



PÁZMÁNY

Pázmány Péter Catholic University

Faculty of Information Technology and Bionics

37. OTDK BIOLOGY SECTION

Analysis and control of stochastic gene regulation network models using compartmental discretization

KECSKÉS, Dorka

Pázmány Péter Catholic University

Faculty of Information Technology and Bionics

Molecular Bionics Engineering BSc

Supervisor: Dr. SZEDERKÉNYI, Gábor

2025

Chapter 1

Introduction

1.1. Context

Gene expression is the process, where the information stored in the DNA is realized to proteins. Complex mechanisms, called gene regulatory networks (GRNs) control which genes are expressed and at what rate [1]. The external control of these gene regulatory networks is an active area of research, various control design concepts have been applied in synthetic biology [2]. Molecular noise is challenging in the design and implementation of these controllers. [3] proposes a control method that ensures the stability of the model even in the presence of noise. In [4] the stabilization of the genetic toggle switch is described. In both cases, a linear feedback control is applied.

In this thesis, I use a model that approximates the time evolution of the protein concentration's probability density function using a partial integro-differential equation (PIDE) [5]. This model describes the production of proteins as bursts of protein concentration, as shown in [6]. In [7] a multidimensional PIDE model was introduced which is able to describe the coregulation of many different genes. A semi-discretization of the multidimensional PIDE model was proposed in [8] introducing a kinetic compartmental description of the system in possibly time-varying linear ODE form.

1.2. Research objectives

The main goals of this research were to investigate the background of stochastic gene regulation network models and to analyze and control a specific model known as the toggle switch. Our objective was to examine its bistable behaviour and explore the conditions under which we can achieve monostable behaviour through appropriate input.

We aimed to create a lookup table that maps the system's inputs to its outputs. Using this our goals were to understand the relation between the inputs and the system's response and to facilitate inversion. Additionally, we intended to apply various control methods and compare them. We aimed to analyze the limits of control by examining the outputs that were reachable within all possible options.

Chapter 2

Background

In this chapter, the most important concepts and notations are introduced that are later used.

2.1. Background in system theory

I feel it would be useful to discuss the basics of systems theory to better understand my thesis.

A system is part of the real world, with a boundary separating it from the environment. The system's effects are its outputs, while the environment's impacts are its inputs. These interactions can be described as scalar— or vector-valued time-dependent functions.

We can define the system operator $\mathcal{S}$, that assigns the inputs to the outputs: $y = \mathcal{S}[u]$

2.1.1 System properties

1. A system $\mathcal{S}$ is called *linear* if

$$\mathcal{S}[c_1 u_1 + c_2 u_2] = c_1 \mathcal{S}[u_1] + c_2 \mathcal{S}[u_2] = c_1 y_1 + c_2 y_2$$

with $c_1, c_2 \in \mathbb{R}$, $u_1, u_2 \in \mathcal{U}$, and $y_1, y_2 \in \mathcal{Y}$.

2. A system $\mathcal{S}$ is *time-invariant* if

$$\mathcal{T}(\tau) \circ \mathcal{S} = \mathcal{S} \circ \mathcal{T}(\tau)$$

where $\mathcal{T}$ is called the time shift operator and

$$\mathcal{T}(\tau)[u(t)] = u(t + \tau).$$

3. In *continuous time systems* the time variable is $t \in \mathcal{T} \subset \mathbb{R}$.
4. In *discrete time systems* the time variable is $t \in \mathcal{T} = \{\dots, t_0, t_1, t_2, \dots\}$.

2.1.2 State-space models

Many dynamical systems can be represented with a state-space model. The representation uses a set of first-order differential (or difference) equations rather than higher-order equations. In this model $u(t)$ is the function describing the inputs, $x(t)$ denotes the state variables, and $y(t)$ describes the outputs. The equations that define a continuous-time, linear time-invariant (CT-LTI) model are the following:

$$\dot{x}(t) = Ax(t) + Bu(t) \quad (\text{state equation})$$

$$y(t) = Cx(t) + Du(t) \quad (\text{output equation})$$

Here A , B , C and D are matrices with constant elements. In the case of a time-variant model, these matrices are the functions of time, while in nonlinear models the equations have the following form:

$$\dot{x}(t) = f(x(t), u(t)) \quad (\text{state equation})$$

$$y(t) = g(x(t), u(t)) \quad (\text{output equation})$$

Observability: The system (A, B, C) is observable, if $x(t_0)$ can be determined from a finite measurement of y .

Controllability: The system (A, B, C) is controllable if, given a finite duration $T > 0$ and two arbitrary points $x_0, x_T \in \mathbb{R}^n$, there exists an appropriate input u such that for initial condition $x(0) = x_0$, the value of the state vector at time T is $x(T) = x_T$.

Stability: A linear system (A, B, C) is internally or asymptotically stable if the solution $x(t)$ of the equation $\dot{x}(t) = Ax(t)$ where $x(t_0) \neq 0$, fulfills $\lim_{t \rightarrow \infty} x(t) = 0$.

2.1.3 Jacobian linearization

Most real-life systems can be represented using nonlinear models. However, we have a much deeper understanding of and more methods for linear models. For this reason, it is useful to approximate nonlinear models with linear ones.

The centre of the linearization has to be an equilibrium point of the system. Considering the linear model with state equation 2.1.2 for an equilibrium point x_{eq} with corresponding equilibrium input u_{eq} with initial condition $x(t_0) = x_{eq}$ and applying constant $u(t) = u_{eq}$ input the system remains fixed, meaning that $x(t) = x_{eq}$ holds. This implies, that $f(x_{eq}, u_{eq}) = 0$.

Define the deviation of x and u as $\delta_x(t) = x(t) - x_{eq}$ and $\delta_u(t) = u(t) - u_{eq}$. Substituting these variables to the nonlinear state equation we get the following:

$$\dot{\delta}_x(t) = f(x_{eq} + \delta_x(t), u_{eq} + \delta_u(t))$$

Let us determine the first-order Taylor expansion of this expression:

$$\dot{\delta}_x(t) \approx f(x_{eq}, u_{eq}) + \left. \frac{\partial f}{\partial x} \right|_{\substack{x=x_{eq} \\ u=u_{eq}}} \delta_x(t) + \left. \frac{\partial f}{\partial u} \right|_{\substack{x=x_{eq} \\ u=u_{eq}}} \delta_u(t)$$

Considering that $f(x_{eq}, u_{eq}) = 0$ we can eliminate the first term:

$$\dot{\delta}_x(t) \approx \left. \frac{\partial f}{\partial x} \right|_{\substack{x=x_{eq} \\ u=u_{eq}}} \delta_x(t) + \left. \frac{\partial f}{\partial u} \right|_{\substack{x=x_{eq} \\ u=u_{eq}}} \delta_u(t)$$

Based on Equation 2.1.3 we can determine the matrices of the linear state equation as

$$A = \left. \frac{\partial f}{\partial x} \right|_{\substack{x=x_{eq} \\ u=u_{eq}}} \quad \text{and} \quad B = \left. \frac{\partial f}{\partial u} \right|_{\substack{x=x_{eq} \\ u=u_{eq}}}$$

These are constant matrices, thus the following equation is linear indeed:

$$\dot{\delta}_x(t) = A\delta_x(t) + B\delta_u(t)$$

Near the equilibrium point, when δ_x and δ_u are small, the linear model approximates the original model with a small error.

2.1.4 The applied control methods

Typically we would like to use the input to reach a desired output of the system, we want to control it. By using feedback, based on the output we compute the required input. Typically, the control is calculated based on the error, which represents the difference between the desired and actual output.

There are numerous methods to control a system; here, I will present the two I employed in broad outline.

PI control In the case of a PI control, we feed back the error with weight K_P (proportional member) and the integral of the error with weight K_I (integral member). Figure 2.1 shows the schematic diagram of the PI control.

pole placement From the state-space model (2.1.2) we can determine the stability of the corresponding system based on the eigenvalues of matrix A . If all the eigenvalues have negative real parts the system is asymptotically stable. However, if any eigenvalue has a positive real part, the system becomes unstable, as the state variables will grow exponentially over time. Moreover, if any eigenvalue has an imaginary part, there are sinusoidal (oscillating) components in the time functions of the states.

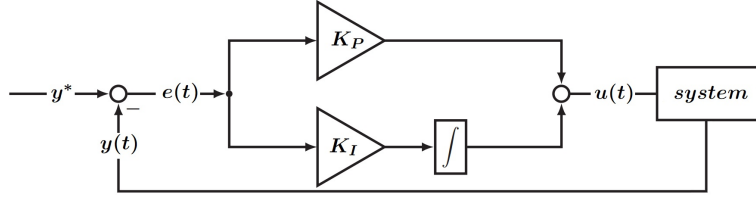


Figure 2.1: Schematic figure of PI control. y^* is the desired output, and $y(t)$ is the actual output. $e(t)$ is the error, the difference between the desired and the actual output. K_P and K_I are the proportional and integral coefficients of the PI control.

Pole placement aims to move the poles to change the system's dynamics (e.g., to stabilize it). Additionally, we can select the value to which the system converges.

To achieve this, instead of feeding back the output, we need to feed back the system's state variables scaled by certain gains and subtract it from a reference, as shown in Figure 2.2.

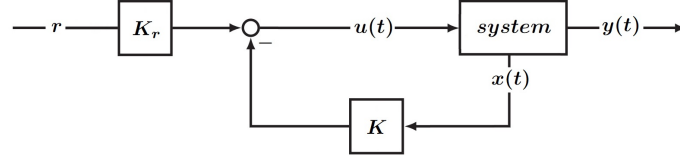


Figure 2.2: Schematic figure of pole placement. r is the reference, $u(t)$ is the input, $x(t)$ contains the state variables, $y(t)$ is the output, K_r is the scaling term for the reference and K is the gain matrix

It should be observed that the input, in this case, is $u = r \cdot K_r - K \cdot x$. The model of the controlled system, containing the regulator parts can be written in the following form:

$$\dot{x} = Ax + Bu$$

$$\dot{x} = Ax + B(rK_r - Kx)$$

$$\dot{x} = Ax - BKx + BrK_r$$

$$\dot{x} = (A - BK)x + BrK_r$$

If we select K such that the eigenvalues of $(A - BK)$ will be negative, then the extended system will be stable.

2.2. Gene regulatory networks

2.2.1 One-dimensional model

Gene expression is determined by stochastically occurring biochemical reactions. Based on the DNA mRNA-s are synthesized through transcription, and the mRNA-s are translated into proteins. This is the

central dogma of molecular biology, shown in Figure 2.3.

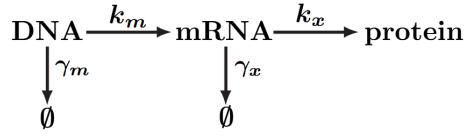


Figure 2.3: The central dogma of molecular biology. k_m and k_x are the reaction rates of transcription and translation, while γ_m and γ_x are the rates of mRNA and protein degradation.

The promoters corresponding to each gene can switch between active and inactive states. The transcription happens almost exclusively if the gene is active, with reaction rate k_m . However because of the transcriptional leakage transcription can occur in an inactive state as well, with reaction rate k_ϵ where $k_\epsilon \ll k_m$. The following equation describes the rate of the transcription considering the different states of the gene:

$$R_T(x) = k_m c(x)$$

Here $x(t)$ is the continuous number of protein molecules in the cell, $c : \mathbb{R}_+ \rightarrow [\epsilon, 1]$ is a Hill function, and k_m is the reaction rate of the transcription when the gene is active. In the case of inducer-based control $c(x) = c(x, I)$ is a function of the amount of inducer(s) as well.

The lifetime of an mRNA is short compared to the lifetime of a protein, so we can consider the process of the translation of an mRNA and the transcription of a few proteins based on this mRNA as a burst, where proteins are synthesized. The mean number of these bursts per cell cycle can be defined as $a = \frac{k_m}{\gamma_m}$ and the mean number of proteins produced in a burst can be defined as $b = \frac{k_x}{\gamma_x}$ where k_m and k_x are the reaction rates of the transcription and the translation while γ_m and γ_x are the rates of mRNA and protein degradation. These two parameters, a and b , characterize the protein production in a cell.

The size of a burst can be modelled with an exponential distribution. The probability of the protein concentration jumping from y to x during a burst is described by the following exponential distribution:

$$\omega(x - y) = \frac{1}{b} \exp \left[-\frac{x - y}{b} \right]$$

Let $p(t, x)$ be the probability density function of the protein concentration in a cell. Then $p(t, x)$ satisfies the following one-dimensional continuous master equation:

$$\frac{\partial p(t, x)}{\partial t} = \frac{\partial}{\partial x} [\gamma_x(x) x p(t, x)] + k_m \int_0^x \beta(x - y) c(y) p(t, y) dy$$

Here $\beta(x - y) = \omega(x - y) - \delta(x - y)$, where $\delta(x - y)$ is the Dirac delta function. [5]

2.2.2 Multidimensional model

In the multidimensional case the gene regulatory network consists of n genes denoted by $G = \{DNA_1, DNA_2, \dots, DNA_n\}$ that express n proteins $X = \{X_1, X_2, \dots, X_n\}$ via the corresponding messenger RNA-s $M = \{mRNA_1, mRNA_2, \dots, mRNA_n\}$. The continuous number of proteins can be represented by an n -dimensional vector $\mathbf{x}$. The transcription rate of X_i depends on its own concentration, this is called self-regulation. Additionally, in the multidimensional case, other proteins can influence (promote or inhibit) the gene's expression, which is called cross-regulation. This way the reaction rate of the transcription of the i -th gene can be written as $R_T^i(\mathbf{x}) = k_m^i c_i(\mathbf{x})$ where $c_i : \mathbb{R}_+^n \rightarrow [\epsilon_i, 1]$ can depend on the concentration of each protein. Similarly, the degradation of protein X_i can depend on any protein thus it can be denoted as $\gamma_x^i(\mathbf{x})$. [9]

For this multidimensional gene regulatory network, we can write the multidimensional master equation in the following partial integro-differential equation (PIDE): [7]

$$\frac{\partial p(t, \mathbf{x})}{\partial t} = \sum_{i=1}^n \frac{\partial}{\partial x_i} [\gamma_x^i(\mathbf{x}) x_i p(t, \mathbf{x})] + \sum_{i=1}^n k_m^i \int_0^{x_i} \beta_i(x_i - y_i) c_i(\mathbf{y}_i) p(t, \mathbf{y}_i) dy_i$$

Here $\mathbf{y}_i = \mathbf{x} + (y_i - x_i)e_i$ and $\beta_i(x) = \omega_i(x) - \delta(x)$.

The existence and mathematical properties of the solution of the above PIDE were shown in [10].

2.3. Compartmental systems

This section is based on [9].

In a compartmental system, the space is partitioned into compartments. These compartments store some kind of real or abstract objects (e.g., mass particles), which can flow from one compartment to another, described by compartmental differential equations. Let q_i be the amount in the i -th compartment in a system with m compartments. The flow of the mass to the i -th compartment from the j -th compartment is denoted by F_{ij} , the mass inflow from the environment to the i -th compartment is denoted by I_i and the mass outflow from the i -th compartment to the environment is denoted by F_{0i} . The change of the mass in the compartments can be given as a set of differential equations:

$$\dot{q}_i = \sum_{j \neq i} (-F_{ji} + F_{ij}) + I_i - F_{0i}$$

Using that $F_{ij} = f_{ij}q_j$ we can rewrite this equation in the following form:

$$\dot{q}_i = -\left(f_{0i} + \sum_{j \neq i} f_{ji}\right)q_i + \sum_{j \neq i} f_{ij}q_j + I_i,$$

By arranging the coefficients in matrix form we get the following equation:

$$\dot{q} = fq + I$$

2.4. Modified kinetic finite volume method

The extended version of the kinetic finite volume method used on the gene regulatory network is detailed in [8].

In the case of the gene regulatory network the compartments are multidimensional intervals of the space of possible concentrations and the "mass" is the probability of the cell concentration being in that interval. The maximum of the concentration of the i -th protein in the cell is L_i . For each $i = 1, 2, \dots, n$ we divide the interval $(0, L_i)$ into N_i equal subintervals. The step sizes of the discretization are $h_i = \frac{L_i}{N_i}$. Let

$$K_\alpha = \times_{i=1}^n K_{\alpha_i}(h_i) = \times_{i=1}^n [(\alpha_i - 1)h_i, \alpha_i h_i],$$

where $\alpha \in \mathbb{N}^n$ is a multi-index and $\bigcup_\alpha K_\alpha = \Omega$, the domain of the discretization. If each compartment has the same size then this size is $h = |K_\alpha| = \prod_{i=1}^n h_i$. We denote the midpoint of the compartment with x_α and the boundaries with $x_\alpha^{i \pm \frac{1}{2}}$.

The probabilities of the compartments can be approximated with the following equation:

$$p_\alpha(t) \approx \frac{1}{h} \int_{K_\alpha} p(t, \mathbf{y}) d\mathbf{y}$$

Similarly, the degradation in a compartment can be described as:

$$\gamma_\alpha^i = \frac{1}{h} \int_{K_\alpha} \gamma_x^i(\mathbf{y}) d\mathbf{y},$$

In Figure 2.4 we can see an illustration of the compartmentalization in a two-dimensional case, like in the case of the toggle switch.

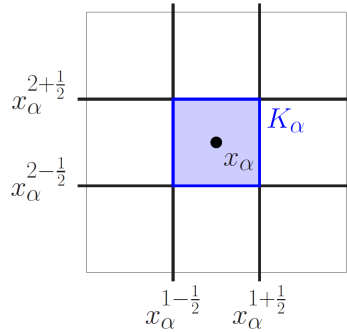


Figure 2.4: Compartments in case of 2-dimensional discretization

In the case of the function $c(x, I)$ instead of the cell average for $c_\alpha^i(I)$ we use the midpoint: $c_\alpha^i(I) = c_i(x_\alpha, I)$. The coefficients can be computed as:

$$b_{\alpha, \alpha_i}^i = \frac{1}{h_i/2} \int_{[(i-1)h_i, (i-1/2)h_i]} \beta_i(x_\alpha^i - y) dy = \frac{1}{h_i/2} \int_{[x_\alpha^i - h_i/2, x_\alpha^i]} \beta_i(x_\alpha^i - y) dy,$$

$$b_{\alpha, j}^i = \frac{1}{h_i} \int_{K_j(h_i)} \beta_i(x_\alpha^i - y) dy, \quad j = 1, 2, \dots, \alpha_i - 1.$$

Instead of the derivative terms difference quotients are used for approximating the degradation:

$$\left. \frac{\partial}{\partial x_i} [\gamma_x^i(\mathbf{x}) x_i p(t, \mathbf{x})] \right|_{K_\alpha} \approx \frac{1}{h_i} (\gamma_{\alpha+e_i}^i x_\alpha^{i+\frac{1}{2}} p_{\alpha+e_i}(t) - \gamma_\alpha^i x_\alpha^{i-\frac{1}{2}} p_\alpha(t))$$

This way we can write the discretized multidimensional master equation in the following form:

$$\dot{p}_\alpha(t) = \sum_{i=1}^n \frac{1}{h_i} (\gamma_{\alpha+e_i}^i x_\alpha^{i+\frac{1}{2}} p_{\alpha+e_i}(t) - \gamma_\alpha^i x_\alpha^{i-\frac{1}{2}} p_\alpha(t)) + \sum_{i=1}^n k_m^i \sum_{j=1}^{\alpha_i} h_{\alpha, j}^i b_{\alpha, j}^i c_{\alpha, j}^i(I) p_{\alpha, j}(t)$$

If we line up the state variables of the compartments to a vector, and write the coefficients into a matrix we can write the equations in the following form:

$$\dot{p}(t) = \Gamma(I(t))p(t)$$

which is a time-dependent (nonautonomous) compartmental model having a strongly connected directed graph.

2.5. Master equations

Chemical master equations are used to model the stochastic dynamics of (bio)chemical reaction networks. It can be shown that the time evolution of the probability density functions of molecules taking part in the reactions can be described in the following ODE form

$$\dot{p}(t) = Ap(t), \quad p(0) \succeq 0 \tag{2.1}$$

where $p(t) \in \mathbb{R}^n$ represents the probability density function of the molecules in each time instant t , and the matrix $A \in \mathbb{R}^{n \times n}$ contains the transition probabilities between the states which can be computed from the reaction rate coefficients. The matrix A has special properties which have important consequences on the qualitative dynamics of (2.1):

1. A is a so-called Metzler matrix (i.e., its off-diagonal elements are nonnegative). This ensures the nonnegative invariance of the dynamics (2.1)

2. A is a special compartmental matrix with a zero column-sum property, i.e. $\sum_{i=1}^n A_{i,j} = 0$. A matrix with properties 1 and 2 is also called a $\mathbb{W}$ -matrix in the literature. It is easy to see from this property that if $\sum_{i=1}^n p_i(0) = 1$ (i.e., if $p(0)$ is a probability distribution) then $\sum_{i=1}^n p_i(t) = 1 \forall t > 0$.
3. It follows from the compartmental property of A that it can have zero eigenvalues, and eigenvalues with strictly negative real parts (purely imaginary eigenvalues are not possible). Moreover, the number of zero eigenvalues can be computed from the influence graph of A which only depends on the structure (i.e., the zero and non-zero elements) of A , but not on its particular parameter values.
4. Using also the properties of eigenvectors corresponding to the zero eigenvalues, we can conclude that the dynamics (2.1) is stable (but not asymptotically stable) in the entire state space.
5. Assuming $\sum_{i=1}^n p_i(0) = 1$, it can be shown that if 0 is an algebraically simple eigenvalue of A then there exists a unique nonnegative equilibrium p^* which is globally asymptotically stable if we restrict the dynamics to the linear manifold $\sum_{i=1}^n p_i(t) = 1$. We note that the mentioned eigenvalue condition is equivalent to the property that the so-called influence graph of A contains exactly one trap (i.e., a strong component with only incoming edges from other nodes but no outgoing edge to other nodes). Clearly, this condition is fulfilled if the whole influence graph of A is strongly connected.

A model of the form (2.1) with the above properties is called an *autonomous master equation*. The dynamics of autonomous master equations is very-well known as well as further interesting and useful properties of the coefficient matrix A . More details on stochastic chemical reactions can be found in the books [11]–[13], while [14], [15] are fundamental references for compartmental matrices and compartmental dynamics.

If we allow time-dependent reaction rate coefficients (e.g., the temperature is changing), then the master equation (2.1) is changed to the following linear time-varying model

$$\dot{p}(t) = A(t)p(t), \quad p(0) \succeq 0 \quad (2.2)$$

where the coefficient matrix $A(t) \in \mathbb{R}^{n \times n}$ is now a function of time. Therefore, (2.2) is called a *nonautonomous master equation*. In our case, the time dependence of A is implied by varying the inducer inputs in the gene regulation networks model. In any fixed time instant t , properties 1-3 are fulfilled for $A(t)$. However, as it can be expected, we know much less about the properties of the dynamics of nonautonomous master equations than on autonomous ones.

Since there might be no equilibria in the time-varying case, we can speak about the stability of solutions. The probability distribution solutions of (2.2) are called *globally asymptotically stable* if for

any two probability distribution solutions $p^{(1)}$ and $p^{(2)}$ we have that $p^{(1)}(t) - p^{(2)}(t) \rightarrow 0$ when $t \rightarrow \infty$.

We know from [16], [17] the following:

Theorem 1 *If $A : [0, \infty) \rightarrow \mathbb{R}^{n \times n}$ is a uniformly continuous and bounded matrix function which is converging to a constant $\mathbb{W}$ -matrix Q which has 0 as an algebraically simple eigenvalue, then any probability distribution solution $x(t)$ of (2.2) will converge to the normed nonnegative eigenvector p (i.e., $p_i \geq 0$, $\sum_{i=1}^n p_i = 1$) of Q corresponding to the 0 eigenvalue.*

The strongest result on the stability of the solutions has been quite recently published in [18], and can be slightly re-stated as follows:

Theorem 2 *Suppose that $A : [0, \infty) \rightarrow \mathbb{R}^{n \times n}$ is a $\mathbb{W}$ -matrix-valued function which is uniformly continuous and bounded on $[0, \infty)$. If the omega-limit set of A contains at least one matrix which has 0 as an algebraically simple eigenvalue, then the probability distribution solutions of the nonautonomous master equation (2.2) are globally asymptotically stable.*

The above theorem will allow us to apply non-slowly-varying (e.g., periodic) inducer inputs to reach preferred distributions for gene regulation networks from any initial distribution.

Chapter 3

Model Analysis

In this chapter, we describe the mathematical model of the studied system and perform a brief analysis of reachability and stability.

3.1. Toggle switch

The chosen model is the genetic toggle switch, in which two proteins repress each other's gene expression as shown in Figure 3.1. We aim to find the needed inducer amounts to reach steady-state probability density with prescribed properties (e.g., expected values).

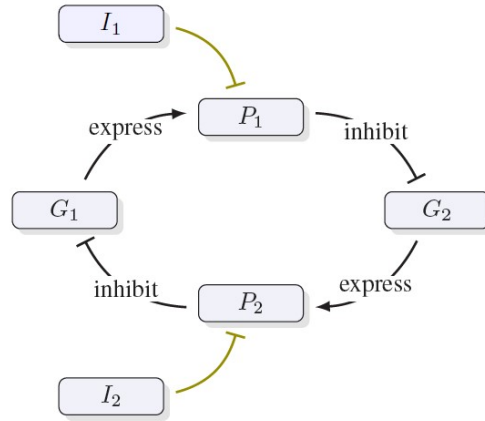


Figure 3.1: Schematic figure of the toggle switch.

Based on [4] the burst coefficients of the system are the following Hill-type functions:

$$c_1(\mathbf{x}, \mathbf{I}) = c_1(x_2, I_1) = \frac{K_{12}(I_1)^H + \epsilon_1 x_2^H}{K_{12}(I_1)^H + x_2^H}$$
$$c_2(\mathbf{x}, \mathbf{I}) = c_2(x_1, I_2) = \frac{K_{21}(I_2)^H + \epsilon_2 x_1^H}{K_{21}(I_2)^H + x_1^H}$$

with

$$K_{12}(I_1) = \theta_{X_2} \left(1 + \left(\frac{I_1}{\theta_{I_1}} \right)^{\mu_{I_1}} \right)$$

$$K_{21}(I_2) = \theta_{X_1} \left(1 + \left(\frac{I_2}{\theta_{I_2}} \right)^{\mu_{I_2}} \right)$$

The positive model parameters of the genetic toggle switch can be seen in Table 3.1.

	γ_x	k_m	b	ϵ	θ_X	θ_I	μ_I
first protein	1	12	6	0.1	31.94	11.65	2
second protein	1	7	$\frac{78}{7}$	0.1	30	9.06×10^{-2}	2

Table 3.1: PIDE parameters of the gene toggle switch.

3.2. Model stability

It was shown in [8] that the coefficient matrix Γ of the system is irreducible independently of the resolution of the discretization. In other words, the influence graph of Γ is strongly connected. This implies that 0 is an algebraically simple eigenvalue of Γ for any constant inducer value. Therefore, we can apply Theorem 1 to conclude that the solutions will converge to a unique stationary distribution function from any initial probability distribution. Moreover, if the system is controlled such that the inputs converge to constant values, then the solution will also converge to a unique solution. In an engineering sense, this means that the properties of the stationary distribution are known and cannot be changed if we know the inducer values to which the inputs converge. Only the dynamics of convergence and the cost of the input (e.g., amount of material needed) can be modified by changing the time functions of the inducers.

3.3. Matlab implementation

The original script I used was written by Mihály A. Vághy. Based on the kinetic semi-discretisation of the master equation described in 2.4 Mihály A. Vághy made a simulation of the toggle switch using MATLAB programming language.

In this simulation, the properties of the genes can be set, like the rate of transcription, translation rate, cross-regulation coefficients, etc. By specifying the net size the coarseness of the discretisation can be changed. The amount of the first and the second inducer added to the system can also be determined. In the first version of the script, this amount is constant. Finally, we can set the precision by changing the time step of the simulation.

The script computes the system's stationary probability density function based on the specified parameters of the simulation. Given that, we can easily compute other properties of the stationary distribution, like the expected values and the deviations of the protein concentrations.

The second version of this script uses PI control to compute the required amount of the inducers to reach a desired output, a desired pair of expected values of the proteins. In this case, we have two inputs and two outputs, requiring a dual PI control: one controller to compute the amount of the first inducer and another for the second inducer. Figure 3.2 shows the schematic diagram of the PI control applied in the script.

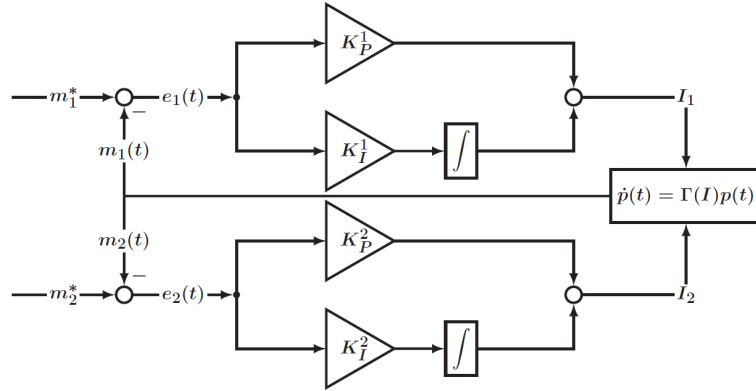
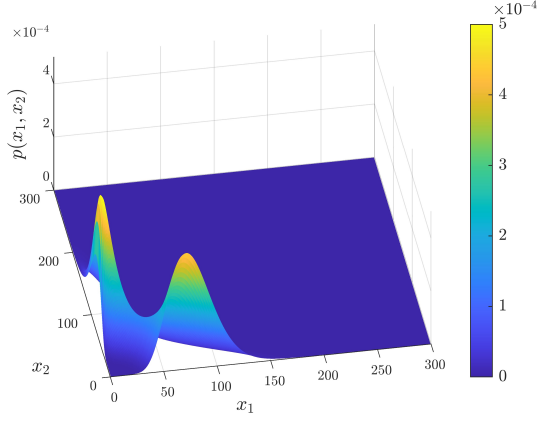


Figure 3.2: Schematic figure of PI control. m_1^* and m_2^* are the desired expected values of the first and the second proteins' concentrations, while $m_1(t)$ and $m_2(t)$ are the actual expected values. $e_1(t) = m_1^* - m_1(t)$ and $e_2(t) = m_2^* - m_2(t)$ are the errors. K_P^1 , K_P^2 , K_I^1 and K_I^2 are the proportional and integral coefficients of the PI control in the first and the second protein.

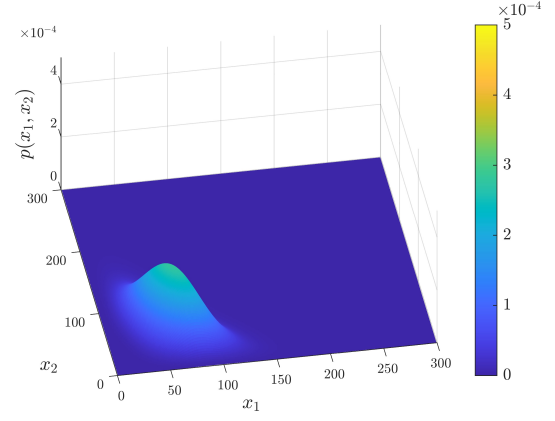
3.4. Reachability of the model

The model described by equation 2.4 is a conservative linear compartmental and kinetic system, with a strongly connected compartmental graph structure, thus there exists a unique asymptotically stable positive equilibrium for constant I . Based on this property, we can compute these steady-state distributions of the toggle switch. In Figure 3.3 on the left we see the steady-state distribution with no control and on the right with constant inducer added to the system. The distribution without control is bimodal, but we can reach unimodal steady-state distribution with control. In the case of a bimodal distribution the expected values of the concentrations will be between the two peaks, thus the actual concentrations in the cell will probably be far from the expected values.

The control has limitations, the expected values of the steady-state distributions will always remain in a reachable set, as seen in Figure 3.4. The simulation script can determine the results quickly, making it possible to compute this set by simulating the expected values in case of different amounts of inducers.



Stationary probability density function
of the uncontrolled system



Stationary probability density function
of the controlled system

Figure 3.3: Steady-state probability density functions of the protein concentrations

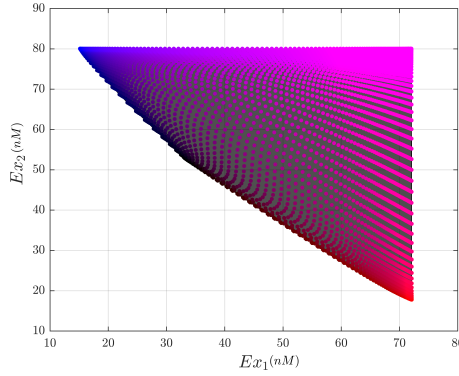


Figure 3.4: Empirical reachability set of expected values. The colour of the dots represents the amount of the inducer needed to reach that expected value. Red stands for the amount of the first, while blue for the amount of the second inducer.

3.5. Model inversion using a lookup table

It is clear from the previous section that it is easy to compute the stationary molecular distribution and its mean if the values of the inducer inputs are given. In this section, we address the inverse problem, i.e., how to compute the constant inducers if we want to reach a given mean value of the stationary distribution. For this, a simple lookup table and derivative-free optimization will be used.

3.5.1 Creating the lookup table

When we add constant inducers to the system, the stationary probability density function is unique, meaning that for every constant inducer pair, the expected values of the protein concentrations are also unique. I wrote a script that iterates through a range of possible inducer values and saves the expected values of the protein concentration (output) for the given inducer amounts (input) in the form of a matrix,

creating a lookup table.

After writing the code, I experimented with determining the appropriate step size and the range for varying the amount of inducers.

The amounts of the inducers have physical limits. However, it can be observed that their effectiveness plateaus well before reaching these limits. Based on the model even if we could add more of the inducers it would not alter the system's behaviour. The limit of the step size was the program's run time. In the last case, I used only 80% and 40% of the maximum amount for the two inducers and divided these intervals into 40-40 parts, so I computed the output for $41 \cdot 41 = 1681$ different inputs.

The expected values of the lookup table can be seen in Figure 3.5. This Figure shows the deviation of the steady-state distribution of the first and the second protein across the reachability set as well.

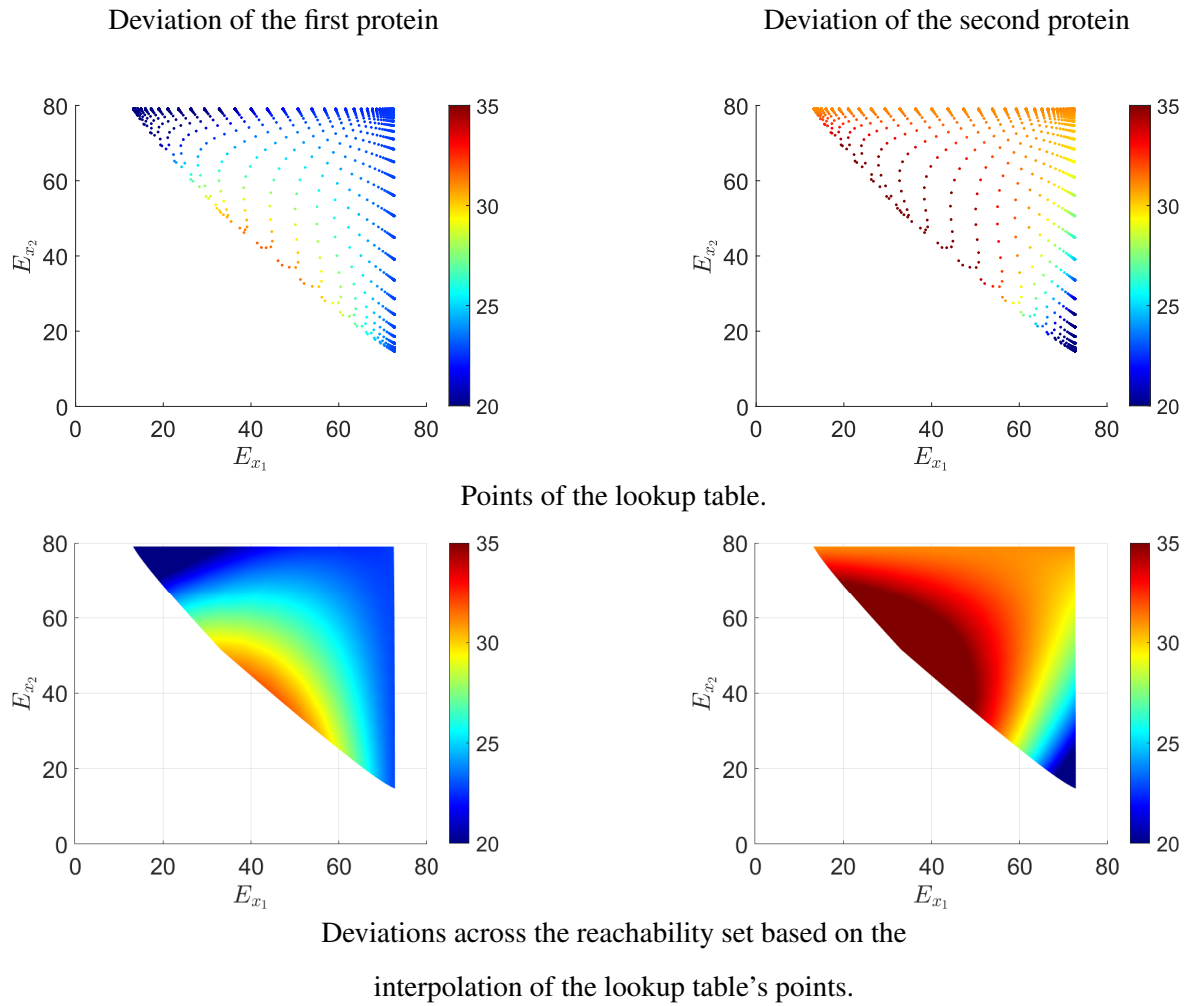
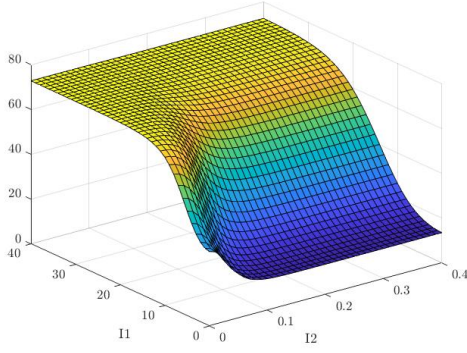
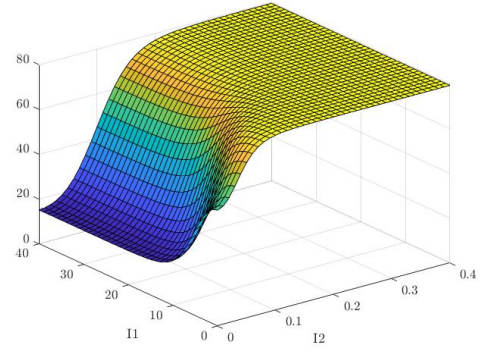


Figure 3.5: Steady state expected values of the lookup table based on iterating through different constant inducer values. The colours on the left represent the deviation of the first protein and on the right the second protein's deviation in nM . On the top the the points of the lookup table are represented, while on the bottom the values of the deviation are calculated by interpolation between the points of the lookup table.



Expected value of the concentration of the first protein



Expected value of the concentration of the second protein

Figure 3.6: Expected values of the steady-state protein concentrations in nM as a function of the amount of the first and the second inducer (I_1, I_2)

I also plotted these matrices to visualise the effect of the inducers on the expected values of the first and second protein concentrations. The result can be seen in Figure 3.6. The figure shows that the first inducer primarily affects the expression of the first gene thus the amount of the first protein, whereas the second inducer mainly influences the second protein. There is a small section where both inducers affect both proteins, corresponding to the area where we have a few of each.

3.5.2 Inversion

Based on this lookup table, we can not only determine the expected values as a function of the inducer but also find the needed amounts of inducer for the desired expected values.

I used an inverter script and modified it based on my parameters. This code utilises the `patternsearch` MATLAB function. The goal of this optimisation-based method is to minimise the loss function, which is the difference between the desired expected values and the actual expected values in our case. To evaluate this loss function the code uses `interp2` to interpolate the lookup table.

This method quickly determines the amounts of the inducers needed to achieve the desired output. It is significantly faster than the simulation with the PI control, as it does not simulate the system but instead utilizes pre-calculated data. However, it is less precise, especially in regions, where the the function's derivative changes rapidly, because the interpolation does not calculate the exact values of the function.

Results I analyzed the results for accuracy, and some of the values are presented in Table 3.2. In the cases shown, we can see that the algorithm finds a solution, that is expected to produce almost exactly the desired output, however, the calculated inducer levels result in slightly different concentrations. This occurs because the interpolation assumes a linear change between the grid points of the lookup table.

desired $E_{x_1}, E_{x_2} (nM)$	computed $I_1, I_2 (nM)$	predicted $E_{x_1}, E_{x_2} (nM)$	actual $E_{x_1}, E_{x_2} (nM)$	PI control $E_{x_1}, E_{x_2} (nM)$
41	9.0592	41.0010	40.7518	41.0186
55	0.0666	55.0009	55.3449	55.0092
70	27.2554	70.0000	69.9995	70.0072
75	0.1785	75.0000	75.0612	75.0024

Table 3.2: E_{x_1} and E_{x_2} denotes the expected values of the protein concentrations' steady-state probability density functions. The first column contains the desired expected values of the concentrations, while the second column lists the inducer amounts computed by the inversion. The third column shows the expected concentrations corresponding to the calculated inducers, as determined by interpolation from the lookup table. In the fourth column the actual E_{x_1} and E_{x_2} represent the expected values of the concentrations obtained through simulation with the same inducer amounts. The last column provides the expected values determined by PI control.

Chapter 4

Control

Our goal was to reach the desired expected values of the system, and we tried a few different ways to achieve it. The PI control of the system depicted in Figure 3.2 was already implemented when I joined the research project. This chapter outlines the two other control methods I employed.

A brief note is that during the control, we allowed only nonnegative inducer inputs with given upper bounds.

4.1. Pole placement

Applying pole placement Regarding our model, we did not apply the pole placement directly to the system, since it is nonlinear, and pole placement is suitable for linear systems. Instead, we used the method below to determine the needed amount of inducers.

The differential equations describing the needed amount of the inducers in the case of the PI control are the following:

$$\begin{aligned} I_1(t) &= K_P^1 e_1(t) + K_I^1 \int_0^t e_1(\tau) d\tau \\ I_2(t) &= K_P^2 e_2(t) + K_I^2 \int_0^t e_2(\tau) d\tau \end{aligned}$$

If we take the time derivative of the equations and express the error from them we get the following:

$$\begin{aligned} \dot{I}_1(t) &= K_P^1 \dot{e}_1(t) + K_I^1 e_1(t) \\ \dot{I}_2(t) &= K_P^2 \dot{e}_2(t) + K_I^2 e_2(t) \\ \dot{e}_1(t) &= -\frac{K_I^1}{K_P^1} e_1(t) + \frac{1}{K_P^1} \dot{I}_1(t) \\ \dot{e}_2(t) &= -\frac{K_I^2}{K_P^2} e_2(t) + \frac{1}{K_P^2} \dot{I}_2(t) \end{aligned} \tag{4.1}$$

From the equations (4.1) we can form a state-space model where the state variables are the errors of the expected values, and the inputs are the derivatives of the inducers. We can compute the needed inducers from the errors based on Equation 4.1. I wrote a script that applies pole placement to this state-space system to compute the needed amount of inducers.

The method was successful as the system achieved the expected values, finding the same steady-state amount of inducers as the PI. Unfortunately, however, this method created oscillating behaviour in the system. The possible reason for this is the nonlinear dynamics of the original system model. The amount of the inducers and the error function can be seen in Figure 4.1 in the case of the original PI control and the case of the one modified with pole placement.

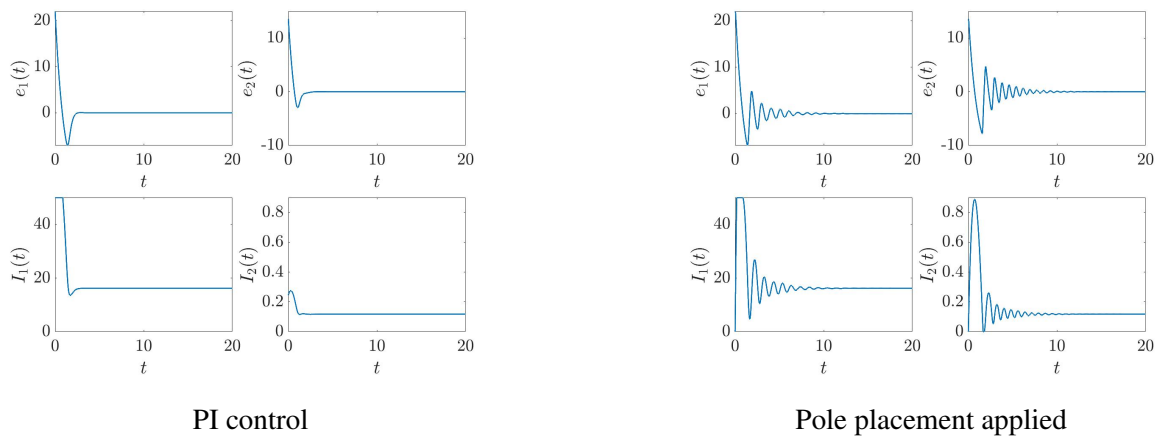


Figure 4.1: Error and the amount of inducers needed for the control. For both the PI control and the pole placement the subfigures show the error of the first and the second protein on the top and the amount of the first and the second inducer on the bottom. In both cases, the control calculates the required constant amounts of the inducers, but with pole placement, an oscillating behaviour appears. $m_1^* = 55 \text{ nM}$, $m_2^* = 65 \text{ nM}$

4.2. Linearization

For the PI control, we used the values of the constants K_P and K_I from previously published literature. The next question was whether we could reach better performance with constants determined by control design. For this, we needed a linearized model.

Typically, I used the model with a mesh size 150×150 . In this case, the state space model has $150 \cdot 150 = 22500$ state variables, so the size of the A matrix of the model is 22500×22500 . Clearly, it is impossible to determine each value of this matrix by hand, but we were hoping that a linearization with a much smaller mesh size would still provide a solution sufficient for our needs.

4.2.1 Nonlinear model implementation

As a start, we chose the 2×2 mesh size, resulting in 4 state variables and a Γ matrix with size 4×4 . The four state variables (p_1, p_2, p_3, p_4) correspond to the four quarters of the concentration space, as shown in Figure 4.2. For example, p_1 represents the probability that for the first protein's and the second protein's concentration $0 \leq x_1 \leq 150$ and $0 \leq x_2 \leq 150$ holds.

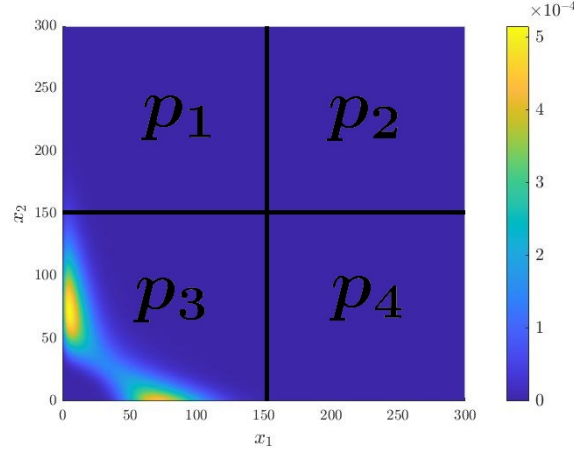


Figure 4.2: Illustration of variables p_1, p_2, p_3 , and p_4 . The axes mark the concentrations of the first and the second proteins. p_1, p_2, p_3 , and p_4 represent the probability of the cell's concentration being inside the corresponding square.

First I had to implement the model, by specifying every element of the corresponding matrices. The system had the following form: $\dot{\mathbf{p}}(t) = \mathbf{f}(\mathbf{p}(t), \mathbf{I}(t)) = \mathbf{\Gamma}(\mathbf{I})\mathbf{p}(t)$. I had access to the description of the discretisation where the elements of the matrices were defined, thus I had to collect the needed parameters and functions in the script.

I simulated the system's time evolution with an ode45 MATLAB function. I could verify the results by comparing them with the original simulation.

4.2.2 Linearization of the model

I linearized the implemented model with the method described in Section 2.1.3. First, I searched for an equilibrium point of the system using the simulation. I saved the equilibrium state vector ($\mathbf{p}_{eq}$) with the corresponding equilibrium input ($\mathbf{I}_{eq}$). Then I used MATLAB's symbolic math toolbox to compute the partial derivative of the $\mathbf{\Gamma}(t)$ matrix. Finally, I substituted the $\mathbf{p}_{eq}$ and $\mathbf{I}_{eq}$ values of the equilibrium to the partial derivatives.

I simulated both the nonlinear and the linearized model and compared the results, as shown in Figure 4.3. The linearized and nonlinear models align almost perfectly, despite the simulation being far from the equilibrium point.

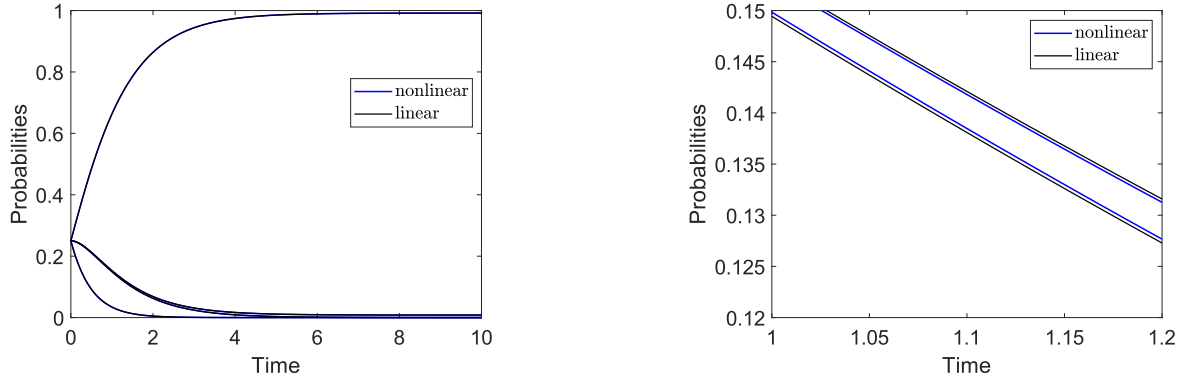


Figure 4.3: Simulation of the linearized (black) and the original nonlinear (blue) model far from the equilibrium point. The two curves overlap so precisely that they cannot be distinguished in the left image. In the right picture, an enlarged section can be seen.

4.3. Linearization based control

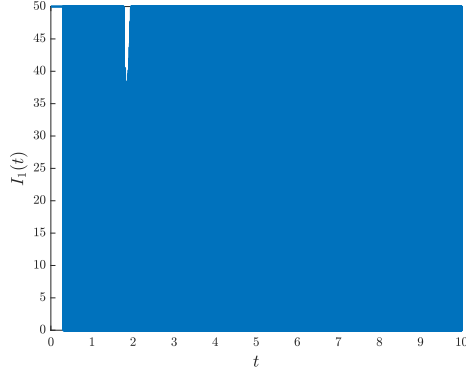
Using a linearized model, I could apply PI tuning for regulatory design. I used the `pdtune` MATLAB function with type PI to determine the optimal K_P and K_I values for the first and the second circuits in the PI controller. These constants can be seen in Table 4.1. I then simulated the system using the PI control with the new constants.

K_P for the first inducer	1.8323e+04
K_I for the first inducer	5.5093e+04
K_P for the second inducer	2.0157e+03
K_I for the second inducer	6.1059e+03

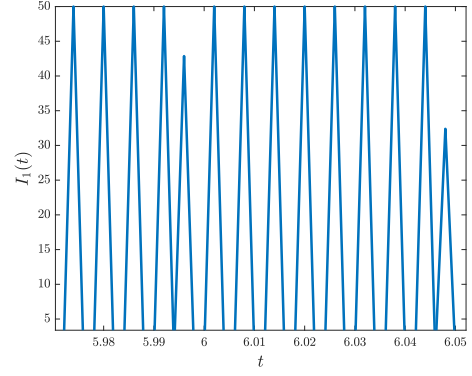
Table 4.1: Constants for PI control calculated by regulatory design for the linearized model.

I checked the results and found something completely different than we expected. The control reached the desired expected values but computed an input with rapid alternations. The values of the first inducer calculated by the new constants can be seen in Figure 4.4

With the regulatory design, we expected to find parameters for the PI control which can determine the required constant amount of inducers as fast as possible. In contrast, the simulation could not determine the required constant amounts whatsoever, instead, it calculated an oscillating input, which is hard to implement in real life. Nevertheless, the error converged to zero, and to our astonishment, we could see at a glance that the steady-state probability density function had a much smaller deviation. The distribution is determined using constant inducers, and we cannot change the deviation. However, this altering input could even allow me to reach an unimodal distribution instead of the original bimodal. The



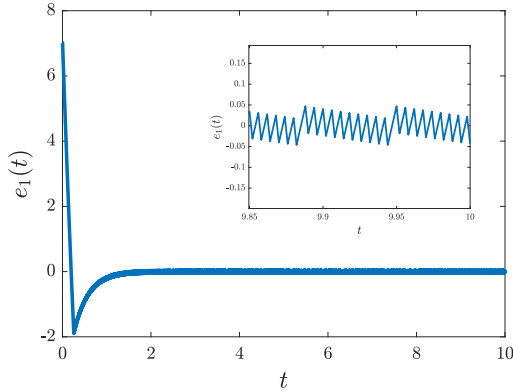
Amount of first inducer



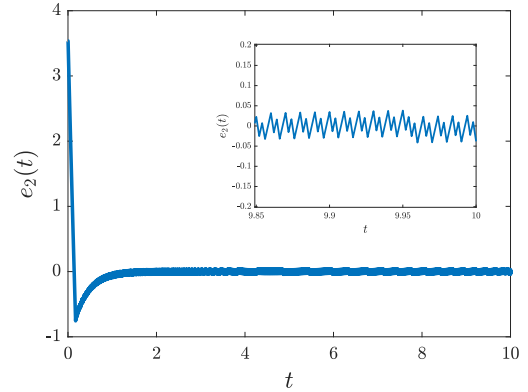
Amount of first inducer, enlarged section

Figure 4.4: Amount of the first inducer computed by the PI control using the constants from Table 4.1. The alterations are so rapid that in the left image the lines cannot be distinguished. The image on the right is an enlarged view to show the alterations. The second inducer's values were similar to this. The desired expected values were $m_1^* = 41 \text{ nM}$ and $m_2^* = 55 \text{ nM}$.

steady-state probability density functions can be seen in Figure 4.6 and the error functions are shown in Figure 4.5.



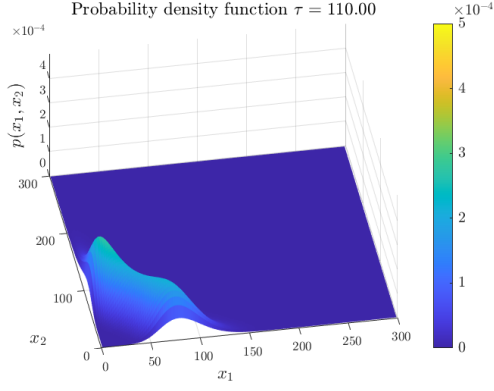
Error of the expected value of the first protein



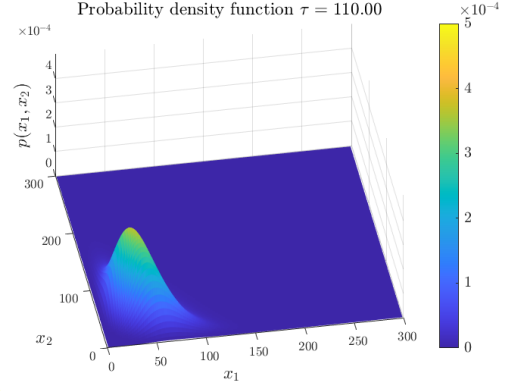
Error of the expected value of the first protein

Figure 4.5: Errors of the expected values of the protein concentrations. The errors converge to zero, except for a small fluctuation. In both cases, a small enlarged section shows the fluctuations.

When the deviation of the probability density function is large, the system's behaviour becomes uncertain. The actual response can be far from the expected values, especially in the case of a bimodal distribution. Regarding this, reaching much smaller deviations is a noteworthy discovery. It was probable, that the reason behind the smaller deviations was the periodicity of the input. The steady-state distribution is determined for constant inducers, but periodic input can change the system's steady-state distribution. This discovery opened up a new area for exploration.

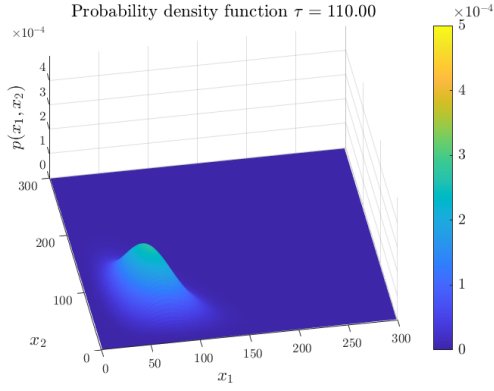


Constant inducer

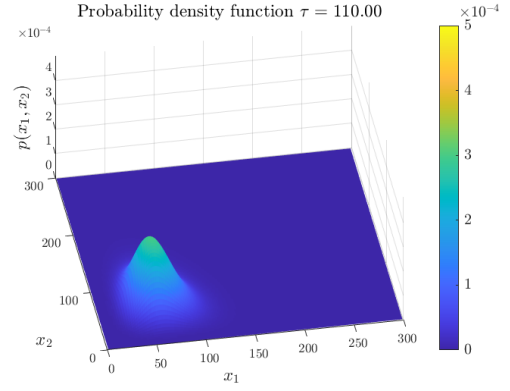


Altering inducer

Desired expected values: $m_1^* = 41 \text{ nM}$ and $m_2^* = 55 \text{ nM}$



Constant inducer



Altering inducer

Desired expected values: $m_1^* = 60 \text{ nM}$ and $m_2^* = 70 \text{ nM}$

Figure 4.6: Steady-state probability density function with different expected values. On the left, we can see the outcome of the original PI control, while on the right the result of the PI using altering input is represented. These examples show that the altering input can generate steady-state probability density functions with much smaller deviations.

Chapter 5

Control by Periodic Inputs

This chapter summarizes the periodic input's effect on the system, and how I examined it.

5.1. Motivation for studying oscillatory inputs

As it has been mentioned in the previous section, numerical experiments showed that non-slowly-varying inputs may qualitatively change the stationary protein distribution. This means that we may reach stationary distributions having the same means but different covariance properties. Note that this is not possible with inputs converging to constant values, since the mapping between means and inducer values is invertible within the reachable set, and the asymptotically stable equilibrium point is unique in that case according to Theorem 1. Another point of motivation is that oscillatory signals are widely used for signal processing and control in other fields such as neural networks and machine learning (see, e.g. [19]). Finally, the stability of the solutions of nonautonomous master equations is sufficiently understood for periodic coefficient matrices (Theorem 2). We add, however, that recent results show that nonautonomous master equations may show in some sense unexpected behaviour for certain non-slowly varying coefficient matrices [17]. Therefore, as a first step, we decided to study the effects of periodic inducer inputs.

5.2. Creating periodic inputs

The simulation's code with PI control calculates the new amount of inducer in every step. Instead, I intended to work with an array of predetermined values for the inducers thus I modified the code to use an array from an external source and wrote a script to create the array with the appropriate values.

Initially, I decided to extract the dominant frequency using a Fourier transform. For this, I saved the oscillating values calculated by the PI control, applied the `fft` MATLAB function on this data and identified the dominant frequencies. The results showed that the first inducer's frequency is $173Hz$ and

the second inducer's is $210Hz$. Then I created two sinusoidal signals with these frequencies, considering the time step and the simulation's duration to determine the sampling points.

I simulated the system using the sinusoidal signal and compared its results with the ones that the PI with altering input produced. The results can be seen in Table 5.1.

		PI control	sinusoidal input
expected values	first protein	41.0482 nM	52.8409 nM
	second protein	55.0093 nM	68.8588 nM

Table 5.1: Comparios of expected values of protein concentrations. Here PI control refers to the PI control that uses the PI constants determined with regulatory design and the desired values are $m_1^* = 41 \text{ } nM$ and $m_2^* = 55 \text{ } nM$. These constants can be seen in Table 4.1. The sinusoidal input has the frequencies $f_1 = 173 \text{ } Hz$ and $f_2 = 210 \text{ } Hz$ as extracted from the Fourier transform described in 5.2.

Table 5.1 shows that the sinusoidal input generated an expected value different from that of the PI's altering input, however, they have the same dominant frequency. This may be because, in addition to the frequency, the amplitude and the mean value of the sinusoidal signal could influence the outcome.

I compared the deviation of the probability density function created using the sinusoidal inducer, with the deviation of the proteins with the same expected values reached by constant inducers. The results can be seen in Table 5.2 The values of the deviations are smaller in the case of the sinusoidal inducers, meaning that periodic input can produce a less uncertain outcome.

		constant input	sinusoidal input
deviations	first protein	24.0044	19.8921
	second protein	31.6012	29.1757

Table 5.2: Deviations of protein concentrations. In these two cases, the probability density functions have almost the same expected values, approximately $m_1 = 53$ and $m_2 = 69$. In the first case, constant inducers were used to reach these expected values, and in the second case, sinusoidal inputs were used with the frequencies $f_1 = 173Hz$ and $f_2 = 210Hz$ as determined with the Fourier transform.

5.3. Relationship between the 6 inputs and the 4 outputs

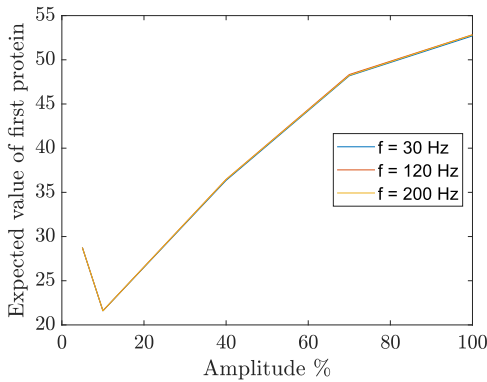
Six possible factors influence the probability density function's expected values in the case of sinusoidal inducers: the frequency, amplitude, and signal's shift from zero. My baseline/reference signal's minimum was zero and its maximum was the maximal amount of the corresponding inducer. I considered it as a signal with 100 % amplitude and 0 % shift. When I apply a shift of a certain percentage I mean that I shift the signal by this percentage of the corresponding inducer's maximum.

To examine the dependence between the input's properties (frequencies, amplitudes, shifts) and the output's properties (expected values, deviations) I simulated several cases and collected the information according to different aspects.

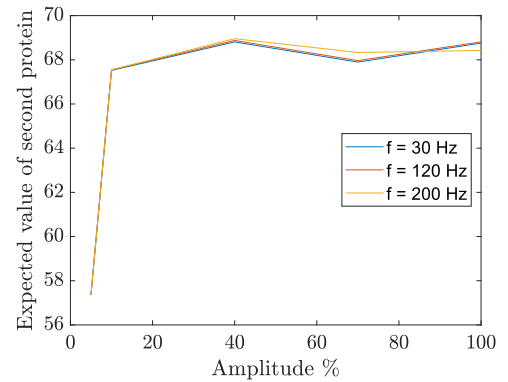
5.3.1 Frequency dependence of the system

Expected values and deviations The first observation was that neither the expected values nor the deviations depended on the inputs' frequencies. Figure 5.1, 5.2 and Table 5.3 demonstrate, that different frequencies result in outputs with almost the same properties.

Figure 5.1 shows that the expected values of the proteins are almost identical for the three examined frequencies. Noticeably, the relationship between the amplitude and the expected values is difficult to identify. In Figure 5.2 we can see that the distributions' deviations are also almost identical as well. Additionally, it can be observed that a larger amplitude tends to result in smaller deviations. This means that a larger amount of inducers generally produces distribution with a smaller deviation. This is true for constant inducers as well. In these simulations, the frequencies of the first and the second inducers were the same, but for Table 5.3 different frequencies were used for the first and the second inducers. In these simulations as well, the expected values remained close to one another. For all the simulated cases the differences between the corresponding expected values were smaller than 1 nM and for the deviations approximately 0.1 nM . These differences are less than 2% of these values' ranges.



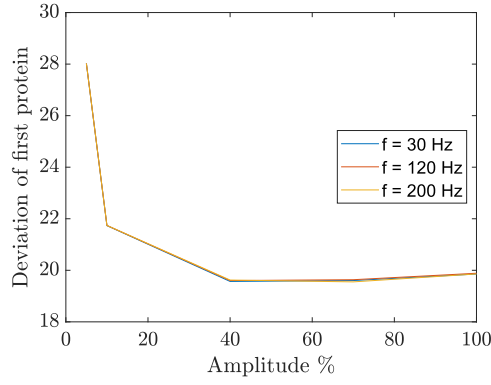
Expected value of the first protein



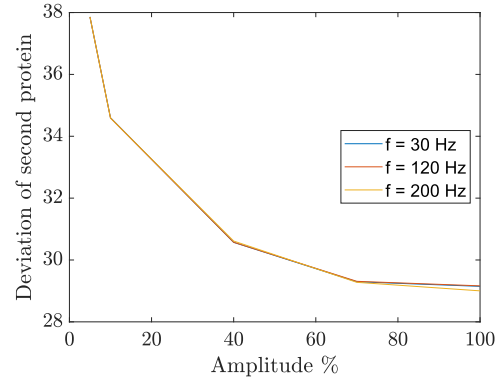
Expected value of the first protein

Figure 5.1: Relationship between the amplitude, the frequency and the expected values. The different lines correspond to different frequencies. It can be seen that the expected values remain consistent across different frequencies. $f = f_1 = f_2$ is the frequency of both the first and the second inducer, and there is no shift.

Based on these and a few additional results I concluded that the inducers' frequencies do not impact the expected values and the deviations of the steady-state probability density function.



Deviation of the first protein



Deviation of the first protein

Figure 5.2: Relationship between the amplitude, the frequency and the deviations. As in the previous figure, the different lines correspond to different frequencies, showing that neither the deviations depend on frequency. $f = f_1 = f_2$ is the frequency of both the first and the second inducer, and there is no shift.

1. frequency (%)	2. frequency (%)	1. expected values (%)	2. expected value (%)
30	30	36.3560	68.8097
30	120	36.3518	68.8826
30	200	36.3276	68.9647
120	30	36.4611	68.8166
120	120	36.4650	68.8741
120	200	36.4363	68.9649
200	30	36.4737	68.8187
200	120	36.4734	68.8837
200	200	36.4548	68.9556

Table 5.3: Expected values of the first and the second inducer in the case of sinusoidal inducers with different frequencies. The expected values are close to each other for every frequency pair. In each case, the amplitude was 40 % of the maximum, and no shifts were applied.

Time evolution In contrast to the steady-state distribution's properties, the frequency altered the system's time evolution. Actually, the expected values of the system do not reach a constant value like in the case of constant inputs, instead, they fluctuate with a small amplitude. Therefore, we can only say that the system has an approximate stationary distribution in this case. However, the fluctuations are so small for frequencies greater than 30 Hz that practically they cannot be observed. The frequency of the inducers affects both the frequency and amplitude of the fluctuation; higher frequencies result in smaller fluctuation amplitudes. The first input affects the fluctuation in the first protein's concentration, while the second input affects the second protein separately.

It appeared reasonable that the frequency of the inducer was equal to the frequency of the fluctuation in the corresponding expected value. The graphs indicated that, indeed, the expected values' time evolution followed the fluctuations of the inducers with a slight delay. The Fourier transforms of the inducer and the expected value confirmed that the frequencies of these functions are identical.

Lower limit of frequency Considering that the frequency does not affect the steady-state expected value of the system, (except for a small fluctuation,) the question arose, what is the smallest possible frequency that results in an appropriate outcome, and whether it has a smaller deviation, than the corresponding distribution reached with constant inducers. Clearly, it depends on the boundaries of appropriateness, but for example, if $f_1 = f_2 = 5Hz$ then the amplitude of the fluctuation is approximately $2 nM$ and the input still alternates rapidly, however, it may be slow enough to be implemented in real life. Meanwhile, the deviations of the probability density function remain as small as in the case of much higher frequencies.

Since the frequency does not affect the steady-state distribution's properties I used $f_1 = f_2 = 100 Hz$ for further simulations.

5.3.2 Amplitude and shift dependence

The effect of the frequency was easy to observe, whereas the relation between the amplitude or the shift and the expected values was hard to observe.

The first idea was that the expected value could depend on the integral of the inducer; however, counterexamples easily disproved this. Another option could have been that the root mean square correlates with the expected values, but the experiments did not confirm this.

I attempted to plot multiple simulation series to identify the relationship between amplitude and expected values, as well as between the shift and expected values, but none yielded satisfactory results.

5.4. Lookup table using periodic input

The next reasonable step was to create a four-dimensional lookup table that maps the four parameters of the input ($amplitude_1$, $amplitude_2$, $shift_1$ and $shift_2$) to the two expected values (E_{x_1} , E_{x_2}). I planned to iterate through a range of the amplitudes and shifts of the first and second inducers. I considered that the maximums of the amplitudes were independent variables. In contrast, the maximums of the shifts depended on the actual amplitude since the sum of the amplitude and the shift must not exceed the maximum amount of the inducer. I wrote a script to iterate through the four ranges. Each

simulation takes about 20 to 30 minutes, making the creation of the four-dimensional matrix quite time-consuming.

After a few attempts, I created a lookup table with a size $8 \times 8 \times 11 \times 11$. The result of running the code can be seen in Figure 5.3. Comparing the original reachability set with the distribution of the four-dimensional lookup table's points shows that new points cannot be reached using periodic input.

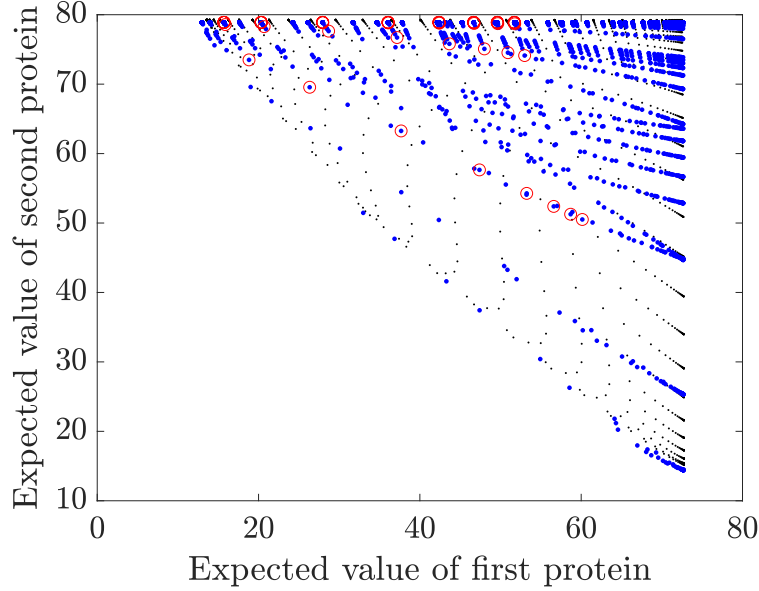


Figure 5.3: Lookup table using periodic input. The blue dots represent the points of the lookup table. The black dots are the points of the lookup table using constant input, to show approximately the reachability set. Red circles marks the points of the subset used in Subsection 5.5.1.

I used identical step sizes between the amplitudes and the shifts for every part of the lookup table. However, it created significantly different distances between the points across distinct parts of the lookup table. I observed a few characteristics regarding the system's response to the 4 inputs but couldn't discern any clear rules.

5.5. Inversion of the four-dimensional lookup table

Similarly to the case of constant inducers, I aimed again to determine the parameters of the periodic input needed to achieve the desired expected values by inverting the lookup table. An $\mathbb{R}^4 \rightarrow \mathbb{R}^2$ function, like this four-dimensional lookup table is generally not invertible, since it maps a higher-dimensional space to a lower-dimensional space. Therefore a unique solution for a pair of expected values cannot be found based on this lookup table.

desired E_{x_1}, E_{x_2}	computed $amp_1, shift_2$	predicted E_{x_1}, E_{x_2}	actual E_{x_1}, E_{x_2}
28	0.1938	28.0000	27.6942
78	0.1274	78.0000	78.4354
50	0.4464	50.0000	49.2468
65	0.0526	65.0002	66.9849
40	0.2898	40.0000	38.6071
70	0.0634	70.0002	72.5825

Table 5.4: Inversion of a two-dimensional subset of the four-dimensional lookup table. The first column contains the desired expected values of the concentrations, while the second column lists the parameters of the inducer amounts computed by the inversion. The third column shows the predicted concentrations corresponding to the calculated inducers, as determined by interpolation from the lookup table. In the fourth column the actual E_{x_1} and E_{x_2} represent the expected values of the concentrations obtained through simulation with the same inducer amounts.

5.5.1 Inversion of two-dimensional subset

First, a two-dimensional subset was inverted, as that may be uniquely invertible. A two-dimensional matrix was created by fixing two parameters of the lookup table and squeezing this subset. The expected values corresponding to the subset's points can be seen in Figure 5.3 circled with red. This matrix was easily invertible using the same optimizing method described in Section 3.5.2.

Table 5.4 shows a few examples of the result. The desired expected values in the first example are close to points of the lookup table, while in the second and the third, they are further. In the last two cases, the error is larger, as the interpolation causes bigger aberration in these results.

5.5.2 Inversion of the entire lookup table

None of the two-dimensional slices seemed to cover the entire reachability set, so I applied optimization-based inversion to the four-dimensional lookup table.

Figure 5.4 shows the result of the inversion with desired expected values $m_1^* = 50 \text{ nM}$ and $m_2^* = 65 \text{ nM}$. A possible outcome could have been that the inversion of the entire lookup table would result in a smaller error than the inversion of the subset since it has more data to rely on, but instead, it calculated a less precise approximation.

Figure 5.4 shows that the inversion based on a 2D subset provides a better solution than the inversion based on the 4D lookup table, and searching for the closest point of the reachability set would give an even more accurate result, highlighting the limitations of these inversion methods in achieving the

desired outcomes.

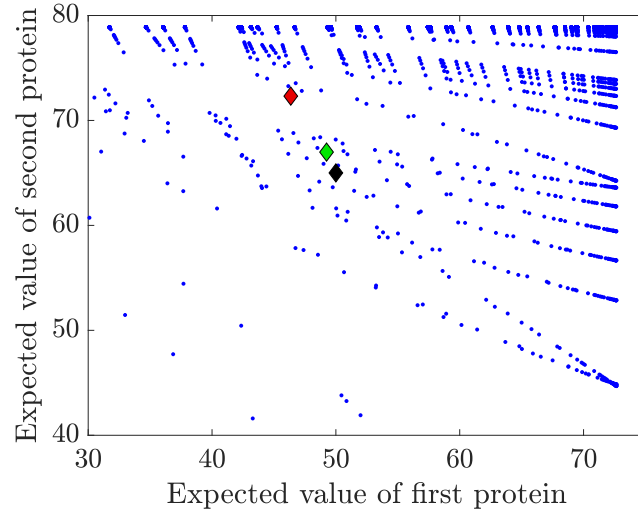


Figure 5.4: Example results of the inversion using the entire lookup and the subset. The black diamond denotes the desired output, the red denotes the output of the four-dimensional inversion and the green denotes the result using the inversion of the two-dimensional subset. Blue dots denote the points of the four-dimensional lookup table.

This unexpected result is due to the lookup table's nonlinearity, which makes the interpolation inaccurate. Moreover, the distances between the lookup table's points vary greatly, so the inversion can result in errors of different magnitudes. In every case, the inversion predicts a result with an accuracy of four decimal places, and it stops searching for a better solution because the error is sufficiently small.

There are two possible options to resolve the problem. First, a lookup table with enough points could be suitable for inversion. The expected values' distances between the lookup table's adjacent points should be under an appropriate limit. Therefore the lookup table would require so many points that computing all of them would be excessively time-consuming.

The second option is to minimise the inversion error by considering the deviation from the closest point of the lookup table. This would be difficult to implement, and it still would not lead to an accurate result.

An appropriate solution is to find an approximation of the result and to create an accurate inversion around that point.

5.5.3 Inversion around arbitrary point of the lookup table

To create a precise inversion in a domain of the reachability set I chose the lookup table's closest point to (50, 65) with the following properties: $amplitude_1 = 0.2$, $amplitude_2 = 0.2$, $shift_1 = 0.2$, $shift_2 = 0$, $m_1 = 50.1851$, $m_2 = 65.7069$. First I fixed the first and last parameters of the inducers, the values

of $amplitude_1$ and $amplitude_2$. In this part of the lookup table increasing the value of $amplitude_1$ resulted in a six times less deviation in the expected values than increasing the value of $shift_1$ by the same rate, so I decided to use six times bigger step sizes for the division of $amplitude_1$. I created the following partitioning:

```
amplitude_2 = [0.41:0.03:59]; shift_1 = [0.185:0.005:0.215];.
```

To create this small lookup table 7×7 simulations were needed, and it took only 20 minutes to calculate it on my laptop. (In contrast, the runtime of calculating the entire four-dimensional lookup table was around 15 hours.) Figure 5.5 shows the results.

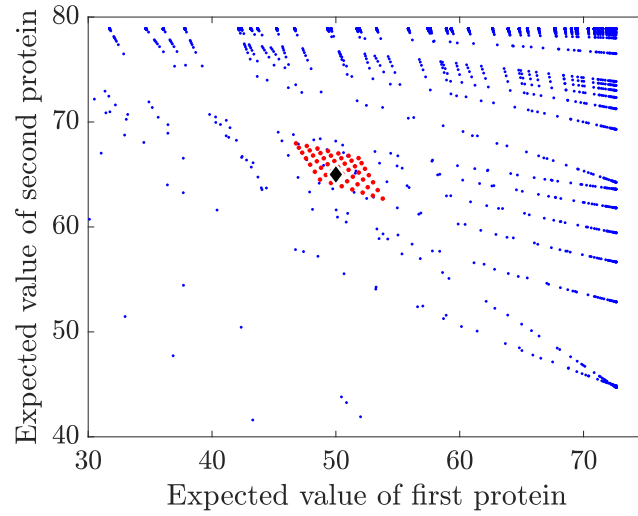


Figure 5.5: Lookup table around expected values (50, 65). Blue dots denote the points of the four-dimensional lookup table, and red dots denote the points of the more finely partitioned lookup table. The black diamond denotes the expected value pair (50, 65).

Using this lookup table the inversion with desired values $m_1^* = 50$ and $m_2^* = 65$ calculated an outcome of $m_1 = 49.9967$ and $m_2 = 65.0078$ which is much more precise than any of the previous results.

In this specific point, the deviation of the first and the second protein was $deviation_1 = 22.7470$ and $deviation_2 = 29.7423$ while the deviations using constant inducers are $deviation_1 = 25.0388$ and $deviation_2 = 32.4635$. These two probability density functions can be seen in Figure 5.6

Repeating these steps around other points of the original lookup table allows one to find the inversion of any point with arbitrary precision since the fineness of the partitioning can be increased to any degree.

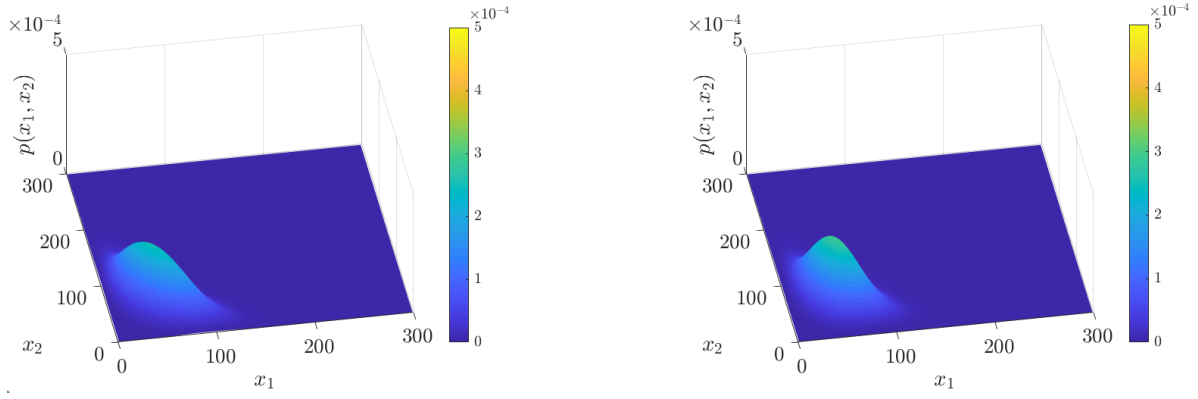


Figure 5.6: Steady-state probability density functions with expected values $m_1^* = 50$ and $m_2^* = 65$. The first function was reached with constant inducers, while the second was reached using sinusoidal inputs.

5.6. Analysis of the steady-state distribution's deviations

I created several lookup tables using different grids of the inputs' parameters. In all instances, the deviations of the probability density function were smaller when achieved with periodic input compared to constant input. Each time I attempted to select sufficiently small step sizes to ensure that the gradient between two adjacent points in the lookup table could be approximated with a linear transition.

Comparison of the lookup tables using constant and periodic input I aimed to compare the deviations for all points in the lookup table, so I created a Figure illustrating these deviations across the tables. I used a lookup table with 19×7 different periodic inputs. Figure 5.7 shows the result. It can be seen, that in every examined part of the reachability set the deviations are smaller when reached with periodic input.

Reaching the same expected values with different periodic inputs Since the four-dimensional lookup table maps to two dimensions regarding the two expected values, multiple ways may exist to achieve the same expected values. I searched for two grids that generate the same expected values.

For the first lookup table I used the same as in 5.7 with size 19×7 while the second one has size 11×11 . The properties of the two grids thus the amplitudes and the shifts used in the two sinusoidal inputs can be seen in Table 5.5.

Using these grids I could reach probability density functions with the same expected values but different deviations with distinct sinusoidal inputs. In this case the deviation of the first protein is smaller in the second lookup table, while the deviation of the second protein is larger for every pair of expected values.

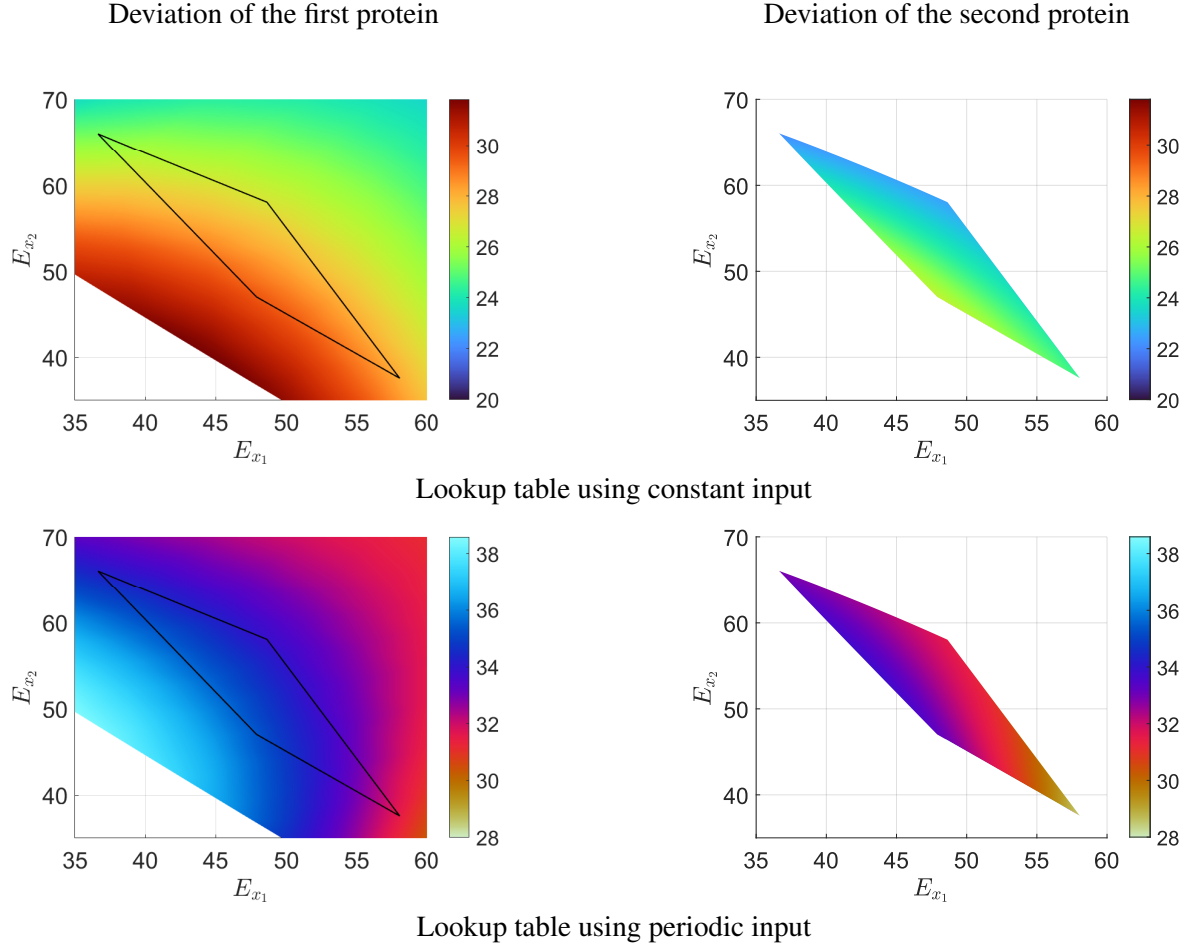


Figure 5.7: Deviation of the steady-state probability density function on the top using constant and on the bottom using periodic inputs. The colours on the left represent the deviation of the first protein and on the right the second protein's deviation in nM . The contour of the lookup table using periodic input is marked with black quadrilateral on the left images. It is easy to observe, that periodic input generates probability density functions with smaller deviations.

	first version	second version
$amplitude_1$ (%)	0.37:0.01:0.55	0.1
$amplitude_2$ (%)	0.05	0.1:0.01:0.2
$shift_1$ (%)	0	0.15:0.005:0.2
$shift_2$ (%)	0.035:0.005:0.065	0

Table 5.5: Values of the grid for the two different lookup tables. Each number represents the percentage of the corresponding inducer's maximum used for the input. The vectors use the notation of MATLAB as `first_value:step_size:last_value`

Chapter 6

Summary and Further Work

The dynamics and control of gene regulation networks was studied in this work. Due to the inherently stochastic nature of protein production, an important requirement from the applied model is to be able to quantify the change of the whole probability distribution of the protein molecules involved. Therefore, a PIDE-based mathematical description was chosen from the literature, where the modeled time-dependent variable is the potentially multidimensional probability density function of the molecular distribution. However, it is known that such PIDE models are computationally difficult to use even for simulation, and they are not directly suitable for controller design. It is a recent result published in 2024, that a special finite volume discretization approach transforms the PIDE model into a generally large dimensional compartmental ODE form which is mathematically a nonautonomous master equation. This means that the extensive theory of such master equations can be used for the analysis of the studied gene regulation networks. The maintenance and computation of multidimensional distribution functions is challenging even in the proposed ODE form, therefore, a simple genetic toggle switch was chosen as an example which illustrates the approach.

The new scientific results of the work related to the toggle switch model are the following:

- It was shown that time-varying inducer (input) values converging to a constant value guarantee the uniqueness and asymptotic stability of a stationary protein distribution independently of the parameter values of the model.
- A computational procedure was proposed to compute constant inducer (input) values which guarantee given mean values for the stationary protein distribution. The procedure is based on building a lookup table containing corresponding mean and inducer values, and derivative free optimization (pattern search).
- It was shown that oscillatory inducer values still guarantee the asymptotic stability of the solution of the ODE-model independently of the parameter values.

- It was shown through computational experiments that oscillatory (periodic) inducer values may result in a stationary distribution having the same expected value as for constant inducer values, but substantially smaller variance. I'm not aware of any similar results in the literature.
- A computational procedure using a lookup table and optimization was proposed to compute the amplitude and frequency of periodic inputs to reach a given expected value.

The main limitations of the work are the following. Although the kinetic discretization largely simplifies the handling of the model compared to the original PIDE form, the curse of dimensionality still applies to the ODE model. In other words, the number of state variables is increasing quickly with the additional proteins involved in the network. It is well-known from other areas (e.g., from Bayesian modeling) that computation with high dimensional probability distribution functions is challenging. Another limitation is that the inducer inputs appear in the ODE model in a way that is difficult to handle for direct nonlinear control design. However, the linearized model is still very useful for control.

There are several open questions that remained unsolved. An important one is to theoretically show the global stability of controlled networks (e.g., by PI control or through more advanced control schemes). Another interesting problem is to prescribe not only the mean value but also other important features of the stationary distribution such as variance or other momenta.

Publications The following posters were accepted and presented at international conferences on the topic of my thesis:

- Gábor Szederkényi, Dorka Kecskés, Mihály Vághy. Stationary distribution control of a gene regulation network using compartmental discretization. 3rd International Conference on Reaction Kinetics, Mechanisms and Catalysis (RKMC 2024), May 22-25, 2024, Budapest
- Gabor Szederkenyi, Dorka Kecskés. Stationary distribution control of a gene regulation network given in PIDE form. IFAC Workshop on Biological Control Systems. November 13, 2024.

Bibliography

- [1] A. Ay and D. N. Arnosti, “Mathematical modeling of gene expression: A guide for the perplexed biologist,” *Critical reviews in biochemistry and molecular biology*, vol. 46, no. 2, pp. 137–151, 2011.
- [2] D. del Vecchio, J. Dy, and Y. Qian, “Control theory meets synthetic biology,” *The Royal Society Interface*, vol. 13, no. 120, p. 20160380, 2016.
- [3] C. Briat, A. Gupta, and M. Khammash, “Antithetic Integral Feedback Ensures Robust Perfect Adaptation in Noisy Biomolecular Networks,” *Cell Systems*, vol. 2, no. 1, pp. 14–26, 2016.
- [4] J.-B. Lugagne, S. S. Carrillo, M. Kirch, A. Köhler, G. Batt, and P. Hersen, “Balancing a genetic toggle switch by real-time feedback control and periodic forcing,” *Nature Communications*, vol. 8, no. 1, 2017.
- [5] N. Friedman, L. Cai, and X. S. Xie, “Linking Stochastic Dynamics to Population Distribution: An Analytical Framework of Gene Expression,” *Physical Review Letters*, vol. 97, no. 16, 2006.
- [6] V. Elgart, T. Jia, A. T. Fenley, and R. Kulkarni, “Connecting protein and mRNA burst distributions for stochastic models of gene expression,” *Physical Biology*, vol. 8, no. 4, p. 046001, 2011.
- [7] M. Pájaro, A. A. Alonso, I. Otero-Muras, and C. Vázquez, “Stochastic modeling and numerical simulation of gene regulatory networks with protein bursting,” *Journal of Theoretical Biology*, vol. 421, pp. 51–70, 2017.
- [8] M. A. Vághy, I. Otero-Muras, M. Pájaro, and G. Szederkényi, “A Kinetic Finite Volume Discretization of the Multidimensional PIDE Model for Gene Regulatory Networks,” *Bulletin of Mathematical Biology*, vol. 86, no. 2, 2024.
- [9] M. A. Vághy, I. Otero-Muras, M. Pájaro, and G. Szederkényi, “A kinetic finite volume discretization of the multidimensional pide model for gene regulatory networks,” *Bulletin of Mathematical Biology*, vol. 86, no. 2, p. 22, 2024.

- [10] J. A. Cañizo, J. A. Carrillo, and M. Pájaro, “Exponential equilibration of genetic circuits using entropy methods,” *Journal of Mathematical Biology*, vol. 78, pp. 373–411, 1-2 Jan. 2019, ISSN: 14321416. DOI: 10.1007/s00285-018-1277-z.
- [11] P. Érdi and J. Tóth, *Mathematical models of chemical reactions: theory and applications of deterministic and stochastic models*. Manchester University Press, 1989.
- [12] N. Van Kampen, *Stochastic processes in physics and chemistry*. Elsevier, 2007.
- [13] P. Érdi and G. Lente, *Stochastic chemical kinetics. Theory and (Mostly) Systems Biological Applications*. Springer, 2014.
- [14] D. H. Anderson, *Compartmental modeling and tracer kinetics*. Springer Science & Business Media, 2013, vol. 50.
- [15] H. Roger and R. J. Charles, *Topics in Matrix Analysis*. Cambridge University Press Cambridge, UK, 1994.
- [16] B. A. Earnshaw and J. P. Keener, “Global asymptotic stability of solutions of nonautonomous master equations,” *SIAM Journal on Applied Dynamical Systems*, vol. 9, no. 1, pp. 220–237, Jan. 2010, ISSN: 1536-0040. DOI: 10.1137/090763421.
- [17] J. Diblík, M. Pituk, and G. Szederkényi, “Large time behavior of nonautonomous linear differential equations with Kirchhoff coefficients,” *Automatica*, vol. 161, p. 111 473, Mar. 2024, ISSN: 0005-1098. DOI: 10.1016/j.automatica.2023.111473.
- [18] M. Pituk, “Global asymptotic stability of nonautonomous master equations: A proof of the Earnshaw–Keener conjecture,” *Journal of Differential Equations*, vol. 364, pp. 456–470, 2023.
- [19] T. Rudner, W. Porod, and G. Csaba, “Design of oscillatory neural networks by machine learning,” *Frontiers in Neuroscience*, vol. 18, Mar. 2024, ISSN: 1662-453X. DOI: 10.3389/fnins.2024.1307525.