics in these cases quickly converged to similar trajectories regardless of their initial population sizes. For the stochastic simulations the initial population size was found to aid survival, with high initial populations surviving the drug for longer amounts of time and hence providing more time for the mutant population to emerge, as expected, but the fixation times of the genetically resistant population were once again not strongly affected by varying T_i or N_i.

ORCID iDs

Joshua D Guthrie https://orcid.org/0000-0002-5160-5363
Daniel A Charlebois https://orcid.org/0000-0001-7426-1789

References

[1] O'Neill J 2016 The review on antimicrobial resistance *Tackling Drug-Resistant Infections Globally: Final Report and Recommendations* (London: Wellcome Trust)
[2] Salmond G P and Welch M 2008 Antibiotic resistance: adaptive evolution *Lancet* **372** S97–103
[3] Fraser D and Kaern M 2009 A chance at survival: gene expression noise and phenotypic diversification strategies *Mol. Microbiol.* **71** 1333–40
[4] Levin B R and Rozen D E 2006 Non-inherited antibiotic resistance *Nat. Rev. Microbiol.* **4** 556–62
[5] Farquhar K S, Rasouli Koohi S and Charlebois D A 2021 Does transcriptional heterogeneity facilitate the development of genetic drug resistance? *BioEssays* **43** 2100043
[6] Charlebois D A, Abdennur N and Kaern M 2011 Gene expression noise facilitates adaptation and drug resistance independently of mutation *Phys. Rev. Lett.* **107** 218101
[7] Munsky B, Neuert G and van Oudenaarden A 2012 Using gene expression noise to understand gene regulation *Science* **336** 183–7
[8] Samoilov M S, Price G and Arkin A P 2006 From fluctuations to phenotypes: the physiology of noise *Nat. Rev. Genet.* **2006** re17
[9] Blake W J, Balázsi G, Kohanski M A, Isaacs F J, Murphy K F, Kuang Y, Cantor C R, Walt D R and Collins J J 2006 Phenotypic consequences of promoter-mediated transcriptional noise *Mol. Cell* **24** 853–65
[10] Brock A, Chang H and Huang S 2009 Non-genetic heterogeneity—a mutation-independent driving force for the somatic evolution of tumours *Nat. Rev. Genet.* **10** 336–42
[11] Kærn M, Elston T C, Blake W J and Collins J J 2005 Stochasticity in gene expression: from theories to phenotypes *Nat. Rev. Genet.* **6** 451–64
[12] Berman J and Krysan D J 2020 Drug resistance and tolerance in fungi *Nat. Rev. Microbiol.* **18** 319–31
[13] Gefen O and Balaban N Q 2009 The importance of being persistent: heterogeneity of bacterial populations under antibiotic stress *FEMS Microbiol. Rev.* **33** 704–17
[14] Wakamoto Y, Dhar N, Chait R, Schneider K, Signorino-Gelo F, Leibler S and McKinney J D 2013 Dynamic persistence of antibiotic-stressed mycobacteria *Science* **339** 91–5
[15] Vallette F M, Olivier C, Lezot F, Oliver L, Cochonneau D, Lalier L, Cartron P-F and Heymann D 2019 Dormant, quiescent, tolerant and persister cells: four synonyms for the same target in cancer *Biochem. Pharmacol.* **162** 169–76
[16] Levin-Reisman I, Ronin I, Gefen O, Braniss I, Shoresh N and Balaban N Q 2017 Antibiotic tolerance facilitates the evolution of resistance *Science* **355** 826–30
[17] Charlebois D A, Balazsi G and Kaern M 2014 Coherent feedforward transcriptional regulatory motifs enhance drug resistance *Phys. Rev.* E **89** 052708
[18] Camellato B, Roney I J, Azizi A, Charlebois D and Kaern M 2019 Engineered gene networks enable non-genetic drug resistance and enhanced cellular robustness *Eng. Biol.* **3** 72–9
[19] Farquhar K S, Charlebois D A, Szenk M, Cohen J, Nevozhay D and Balázsi G 2019 Role of network-mediated stochasticity in mammalian drug resistance *Nat. Commun.* **10** 2766
[20] Bar-Even A, Paulsson J, Maheshri N, Carmi M, O'Shea E, Pilpel Y and Barkai N 2006 Noise in protein expression scales with natural protein abundance *Nat. Genet.* **38** 636–43
[21] Newman J R S, Ghaemmaghami S, Ihmels J, Breslow D K, Noble M, DeRisi J L and Weissman J S 2006 Single-cell proteomic analysis of *S. cerevisiae* reveals the architecture of biological noise *Nature* **441** 840–6
[22] Liu J, François J-M and Capp J-P 2019 Gene expression noise produces cell-to-cell heterogeneity in eukaryotic homologous recombination rate *Front. Genet.* **10** 475
[23] Uphoff S, Lord N D, Okumus B, Potvin-Trottier L, Sherratt D J and Paulsson J 2016 Stochastic activation of a DNA damage response causes cell-to-cell mutation rate variation *Science* **351** 1094–7
[24] Bódi Z *et al* 2017 Phenotypic heterogeneity promotes adaptive evolution *PLoS Biol.* **15** e2000644
[25] Zhang Z, Qian W and Zhang J 2009 Positive selection for elevated gene expression noise in yeast *Mol. Syst. Biol.* **5** 299
[26] You S-T and Leu J-Y 2020 Making sense of noise *Evolutionary Biology—A Transdisciplinary Approach* ed P Pontarotti (Cham: Springer) pp 379–91
[27] Freddolino P L, Yang J, Momen-Roknabadi A and Tavazoie S 2018 Stochastic tuning of gene expression enables cellular adaptation in the absence of pre-existing regulatory circuitry *eLife* **7** e31867
[28] Ghalambor C K, Hoke K L, Ruell E W, Fischer E K, Reznick D N and Hughes K A 2015 Non-adaptive plasticity potentiates rapid adaptive evolution of gene expression in nature *Nature* **525** 372–5
[29] van Rhijn N and Bromley M 2021 The consequences of our changing environment on life threatening and debilitating fungal diseases in humans *J. Fungi* **7** 367
[30] Sears D and Schwartz B S 2017 *Candida auris*: an emerging multidrug-resistant pathogen *Int. J. Infect. Dis.* **63** 95–8
[31] Farquhar K S, Flohr H and Charlebois D A 2020 Advancing antimicrobial resistance research through quantitative modeling and synthetic biology *Front. Bioeng. Biotechnol.* **8** 583415
[32] Nevozhay D, Adams R M, Itallie E V, Bennett M R and Balazsi G 2012 Mapping the environmental fitness landscape of a synthetic gene circuit *PLoS Comput. Biol.* **8** e1002480
[33] González C, Ray J C J, Manhart M, Adams R M, Nevozhay D, Morozov A V and Balázsi G 2015 Stress-response balance drives the evolution of a network module and its host genome *Mol. Syst. Biol.* **11** 827

[34] Baishya J and Wakeman C A 2019 Selective pressures during chronic infection drive microbial competition and cooperation *npj Biofilms Microbiome* **5** 16
[35] Baishya J, Bisht K, Rimbey J N, Yihunie K D, Islam S, Al Mahmud H, Waller J E and Wakeman C A 2021 The impact of intraspecies and interspecies bacterial interactions on disease outcome *Pathogens* **10** 96
[36] Hibbing M E, Fuqua C, Parsek M R and Peterson S B 2010 Bacterial competition: surviving and thriving in the microbial jungle *Nat. Rev. Microbiol.* **8** 15–25
[37] Charlebois D A and Balázsi G 2019 Modeling cell population dynamics *Silico Biol.* **13** 21–39
[38] Ackermann M 2015 A functional perspective on phenotypic heterogeneity in microorganisms *Nat. Rev. Microbiol.* **13** 497–508
[39] Bell G and MacLean C 2018 The search for 'evolution-proof' antibiotics *Trends Microbiol.* **26** 471–83
[40] Fridman O, Goldberg A, Ronin I, Shoresh N and Balaban N Q 2014 Optimization of lag time underlies antibiotic tolerance in evolved bacterial populations *Nature* **513** 418–21
[41] (2021) MATLAB Version 9.10.0 (R2021a) (Natick, Massachusetts: The MathWorks Inc.)
[42] Klironomos F D, Berg J and Collins S 2013 How epigenetic mutations can affect genetic evolution: model and mechanism *BioEssays* **35** 571–8
[43] Gillespie D T 1976 A general method for numerically simulating the stochastic time evolution of coupled chemical reactions *J. Comput. Phys.* **22** 403–34
[44] Gillespie D T 1977 Exact stochastic simulation of coupled chemical reactions *J. Phys. Chem.* **81** 2340–61
[45] Longo V D, Shadel G S, Kaeberlein M and Kennedy B 2012 Replicative and chronological aging in *Saccharomyces cerevisiae Cell Metab.* **16** 18–31
[46] Ramia N E, Mangavel C, Gaiani C, Muller-Gueudin A, Taha S, Revol-Junelles A-M and Borges F 2020 Nested structure of intraspecific competition network in *Carnobacterium maltaromaticum Sci. Rep.* **10** 7335
[47] Karr J R, Sanghvi J C, Macklin D N, Gutschow M V, Jacobs J M, Bolival B Jr, Assad-Garcia N, Glass J I and Covert M W 2012 A whole-cell computational model predicts phenotype from genotype *Cell* **150** 389–401
[48] Ribeiro A 2008 Dynamics and evolution of stochastic bistable gene networks with sensing in fluctuating environments *Phys. Rev.* E **78** 061902
[49] Weiße A Y, Oyarzúnc D A, Danos V and Swain P S 2015 Mechanistic links between cellular trade-offs, gene expression, and growth *Proc. Natl. Acad. Sci. USA* **112** E1038
[50] Jo M C, Liu W, Gu L, Dang W and Qin L 2015 High-throughput analysis of yeast replicative aging using a microfluidic system *Proc. Natl Acad. Sci. USA* **112** 9364–9
[51] Charlebois D A, Hauser K, Marshall S and Balazsi G 2018 Multiscale effects of heating and cooling on genes and gene networks *Proc. Natl Acad. Sci. USA* **115** E10797
[52] Acar M, Mettetal J T and Van Oudenaarden A 2008 Stochastic switching as a survival strategy in fluctuating environments *Nat. Genet.* **40** 471–5
[53] Beaumont H J E, Gallie J, Kost C, Ferguson G C and Rainey P B 2009 Experimental evolution of bet hedging *Nature* **462** 90–3
[54] Singh A and Hespanha J P 2010 Stochastic hybrid systems for studying biochemical processes *Phil. Trans. R. Soc.* A **368** 4995–5011
[55] Ghusinga K R, Dennehy J J and Singh A 2017 First-passage time approach to controlling noise in the timing of intracellular events *Proc. Natl Acad. Sci. USA* **114** 693–8
[56] Baym M, Stone L K and Kishony R 2016 Multidrug evolutionary strategies to reverse antibiotic resistance *Science* **351** aad3292
[57] Dar R D, Hosmane N N, Arkin M R, Siliciano R F and Weinberger L S 2014 Screening for noise in gene expression identifies drug synergies *Science* **344** 1392–6
[58] Melnyk A H, Wong A and Kassen R 2015 The fitness costs of antibiotic resistance mutations *Evol. Appl.* **8** 273–83
[59] Alon U 2020 *An Introduction to Systems Biology: Design Principles of Biological Circuits* 2nd edn (Boca Raton, FL: CRC Press)
[60] Charlebois D A, Diao J, Nevozhay D and Balázsi G 2018 Negative regulation gene circuits for efflux pump control *Methods Mol. Biol.* **1772** 25–43

Chapter 3

Conclusion

"For objective description and harmonious comprehension, it is necessary in almost every field of knowledge to pay attention to the circumstances under which evidence is obtained."

-Niels Bohr

3.1 Physics-Based Modelling Provides Valuable Insights into the Evolution of Drug Resistance

In Chapter 2, a model used to predict the evolutionary dynamics of AMR in a population consisting of susceptible, genetically resistant, and non-genetically resistant cells was proposed. The model was presented as a set of deterministic coupled population growth equations and as a set of stochastic reactions. The deterministic growth equations were numerically solved to investigate the evolutionary dynamics of the population in large cell number regimes where stochastic effects are negligible. Stochastic trajectories of the reaction system were simulated using the SSA to analyze the model in low cell number regimes where stochastic effects can alter the evolutionary dynamics of the population. The stochastic formulation of the model was primarily used to investigate population survival and statistical distributions of genetic mutation appearance and fixation probabilities over time.

The analysis of this model led to multiple quantitative conclusions about the potential evolutionary dynamics of cell populations undergoing drug treatment:

1. The survival of a cell population undergoing drug treatment is aided by non-genetic phenotypic heterogeneity.

2. Increased survival due to non-genetic phenotypic heterogeneity increases the probability of a genetic drug resistance mutation appearing in the population.

3. Intraspecific competition effects between non-genetically resistant and genetically resistant cells lead to longer genetic fixation times within the population.

Conclusion 1 has been documented in the literature [10, 20, 50, 80, 81], showing that our model can provide quantitative support for empirical and qualitative biological evidence. Conclusion 2 is in agreement with previous studies suggesting that non-genetic phenotypic heterogeneity increases the probability of a beneficial genetic mutation arising in the population [10, 20]. Conclusion 3 is a novel prediction about the evolutionary dynamics of cell populations consisting of susceptible, non-genetically resistant, and genetically resistant cells, which compete with each other while undergoing drug treatment. It leads to an experimentally testable hypothesis that the survival of non-genetically resistant cells results in a subpopulation that can directly compete with genetically resistant cells, hindering the overall evolution of the genetic mutation towards fixation. This result has not previously been suggested in the literature, which has leaned towards the idea that non-genetic resistance is beneficial for the development of permanent genetic resistance [10, 20, 80].

The phenomenological model we have presented incorporates population-level characteristics that arise due to the underlying physical processes that occur during gene expression and result in non-genetic AMR. Additionally, the analysis was carried out using numerical and Monte Carlo simulation methods commonly used in physics [45]. The research presented in this thesis has shown that taking a physics-based approach to produce predictive quantitative models and investigate the complex dynamics of AMR can provide valuable insights by reinforcing qualitative biological reasoning and experimental findings, along with enabling the development of novel quantitative hypotheses that can be tested experimentally.

3.2 Limitations and Future Directions

The work presented in Chapter 2 is a computational study investigating the evolutionary dynamics of drug resistance at the cell population level and hence has several limitations that should be noted. Future research utilizing similar methods will involve addressing these limitations and testing model predictions experimentally to deepen our understanding of AMR.

Firstly, the model does not explicitly incorporate the underlying mechanisms of drug resistance in cells [56]. Additionally, the overall structure and parameters of the model were based on *in vitro* studies of AMR in yeast, which may not accurately represent the cell population dynamics of biological systems undergoing drug treatment in other settings [100]. A phenomenological approach was taken by modelling population-level characteristics that arise due to the mechanisms that we considered to be the most influential for cell population AMR evolution given the current state of understanding in the literature [20], with phenotypic differences in the population being modelled by three distinct subgroups. In particular, the population-level phenotypic characteristics generated by beneficial genetic mutations and non-genetic gene expression noise were incorporated into our model, as they are universal to cells [20]. The model may need to be extended to incorporate more phenotypic differences, such as including population-level characteristics that arise due to other genetic (e.g., horizontal gene transfer in bacteria [101]) and non-genetic (e.g., persistence in bacteria [77] and heteroresistance in fungi [56]) mechanisms, to capture cell population dynamics during drug-treatment more accurately. Antifungal tolerance, defined as the ability of a susceptible fungal strain to grow slowly in the presence of an antifungal drug [62], should also be incorporated to more robustly model the dynamics of the susceptible subpopulation. Other considerations, such as incorporating a spatial component [102] and modelling the fitness costs associated with mutations [103] and non-genetic phenotypic heterogeneity [76, 104] should be investigated and incorporated into the model to further advance our understanding of AMR evolution. Mechanistic approaches that explicitly incorporate mechanisms of AMR and cell-to-cell differences caused by them (e.g., individual cell size and functions) [105] should also be considered to further investigate the interplay between genetic and non-genetic AMR.

Secondly, the quantitative measure of evolution used in our study was limited to the genetic fixation time of the genetically resistant population. Although genetic fixation is a useful measure of evolution in cell populations, more in-depth evolutionary investigations, such as considering fitness landscapes [106, 107], allele frequencies [107], and frequency-dependent selection [108], could be applied within the context of the model we have proposed to make additional predictions about the evolution of AMR in cell populations. Other mathematical formulations of the proposed model could also be investigated to make additional predictions, such as applying a first-passage time framework to further investigate first-appearance and fixation times of the genetic mutations [75, 109], coupling the ODEs to stochastic hybrid processes [110] to better model cell population behaviour in fluctuating environments, and incorporating drug concentration equations [111] to more robustly model fluctuating drug levels.

Lastly, the work presented in this thesis is computational. The structure and parameters of the model were based on empirical observations of drug-resistant cell populations, but the conclusions we have drawn from the analysis of the model require experimental validation before they can be accepted as conclusive evidence for the evolutionary dynamics of AMR in cell populations. To test predictions made regarding population survival and the first appearance times of genetic mutations, synthetic biology methods could be used to control levels of non-genetically resistant cells in an experimental cell population using inducible synthetic gene circuits, with DNA sequencing being used to track the first appearance and frequency of genetic mutations in the population [10, 63, 76]. Our predictions regarding the effect of intraspecific competition between subpopulations could be tested using competition assays [112] while also utilizing synthetic gene circuits to tune the fraction of susceptible and non-genetically resistant cells [10]. Future work will require close collaboration between theoretical and experimental researchers to validate the quantitative predictions made by mathematical models of AMR and to guide the development of future models.

Along with providing a more robust understanding of AMR mechanisms and evolution, predictive, physics-based modelling may someday be used to aid medical treatments and drug discovery, reducing the medical, societal and economic impacts of AMR [10, 11, 20]. First appearance and genetic fixation distributions could potentially be used to guide treatment strategies by providing a quantitative framework for predicting the development of AMR in infections, allowing clinicians to determine when certain drugs should be substituted or combined to reduce the selection of resistance mutations [113, 114]. The effects of intraspecific competition [24, 25] and the fitness costs associated with non-genetic [76, 104] and genetic AMR [103] could be exploited to hinder or prevent the development of drug-resistant infections, potentially by periodically fluctuating the drug environment based on timescales suggested by AMR models to ensure that drug-susceptible cells remain dominant in the population. Predictive modelling may also aid in the development of "evolution-proof" antimicrobial drugs [27] by providing drug development researchers with tools to investigate AMR mechanisms and the evolutionary dynamics of AMR to optimize antimicrobial compounds that reduce, or eliminate, the possibility of genetic AMR mutations and evolution [11]. This further demonstrates that AMR research will require close collaboration between researchers in many different fields.

Based on the work presented in this thesis and elsewhere in the literature, I anticipate that quantitative modelling that incorporates aspects of physics methodology, along with utilizing physical theories and methods, will continue to be a valuable tool for understanding AMR at a fundamental level and to guide new developments towards reducing the global impacts of AMR. AMR is a complex problem that will require an interdisciplinary approach to better understand the mechanisms and consequences of resistance at various scales, from the molecular underpinnings of resistance to the large-scale behaviour of cell populations that cause drug-resistant infections and threaten global health security.

References

[1] J. D. Guthrie and D. A. Charlebois, “Non-genetic resistance facilitates survival while hindering the evolution of drug resistance due to intraspecific competition,” *Physical Biology*, vol. 19, no. 6, p. 066 002, 2022.

[2] S. A. Shankarnarayan, J. D. Guthrie, and D. A. Charlebois, “Machine learning for antimicrobial resistance research and drug development,” *The Global Antimicrobial Resistance Epidemic: Innovative Approaches and Cutting-Edge Solutions*, 2022.

[3] W. H. Organization *et al.*, *Antimicrobial resistance: global report on surveillance*. World Health Organization, 2014.

[4] J. O’Neill and The Review on Antimicrobial Resistance, *Tackling Drug-Resistant Infections Globally: Final Report and Recommendations*. London: Wellcome Trust, 2016.

[5] J. Powers, “Antimicrobial drug development–the past, the present, and the future,” *Clinical Microbiology and Infection*, vol. 10, pp. 23–31, 2004.

[6] N. R. Naylor *et al.*, “Estimating the burden of antimicrobial resistance: A systematic literature review,” *Antimicrobial Resistance & Infection Control*, vol. 7, pp. 1–17, 2018.

[7] T. Pulingam *et al.*, “Antimicrobial resistance: Prevalence, economic burden, mechanisms of resistance and strategies to overcome,” *European Journal of Pharmaceutical Sciences*, vol. 170, p. 106 103, 2022.

[8] K. D. McCubbin *et al.*, “Knowledge gaps in the understanding of antimicrobial resistance in canada,” *Frontiers in public health*, vol. 9, p. 726 484, 2021.

[9] D. A. Charlebois and M. Kærn, “What all the noise is about: The physical basis of cellular individuality,” *Canadian Journal of Physics*, vol. 90, no. 10, pp. 919–923, 2012.

[10] K. S. Farquhar, H. Flohr, and D. A. Charlebois, “Advancing antimicrobial resistance research through quantitative modeling and synthetic biology,” *Front. Bioeng. Biotechnol.*, vol. 8, p. 583 415, 2020.

[11] D. A. Charlebois, “Quantitative systems-based prediction of antimicrobial resistance evolution,” *NPJ Systems Biology and Applications*, vol. 9, no. 1, p. 40, 2023.

[12] H. Miller, Z. Zhou, J. Shepherd, A. Wollman, and M. Leake, “Single-molecule techniques in biophysics: A review of the progress in methods and applications,” *Reports on Progress in Physics*, vol. 81, Apr. 2017. DOI: 10.1088/1361-6633/aa8a02.

[13] C. M. Dobson, “Biophysical techniques in structural biology,” *Annual Review of Biochemistry*, vol. 88, no. 1, pp. 25–33, 2019. DOI: 10.1146/annurev-biochem-013118-111947.

[14] A. W. Serohijos and E. I. Shakhnovich, “Merging molecular mechanism and evolution: Theory and computation at the interface of biophysics and evolutionary population genetics,” *Current opinion in structural biology*, vol. 26, pp. 84–91, 2014.

[15] E. Y. Frisman, O. Zhdanova, and G. Neverova, “Ecological and genetic models in population biophysics,” *Biophysics*, vol. 65, pp. 810–825, 2020.

[16] M. Kaern, T. C. Elston, W. J. Blake, and J. J. Collins, “Stochasticity in gene expression: From theories to phenotypes,” *Nat. Rev. Genet.*, vol. 6, pp. 451–464, 2005.

[17] B. Munsky, G. Neuert, and A. van Oudenaarden, “Using gene expression noise to understand gene regulation,” *Science*, vol. 336, pp. 183–187, 2012.

[18] A. Sanchez, S. Choubey, and J. Kondev, “Regulation of noise in gene expression,” *Annual review of biophysics*, vol. 42, pp. 469–491, 2013.

[19] R. Allen and B. Waclaw, “Antibiotic resistance: A physicist’s view,” *Physical biology*, vol. 13, no. 4, p. 045 001, 2016.

[20] K. S. Farquhar, S. R. Koohi, and D. A. Charlebois, “Does transcriptional heterogeneity facilitate the development of genetic drug resistance?” *BioEssays*, vol. 43, p. 2 100 043, 2021.

[21] P Borst, “Genetic mechanisms of drug resistance: A review,” *Acta Oncologica*, vol. 30, no. 1, pp. 87–105, 1991.

[22] G. P. C. Salmond and M. Welch, “Antibiotic resistance: Adaptive evolution,” *The Lancet*, vol. 372, S97–S103, 2008.

[23] N. M. Revie, K. R. Iyer, N. Robbins, and L. E. Cowen, “Antifungal drug resistance: Evolution, mechanisms and impact,” *Current opinion in microbiology*, vol. 45, pp. 70–76, 2018.

[24] J. Baishya and C. A. Wakeman, “Selective pressures during chronic infection drive microbial competition and cooperation,” *npj Biofilms Microbiomes*, vol. 5, p. 16, 2019.

[25] J. Baishya *et al.*, “The impact of intraspecies and interspecies bacterial interactions on disease outcome,” *Pathogens*, vol. 10, p. 96, 2021.

[26] M. E. Hibbing, C. Fuqua, M. R. Parsek, and S. B. Peterson, “Bacterial competition: Surviving and thriving in the microbial jungle,” *Nat. Rev. Microbiol.*, vol. 8, pp. 15–25, 2010.

[27] G. Bell and C. MacLean, "The search for 'evolution-proof' antibiotics," *Trends Microbiol.*, vol. 26, pp. 471–483, 6 2018.

[28] D. Botstein, S. A. Chervitz, and M. Cherry, "Yeast as a model organism," *Science*, vol. 277, no. 5330, pp. 1259–1260, 1997.

[29] D. Botstein and G. R. Fink, "Yeast: An experimental organism for 21st century biology," *Genetics*, vol. 189, no. 3, pp. 695–704, 2011.

[30] J. Nielsen, "Yeast systems biology: Model organism and cell factory," *Biotechnology journal*, vol. 14, no. 9, p. 1 800 421, 2019.

[31] A. Huppert and G. Katriel, "Mathematical modelling and prediction in infectious disease epidemiology," *Clinical microbiology and infection*, vol. 19, no. 11, pp. 999–1005, 2013.

[32] B. Alberts *et al.*, *Molecular Biology of the Cell*, 7th. W. W. Norton & Company, 2021.

[33] F. Crick, "Central dogma of molecular biology," *Nature*, vol. 227, no. 5258, pp. 561–563, 1970.

[34] F. H. Crick, "On protein synthesis," in *Symp Soc Exp Biol*, vol. 12, 1958, p. 8.

[35] V. Shahrezaei and P. S. Swain, "Analytical distributions for stochastic gene expression," *Proceedings of the National Academy of Sciences*, vol. 105, no. 45, pp. 17 256–17 261, 2008.

[36] M. Mahner and M. Kary, "What exactly are genomes, genotypes and phenotypes? and what about phenomes?" *Journal of theoretical biology*, vol. 186, no. 1, pp. 55–63, 1997.

[37] P. N. Benfey and T. Mitchell-Olds, "From genotype to phenotype: Systems biology meets natural variation," *Science*, vol. 320, no. 5875, pp. 495–497, 2008.

[38] M. B. Elowitz, A. J. Levine, E. D. Siggia, and P. S. Swain, "Stochastic gene expression in a single cell," *Science*, vol. 297, no. 5584, pp. 1183–1186, 2002. DOI: 10.1126/science.1070919.

[39] P. S. Swain, M. B. Elowitz, and E. D. Siggia, "Intrinsic and extrinsic contributions to stochasticity in gene expression," *Proceedings of the National Academy of Sciences*, vol. 99, no. 20, pp. 12 795–12 800, 2002. DOI: 10.1073/pnas.162041399.

[40] L. Cai, N. Friedman, and X. S. Xie, "Stochastic protein expression in individual cells at the single molecule level," *Nature*, vol. 440, no. 7082, 358–362, 2006, ISSN: 0028-0836.

[41] R. Phillips, J. Kondev, J. Theriot, and H. Garcia, *Physical Biology of the Cell*, 2nd. Garland Science, 2012. DOI: 10.1201/9781134111589.

[42] D. Charlebois, "Computational investigations of noise-mediated cell population dynamics," Ph.D. dissertation, University of Ottawa, 2014.

[43] F. Horn and R. Jackson, "General mass action kinetics," *Arch. Rational Mech. Anal.*, vol. 47, no. 81-116, 1972. DOI: 10.1007/BF00251225.

[44] D. Charlebois, "An algorithm for the stochastic simulation of gene expression and cell population dynamics," M.S. thesis, University of Ottawa, 2010.

[45] D. T. Gillespie, "A general method for numerically simulating the stochastic time evolution of coupled chemical reactions," *J. Comput. Phys*, vol. 22, 403–434, 1976.

[46] D. T. Gillespie, "Exact stochastic simulation of coupled chemical reactions," *J. Phys. Chem.*, vol. 81, 2340–2361, 1977.

[47] D. T. Gillespie, "Stochastic chemical kinetics," in *Handbook of Materials Modeling: Methods*, Springer, 2005, pp. 1735–1752.

[48] M. Scott, B. Ingalls, and M. Kærn, "Estimations of intrinsic and extrinsic noise in models of nonlinear genetic networks," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 16, no. 2, 2006.

[49] P. Sjöberg, P. Lötstedt, and J. Elf, "Fokker–planck approximation of the master equation in molecular biology," *Computing and Visualization in Science*, vol. 12, pp. 37–50, 2009.

[50] D. Fraser and M. Kaern, "A chance at survival: Gene expression noise and phenotypic diversification strategies," *Molec. Microbiol.*, vol. 71, pp. 1333–1340, 2009.

[51] M. Thattai and A. Van Oudenaarden, "Intrinsic noise in gene regulatory networks," *Proceedings of the National Academy of Sciences*, vol. 98, no. 15, pp. 8614–8619, 2001.

[52] E. Gibney and C. Nolan, "Epigenetics and gene expression," *Heredity*, vol. 105, no. 1, pp. 4–13, 2010.

[53] *One Health: Fungal Pathogens of Humans, Animals, and Plants: Report on an American Academy of Microbiology Colloquium held in Washington, DC, on October 18, 2017.* American Society for Microbiology, 2019. DOI: 10.1128/AAMCol.18Oct.2017.

[54] M. I. Hutchings, A. W. Truman, and B. Wilkinson, "Antibiotics: Past, present and future," *Current Opinion in Microbiology*, vol. 51, pp. 72–80, 2019, Antimicrobials, ISSN: 1369-5274. DOI: 10.1016/j.mib.2019.10.008.

[55] J. S. Jiri Houst and V. Havlicek, "Antifungal drugs.," *Metabolites*, vol. 10, 2020. DOI: 10.3390/metabo10030106.

[56] Y. Lee, N. Robbins, and L. E. Cowen, "Molecular mechanisms governing antifungal drug resistance," *npj Antimicrobials and Resistance*, vol. 1, no. 1, p. 5, 2023.

[57] P. G. Pappas, M. S. L. an Maiken Cavling Arendrup, L. Ostrosky-Zeichner, and B. J. Kullberg, "Invasive candidiasis," *Nature Reviews Disease Primers*, vol. 4, no. 18026, 2018. DOI: 10.1038/nrdp.2018.26.

[58] S. R. Lockhart *et al.*, "Simultaneous Emergence of Multidrug-Resistant Candida auris on 3 Continents Confirmed by Whole-Genome Sequencing and Epidemiological Analyses," *Clinical Infectious Diseases*, vol. 64, no. 2, pp. 134–140, Dec. 2016, ISSN: 1058-4838. DOI: 10.1093/cid/ciw691.

[59] D. Sears and B. S. Schwartz, "*Candida auris*: An emerging multidrug-resistant pathogen," *Int. J. Infect. Dis.*, vol. 63, pp. 95–98, 2017.

[60] A. Chowdhary, C. Sharma, and J. F. Meis, "Candida auris: A rapidly emerging cause of hospital-acquired multidrug-resistant fungal infections globally," *PLOS Pathogens*, vol. 13, no. 5, pp. 1–10, May 2017. DOI: 10.1371/journal.ppat.1006290.

[61] K. Forsberg *et al.*, "Candida auris: The recent emergence of a multidrug-resistant fungal pathogen," *Medical Mycology*, vol. 57, no. 1, pp. 1–12, Jul. 2018, ISSN: 1369-3786. DOI: 10.1093/mmy/myy054.

[62] J. Berman and D. J. Krysan, "Drug resistance and tolerance in fungi," *Nat. Rev. Microbiol.*, vol. 18, pp. 319–331, 2020.

[63] C. Gonzalez *et al.*, "Stress-response balance drives the evolution of a network module and its host genome," *Mol. Syst. Biol.*, vol. 11, p. 827, 2015.

[64] D. A. Charlebois and G. Balazsi, "Modeling cell population dynamics," *In Silico Biol.*, vol. 13, pp. 21–39, 2019.

[65] J. Lin, K. Nishino, M. C. Roberts, M. Tolmasky, R. I. Aminov, and L. Zhang, *Mechanisms of antibiotic resistance*, 2015.

[66] R. Hershberg, "Mutation—the engine of evolution: Studying mutation and its role in the evolution of bacteria," *Cold Spring Harbor perspectives in biology*, vol. 7, no. 9, a018077, 2015.

[67] K. C. Smith, "Spontaneous mutagenesis: Experimental, genetic and other factors," *Mutation Research/Reviews in Genetic Toxicology*, vol. 277, no. 2, pp. 139–162, 1992.

[68] S. J. Balin and M. Cascalho, "The rate of mutation of a single gene," *Nucleic acids research*, vol. 38, no. 5, pp. 1575–1582, 2010.

[69] C. F. Baer, M. M. Miyamoto, and D. R. Denver, "Mutation rate variation in multicellular eukaryotes: Causes and consequences," *Nature Reviews Genetics*, vol. 8, no. 8, pp. 619–631, 2007.

[70] L. Loewe and W. G. Hill, "The population genetics of mutations: Good, bad and indifferent," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 365, no. 1544, pp. 1153–1167, 2010.

[71] K. S. Farquhar, D. A. Charlebois, M. Szenk, J. Cohen, D. Nevozhay, and G. Balazsi, "Role of network-mediated stochasticity in mammalian drug resistance," *Nat. Commun.*, vol. 10, p. 2766, 2019.

[72] V. Longo, G. Shadel, M. Kaeberlein, and B. Kennedy, "Replicative and chronological aging in *Saccharomyces cerevisiae*," *Cell Metabolism*, vol. 16, no. 1, pp. 18–31, 2012, ISSN: 1550-4131. DOI: https://doi.org/10.1016/j.cmet.2012.06.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1550413112002379.

[73] M. S. D'Arcy, "Cell death: A review of the major forms of apoptosis, necrosis and autophagy," *Cell Biology International*, vol. 43, no. 6, pp. 582–592, 2019. DOI: 10.1002/cbin.11137.

[74] M. S. Samoilov, G. Price, and A. P. Arkin, "From fluctuations to phenotypes: The physiology of noise," *Nat. Rev. Genet.*, vol. 2006, re17, 2006.

[75] D. A. Charlebois, N. Abdennur, and M. Kaern, "Gene expression noise facilitates adaptation and drug resistance independently of mutation," *Phys. Rev. Lett.*, vol. 107, p. 218 101, 2011.

[76] D. Nevozhay, R. M. Adams, E. V. Itallie, M. R. Bennett, and G. Balazsi, "Mapping the environmental fitness landscape of a synthetic gene circuit," *PLoS Comput. Biol.*, vol. 8, pp. 1–17, 2012.

[77] O. Gefen and N. Q. Balaban, "The importance of being persistent: Heterogeneity of bacterial populations under antibiotic stress," *FEMS Microbiol. Rev.*, vol. 33, pp. 704–717, 2009.

[78] Z. Patwa and L. M. Wahl, "The fixation probability of beneficial mutations," *Journal of The Royal Society Interface*, vol. 5, no. 28, pp. 1279–1289, 2008.

[79] F. D. Klironomos, J. Berg, and S. Collins, "How epigenetic mutations can affect genetic evolution: Model and mechanism," *BioEssays*, vol. 35, pp. 571–578, 2013.

[80] B. R. Levin and D. E. Rozen, "Non-inherited antibiotic resistance," *Nat. Rev. Microbiol.*, vol. 4, pp. 556–562, 2006.

[81] Z. Bodi *et al.*, "Phenotypic heterogeneity promotes adaptive evolution," *PLoS Biol.*, vol. 15, e2000644, 2017.

[82] A. Tsoularis and J. Wallace, "Analysis of logistic growth models," *Mathematical Biosciences*, vol. 179, no. 1, pp. 21–55, 2002, ISSN: 0025-5564. DOI: https://doi.org/10.1016/S0025-5564(02)00096-2. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0025556402000962.

[83] P. Verhulst, "Notice sur la loi que la population suit dans son accroissement.," *Correspondence Mathematique et Physique (Ghent)*, vol. 10, no. 113-121, 1838.

[84] J. Monod, "The growth of bacterial cultures," *Annual Review of Microbiology*, vol. 3, no. 1, pp. 371–394, 1949. DOI: 10.1146/annurev.mi.03.100149.002103.

[85] K. Korolev, J. Xavier, and J. Gore, "Turning ecology and evolution against cancer," *Nature reviews. Cancer*, vol. 14, Apr. 2014. DOI: 10.1038/nrc3712.

[86] F. Courchamp, T. Clutton-Brock, and B. Grenfell, "Inverse density dependence and the allee effect," *Trends in Ecology & Evolution*, vol. 14, no. 10, pp. 405–410, 1999, ISSN: 0169-5347. DOI: 10.1016/S0169-5347(99)01683-3.

[87] A. Kramer, B. Dennis, A. Liebhold, and J. Drake, “The evidence for allee effects,” *Population Ecology*, vol. 51, pp. 341–354, Jul. 2009. DOI: 10.1007/s10144-009-0152-6.

[88] J Baranyi, T. Roberts, and P McClure, “A non-autonomous differential equation to model bacterial growth,” *Food microbiology*, vol. 10, no. 1, pp. 43–59, 1993.

[89] F. Baty and M.-L. Delignette-Muller, “Estimating the bacterial lag time: Which model, which precision?” *International Journal of Food Microbiology*, vol. 91, no. 3, pp. 261–277, 2004, ISSN: 0168-1605. DOI: 10.1016/j.ijfoodmicro.2003.07.002.

[90] D. A. Charlebois, “Effect and evolution of gene expression noise on the fitness landscape,” *Physical Review E*, vol. 92, no. 2, p. 022 713, 2015.

[91] MATLAB, *version 9.10.0 (R2021a)*. Natick, Massachusetts: The MathWorks Inc., 2021.

[92] L. F. Shampine and M. W. Reichelt, “The matlab ode suite,” *SIAM Journal on Scientific Computing*, vol. 18, no. 1, pp. 1–22, 1997. DOI: 10.1137/S1064827594276424.

[93] J. R. Dormand and P. J. Prince, “A family of embedded runge-kutta formulae,” *Journal of Computational and Applied Mathematics*, vol. 6, pp. 19–26, 1980.

[94] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, 3rd ed. USA: Cambridge University Press, 2007, ISBN: 0521880688.

[95] D. A. Charlebois, J. Intosalmi, D. Fraser, and M. Kærn, “An algorithm for the stochastic simulation of gene expression and heterogeneous population dynamics,” *Communications in Computational Physics*, vol. 9, no. 1, pp. 89–112, 2011.

[96] D. A. Charlebois and M. Kærn, “An accelerated method for simulating population dynamics,” *Communications in Computational Physics*, vol. 14, no. 2, pp. 461–476, 2013.

[97] M. A. Gibson and J. Bruck, “Efficient exact stochastic simulation of chemical systems with many species and many channels,” *The journal of physical chemistry A*, vol. 104, no. 9, pp. 1876–1889, 2000.

[98] D. T. Gillespie, “Approximate accelerated stochastic simulation of chemically reacting systems,” *The Journal of chemical physics*, vol. 115, no. 4, pp. 1716–1733, 2001.

[99] D. T. Gillespie, “The chemical langevin equation,” *The Journal of Chemical Physics*, vol. 113, no. 1, pp. 297–306, 2000.

[100] M. N. Dudley and P. G. Ambrose, “Pharmacodynamics in the study of drug resistance and establishing in vitro susceptibility breakpoints: Ready for prime time,” *Current opinion in microbiology*, vol. 3, no. 5, pp. 515–521, 2000.

[101] H. Ochman, J. G. Lawrence, and E. A. Groisman, "Lateral gene transfer and the nature of bacterial innovation," *nature*, vol. 405, no. 6784, pp. 299–304, 2000.

[102] M. Baym *et al.*, "Spatiotemporal microbial evolution on antibiotic landscapes," *Science*, vol. 353, no. 6304, pp. 1147–1151, 2016.

[103] A. H. Melnyk, A. Wong, and R. Kassen, "The fitness costs of antibiotic resistance mutations," *Evol. Appl.*, vol. 8, pp. 273–283, 2015.

[104] W. J. Blake *et al.*, "Phenotypic consequences of promoter-mediated transcriptional noise," *Mol. Cell*, vol. 24, pp. 853–865, 2006.

[105] A. C. Birkegård, T. Halasa, N. Toft, A. Folkesson, and K. Græsbøll, "Send more data: A systematic review of mathematical models of antimicrobial resistance," *Antimicrobial Resistance & Infection Control*, vol. 7, no. 1, pp. 1–12, 2018.

[106] E. Pitzer and M. Affenzeller, "A comprehensive survey on fitness landscape analysis," *Recent advances in intelligent engineering systems*, pp. 161–191, 2012.

[107] H. A. Orr, "Fitness and its role in evolutionary genetics," *Nature Reviews Genetics*, vol. 10, no. 8, pp. 531–539, 2009.

[108] D. A. Charlebois and G. Balázsi, "Frequency-dependent selection: A diversifying force in microbial populations," *Molecular systems biology*, vol. 12, no. 8, p. 880, 2016.

[109] K. R. Ghusinga, J. J. Dennehy, and A. Singh, "First-passage time approach to controlling noise in the timing of intracellular events," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 114, 693–698, 2017.

[110] A. Singh and J. P. Hespanha, "Stochastic hybrid systems for studying biochemical processes," *Philos. Trans. Royal Soc. A*, vol. 368, pp. 4995–5011, 2010.

[111] J. Siepmann and F. Siepmann, "Mathematical modeling of drug delivery," *International journal of pharmaceutics*, vol. 364, no. 2, pp. 328–343, 2008.

[112] N. E. Ramia *et al.*, "Nested structure of intraspecific competition network in *Carnobacterium maltaromaticum*," *Sci. Rep.*, vol. 10, p. 7335, 2020.

[113] M. Baym, L. K. Stone, and R. Kishony, "Multidrug evolutionary strategies to reverse antibiotic resistance," *Science*, vol. 351, aad3292, 2016.

[114] R. D. Dar, N. N. Hosmane, M. R. Arkin, R. F. Siliciano, and L. S. Weinberger, "Screening for noise in gene expression identifies drug synergies," *Science*, vol. 344, pp. 1392–1396, 2020.

Appendix A: Code

The following sections present the source code I wrote to produce the results presented in this thesis. I wrote all of the codes utilizing suggestions and feedback from my supervisor Dr. Daniel Charlebois. The programming languages used were chosen to balance computational efficiency and ease of use. For computationally intensive algorithms (the SSA in particular) C was used. Python was used to analyze and plot the results obtained from the C code. MATLAB was used to create the deterministic simulations and analyze the results. The code shown below can be altered to produce all of the results presented in this thesis.

A.1 Deterministic Algorithms

A.1.1: MATLAB code used to numerically solve the coupled ODE system presented in Chapter 2 using ode45 [92].

```
function [t, X] = ODE_solver_model2_scenario1(t_i,t_end,dt,S,N,G1)
%Solves a system of coupled ODEs via MATLAB solver (default solver: ode45)
[t X] = ode45(@equations,t_i:dt:t_end,[S N G1]);

end


function dx = equations(t,x)

    global n k kS rSN rNS rG1S dS kN rG1N dN kG1 dG1

    dx = zeros(3,1);

    dx(1) = x(1)*kS*(k^n)/(k^n + (x(1)+x(2)+x(3))^n) + x(2)*rSN - x(1)*rNS - x(1)*rG1S - x(1)*dS;
    dx(2) = x(2)*kN*(k^n)/(k^n + (x(1)+x(2)+x(3))^n) + x(1)*rNS - x(2)*rG1N - x(2)*rSN - x(2)*dN;
    dx(3) = x(3)*kG1*(k^n)/(k^n + (x(1)+x(2)+x(3))^n) + x(1)*rG1S + x(2)*rG1N - x(3)*dG1;


end
```

A.1.2: MATLAB code used to run the deterministic simulations and produce the time series and heatmap plots presented in Chapter 2 for the constant drug environment.

```
function deterministic_simulations(model, scenario, relative_to_N0, cidal, deathfactor, ...
                                   G1_susceptible, G2_susceptible, log_bool, save_time_series, ↙
                                       ↪ ...
                                   plot_heatmaps, linspace_n, save_heatmaps, deathrate_sweep, ...
                                   t_end, dt, S_i, N_i, G1_i, G2_i, save_to)

    % Joshua Guthrie, Charlebois Laboratory, University of Alberta
% Simulation code using deterministic ODE models for quantitative
% non-genetic/genetic drug resistance/evolution study. Covers numerical simulations
% for multiple models, scenarios, and drug types, as well as result plotting.

global N0 k n kS kN kG1 kG2 rSN rNS rG1S rG1N rG2G1 dS dN dG1 dG2 kN_dsweep kG_dsweep
kN_list = [0.1733 0.2600 0.3466]; kG_list = [0.3466 0.3466 0.3466];
dN_list = [0.1 0.5 1.0];
dS_list = [1 1 1];



%% Scenario 1
% Run Scenario 1 (no G2) for different values of kG1 and kN
if scenario == 1
    t_list = [];
    S_list = [];
    N_list = [];
    G1_list = [];
    dN_dt_list = [];
    t_est_list = [];
    t_fix_list = [];
    legend_list = [];

    % run simulations for each kN and kG
    for i = 1:length(kN_list)
        kN = kN_list(i); kG1 = kG_list(i);
        % if using cidal drugs, specify death rates
        if cidal
            dN = dN_list(i);
            dS = dS_list(i);
            if G1_susceptible
                dG1 = 0.5;
            else
                dG1 = 1/156;
            end
        else
            dS = 1/156; dN = 1/156; dG1 = 1/156; dG2 = 0.0;
        end
        % run simulations, save results to lists
        [t, S, N, G1, ~, dN_dt, t_est, t_fix] = simulate(model,scenario,relative_to_N0,t_end,↙
            ↳ dt,S_i,N_i,G1_i,G2_i);
        if cidal
            legend_text = sprintf('\\delta_N = %.1f /hr', dN);
        else
            legend_text = sprintf('k_N = %.4f /hr', kN);
        end
        t_list = [t_list, t];
        S_list = [S_list, S];
        N_list = [N_list, N];
        G1_list = [G1_list, G1];
        dN_dt_list = [dN_dt_list, dN_dt];
        t_est_list = [t_est_list, t_est];
        t_fix_list = [t_fix_list, t_fix];
        legend_list = [legend_list, {legend_text}];
    end

    % plot simulation results
    % create the time series file name
    if save_time_series
       directory = save_to;
       if cidal
           drug_type = 'cidal';
           deathfactor_str = sprintf('_deathfactor%d',deathfactor);
       else
           drug_type = 'static';
           deathfactor_str ='';
       end
       if G1_susceptible
           G1_susceptible_str = '_G1susceptible';
       else
           G1_susceptible_str = '';
       end

       if G2_susceptible
           G2_susceptible_str = '_G2susceptible';
       else
           G2_susceptible_str = '';
       end

       if log_bool
           log_str = '_loglog';
       else
           log_str = '';
       end

       TS_filename = sprintf("%smodel%d_scenario%d_%s%s%s%s_Ni%0.1e%s_TimeSeries",directory,↙
           ↳ model,scenario,...
                                 drug_type,deathfactor_str,G1_susceptible_str,G2_susceptible_str,↙
                                     ↳  N_i, log_str);
    else
        TS_filename = 'n/a';
    end
```

```
        plot_simulation_results(t_list,S_list,N_list,G1_list,'n/a',dN_dt_list,legend_list,↙
            ↳ log_bool,save_time_series,TS_filename)

        % run large amount of simulations and plot t_est and t_fix heatmaps
        if plot_heatmaps
            [kG_used_est, kG_used_fix, kN_used_est, kN_used_fix, t_est_list, t_fix_list] = ...
            population_times(model, scenario, relative_to_N0, t_end, dt,  ...
                             S_i, N_i, G1_i, G2_i, cidal, G1_susceptible, ...
                             G2_susceptible, deathfactor, linspace_n);
            if save_heatmaps
               directory = save_to;
               if cidal
                   drug_type = 'cidal';
                   deathfactor_str = sprintf('_deathfactor%d',deathfactor);
               else
                   drug_type = 'static';
                   deathfactor_str ='';
               end
               if G1_susceptible
                   G1_susceptible_str = '_G1susceptible';
               else
                   G1_susceptible_str = '';
               end

               if G2_susceptible
                   G2_susceptible_str = '_G2susceptible';
               else
                   G2_susceptible_str = '';
               end

               HM_filename_est = sprintf("%smodel%d_scenario%d_%s%s%s%s_Ni%0.1↙
                   ↳ e_establishment_heatmap",directory,model,scenario,...
                                     drug_type,deathfactor_str,G1_susceptible_str,↙
                                         ↳ G2_susceptible_str, N_i);
               HM_filename_fix = sprintf("%smodel%d_scenario%d_%s%s%s%s_Ni%0.1e_fixation_heatmap↙
                   ↳ ",directory,model,scenario,...
                                     drug_type,deathfactor_str,G1_susceptible_str,↙
                                         ↳ G2_susceptible_str, N_i);
            else
                HM_filename_est = 'n/a';
                HM_filename_fix = 'n/a';
            end
            plot_population_times(kG_used_est, kN_used_est, t_est_list, 'Establishment', scenario↙
                ↳ , save_heatmaps, HM_filename_est)
            plot_population_times(kG_used_fix, kN_used_fix, t_fix_list, 'Fixation', scenario, ↙
                ↳ save_heatmaps, HM_filename_fix)
        end

        % run a sweep of varying cidal drug death rate values
        if deathrate_sweep
            [dG_used_est, dG_used_fix, dN_used_est, dN_used_fix, t_est_list, t_fix_list] = ...
            population_times_delta(model, scenario, relative_to_N0, t_end, dt,  ...
                                   S_i, N_i, G1_i, G2_i, G1_susceptible, ...
                                   G2_susceptible, kN_dsweep, kG_dsweep, linspace_n);
            if save_heatmaps
               directory = save_to;
               if G1_susceptible
                   G1_susceptible_str = '_G1susceptible';
               else
                   G1_susceptible_str = '';
               end

               if G2_susceptible
                   G2_susceptible_str = '_G2susceptible';
               else
                   G2_susceptible_str = '';
               end

               HM_filename_est = sprintf("%smodel%d_scenario%d_%s%s_Ni%0.1e_kN%0.4f_kG%0.4↙
                   ↳ f_deathrate_establishment_heatmap",directory,model,scenario,...
                                     G1_susceptible_str,G2_susceptible_str, N_i, kN_dsweep, ↙
                                         ↳ kG_dsweep);
               HM_filename_fix = sprintf("%smodel%d_scenario%d_%s%s_Ni%0.1e_kN%0.4f_kG%0.4↙
                   ↳ f_deathrate_fixation_heatmap",directory,model,scenario,...
                                     G1_susceptible_str,G2_susceptible_str, N_i, kN_dsweep, ↙
                                         ↳ kG_dsweep);
            else
                HM_filename_est = 'n/a';
                HM_filename_fix = 'n/a';
            end
            plot_population_times_delta(dG_used_est, dN_used_est, t_est_list, 'Establishment', ↙
                ↳ kN_dsweep, kG_dsweep, save_heatmaps, HM_filename_est)
            plot_population_times_delta(dG_used_fix, dN_used_fix, t_fix_list, 'Fixation', ↙
                ↳ kN_dsweep, kG_dsweep, save_heatmaps, HM_filename_fix)
        end
    end

    %% Scenario 2
    % Run Scenario 2 simulations for different values of kN, kG1, kG2
    if scenario == 2
        t_list = [];
```

```
S_list = [];
N_list = [];
G1_list = [];
G2_list = [];
dN_dt_list = [];
t_est_list = [];
t_fix_list = [];
legend_list = [];

% run simulations for all kN and kG
for i = 1:length(kN_list)
    kN = kN_list(i); kG1 = 0.1733; kG2 = kG_list(i);

    % set cidal drug death rates
    if cidal
        dN = 0.1733*deathfactor;
        dS = 0.3466*deathfactor;

        if G1_susceptible
            dG1 = 0.2600*deathfactor;
        else
            dG1 = 1/156;
        end
        if G2_susceptible
            dG2 = 0.1733*deathfactor;
        else
            dG2 = 0.0;
        end
    else
        dS = 1/156; dN = 1/156; dG1 = 1/156; dG2 = 0.0;
    end

    % run simulations, save results to lists
    [t, S, N, G1, G2, dN_dt, t_est, t_fix] = simulate(model,scenario,relative_to_N0,t_end↙
        ↳ ,dt,S_i,N_i,G1_i,G2_i);
    if i == 1
        legend_text = sprintf('k_N = %.4f /hr,\nk_{G2} = %.4f /hr\n', [kN kG2]);
    else
        legend_text = sprintf('\nk_N = %.4f /hr,\nk_{G2} = %.4f /hr\n', [kN kG2]);
    end
    t_list = [t_list, t];
    S_list = [S_list, S];
    N_list = [N_list, N];
    G1_list = [G1_list, G1];
    G2_list = [G2_list, G2];
    dN_dt_list = [dN_dt_list, dN_dt];
    t_est_list = [t_est_list, t_est];
    t_fix_list = [t_fix_list, t_fix];
    legend_list = [legend_list, {legend_text}];
end
% plot simulation results
% create the time series file name
if save_time_series
   directory = save_to;
   if cidal
       drug_type = 'cidal';
       deathfactor_str = sprintf('_deathfactor%d',deathfactor);
   else
       drug_type = 'static';
       deathfactor_str ='';
   end
   if G1_susceptible
       G1_susceptible_str = '_G1susceptible';
   else
       G1_susceptible_str = '';
   end

   if G2_susceptible
       G2_susceptible_str = '_G2susceptible';
   else
       G2_susceptible_str = '';
   end

   if log_bool
       log_str = '_loglog';
   else
       log_str = '';
   end

   TS_filename = sprintf("%smodel%d_scenario%d_%s%s%s%s_Ni%0.1e%s_TimeSeries",directory,↙
       ↳ model,scenario,...
                          drug_type,deathfactor_str,G1_susceptible_str,G2_susceptible_str,↙
                              ↳  N_i, log_str);
else
    TS_filename = 'n/a';
end

plot_simulation_results(t_list,S_list,N_list,G1_list,G2_list,dN_dt_list,legend_list,↙
    ↳ log_bool, save_time_series, TS_filename)

% run large amount of simulations and plot t_est and t_fix heatmaps
if plot_heatmaps
```

```
            [kG_used_est, kG_used_fix, kN_used_est, kN_used_fix, t_est_list, t_fix_list] = ...
            population_times(model, scenario, relative_to_N0, t_end, dt,  ...
                             S_i, N_i, G1_i, G2_i, cidal, G1_susceptible, ...
                             G2_susceptible, deathfactor, linspace_n);
            if save_heatmaps
               directory = save_to;
               if cidal
                   drug_type = 'cidal';
                   deathfactor_str = sprintf('_deathfactor%d',deathfactor);
               else
                   drug_type = 'static';
                   deathfactor_str ='';
               end
               if G1_susceptible
                   G1_susceptible_str = '_G1susceptible';
               else
                   G1_susceptible_str = '';
               end

               if G2_susceptible
                   G2_susceptible_str = '_G2susceptible';
               else
                   G2_susceptible_str = '';
               end

               HM_filename_est = sprintf("%smodel%d_scenario%d_%s%s%s%s_Ni%0.1↙
                   ↳ e_establishment_heatmap",directory,model,scenario,...
                                     drug_type,deathfactor_str,G1_susceptible_str,↙
                                          ↳ G2_susceptible_str, N_i);
               HM_filename_fix = sprintf("%smodel%d_scenario%d_%s%s%s%s_Ni%0.1e_fixation_heatmap↙
                   ↳ ",directory,model,scenario,...
                                     drug_type,deathfactor_str,G1_susceptible_str,↙
                                          ↳ G2_susceptible_str, N_i);
            else
                HM_filename_est = 'n/a';
                HM_filename_fix = 'n/a';
            end

            plot_population_times(kG_used_est, kN_used_est, t_est_list, 'Establishment', 2, ↙
                ↳ save_heatmaps, HM_filename_est)
            plot_population_times(kG_used_fix, kN_used_fix, t_fix_list, 'Fixation', 2, ↙
                ↳ save_heatmaps, HM_filename_fix)
        end
    end
end

%% functions
function [time, S, N, G1, G2, dN_dt, t_est, t_fix] = simulate(model,scenario,relative_to_N0,t_end↙
    ↳ ,dt,S_i,N_i,G1_i,G2_i)
    % This fuction runs the simulations for specified simulation parameters
    % and numerical values.
    % Input:
    % [int or float] simulation time (t_end), [int or float] time step (dt), [int or float] ↙
        ↳ initial population concentrations,
    % [int] simulation scenario
    % Output:
    % time, population concentrations as functions of time, conservation dN/dt as a
    % function of time, establishment time (t_est) and fixation time (t_fix) for the given ↙
        ↳ scenario

    % bring globals into function space
    global N0 k n kS kN kG1 kG2 rSN rNS rG1S rG1N rG2G1 dS dN dG1 dG2

    % Determine which ODE solver to use based on scenario and model
    if scenario == 2
        if model == 1
            [t, X] = ODE_solver_model1_scenario2(t_end,dt,S_i,N_i,G1_i,G2_i);
        elseif model == 2
            [t, X] = ODE_solver_model2_scenario2(t_end,dt,S_i,N_i,G1_i,G2_i);
        end
    elseif scenario == 1
        if model == 1
            [t, X] = ODE_solver_model1_scenario1(t_end,dt,S_i,N_i,G1_i);
        elseif model == 2
            [t, X] = ODE_solver_model2_scenario1(t_end,dt,S_i,N_i,G1_i);
        end
    end

    % simulation results
    time = t;
    S = X(:,1);
    N = X(:,2);
    G1 = X(:,3);
    if scenario == 2
        G2 = X(:,4);
    elseif scenario == 1
        G2 = 'n/a';
    end


    % calculate conservation equation results based on scenario and model
    dN_dt_list = [];
    for i = 1:length(S)
        if model == 1
            if scenario == 2
                dN_dt_calc = S(i)*kS + N(i)*kN + G1(i)*kG1 +G2(i)*kG2 - S(i)*dS - N(i)*dN - G1(i)↙
                    ↳ *dG1 - G2(i)*dG2;
            elseif scenario == 1
                dN_dt_calc = S(i)*kS + N(i)*kN + G1(i)*kG1 - S(i)*dS - N(i)*dN - G1(i)*dG1 ;
            end
        elseif model == 2
            if scenario == 2
                dN_dt_calc =  S(i)*kS*(k^n)/(k^n + (S(i)+N(i)+G1(i)+G2(i))^n) ...
                    + N(i)*kN*(k^n)/(k^n + (S(i)+N(i)+G1(i)+G2(i))^n) ...
                    + G1(i)*kG1*(k^n)/(k^n + (S(i)+N(i)+G1(i)+G2(i))^n) ...
                    + G2(i)*kG2*(k^n)/(k^n + (S(i)+N(i)+G1(i)+G2(i))^n) ...
                    - S(i)*dS - N(i)*dN - G1(i)*dG1 - G2(i)*dG2;
            elseif scenario == 1
                dN_dt_calc = S(i)*kS*(k^n)/(k^n + (S(i)+N(i)+G1(i))^n) ...
                    + N(i)*kN*(k^n)/(k^n + (S(i)+N(i)+G1(i))^n) ...
                    + G1(i)*kG1*(k^n)/(k^n + (S(i)+N(i)+G1(i))^n) ...
                    - S(i)*dS - N(i)*dN - G1(i)*dG1;
            end
        end
        dN_dt_list(i,:) = dN_dt_calc;
    end
    dN_dt = dN_dt_list;

    % establishment and fixation time calculation
    establishment = false;
    fixation = false;
    t_est = "> t_end";
    t_fix = "> t_end";
    index = 1;
    pop_fraction = 0;
    if scenario == 2
        population = G2;
    elseif scenario == 1
        population = G1;
    end
    % calculate relative to either N_0 or N_tot
    if relative_to_N0
        % loop through the results until fixation is found
        while fixation == false && index <= length(population)
            pop_t = t(index);
            pop_fraction = population(index)/N0;
            % check for establishment
            if establishment == false && pop_fraction > 0.05
                t_est = pop_t;
                establishment = true;
            end
            % check for fixation (exit loop if found)
            if pop_fraction > 0.95
                t_fix = pop_t;
                fixation = true;
            end
            index = index + 1;
        end
    else
        while fixation == false && index <= length(population)
            pop_t = t(index);
            if scenario == 2
                pop_fraction = population(index) / (S(index) + N(index) + G1(index) + population(↙
                    ↳ index));
            elseif scenario == 1
                pop_fraction = population(index) / (S(index) + N(index) + population(index));
            end
            % check for establishment
            if establishment == false && pop_fraction > 0.05
                t_est = pop_t;
                establishment = true; % to make sure t_est isn't overwritten
            end
            % check for fixation (exits loop if found)
            if pop_fraction > 0.95
                t_fix = pop_t;
                fixation = true;
            end
            index = index + 1;
        end
    end
    fprintf("t_est = %0.2f, t_fix = %0.2f\n", t_est, t_fix) % print est/fix time results to ↙
        ↳ command window
end


function [kG_used_est, kG_used_fix, kN_used_est, kN_used_fix, t_est_list, t_fix_list] = ...
    population_times(model, scenario, relative_to_N0, t_end, dt, S_i, N_i, G1_i, G2_i, ...
                    cidal, G1_susceptible, G2_susceptible, death_factor, linspace_n)
    % Create simulations for large variations of kG1 and kN
    % Input:
    % [int] number of combinations to consider (linspace_n),
    % all parameters for simulate() function, [int or float] cidal drug
    % death factor (death_factor)
    % Output:
```

```
    % lists of the kG and kN values used for both the establishment and
    % fixation times, lists for the corresponding establishment and
    % fixation time results

    % bring globals into function space

    global N0 k n kS kN kG1 kG2 rSN rNS rG1S rG1N rG2G1 dS dN dG1 dG2
    % create intial lists of kG and kN values and create array of all
    % combinations of these values
    kG_list = linspace(0.1733,0.3466,linspace_n);
    kN_list = linspace(0.1733,0.3466,linspace_n);
    kG_kN_combinations = combvec(kG_list,kN_list);

    kG_used_est = [];
    kG_used_fix = [];
    kN_used_est = [];
    kN_used_fix = [];
    t_est_list = [];
    t_fix_list = [];

    % Go through all combinations of kG and kN and run simulations
    for i = 1:length(kG_kN_combinations)
        combination = kG_kN_combinations(:,i); % gets a combination
        kG = combination(1);
        kN = combination(2);

        % change values based on scenario
        if scenario == 1
            kG1 = kG;
        elseif scenario == 2
            kG2 = kG;
            kG1 = 0.1733;
        end

        % use death factor if one is given
        if cidal
            dN = 0.1733*death_factor;
            dS = 0.3466*death_factor;
            if G1_susceptible
                dG1 = 0.2600*death_factor;
            else
                dG1 = 1/156;
            end
            if G2_susceptible
                dG2 = 0.2600*death_factor;
            else
                dG2 = 0.0;
            end
        else
            dS = 1/156; dN = 1/156; dG1 = 1/156; dG2 = 0.0;
        end

        % check if kN <= kG (as required)
        if kN <= kG
            [~,~,~,~,~,~, t_est, t_fix] = simulate(model,scenario,relative_to_N0,t_end, dt, S_i, ↙
                ↳ N_i, G1_i, G2_i); % run simulation
            % if t_est is found, save it and the kG and kN used to the result lists
            if isnumeric(t_est)
                kG_used_est = [kG_used_est,kG];
                kN_used_est = [kN_used_est,kN];
                t_est_list = [t_est_list,t_est];
            end
            % same for t_fix
            if isnumeric(t_fix)
                kG_used_fix = [kG_used_fix,kG];
                kN_used_fix = [kN_used_fix,kN];
                t_fix_list = [t_fix_list,t_fix];
            end
        end
    end
end

function [dG_used_est, dG_used_fix, dN_used_est, dN_used_fix, t_est_list, t_fix_list] = ...
    population_times_delta(model, scenario, relative_to_N0, t_end, dt, S_i, N_i, G1_i, G2_i, ...
                           G1_susceptible, G2_susceptible, kN_dsweep, kG_dsweep, linspace_n)
    % Create simulations for large variations of kG1 and kN
    % Input:
    % [int] number of combinations to consider (linspace_n),
    % all parameters for simulate() function, [int or float] cidal drug
    % death factor (death_factor)
    % Output:
    % lists of the kG and kN values used for both the establishment and
    % fixation times, lists for the corresponding establishment and
    % fixation time results

    % bring globals into function space

    global N0 k n kS kN kG1 kG2 rSN rNS rG1S rG1N rG2G1 dS dN dG1 dG2
    % create intial lists of kG and kN values and create array of all
    % combinations of these values

    kN = kN_dsweep;
    if scenario == 1
        kG1 = kG_dsweep;
    elseif scenario == 2
        kG1 = 0.1733;
        kG2 = kG_dsweep;
    end

    dN_list = linspace(0.1,1.0,linspace_n);
    dG_list = linspace(1/156,0.05,linspace_n);
    dG_dN_combinations = combvec(dG_list,dN_list);

    dG_used_est = [];
    dG_used_fix = [];
    dN_used_est = [];
    dN_used_fix = [];
    t_est_list = [];
    t_fix_list = [];

    % Go through all combinations of dG and dN and run simulations
    for i = 1:length(dG_dN_combinations)
        combination = dG_dN_combinations(:,i); % gets a combination
        dG1 = combination(1);
        dN = combination(2);
        dS = 1.0;

        [~,~,~,~,~,~, t_est, t_fix] = simulate(model,scenario,relative_to_N0,t_end, dt, S_i, N_i,↙
            ↳ G1_i, G2_i); % run simulation
        % if t_est is found, save it and the kG and kN used to the result lists
        if isnumeric(t_est)
            dG_used_est = [dG_used_est,dG1];
            dN_used_est = [dN_used_est,dN];
            t_est_list = [t_est_list,t_est];
        end
        % same for t_fix
        if isnumeric(t_fix)
            dG_used_fix = [dG_used_fix,dG1];
            dN_used_fix = [dN_used_fix,dN];
            t_fix_list = [t_fix_list,t_fix];
        end
    end
end

function plot_simulation_results(t,S,N,G1,G2,dN_dt,legend_set,log_bool, savefig_bool, file_name)
    % Plots the results of a given simulation
    % Inputs:
    % [list] 1D lists containing results for different simulations, [set]
    % legend string for different kN kG values (legend_set), bool to use
    % loglog instead of regular plots (log_bool), bool to save figure to a
    % file (savefig_bool), [str] name of file with or without extension
    % (default is .fig)

    % Create figure and subplots, for each population plot all results on the same subplot
    fig = figure;
    figure('DefaultAxesFontSize',24);

    subplot(2,3,1);
    % plot each of the simulation results in the input lists
    for i = 1:length(S(1,:))
        % plot in loglog
        if log_bool
            % for different simulation sets, use different line formats
            if i == 1
                loglog(t(:,i),S(:,i),'LineWidth',4);
            elseif i == 2
                loglog(t(:,i),S(:,i), '-.','LineWidth',4);
            elseif i == 3
                loglog(t(:,i),S(:,i),'--','LineWidth',4);
            end
        else
            if i == 1
                plot(t(:,i),S(:,i),'LineWidth',4);
            elseif i == 2
                plot(t(:,i),S(:,i), '-.','LineWidth',4);
            elseif i == 3
                plot(t(:,i),S(:,i),'--','LineWidth',4);
            end
        end
        hold on
    end
    hold off
    % axis labels and title
    xlabel('t (hr)'); ylabel('S(t) (cells/mL)');

    xlim([10^-1 10^5]);
    ylim([10^-5 10^9]);

    xticks([ 10^-1 10^2 10^5])
    xticklabels({'10^{-1}','10^{2}','10^{5}'})

    ttl = title('A)');
    ttl.Units = 'Normalize';
    ttl.Position(1) = -0.36;
    ttl.HorizontalAlignment = 'left';

%     subplot(2,3,2);
% %     %plot(0,0,  0,0,  0,0, 'LineWidth', 2);
% %     plot(0,0,'LineWidth',3);
% %     hold on
% %     plot(0,0, '-.','LineWidth',3);
% %     hold on
% %     plot(0,0,'--','LineWidth',3);
% %     hold off
% %     axis off;
%     ttl = title('A)');
%     ttl.Units = 'Normalize';
%     ttl.Position(1) = -0.33;
%     ttl.HorizontalAlignment = 'left';
    % plot legend
    l = legend(legend_set, 'Location', 'southwest');
    l.FontSize = 18;

    subplot(2,3,2);
    % same comments as above
    for i = 1:length(N(1,:))
        if log_bool
            if i == 1
                loglog(t(:,i),N(:,i),'LineWidth',4);
            elseif i == 2
                loglog(t(:,i),N(:,i), '-.','LineWidth',4);
            elseif i == 3
                loglog(t(:,i),N(:,i),'--','LineWidth',4);
            end
        else
            if i == 1
                plot(t(:,i),N(:,i),'LineWidth',4);
            elseif i == 2
                plot(t(:,i),N(:,i), '-.','LineWidth',4);
            elseif i == 3
                plot(t(:,i),N(:,i),'--','LineWidth',4);
            end
        end
        hold on
    end
    hold off
    xlabel('t (hr)'); ylabel('N(t) (cells/mL)');

    xlim([10^-1 10^5]);
    ylim([10^-5 10^9]);

    xticks([ 10^-1 10^2 10^5])
    xticklabels({'10^{-1}','10^{2}','10^{5}'})

    ttl = title('B)');
    ttl.Units = 'Normalize';
    ttl.Position(1) = -0.35;
    ttl.HorizontalAlignment = 'left';
    %title('N')

    subplot(2,3,3);
    for i = 1:length(G1(1,:))
        if log_bool
            if i == 1
                loglog(t(:,i),G1(:,i),'LineWidth',4);
            elseif i == 2
                loglog(t(:,i),G1(:,i), '-.','LineWidth',4);
            elseif i == 3
                loglog(t(:,i),G1(:,i),'--','LineWidth',4);
            end
        else
            if i == 1
                plot(t(:,i),G1(:,i),'LineWidth',4);
            elseif i == 2
                plot(t(:,i),G1(:,i), '-.','LineWidth',4);
            elseif i == 3
                plot(t(:,i),G1(:,i),'--','LineWidth',4);
            end
        end
        hold on
    end
    hold off
    xlabel('t (hr)'); ylabel('G(t) (cells/mL)');

    xlim([10^-1 10^5]);
    ylim([10^-5 10^9]);

    xticks([ 10^-1 10^2 10^5])
    xticklabels({'10^{-1}','10^{2}','10^{5}'})

    ttl = title('C)');
    ttl.Units = 'Normalize';
    ttl.Position(1) = -0.35;
    ttl.HorizontalAlignment = 'left';
    %title('G')
```

```
if isnumeric(G2)
    subplot(2,3,5);
    for i = 1:length(G2(1,:))
        if log_bool
            if i == 1
                loglog(t(:,i),G2(:,i),'LineWidth',4);
            elseif i == 2
                loglog(t(:,i),G2(:,i), '-.','LineWidth',4);
            elseif i == 3
                loglog(t(:,i),G2(:,i),'--','LineWidth',4);
            end
        else
            if i == 1
                plot(t(:,i),G2(:,i),'LineWidth',4);
            elseif i == 2
                plot(t(:,i),G2(:,i), '-.','LineWidth',4);
            elseif i == 3
                plot(t(:,i),G2(:,i),'--','LineWidth',4);
            end
        end
        hold on
    end
    hold off
    xlabel('t (hr)'); ylabel('G_2(t) (cells/mL)');
    %title('G_2')
end

subplot(2,3,4);
for i = 1:length(G1(1,:))
    T = S(:,i) + N(:,i) + G1(:,i);
    if log_bool
        if i == 1
            loglog(t(:,i),T,'LineWidth',4);
        elseif i == 2
            loglog(t(:,i),T, '-.','LineWidth',4);
        elseif i == 3
            loglog(t(:,i),T,'--','LineWidth',4);
        end
    else
        if i == 1
            plot(t(:,i),T,'LineWidth',4);
        elseif i == 2
            plot(t(:,i),T, '-.','LineWidth',4);
        elseif i == 3
            plot(t(:,i),T,'--','LineWidth',4);
        end
    end
    hold on
end
hold off
xlabel('t (hr)'); ylabel('T(t) (cells/mL)');

xlim([10^-1 10^5]);
ylim([10^-5 10^9]);

xticks([ 10^-1 10^2 10^5])
xticklabels({'10^{-1}','10^{2}','10^{5}'})

ttl = title('D)');
ttl.Units = 'Normalize';
ttl.Position(1) = -0.35;
ttl.HorizontalAlignment = 'left';

subplot(2,3,5);
for i = 1:length(G1(1,:))
    if isnumeric(G2)
        ratio = G2(:,i)./N(:,i);
        ratio_string = 'G_2(t)/N(t)';
        ratio_title_string = 'G_2/N';
    else
        ratio = G1(:,i)./(S(:,i) + N(:,i) + G1(:,i));
        ratio_string = 'G(t)/T(t)';
        ratio_title_string = 'G/T Ratio';

    end
    if log_bool
        if i == 1
            loglog(t(:,i),ratio,'LineWidth',4);
        elseif i == 2
            loglog(t(:,i),ratio, '-.','LineWidth',4);
        elseif i == 3
            loglog(t(:,i),ratio,'--','LineWidth',4);
        end
    else
        if i == 1
            plot(t(:,i),ratio,'LineWidth',4);
        elseif i == 2
            plot(t(:,i),ratio, '-.','LineWidth',4);
        elseif i == 3
            plot(t(:,i),ratio,'--','LineWidth',4);
        end
    end
```

```
        hold on
    end
    hold off
    xlabel('t (hr)'); ylabel(ratio_string);

    xlim([10^-1 10^5]);
    ylim([10^-10 10^1]);

    xticks([ 10^-1 10^2 10^5])
    xticklabels({'10^{-1}','10^{2}','10^{5}'})


    ttl = title('E)');
    ttl.Units = 'Normalize';
    ttl.Position(1) = -0.39;
    ttl.HorizontalAlignment = 'left';
    %title(ratio_title_string)

    subplot(2,3,6);
    for i = 1:length(dN_dt(1,:))
        if i == 1
            semilogx(t(:,i),dN_dt(:,i),'LineWidth',4);
        elseif i == 2
            semilogx(t(:,i),dN_dt(:,i), '-.','LineWidth',4);
        elseif i == 3
            semilogx(t(:,i),dN_dt(:,i),'--','LineWidth',4);
        end
%          if log_bool
%              if i == 1
%                  loglog(t(:,i),dN_dt(:,i),'LineWidth',2);
%              elseif i == 2
%                  loglog(t(:,i),dN_dt(:,i), '-.','LineWidth',2);
%              elseif i == 3
%                  loglog(t(:,i),dN_dt(:,i),'--','LineWidth',2);
%              end
%          else
%              if i == 1
%                  plot(t(:,i),dN_dt(:,i),'LineWidth',2);
%              elseif i == 2
%                  plot(t(:,i),dN_dt(:,i), '-.','LineWidth',2);
%              elseif i == 3
%                  plot(t(:,i),dN_dt(:,i),'--','LineWidth',2);
%              end
%          end
        hold on
    end
    hold off
    xlabel('t (hr)'); ylabel('dT/dt (cells/mLhr)');

    xlim([10^-1 10^5]);


    xticks([ 10^-1 10^2 10^5])
    xticklabels({'10^{-1}','10^{2}','10^{5}'})

    ttl = title('F)');
    ttl.Units = 'Normalize';
    ttl.Position(1) = -0.35;
    ttl.HorizontalAlignment = 'left';
    %title('dT/dt')


    set(gcf, 'PaperUnits', 'inches');
    x_width=18 ;y_width=12;
    set(gcf, 'PaperPosition', [0 0 x_width y_width]);
    orient(fig,'landscape');
    if savefig_bool
        saveas(gcf, append(file_name, '.png'))
        %save(append(file_name, '.mat'),'t','S','N','G1','G2','dN_dt')
    end
end


function plot_population_times(kG_list, kN_list, population_time_list, population_type, scenario,↙
    ↳ savefig_bool, file_name)
    % Plot heatmaps using the kG, kN and t_est or t_fix results for a large
    % amount of simulations.
    % Input:
    % [list] lists of the kG, kN used and t_est or t_fix results, [str]
    % quantitative measure used ("Establishment" or "Fixation") for plot
    % titles (population_type), [int] scenario simulated for plot titles
    % and file names(scenario), [bool] to save figure or not
    % (savefig_bool),[str] name of file to save (with or without file type extension, default is ↙
        ↳ .fig if there is none)

    % create figure
    fig = figure;

    % put input lists into a table to make the heatmaps with
    X = kG_list(:); Y = kN_list(:); Z = population_time_list(:);
    tbl = table(X,Y,Z);
    % create the heatmap
    hHM = heatmap(tbl,'X', 'Y', 'ColorVariable','Z','ColorMethod', 'none', 'CellLabelColor','none↙
        ↪ ',...
                'GridVisible','off','MissingDataLabel', 'No Data', 'MissingDataColor','w', ...
                'FontSize', 20);%,'ColorScaling','log');
    hHM.NodeChildren(3).YDir='normal'; % flips the y-axis to make it increasing (matlab default ↙
        ↪ is decreasing)
    hHM.Colormap = jet;


    % set axis labels and title
    ylabel('k_N (/hr)')
    if scenario == 2
        xlabel('k_{G2} (/hr)')
        title('')%append(population_type,' Time as a Function of k_N and k_{G2} (k_N <= k_{G2})')↙
            ↪ )
    else
        xlabel('k_{G} (/hr)')
        title('')%append(population_type,' Time as a Function of k_N and k_{G} (k_N <= k_{G})'))
    end
    colorbar; % adds a color bar for the heatmap

    set(gcf, 'PaperUnits', 'inches');
    x_width=10 ;y_width=7;
    set(gcf, 'PaperPosition', [0 0 x_width y_width]);

    if population_type == "Establishment"
        annotation('textarrow',[0.98,0.98],[0.7,0.7],'string','{\it \tau_{est}} (hr)', 'FontSize'↙
            ↪ , 22, ...
            'HeadStyle','none','LineStyle','none','HorizontalAlignment','right','TextRotation'↙
                ↪ ,90);
        annotation('textarrow',[0.025,0.025],[0.97,0.97],'string','A)', 'FontWeight', 'bold', ...
            'FontSize', 22, 'HeadStyle','none','LineStyle','none','HorizontalAlignment','center')↙
                ↪ ;
    else
        annotation('textarrow',[0.98,0.98],[0.7,0.7],'string','{\it \tau_{fix}} (hr)', 'FontSize'↙
            ↪ , 22, ...
            'HeadStyle','none','LineStyle','none','HorizontalAlignment','right','TextRotation'↙
                ↪ ,90);
        annotation('textarrow',[0.025,0.025],[0.97,0.97],'string','A)', 'FontWeight', 'bold', ...
            'FontSize', 22, 'HeadStyle','none','LineStyle','none','HorizontalAlignment','center')↙
                ↪ ;
    end

    % rotate x-axis labels
    s = struct(hHM);
    s.XAxis.TickLabelRotation = 60;  % angled
    s.XDisplayLabels = compose('%.3f',str2double(hHM.XDisplayLabels));
    s.YDisplayLabels = compose('%.4f',str2double(hHM.YDisplayLabels));

    %s.FontSize = 60;
    % save the file
    if savefig_bool
        saveas(gcf, append(file_name,'.png'))
        %save(append(file_name, '.mat'), 'kG_list', 'kN_list', 'population_time_list')
    end

    close(fig)
end

function plot_population_times_delta(dG_list, dN_list, population_time_list, population_type, ↙
    ↪ kN_dsweep, kG_dsweep, savefig_bool, file_name)
    % create figure
    fig = figure;

    % put input lists into a table to make the heatmaps with
    X = dG_list(:); Y = dN_list(:); Z = population_time_list(:);
    tbl = table(X,Y,Z);

    % create the heatmap
    hHM = heatmap(tbl,'X', 'Y', 'ColorVariable','Z','ColorMethod', 'none', 'CellLabelColor','none↙
        ↪ ',...
                'GridVisible','off','MissingDataLabel', 'No Data', 'MissingDataColor','w', ...
                'FontSize', 20);
    hHM.NodeChildren(3).YDir='normal'; % flips the y-axis to make it increasing (matlab default ↙
        ↪ is decreasing)
    hHM.Colormap = jet;


    % set axis labels and title
    ylabel('\delta_N (/hr)')
    xlabel('\delta_G (/hr)')
    title('')

    colorbar; % adds a color bar for the heatmap

    set(gcf, 'PaperUnits', 'inches');
    x_width=10 ;y_width=7.2;
    set(gcf, 'PaperPosition', [0 0 x_width y_width]);

    if population_type == "Establishment"
        annotation('textarrow',[0.98,0.98],[0.7,0.7],'string','{\it \tau_{est}} (hr)', 'FontSize'↙
            ↳ , 22, ...
            'HeadStyle','none','LineStyle','none','HorizontalAlignment','right','TextRotation'↙
                ↳ ,90);
        annotation('textarrow',[0.025,0.025],[0.97,0.97],'string','A)', 'FontWeight', 'bold', ...
            'FontSize', 22, 'HeadStyle','none','LineStyle','none','HorizontalAlignment','center')↙
                ↳ ;
    else
        annotation('textarrow',[0.983,0.983],[0.7,0.7],'string','{\it \tau_{fix}} (hr)', '↙
            ↳ FontSize', 22, ...
            'HeadStyle','none','LineStyle','none','HorizontalAlignment','right','TextRotation'↙
                ↳ ,90);
        annotation('textarrow',[0.025,0.025],[0.97,0.97],'string','B)', 'FontWeight', 'bold', ...
            'FontSize', 22, 'HeadStyle','none','LineStyle','none','HorizontalAlignment','center')↙
                ↳ ;
    end

    % rotate x-axis labels
    s = struct(hHM);
    s.XAxis.TickLabelRotation = 60;  % angled
    s.XDisplayLabels = compose('%.3f',str2double(hHM.XDisplayLabels));
    s.YDisplayLabels = compose('%.4f',str2double(hHM.YDisplayLabels));



    %s.FontSize = 60;
    % save the file
    if savefig_bool
        saveas(gcf, append(file_name,'.png'))
        %save(append(file_name, '.mat'), 'kG_list', 'kN_list', 'population_time_list')
    end

    close(fig)
end
```

A.1.3: MATLAB code used to run the deterministic simulations and produce the time series and heatmap plots presented in Chapter 2 for the fluctuating drug environment.

```
function deterministic_simulations(model, scenario, relative_to_N0, cidal, deathfactor, ...
                                   G1_susceptible, G2_susceptible, log_bool, save_time_series, ↙
                                        ↳ ...
                                   plot_heatmaps, linspace_n, save_heatmaps, deathrate_sweep, ...
                                   t_end, dt, S_i, N_i, G1_i, G2_i, save_to)

    % Joshua Guthrie, Charlebois Laboratory, University of Alberta
    % Simulation code using deterministic ODE models for quantitative
    % non-genetic/genetic drug resistance/evolution study. Covers numerical simulations
    % for multiple models, scenarios, and drug types, as well as result plotting.

    global N0 k n kS kN kG1 kG2 rSN rNS rG1S rG1N rG2G1 dS dN dG1 dG2 kN_dsweep kG_dsweep
    kN_list = [0.1733 0.2600 0.3466]; kG_list = [0.3466 0.3466 0.3466];
    dN_list = [0.1 0.5 1.0];
    dS_list = [1 1 1];



    %% Scenario 1
    % Run Scenario 1 (no G2) for different values of kG1 and kN
    if scenario == 1
        t_list = [];
        S_list = [];
        N_list = [];
        G1_list = [];
        dN_dt_list = [];
        t_est_list = [];
        t_fix_list = [];
        legend_list = [];

        % run simulations for each kN and kG
        for i = 1:length(kN_list)
            kN = kN_list(i); kG1 = kG_list(i);
            % if using cidal drugs, specify death rates
            if cidal
                dN = dN_list(i);
                dS = dS_list(i);
                if G1_susceptible
                    dG1 = 0.5;
                else
                    dG1 = 1/156;
                end
            else
                dS = 1/156; dN = 1/156; dG1 = 1/156; dG2 = 0.0;
            end
            % run simulations, save results to lists
            [t, S, N, G1, ~, dN_dt, t_est, t_fix] = simulate(model,scenario,relative_to_N0,t_end,↙
                ↳ dt,S_i,N_i,G1_i,G2_i);
            if cidal
                legend_text = sprintf('\\delta_N = %.1f /hr', dN);
            else
                legend_text = sprintf('k_N = %.4f /hr', kN);
            end
            t_list = [t_list, t];
            S_list = [S_list, S];
            N_list = [N_list, N];
            G1_list = [G1_list, G1];
            dN_dt_list = [dN_dt_list, dN_dt];
            t_est_list = [t_est_list, t_est];
            t_fix_list = [t_fix_list, t_fix];
            legend_list = [legend_list, {legend_text}];
        end

        % plot simulation results
        % create the time series file name
        if save_time_series
           directory = save_to;
           if cidal
               drug_type = 'cidal';
               deathfactor_str = sprintf('_deathfactor%d',deathfactor);
           else
               drug_type = 'static';
               deathfactor_str ='';
           end
           if G1_susceptible
               G1_susceptible_str = '_G1susceptible';
           else
               G1_susceptible_str = '';
           end

           if G2_susceptible
               G2_susceptible_str = '_G2susceptible';
           else
               G2_susceptible_str = '';
           end

           if log_bool
               log_str = '_loglog';
           else
               log_str = '';
           end

           TS_filename = sprintf("%smodel%d_scenario%d_%s%s%s%s_Ni%0.1e%s_TimeSeries",directory,↙
                ↳ model,scenario,...
                                 drug_type,deathfactor_str,G1_susceptible_str,G2_susceptible_str,↙
                                      ↳  N_i, log_str);
        else
            TS_filename = 'n/a';
        end

        plot_simulation_results(t_list,S_list,N_list,G1_list,'n/a',dN_dt_list,legend_list,↙
             ↳ log_bool,save_time_series,TS_filename)

        % run large amount of simulations and plot t_est and t_fix heatmaps
        if plot_heatmaps
            [kG_used_est, kG_used_fix, kN_used_est, kN_used_fix, t_est_list, t_fix_list] = ...
            population_times(model, scenario, relative_to_N0, t_end, dt,  ...
                             S_i, N_i, G1_i, G2_i, cidal, G1_susceptible, ...
                             G2_susceptible, deathfactor, linspace_n);
            if save_heatmaps
               directory = save_to;
               if cidal
                   drug_type = 'cidal';
                   deathfactor_str = sprintf('_deathfactor%d',deathfactor);
               else
                   drug_type = 'static';
                   deathfactor_str ='';
               end
               if G1_susceptible
                   G1_susceptible_str = '_G1susceptible';
               else
                   G1_susceptible_str = '';
               end

               if G2_susceptible
                   G2_susceptible_str = '_G2susceptible';
               else
                   G2_susceptible_str = '';
               end

               HM_filename_est = sprintf("%smodel%d_scenario%d_%s%s%s%s_Ni%0.1↙
                    ↳ e_establishment_heatmap",directory,model,scenario,...
                                     drug_type,deathfactor_str,G1_susceptible_str,↙
                                          ↳ G2_susceptible_str, N_i);
               HM_filename_fix = sprintf("%smodel%d_scenario%d_%s%s%s%s_Ni%0.1e_fixation_heatmap↙
                    ↳ ",directory,model,scenario,...
                                     drug_type,deathfactor_str,G1_susceptible_str,↙
                                          ↳ G2_susceptible_str, N_i);
            else
                HM_filename_est = 'n/a';
                HM_filename_fix = 'n/a';
            end
            plot_population_times(kG_used_est, kN_used_est, t_est_list, 'Establishment', scenario↙
                 ↳ , save_heatmaps, HM_filename_est)
            plot_population_times(kG_used_fix, kN_used_fix, t_fix_list, 'Fixation', scenario, ↙
                ↳ save_heatmaps, HM_filename_fix)
        end

        % run a sweep of varying cidal drug death rate values
        if deathrate_sweep
            [dG_used_est, dG_used_fix, dN_used_est, dN_used_fix, t_est_list, t_fix_list] = ...
            population_times_delta(model, scenario, relative_to_N0, t_end, dt,  ...
                                   S_i, N_i, G1_i, G2_i, G1_susceptible, ...
                                   G2_susceptible, kN_dsweep, kG_dsweep, linspace_n);
            if save_heatmaps
               directory = save_to;
               if G1_susceptible
                   G1_susceptible_str = '_G1susceptible';
               else
                   G1_susceptible_str = '';
               end

               if G2_susceptible
                   G2_susceptible_str = '_G2susceptible';
               else
                   G2_susceptible_str = '';
               end

               HM_filename_est = sprintf("%smodel%d_scenario%d_%s%s_Ni%0.1e_kN%0.4f_kG%0.4↙
                   ↳ f_deathrate_establishment_heatmap",directory,model,scenario,...
                                     G1_susceptible_str,G2_susceptible_str, N_i, kN_dsweep, ↙
                                         ↳ kG_dsweep);
               HM_filename_fix = sprintf("%smodel%d_scenario%d_%s%s_Ni%0.1e_kN%0.4f_kG%0.4↙
                   ↳ f_deathrate_fixation_heatmap",directory,model,scenario,...
                                     G1_susceptible_str,G2_susceptible_str, N_i, kN_dsweep, ↙
                                         ↳ kG_dsweep);
            else
                HM_filename_est = 'n/a';
                HM_filename_fix = 'n/a';
            end
            plot_population_times_delta(dG_used_est, dN_used_est, t_est_list, 'Establishment', ↙
                ↳ kN_dsweep, kG_dsweep, save_heatmaps, HM_filename_est)
            plot_population_times_delta(dG_used_fix, dN_used_fix, t_fix_list, 'Fixation', ↙
                ↳ kN_dsweep, kG_dsweep, save_heatmaps, HM_filename_fix)
        end
    end
end

%% functions
function [time, S, N, G1, G2, dN_dt, t_est, t_fix] = simulate(model,scenario,relative_to_N0,t_end↙
    ↳ ,dt,S_i,N_i,G1_i,G2_i)
    % This fuction runs the simulations for specified simulation parameters
    % and numerical values.
    % Input:
    % [int or float] simulation time (t_end), [int or float] time step (dt), [int or float] ↙
        ↳ initial population concentrations,
    % [int] simulation scenario
    % Output:
    % time, population concentrations as functions of time, conservation dN/dt as a
    % function of time, establishment time (t_est) and fixation time (t_fix) for the given ↙
        ↳ scenario

    % bring globals into function space
    global N0 k n kS kN kG1 kG2 rSN rNS rG1S rG1N rG2G1 dS dN dG1 dG2

    G2 = 'n/a';

    % fluctuate the environment, run sims, and combine results (only for model 2, scenario 1)
    % distinquish between drug and no-drug environments
    fluc_interval = 12; %hours
    kS_drug = kS; kN_drug = kN; kG1_drug = kG1; rG1S_drug = rG1S; dS_drug = dS; dN_drug = dN; ↙
        ↳ dG1_drug = dG1;
    kS_no = 0.3466; kN_no = 0.1733; kG1_no = 0; rG1S_no = rG1N; dS_no = 1/156; dN_no = 1/156; ↙
        ↳ dG1_no = 1/156;


    t_fluc = 0:fluc_interval:t_end;
    S_i_fluc = S_i; N_i_fluc = N_i; G1_i_fluc = G1_i;

    for i = 1:(length(t_fluc)-1)
        % start with drug, remove drug on even intervals
        if rem(i,2)==0
            kS = kS_no; kN = kN_no; kG1 = kG1_no; rG1S = rG1S_no; dS = dS_no; dN = dN_no; dG1 = ↙
                ↳ dG1_no;
        else
            kS = kS_drug; kN = kN_drug; kG1 = kG1_drug; rG1S = rG1S_drug; dS = dS_drug; dN = ↙
                ↳ dN_drug; dG1 = dG1_drug;
        end

        t_i = t_fluc(i);
        t_end_fluc = t_fluc(i+1);

        [t, X] = ODE_solver_model2_scenario1(t_i,t_end_fluc,dt,S_i_fluc,N_i_fluc,G1_i_fluc);

        % calculate dT/dt
        dN_dt_temp = [];
        for j = 1:length(X(:,1))
            dN_dt_calc = X(j,1)*kS*(k^n)/(k^n + (X(j,1)+X(j,2)+X(j,3))^n) ...
                        + X(j,2)*kN*(k^n)/(k^n + (X(j,1)+X(j,2)+X(j,3))^n) ...
                        + X(j,3)*kG1*(k^n)/(k^n + (X(j,1)+X(j,2)+X(j,3))^n) ...
                        - X(j,1)*dS - X(j,2)*dN - X(j,3)*dG1;

            dN_dt_temp(j,:) = dN_dt_calc;
        end

        % update lists
        if i == 1
            time = t;
            S = X(:,1);
            N = X(:,2);
            G1 = X(:,3);
            dN_dt = dN_dt_temp;
        else
            time = [time; t];
            S = [S; X(:,1)];
            N = [N; X(:,2)];
            G1 = [G1; X(:,3)];
            dN_dt = [dN_dt; dN_dt_temp];
        end

        S_i_fluc = S(end);
        N_i_fluc = N(end);
        G1_i_fluc = G1(end);
    end

    % reset for safety
    kS = kS_drug; kN = kN_drug; kG1 = kG1_drug; rG1S = rG1S_drug; dS = dS_drug; dN = dN_drug; dG1↙
        ↳ = dG1_drug;


    % calculate conservation equation results based on scenario and model

%       for i = 1:length(S)
%           if model == 1
%               if scenario == 2
%                   dN_dt_calc = S(i)*kS + N(i)*kN + G1(i)*kG1 +G2(i)*kG2 - S(i)*dS - N(i)*dN - G1(↙
    ↳ i)*dG1 - G2(i)*dG2;
%               elseif scenario == 1
%                   dN_dt_calc = S(i)*kS + N(i)*kN + G1(i)*kG1 - S(i)*dS - N(i)*dN - G1(i)*dG1 ;
%               end
%           elseif model == 2
%               if scenario == 2
%                   dN_dt_calc =  S(i)*kS*(k^n)/(k^n + (S(i)+N(i)+G1(i)+G2(i))^n) ...
%                       + N(i)*kN*(k^n)/(k^n + (S(i)+N(i)+G1(i)+G2(i))^n) ...
%                       + G1(i)*kG1*(k^n)/(k^n + (S(i)+N(i)+G1(i)+G2(i))^n) ...
%                       + G2(i)*kG2*(k^n)/(k^n + (S(i)+N(i)+G1(i)+G2(i))^n) ...
%                       - S(i)*dS - N(i)*dN - G1(i)*dG1 - G2(i)*dG2;
%               elseif scenario == 1
%                   dN_dt_calc = S(i)*kS*(k^n)/(k^n + (S(i)+N(i)+G1(i))^n) ...
%                       + N(i)*kN*(k^n)/(k^n + (S(i)+N(i)+G1(i))^n) ...
%                       + G1(i)*kG1*(k^n)/(k^n + (S(i)+N(i)+G1(i))^n) ...
%                       - S(i)*dS - N(i)*dN - G1(i)*dG1;
%               end
%           end
%           dN_dt_list(i,:) = dN_dt_calc;
%       end
%       dN_dt = dN_dt_list;

    % establishment and fixation time calculation
    establishment = false;
    fixation = false;
    t_est = "> t_end";
    t_fix = "> t_end";
    index = 1;
    pop_fraction = 0;
    if scenario == 2
        population = G2;
    elseif scenario == 1
        population = G1;
    end
    % calculate relative to either N_0 or N_tot
    if relative_to_N0
        % loop through the results until fixation is found
        while fixation == false && index <= length(population)
            pop_t = time(index);
            pop_fraction = population(index)/N0;
            % check for establishment
            if establishment == false && pop_fraction > 0.05
                t_est = pop_t;
                establishment = true;
            end
            % check for fixation (exit loop if found)
            if pop_fraction > 0.95
                t_fix = pop_t;
                fixation = true;
            end
            index = index + 1;
        end
    else
        while fixation == false && index <= length(population)
            pop_t = time(index);
            if scenario == 2
                pop_fraction = population(index) / (S(index) + N(index) + G1(index) + population(↙
                    ↳ index));
            elseif scenario == 1
                pop_fraction = population(index) / (S(index) + N(index) + population(index));
            end
            % check for establishment
            if establishment == false && pop_fraction > 0.05
                t_est = pop_t;
                establishment = true; % to make sure t_est isn't overwritten
            end
            % check for fixation (exits loop if found)
            if pop_fraction > 0.95
                t_fix = pop_t;
                fixation = true;
            end
            index = index + 1;
        end
    end
    fprintf("t_est = %0.2f, t_fix = %0.2f\n", t_est, t_fix) % print est/fix time results to ↙
        ↳ command window
end


function [kG_used_est, kG_used_fix, kN_used_est, kN_used_fix, t_est_list, t_fix_list] = ...
    population_times(model, scenario, relative_to_N0, t_end, dt, S_i, N_i, G1_i, G2_i, ...
                     cidal, G1_susceptible, G2_susceptible, death_factor, linspace_n)
    % Create simulations for large variations of kG1 and kN
    % Input:
    % [int] number of combinations to consider (linspace_n),
    % all parameters for simulate() function, [int or float] cidal drug
    % death factor (death_factor)
    % Output:
    % lists of the kG and kN values used for both the establishment and
    % fixation times, lists for the corresponding establishment and
    % fixation time results

    % bring globals into function space

    global N0 k n kS kN kG1 kG2 rSN rNS rG1S rG1N rG2G1 dS dN dG1 dG2
    % create intial lists of kG and kN values and create array of all
    % combinations of these values
    kG_list = linspace(0.1733,0.3466,linspace_n);
    kN_list = linspace(0.1733,0.3466,linspace_n);
    kG_kN_combinations = combvec(kG_list,kN_list);

    kG_used_est = [];
    kG_used_fix = [];
    kN_used_est = [];
    kN_used_fix = [];
    t_est_list = [];
    t_fix_list = [];

    % Go through all combinations of kG and kN and run simulations
    for i = 1:length(kG_kN_combinations)
        combination = kG_kN_combinations(:,i); % gets a combination
        kG = combination(1);
        kN = combination(2);

        % change values based on scenario
        if scenario == 1
            kG1 = kG;
        elseif scenario == 2
            kG2 = kG;
            kG1 = 0.1733;
        end

        % use death factor if one is given
        if cidal
            dN = 0.1733*death_factor;
            dS = 0.3466*death_factor;
            if G1_susceptible
                dG1 = 0.2600*death_factor;
            else
                dG1 = 1/156;
            end
            if G2_susceptible
                dG2 = 0.2600*death_factor;
            else
                dG2 = 0.0;
            end
        else
            dS = 1/156; dN = 1/156; dG1 = 1/156; dG2 = 0.0;
        end

        % check if kN <= kG (as required)
        if kN <= kG
            [~,~,~,~,~,~, t_est, t_fix] = simulate(model,scenario,relative_to_N0,t_end, dt, S_i, ↙
                ↳ N_i, G1_i, G2_i); % run simulation
            % if t_est is found, save it and the kG and kN used to the result lists
            if isnumeric(t_est)
                kG_used_est = [kG_used_est,kG];
                kN_used_est = [kN_used_est,kN];
                t_est_list = [t_est_list,t_est];
            end
            % same for t_fix
            if isnumeric(t_fix)
                kG_used_fix = [kG_used_fix,kG];
                kN_used_fix = [kN_used_fix,kN];
                t_fix_list = [t_fix_list,t_fix];
            end
        end
    end
end

function [dG_used_est, dG_used_fix, dN_used_est, dN_used_fix, t_est_list, t_fix_list] =
    population_times_delta(model, scenario, relative_to_N0, t_end, dt, S_i, N_i, G1_i, G2_i, ...
                          G1_susceptible, G2_susceptible, kN_dsweep, kG_dsweep, linspace_n)
    % Create simulations for large variations of kG1 and kN
    % Input:
    % [int] number of combinations to consider (linspace_n),
    % all parameters for simulate() function, [int or float] cidal drug
    % death factor (death_factor)
    % Output:
    % lists of the kG and kN values used for both the establishment and
    % fixation times, lists for the corresponding establishment and
    % fixation time results

    % bring globals into function space

    global N0 k n kS kN kG1 kG2 rSN rNS rG1S rG1N rG2G1 dS dN dG1 dG2
    % create intial lists of kG and kN values and create array of all
    % combinations of these values

    kN = kN_dsweep;
    if scenario == 1
        kG1 = kG_dsweep;
    elseif scenario == 2
        kG1 = 0.1733;
        kG2 = kG_dsweep;
    end

    dN_list = linspace(0.1,1.0,linspace_n);
    dG_list = linspace(1/156,0.05,linspace_n);
    dG_dN_combinations = combvec(dG_list,dN_list);

    dG_used_est = [];
    dG_used_fix = [];
    dN_used_est = [];
    dN_used_fix = [];
    t_est_list = [];
    t_fix_list = [];

    % Go through all combinations of dG and dN and run simulations
    for i = 1:length(dG_dN_combinations)
        combination = dG_dN_combinations(:,i); % gets a combination
        dG1 = combination(1);
        dN = combination(2);
        dS = 1.0;

        [~,~,~,~,~,~, t_est, t_fix] = simulate(model,scenario,relative_to_N0,t_end, dt, S_i, N_i,↙
            ↳ G1_i, G2_i); % run simulation
        % if t_est is found, save it and the kG and kN used to the result lists
        if isnumeric(t_est)
            dG_used_est = [dG_used_est,dG1];
            dN_used_est = [dN_used_est,dN];
            t_est_list = [t_est_list,t_est];
        end
        % same for t_fix
        if isnumeric(t_fix)
            dG_used_fix = [dG_used_fix,dG1];
            dN_used_fix = [dN_used_fix,dN];
            t_fix_list = [t_fix_list,t_fix];
        end
    end
end

function plot_simulation_results(t,S,N,G1,G2,dN_dt,legend_set,log_bool, savefig_bool, file_name)
    % Plots the results of a given simulation
    % Inputs:
    % [list] 1D lists containing results for different simulations, [set]
    % legend string for different kN kG values (legend_set), bool to use
    % loglog instead of regular plots (log_bool), bool to save figure to a
    % file (savefig_bool), [str] name of file with or without extension
    % (default is .fig)

    % Create figure and subplots, for each population plot all results on the same subplot
    fig = figure;
    figure('DefaultAxesFontSize',28);

    subplot(2,3,1);
    % plot each of the simulation results in the input lists
    for i = 1:length(S(1,:))
        % plot in loglog
        if log_bool
            % for different simulation sets, use different line formats
            if i == 1
                loglog(t(:,i),S(:,i),'LineWidth',4);
            elseif i == 2
                loglog(t(:,i),S(:,i), '-.','LineWidth',4);
            elseif i == 3
                loglog(t(:,i),S(:,i),'--','LineWidth',4);
            end
        else
            if i == 1
                plot(t(:,i),S(:,i),'LineWidth',4);
            elseif i == 2
                plot(t(:,i),S(:,i), '-.','LineWidth',4);
            elseif i == 3
                plot(t(:,i),S(:,i),'--','LineWidth',4);
            end
        end
        hold on
    end
    hold off
    % axis labels and title
    xlabel('t (hr)'); ylabel('S(t) (cells/mL)');

    xlim([10^-2 10^3]);
    ylim([10^-5 10^9]);
    xticks([ 10^-2 10^0 10^3])
    xticklabels({'10^{-2}','10^{0}','10^{3}'})

    ttl = title('A)');
    ttl.Units = 'Normalize';
    ttl.Position(1) = -0.36;
    ttl.HorizontalAlignment = 'left';

%      subplot(2,3,2);
% %      %plot(0,0,  0,0,  0,0, 'LineWidth', 2);
% %      plot(0,0,'LineWidth',3);
% %      hold on
% %      plot(0,0, '-.','LineWidth',3);
% %      hold on
% %      plot(0,0,'--','LineWidth',3);
% %      hold off
% %      axis off;
%      ttl = title('A)');
%      ttl.Units = 'Normalize';
%      ttl.Position(1) = -0.33;
%      ttl.HorizontalAlignment = 'left';
    % plot legend
    l = legend(legend_set, 'Location', 'southwest');
    l.FontSize = 18;

    subplot(2,3,2);
    % same comments as above
    for i = 1:length(N(1,:))
        if log_bool
            if i == 1
                loglog(t(:,i),N(:,i),'LineWidth',4);
            elseif i == 2
                loglog(t(:,i),N(:,i), '-.','LineWidth',4);
            elseif i == 3
                loglog(t(:,i),N(:,i),'--','LineWidth',4);
            end
        else
            if i == 1
                plot(t(:,i),N(:,i),'LineWidth',4);
            elseif i == 2
                plot(t(:,i),N(:,i), '-.','LineWidth',4);
            elseif i == 3
                plot(t(:,i),N(:,i),'--','LineWidth',4);
            end
        end
        hold on
    end
    hold off
    xlabel('t (hr)'); ylabel('N(t) (cells/mL)');

    xlim([10^-2 10^3]);
    ylim([10^-5 10^9]);

    xticks([ 10^-2 10^0 10^3])
    xticklabels({'10^{-2}','10^{0}','10^{3}'})

    ttl = title('B)');
    ttl.Units = 'Normalize';
    ttl.Position(1) = -0.35;
    ttl.HorizontalAlignment = 'left';
    %title('N')

    subplot(2,3,3);
    for i = 1:length(G1(1,:))
        if log_bool
            if i == 1
                loglog(t(:,i),G1(:,i),'LineWidth',4);
            elseif i == 2
                loglog(t(:,i),G1(:,i), '-.','LineWidth',4);
            elseif i == 3
                loglog(t(:,i),G1(:,i),'--','LineWidth',4);
            end
        else
            if i == 1
                plot(t(:,i),G1(:,i),'LineWidth',4);
            elseif i == 2
                plot(t(:,i),G1(:,i), '-.','LineWidth',4);
            elseif i == 3
                plot(t(:,i),G1(:,i),'--','LineWidth',4);
            end
        end
        hold on
    end
    hold off
    xlabel('t (hr)'); ylabel('G(t) (cells/mL)');

    xlim([10^-2 10^3]);
    ylim([10^-5 10^9]);
%
    xticks([ 10^-2 10^0 10^3])
    xticklabels({'10^{-2}','10^{0}','10^{3}'})

    ttl = title('C)');
    ttl.Units = 'Normalize';
    ttl.Position(1) = -0.35;
    ttl.HorizontalAlignment = 'left';
    %title('G')

    if isnumeric(G2)
        subplot(2,3,5);
        for i = 1:length(G2(1,:))
            if log_bool
                if i == 1
                    loglog(t(:,i),G2(:,i),'LineWidth',4);
                elseif i == 2
                    loglog(t(:,i),G2(:,i), '-.','LineWidth',4);
                elseif i == 3
                    loglog(t(:,i),G2(:,i),'--','LineWidth',4);
                end
            else
                if i == 1
                    plot(t(:,i),G2(:,i),'LineWidth',4);
                elseif i == 2
                    plot(t(:,i),G2(:,i), '-.','LineWidth',4);
                elseif i == 3
                    plot(t(:,i),G2(:,i),'--','LineWidth',4);
                end
            end
            hold on
        end
        hold off
        xlabel('t (hr)'); ylabel('G_2(t) (cells/mL)');
        %title('G_2')
    end

    subplot(2,3,4);
    for i = 1:length(G1(1,:))
        T = S(:,i) + N(:,i) + G1(:,i);
        if log_bool
            if i == 1
                loglog(t(:,i),T,'LineWidth',4);
            elseif i == 2
                loglog(t(:,i),T, '-.','LineWidth',4);
            elseif i == 3
                loglog(t(:,i),T,'--','LineWidth',4);
            end
        else
            if i == 1
                plot(t(:,i),T,'LineWidth',4);
            elseif i == 2
                plot(t(:,i),T, '-.','LineWidth',4);
            elseif i == 3
                plot(t(:,i),T,'--','LineWidth',4);
            end
        end
        hold on
    end
    hold off
    xlabel('t (hr)'); ylabel('T(t) (cells/mL)');

    xlim([10^-2 10^3]);
    ylim([10^-5 10^9]);

    xticks([ 10^-2 10^0 10^3])
    xticklabels({'10^{-2}','10^{0}','10^{3}'})
    ttl = title('D)');
    ttl.Units = 'Normalize';
    ttl.Position(1) = -0.35;
    ttl.HorizontalAlignment = 'left';

    subplot(2,3,5);
    for i = 1:length(G1(1,:))
        if isnumeric(G2)
            ratio = G2(:,i)./N(:,i);
            ratio_string = 'G_2(t)/N(t)';
            ratio_title_string = 'G_2/N';
        else
            ratio = G1(:,i)./(S(:,i) + N(:,i) + G1(:,i));
            ratio_string = 'G(t)/T(t)';
            ratio_title_string = 'G/T Ratio';

        end
        if log_bool
            if i == 1
                loglog(t(:,i),ratio,'LineWidth',4);
            elseif i == 2
                loglog(t(:,i),ratio, '-.','LineWidth',4);
            elseif i == 3
                loglog(t(:,i),ratio,'--','LineWidth',4);
            end
        else
            if i == 1
                plot(t(:,i),ratio,'LineWidth',4);
            elseif i == 2
                plot(t(:,i),ratio, '-.','LineWidth',4);
            elseif i == 3
                plot(t(:,i),ratio,'--','LineWidth',4);
            end
        end
        hold on
    end
    hold off
    xlabel('t (hr)'); ylabel(ratio_string);

    xlim([10^-2 10^3]);
    ylim([10^-10 10^1]);

    xticks([ 10^-2 10^0 10^3])
    xticklabels({'10^{-2}','10^{0}','10^{3}'})


    ttl = title('E)');
    ttl.Units = 'Normalize';
    ttl.Position(1) = -0.39;
    ttl.HorizontalAlignment = 'left';
    %title(ratio_title_string)

    subplot(2,3,6);
    for i = 1:length(dN_dt(1,:))
        if i == 1
            semilogx(t(:,i),dN_dt(:,i),'LineWidth',2);
        elseif i == 2
            semilogx(t(:,i),dN_dt(:,i), '-.','LineWidth',2);
        elseif i == 3
            semilogx(t(:,i),dN_dt(:,i),'--','LineWidth',2);
        end
%         if log_bool
%             if i == 1
%                 loglog(t(:,i),dN_dt(:,i),'LineWidth',2);
%             elseif i == 2
%                 loglog(t(:,i),dN_dt(:,i), '-.','LineWidth',2);
%             elseif i == 3
%                 loglog(t(:,i),dN_dt(:,i),'--','LineWidth',2);
%             end
%         else
%             if i == 1
%                 plot(t(:,i),dN_dt(:,i),'LineWidth',2);
%             elseif i == 2
%                 plot(t(:,i),dN_dt(:,i), '-.','LineWidth',2);
%             elseif i == 3
%                 plot(t(:,i),dN_dt(:,i),'--','LineWidth',2);
%             end
%         end
        hold on
    end
    hold off
    xlabel('t (hr)'); ylabel('dT/dt (cells/mLhr)');

    xlim([10^-2 10^3]);
%
%
    xticks([ 10^-2 10^0 10^3])
    xticklabels({'10^{-2}','10^{0}','10^{3}'})

    ttl = title('F)');
    ttl.Units = 'Normalize';
    ttl.Position(1) = -0.35;
    ttl.HorizontalAlignment = 'left';
    %title('dT/dt')


    set(gcf, 'PaperUnits', 'inches');
    x_width=18 ;y_width=12;
    set(gcf, 'PaperPosition', [0 0 x_width y_width]);
    orient(fig,'landscape');
    if savefig_bool
        saveas(gcf, append(file_name, '.png'))
        %save(append(file_name, '.mat'),'t','S','N','G1','G2','dN_dt')
    end
end


function plot_population_times(kG_list, kN_list, population_time_list, population_type, scenario,↙
    ↳  savefig_bool, file_name)
    % Plot heatmaps using the kG, kN and t_est or t_fix results for a large
    % amount of simulations.
    % Input:
    % [list] lists of the kG, kN used and t_est or t_fix results, [str]
    % quantitative measure used ("Establishment" or "Fixation") for plot
    % titles (population_type), [int] scenario simulated for plot titles
    % and file names(scenario), [bool] to save figure or not
    % (savefig_bool),[str] name of file to save (with or without file type extension, default is ↙
        ↳ .fig if there is none)

    % create figure
    fig = figure;

    % put input lists into a table to make the heatmaps with
    X = kG_list(:); Y = kN_list(:); Z = population_time_list(:);
    tbl = table(X,Y,Z);

    % create the heatmap
    hHM = heatmap(tbl,'X', 'Y', 'ColorVariable','Z','ColorMethod', 'none', 'CellLabelColor','none↙
        ↳ ',...
                  'GridVisible','off','MissingDataLabel', 'No Data', 'MissingDataColor','w', ...
                  'FontSize', 20);%,'ColorScaling','log');
    hHM.NodeChildren(3).YDir='normal'; % flips the y-axis to make it increasing (matlab default ↙
        ↳ is decreasing)
    hHM.Colormap = jet;


    % set axis labels and title
    ylabel('k_N (/hr)')
    if scenario == 2
        xlabel('k_{G2} (/hr)')
        title('')%append(population_type,' Time as a Function of k_N and k_{G2} (k_N <= k_{G2})')↙
            ↳ )
    else
        xlabel('k_{G} (/hr)')
        title('')%append(population_type,' Time as a Function of k_N and k_{G} (k_N <= k_{G})'))
    end
    colorbar; % adds a color bar for the heatmap

    set(gcf, 'PaperUnits', 'inches');
    x_width=10 ;y_width=7;
    set(gcf, 'PaperPosition', [0 0 x_width y_width]);

    if population_type == "Establishment"
        annotation('textarrow',[0.98,0.98],[0.7,0.7],'string','{\it \tau_{est}} (hr)', 'FontSize'↙
            ↳ , 22, ...
            'HeadStyle','none','LineStyle','none','HorizontalAlignment','right','TextRotation'↙
                ↳ ,90);
        annotation('textarrow',[0.025,0.025],[0.97,0.97],'string','A)', 'FontWeight', 'bold', ...
            'FontSize', 22, 'HeadStyle','none','LineStyle','none','HorizontalAlignment','center')↙
                ↳ ;
    else
        annotation('textarrow',[0.98,0.98],[0.7,0.7],'string','{\it \tau_{fix}} (hr)', 'FontSize'↙
            ↳ , 22, ...
            'HeadStyle','none','LineStyle','none','HorizontalAlignment','right','TextRotation'↙
                ↳ ,90);
        annotation('textarrow',[0.025,0.025],[0.97,0.97],'string','A)', 'FontWeight', 'bold', ...
            'FontSize', 22, 'HeadStyle','none','LineStyle','none','HorizontalAlignment','center')↙
                ↳ ;
    end

    % rotate x-axis labels
    s = struct(hHM);
    s.XAxis.TickLabelRotation = 60;  % angled
    s.XDisplayLabels = compose('%.3f',str2double(hHM.XDisplayLabels));
    s.YDisplayLabels = compose('%.4f',str2double(hHM.YDisplayLabels));

    %s.FontSize = 60;
    % save the file
    if savefig_bool
        saveas(gcf, append(file_name,'.png'))
        %save(append(file_name, '.mat'), 'kG_list', 'kN_list', 'population_time_list')
    end
    close(fig)
end

function plot_population_times_delta(dG_list, dN_list, population_time_list, population_type, ↙
    ↳ kN_dsweep, kG_dsweep, savefig_bool, file_name)
    % create figure
    fig = figure;

    % put input lists into a table to make the heatmaps with
    X = dG_list(:); Y = dN_list(:); Z = population_time_list(:);
    tbl = table(X,Y,Z);

    % create the heatmap
    hHM = heatmap(tbl,'X', 'Y', 'ColorVariable','Z','ColorMethod', 'none', 'CellLabelColor','none↙
        ↳ ',...
                 'GridVisible','off','MissingDataLabel', 'No Data', 'MissingDataColor','w', ...
                 'FontSize', 20);
    hHM.NodeChildren(3).YDir='normal'; % flips the y-axis to make it increasing (matlab default ↙
        ↳ is decreasing)
    hHM.Colormap = jet;


    % set axis labels and title
    ylabel('\delta_N (/hr)')
    xlabel('\delta_G (/hr)')
    title('')

    colorbar; % adds a color bar for the heatmap

    set(gcf, 'PaperUnits', 'inches');
    x_width=10 ;y_width=7.2;
    set(gcf, 'PaperPosition', [0 0 x_width y_width]);

    if population_type == "Establishment"
        annotation('textarrow',[0.98,0.98],[0.7,0.7],'string','{\it \tau_{est}} (hr)', 'FontSize'↙
            ↳ , 22, ...
            'HeadStyle','none','LineStyle','none','HorizontalAlignment','right','TextRotation'↙
                ↳ ,90);
        annotation('textarrow',[0.025,0.025],[0.97,0.97],'string','A)', 'FontWeight', 'bold', ...
            'FontSize', 22, 'HeadStyle','none','LineStyle','none','HorizontalAlignment','center')↙
                ↳ ;
    else
        annotation('textarrow',[0.983,0.983],[0.7,0.7],'string','{\it \tau_{fix}} (hr)', '↙
            ↳ FontSize', 22, ...
            'HeadStyle','none','LineStyle','none','HorizontalAlignment','right','TextRotation'↙
                ↳ ,90);
        annotation('textarrow',[0.025,0.025],[0.97,0.97],'string','B)', 'FontWeight', 'bold', ...
            'FontSize', 22, 'HeadStyle','none','LineStyle','none','HorizontalAlignment','center')↙
                ↳ ;
    end

    % rotate x-axis labels
    s = struct(hHM);
    s.XAxis.TickLabelRotation = 60;  % angled
    s.XDisplayLabels = compose('%.3f',str2double(hHM.XDisplayLabels));
    s.YDisplayLabels = compose('%.4f',str2double(hHM.YDisplayLabels));



    %s.FontSize = 60;
    % save the file
    if savefig_bool
        saveas(gcf, append(file_name,'.png'))
        %save(append(file_name, '.mat'), 'kG_list', 'kN_list', 'population_time_list')
    end

    close(fig)
end
```

A.2 Stochastic Algorithms

A.2.1: C code used to simulate the reaction system presented in Chapter 2 using the implementation of the SSA presented in Algorithm 1 (Chapter 1) [45, 47] for the constant drug environment.

```
// Joshua Guthrie, Charlebois Laboratory, University of Alberta

/* REACTIONS
   *******************************************************************
   0: S ----> 2S       (parameter = k_S*z(N_tot))
   1: N ----> 2N       (parameter = k_N*z(N_tot))
   2: G1 ----> 2G1     (parameter = k_G1*z(N_tot))
   3: S ----> N        (parameter = r_NS)
   4: N ----> S        (parameter = r_SN)
   5: S ----> G1       (parameter = r_G1S)
   6: N ----> G1       (parameter = r_G1N)
   7: S ----> 0        (parameter = delta_S)
   8: N ----> 0       (parameter = delta_N)
   9: G1 ----> 0      (parameter = delta_G1)
   *******************************************************************
*/

#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>
#include <math.h>
#include <time.h>

int main(int argc, char *argv[]){
    /* Command line arguments:
       argv[1] = k_N [float]
       argv[2] = k_G1 [float]
       argv[3] = delta_S [float]
       argv[4] = delta_N [float]
       argv[5] = N_i [int]
       argv[6] = t_end [float]
       argv[7] = num_runs [int]
       argv[8] = distributions outfile name [str]
       argv[9] = number for sample trajectory to save (0 for no save) [int]
       argv[10] = sample trajectory outfile name [str]
       argv[11] = stop at G1 appearance, establishment, or fixation (1, 2, or 3, use 0 otherwise)↙
           ↳  [int]
       argv[12] = measure relative to initial or total population (0 for initial (default), 1 for↙
           ↳  total) [int]
    */

    int sample_traj = atoi(argv[9]);

    int rel_to = atoi(argv[12]);

    FILE *outfile_est_fix = fopen(argv[8], "w");
    FILE *outfile = NULL;
    if (sample_traj > 0) {
        outfile = fopen(argv[10], "w");
    }


    /* INITIALIZATION */

    // Barayni-Hill coefficients
    double k = 1e+7;
    double n = 2;

    // reaction parameters c_i (monomolecular reactions, k_i = c_i)
    float k_S = 0.0;
    float k_N = atof(argv[1]);
    float k_G1 = atof(argv[2]);
    float r_NS = 0.0625;
    float r_SN = 0.0035;
    float r_G1S = 0.0;
    float r_G1N = (1e-6)/3;
    float delta_S = atof(argv[3]);
    float delta_N = atof(argv[4]);
    float delta_G1 = 1/156;

    // state changes for each reaction
    int V[10][3] = {{+1, 0, 0},
                    {0, +1, 0},
                    {0, 0, +1},
                    {-1, +1, 0},
                    {+1, -1, 0},
                    {-1, 0, +1},
                    {0, -1, +1},
                    {-1, 0, 0},
                    {0, -1, 0},
                    {0, 0, -1}};


    float t_end = atof(argv[6]); // time to simulate, hours
    int num_runs = atoi(argv[7]); // Number of runs (trajectories)

    // initialize state (number of cells for each population) and time
    long int S = 550000000;
    long int N = atoi(argv[5]);
    long int G1 = 0;
    double t = 0.0; // hours

    // conservation equations
    long int N_tot = S + N + G1;
    double hill = pow(k,n)/(pow(k,n) + pow(N_tot,n));
    double dNtot_dt = k_S*hill*S + k_N*hill*N + k_G1*hill*G1 - delta_S*S - delta_N*N - delta_G1*↙
        ↳ G1;

    // Print header for appearance, establishment, fixation time output file
    fprintf(outfile_est_fix, "#Stochastic Simulation Algorithm Results (Establishment/Fixation ↙
        ↳ Times)\n#\n");

    fprintf(outfile_est_fix, "#Reaction Parameters (/hr)\n↙
        ↳ #----------------------------------------------------------------------\n");
    fprintf(outfile_est_fix, "#Growth Rates: k_S = %.4f, k_N = %.4f, k_G1 = %.4f\n", k_S, k_N, ↙
        ↳ k_G1);
    fprintf(outfile_est_fix, "#Switching Rates: r_NS = %.4f, r_SN = %.4f, r_G1S = %.4f, r_G1N = ↙
        ↳ %.4e\n", r_NS, r_SN, r_G1S, r_G1N);
    fprintf(outfile_est_fix, "#Death Rates: d_S = %0.1f, d_N = %0.1f, d_G1 = %0.1f\n", delta_S, ↙
        ↳ delta_N, delta_G1);
    fprintf(outfile_est_fix, "↙
        ↳ #----------------------------------------------------------------------\n#\n");

    fprintf(outfile_est_fix, "#Simulation Parameters\n↙
        ↳ #----------------------------------------------------------------------\n");
    fprintf(outfile_est_fix, "#Barayni-Hill: k = %.0e, n = %.0f\n", k, n);
    fprintf(outfile_est_fix, "#Time (hr): t_i = %0.1f, t_end = %0.1f\n", t, t_end);
    fprintf(outfile_est_fix, "#Initial State: S = %ld, N = %ld, G1 = %ld, N_tot = %ld, dN_tot/dt ↙
        ↳ = %0.10f\n", S, N, G1, N_tot, dNtot_dt);
    fprintf(outfile_est_fix, "↙
        ↳ #----------------------------------------------------------------------\n#\n");
    fprintf(outfile_est_fix, "#ESTABLISHMENT/FIXATION TIMES (if time is negative it wasn't found ↙
        ↳ within t_end)\n");
    fprintf(outfile_est_fix, "#Measured relative to: %d\n", rel_to);
    fprintf(outfile_est_fix, "#Columns: Run (Trajectory) Number, G1 t_appearance, G1 t_est, G1 ↙
        ↳ t_fix\n#----------------------------------------------------------------------\n");

    if ((outfile != NULL)&&(sample_traj > 0)){
        // Print header for data file
        fprintf(outfile, "#Stochastic Simulation Algorithm Results\n#\n");

        fprintf(outfile, "#Reaction Parameters (/hr)\n↙
            ↳ #----------------------------------------------------------------------\n");
        fprintf(outfile, "#Growth Rates: k_S = %.4f, k_N = %.4f, k_G1 = %.4f\n", k_S, k_N, k_G1);
        fprintf(outfile, "#Switching Rates: r_NS = %.4f, r_SN = %.4f, r_G1S = %.4f, r_G1N = %.4e\↙
            ↳ n", r_NS, r_SN, r_G1S, r_G1N);
        fprintf(outfile, "#Death Rates: d_S = %0.1f, d_N = %0.1f, d_G1 = %0.1f\n", delta_S, ↙
            ↳ delta_N, delta_G1);
        fprintf(outfile, "↙
            ↳ #----------------------------------------------------------------------\n#\n");

        fprintf(outfile, "#Simulation Parameters\n↙
            ↳ #----------------------------------------------------------------------\n");
        fprintf(outfile, "#Barayni-Hill: k = %.0e, n = %.0f\n", k, n);
        fprintf(outfile, "#Time (hr): t_i = %0.1f, t_end = %0.1f\n", t, t_end);
        fprintf(outfile, "#Initial State: S = %ld, N = %ld, G1 = %ld, N_tot = %ld, dN_tot/dt = ↙
            ↳ %0.10f\n", S, N, G1, N_tot, dNtot_dt);
        fprintf(outfile, "↙
            ↳ #----------------------------------------------------------------------\n#\n");

        fprintf(outfile, "#DATA (Columns: Row Number, Time, S, N, G1, N_tot, dN_tot/dt)\n↙
            ↳ #----------------------------------------------------------------------\n");
        fprintf(outfile, "%d    %0.20f    %ld    %ld    %ld    %ld    %0.20f\n", 0, t, S, N, G1, ↙
            ↳ N_tot, dNtot_dt);
    }


    srand(time(NULL)); // initialize random number generator

    int stop_at = atoi(argv[11]);
    for (int i = 1; i <= num_runs; i++){
        printf("Trajectory %d running...\n", i);

        // initialize state (number of cells for each population) and time
        long int S = 550000000;
        long int N = atoi(argv[5]);
        long int G1 = 0;

        double t = 0.0; // hours
        long int N_0 = S + N + G1;

        // conservation equations
        long int N_tot = S + N + G1;

        double hill = pow(k,n)/(pow(k,n) + pow(N_tot,n));
        double dNtot_dt = k_S*hill*S + k_N*hill*N + k_G1*hill*G1 - delta_S*S - delta_N*N - ↙
            ↳ delta_G1*G1;

        // loop control bools
        bool G1_app_found = false;
        bool G1_est_found = false;
        bool G1_fix_found = false;


        // initialize establishment and fixation times (-1.0 is a dummy value)
        double G1_app = -1.0;
        double G1_est = -1.0;
        double G1_fix = -1.0;

        float G1_frac = -1.0;
        if (i == sample_traj){
            long int row_num = 1;
            while (t < t_end){
                /* Runs until t_end */

                // STEP 1: calculate propensity functions and their sum a_0
                double a[10] = {(k_S*hill)*S, (k_N*hill)*N, (k_G1*hill)*G1,
                                r_NS*S, r_SN*N, r_G1S*S, r_G1N*N,
                                delta_S*S, delta_N*N, delta_G1*G1};

                double a_0 = 0.0;
                for (int i = 0; i < 10; i++) a_0 += a[i];

                // STEP 2: generate uniform random numbers in unit interval, calculate tau and j
                double r1 = (double)rand() / (double)RAND_MAX;
                while (r1 == 0.0) r1 = (double)rand() / (double)RAND_MAX;

                double tau = (1/a_0)*log(1/r1);

                double r2 = (double)rand() / (double)RAND_MAX;
                int j = 0;
                double a_sum = a[j];
                while (a_sum < r2*a_0){
                    j += 1;
                    a_sum += a[j];
                }

                // STEP 3: Update
                t += tau;

                S += V[j][0];
                N += V[j][1];
                G1 += V[j][2];

                N_tot = S + N + G1;
                hill = pow(k,n)/(pow(k,n) + pow(N_tot,n));
                dNtot_dt = k_S*hill*S + k_N*hill*N + k_G1*hill*G1 − delta_S*S − delta_N*N − ↙
                    ↳ delta_G1*G1;

                // print out the sample trajectory results
                fprintf(outfile, "%ld    %0.20f    %ld    %ld    %ld    %ld    %0.20f\n", row_num↙
                    ↳ , t, S, N, G1, N_tot, dNtot_dt);

                // check for establishment and fixation for the G populations

                if (rel_to == 1){
                    G1_frac = (float) G1/N_tot;
                }
                else {
                    G1_frac = (float) G1/N_0;
                }

                // Check G1 and G2 first appearance
                if ((G1_app_found == false) && (G1 > 0)){
                    G1_app = t;
                    G1_app_found = true;
                }

                // Check G1 t_est and t_fix
                if ((G1_est_found == false) && (G1_frac > 0.05)){
                    G1_est = t;
                    G1_est_found = true;
                }
                else if ((G1_fix_found == false) && (G1_frac > 0.95)){
                    G1_fix = t;
                    G1_fix_found = true;
                }

                row_num++;
            }
        }
        else {
            while (t < t_end){
                /* Runs until t_end OR G2 fixates OR until argv[11] parameter is found */

                // STEP 1: calculate propensity functions and their sum a_0
                double a[10] = {(k_S*hill)*S, (k_N*hill)*N, (k_G1*hill)*G1,
                                r_NS*S, r_SN*N, r_G1S*S, r_G1N*N,
                                delta_S*S, delta_N*N, delta_G1*G1};

                double a_0 = 0.0;
                for (int i = 0; i < 10; i++) a_0 += a[i];

                // STEP 2: generate uniform random numbers in unit interval, calculate tau and j
                double r1 = (double)rand() / (double)RAND_MAX;
                while (r1 == 0.0) r1 = (double)rand() / (double)RAND_MAX;

                double tau = (1/a_0)*log(1/r1);

                double r2 = (double)rand() / (double)RAND_MAX;
                int j = 0;
                double a_sum = a[j];
                while (a_sum < r2*a_0){
                    j += 1;
                    a_sum += a[j];
                }

                // STEP 3: Update
                t += tau;

                S += V[j][0];
                N += V[j][1];
                G1 += V[j][2];

                N_tot = S + N + G1;
                hill = pow(k,n)/(pow(k,n) + pow(N_tot,n));

                // check for establishment and fixation for the G populations
                if (rel_to == 1){
                    G1_frac = (float) G1/N_tot;
                }
                else {
                    G1_frac = (float) G1/N_0;
                }

                // Check G1 and G2 first appearance
                if ((G1_app_found == false) && (G1 > 0)){
                    G1_app = t;
                    G1_app_found = true;
                    if (stop_at == 1) break;
                }

                // Check G1 t_est and t_fix
                if ((G1_est_found == false) && (G1_frac > 0.05)){
                    G1_est = t;
                    G1_est_found = true;
                    if (stop_at == 2) break;
                }
                else if ((G1_fix_found == false) && (G1_frac > 0.95)){
                    G1_fix = t;
                    G1_fix_found = true;
                    break;
                }
            }
        }
        fprintf(outfile_est_fix, "%d    %0.20f    %0.20f    %0.20f\n", i, G1_app, G1_est, G1_fix)
            ;
        printf("Results: G1_app = %0.20f, G1_est = %0.20f, G1_fix = %0.20f\n\n", G1_app, G1_est,
            G1_fix);
    }

    fclose(outfile_est_fix);
    if (outfile != NULL) fclose(outfile);
    return 0;
}
```

A.2.2: C code used to simulate the reaction system presented in Chapter 2 using the implementation of the SSA presented in Algorithm 1 (Chapter 1) [45, 47] for the fluctuating drug environment.

```
// Joshua Guthrie, Charlebois Laboratory, University of Alberta

/* REACTIONS
   *******************************************************************
   0: S ----> 2S      (parameter = k_S*z(N_tot))
   1: N ----> 2N      (parameter = k_N*z(N_tot))
   2: G1 ----> 2G1    (parameter = k_G1*z(N_tot))
   3: S ----> N       (parameter = r_NS)
   4: N ----> S       (parameter = r_SN)
   5: S ----> G1      (parameter = r_G1S)
   6: N ----> G1      (parameter = r_G1N)
   7: S ----> 0       (parameter = delta_S)
   8: N ----> 0      (parameter = delta_N)
   9: G1 ----> 0     (parameter = delta_G1)
   *******************************************************************
*/

#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>
#include <math.h>
#include <time.h>

int main(int argc, char *argv[]){
    /* Command line arguments:
       argv[1] = k_N [float]
       argv[2] = k_G1 [float]
       argv[3] = delta_S [float]
       argv[4] = delta_N [float]
    argv[5] = N_i [int]
    argv[6] = t_end [float]
    argv[7] = num_runs [int]
    argv[8] = distributions outfile name [str]
    argv[9] = number for sample trajectory to save (0 for no save) [int]
    argv[10] = sample trajectory outfile name [str]
    argv[11] = stop at G1 appearance, establishment, or fixation (1, 2, or 3, use 0 otherwise)↙
        ↳ [int]
    argv[12] = measure relative to initial or total population (0 for initial (default), 1 for↙
        ↳ total) [int]
*/

int sample_traj = atoi(argv[9]);

int rel_to = atoi(argv[12]);

FILE *outfile_est_fix = fopen(argv[8], "w");
FILE *outfile = NULL;
if (sample_traj > 0) {
    outfile = fopen(argv[10], "w");
}


/* INITIALIZATION */

// Barayni-Hill coefficients
double k = 1e+7;
double n = 2;

// reaction parameters c_i (monomolecular reactions, k_i = c_i)
float k_S = 0.0;
float k_N = atof(argv[1]);
float k_G1 = atof(argv[2]);
float r_NS = 0.0625;
float r_SN = 0.0035;
float r_G1S = 0.0;
float r_G1N = (1e-6)/3;
float delta_S = atof(argv[3]);
float delta_N = atof(argv[4]);
float delta_G1 = 1/156;

// state changes for each reaction
int V[10][3] = {{+1, 0, 0},
                {0, +1, 0},
                {0, 0, +1},
                {-1, +1, 0},
                {+1, -1, 0},
                {-1, 0, +1},
                {0, -1, +1},
                {-1, 0, 0},
                {0, -1, 0},
                {0, 0, -1}};


float t_end = atof(argv[6]); // time to simulate, hours
int num_runs = atoi(argv[7]); // Number of runs (trajectories)

// initialize state (number of cells for each population) and time
long int S = 550000;
long int N = atoi(argv[5]);
long int G1 = 0;
double t = 0.0; // hours

// conservation equations
long int N_tot = S + N + G1;
double hill = pow(k,n)/(pow(k,n) + pow(N_tot,n));
double dNtot_dt = k_S*hill*S + k_N*hill*N + k_G1*hill*G1 - delta_S*S - delta_N*N - delta_G1*↙
    ↳ G1;

// Print header for appearance, establishment, fixation time output file
fprintf(outfile_est_fix, "#Stochastic Simulation Algorithm Results (Establishment/Fixation ↙
    ↳ Times)\n#\n");

fprintf(outfile_est_fix, "#Reaction Parameters (/hr)\n↙
    ↳ #--------------------------------------------------------------------\n");
fprintf(outfile_est_fix, "#Growth Rates: k_S = %.4f, k_N = %.4f, k_G1 = %.4f\n", k_S, k_N, ↙
    ↳ k_G1);
fprintf(outfile_est_fix, "#Switching Rates: r_NS = %.4f, r_SN = %.4f, r_G1S = %.4f, r_G1N = ↙
    ↳ %.4e\n", r_NS, r_SN, r_G1S, r_G1N);
fprintf(outfile_est_fix, "#Death Rates: d_S = %0.1f, d_N = %0.1f, d_G1 = %0.1f\n", delta_S, ↙
    ↳ delta_N, delta_G1);
fprintf(outfile_est_fix, "↙
    ↳ #--------------------------------------------------------------------\n#\n");

fprintf(outfile_est_fix, "#Simulation Parameters\n↙
    ↳ #--------------------------------------------------------------------\n");
fprintf(outfile_est_fix, "#Barayni-Hill: k = %.0e, n = %.0f\n", k, n);
fprintf(outfile_est_fix, "#Time (hr): t_i = %0.1f, t_end = %0.1f\n", t, t_end);
fprintf(outfile_est_fix, "#Initial State: S = %ld, N = %ld, G1 = %ld, N_tot = %ld, dN_tot/dt ↙
    ↳ = %0.10f\n", S, N, G1, N_tot, dNtot_dt);
fprintf(outfile_est_fix, "↙
    ↳ #--------------------------------------------------------------------\n#\n");
    fprintf(outfile_est_fix, "#ESTABLISHMENT/FIXATION TIMES (if time is negative it wasn't found ↙
        ↪ within t_end)\n");
    fprintf(outfile_est_fix, "#Measured relative to: %d\n", rel_to);
    fprintf(outfile_est_fix, "#Columns: Run (Trajectory) Number, G1 t_appearance, G1 t_est, G1 ↙
        ↪ t_fix\n#-----------------------------------------------------------------------\n");

    if ((outfile != NULL)&&(sample_traj > 0)){
        // Print header for data file
        fprintf(outfile, "#Stochastic Simulation Algorithm Results\n#\n");

        fprintf(outfile, "#Reaction Parameters (/hr)\n↙
            ↪ #-----------------------------------------------------------------------\n");
        fprintf(outfile, "#Growth Rates: k_S = %.4f, k_N = %.4f, k_G1 = %.4f\n", k_S, k_N, k_G1);
        fprintf(outfile, "#Switching Rates: r_NS = %.4f, r_SN = %.4f, r_G1S = %.4f, r_G1N = %.4e\↙
            ↪ n", r_NS, r_SN, r_G1S, r_G1N);
        fprintf(outfile, "#Death Rates: d_S = %0.1f, d_N = %0.1f, d_G1 = %0.1f\n", delta_S, ↙
            ↪ delta_N, delta_G1);
        fprintf(outfile, "↙
            ↪ #-----------------------------------------------------------------------\n#\n");

        fprintf(outfile, "#Simulation Parameters\n↙
            ↪ #-----------------------------------------------------------------------\n");
        fprintf(outfile, "#Barayni-Hill: k = %.0e, n = %.0f\n", k, n);
        fprintf(outfile, "#Time (hr): t_i = %0.1f, t_end = %0.1f\n", t, t_end);
        fprintf(outfile, "#Initial State: S = %ld, N = %ld, G1 = %ld, N_tot = %ld, dN_tot/dt = ↙
            ↪ %0.10f\n", S, N, G1, N_tot, dNtot_dt);
        fprintf(outfile, "↙
            ↪ #-----------------------------------------------------------------------\n#\n");

        fprintf(outfile, "#DATA (Columns: Row Number, Time, S, N, G1, N_tot, dN_tot/dt)\n↙
            ↪ #-----------------------------------------------------------------------\n");
        fprintf(outfile, "%d    %0.20f    %ld    %ld    %ld    %ld    %0.20f\n", 0, t, S, N, G1, ↙
            ↪ N_tot, dNtot_dt);
    }


    srand(time(NULL)); // initialize random number generator

    int stop_at = atoi(argv[11]);
    for (int i = 1; i <= num_runs; i++){
        printf("Trajectory %d running...\n", i);

        // initialize state (number of cells for each population) and time
        long int S = 550000;
        long int N = atoi(argv[5]);
        long int G1 = 0;

        double t = 0.0; // hours
        long int N_0 = S + N + G1;

        // conservation equations
        long int N_tot = S + N + G1;

        double hill = pow(k,n)/(pow(k,n) + pow(N_tot,n));
        double dNtot_dt = k_S*hill*S + k_N*hill*N + k_G1*hill*G1 - delta_S*S - delta_N*N - ↙
            ↪ delta_G1*G1;

        // loop control bools
        bool G1_app_found = false;
        bool G1_est_found = false;
        bool G1_fix_found = false;


        // initialize establishment and fixation times (-1.0 is a dummy value)
        double G1_app = -1.0;
        double G1_est = -1.0;
        double G1_fix = -1.0;

        float G1_frac = -1.0;


        if (i == sample_traj){
            long int row_num = 1;
            while (t < t_end){
                /* Runs until t_end */
                // fluctuate the drug
                if (((t > 24.0)&&(t <= 48.0))||((t > 72.0)&&(t <= 96.0))||((t > 120.0)&&(t <= ↙
                    ↪ 144.0))||((t > 168.0)&&(t <= 192.0))
                    ||((t > 216.0)&&(t <= 240.0))||((t > 264.0)&&(t <= 288.0))||((t > 312.0)&&(t ↙
                        ↪ <= 336.0))||((t > 360.0)&&(t <= 384.0))
                    ||((t > 408.0)&&(t <= 432.0))||((t > 456.0)&&(t <= 480.0))){
                    // no drug condition
                    k_S = 0.3466;
                    k_N = 0.2600;
                    k_G1 = 0.1733;
                    r_NS = 0.0625;
                    r_SN = 0.0035;
                    r_G1S = (1e-6)/3;
                    r_G1N = (1e-6)/3;
                    delta_S = 1/156;
                    delta_N = 1/156;
                    delta_G1 = 1/156;
                    }
                else {
                    // drug
                    k_S = 0.0;
                    k_N = atof(argv[1]);
                    k_G1 = atof(argv[2]);
                    r_NS = 0.0625;
                    r_SN = 0.0035;
                    r_G1S = 0.0;
                    r_G1N = (1e-6)/3;
                    delta_S = atof(argv[3]);
                    delta_N = atof(argv[4]);
                    delta_G1 = 1/156;
                }

                // STEP 1: calculate propensity functions and their sum a_0
                double a[10] = {(k_S*hill)*S, (k_N*hill)*N, (k_G1*hill)*G1,
                                r_NS*S, r_SN*N, r_G1S*S, r_G1N*N,
                                delta_S*S, delta_N*N, delta_G1*G1};

                double a_0 = 0.0;
                for (int i = 0; i < 10; i++) a_0 += a[i];

                // STEP 2: generate uniform random numbers in unit interval, calculate tau and j
                double r1 = (double)rand() / (double)RAND_MAX;
                while (r1 == 0.0) r1 = (double)rand() / (double)RAND_MAX;

                double tau = (1/a_0)*log(1/r1);

                double r2 = (double)rand() / (double)RAND_MAX;
                int j = 0;
                double a_sum = a[j];
                while (a_sum < r2*a_0){
                    j += 1;
                    a_sum += a[j];
                }

                // STEP 3: Update
                t += tau;

                S += V[j][0];
                N += V[j][1];
                G1 += V[j][2];

                N_tot = S + N + G1;
                hill = pow(k,n)/(pow(k,n) + pow(N_tot,n));
                dNtot_dt = k_S*hill*S + k_N*hill*N + k_G1*hill*G1 - delta_S*S - delta_N*N - ↙
                    ↪ delta_G1*G1;

                // print out the sample trajectory results
                fprintf(outfile, "%ld    %0.20f    %ld    %ld    %ld    %ld    %0.20f\n", row_num↙
                    ↪ , t, S, N, G1, N_tot, dNtot_dt);

                // check for establishment and fixation for the G populations

                if (rel_to == 1){
                    G1_frac = (float) G1/N_tot;
                }
                else {
                    G1_frac = (float) G1/N_0;
                }

                // Check G1 and G2 first appearance
                if ((G1_app_found == false) && (G1 > 0)){
                    G1_app = t;
                    G1_app_found = true;
                }

                // Check G1 t_est and t_fix
                if ((G1_est_found == false) && (G1_frac > 0.05)){
                    G1_est = t;
                    G1_est_found = true;
                }
                else if ((G1_fix_found == false) && (G1_frac > 0.95)){
                    G1_fix = t;
                    G1_fix_found = true;
                }

                row_num++;
            }
        }
        else {
            while (t < t_end){
                /* Runs until t_end OR G2 fixates OR until argv[11] parameter is found */

                // fluctuate the drug
                if (((t > 24.0)&&(t <= 48.0))||((t > 72.0)&&(t <= 96.0))||((t > 120.0)&&(t <= ↙
                    ↪ 144.0))||((t > 168.0)&&(t <= 192.0))
                    ||((t > 216.0)&&(t <= 240.0))||((t > 264.0)&&(t <= 288.0))||((t > 312.0)&&(t ↙
                        ↪ <= 336.0))||((t > 360.0)&&(t <= 384.0))
                    ||((t > 408.0)&&(t <= 432.0))||((t > 456.0)&&(t <= 480.0))){
                    // no drug condition
                k_S = 0.3466;
                k_N = 0.2600;
                k_G1 = 0.1733;
                r_NS = 0.0625;
                r_SN = 0.0035;
                r_G1S = (1e-6)/3;
                r_G1N = (1e-6)/3;
                delta_S = 1/156;
                delta_N = 1/156;
                delta_G1 = 1/156;
                }
            else {
                // drug
                k_S = 0.0;
                k_N = atof(argv[1]);
                k_G1 = atof(argv[2]);
                r_NS = 0.0625;
                r_SN = 0.0035;
                r_G1S = 0.0;
                r_G1N = (1e-6)/3;
                delta_S = atof(argv[3]);
                delta_N = atof(argv[4]);
                delta_G1 = 1/156;
            }

            // STEP 1: calculate propensity functions and their sum a_0
            double a[10] = {(k_S*hill)*S, (k_N*hill)*N, (k_G1*hill)*G1,
                            r_NS*S, r_SN*N, r_G1S*S, r_G1N*N,
                            delta_S*S, delta_N*N, delta_G1*G1};

            double a_0 = 0.0;
            for (int i = 0; i < 10; i++) a_0 += a[i];

            // STEP 2: generate uniform random numbers in unit interval, calculate tau and j
            double r1 = (double)rand() / (double)RAND_MAX;
            while (r1 == 0.0) r1 = (double)rand() / (double)RAND_MAX;

            double tau = (1/a_0)*log(1/r1);

            double r2 = (double)rand() / (double)RAND_MAX;
            int j = 0;
            double a_sum = a[j];
            while (a_sum < r2*a_0){
                j += 1;
                a_sum += a[j];
            }

            // STEP 3: Update
            t += tau;

            S += V[j][0];
            N += V[j][1];
            G1 += V[j][2];

            N_tot = S + N + G1;
            hill = pow(k,n)/(pow(k,n) + pow(N_tot,n));

            // check for establishment and fixation for the G populations
            if (rel_to == 1){
                G1_frac = (float) G1/N_tot;
            }
            else {
                G1_frac = (float) G1/N_0;
            }

            // Check G1 and G2 first appearance
            if ((G1_app_found == false) && (G1 > 0)){
                G1_app = t;
                G1_app_found = true;
                if (stop_at == 1) break;
            }

            // Check G1 t_est and t_fix
            if ((G1_est_found == false) && (G1_frac > 0.05)){
                G1_est = t;
                G1_est_found = true;
                if (stop_at == 2) break;
            }
            else if ((G1_fix_found == false) && (G1_frac > 0.95)){
                G1_fix = t;
                G1_fix_found = true;
                break;
            }
        }
    }
    fprintf(outfile_est_fix, "%d    %0.20f    %0.20f    %0.20f\n", i, G1_app, G1_est, G1_fix)↙
        ↳ ;
    printf("Results: G1_app = %0.20f, G1_est = %0.20f, G1_fix = %0.20f\n\n", G1_app, G1_est, ↙
        ↳ G1_fix);
}

fclose(outfile_est_fix);
```

```
    if (outfile != NULL) fclose(outfile);
    return 0;
}
```

A.2.3: Python code used to plot individual trajectories produced by my C implementation of the SSA.

```
import matplotlib.pyplot as plt
import argparse
plt.rcParams.update({'font.size': 10})

parser = argparse.ArgumentParser(description='Get data file')
parser.add_argument('-i', "--infile", default=["SSA_output.txt"], type=str, nargs = '+', dest = '
    ↪ INFILE', help='Name of SSA data file (.txt)')
parser.add_argument('-s', "--scenario", default=[1], type=int, nargs = '+', dest = "SCENARIO", 
    ↪ help='Integer for scenario (1 or 2)')
parser.add_argument('-o', "--plot1", default=["SSA_trajectory.png"], type=str, nargs = '+', dest 
    ↪ = 'PLOT1', help='Name of trajectory plot file (.png)')
parser.add_argument('-f', "--plot2", default=["SSA_trajectory_subplots.png"], type=str, nargs = '
    ↪ +', dest = 'PLOT2', help='Name of subplots file (.png)')
args = parser.parse_args()

t = []
S = []
N = []
G1 = []
G2 = []
N_tot = []
dNtot_dt = []

G1_d_N = []
G2_d_N = []

counter = 0
# pull SSA data
with open(args.INFILE[0], 'r') as f:
    for line in f:
        if not line.startswith("#"):
            if counter < 3000000 or counter % 1000000 == 0:
                data = line.split()

                t.append(float(data[1]))
                S.append(float(data[2]))
                N.append(float(data[3]))
                G1.append(float(data[4]))
                if args.SCENARIO[0] == 2:
                    G2.append(float(data[5]))
                    N_tot.append(float(data[6]))
                    dNtot_dt.append(float(data[7]))
                else:
                    N_tot.append(float(data[5]))
                    dNtot_dt.append(float(data[6]))

                try:
                    G1_d_N.append(float(data[4])/float(data[3]))
                    if args.SCENARIO[0] == 2:
                        G2_d_N.append(float(data[5])/float(data[3]))
                except:
                    G1_d_N.append(0.0)
                    if args.SCENARIO[0] == 2:
                        G2_d_N.append(0.0)

            counter += 1

# make plots
fig1 = plt.figure(figsize=(5,4))
plt.xscale("log")
plt.yscale("log")

plt.xlabel("t (hr)")
plt.ylabel("cells")
#plt.title("SSA Results")

plt.plot(t,S, ls = "-", label = "S")
plt.plot(t,N, ls = "-.", label = "N")
plt.plot(t,G1, ls = ":", label = "G")

if args.SCENARIO[0] == 2:
    plt.plot(t,G2, ls = "--", label = "G2")

plt.legend(loc="best")

plt.gcf().text(0.03, 0.92, "A)", weight = 'bold')

plt.tight_layout()
plt.savefig(fname = args.PLOT1[0], dpi = 100)

# fig2 = plt.figure(2, figsize = (16,12))
# plt.subplot(4,3,1)
# plt.xscale("log")
# plt.yscale("log")
# plt.ylabel("S")
# plt.plot(t,S)

# plt.subplot(4,3,2)
# plt.xscale("log")
# plt.yscale("log")
# plt.title("SSA Individual Results and Ratios")
# plt.ylabel("N")
# plt.plot(t,N)

# plt.subplot(4,3,3)
# plt.xscale("log")
# plt.yscale("log")
# plt.ylabel("G1")
# plt.plot(t,G1)

# if args.SCENARIO[0] == 2:
#     plt.subplot(4,3,4)
#     plt.xscale("log")
#     plt.yscale("log")
#     plt.ylabel("G2")
#     plt.plot(t,G2)

# plt.subplot(4,3,5)
# plt.xscale("log")
# plt.yscale("log")
# plt.ylabel("N_tot")
# plt.plot(t,N_tot)

# plt.subplot(4,3,6)
# plt.xscale("symlog")
# plt.yscale("symlog")
# plt.ylabel("dN_tot/dt")
# plt.plot(t,dNtot_dt)

# plt.subplot(4,3,7)
# plt.xscale("log")
# plt.yscale("log")
# plt.ylabel("G1/N")
# plt.plot(t,G1_d_N)

# if args.SCENARIO[0] == 2:
#     plt.subplot(4,3,8)
#     plt.xscale("log")
#     plt.yscale("log")
#     plt.xlabel("time [hr]")
#     plt.ylabel("G2/N")
#     plt.plot(t,G2_d_N)

# plt.savefig(args.PLOT2[0])
```

A.2.4: Python code used to calculate first appearance and genetic fixation times from the results produced by my C implementation of the SSA.

```
import matplotlib.pyplot as plt
import argparse
import numpy as np

parser = argparse.ArgumentParser(description='Get data file')
parser.add_argument('-i', "--infile", default=["SSA_output.txt"], type=str, nargs = '+', dest = '↙
    ↪ INFILE', help='Name of SSA data file (.txt)')
args = parser.parse_args()


fix1 = "N/A"
est1 = "N/A"
est1_found = False
fix2 = "N/A"
est2 = "N/A"
est2_found = False

done1 = False
done2 = False

counter = 0
# pull SSA data
with open(args.INFILE[0], 'r') as f:
    for line in f:
        if not line.startswith("#"):
            data = line.split()
            t = float(data[1])
            G1 = float(data[4])
            G2 = float(data[5])
            N_tot = float(data[6])

            pop_frac1 = G1/N_tot
            pop_frac2 = G2/N_tot

            if est1_found == False and pop_frac1 > 0.05:
                est1 = t
                est1_found = True

            if done1 == False and pop_frac1 > 0.95:
                fix1 = t
                done1 = True

            if est2_found == False and pop_frac2 > 0.05:
                est2 = t
                est2_found = True

            if pop_frac2 > 0.95:
                fix2 = t
                done2 = True

            if done1 and done2:
                break

            counter += 1


print("\nG1 Establishment time: {}\nG1 Fixation time: {}".format(est1, fix1))
print("\nG2 Establishment time: {}\nG2 Fixation time: {}".format(est2, fix2))
```

A.2.5: Python code used to create the first appearance and fixation probability distributions shown in Chapter 2.

```
import matplotlib.pyplot as plt
import numpy as np
import argparse
plt.rcParams.update({'font.size': 10})


parser = argparse.ArgumentParser(description='Get data file')
parser.add_argument('-i', "--infile", default=["SSA_output_times.txt"], type=str, nargs = '+', ↩
    ↪ dest = 'INFILE', help='Name of SSA data file (.txt)')
parser.add_argument('-s', "--scenario", default=[1], type=int, nargs = '+', dest = "SCENARIO", ↩
    ↪ help='Integer for scenario (1 or 2)')
parser.add_argument('-o', "--plot", default=["SSA_distributions.png"], type=str, nargs = '+', ↩
    ↪ dest = 'PLOT', help='Name of distribution plot outfile (.png)')
args = parser.parse_args()


G1_app = []
G1_est = []
G1_fix = []
G2_app = []
G2_est = []
G2_fix = []

# pull SSA data
num_traj = 0
with open(args.INFILE[0], 'r') as f:
    for line in f:
        if not line.startswith("#"):
            data = line.split()

            if float(data[1]) >= 0:
                G1_app.append(float(data[1]))
            if float(data[2]) >= 0:
                G1_est.append(float(data[2]))
            if float(data[3]) >= 0:
                G1_fix.append(float(data[3]))

            if args.SCENARIO[0] == 2:
                if float(data[4]) >= 0:
                    G2_app.append(float(data[4]))
                if float(data[5]) >= 0:
                    G2_est.append(float(data[5]))
                if float(data[6]) >= 0:
                    G2_fix.append(float(data[6]))

            num_traj += 1

# calculate statistics and make histograms
fig = plt.figure(figsize=(4.5, 3.5))

# appearance
print("G1 t_app found for {} out of {} trajectories".format(len(G1_app), num_traj))
if len(G1_app) > 0:
    print("G1 t_app Mean: ", np.mean(G1_app))
    print("G1 t_app STD: ", np.std(G1_app))
    print("G1 t_app Noise: ", np.std(G1_app)/np.mean(G1_app))

    weights = np.ones_like(G1_app) / 100000
    #plt.title(r"$G$ $t_{appear}$")
    plt.ylabel(r"$P_{\tau}$")
    plt.xlabel("t (hr)")
    plt.hist(G1_app, bins = 20, weights = weights, color = "grey", label = "Mean = {:.3f} hr\nCV ↙
        ↳ = {:.3f}".format(np.mean(G1_app), np.std(G1_app)/np.mean(G1_app)))
    #plt.legend(loc="best")

    plt.gcf().text(0.001, 0.92, "A)", weight = 'bold')
    plt.gcf().text(0.67, 0.85, "Mean = {:.3f} hr\nCV = {:.3f}".format(np.mean(G1_app), np.std(↙
        ↳ G1_app)/np.mean(G1_app)), bbox = dict(boxstyle='round', facecolor='none', edgecolor='↙
        ↳ black'))

plt.tight_layout()
#plt.savefig(fname = 'SSA_lowNfitness_appear.png', dpi = 300)
print("\n")

fig = plt.figure(figsize=(3.5, 2.5))
print("G1 t_est found for {} out of {} trajectories".format(len(G1_est), num_traj))
if len(G1_est) > 0:
    print("G1 t_est Mean: ", np.mean(G1_est))
    print("G1 t_est STD: ", np.std(G1_est))
    print("G1 t_est Noise: ", np.std(G1_est)/np.mean(G1_est))

    weights = np.ones_like(G1_est) / 100000
    #plt.title(r"$G_1$ $t_{est}$")
    plt.ylabel(r"$P_{\tau_{est}}$")
    plt.xlabel("t (hr)")
    plt.hist(G1_est, bins = 20, weights = weights, color = "grey", label = "Mean = {:.3f} hr\nCV ↙
        ↳ = {:.3f}".format(np.mean(G1_est), np.std(G1_est)/np.mean(G1_est)))


    plt.gcf().text(0.001, 0.92, "H)", weight = 'bold')
    plt.gcf().text(0.55, 0.77, "Mean = {:.3f} hr\nCV = {:.3f}".format(np.mean(G1_est), np.std(↙
        ↳ G1_est)/np.mean(G1_est)), bbox = dict(boxstyle='round', facecolor='none', edgecolor='↙
        ↳ black'))

plt.tight_layout()
#plt.savefig(fname = 'SSA_high_est.png', dpi = 300)
print("\n")

fig = plt.figure(figsize=(4.5, 3.5))
print("G1 t_fix found for {} out of {} trajectories".format(len(G1_fix), num_traj))
if len(G1_fix) > 0:
    print("G1 t_fix Mean: ", np.mean(G1_fix))
    print("G1 t_fix STD: ", np.std(G1_fix))
    print("G1 t_fix Noise: ", np.std(G1_fix)/np.mean(G1_fix))

    weights = np.ones_like(G1_fix) / 100000
    #plt.title(r"$G_1$ $t_{fix}$")
    plt.ylabel(r"$P_{\tau_{fix}}$")
    plt.xlabel("t (hr)")
    plt.hist(G1_fix, bins = 20, weights = weights, color = "grey", label = "Mean = {:.3f} hr\nCV ↙
        ↳ = {:.3f}".format(np.mean(G1_fix), np.std(G1_fix)/np.mean(G1_fix)))


    plt.gcf().text(0.001, 0.92, "B)", weight = 'bold')
    plt.gcf().text(0.67, 0.85, "Mean = {:.3f} hr\nCV = {:.3f}".format(np.mean(G1_fix), np.std(↙
        ↳ G1_fix)/np.mean(G1_fix)), bbox = dict(boxstyle='round', facecolor='none', edgecolor='↙
        ↳ black'))

plt.tight_layout()
#plt.savefig(fname = 'SSA_lowNfitness_fix.png', dpi = 300)
print("\n")

# if args.SCENARIO[0] == 2:
#     print("G2 t_app found for {} out of {} trajectories".format(len(G2_app), num_traj))
#     if len(G2_app) > 0:
#         print("G2 t_app Mean: ", np.mean(G2_app))
#         print("G2 t_app STD: ", np.std(G2_app))
#         print("G2 t_app Noise: ", np.std(G2_app)/np.mean(G2_app))

#         plt.subplot(3,3,i)
#         plt.title(r"$G_2$ $t_{app}$")
#         plt.ylabel("counts")
#         plt.xlabel("time (hr)")
#         plt.hist(G2_app, bins = 10, label = "Mean = {:.3f}\nSTD = {:.3f}\nNoise = {:.3f}".↙
    ↳ format(np.mean(G2_app), np.std(G2_app), np.std(G2_app)/np.mean(G2_app)))
#         plt.legend(loc="best")

#         i += 1
#     print("\n")

#     print("G2 t_est found for {} out of {} trajectories".format(len(G2_est), num_traj))
#     if len(G2_est) > 0:
#         print("G2 t_est Mean: ", np.mean(G2_est))
#         print("G2 t_est STD: ", np.std(G2_est))
#         print("G2 t_est Noise: ", np.std(G2_est)/np.mean(G2_est))

#         plt.subplot(3,3,i)
#         plt.title(r"$G_2$ $t_{est}$")
#         plt.ylabel("counts")
#         plt.xlabel("time (hr)")
#         plt.hist(G2_est, bins = 10, label = "Mean = {:.3f}\nSTD = {:.3f}\nNoise = {:.3f}".↙
    ↳ format(np.mean(G2_est), np.std(G2_est), np.std(G2_est)/np.mean(G2_est)))
#           plt.legend(loc="best")

#           i += 1
#       print("\n")

#       print("G2 t_fix found for {} out of {} trajectories".format(len(G2_fix), num_traj))
#       if len(G2_fix) > 0:
#           print("G2 t_fix Mean: ", np.mean(G2_fix))
#           print("G2 t_fix STD: ", np.std(G2_fix))
#           print("G2 t_fix Noise: ", np.std(G2_fix)/np.mean(G2_fix))

#           plt.subplot(3,3,i)
#           plt.title(r"$G_2$ $t_{fix}$")
#           plt.ylabel("counts")
#           plt.xlabel("time (hr)")
#           plt.hist(G2_fix, bins = 10, label = "Mean = {:.3f}\nSTD = {:.3f}\nNoise = {:.3f}".↙
    ↳ format(np.mean(G2_fix), np.std(G2_fix), np.std(G2_fix)/np.mean(G2_fix)))
#           plt.legend(loc="best")
#       print("\n")

#plt.tight_layout()
#plt.show()
```

A.2.6: Python code used to create the plots in Chapter 2 showing the probability of a genetic mutation occurring before population extinction.

```
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams.update({'font.size': 10})


results_dir = "/home/jguthrie/cLab/RESEARCH/S-N-G/git_files/S-N-G/final_results/dS_dN/"
#dirs = ['df1', 'df1.5', 'df2']
dirs = ['df2', 'df4']
file_names = ['/kN_0.1733.txt', '/kN_0.2022.txt', '/kN_0.2311.txt', '/kN_0.2600.txt', '/kN_0↙
    ↳ .2889.txt', '/kN_0.3178.txt', '/kN_0.3466.txt']



df1_vals = [0.99989, 1, 1, 1, 1, 1, 1]
# kN values used
kN_vals = [0.1733, 0.2022, 0.2311, 0.2600, 0.2889, 0.3178, 0.3466]


means = {"df1": [0.99989, 1, 1, 1, 1, 1, 1],
         "df2": [],
         "df4": []}
stds = {"df1": [0, 0, 0, 0, 0, 0, 0],
         "df2": [],
         "df4": []}

# pull SSA data
for dir in dirs:
    #used to store t_appearance data from the files
    tapp_data = [[],[],[],[],[],[],[]]
    file_num = 0

    directory = results_dir + dir
    for fn in file_names:
        with open(directory+fn, 'r') as f:
            for line in f:
                if not line.startswith("#"):
                    tapp_data[file_num].append(float(line.split()[1]))
        file_num += 1


    for data in tapp_data:
        data_chunks = [data[:10000], data[10000:20000], data[20000:30000], data[30000:40000], ↙
            ↳ data[40000:50000],
                    data[50000:60000], data[60000:70000], data[70000:80000], data[80000:90000], ↙
                        ↳ data[90000:100000]]
        tapp = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
        for i in range(len(data_chunks)):
            for sim in data_chunks[i]:
                if sim >= 0:
                    tapp[i] += 1
            tapp[i] = tapp[i]/10000

        means[dir].append(np.mean(tapp))
        stds[dir].append(np.std(tapp))



plt.figure(figsize=(4.5,3.5))
plt.errorbar(kN_vals, means['df1'], yerr = stds['df1'], color = 'red', linestyle = '-.', ↙
    ↳ linewidth = 1, elinewidth = 2, capsize = 6, label = '$f_{\delta} = 1$')
plt.errorbar(kN_vals, means['df2'], yerr = stds['df2'], color = 'blue', linestyle = '-', ↙
    ↳ linewidth = 1, elinewidth = 2, capsize = 6, label = '$f_{\delta} = 2$')
```

```
plt.errorbar(kN_vals, means['df4'], yerr = stds['df4'], color = 'green', linestyle = '--', ↙
    ↳ linewidth = 1, elinewidth = 2, capsize = 6, label = '$f_{\delta} = 4$')
plt.xlabel("$k_N$ (/hr)")
plt.ylabel("$P_G$")
#plt.grid()

plt.legend(loc = 'best')
plt.ylim((0,1.1))
plt.gcf().text(0.03, 0.92, "A)", weight = 'bold')
#plt.title("Survival Probability, Cidal d_N = 2*0.1733, 10000 Runs Each")
plt.tight_layout()
plt.savefig(fname = 'SSA_survival.png', dpi = 300)
```

A.2.7: Python code used to compare the fixation times found from the ODE and SSA methods.

```
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams.update({'font.size': 10})

dN = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]
ODE = [144.63, 71.9, 47.92, 35.99, 28.86, 24.12, 20.76, 18.26, 16.33, 14.84]
SSA = [138.9, 71.91, 48.62, 36.74, 29.58, 24.78, 21.32, 18.74, 16.74, 15.16]
ssa_std = [8.49, 5.44, 3.93, 3.04, 2.50, 2.11, 1.8, 1.58, 1.42, 1.27]

plt.figure(figsize=(4.5,3.5))
plt.plot(dN, ODE, marker = "o", linestyle = '', label = "ODE")
plt.errorbar(dN, SSA, yerr=ssa_std,ecolor = 'black', fmt = 'x', label = "SSA", elinewidth = 1, ↙
    ↳ capsize = 3, markersize=5)
plt.xlabel("$\delta_N$ (/hr)")
plt.ylabel(r"$\tau_{fix}$ (hr)")
#plt.grid()

plt.legend(loc = 'best')
#plt.gcf().text(0.03, 0.92, "A)", weight = 'bold')
#plt.title("Survival Probability, Cidal d_N = 2*0.1733, 10000 Runs Each")
plt.tight_layout()
plt.savefig(fname = 'ODE_SSA_comp_Gi_1.png', dpi = 300)
```