SRI VENKATESWARA INTERNSHIP PROGRAM FOR RESEARCH IN ACADEMICS (SRI – VIPRA)

SRI-VIPRA

Project Report of 2025: SVP-2530

Statistical and Mathematical Analysis of Few Biology Systems.

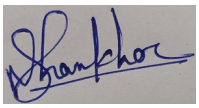
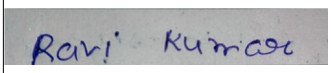
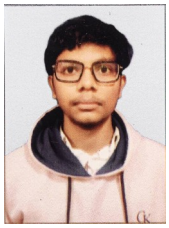
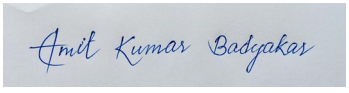
IQAC
Sri Venkateswara College
University of Delhi
Benito Juarez Road, Dhaula Kuan, New Delhi
New Delhi -110021

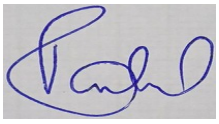
SRIVIPRA PROJECT 2025

Title : Statistical and Mathematical Analysis of Few Biology Systems.

Name of Mentor : **Ram Lal Awasthi** **Name of Department :** **Physics** **Designation :** **Assistant Professor**	**Photo** 

List of students under the SRIVIPRA Project

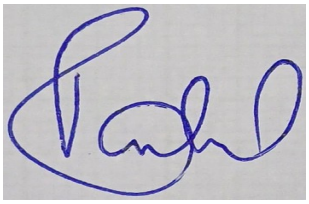
S.No	Photo	Name of the student	Roll number	Course	Signature
1		Deepak Singh	1824006	B.Sc. (H) Physics	
2		Niku	1824011	B.Sc. (H) Physics	
3		Ravi Kumar	1824033	B.Sc. (H) Physics	
4		Amit Kumar Badyakar	1824052	B.Sc. (H) Physics	



Signature of Mentor

Certificate of Originality

This is to certify that the aforementioned students from Sri Venkateswara College have participated in the summer project **SVP-2530** titled "**Statistical and Mathematical Analysis of Few Biology Systems.**".The participants have carried out the research project work under my guidance and supervision from 1st July, 2024 to 30th September 2025. The work carried out is original and carried out in an offline mode.



Signature of Mentor

Acknowledgements

We would like to thank The Principal SVC (Prof. V. Ravi), The SRI-VIPRA 2025-26 Team, IQAC – SVC and TIC, Department of Physics (Dr. Narender Kumar) for all the support and the opportunity.

We would also like to express our gratitude towards Dr. Ram Lal Awasthi for the supervision of the project.

Deepak Singh
Niku
Ravi Kumar
Amit Kumar Badyakar

SVP-2530 : Statistical and Mathematical Analysis of Few Biology Systems

A SRI-VIPRA 2025 project report

by

Deepak Singh (1824006),

Niku (1824011),

Ravi Kumar (1824033)

and

Amit Kumar Badyakar (1824052)

in supervision of

Dr. Ram Lal Awasthi

Submitted to Sri Venkateswara College.

Department of Physics

Sri Venkateswara College

University of Delhi
Dhaula Kuan, New Delhi, Delhi 110021

Certificate

This is to certify that the project report titled "**SVP-2530 : Statistical and Mathematical Analysis of Few Biology Systems**" submitted to the **Sri Venkateswara College** as the part of Sri-Vipra 2025, is the record of bona fide research work done by Deepak Singh (1824006), Niku (1824011), Ravi Kumar (1824033) and Amit Kumar Badyakar (1824052) in my supervision.

Dr. Ram Lal Awasthi

(Supervisor)

Assistant Professor

Sri Venkateswara College

University of Delhi

New Delhi-110021

ABSTRACT:

Mathematical modeling of Biological systems is no longer a new and exotic area of interest. It has become an essential part of any kind of scientific endavour, including biological science. Here, we have tried to ventured in to four important sub-area of modeling in biological systems namely : 1. Ecology, 2. Logistic map, 3. Gene Networks and 4. Pharmacokinetics. The purpose of study is two folded. 1. To understand the biological systems from physicist point of view and 2. To find the prospect of implementing our knowledge in Mathematics, Physics and Statistic in to those systems by introducing new models to explain the behaviour of the systems and most importantly to make predictions about their dynamics/evolution. We have stressed on finding exotic behaviour of the system's response under the uncertainties present in the model parameters. We found that even a small change, at times, may cause a dramatic change in the trjectory of evolution of system. Such a systems would lie in a category of sensitive systems.

Key Words: Predator-Pray, Population, Lotka-Volterra, Holling Function, Gompertz Growth, Crop, pest, Gene, Network, DNA, mRNA, Transcription, Translation, Regulation, Negative Feedback, Gamma Function, Phase Portrait, Pharmacology, Compartment Model, Dose, Logistic growth, Bifurcation, Chaos, Cobweb, Lyapunov, Feigenbaum, Mandelbrot Set, Fractal.

Contents

List of Figures

List of Tables

CHAPTER 1

Predator-Prey Model in Agriculture

1.1 Introduction

A simplest predator-pray model is described by two interacting populations. Of which, pray typically grows in absence of predator assuming the inexhaustive abundance of resources on which pray relies, while the growth of population of predator depends on the population of pray. A mathematical model, like the given below, roughly mimics the behavior of this problem from ecology [1, 2]:

$$\frac{dN}{dt} = \alpha N - \beta NP \tag{1.1}$$

$$\frac{dP}{dt} = -\gamma P + \delta NP \tag{1.2}$$

where $\alpha, \beta, \gamma, \delta$ are the model parameters. Also, N and P are the instantaneous populations of pray and predator respectively. The parameters of the system are predicted by fitting the model with the observed data. A more formal definition of the parameters goes like following:

- α is the natural growth rate of the prey in the absence of predators,
- β is the predation rate coefficient,
- γ is the natural death rate of predators in the absence of prey.
- δ is the rate at which consumed prey contributes to predator growth.

This model explains that when predators are less in number, the prey population increases rapidly. With rapidly growing population, prey become more abundant, the predator population also starts to rise due to abundence of food. However, when predators population increases, the prey population starts reducing due to consumption, which causes the predator population to fall again. This leads to a periodic behavior where both populations oscillate around an equilibrium point. This toy model provides a clear picture of interacting population dynamics, though it ignores several real-world factors such as limited food, handling time, and environmental effects etc.

In this study we discuss a more realistic model on the prospects of human intervention in the crop production using different methods. As the population of the world grows, agricultural land shrinks. This raises a serious concern about the survival of growing population with the current crop production.

A mathematical model for finding the ways to improve crop production is studied in the line of Sudhakar et al. [3]. To properly analyze the process of crop production and predict its future behavior under different conditions, it is useful to build a mathematical model. Such a model helps us understand how multiple factors - like crop growth rate, pest population, and predator activity - interact and influence overall production. By simulating these conditions, we can test various agricultural strategies and find how to balance crop yield with pest control more effectively.

A simple way to improve crop production is to kill pests as they harm the crop. Several methods are applied to kill the pests, like chemical pesticides or organic pesticides, and sometime natural predators are introduced in the system to control the dynamics. A nice method should not upset the eco-system to the stage of non-reversibility. Pesticides have a disadvantage as a part of it gets absorbed to the crop, which further comes into food chain, and may cause serious physiological damage to the consumer.

To see trend of using pesticide on crop production people try to model the system with different parameters which impact the production of crop. In this study we have worked on the model of [3] for agricultural pre-predator model. While, we introduce different growth equation and Holling functions to get a model which mimics the real problems more accurately.

1.2 Holling Function

Holling functions in echology is a mathematical representation of food consumption by an species depending on the availability of food. The food ofcourse means the pery. There are three kinds of Holling functions.

1. **Type I:**

$$f(N) = \alpha N$$

Says, the consumption of predator linearly increases with the population density of prey. Seems unrealstic but used in Lotka-Volterra predator-prey model. As linear means each and every prey is consumed by predator. It also means that predator has no limits on consumption.

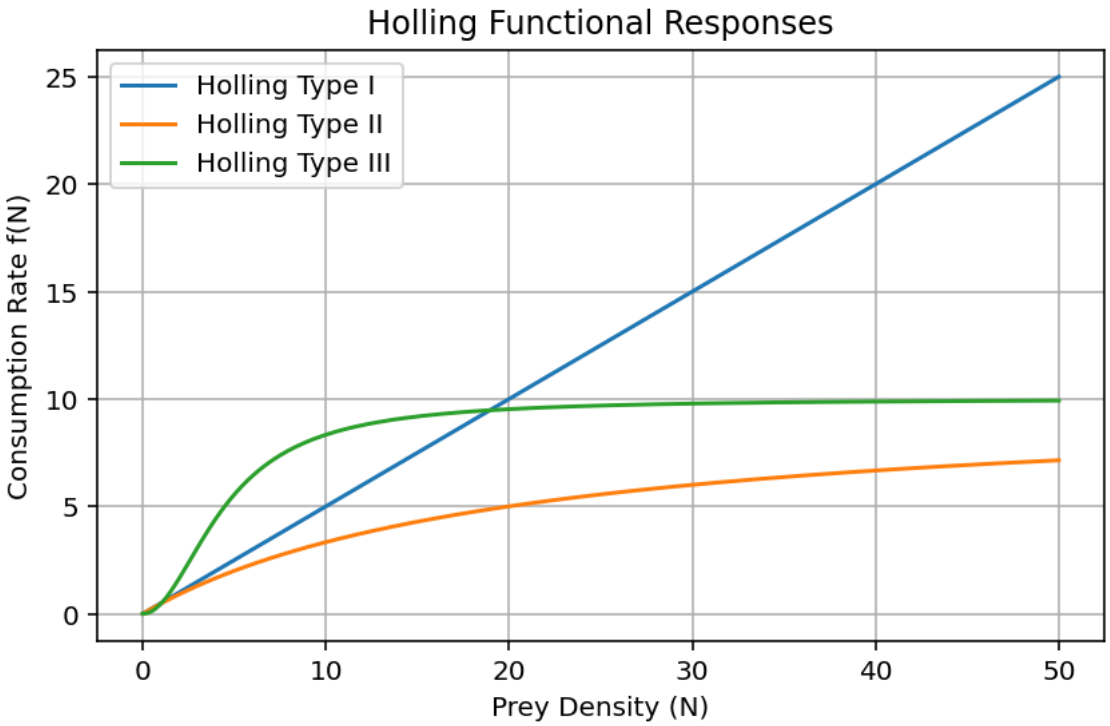


Figure 1.1: Holling fun vs prey density. The attack rate α is constant.

2. **Type II:**

$$f(N) = \frac{\alpha N}{1 + \alpha h N}$$

Is more realistic and follows the realistic assumption that consumer has limited capacity of consumtion. Depends on attack rate α and the handling time (h) which means for

small prey density N behavior of function is linear while for larger N it get saturated due to h as predator need more time for process (catch and consume)it.

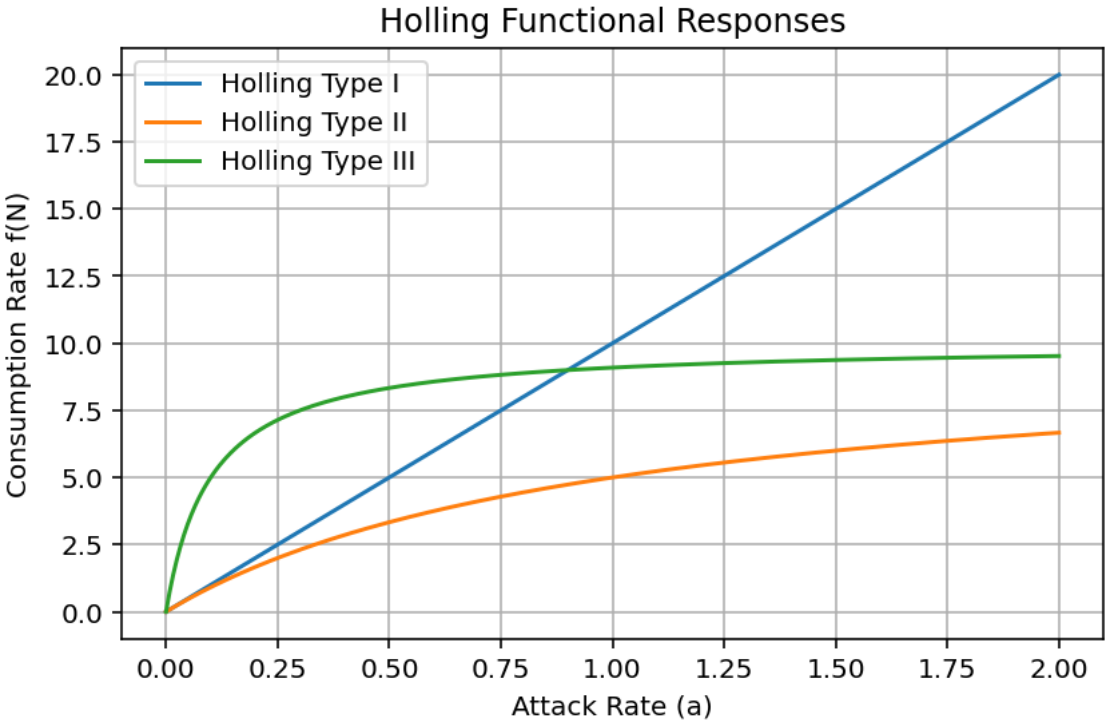


Figure 1.2: Holling fun vs attack rate keeping the prey density (N) constant.

3. **Type III:**

$$f(N) = \frac{\alpha N^k}{1 + \alpha h N^k}$$

This response function is similar to Type-II in a way that at higher values of population density of prey the saturation in cosumption happens. But, at low density it reflects the fact of challenges in finding the food. The learning of catching prey improves with increase in the discovery rate of pray as more encounters with prey occur when population of prey increases. In our model we consider the quadratic dependence on the density making $k = 2$.

Fig.1.1 shows three types of Holling functions vs prey density when attack rate (α) is constant. In Type-I, the curve increases straight without any limit. In Type-II, the rate increases fast first and then becomes constant after some level of prey because of handling time. In Type-III, at low prey level, the increase is slow, then it becomes faster and later again slow. So, Type-III is more realistic as it shows how predators change their feeding behavior.

Fig.1.2 show how change in attack rate, α, affects Type-II and Type-III Holling functions. When α increases, the curve goes higher and reaches its limit faster. In Type-II, saturation happens early, while in Type-III it takes longer time to become stable. It means large α helps predators to attack more, but after a limit, handling time controls it.

1.3 Growth Equations

Three major categories of population (or growth) models as listed below are:

Exponential grow – assumes unlimited resources, so growth continues without bound.

$$\frac{dN}{dt} = rN \tag{1.3}$$

where N is population at any time t and r is intrinsic growth rate. Such a growth is impractical in real-world scenarios because resources and space are always limited.

Logistic growth – introduces a carrying capacity, meaning the population slows down as it approaches the environment's maximum capability to support the population. It's symmetric and assumes that the population will always reach the carrying capacity if given enough time.

$$\frac{dN}{dt} = rN\left(1 - \frac{N}{K}\right) \tag{1.4}$$

where N is population at any time t and r is intrinsic growth rate and K is carrying capacity.

Gompertz growth – an asymmetric growth model. Growth starts slowly due to more realistic factors like adaptation or stress, acceleration in the middle phase after adaptation phase is over, and then saturation, as expected.

$$\frac{dN}{dt} = rN\left(\frac{K}{N}\right) \tag{1.5}$$

where N is population at any time t and r is intrinsic growth rate.

Fig.1.3 shows Exponential, Logistic, and Gompertz growth curves. Exponential growth keeps increasing without any limit, which is not possible for crops. Logistic growth slows

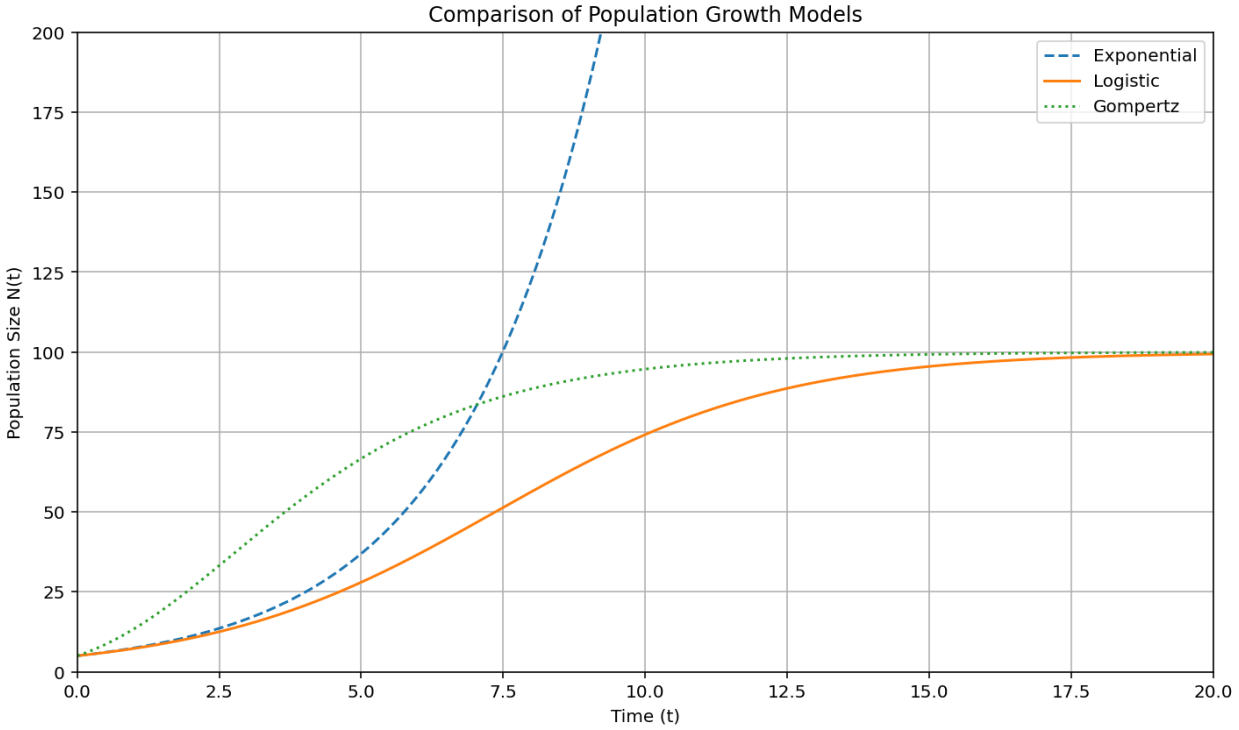


Figure 1.3: Comparision of the population Growth in different growth models.

down near the carrying capacity and becomes stable. Gompertz growth starts slowly, increases in middle, and then saturates early. It matches crop growth better because in real life crops face harsh environmental conditions stress, pests, and limited nutrients. Later it learns how to harness the available resources.

We choose Gompertz growth as it reflects real-world conditions more realistically. For example, in crops growth initially there may be slow growth due to adaptation to soil, climate, or resource stress, and it may never reach full carrying capacity because of disease, pests, or environmental limitations. Unlike logistic growth, Gompertz captures this early slowdown and asymmetry, making it more suitable for practical modeling.

1.4 Model Formation

In this section, we will discribe the mathematical model of our system of interest with different parameter and functional dependence. We have four varibale in the system. Parameters with their values used are shown in Table 1.1. The variables of the system are as following:

1. **Crop:** corp itself ; say P_1

2. **Susceptible Pest:** Safe pest as they are not infected till now; say P_2

3. **Infected Pest:** pest which are got infection from pesticide; say P_3

4. **Predator:** Natural enemy of pest : say P_4

We will study all three cases by varying the types of Holling functions as discussed earlier to make a good comparision about the behavour of system under different assumptions.

We are looking at a case of agricultural production with crop density P_1 which is attacked by a pest with density P_2 . To control the pest, we use a pesticide, which makes some pests infected, forming a infected pest population with density P_3. At the same time, there is a natural predator in the field that eats both the crop and the pests, with a Population density P_4 . We want to see how all these populations interact and affect each other over time.

1.4.1 Case-1: Holling Type-I

The model is identical to [3], which uses the logitic growth for growth and **Holling Type-I** functions for interaction.

$$\frac{dP_1}{dt} = r_1 P_1 \left(1 - \frac{P_1}{K_1}\right) - \lambda P_1 P_2 - \rho P_1 P_3 \tag{1.6}$$

$$\frac{dP_2}{dt} = \lambda P_1 P_2 - \sigma P_2 + r_2 P_2 \left(1 + \frac{P_2 + \eta P_3}{K_2}\right) - \alpha P_2 P_3 - \beta P_2 P_4 \tag{1.7}$$

$$\frac{dP_3}{dt} = \alpha P_2 P_3 + \rho P_1 P_3 - \gamma P_3 P_4 - \omega P_3 \tag{1.8}$$

$$\frac{dP_4}{dt} = l\beta P_2 P_4 + (n_1 - n_2)(\gamma) P_3 P_4 + d\left(1 - \frac{P_2 + \eta P_3}{K_2}\right) P_4 - \mu P_4 - \delta (P_4)^2 \tag{1.9}$$

Fig.1.4 shown above show the crop production trend and its interaction with pest and predator when we are using Holling Type-I and logistic growth. The prameters values are taken from Table1.1.

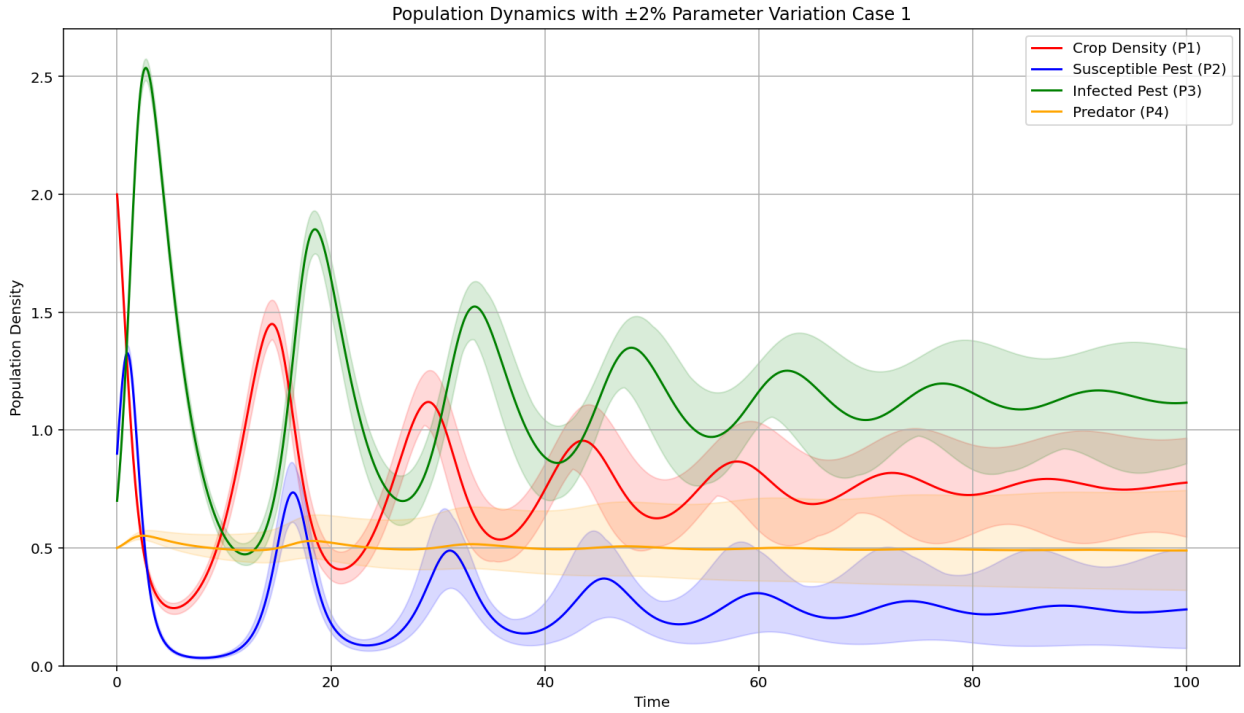


Figure 1.4: Evolution of variable density with time. The error bands around the curves represent the fluctuation in the values of variable due to 2% errors in the model parameters.

1.4.2 Case-2: Holling Type-II

Next, we apply **logistic growth** equation and **Holling Type-II** function we introduce handling time in our equation.

$$\frac{dP_1}{dt} = r_1 P_1\left(1-\frac{P_1}{K_1}\right) - \frac{\lambda P_1 P_2}{1+\lambda h_1 P_1} - \frac{\rho P_1 P_3}{1+\rho h_2 P_1} \tag{1.10}$$

$$\frac{dP_2}{dt} = \frac{\lambda P_1 P_2}{1+\lambda h_1 P_1} + r_2 P_2\left(1-\frac{P_2+\eta P_3}{K_2}\right) - \frac{\alpha P_3 P_2}{1+\alpha h_3 P_2} - \frac{\beta P_4 P_2}{1+\beta h_4 P_2} - \sigma P_2 \tag{1.11}$$

$$\frac{dP_3}{dt} = \frac{\alpha P_3 P_2}{1+\alpha h_3 P_2} + \frac{\rho P_1 P_3}{1+\rho h_2 P_1} - \frac{\gamma P_4 P_3}{1+\gamma h_5 P_3} - \omega P_3 \tag{1.12}$$

$$\frac{dP_4}{dt} = l\left(\frac{\beta P_4 P_2}{1+\beta h_4 P_2}\right) + (n_1-n_2)\frac{\gamma P_4 P_3}{1+\gamma h_5 P_3} + d\left(1-\frac{P_2+nP_3}{K_2}\right)P_4 - \mu P_4 - n\delta P_4^2 \tag{1.13}$$

Fig.1.5 shown above exhibits the crop production trend and its interaction with pest and predator when we apply Holling Type-2 and logitic growth. The prameters values are taken from Table 1.1.

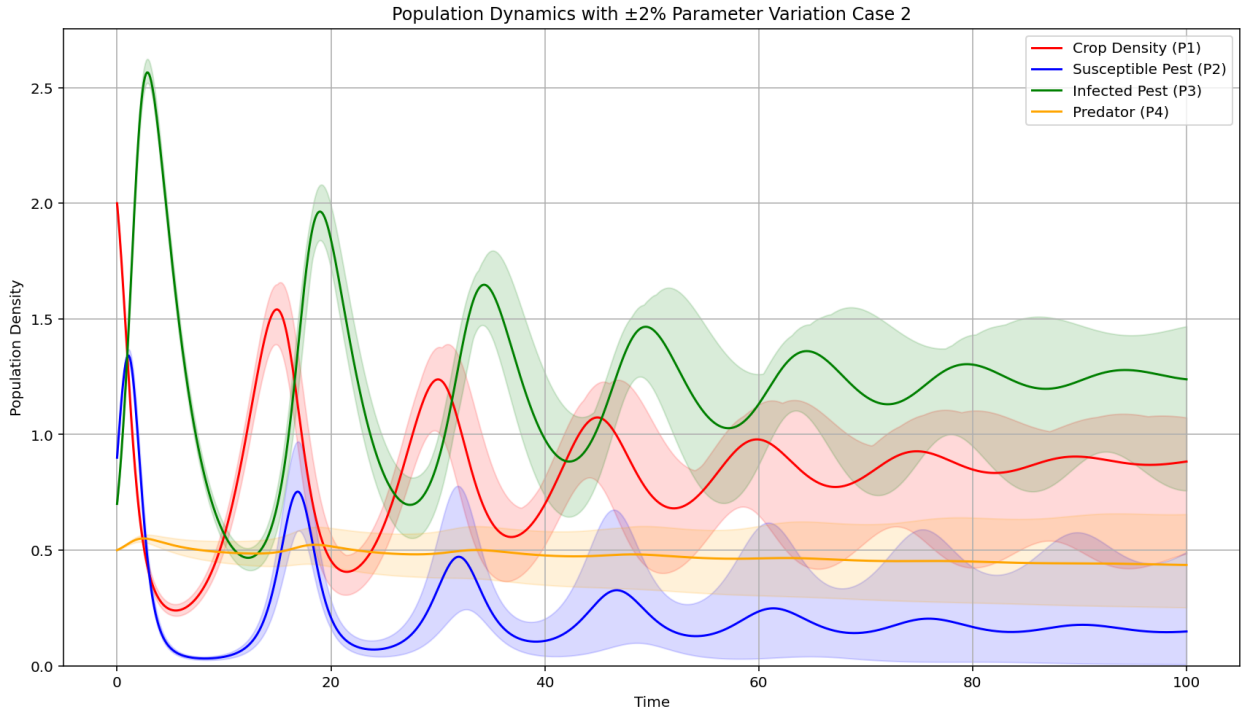


Figure 1.5: Evolution of variable density with time. The error bands around the curves represent the fluctuation in the values of variable due to 2% errors in the model parameters.

1.4.3 Case-3 : Holling Type-III

In this case we used **Gompertz function** for growth which talk about exponetial growth and **Holling Type-II** for prey predator and crop interaction.

$$\frac{dP_1}{dt} = C_1 P_1 \log\left(\frac{K_1}{P_1}\right) - \frac{\lambda P_1 P_2}{1+\lambda h_1 P_1} - \frac{\rho P_1 P_3}{1+\rho h_2 P_1} \tag{1.14}$$

$$\frac{dP_2}{dt} = \frac{\lambda P_1 P_2}{1+\lambda h_1 P_1} + C_2 P_2 \log\left(\frac{K_2}{P_2+nP_3}\right) - \frac{\alpha P_3 P_2}{1+\alpha h_3 P_2} - \frac{\beta P_4 P_2}{1+\beta h_4 P_2} - \sigma P_2 \tag{1.15}$$

$$\frac{dP_3}{dt} = \frac{\alpha P_3 P_2}{1+\alpha h_3 P_2} + \frac{\rho P_1 P_3}{1+\rho h_2 P_1} - \frac{\gamma P_4 P_3}{1+\gamma h_5 P_3} - \omega P_3 \tag{1.16}$$

$$\frac{dP_4}{dt} = l\left(\frac{\beta P_4 P_2}{1+\beta h_4 P_2}\right) + (n_1 - n_2)\frac{\gamma P_4 P_3}{1+\gamma h_5 P_3} + d\left(1 - \frac{P_2+nP_3}{K_2}\right) P_4 - \mu P_4 - n\delta(P_4)^2 \tag{1.17}$$

Fig. 1.6 shown above depicts the crop production trend and its interaction with pest and predator when we are use Holling Type-II and Gompertz growth. The prameters values

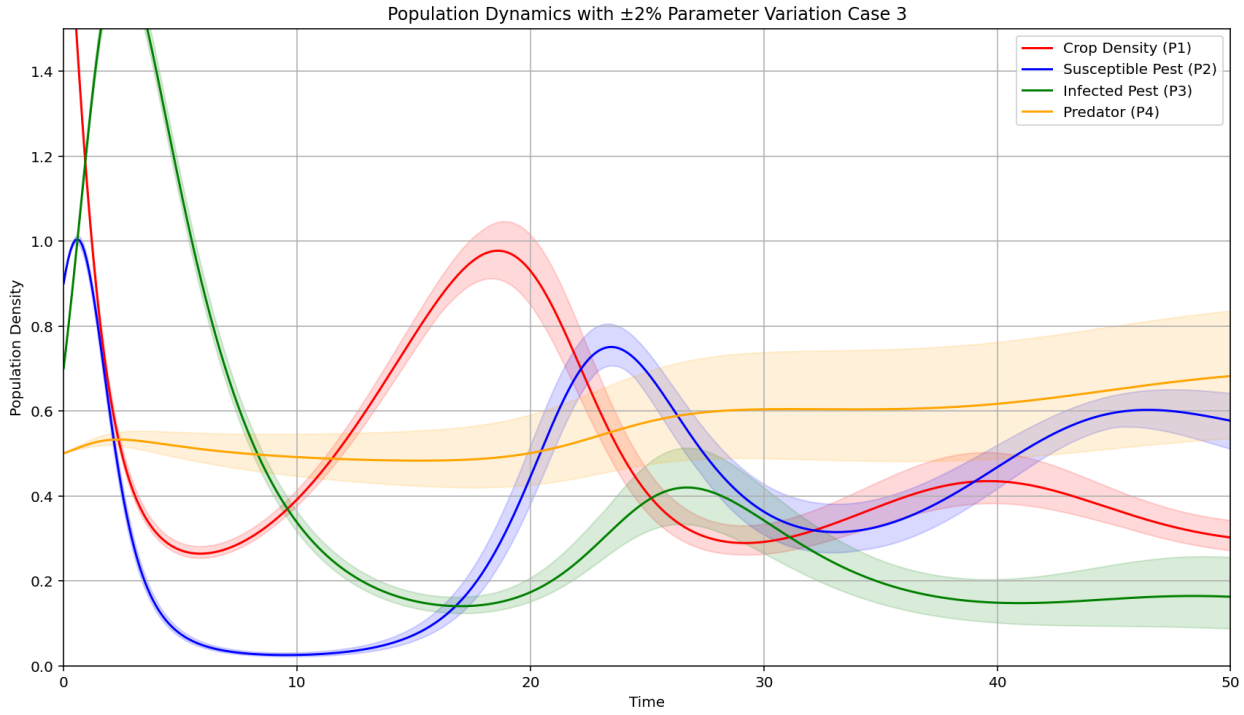


Figure 1.6: Evolution of variable density with time for Gompertz growth function and Holling Type-II for interaction. The error bands around the curves represent the fluctuation in the values of variable due to 2% errors in the model parameters.

are taken from Table1.1.

Note that for a conservative error of 10% in the model parameters can introduce a drmatically large fluctuation from the average values. Therefore, the precision of model parameters play a crucial role in determining the prediction for the growth of crop. A sensitive parameter search is very much required for the study of such systems. This part of the problem will be studied in near future.

1.4.4 Simulation, Parameter selection and their variation

To solve the coupled differential equations of the system we have used Runge–Kutta 4 (RK4) scheme throughout the report, which uses four slopes to calculate the next point of the system. This method gives good accuracy $\mathcal{O}(h^5)$. The parameters are chosen in such a way that the system stays stable for some time. For the purpose of comparision we have chosen to use parameter used by [3]. Any new variables present in the system is also selected in a way that they match the order and range of other parameters used in the model.

To incorporate the possibility of fluctuation in values of parameters present in the system

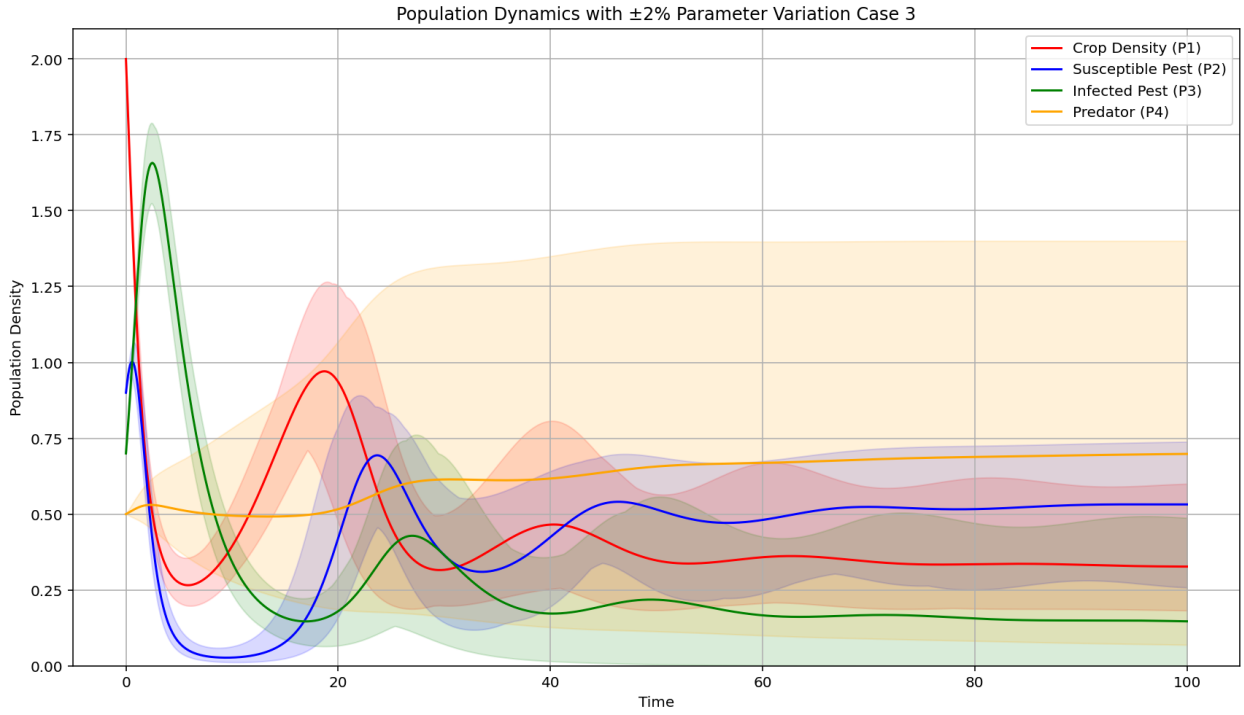


Figure 1.7: Evolution of variable density with time for Gompertz growth function and Holling Type-II for interaction. The error bands around the curves represent the fluctuation in the values of variable due to 10% errors in the model parameters.

we has adopted a uniform fluctuation of 2% for all parameters. We could not find any error estimates of the parameters for such a system. Therefore, we adopted the said variation, to observe this effect. This small error helps to show how sensitive the model is without disturbing its overall stability. If we introduce a larger error, such as 10 %, the error band spreads much more, making the system to diagress by large margin. Therefore, the study of error estimates is necessary for making the right prediction and the right environment for optimal production of crops.

1.5 Conclusion

In this study, we made a simple crop–pest–predator model to understand how they interact in field condition. We used different growth equations and Holling functions for interaction among species to see their effect on system stability. From all the cases, it is observed that Gompertz growth function gives better result for crop because it shows slow start and early saturation which happens in real life crops. Between all Holling functions, Type II gives more stable and realistic result as it includes handling time of predator.

Fig. 1.4,1.5 and 1.6 show variation of four populations with time:

- Case-1 (Logistic + Holling I): the populations goes up and down continuously. The system became stable after some time.
- Case-2 (Logistic + Holling II): Oscillations increases, and the system takes a bit more time to becomes balanced.
- Case-3 (Gompertz + Holling II): System becomes most stable. Crop growth is smooth, pest is controlled, and predator remains positive and limited.

The combination of Gompertz growth with Holling Type II gives most stable and smooth behavior. Crop growth remains positive, pest is controlled, and predator population stays limited. A small fluctuation in the parameters impact the system least compared to other two models and crop production stays stable. So, this model can be used to study real crop–pest–predator system and help in planning better pest control and improving crop yield. Though, a large fluctuation in the parameters can cause a catestrophic changes in the population of predator and also impacts the crop production in all cases.

Parameter	Value	Description
r_1	0.53	Crop growth rate
K_1	5	Carrying capacity for crop
λ	0.45	Susceptible Pest attack rate on crop
σ	0.20	Susceptible Pest natural death rate
d	0.4	predator growth rate*
ρ	0.3	Infected Pest attack rate on crop
r_2	0.6	Susceptible Pest growth rate
η	1	Influence of individual predators on the pest growth rate per capita
K_2	10	Carrying capacity for Susceptible Pest
α	0.5	Infected pest attack rate on Susceptible Pest
γ	0.2	Predator attack rate on infected Pest
β	0.2	Predator attack rate on Susceptible Pest
l	0.5	Predator species contributions from susceptible pests
ω	0.25	Infected Pest natural death rate
n_1	0.3	No of infected pest caught by predator
n_2	0.15	No of infected pest caught by predator which cause death
μ	0.35	Predator natural death rate
δ	0.11	Predator density dependent death rate
c_1	00	Intrinsic growth rate of crop
c_2	00	Intrinsic growth rate of Susceptible Pest
h_1	00	Susceptible Pest handling time for crop
h_2	00	Infected Pest handling time for crop
h_3	00	Infected Pest handling time for Susceptible Pest
h_4	00	Predator handling time for Susceptible Pest
h_5	00	Predator handling time for infected Pest

Table 1.1: Parameter with value and description

CHAPTER 2

Logistic Map

2.1 Introduction

Several biological populations, like species having annual breeding patterns generally evolve in discrete time steps. In such cases, continuous models may not accurately capture the population dynamics. Thus, discrete-time models enable us to study population changes from one generation to the next and uncover behaviors that can be missed in continuous models.

The logistic map is a simple discrete-time model that portrays essential features of population growth under resource limitation. It is defined as:

$$x_{n+1} = r x_n (1 - x_n), \tag{2.1}$$

where x_n is the normalized population at generation n, and r is the growth rate parameter. Although it looks utterly simple in plain sight, the logistic map is extremely interesting as it exhibits a wide variety of behaviors—from stable equilibrium to periodic oscillations, and eventually to chaotic dynamics—as r varies.

Nonlinear dynamical systems can showcase a wide range of behaviors depending on the system parameters. A characteristic of nonlinear systems is their ability to produce unpredictable and complex behavior, even when governed by rules that are deterministic. This phenomenon is known as *chaos*. Any tiny difference in the initial starting points can provide us with completely different outcomes over time. Thus, the chaos is characterized

by sensitive dependence on initial conditions.

Studying the logistic map helps us understand how nonlinear response in systems can generate complex dynamics. It lets us explore the transition from order to chaos, making it a valuable model for exploring population dynamics and the underlying principles of nonlinear biological systems.

2.2 The Logistic Growth Model (Continuous Form)

The logistic growth model is a classical approach to describe population growth under limited resources. It is represented by the differential equation:

$$\frac{dN}{dt} = rN\left(1 - \frac{N}{K}\right), \tag{2.2}$$

where $N(t)$ is the population size at time t, r is the intrinsic growth rate, and K is the carrying capacity of the environment. The term $\left(1 - \frac{N}{K}\right)$ introduces a feedback mechanism that slows growth as the population approaches K, resulting in a sigmoidal (S-shaped) growth curve.

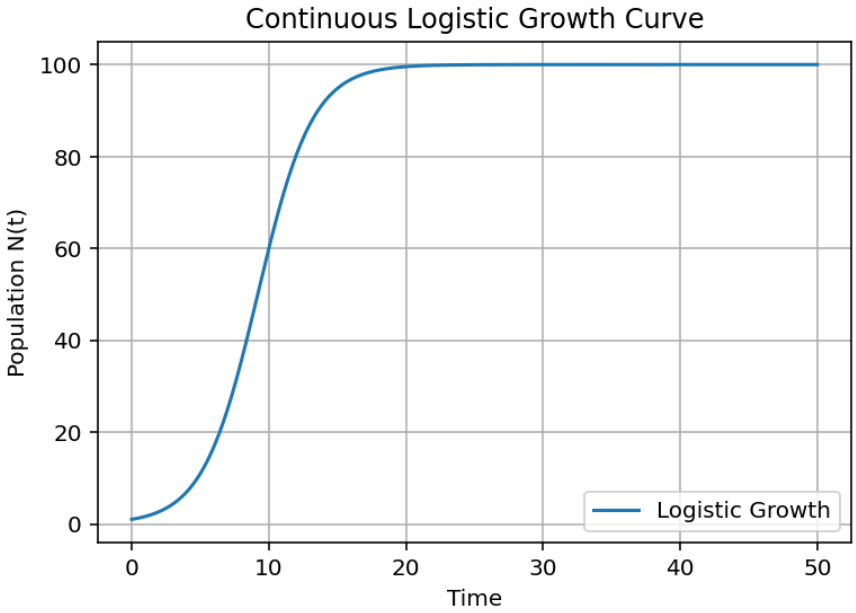


Figure 2.1: Continuous logistic growth curve showing population approaching the carrying capacity.

2.2.1 Transition from Continuous to Discrete Systems

Although the continuous logistic model describes populations in continuous time, many biological populations reproduce in discrete generations. To model such systems, the continuous differential equation can be transformed into a discrete-time form. This transition allows analysis of population changes from one generation to the next and enables exploration of dynamics such as periodic oscillations and chaos, which do not appear in the continuous model.

2.2.2 Derivation of the Logistic Map Equation

Starting from the normalized population $x_n = \frac{N_n}{K}$, where N_n is the population at generation n, the discrete-time logistic map can be derived as:

$$x_{n+1} = rx_n(1 - x_n), \tag{2.3}$$

where r is the growth rate parameter. This simple nonlinear recurrence relation captures the essential feedback mechanism of population growth while operating in discrete time steps.

2.2.3 Interpretation of Parameters

- r – The growth rate parameter, which determines how quickly the population increases. Small values of r lead to stable population levels, while larger values can produce oscillations and chaotic dynamics.
- x_n – The normalized population, defined as the fraction of the carrying capacity at generation n. It allows comparison of populations across different environments and simplifies analysis of the system's dynamics.

2.3 Mathematical Analysis of the Logistic Map

2.3.1 Fixed Points and Their Stability

A fixed point x^* of the logistic map satisfies:

$$x^* = rx^*(1 - x^*). \tag{2.4}$$

Solving this gives two fixed points:

$$x^* = 0, \quad x^* = 1 - \frac{1}{r}. \tag{2.5}$$

The stability of a fixed point is determined by the magnitude of the derivative of the map at that point:

$$f'(x) = \frac{d}{dx}\left[rx(1 - x)\right] = r(1 - 2x). \tag{2.6}$$

A fixed point is stable if $|f'(x^*)| < 1$ and unstable if $|f'(x^*)| > 1$.

2.3.2 Linearization and Eigenvalue Analysis

Linearizing the logistic map around a fixed point gives:

$$x_{n+1} - x^* \approx f'(x^*)(x_n - x^*). \tag{2.7}$$

Here, $f'(x^*)$ acts as an eigenvalue that governs the local behavior. If $|f'(x^*)| < 1$, perturbations decay and the fixed point is stable; if $|f'(x^*)| > 1$, perturbations grow and the fixed point is unstable, leading to oscillations or more complex dynamics.

2.3.3 Period Doubling and Bifurcation Phenomena

As the growth rate r increases beyond certain thresholds, the system undergoes *period-doubling bifurcations*. The population oscillates between two, four, eight, etc., values in successive generations. This cascade of period doublings is a hallmark of nonlinear discrete systems and serves as a pathway to chaos.

2.3.4 Routes to Chaos (Overview)

The logistic map exhibits multiple routes to chaos, with period-doubling being the most well-known. Other routes include quasiperiodicity and intermittency. Chaos in this context is characterized by:

- Sensitive dependence on initial conditions
- Aperiodic long-term behavior
- Dense periodic points within the chaotic region

These phenomena illustrate how simple deterministic rules can generate complex, unpredictable dynamics, making the logistic map a paradigmatic example in the study of nonlinear systems and chaos theory.

2.4 Numerical Simulation and Visualizations

2.4.1 Iterative Computation of the Logistic Map

The logistic map is studied numerically by iterating the recurrence relation

$$x_{n+1} = r x_n (1 - x_n)$$

over multiple generations. For small growth rates, the population stabilizes to a fixed point, while higher growth rates lead to oscillations or chaotic fluctuations. Figure 2.2 shows

an example of iterative computation for a moderate growth rate, illustrating convergence towards a periodic orbit.

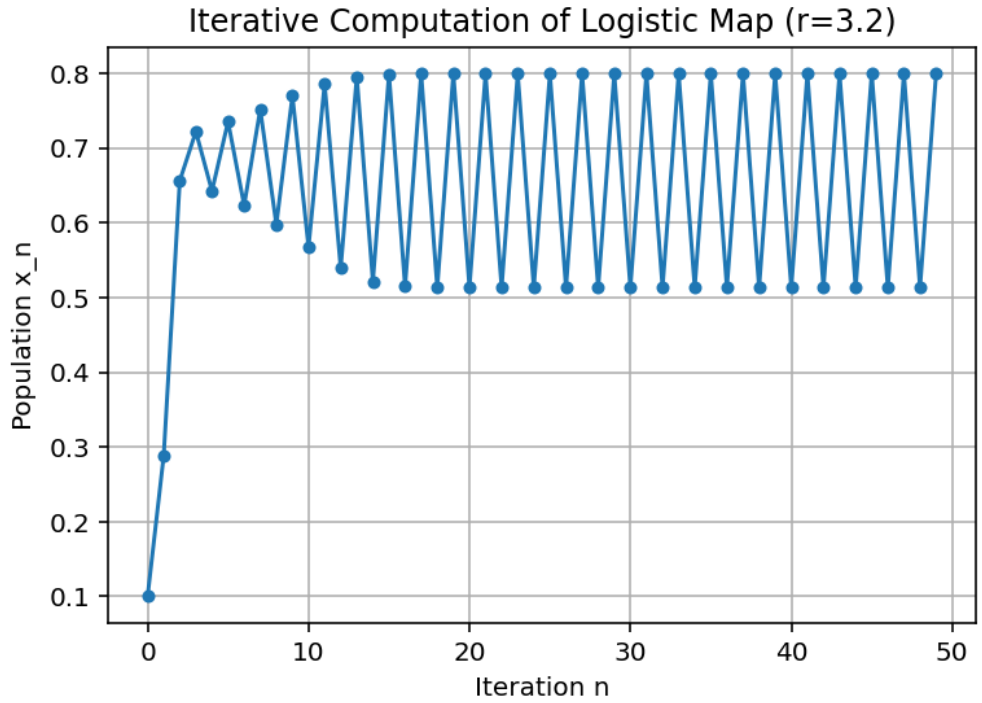


Figure 2.2: Iterative computation of the logistic map for a specific growth rate r.

2.4.2 Time Series Analysis for Various Growth Rates

Time series analysis reveals how different growth rates influence the dynamics of the logistic map. Low values of r lead to smooth convergence to a fixed population, intermediate values produce periodic oscillations, and high values result in irregular, chaotic behavior. Figure 2.3 depicts the population evolution for several growth rates, demonstrating these patterns clearly.

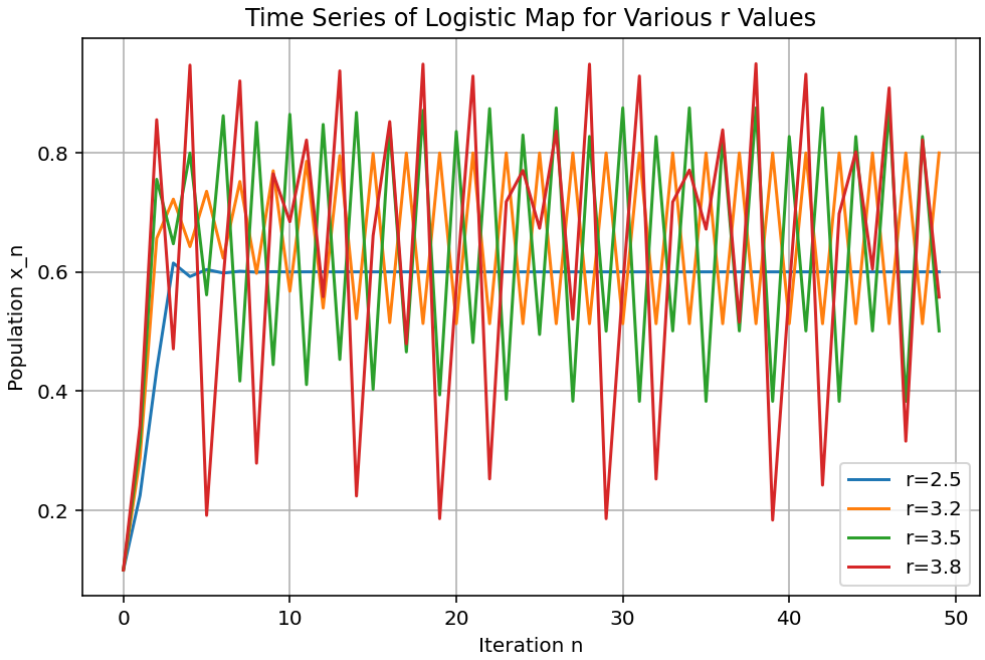


Figure 2.3: Time series of the logistic map for various growth rates r.

2.4.3 Cobweb Diagrams

Cobweb diagrams provide a graphical method to understand the iteration process of the logistic map. By plotting successive iterations on the function curve and the identity line, one can visualize convergence to fixed points, periodic orbits, or divergence into chaotic behavior. Figure 2.4 shows a cobweb plot for a representative growth rate, highlighting the approach to a period-2 cycle.

2.4.4 Bifurcation Diagram

The bifurcation diagram captures the long-term behavior of the logistic map as the growth rate r varies. It clearly demonstrates the period-doubling route to chaos: stable fixed points transition into period-2, period-4, and higher periodic orbits before becoming chaotic. Figure 2.5 visualizes this progression, revealing the structural changes in system dynamics and critical points of bifurcation.

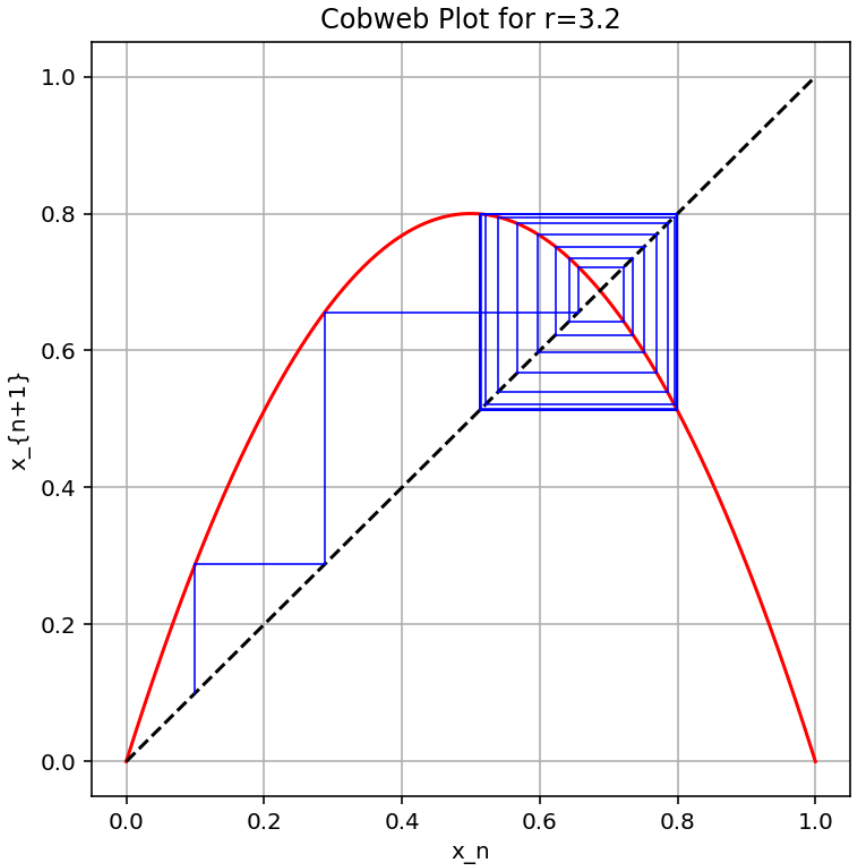


Figure 2.4: Cobweb diagram of the logistic map illustrating convergence and oscillatory behavior.

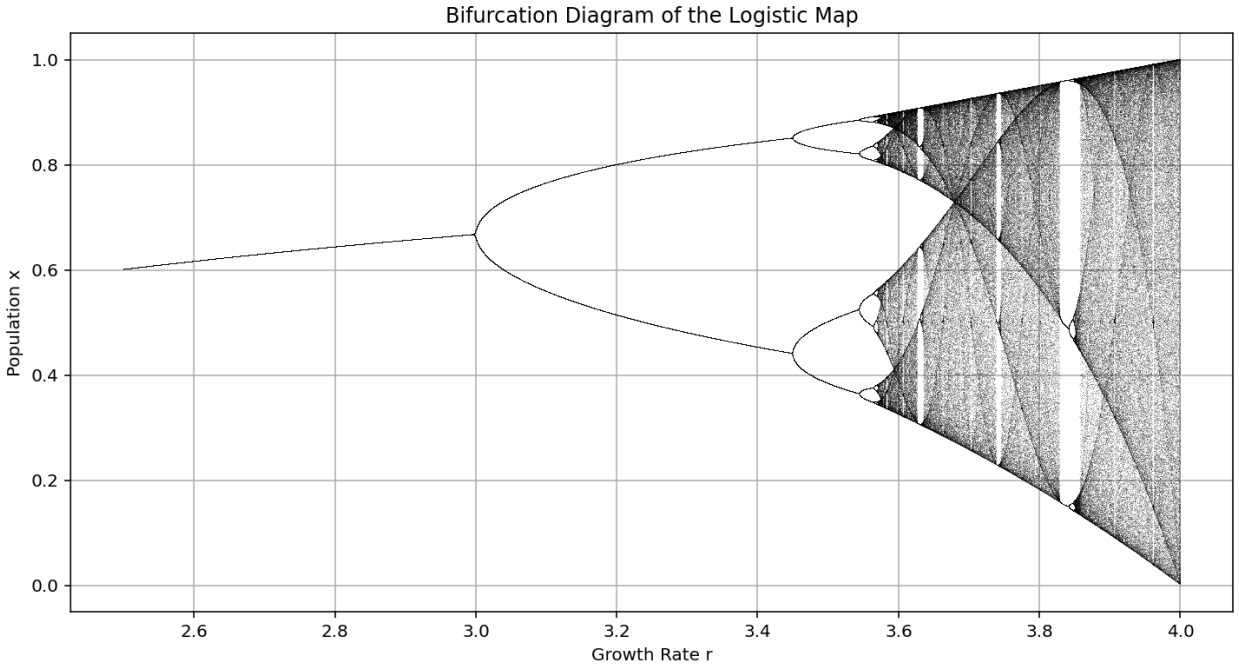


Figure 2.5: Bifurcation diagram showing period doubling and the transition to chaos as a function of growth rate r.

2.4.5 Lyapunov Exponent Plot

The Lyapunov exponent provides a quantitative measure of chaos by indicating the sensitivity to initial conditions. Positive exponents correspond to chaotic regimes, while negative values indicate stability. Figure 2.6 presents the Lyapunov exponent as a function of r, showing ordered and chaotic regions and complementing the insights from the bifurcation diagram.

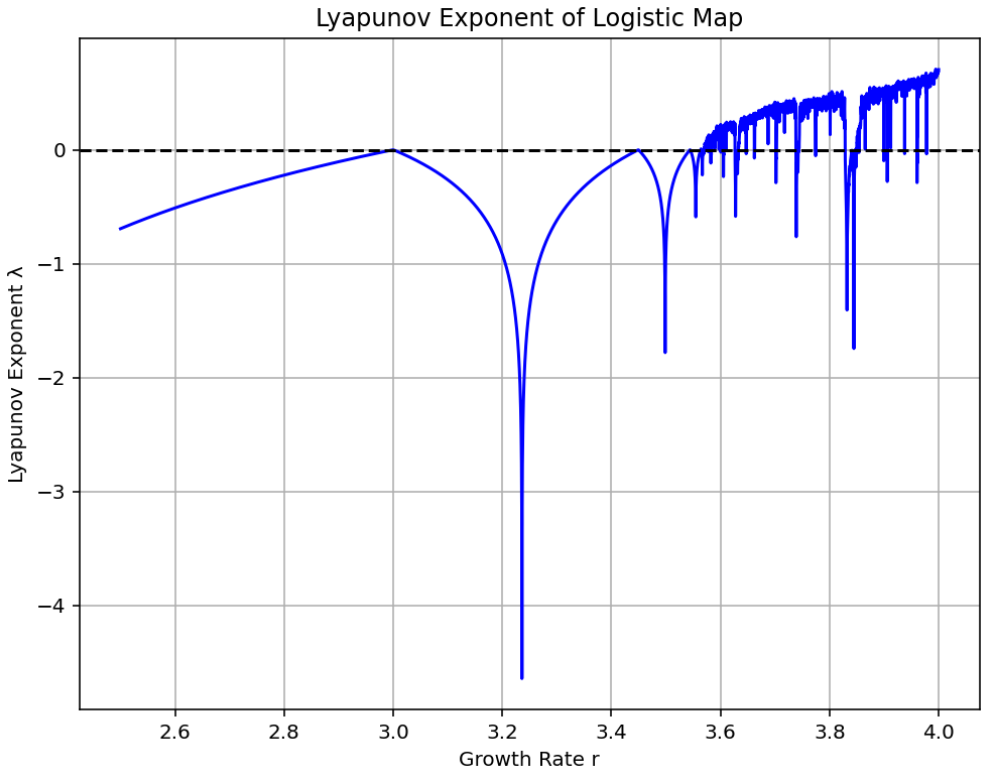


Figure 2.6: Lyapunov exponent of the logistic map as a function of growth rate r, with positive values indicating chaotic behavior.

2.5 Discussion of Results

2.5.1 Behaviour Classification Based on r

The simulations demonstrate that the logistic map exhibits distinct behaviors depending on the growth rate r. For low values of r ($0 < r < 3$), the population converges to a stable fixed point, corresponding to a predictable long-term population size. As r increases ($3 < r < 3.57$), the system undergoes period-doubling bifurcations, producing periodic

oscillations with increasing periods. Beyond this threshold, for $r > 3.57$, the map enters a chaotic regime, characterized by highly irregular, aperiodic fluctuations. These classifications are consistent with the theoretical predictions of nonlinear dynamical systems.

Table 2.1: Long-Term Behavior of the Logistic Map for Different Values of r

Range of r	**Long-Term Behavior of x_n**	**Description**
$0 < r \leq 1$	Extinction ($x_n \to 0$)	The population inevitably dies out.
$1 < r \leq 3$	Stable Fixed Point $\left(x_n \to \frac{r-1}{r}\right)$	The population converges to a single, non-zero steady-state value.
$3 < r \leq 3.449\ldots$	Period Doubling (Period 2, 4, 8, ...)	The population oscillates between a set of 2, 4, 8, etc., distinct values (cycles).
$r \approx 3.5699456\ldots$	Onset of Chaos	The point where the period-doubling cascade infinitely accelerates, marking the transition to chaos.
$3.5699456\ldots < r \leq 4$	Chaotic Regime	The population never settles into a stable value or repeating cycle (for most values of r). It exhibits sensitive dependence on initial conditions.
$r > 4$	Population Explosion	x_n values exceed 1 (the carrying capacity), meaning the model breaks down as population must remain ≤ 1.

2.5.2 Comparison Between Theoretical and Simulated Outcomes

The theoretical analysis of fixed points, stability, and bifurcations aligns closely with the numerical simulations. The fixed point $x^* = 1 - 1/r$ accurately predicts the stable population

for lower growth rates, while cobweb diagrams and iterative computations visually confirm the convergence to this point. Similarly, the period-doubling sequence predicted analytically is clearly observed in both the time series and bifurcation diagrams. This agreement reinforces the validity of the logistic map as a model for discrete-time population dynamics.

2.5.3 Interpretation of Chaotic Behaviour

Chaos in the logistic map arises due to the nonlinear feedback inherent in the recurrence relation. In this regime, the system shows sensitive dependence on initial conditions: even infinitesimal differences in starting populations lead to drastically different trajectories. The bifurcation diagram and Lyapunov exponent plot quantify this behavior, with positive Lyapunov exponents marking regions of unpredictability. This demonstrates that deterministic rules can generate complex, seemingly random outcomes without any stochastic influence.

2.5.4 Relation to Real Biological Systems

The chaotic dynamics observed in the logistic map provide insight into real biological populations that exhibit unpredictable fluctuations, such as certain insect or microbial populations. Environmental limitations, resource competition, and generational reproduction create discrete, nonlinear feedback mechanisms similar to those captured by the logistic map. Understanding these dynamics can help in predicting population trends, managing ecosystems, and interpreting irregular patterns in natural populations, even when the underlying processes are deterministic.

2.6 Extended Analysis

2.6.1 Estimation of Feigenbaum Constants

The logistic map exhibits a universal period-doubling route to chaos. The ratio of successive bifurcation intervals approaches a constant value, known as the Feigenbaum constant $\delta \approx 4.669$. This universality applies to a wide class of one-dimensional maps with a sin-

gle quadratic maximum. Using the bifurcation diagram (Figure 2.5), the intervals between successive period-doubling points can be measured to estimate δ, demonstrating the deep connection between discrete nonlinear systems and universal scaling laws in chaos theory.

2.6.2 Period-3 Windows and the Onset of Chaos

Embedded within the chaotic regime are narrow windows where periodic behavior re-emerges, such as period-3 cycles. According to the period-three theorem, the existence of a period-3 orbit implies the presence of periodic orbits of all higher orders, as well as chaotic dynamics. These windows are visible as small islands of regular oscillations in the bifurcation diagram, illustrating that chaos is interspersed with pockets of order. Such structures emphasize the intricate and self-similar nature of the logistic map's dynamics.

2.6.3 Lyapunov Exponents and Fractal Dimension

Lyapunov exponents quantify the sensitivity of the system to initial conditions. Positive exponents indicate exponential divergence of nearby trajectories, confirming chaotic behavior. Moreover, the attractor of the logistic map in the chaotic regime exhibits a fractal structure. The fractal (or Hausdorff) dimension provides a measure of the complexity of the attractor, capturing the dense but non-continuous nature of chaotic trajectories. Lyapunov exponents and fractal dimension together offer a rigorous way to characterize the unpredictability and geometrical complexity of the logistic map.

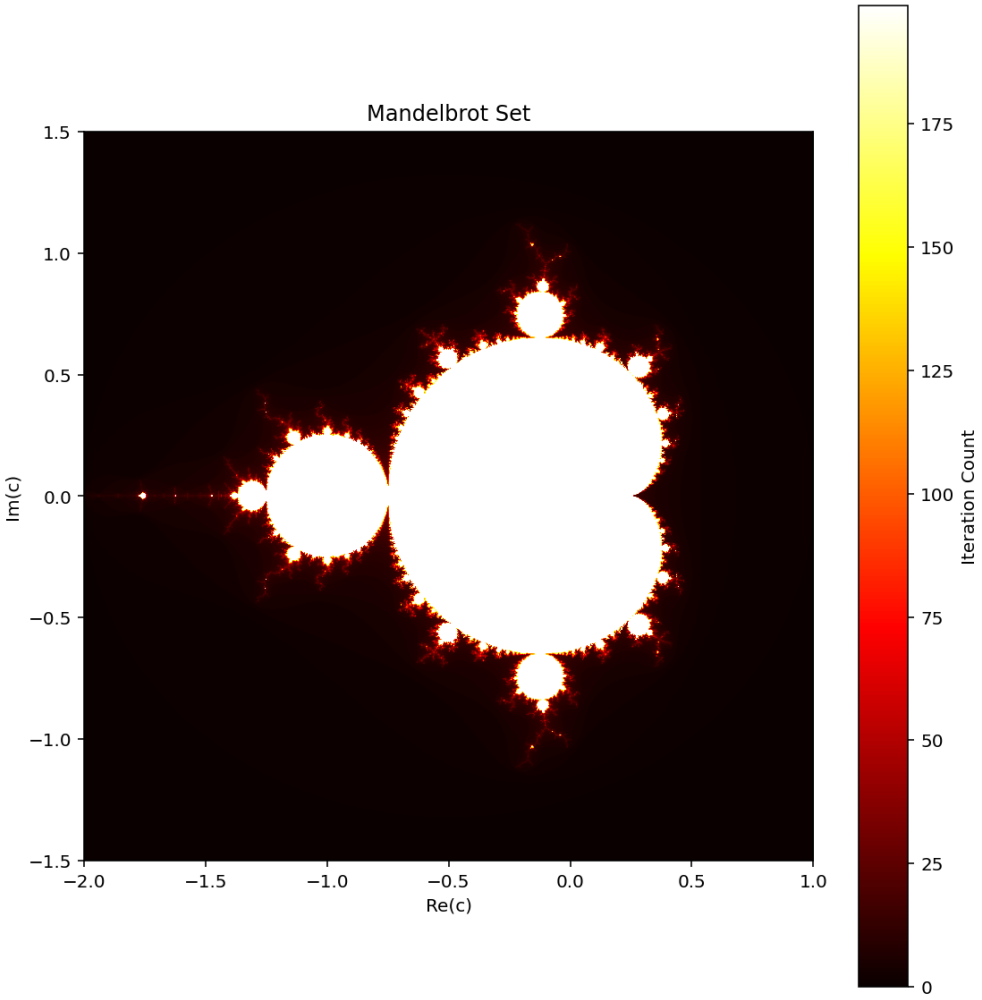
2.6.4 Connection to the Mandelbrot Set

The complex behavior of the logistic map is not limited to real numbers. When the logistic map is extended to the complex plane, it relates closely to the Mandelbrot set, a quintessential example of a fractal in complex dynamics. The Mandelbrot set is defined by iterating the quadratic map

$$z_{n+1} = z_n^2 + c,$$

where z and c are complex numbers. Interestingly, the period-doubling route to chaos observed in the real logistic map mirrors the bifurcation structure found along the boundary of the Mandelbrot set. Regions of stability and chaos in the logistic map correspond to connected and disconnected regions in the Mandelbrot set, highlighting the universality of nonlinear dynamics.

This connection underscores that the intricate, self-similar structures observed in the logistic map's bifurcation diagram and its chaotic attractors are not isolated phenomena. Instead, they are part of a broader framework of complex dynamical systems where simple iterative rules can generate fractals and chaos in both real and complex domains. Visualizing the Mandelbrot set alongside logistic map bifurcations provides a powerful demonstration of how mathematical iteration bridges simple deterministic rules and infinite complexity.



2.7 Conclusion

This study explored the logistic map as a fundamental model of discrete-time population dynamics and a prototype of chaos in biological systems. Through theoretical analysis and numerical simulations, we classified the behavior of the logistic map across different growth rates, observing stable fixed points, periodic oscillations, and chaotic regimes. The iterative computations, cobweb diagrams, bifurcation diagram and Lyapunov exponent plots provided a comprehensive visualization of these dynamics.

The simulations highlighted key insights: deterministic systems with simple nonlinear feedback can produce highly complex and unpredictable behavior; small changes in initial conditions or growth parameters can lead to drastically different outcomes; and the onset of chaos occurs through a well-defined period-doubling route. These findings underscore the sensitive dependence on initial conditions and the fractal structure inherent in chaotic systems.

The logistic map serves not only as a mathematical abstraction but also as a valuable tool for understanding real-world biological phenomena, including population fluctuations, resource-limited growth, and patterns in epidemics or gene frequencies. Beyond biology, its relevance extends to complex systems in economics, neuroscience, and ecology, illustrating how simple rules can generate intricate, emergent behaviors. Overall, the logistic map provides a bridge between order and chaos, offering profound insights into the dynamics of both natural and artificial systems.

CHAPTER 3

Gene Networks

3.1 Introduction

Our whole body is controlled by the gene and it's products. Gene is considered to be one of the fundamental unit which control almost everything in our body. This makes the study of gene more important than any other field of biology. Genes undergo the process of transcription to produce messenger ribonucleic acid (mRNA) which further undergoes the process of translation to produce the protein and the protein eventually led to expression of characters and different characteristics in our body. Transcription is a process of copying genetic information from Deoxyribonucleic acid (DNA) into mRNA, and translation is a process where the genetic code carried by mRNA is used by ribosomes to synthesize proteins, forming a chain of amino acids. In our inner body gene are mostly not isolated they are always in interaction with each other and other regulatory substances. These substances take part in production or degradation of mRNA and DNA produced by gene. This whole interaction between the gene and it's regulators and surroundings create a whole network interconnection, which is known as Gene Network. [4].

The basic idea behind Gene Regulatory Networks (GRNs) is the behaviour of gene to not act in isolation but act in a special interconnected manner. Transcription factors - a type of protein - which regular the process of transcription plays a important role in GRNs. These factors are also produces by genes. So,for a continuous flow, both genes have to interact which give rise to different conditions like feedback loop, oscillation, bistability. Gene regulatory networks are central to many critical biological processes including embryonic

development, cell cycle regulation, immune responses, and stress adaptation. A disruption in GRN function is often associated with diseases, particularly cancer, where the deregulation of gene expression programs leads to uncontrolled cell proliferation and metastasis.

We will study mathematical models of the GRNs. Mostly these type of system are modelled through ODEs because, in general, the dynamics of any system is represented by relations of rate of change of dynamical variables of the system. For a realistic model we have a set of coupled differential equations. We use different numerical method to solve and stimulate them on computer. The generalized RK4 is a popular and quite accurate algorithm to solve such systems.

In our system we have mRNas, proteins or may also include transcription factors. In toy model we may only include constant degradation and production rate of mRNA and protein but in more realistic models governing complex dynamics, such as regulatory feedback and bistability, the system become complex to interpret before solving it. We employ RK4 method to solve the system [6]. A custom generalized RK4 routine has been implemented in Python as part of this project. This function supports arbitrary number of functions $f(t, y)$, fixed time-stepping, and clear code transparency.

3.1.1 Generalized Rungekutta Method

```
import numpy as np
def rk4(f,t,y0):
N = len(t)
n = len(y0)
y = np.zeros((N,n))
y[0] = y0

for i in range(1,N):
h = t[i] - t[i-1]
t_previous = t[i-1]

y_old = y[i-1]

k1 = h*np.array(f(t_previous,y_old))
```

```
    k2 = h*np.array(f(t_previous + h/2,y_old + k1/2))
    k3 = h*np.array(f(t_previous + h/2,y_old + k2/2))
    k4 = h*np.array(f(t_previous + h,y_old + k3))

    y[i] = y_old + (k1 + 2*k2 + 2*k3 + k4)/6
    return y
```

Listing 3.1: Generalized RK4

Here, h is the time step, and $f(t, y)$ defines the system of ODEs. The method provides a local error of order $\mathcal{O}(h^5)$ and a global error of order $\mathcal{O}(h^4)$, making it more accurate than Euler's method or the second-order methods. We will use this method to find time series solution ans phase portraits. For more details on Runge Kutta 4^{th} order method see [5] or [6].

3.2 Introducing Gene Networks Models

Instead of directly jumping into modelling bistable and feedback system we start with the simple toy model as a foundational method. A gene that is transcribed at a constant rate and degraded proportionally to its concentration. The regulation of gene expression begins with transcription — the synthesis of mRNA from a DNA template and this mRNA in turn goes under translation to produce protein. While modelling our system we have to keep this in mind that the amount of the gene itself does not change, instead only the concentration of mRNA and protein changes. Initially, we will consider the regulated expression of a single gene. Later, we will also consider cases where genes can interact by regulating each others' expression.

3.2.1 Simple Model

In our model the two main variables are :

1. $M(t)$ = Concentration of mRNA.
2. $P(t)$ = Concentration of Protein.

These two varibale will be controlled by their production and degradation. The process of mRNA production is called transcription and the production of protein is called translation. Here we assume that:

1. The rates of mRNA and protein degradation are proportional to their concentration .
2. The rate of translation is proportional to the mRNA concentration.

Thus,

$$\frac{dM}{dt} = R(t) - \mu M \tag{3.1}$$
$$\frac{dP}{dt} = kM - \nu P \tag{3.2}$$

Where, μ is degradation rate of mRNA, ν is degradation rate of Protein, k is Translation rate and $R(t)$ is the Transcription rate. For in depth details about this system see [1, 2]. A sensible biologically system will have positive μ, ν, k and $R(t) \geq 0$. Also the equilibrium of our model depends upon the nature of $R(t)$. If $R(t)$ is constant the model will have a equilibrium point else it will not reach to equilibrium.

3.2.2 Constant Transcription

The model behaviour depends on the nature of $R(t)$. We will study the behavior of different cases for $R(t)$ starting with constant $R(t)$ i.e. the transcription rate is constant. Let $R(t) = r$. Thus,

$$\frac{dM}{dt} = r - \mu M \tag{3.3}$$
$$\frac{dP}{dt} = kM - \nu P \tag{3.4}$$

You can easily find the stimulation and solution of these differential equations in [?]. Here we consider a small variance, as possible uncertainty, in the values of μ, ν, k to check for stability, robustness and requirement of precision study in the model parameters. The mRNA nullcline for the system is at $M = \frac{r}{\mu}$ and the Protein nullcline for the system is at $M = \frac{\nu P}{k}$.

[1]For Complete Solution refer to [2] Chapter 12 Introducing gene networks

3.2.3 Python Implementation

```
import numpy as np
import matplotlib.pyplot as plt
from Rungekutta import rk4

y0 = [0,0]
t = np.linspace(0,20,1000)
R = 2
mu = 1
nu = 0.5
k = 1

e = 0.01
iterations = 200
rng = np.random.default_rng(1234)

def ConstantTranscription_factory(R_val):
    def f(t_local, y):
        M, P = y
        dMdt = R_val - mu * M
        dPdt = k * M - nu * P
        return [dMdt, dPdt]
    return f

Total_M = []
Total_P = []

for i in range(iterations):
    r1 = rng.uniform(R * (1 - e), R * (1 + e))
    ode_fun = ConstantTranscription_factory(r1)
    sol = rk4(ode_fun, t, y0)
    sol = np.array(sol)
    M = sol[:, 0]
    P = sol[:, 1]
   Total_M.append(M)
```

```
    Total_P.append(P)

M_arr = np.array(Total_M)

P_arr = np.array(Total_P)
mean_M = M_arr.mean(axis=0)
min_M  = M_arr.min(axis=0)
max_M  = M_arr.max(axis=0)

mean_P = P_arr.mean(axis=0)
min_P  = P_arr.min(axis=0)
max_P  = P_arr.max(axis=0)

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))
ax1.plot(t, mean_M, label='Mean M')
ax1.plot(t, max_M, color='red',   linewidth=0.6, label='Max M')
ax1.plot(t, min_M, color='green', linewidth=0.6, label='Min M')
ax1.fill_between(t, min_M, max_M, color='cyan', alpha=0.5)
ax1.set_title('M (mRNA)')
ax1.grid(True)
ax1.legend()

ax2.plot(t, mean_P, label='Mean P')
ax2.plot(t, max_P, color='red',   linewidth=0.6, label='Max P')
ax2.plot(t, min_P, color='green', linewidth=0.6, label='Min P')
ax2.fill_between(t, min_P, max_P, color='cyan', alpha=0.5)
ax2.set_title('P (Protein)')
ax2.grid(True)
ax2.legend()
plt.tight_layout()
plt.show()
```

Listing 3.2: Python Code for error stimulation

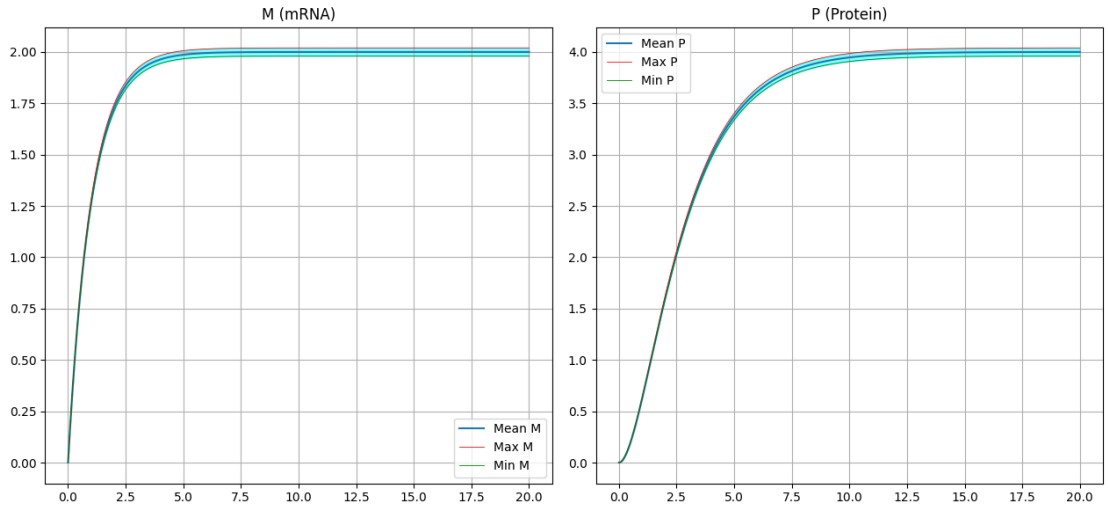


Figure 3.1: Time-course of single gene model, with $r = 2, \mu = 1, \nu = 0.5, k = 1$ and initial concentrations $P(0) = M(0 = 0)$ with a error of 0.01 in r

Fig.3.1 suggests that for small uncertainty protein production remains close to mean value. We find that in the beginning transcription fluctuation is absent. Only a small fluctuations as time increases which increases with proportionality of amount of variation in values of parameters. Thus late time effects varies and may be large from system to system. See Fig.**??** and 3.2.

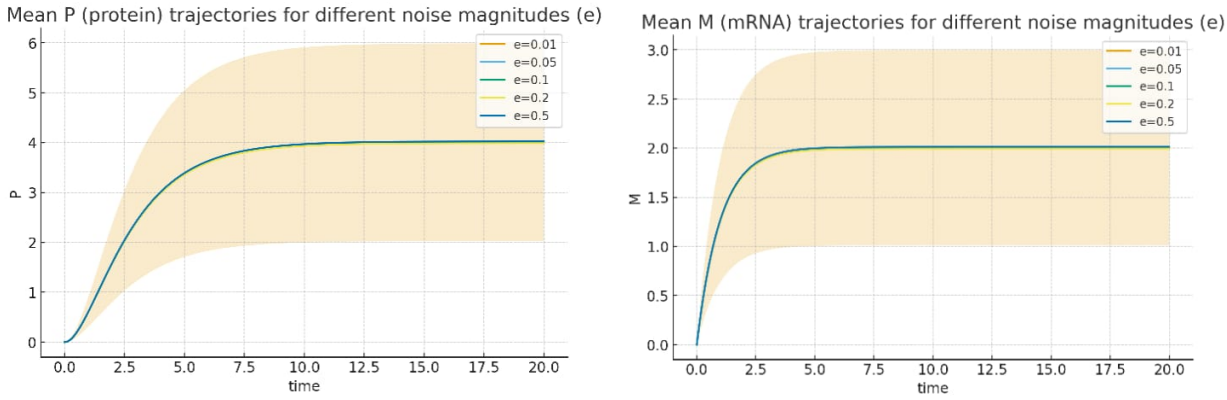


Figure 3.2: Protein and mRNA Graph for large uncerainty in model parameters.

Thus, We can say that the with the increasing error in value of parameters the standard deviation from the mean position also increases and predictability of model is compromised as lot.

3.2.4 Phase Portrait Graph

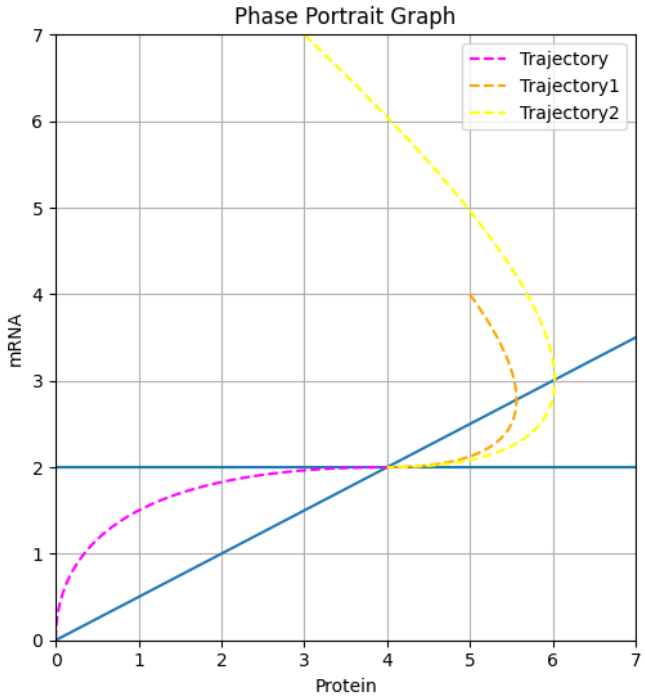


Figure 3.3: Phase Portrait graph showing nullclines and stable point and different trajectories

3.2.5 Oscillating Transcription

In a real biological system it is rare to find a system with constant transcription. A gene is not always on. A gene is not working all the time. Instead, a gene remains in two states: ON and OFF. Gene oscillate between these two states. So, to make a more realistic model, we will consider the oscillating transcription. A simple model is a sinusoidal transcription rate to mimic the behavior of on and off states.

Let us take the transcription function as $R(t) = r0(1 + \delta \sin(\omega t))$. This has an average value of r_0 but varies between 0 and $2r_0$, with the period depending on ω.

3.2.6 Python Implementation

Listing 3.3: Python Code for Oscillating transcription model

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

```
from Rungekutta import rk4

y0 = [0,0]
t = np.linspace(0,20,1000)
mu = 1
nu = 0.5
k = 1
r = 2
delta = 0.75
omega = 2*np.pi

def R(t):
return r*(1 + delta*(np.sin(omega*t)))

def OscillatingTranscription(t,y):
M,P = y
dMdt = R(t)  - mu*M
dPdt =k*M - nu*P
return [dMdt,dPdt]

sol = rk4(OscillatingTranscription,t,y0)

mRNA = sol[:,0]
Protein = sol[:,1]

y01 = [4,5]
y02 = [7,3]
sol1 = rk4(OscillatingTranscription,t,y01)
sol2 = rk4(OscillatingTranscription,t,y02)

mRNA1 = sol1[:,0]
Protein1 = sol1[:,1]
mRNA2 = sol2[:,0]
Protein2 = sol2[:,1]
```

```
P_values = np.linspace(0,20,1000)
M_null = R(t)/mu
P_null = (nu*P_values)/k

fig , (ax1 , ax2) = plt.subplots(1,2)
ax1.plot(t,mRNA,label='mRNA')
ax1.plot(t,Protein,label='Protein')
ax1.set(xlabel = 'Time' , ylabel = 'Concentration of
    Cells' , title = 'Time Course Graph')
ax1.legend()
ax1.grid(True)

ax2.plot(P_values,M_null)
ax2.plot(P_values,P_null)
ax2.plot(Protein,mRNA,'magenta',label='Trajectory',
   linestyle='--')
ax2.plot(Protein1,mRNA1,'orange',label='Trajectory1'
   ,linestyle='--')
ax2.plot(Protein2,mRNA2,'yellow',label='Trajectory2'
   ,linestyle='--')
ax2.set(xlabel = 'Protein' , ylabel = 'mRNA' , title
    = 'Phase Portrait Graph')
ax2.legend()
ax2.grid(True)
ax2.axis([0,7,0,7])
plt.show()
```

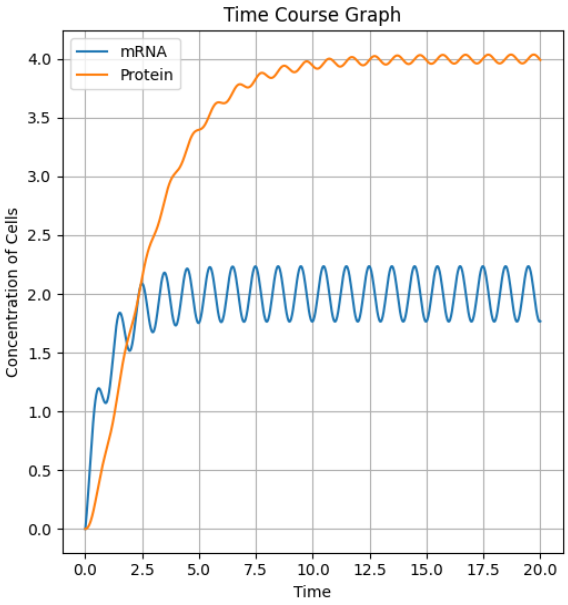
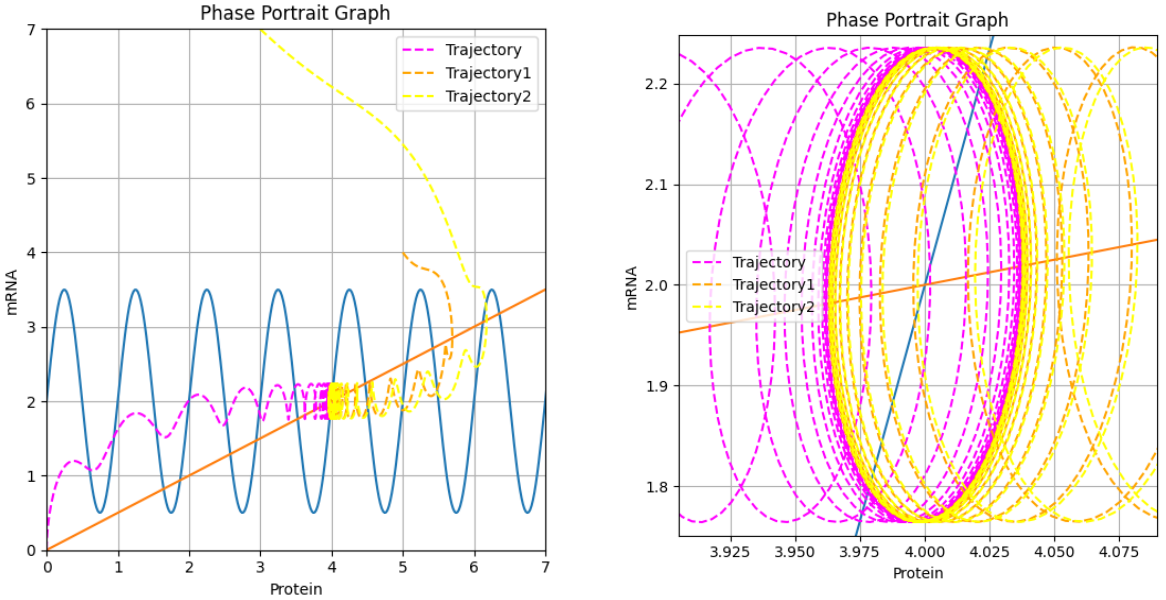


Figure 3.4: Time-course of single gene oscillating transcription model.

Fig.3.4 depicts more realistic gene transcription. A phase delayed behavior of protein production following the abundence of mRNA is clearly visible.



(a) Phase Portrait for Given Model

(b) No stable point in oscillating model

Figure 3.5: General caption.

In the Fig.3.5 we clearly see that the concentrations keep oscillating around a particular value but they never converges to a single concentration. Fig.3.5b is zoomed out image of Fig.3.5a. This behavior is expected because if the gene is getting on and off again and again it keeping varying the value of concentration and does not converge or saturate.

3.3 Biological Significance and Interpretation

The two transcription models analyzed so far: One with constant transcription rate and other with oscillating transcription rate providing foundational insights into different types of gene expression patterns observed in biological system.

3.3.1 Constant Transcription

Although the constant transcription model not really found in biological system but represents toy model of genes representing basic cellular functioning of biological system.

The main result which can be interpreted from our modelling are as follows:

- The equilibrium position is robust to initial conditions.From whatever initial condition we start with we eventually converge/end at the same state.
- For small errors the system remains almost same the standard deviation is very small.
- With the increasing error the standard deviation increases. By changing the value of R, we can change the state of the system.

3.3.2 Oscillating Transcription

In the oscillating model the main result is that system doesn't converges to a particular state instead it keep oscillating around a particular concentration. In the biological system it basically represent the systems which work in a cycle or genes which works going on and off in between.

- Circadian genes in mammals,which are transcribed rhythmically with a 24-hour period.
- Genes regulated by cell cycle oscillations or periodic environmental cues(e.g.light/dark cycles).

The message we can take out is that the noises can affect the functioning of gene and a change in the way gene transcribe the mRNA can change the stability of system.

3.4 Auto Regulation of genes

In our previous models we simply considered one mRNA and one Protein differential equation which act independently. The real cell systems are not the only the said type of systems. Instead, a real systems can also be auto regulated. Auto regulation of genes mean that the protein produced by the mRNA itself effect the rate at which the mRNA is produced. The protein produced itself act as a transcription factor for the gene. This regulation can be of two types:-

- Negative Feedback
- Positive Feedback

To model these loops we will change the differential equations 3.1 and 3.2 by introducing a function $f(P)$ in the equation 3.1 to complete the feedback loop. So the model looks like:

$$\frac{dM}{dt} = f(P) - \mu M \tag{3.5}$$

$$\frac{dP}{dt} = kM - \nu P \tag{3.6}$$

where μ and ν are degradation rates, k is the translation rate. The kind of feedback loop which our model will represent depends upon the nature of function $f(P)$. We can introduce different kind of functions to study different biological phenomenon.

3.5 Negative Feedback

We have already discussed that in a biological system the DNA produces mRNA and the mRNA further produces protein. Whenever the protein produced by the mRNA effects the production of mRNA, by repressing it, is called a negative feedback. The mRNA produces the protein and the protein represses the mRNA thus reducing the protein production hence reduction in suppression of mRNA production. Suach a process is called negative feedback loop.

3.5.1 Model

Instead of using the hill function to represent the behaviour of the repression loop we have used the gamma function. Why we gave importance to gamma function will be explained later.

$$\begin{aligned} f(P) &= \frac{1}{1+\alpha g(P)} \quad \text{The Hill function} \\ g(P) &= \frac{P^{K-1}e^{\frac{-P}{\theta}}}{\Gamma(K)\,\theta^{K}}. \quad \text{The Gamma function.} \end{aligned}$$

where α, K, θ are parameters of Hill and Gamma repression function.

3.5.2 Why Gamma instead of Hill

In most of models hill function is used to model the system,but the Hill function is monotonic function it basically represent the cooperative binding of transcription factor and through gamma function in a repression model we basically get one stable state only [2].

There are many system in biology that are found to not follow the hill function behaviour instead they follow different kind of behaviour. The gamma function is a non monotonic function weak repression at very low and very high P and maximal repression around an intermediate concentration. This type of behaviour is exhibit by many genes mainly those interactions which take place in different steps also with the help of gamma function we can study the bistability of the system.

3.5.3 Equilibria Analysis

The nullclines are

$$\begin{aligned} M &= \frac{f(P)}{\mu} \\ M &= \frac{\nu}{k}P \end{aligned}$$

Because $f(P)$ is U-shaped the two nullclines may intersect either once or three times.

3.5.3.0.1 Linear stability : Linearization yields

$$J = \begin{pmatrix} -\mu & f'(P^*) \\ k & -\nu \end{pmatrix}, \qquad \det(J) = \mu\nu - kf'(P^*).$$

Since $\mathrm{tr}(J) = -\mu - \nu < 0$, stability reduces to $\det(J) > 0$. Using

$$f'(P) = -\frac{\alpha\, g'(P)}{\big(1+\alpha g(P)\big)^2}, \qquad g'(P) = g(P)\Big(\frac{K-1}{P} - \frac{1}{\theta}\Big),$$

Now the stability depends upon the sign of $f^{'}(p)$.As $g^{'}(p)$ changes sign at $P = (K-1)\theta$ which is also the mode of the gamma function.To the left of mode $f^{'}(p)$ is negative and positive on the right of mode. Mostly the left and right interactions are stable and the middle point of interaction is unstable. Even a little shift from middle point will take the system to anyone on right or left state.

3.5.4 Python implementation

The code given below is a basic code which can be used for the implementation.The graph Given is generated from a more complex code just not to make coding part difficult I am providing basic code for advance graph anyone can modify it.

```
import numpy as np
import matplotlib.pyplot as plt
from Rungekutta import rk4
import math as mt


t     = np.linspace(0,20,50)
y0    = [5,5]
K     = 5
theta = 1
alpha = 50
mu    = 1
k     = 10
nu    = 0.5

```

```
15  def f(P):
16  f = 1/(1 + (alpha*((P**(K-1))*np.exp(-P/theta)))/(mt.gamma(K)*(
        theta**K)))
17  return f
18
19  def NegativeFeedback(t,y):
20  M,P  = y
21  dMdt = f(P) - mu*M
22  dPdt = k - nu*P
23  return [dMdt,dPdt]
24
25  sol     = rk4(NegativeFeedback,t,y0)
26  mRNA    = sol[:,0]
27  Protein = sol[:,1]
28
29  y01 = [0.2,1]
30  y02 = [0.5,15]
31  y03 = [0.21,10]
32  y04 = [0.1,5]
33
34  sol1 = rk4(NegativeFeedback,t,y01)
35  sol2 = rk4(NegativeFeedback,t,y02)
36  sol3 = rk4(NegativeFeedback,t,y03)
37  sol4 = rk4(NegativeFeedback,t,y04)
38
39  mRNA1    = sol1[:,0]
40  Protein1 = sol1[:,1]
41  mRNA2    = sol2[:,0]
42  Protein2 = sol2[:,1]
43  mRNA3    = sol3[:,0]
44  Protein3 = sol3[:,1]
45  mRNA4    = sol4[:,0]
46  Protein4 = sol4[:,1]
47
48  P_values = np.linspace(0,400,1000)
```

```
M_null    = (f(P_values))/mu
P_null    = (nu*P_values)/k

fig , (ax1,ax2) = plt.subplots(1,2)
ax1.plot(t,mRNA,label='mRNA')
ax1.plot(t,Protein,label='Protein')
ax1.set(xlabel='Time',ylabel='Concentration of Cells',title='Time
   Course Graph')
ax1.legend()
ax1.grid(True)

ax2.plot(P_values,M_null,color="red",label='mRNA Nullcline')
ax2.plot(P_values,P_null,color="blue",label='Protein Nullcline')
ax2.plot(Protein,mRNA,'lawngreen',label='Trajectory',linestyle='--
   ')
ax2.plot(Protein1,mRNA1,'orange',label='Trajectory1',linestyle='--
   ')
ax2.plot(Protein2,mRNA2,'yellow',label='Trajectory2',linestyle='--
   ')
ax2.plot(Protein3,mRNA3,'pink',label='Trajectory3',linestyle='--')
ax2.plot(Protein3,mRNA3,'magenta',label='Trajectory3',linestyle='
   --')
ax2.plot(Protein4,mRNA4,'cyan',label='Trajectory4',linestyle='--')
ax2.set(xlabel='Protein',ylabel='mRNA',title='Phase Portrait Graph
   ')
ax2.legend()
ax2.grid(True)
ax2.axis([0,25,0,1.2])
plt.show()
```

Listing 3.4: Python Code for Auto Repression model

3.5.5 Time Course Graph

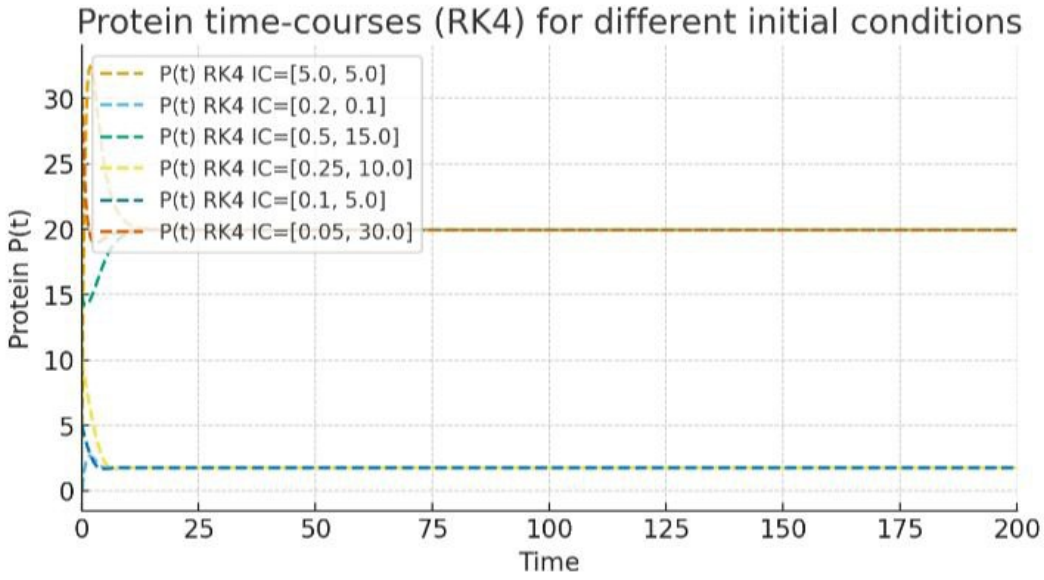


Figure 3.6: Time-course of Auto gene repression

3.5.6 Phase Portrait Graph

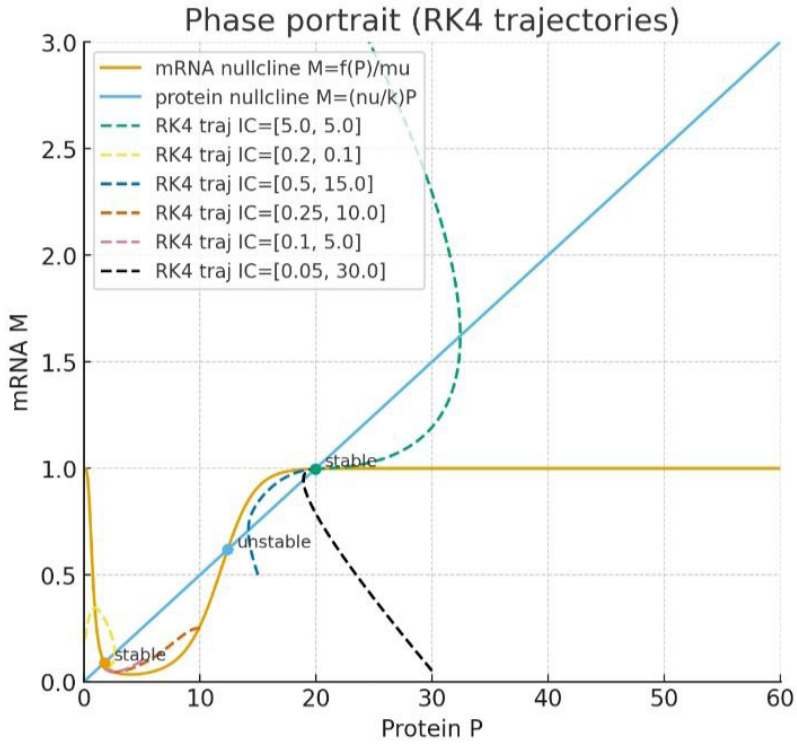


Figure 3.7: Phase Portrait graph showing nullclines and stable point and different trajectories

With parameters

$$K = 5,\ \theta = 1,\ \alpha = 150,\ \mu = 1,\ k = 10,\ \nu = 0.5,$$

We get three equilibria points:

$$\begin{aligned} P_1 &\approx 1.762466, \quad M_1 \approx 0.088107, \quad \text{stable}, \\ P_2 &\approx 12.384780, \quad M_2 \approx 0.619240, \quad \text{unstable}, \\ P_3 &\approx 19.957439, \quad M_3 \approx 0.997872, \quad \text{stable}. \end{aligned}$$

Linear stability computed from the Jacobian confirms the stability classification

3.5.7 Different number of Equilibria Points

By varying α (repression strength) or k (translation gain) we observe a transition between state with one equilibrium to state with three equilibria. This transition occurs via saddle-node bifurcations: as α or k cross critical values, a pair of equilibria (stable + unstable) is created..Some calculations for equilibria are given below in the form of table

(α,k)	num equilibria	example roots (P^*)
(50,5)	1	[1.93]
(50,10)	3	[2.55, 9.86, 19.99]
(150,10)	3	[1.76, 12.38, 19.96]
(500,10)	3	[1.26, 14.93, 19.85]

Thus increasing α (stronger mid-range repression) or increasing k (flatter protein-nullcline) favors bistability.

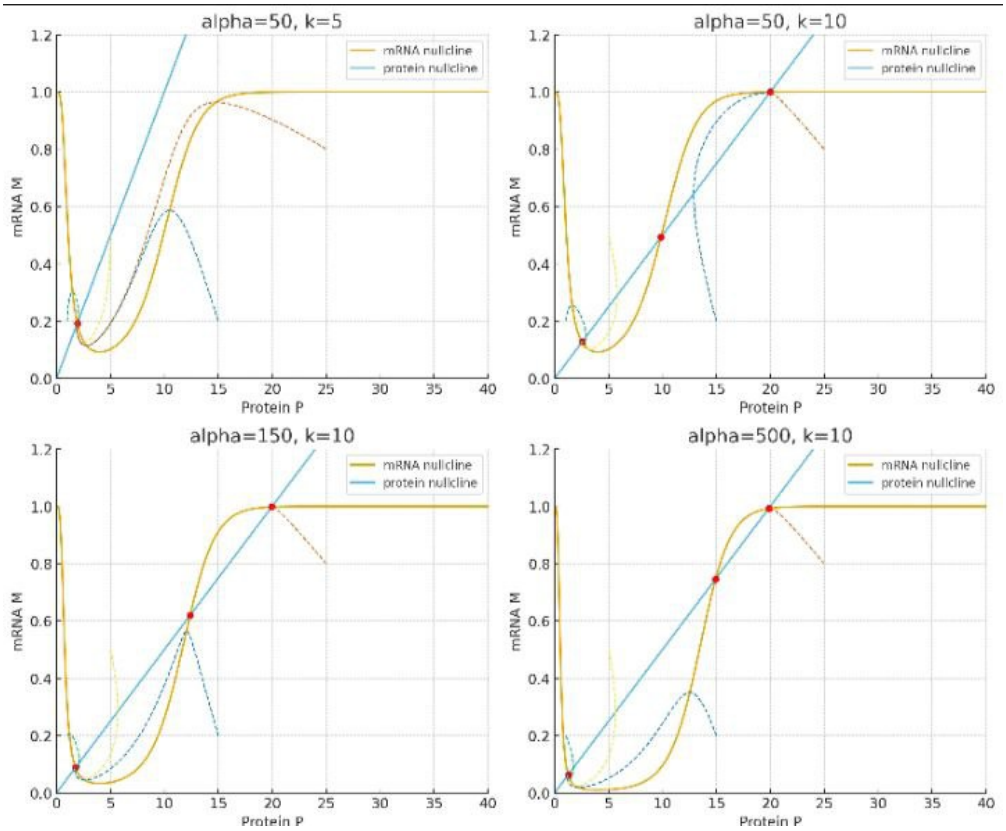


Figure 3.8: Different equlibrium states

3.5.8 Biological mapping and significance

The parameters of gamma function can be mapped to biology they can represent some kind of things in biology which can affect our model they also show biological significance and using gamma function we can represent different kind of biological phenomenon eg. the gene p53 and protein mdm2 represent this kind of relationship.

- K: number of sequential steps required for repressor. It represent the number of steps in which the repressor acts.
- θ: It can represent the time for which a repressor can act by varying theta we can make the repressor effect more broader meaning it can act for more time.
- α: It basically represent the repressor strength.

CHAPTER 4

Pharmacokinetics

4.1 Introduction

Pharmacokinetics, branch of pharmacology, that statistically examines the time course of a drug in the body, described by four major processes: 1. Absorption, 2. Distribution, 3. Metabolism, and 4. Excretion (ADME). Mathematical modeling provides a quantitative framework to describe drug concentration–time profiles, allowing predictions of how drugs behave under different conditions. Understanding these processes is important for optimizing dosage regimens, minimizing toxicity, and improving therapeutic outcomes. Mathematical models simplify the complex interactions between drugs and biological systems, making them easier to analyze. In this report, we focus on compartmental models, particularly the one-compartment and two-compartment models, which form the foundation of pharmacokinetic modeling.

4.2 One Compartment Model

The one-compartment model assumes that, after administration, the drug instantly distributes throughout a single, homogeneous compartment representing the body. Although a simplification, this model is widely used because of its practicality and ease of calculation. This model is particularly useful in:

1. Describing drugs that distribute rapidly and uniformly within the body.

2. Designing simple dosage regimens where high precision is not required.

3. Bioavailability studies: For orally administered drugs that are rapidly absorbed, the one-compartment model helps estimate the fraction absorbed (F) and the absorption rate constant (k).

4. This model can be used to compare different formulations of the same drug (e.g., tablet vs. solution)

5. Clinical applications: Provides estimates of clearance, half-life and volume of distribution that are essential in drug dosing and therapeutic drug monitoring.

To illustrate this we will take an example of intravenous bolus drug that goes directly in blood. Intravenous bolus studies: Describes the plasma drug concentration after an IV bolus injection. Considering C is concentration of drug at time t, k_e is constant rate of elimination:

$$\frac{dC(t)}{dt} = -k_e C(t) \tag{4.1}$$

Upon solving for one dose, we get:

$$C(t) = C_0 e^{-k_e t}$$

where C_0 is initial concentration of dose given. The concentration will decay exponentially as show in figure-1.

Let rate of elimination be 0.231 and amount of drug injected is 6.297ml per unit volume of distribution. To solve this equation numerically, this python code is used:

Listing 4.1: One Compartment Model

```
import numpy as np
import matplotlib.pyplot as plt
from rk4 import RK4_solver
def one_comp(C,t,para):
    ke = para[0]
    dC= -ke*C #drug concentration in bloodstream
    return dC
```

```
t = np.linspace(0, 30,300)
U0 = [6.297]

para = [0.231]
para_error = [0.03]

all_x=[]
for i in range(500):
    params=[]
    for j in range(len(para)):
        p = para[j]
        err = para_error[j]
        random_values = np.random.normal(p,err)
        params.append(random_values)
    sol = RK4_solver(one_comp, U0, t, params)
    all_x.append(sol[:,0])
all_x = np.array(all_x)

x_mean = np.mean(all_x, axis=0)
x_std = np.std(all_x, axis=0)

#plotting
plt.plot(t, x_mean, label="con. in blood")
plt.fill_between(t, x_mean-x_std, x_mean+x_std, color="blue", alpha
   =0.3, label="blood con. +- std")
plt.xlabel("Time")
plt.ylabel("Drug concentration")
plt.legend()
plt.show()
```

The graph illustrates the exponential decay of drug concentration in bloodstream for one compartment model. The dark blue line represents the mean drug concentration, calculated across multiple simulations, while the lighter blue shaded region shows the variability in concentration due to error in elimination rate constant k_e. The half-life of the drug, given by $t_{\frac{1}{2}} = \frac{ln2}{k_e}$, can be inferred from the slope of the mean trajectory

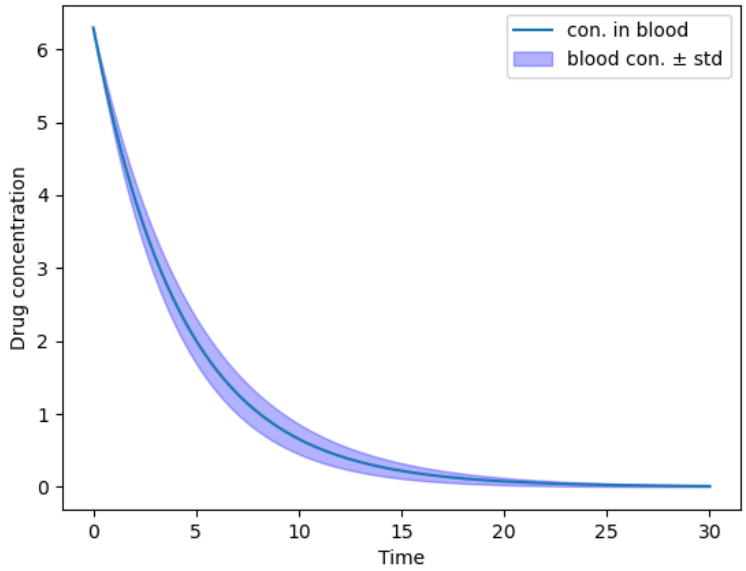


Figure 4.1: Concentration–time curve for one-compartment model.

4.2.1 Repeated dosing

For repeated dosing, suppose a dose is administered at regular intervals of τ. In this simulation, the initial concentration for each interval,C_0, is calculated as the sum of the residual concentration from the previous dose and the newly administered dose ($C_0 = C_{\text{remain}} + \frac{\text{Dose}}{V_d}$). The plasma concentration over each dosing interval is then computed numerically using the RK4 method.

Listing 4.2: repeated dose

```
import numpy as np
import matplotlib.pyplot as plt
from rk4 import RK4


Dose = 100
Vd = 20
Ke = 0.2
tau = 8
num_doses = 5


interval = 100
```

```
# Initialize lists to store the full time and concentration profiles
time_profile = []
conc_profile = []

C_remain = 0 # remain concentration of dose after time tau
for i in range(num_doses):
    start_time = i * tau
    end_time = (i + 1) * tau
    t_interval = np.linspace(start_time, end_time, interval)

    # C0 = remain from last interval + new dose
    C0 = C_remain + Dose / Vd

    def one_comp_iv(t, C):
        dCdt = -Ke * C
        return np.array([dCdt])

    Sol = RK4(one_comp_iv, np.array([C0]), t_interval)

    time_profile.extend(t_interval)
    conc_profile.extend(Sol[:, 0])

    C_remain = Sol[-1, 0]

# --- Plotting ---
plt.figure(figsize=(12, 7))
plt.plot(time_profile, conc_profile, label=f"IV Bolus: {Dose} mg
   every {tau}h", color="blue")
plt.xlabel("Time (hours)")
plt.ylabel("Drug Concentration (mg/L)")
plt.title("One Compartment Model: Repeated IV Bolus Dosing")
plt.legend()
plt.grid(True, which='both', linestyle='--', linewidth=0.5)
plt.show()
```

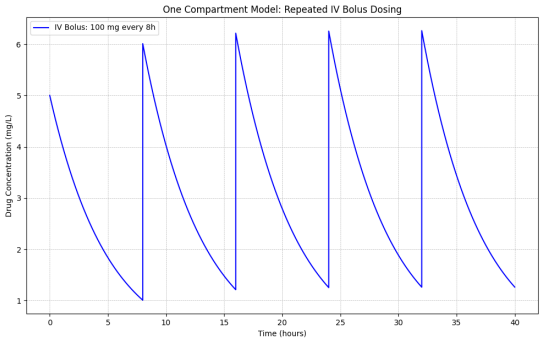


Figure 4.2: For τ = 8h

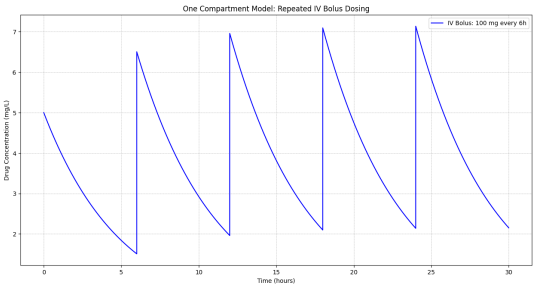


Figure 4.3: For τ = 6h

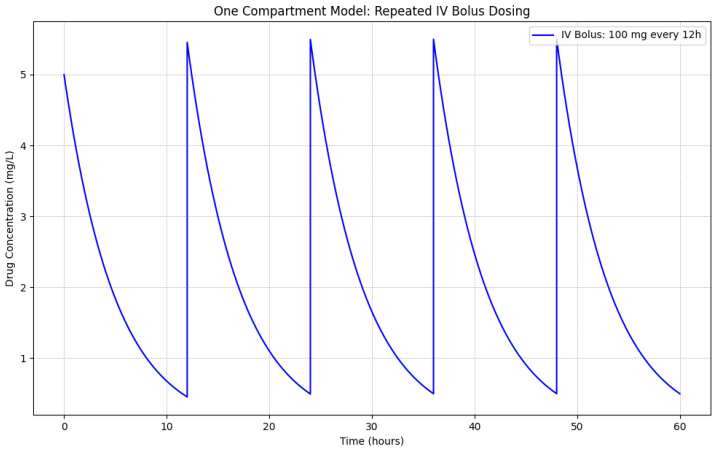


Figure 4.4: For τ=12h

Fig.4.2 and 4.3 show that the drug concentration rises with each new dose. It is not the same amount every time, because some of the previous dose remains in the body and adds to the new one. This build-up, called accumulation, shows why choosing the right time gap between doses is important to keep the drug effective without causing harm. According to Fig. 4.4, for a drug with these parameters, it is best to take it twice in a day i.e., every 12 hours.

4.3 Two Compartment Model

The two-compartment model assumes that, after administration, the drug goes in two, homogeneous compartments (i.e., central and peripheral compartments) representing the body. Central compartment consists of blood and well perfused organs like heart, lungs and peripheral compartment consists of less-perfused tissues, for example muscle tissue, skin, connective

tissue. In case of drug being introduced orally into the body, it is absorbed in gastrointestinal tract and enters in bloodstream (central compartment) then in peripheral. Consider, $A_G(t)$ = amount of drug in gut,

$A_C(t)$ = amount of drug in central compartment

$A_P(t)$ = amount in peripheral compartment.

k_a = rate of absorption (gut to central)

k_{12} = rate of transfer from central to peripheral

k_{21} = peripheral to central transfer rate

k_e = elimination rate from central compartment

F = bioavailability fraction (0-1)

We have considered $F = 1$, that means, the same amount of drug reaches the central compartment as that given in one dose (no first pass reduction on being metabolised in gut). Then:

$$\frac{dA_G}{dt} = -k_a A_G \tag{4.2}$$

$$\frac{dA_C}{dt} = k_a A_G - k_{12} A_C + k_{21} A_P - k_e A_C \tag{4.3}$$

$$\frac{dA_P}{dt} = k_{12} A_C - k_{12} A_P \tag{4.4}$$

Let there be a drug with rate of absorption (k_a) 0.8 h^{-1}, rate of transfer from central to peripheral (k_{12}) 0.05 h^{-1}, peripheral to central transfer rate (k_{21}) 0.02 h^{-1} and elimination rate from central compartment (k_e) 0.01 h^{-1}. To solve this system of equation numerically, the code used is:

```
import numpy as np
import matplotlib.pyplot as plt
from rk4 import RK4_solver

XG0=100 #initial amount of drug in gastrointestinal (in mg)
XC0=0 #initial amount of drug in bloodstream
XP0=0 #initial amount of drug in P
def deri(T,t,para):
```

```
    XG,XC,XP=T
    ka, k12, k21, ke = para
    dG= -ka*XG
    dC= ka*XG - k12*XC + k21*XP - ke*XC
    dP = k12*XC - k12*XP
    return np.array([dG,dC,dP])

t = np.linspace(0, 40,300)
U0 = [100,0,0]

para = [0.8,.05,.02, .04] #giving values to parameters
para_error = [0.01, 0.0003, 0.003, 0.01] #erorr posible in each
   parameter

#monte-carlo
all_x=[]
all_y=[]
for i in range(500):
    params=[]
    for i in range(len(para)):
        p = para[i]
        err = para_error[i]
        random_values = np.random.normal(p,err)
        params.append(random_values)
    sol = RK4_solver(deri, U0, t, params)
    all_x.append(sol[:,0])
    all_y.append(sol[:,1])
all_x = np.array(all_x)
all_y = np.array(all_y)

x_mean = np.mean(all_x, axis=0)
x_std = np.std(all_x, axis=0)
y_mean = np.mean(all_y, axis=0)
y_std  = np.std(all_y, axis=0)
```

```
#plotting
plt.plot(t, x_mean, label="con. in blood", color="blue")
plt.fill_between(t, x_mean-x_std, x_mean+x_std, color="blue", alpha
   =0.3, label="blood con. +/- std")

plt.plot(t, y_mean, label="tissue con.", color="red")
plt.fill_between(t, y_mean-y_std, y_mean+y_std, color="red", alpha
   =0.3, label="tissue con. +/- std")
plt.xlabel("Time")
plt.ylabel("Drug concentration")
plt.legend()
plt.grid()
plt.show()
plt.figure(figsize=(6,6))

for i in range(len(all_x)):
    plt.plot(all_x[i], all_y[i], color='gray', alpha=0.1)
plt.plot(x_mean, y_mean, color='blue', linewidth=2, label='Mean
   Trajectory')
plt.xlabel("tissue Concentration")
plt.ylabel("bloodstream Concentration")
plt.title("Phase Portrait of Drug Concentration")
plt.legend()
plt.grid(True)
plt.show()
```

I have used random values of all the parameters. One can change according to the drug. The graph in figure-5 shows how the drug concentration in central and peripheral compartments changes with time. The solid lines show the mean, whereas the lighte-colored regions represent the possible range of drug concentrations. There are chances of error in parameters while measuring, hence I have included the error part as well. In the code above, error in each parameter is assumed.

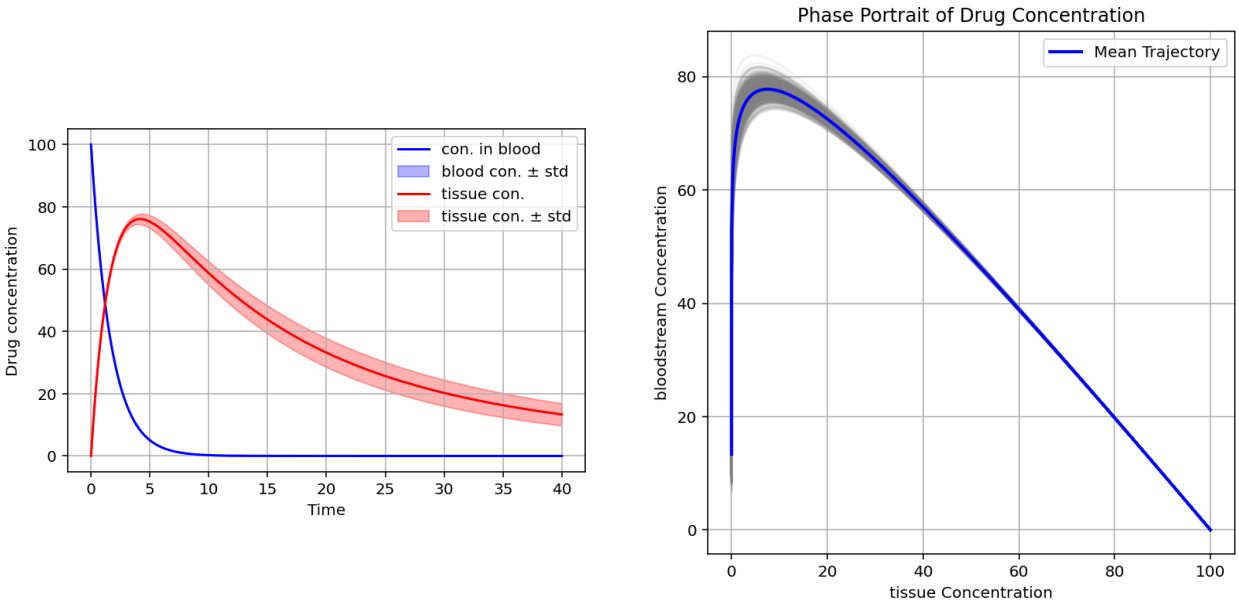


Figure 4.5: For $k_a = 0.8h^{-1}$ and $k_e = 0.04h^{-1}$

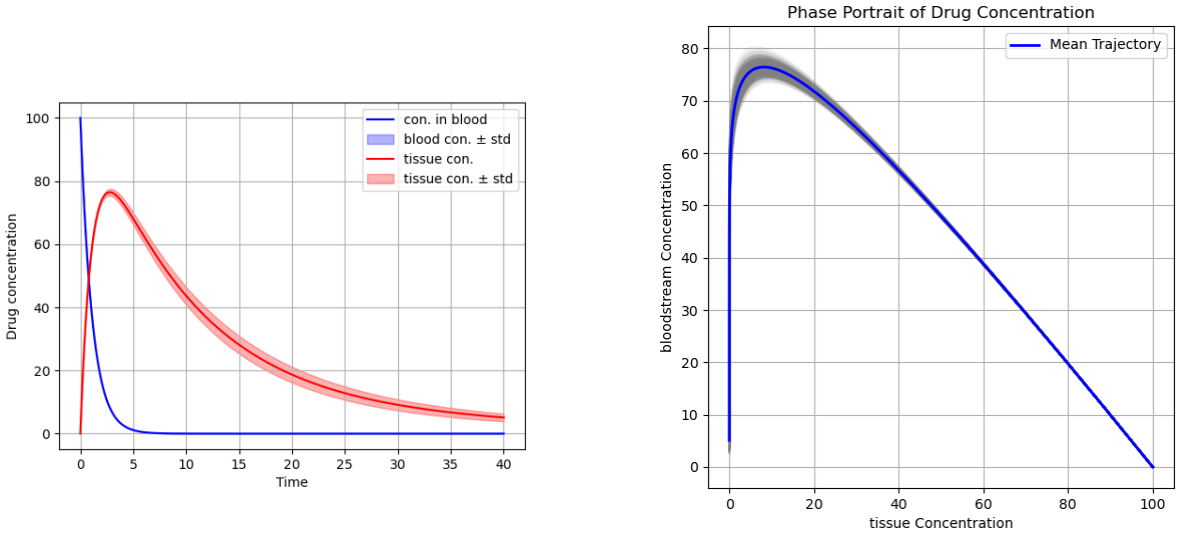


Figure 4.6: For $k_a = 1.2h^{-1}$ and $k_e = 0.08h^{-1}$

4.3.1 Phase Portrait

A phase portrait to visualize the behavior of a system over time in its state space rather than just as a function of time for this system is shown in the right hand side of Fig.?? and ?? is Phase portrait i.e., concentration of drug in bloodstream vs in tissues (peripheral compartment). The blue line represents the average behavior of the drug's concentration profile whereas the fainter grey lines represent the variability in drug concentration profiles.

At the start, the bloodstream concentration rises very quickly, while the drug has not yet had time to distribute into the tissues, so the tissue concentration remains low. The peak of the curve represents the maximum concentration of the drug in the bloodstream. At this point, the process of the drug moving from the bloodstream into the tissues becomes dominant. As the curve moves to the right and downwards from the peak, it shows that the bloodstream concentration is falling as the tissue concentration is rising. After the initial distribution, the final and longest phase is elimination. As the drug is eliminated from the bloodstream, the drug that had previously entered the tissues moves back into the bloodstream to be cleared, causing the concentration in both compartments to fall.

4.4 Conclusion

We studied how drugs move and are eliminated in the body using simple one and two compartment models. The one-compartment model showed that drug levels decrease exponentially over time, while the two-compartment model provided a more realistic picture, showing how drugs move between the blood and other tissues and are eventually eliminated (i.e, bi-exponentially). By using numerical methods, we could predict drug levels and see how small changes in parameters can affect concentration. By including uncertainty in the parameters, we also saw the range of possible drug concentrations. These insights are useful for deciding safe doses, timing medication, and understanding how long a drug stays in the body.

Bibliography

[1] J.D.Murray *Mathematical Biology I & II*,3rd edition, Springer,2002

[2] Alex Best. *Introducing Mathematical Biology*,2023. Available Online: https://sheffield.pressbooks.pub/introducingmathematicalbiology/

[3] S. Yadav and V. Kumar (2013), Study of Prey-Predator system with additional food effective pest control techniques in agriculture, Iranian Journal of Science, Springer, 2023

[4] Bruce Alberts et al. *Molecular Biology of the Cell*,6th edition,Garland Science,2014.

[5] Burden,R.L.,Faires,J.D.(2010). Numerical Analysis(9th ed.).Brooks/Cole,Cengage Learning.

[6] E. Süli and D. F. Mayers, An Introduction to Numerical Analysis, Cambridge University Press, 2003.