TOWARDS MODULAR, SCALABLE AND OPTIMAL DESIGN OF TRANSCRIPTIONAL LOGIC SYSTEMS

A THESIS SUBMITTED TO

THE UNIVERSITY OF KENT AT CANTERBURY

IN THE SUBJECT OF COMPUTER SCIENCE

FOR THE DEGREE

OF DOCTOR OF PHILOSOPHY.

By

Nicolae Radu Zabet

October 2010

Abstract

Living organisms can perform computations through various mechanisms. Understanding the limitations of these computations is not only of practical relevance (for example in the context of synthetic biology) but will most of all provide new insights into the design principles of living systems.

This thesis investigates the conditions under which genes can perform logical computations and how this behaviour can be enhanced. In particular, we identified three properties which characterise genes as computational units, namely: the noise of the gene expression, the slow response times and the energy cost of the logical operation.

This study examined how biological parameters control the computational properties of genes and what is the functional relationship between various computational properties. Specifically, we found that there is a three-way trade-off between speed, accuracy and metabolic cost, in the sense that under fixed metabolic cost the speed can be increased only by reducing the accuracy and vice-versa. Furthermore, higher metabolic cost resulted in better trade-offs between speed and accuracy. In addition, we showed that genes with leak expression are sub-optimal compared with leak-free genes. However, the cost to reduce the leak rate can be significant and, thus, genes prefer to handle poorer speed-accuracy behaviour than to increase the energy cost. Moreover, we identified another accuracy-speed trade-off under fixed metabolic cost, but this time the trade-off is controlled by the position of the switching threshold of the gene. In particular, there are two optimal configurations, one for speed and another one for accuracy, and all configurations in between lie on an optimal trade-off curve.

Finally, we showed that a negatively auto-regulated gene can display better trade-offs between speed and accuracy compared with a simple one (a gene without feedback) when the two systems have equal metabolic cost. This optimality of the negative auto-regulation is controlled by the leak rate of the gene, in the sense that higher leak rates lead to faster systems and lower leak rates to more accurate ones. This in conjunction with the fact that many genes display low but non-vanishing leak rates can indicate the reason why negative auto-regulation is a network motif (has high occurrence in genetic networks).

These trade-offs that we identified in this thesis indicate that there are some physical limits which constrain the computations performed by genes and further enhancement usually comes at the cost of impairing at least one property.

Acknowledgements

Firstly, I am really grateful to my supervisor Dr. Dominique Chu for believing in me and for his support and valuable guidance over the last three years. His passion for research was an inspiration for me.

Furthermore, I would like to thank my supervisory panel, Dr. Tim Hopkins and Dr. Colin Johnson, for all their support, encouragements and valuable comments on my research.

In addition, I would like to thank Professor Andy Hone for the great collaboration we had and for his help during various stages of my PhD. I would also like to use this opportunity to thank David Barnes for his continuous support during my PhD.

I owe a big thank to my wife Dana for all her support, understanding and love during my PhD. I would also like to thank her and Chris Silles for proofreading and correcting the English of my thesis and my articles.

Also, I would like to present my deepest love and respect to my parents, Mariana and Alecsandru, for raising me and most importantly for instilling in me the intense desire to learn.

Finally, I would like to acknowledge and thank the School of Computing at the University of Kent for the financial support, which made my PhD possible.

Dedication

For my wife Dana, my mother Mariana and my father Alecsandru.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Currently, computing hardware is made from inorganic materials (silicon). However, in principle, computations can also be performed on other mediums, including components of living systems such as biological molecules. The viability of using biological molecules in computational problems was proven both theoretically and experimentally in previous studies [29, 175]. These studies modelled and some even built systems able to perform different types of computations, such as: numerical computations (e.g., arithmetic units [38] or counters [52]), combinatorial problems (e.g., Hamiltonian Path Problem [3] or satisfiability problem [99]) and logic computations (e.g., logic gates [32], binary arithmetic units [81, 121, 96] and even a tic-tac-toe game [160]).

This thesis aims to examine theoretically various aspects of the design of computational systems constructed from biological molecules, from an interdisciplinary perspective, reuniting knowledge from biology, computer science, mathematics, physics, engineering, and chemistry. Particularly, we want to identify a modular and optimal design of genetic logic computational systems.

Usually, we want computations to be performed as fast as possible, but computations carried out using biological molecules are much slower compared with silicon based ones [3, 29]. Thus, first we need to answer the question: what are these biological computations best suited for? Despite the slow speed, these biological computational units have the advantage of miniaturisation, low energy consumption and natural interaction with biological organisms. This indicates that one area where these bio-computers can be very useful is in biological applications.

For example, these biological computational units can be used in smart drug delivery systems. The bio-computer can perform *in situ* diagnosis of a disease and, depending on the diagnosis, it can release the appropriate drug [23]. Alternatively, bacterial cells can be engineered to detect high density of mammalian cells (usually associated with tumours) and then invade these cells and release a chemical kill signal [7]. Other examples of possible applications include: bacterial cells able to 'eat' oil spills [107], sensor cells aimed to detect the presence or absence of a substance in an inaccessible environment [164], tissue engineering and fabrication of biomaterials [16], mechanisms which control the density of a bacterial population [183] or cells engineered to produce synthetic drugs [180]. To summarise, we can say that these bio-computers may prove to be of extreme importance for applications in the pharmaceutical industry, environmental applications and various branches of the economy.

From the above examples we can infer that, in general, we may want cells to take decisions based on various internal and external factors. Often, these decisions need to be taken based on the presence or the absence of various chemical factors and, thus, the decision systems can be modelled by logic functions. This logic function approach is inspired by the fact that cells seem to display a Boolean logical behaviour. For example, the proteins associated with the lactose metabolism are produced only when the glucose is absent and the lactose is present and this is often approximated by a NIMPLIES gate [148]. Although there are various types of computations that can be performed in biological systems, in this thesis, we will limit our focus only to logical computations, which, as we have seen above, are of extreme importance in biological applications.

The first step to engineer logical computations within the cell consists of building a library of elementary components [9, 172] and, using the analogy with electronic circuits, this library should include: logic gates [71, 32, 39], memory units

[57, 91], clocks [47], counters [52], pulse generators [17]; see http://partsregistry.org. Once this toolbox is complete, more complex logical systems can be constructed. In this contribution, we will address only logic gates and logic gate systems, but similar mechanisms can be employed in the design and analysis of other types of parts (such as memory units, clocks or counters).

Depending on the biological molecules used in constructing logic gates, there are mainly three types of biochemical logic gate: (*i*) DNA based, (*ii*) enzymatic and (*iii*) genetic logic gates. DNA based logic gates use the fact that two complementary DNA strands anneal (bind). The input of the gate is a DNA strand able to anneal to a second DNA strand (representing the gate) and to perform a transformation on a third DNA strand (the output).

Alternatively, logic gates can be implemented from allosteric enzymes. These proteins can selectively catalyse the transformation of a substrate into a product only when the enzyme is activated. The state of the enzyme is controlled by inducer molecules which represent the input in the system, while the output is represented by the transformed product protein.

Finally, logic gates can be constructed from genes by exploiting the fact that genes express output proteins based on the occupancy state of their *cis*-regulatory area. If we consider the regulatory molecules to be the input of the system and the product protein the output, then a gene can mimic the behaviour of a gate, i.e., the gene processes one or more inputs and, based on a built-in function, produces an output. Genetic logic gates that integrate multiple inputs in the *cis*-regulatory area are called transcriptional logic gates. In this type of model, we can think of cells as being a sort of Turing machine, where environmental signals (chemical or physical) activate an existing gene program (encoded on DNA) which performs certain tasks by expressing the appropriate proteins [153].

In addition to being able to mimic logic behaviour, logic gates also need to address various design aspects, such as: (*i*) to have a reset mechanism, (*ii*) to possess an addressing mechanism, (*iii*) to allow interconnection and (*iv*) to be easily fine-tuned. All three types of biochemical logic gate can display a *reset* *mechanism.* In the case of DNA and enzymatic gates, the reset mechanism consists of adding another substance able to process the output molecules [41]. For genetic logic gates, the cell has a simple built-in auto-reset mechanism through the decay process, i.e., proteins (both output and input ones) are decayed by either active degradation (carried out by large macromolecules within the cell) or dilution (due to the increase in size of the cell).

Moreover, signals in a molecular based logic gate are not separated spatially, as in the case of electronic circuits, but they are encoded by proteins, which flow together in a common compartment [174, 8]. These signals are *addressed* (they are plugged in the input of a gate) based on the specificity of the encoding protein, in the sense that the encoding protein can react only with specific gates. If we have a high number of signals in a single compartment, then these signals can affect each other (there can be unwanted reactions between different proteins) and this leads to cross-talk. Thus, the addressing mechanism used by biological molecules puts an upper limit on the number of signals that a logical system can have. Nevertheless, the advantage of this addressing mechanism is that multicasting and broadcasting are easier to implement in a molecular based system compared with an electronic one. These two mechanisms (multicasting and broadcasting) assume that one signal can be fed simultaneously into multiple gates (or in all gates, in the case of broadcasting).

Furthermore, to allow the interconnection of logic gates, we need that the output and the inputs to be of the same type so that one gate's output can be fed into the input of another one. In the literature, this is, sometimes, referred to as *modularity of the design* [147, 150]. All types of gate (DNA, enzymatic and genetic) can have inputs and outputs of the same type, namely molecules and, thus, interconnection can be ensured by all types of biological gate.

Finally, we want to have fine *control* over both the behaviour of the gate and its parameters. This higher level of control is essential in systems that need to function with high precision. For example, in a smart drug delivery system we want the bio-computer to release a drug only after several conditions are *strictly* met. Also, we do not want the wrong drug to be released. We consider this higher degree of control to be an essential aspect of the logic gates which we aim to investigate in this thesis. Enzymatic gates can be fine-tuned by changing the concentrations of the gates [120], while genetic ones can be evolved (by performing point mutations on the gene) to optimise the parameters, but also to change the behaviour of the gate [182, 174, 105, 145]. The change of behaviour results from the fact that, in the case of genetic logic gates, the complexity of the logic gate is combinatorially built in the *cis*-regulatory area (as opposed to enzymatic logic gates, where the complexity arises from the complexity of the enzyme) [32].

In this thesis, we will limit our attention only to genetic logic gates and particularly to transcriptional logic gates, which are genetic logic gates that integrate inputs in the *cis*-regulatory area. Nevertheless, similar approaches can also be employed to design and analyse logic gates built from other biological molecules, such as DNA based logic gates or enzymatic ones.

1.1 Challenges of Designing Transcriptional Logic Gates

The input and output of the gene are quantified by the concentration of the relevant proteins. Logic gates strictly assume an output with two discrete values, but the output of a gene is continuous in nature (concentrations are continuous) [153]. In the case when the gene regulation function has a sigmoid shape (displays two plateaus, one for low and one for high expression rates), we can consider this sigmoid output to be *binary*, i.e., the low output concentration codes for a logical value of 0 while the high output concentration for a logical value of 1. Nevertheless, the quality of this approximation depends on the steepness of the gene regulation function in the sense that a steeper gene regulation function leads to better binary behaviour. Thus, when modelling transcriptional logic gates we first need to ensure that the gene expression function has a sigmoid shape and high steepness; see Figure 4.

Logic gates are usually meant to be components of larger logical systems and thus we need to also address interconnection between logic gates. Transcriptional logic gates use molecules as inputs and outputs and, thus, they display a *modular* design which allows feeding the output of one gate (the upstream gate) into the input of another (the downstream gate). Nevertheless, the process of connecting two biological logic gates is not simple, mainly because the parameters of the two gates which are connected do not always match. More precisely, if the regulatory threshold of the downstream gate (the input value which delimits the high and the low outputs) is not margined by the low and the high abundance of the output of the upstream gate, then changes in the upstream gate will not be reflected in the downstream one [10, 174]. In the case of transcriptional logic gates, one solution consists of using direct evolution to change the synthesis rate of the upstream gene or the threshold of the downstream one adequately, so that the parameters of two gates will match [182] (for more details see page 53). Note that, due to lower controllability, in the case of enzymatic or DNA logic gates, the only way to achieve this matching is through adding additional gates, such as amplifier ones [10, 147, 189].

Furthermore, when interconnecting a large number of logic gates, the difference between the high and the low output states of a gate tend to be reduced at the end of the cascade [103, 147]. This reduction between high and low steady states can make it difficult to distinguish between the two binary values of the output (whether we have a low or a high output). We want our design to be *scalable*, in the sense that adding more genes will not change the binary quality of the output of the system. In the case of enzymatic and DNA logic gates, this is ensured by constructing amplification logical gates and inserting them in the network where the signal quality has significantly degraded [147, 189]. Alternatively, in the case of enzymatic reactions one could add new reactions or change the reaction stoichiometry (the number of reactants or products in a reaction) to achieve this scalability [103]. The first solution would increase both the propagation time of a signal and the noise [32], which is undesirable, while the second one cannot be applied in the case of genetic systems. Nevertheless, we can use again direct evolution to select desired parameters and, thus, to achieve scalability.

In addition to these design features of transcriptional logic gates, we are also concerned with their quality. The main drawback of transcriptional logic gates compared with other molecular logic gates is the fact that they are much slower, i.e., enzymatic logic gates switch in the orders of a tenth of a second [29] while genes need tens of minutes [5]. Moreover, due to low copy numbers and slow reactions, genes are affected by noise [87]. This can make it difficult to distinguish whether the output is in a high or low state. Additionally, each gene has a metabolic cost attached to it and this cost is limited by the availability of resources, in the sense that the production rate of a gene cannot be increased further unless more resources are available. Previous research has tried to address these aspects in an independent fashion, i.e., looking at either noise or switching time one at a time. We consider that an integrated *optimality* analysis which investigates simultaneously the speed, accuracy and metabolic cost of genes or gene systems is essential to provide better (i.e., faster and more accurate) transcriptional logic gates.

Note that there are two types of noise that affect a genetic logic gate, namely analogue noise and digital noise. *Analogue noise* is the noise in the output resulting from intrinsic fluctuations and fluctuations in the input controlled by the shape of the gate. *Digital noise* is the noise in reading the gate output and is caused when the output, which is affected by analogue noise, will have the wrong concentration at the time of reading. In this thesis we address only analogue noise, which we simply call noise, while we assume that digital noise can be addressed by specific error connection techniques [132].

1.2 Aim and Objectives

The aim of this thesis can be summarized by the following sentence:

> *We aim to provide a modular, scalable and optimal design for logic systems built from genes.*

In particular, we aim to determine the relationship between the parameters of the genes and the computational properties that characterise a system, but also to investigate how changes in one computational property affect the other properties.

This aim will be achieved by attaining several objectives. First, we want to design modular and scalable transcriptional logic systems. To design logic gates from genes we need to ensure that the output concentration as a function of the input(s) displays a sharp sigmoid shape. This allows the approximation of the gene output by a *binary* output, which is a key requirement of logical systems. Furthermore, the *modularity* of the design is achieved by imposing a condition that the parameters of two interconnected genes match, in the sense that the output range of an upstream gene should span the transition area of the downstream gene. Additionally, we want the design of the logic system to be *scalable*, in the sense that adding or removing gates from the system should not significantly worsen the quality of the binary behaviour of the output.

Once we model a working logical system using genes, we want to *optimise* its behaviour. The optimality should address three properties, namely: switching time, output noise and metabolic cost. The first two properties aim to reduce the main drawbacks of transcriptional logic gates compared with enzymatic ones. Our analysis also investigates metabolic cost, due to the fact that we want the optimality of the systems to be valid even for cells living in environments with limited resources. As opposed to previous attempts, we aim to perform an *integrated* analysis with the objective to identify methods to optimise these three properties simultaneously. Alternatively, we will also examine whether certain network topologies can improve these genetic logic systems.

1.3 Original Contributions

Previous research mainly investigated noise, switching speed and metabolic cost of genes (or gene networks) as stand-alone properties. This independent approach identified possible solutions to optimise a certain property without investigating how these solutions affect other properties. In this thesis, we considered an integrated approach and determined the functional relationship between output noise, switching time and metabolic cost of binary genes (genes with two expression levels, high and low). The main result of this analysis is that under fixed metabolic cost, there is a trade-off between the output noise and the switching time.

Our analysis addresses two cases: (*i*) instantaneous input change and (*ii*) non-instantaneous input change. In the case of the former, the trade-off between speed and accuracy is controlled by the decay rate of the protein and can be enhanced by increasing the metabolic cost, i.e., higher metabolic cost results in better trade-off curves. Furthermore, we showed analytically that a gene which does not have a leak rate displays a better trade-off curve compared with a gene with equal metabolic cost but with non-vanishing leak rate, where the leak rate is the quantity of the output of the gene which is expressed when the gene is fully repressed (in the repression case) or when it is not activated (in the activation case).

In the case of non-instantaneous input change, the position of the threshold controls a trade-off between output noise and switching time. Particularly, there are two configurations which optimise the system in either output noise or switching time. The trade-off curve bounded by these two values is the optimal trade-off curve and points residing outside this optimal trade-off curve are sub-optimal in both speed and accuracy.

1.4 Overview of the Thesis

The thesis is structured as follows:

Chapter 2 reviews various mechanisms to perform computations using biological molecules. We establish that transcriptional logic gates can be used to implement basic logic gates and we present the advantages and disadvantages which these genetic gates have compared with other types of biologically inspired logic gates (such as enzymatic ones).

Chapter 3 introduces the methods used to describe the stochastic behaviour of chemical reaction systems and genetic systems. These methods are used in the research part of the thesis to compute the noise levels for different genetic systems. There are two categories of stochastic methods used in this contribution: (*i*) analytical methods to compute the noise levels and (*ii*) stochastic simulations to validate the analytical results. Moreover, we describe how we use these methods to generate the simulation results and also list various software used.

Chapter 4 describes the model of the switching mechanism necessary to construct the logic gates. A switch is constructed from a single gene (the binary gene) with multiple binding sites and regulated by transcription factors (TF) monomers. In addition, we describe the three properties that we are interested in: output noise, switching time and metabolic cost.

Chapter 5 presents an integrated analysis of binary genes as stand-alone elements, where the regulatory input of the gene is changed instantaneously. Our analysis investigated the three properties (noise, time and cost) simultaneously. The functional relationship between the three properties shows that, under fixed metabolic cost, there is a trade-off between the output noise and switching time and this trade-off is controlled by the decay rate of the protein. Furthermore, we found that higher metabolic cost results in better trade-off curves, and leak-free systems are optimal in terms of noise and time compared with the non-vanishing leak systems.

Chapter 6 extends the analysis from the previous chapter and considers the case of non-instantaneous input change. The optimality analysis identified a trade-off between switching time and output noise under fixed metabolic cost, but this time the trade-off is controlled by the position of the regulatory threshold relative to the high and low input abundances. The results show that there is an optimal trade-off curve which is delimited by the optimal configuration in speed and the one in accuracy, and any other configuration (outside this optimal trade-off curve) is sub-optimal. In addition, we also investigated how negative auto-regulation affects the trade-off curves and found that this feedback mechanism can enhance at least one of the properties (speed or accuracy).

Chapter 7 shows how this optimality analysis is useful in the context of genetic logic systems. As a case study, we modelled a binary full-adder using five logic gates. The design of the full-adder is modular and also scalable. The optimality analysis indicated how this system can be optimised in terms of speed or accuracy.

Chapter 8 draws the conclusions of the thesis, critically analyses the results and also indicates future research directions.

1.5 Publication List

Some of the work presented in this thesis was published by the author in the following peer-reviewed journal articles and conference papers.

Journal Articles

- Chu, D. F.; Zabet, N. R. & Hone, A. N. W. (2011), 'Optimal Parameter Settings for Information Processing in Gene Regulatory Networks', *BioSystems* doi:10.1016/j.biosystems.2011.01.006

- Zabet, N. R. & Chu, D. F. (2010), 'Computational limits to binary genes', *Journal of the Royal Society Interface* 7, 945-954. doi: 10.1098/rsif.2009.0474

- Chu, D.; Zabet, N. R. & Mitavskiy, B. (2009), 'Models of transcription factor binding: Sensitivity of activation functions to model assumptions', *Journal of Theoretical Biology* 257(3), 419-429. doi: 10.1016/j.jtbi.2008.11.026

Peer-reviewed Conference Papers

- Zabet, N. R.; Hone, A. N. W. & Chu, D. F. (2010), 'Design Principles of Transcriptional Logic Circuits', In *Artificial Life XII: Proceedings of the 12th International Conference on the Synthesis and Simulation of Living Systems*, 19-23 August, Odense, Denmark, MIT Press.

- Zabet, N. R. & Chu, D. F. (2010), 'Stochasticity and Robustness in Bi-Stable Systems', In *Bioinformatics and Biomedical Engineering (iCBBE), 2010 4th International Conference on*, 18-20 June, Chengdu, China, pp. 1-4, IEEE Xplore. doi: 10.1109/ICBBE.2010.5518099

Chapter 2

Molecular Computing

In this chapter, we will review various methods to perform computations using biological molecules and special emphasis will be placed on genetic logic gates. In addition, we will compare genetic logic gates to other types of molecular logic gates, identifying their advantages and disadvantages. Furthermore, we show that certain limitations of the basic components can hamper the increase in complexity of these transcriptional logic systems. This justifies the need for an integrated analysis which aims to optimise simultaneously multiple properties of stand-alone transcriptional logic gates, but also of transcriptional logic systems (systems consisting of many gates).

2.1 Introduction

Traditionally, biological molecules have been used to perform two types of computations: hard combinatorial problems and logic functions. The paradigm of molecular computing was first introduced by Adelman [3] who used DNA oligonucleotides to solve a hard combinatorial problem, the Hamiltonian path problem (HPP). Although this approach proved to be successful and was applied even to other combinatorial problems, such as the satisfiability problem [99], researchers raised concerns regarding the scalability of DNA computing. Alternatively, biological molecules can be used to perform logic functions, but these systems are slower compared with silicon based ones. Nevertheless, molecular logic gates have the advantage of naturally interacting with biological systems. Benenson *et al.* showed as a *proof of concept* that this type of systems can be used in smart drug delivery system where they can perform *in situ* diagnosis of a disease and administration of the appropriate drug [23]. The purpose of these genetic logic gates is not limited only to smart drug delivery systems, but rather include sensor arrays (to detect the presence or absence of a substance in the environment), improved drug synthesis and even the processing of certain chemicals (e.g., bacterial cells able to clean oil spills). Given the importance of their multiple applications, we consider that further research of these molecular logic systems is essential. Thus, in this thesis, we will focus on logical computations performed by biological molecules.

We consider it essential that molecular logic gates display a modular design and a reset mechanism. Thus, we will evaluate various designs of molecular logic gates with respect to these requirements. Furthermore, we will also examine the degree of control over the parameters of these logic gates.

DNA molecules can also be used to implement logic gates and even complex Boolean systems (such as full-adder or a tic-tac-toe game) [34]. These gates exploit the fact that complementary DNA strands anneal (bind), and based on the presence or absence of a chemical input (usually an oligonucleotide) they produce a chemical or photonic output. The main drawback of DNA logic gates is that they are affected by low speeds.

A faster solution for biological gates can be implemented with the help of allosteric enzymes. These enzymes have binding sites where inducers can bind and, depending on the occupancy state, the enzymes can selectively catalyse the transformation of a substrate into a product. However, the complexity in enzymatic logic gates results from the complexity of the enzymes and is not built in a modular fashion.

By comparison, genes offer a solution to this modularity problem. They have a *cis*-regulatory area, which controls the activity of the gene and integrates multiple regulatory inputs. This indicates that the complexity of a genetic system can be built combinatorially in this *cis*-regulatory area [32, 9]. The advantage of having a logic gate built combinatorially consists of the fact that new logic gates can be constructed just by rearranging the *cis*-regulatory area in an automatic fashion, thus simplifying the process.

Several models of logic gates constructed from genes were proposed theoretically [32, 78, 146, 58], but also engineered synthetically within live cells [182, 71, 105, 39, 8, 145]. What all these models have in common is that the inputs of the gates are represented by the transcription factors, which regulate the gene, and the output by the expressed protein. Although there are various approaches by which these genetic logic gates integrate multiple inputs, in this thesis, we will focus only on transcriptional logic gates, which are logic gates built from single genes where the multiple inputs are integrated in the *cis*-regulatory area. Note that these types of systems (transcriptional regulation ones) are the most implemented and best characterised modules at present [9].

The most important advantage of genetic logic gates consists of the fact that they can be evolved to change the behaviour of the gate, but also to fine-tune its parameters [182]. This aspect allows an extra degree of freedom to genetic logic systems compared with other types of logic gates (DNA and enzymatic ones). In particular, this means that synthetic biologists have a greater control on the behaviour of these genetic gates.

The signals of these logic gates are, in most cases, encoded by molecules and they are quantified by the concentration of the species. Nevertheless, while Boolean logic assumes only two discrete values, the concentration which quantifies the signals is a continuous measure. Despite this, it is usually very practical to make an abstraction and assume that these systems mimic a logical value. In addition, due to the fact that these signals flow in a common compartment they are prone to crosstalk, i.e., different signals affect one another undesirably. This limits the maximum number of signals that can flow in a common compartment and, thus, can reduce the number of applications where these systems can be implemented.

As mentioned in the previous chapter we consider only molecular logic gates that have a modular design and a reset mechanism. Modularity assumes that the inputs and the output are of the same type so that the output of a gate can be fed into the input of another one. All three types of logic gate (DNA based, enzymatic and genetic ones) have a modular design. However, in certain cases, the DNA based gates and the allosteric enzymatic ones also display a non-modular design, especially when the input is a chemical substance and the output is photonic (chemophotonic gates). Moreover, modularity also addresses issues related to parameter mismatch, in the sense that gates cannot always be connected with their original set of parameters and require various approaches to set their parameters accordingly.

The reset mechanism ensures that, once an input signal is no longer fed into the system, the output will automatically reflect the change. DNA based gates and enzymatic ones need an explicit reset mechanism, meaning that a step involving chemical reactions is required to clear the output of the gate. Genetic gates, however, have a built-in automatic reset mechanism, the decay of the signals. The decay is ensured by active breakdown or dilution inside a cell and is responsible for clearing a signal that is not sustained.

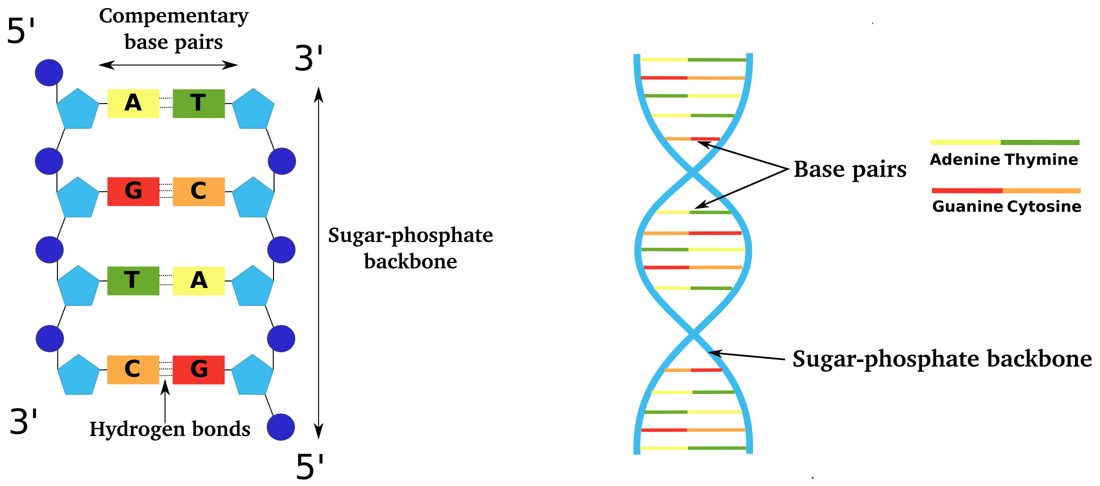
Finally, we will investigate the computational properties of transcriptional logic gates. In particular, we review noise in gene expression, response time and metabolic cost and how these three properties are related to the biological parameters of the genes. We will show that these three properties influence one another and we need to investigate how changes in one property affect the other properties.

This chapter is divided up as follows. We will start by presenting briefly the biological background related to molecular computing in bacterial cells. Next, we will review how DNA molecules were used to solve hard combinatorial problems (see section 2.3) and to engineer logic gates (see subsection 2.4.1). Furthermore, we will present various attempts to both model and construct logic gates using enzymes (see subsection 2.4.2) and genes (see subsection 2.4.2). Finally, in section

2.5 we review the computational properties of transcriptional logic gates. At the end of this chapter, we will draw some conclusions.

2.2 Biological Background

Living organisms store their genetic information in the DNA (Deoxyribonucleic acid). The DNA consists of oligonucleotides (or DNA strands), which are chains of nucleotides attached to a sugar-phosphate backbone [6]; see Figure 1. The information on the DNA is encoded by four nucleotides (adenine A, guanine G, cytosine C and thymine T) which pair up with each other using hydrogen bonds to form units called base pairs, i.e., adenine pairs up with thymine using two hydrogen bonds (A-T) while cytosine with guanine using three hydrogen bonds (C-G). The DNA strands have orientation, which is ensured by the two endings: a 5′ and a 3′ one. Two strands will bind (anneal) and form a double helix structure if they have complementary base pairs and opposite polarities, i.e., one strand extends from 5′ to 3′ while the other from 3′ to 5′ and the base pairs are complementary under this orientation; see Figure 1.



(a) DNA structure [6]

(b) Double helix shape

Figure 1: *DNA structure.* The DNA has a double helix structure.

Specific regions on the DNA which encode proteins are called genes. A gene has three sections: promoter, coding area and termination site. Proteins are synthesised in a two step process, where initially the gene is transcribed and then the transcript is used as a template in the translational process; see Figure 2. The transcription takes place when an RNA polymerase (RNAp) molecule binds to the promoter and moves downstream on the DNA, where it splits the two DNA strands. The RNAp molecule reads one of the strands and creates a complementary, anti-parallel RNA strand called messenger RNA (mRNA). Note, however, that the mRNA includes uracil (U) in all instances where thymine (T) would have occurred. When it reaches the termination site of the gene, the RNAp molecule detaches from the gene and releases the mRNA molecule. The mRNA molecule provides a template for the second step in protein synthesis, the translation, which is carried out by large macromolecular assemblies within the cell [176].

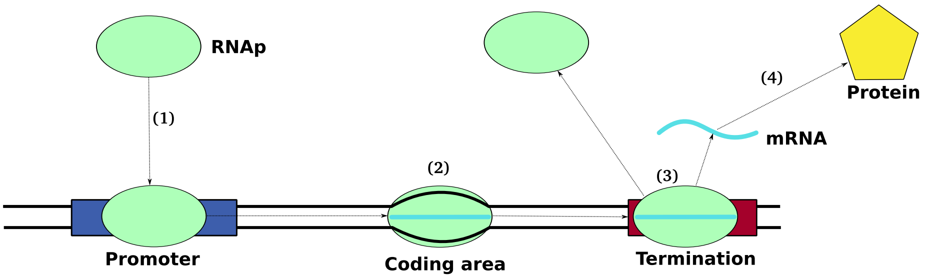
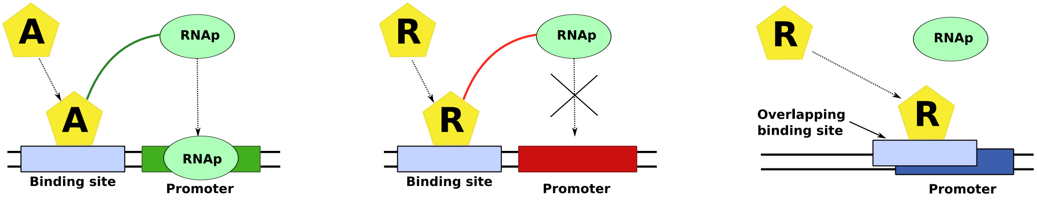


Figure 2: *Protein synthesis.* The RNAp molecule binds to the promoter (1) and copies the information stored in the gene sequence to a new molecule messenger RNA (mRNA) (2). At the termination area, the RNAp unbinds from the gene and releases the mRNA molecule (3), which is translated into the output protein (4).

Gene activity is mediated by site-specific transcription factors (TFs). Their binding to defined regions on the DNA (binding sites) determines the rate at which their target genes are transcribed. TFs control gene activity by either increasing (activators) or reducing (repressors) the transcription rate of their target genes; see Figure 3. Usually, activation is achieved by additional binding sites where regulatory molecules bind and change the affinity between the promoter and the RNAp molecules. Repression is obtained when the binding site overlaps the gene promoter and, thus, obstructs the transcription process [5]. All the binding sites for one or more TFs, which regulate the activity of a gene, are generically called the gene *cis*-regulatory area.



(a) activation (b) repression (c) repression

Figure 3: *Gene regulation.* The activity of a gene can be regulated by transcription factors. (a) Bound activator molecules (A) to the gene binding site can attract RNAp molecules or they can increase the affinity between the gene promoter and the RNAp molecule (green line). (b) The repressor molecules (R) have the opposite effect by repelling RNAp molecules or decreasing the affinity between the promoter and the RNAp molecules (red line). (c) Most commonly, repression is achieved when the binding site overlaps with the gene promoter, thus, blocking the RNAp binding to the promoter.

The gene activity function (or gene regulation) is usually modelled as a hyperbola or a sigmoid function, where the latter (sigmoid function) is achieved when the gene has more than one binding site or when the molecules can bind only in dimers or oligomers (molecules consisting of two similar subunits called monomers) [2, 21, 22, 75, 27, 36]. Note that the gene regulation function has been derived using statistical thermodynamic models [2, 21, 22, 27], chemical kinetics [75] and even Markov chains [36].

The Hill function is traditionally used as the gene regulation function due to the fact that it can capture both the shape of a hyperbola or a sigmoid shapes and that the parameter which controls the steepness (the Hill coefficient) is usually approximated by the number of binding sites (in the case when only monomers regulate a gene),

$$\phi(x) = \frac{x^l}{x^l + K^l} \quad \text{and} \quad \bar{\phi}(x) = \frac{K^l}{x^l + K^l}. \tag{1}$$

The first function, $\phi(x)$, represents the regulation function in the activation case, while the second one, $\bar{\phi}(x)$, represents the regulation function in the repression case. The parameter K is the threshold and represents the regulatory input concentration required for half activation of the gene. The Hill coefficient is denoted by l and quantifies the steepness of the regulation function.

A statistical study on a bacterial genome indicated that a significant number of genes have more than one binding site for the same TF; 37% of the genes in *E.coli* have more than one binding site [78]. In the context of the gene regulation functions, this indicates that many bacterial genes display sigmoid regulation functions. Furthermore, closer examination of the statistical data revealed that several genes display a steep response to regulatory input ($l \geq 4$); see Figure 4.

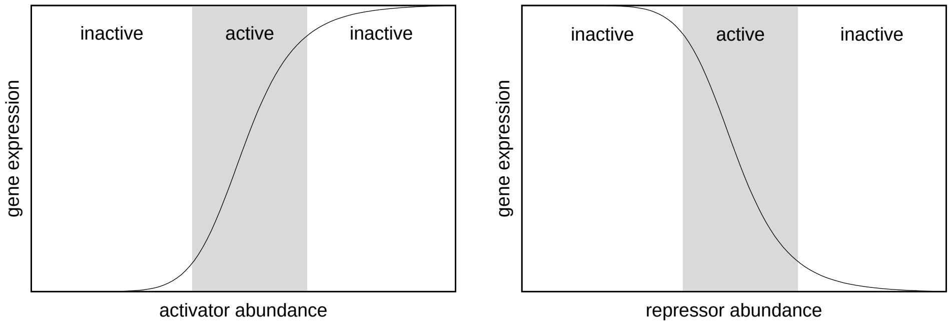


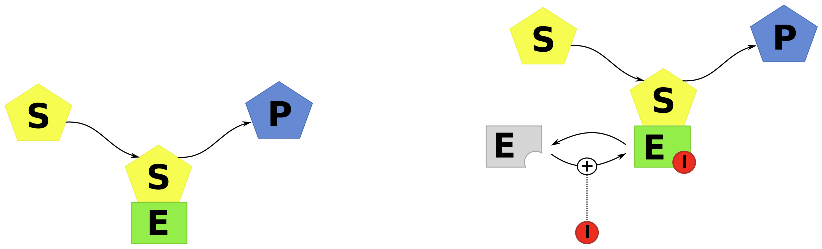
Figure 4: *Hill regulation function.*

In addition, genes are usually regulated by more than one species and, thus, the regulation function of the gene has a more complicated shape with more than one regulatory input. Altogether, we can conclude that genes can display sharp switching behaviour and that they are capable of integrating multiple inputs.

Alternatively to genes, enzymes are also able to process information [29]. An enzyme is a biomolecule, usually a protein, which can catalyse the transformation of a substrate into a product. The main difference between enzymes and other catalysts is that enzymes are selective, meaning that they catalyse only the transformation of some specific substrates and do not affect other species. The transformation mechanism assumes that an enzyme E binds reversibly to a substrate S and then converts irreversibly the substrate into a product P,

$$E + S \rightleftharpoons ES \rightarrow E + P. \qquad (2)$$

Note that this mechanism is usually known as the Michaelis-Menten enzyme kinetics [110], which is also called the 'basic enzyme reaction', e.g. [116].



(a) Enzymes (b) Allosteric enzymes

Figure 5: *Enzymes.* (a) An enzyme can catalyse the transformation of a substrate into a product. (b) Allosteric enzymes are able to catalyze the transformation of substrate only in the active form and this is usually controlled by ligands which bind to the enzyme.

Allosteric enzymes are enzymes that have more than one conformation and a set of uncorrelated binding sites [111, 112]. The binding or unbinding of the inducers to the binding sites determines the current conformation of the enzymes. In particular, inducers can either activate or deactivate an enzyme and only in the active state can the enzyme catalyse the transformation of the substrate. If we consider the inducers to be the inputs of a system, then allosteric enzymes can be viewed as systems able to integrate multiple inputs (if they have multiple binding sites for different inducers) and display a switch-like behaviour (if they have multiple binding sites for the same inducer).

In the following sections, we will use the notions introduced here to review how DNA molecules, allosteric enzymes and genes are able to perform both numerical and logical computations.

2.3 DNA Computing and Hard Combinatorial Problems

Adleman was the first one to use the DNA strands to solve an intensive computational problem, the Hamiltonian path problem (HPP) [3]. This problem identifies whether there is a route which starts from a selected node (start node), passes through all nodes exactly once and ends in another selected node (ending node); see Figure 6. This is a well known to be an NP-complete problem [3].

The algorithm used by Adleman can be summarised in three steps.

1. The input of the problem is first encoded on DNA molecules (encoding step)
2. The DNA molecules are put in a test tube where they form solutions and under appropriate condition certain solutions are selected (computing step)
3. Finally, the DNA molecules encoding solutions to the problem are extracted from the test tube (extraction step).

The speed performance of DNA computing is not impressive. The solution to the HPP in the case of seven nodes (see Figure 6) was obtained after seven days. Nevertheless, the experiment revealed some impressive advantages such as the massive parallelism, energy efficiency (approximately 2×10^{19} *operations/J*) and compact information storage.

The satisfiability problem (SAT) also received great attention in the field of DNA computing [99]. The SAT problem identifies whether there is any combination of the input values of a logical function, which ensure that the function will evaluate TRUE. This is also known to be an NP-complete problem [99]. Any logical function can be written in conjunctive normal form, where clauses are connected by AND operator and each clause contains combination of literals (variables or their negation) connected by OR operator; e.g., $F = (x \vee y) \wedge (\bar{x} \vee \bar{y})$. Writing the logical expression into conjunctive normal form reduces the SAT problem to finding values for which each clause evaluates TRUE.

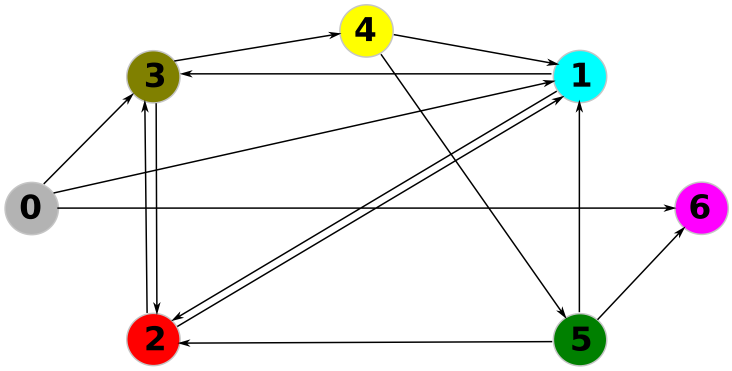


Figure 6: *Hamiltonian path problem.* This graph has a Hamiltonian path starting from node 0 and ending in node 6, $0 \to 1,\ 1 \to 2,\ 2 \to 3,\ 3 \to 4,\ 4 \to 5,\ 5 \to 6$.

Similar to the case of the Hamiltonian path problem the problem is encoded on DNA strands [28, 101], but was also encoded on RNA strands [49]. Solutions that do not meet the satisfiability condition are removed successively by considering each clause individually and, thus, the number of cycles per experiment equals to the number of clauses in the logical formula. Braich *et al.* where able to solve a SAT problem with 20 variables, 24 clauses and 3 literals per clause, which required searching through more than 1 million solutions.

One of the main disadvantages of DNA computing consists of the fact that the method requires a series of manual steps. In the case of the SAT problem the number of manual steps increases linearly with the number of clauses. Sakamoto *et al.* overcame this problem by combining all computation cycles (for all clauses) into one cycle [142]. The solution involves the encoding of literal strings (conjunctions of literals from each clause) into a single stranded DNA sequence (ssDNA). For example $F = (x \vee y) \wedge (\bar{x} \vee \bar{y})$ leads to the following four literal strings $x - \bar{x},\ x - \bar{y},\ y - \bar{x},\ y - \bar{y}$. ssDNAs of literals and their complements will form hairpins which can then be removed. Literal strings that do not have hairpins are valid solutions. Nevertheless, despite the advantage of unassisted experiments, this method assumes higher effort in the initial encoding step and high inefficiency with respect to the used DNA quantity compared with the other methods.

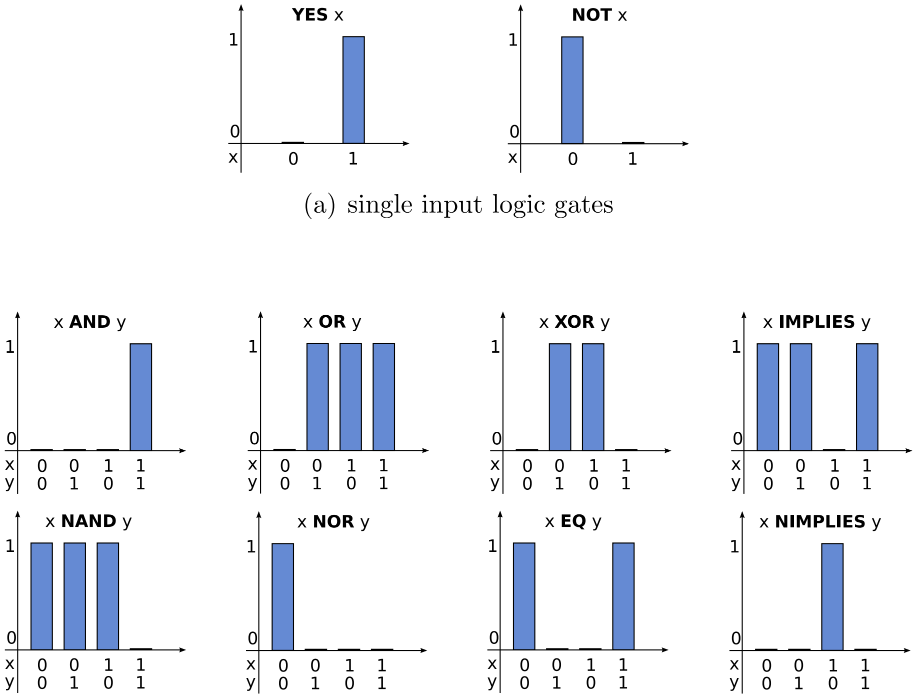
Taking the unassisted process a step forward, Baumgardner *et al.* [18] solved the HPP problem *in vivo* by using the recombination site *hixC* [98], i.e., Hin recombinase inverts DNA fragments which are delimited by *hixC* sites. Although, the implementation addressed a simple 3 node - 3 edge graph, it stands as a proof of concept. The nodes, except the last one, where encoded using fluorescent proteins so that the output could be visualised. This represents the main disadvantage of the method, in the sense that adding more nodes would require new fluorescent proteins and this imposes an upper limit on the number of nodes in the graph. Moreover, even for a small number number of nodes it can be difficult to distinguish between similar colours.

A solution to the *in vivo* scalability problem of the experiment led by Baumgardner *et al.* [18] was proposed by Frisco *et al.* [54]. They suggested that the nodes should be encoded using bacteriophage DNA in such a way that the virus is functional only when a Hamiltonian path exists. In this scenario the existence of a solution can be assessed more easily by verifying whether the virus is present or not (the formation of plaque if we considered the operon for the production of the capsid and tail proteins of the bacteriophage λ).

Despite all these improvements, researchers still raised concerns as to whether hard combinatorial problems are the most suitable application for DNA computing [6, 73]. Hartmanis showed that increasing the size of the HPP problem (by adding more nodes) would require so much DNA that the experiments would become impractical. He computed that a graph with 200 nodes would need a DNA molecule heavier than the Earth [73]. This suggests that DNA based computing might be aimed for other types of problems (not NP complete), such as performing logical computations for example [6].

2.4 Molecular Logic Gates

Logic gates are systems which perform logic operations on one or more inputs and produce a binary output. The inputs and outputs of logic gates are restricted to two discrete logical values: 0 (or false) and 1 (or true). The simplest logic gates transform one input into an output, as seen in Figure 7(a), but, in order to implement more complex logic functions, logic gates with at least two inputs are required; see Figure 7(b).



(a) single input logic gates

(b) double input logic gates

Figure 7: *Boolean Logic.* Note that IMPLIES gate is also known as AND NOT gate or IF gate, while NIMPLIES as NAND NOT or NOT IF.

Biological molecules can be used to implement systems that mimic logic gates. We use the word mimic because the output of the molecular logic gates is continuous while logic gates strictly require an output with only two possible values. Nevertheless, this abstraction (of assuming molecular systems to mimic logic gates) is often required in dealing with larger systems and can also support the design of new synthetic ones.

Next, we will review three mechanisms based on biological molecules which display logic behaviour: (*i*) DNA based, (*ii*) enzymatic and (*iii*) genetic logic gates.

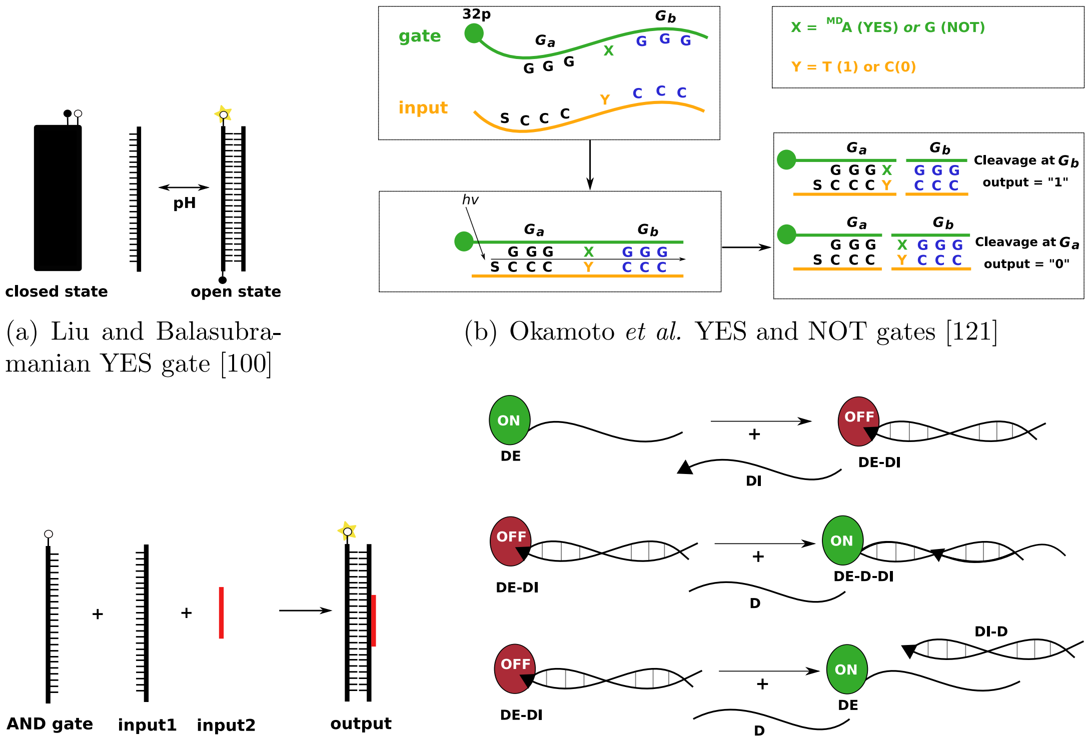
2.4.1 DNA Logic Gates

DNA molecules were used to implement logic functions based on chemical or photonic inputs [34]. A first example of DNA based logic gates is Liu and Balasubramanian's YES gate [100]. Initially a single strand oligonucleotide (X) is in a closed conformation; see Figure 8(a). Increasing the pH levels leads to conformational changes, i.e., the original oligonucleotide (X) attaches to another oligonucleotide (Y) leading to an open state complex. The output is measured by ensuring that a fluorescence marker is turned on only in the open state. The switching speed, although still slow compared with electronic devices, is impressive for molecular gates; the YES gate can be switched in approximately 5 seconds. The input in this gate is the pH level while the output is the conformational state of the DNA.

A different design of logic gates using DNA strands was proposed by Saghatelian *et al.* [141]. They built an AND logic gate consisting of an oligonucleotide which has a fluorescent protein attached at one end; see Figure 8(c). Only in the presence of the complementary oligonucleotide (the first input), a Hoechst protein can bind to the complex and turn on the fluorescent protein of the gate. Note that, this gate consists of three different types of inputs/outputs, an oligonucleotide and a Hoechst protein as inputs and fluorescence as output.

Okamoto *et al.* designed logic gates, and even a full-adder, based on hole transport technology [121]. The logic gates consists of a single DNA strand in which one or more transport bases ($X = {}^{\mathrm{MD}}A$ to allow transport and $X = G$ to repress it) are flanked by two GGG sites (a proximal G_a and a distal one G_b); see Figure 8(b). Note that ${}^{\mathrm{MD}}A$ is the adenine base (A) which was modified to enhance hole transport. The input in the gate consists of another ssDNA which has as complement for the ${}^{\mathrm{MD}}A$ site either thymine (T) or cytosine (C) which code for the logic values ($T \equiv 1$ and $C \equiv 0$). If the input contains thymine then the hole transport is enhanced while in the case of cytosine the hole transport is reduced. The output of the gate is represented by the ratio (G_b/G_a) of the cleavage on the proximal site to the one on the distal site after exposure to photoirradiation. The inputs of these gates are base sequences while the output is the cleavage rate.

A lot of effort has also been invested into designing hybrid logic gates where DNA strands and allosteric enzymes work together to implement the logic behaviour. Gianneschi and Ghadiri [59] constructed logic gates consisting of an enzyme which has attached a single strand DNA molecule (DE); see Figure 8(d). When a DNA strand attached to an enzyme inhibitor (DI) is added together with the gate (DE), the two DNA strands form a single double strand and the inhibitor gets attached to the enzyme, resulting in the inactivation of the latter. The switching off mechanism was achieved by adding single DNA strand that was either able to move the inhibitor further from the enzyme or to replace the $DI-D$ complex from the enzyme and take its place. The inputs of these gates are DNA strands and the output is the state of an enzyme (active or inactive).



(a) Liu and Balasubramanian YES gate [100]

(b) Okamoto *et al.* YES and NOT gates [121]

(c) Saghatelian *et al.* AND gate[141]

(d) Gianneschi and Ghadiri YES and NOT gates [59]

Figure 8: *Non modular DNA logic gates.*

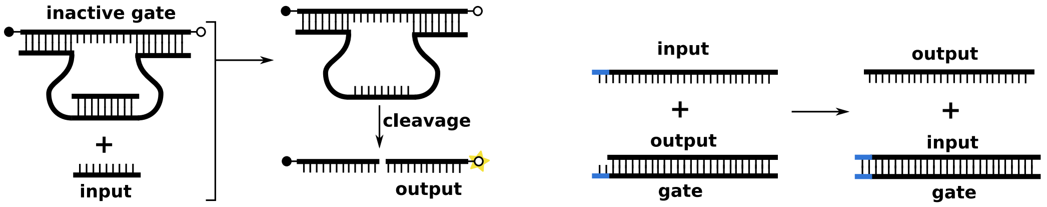
Modularity in design can be achieved when the inputs and the output are of the same type [147]. However, the four gates presented above lack this property and, thus, they do not have a modular design. Below we present two types of logic gates which implement Boolean logic with the help of DNA strands and also have a modular design.

Stojanovic and co-workers [158, 159] implemented logic gates using deoxyribozymes (DNA molecules which are able to catalyse the transformation of other proteins). Their model consists of deoxyribozymes which are able to cleave an oligonucleotide when the former is in an active state. The state of the deoxyribozymes logic gates (active/inactive) is regulated by input oligonucleotides; see Figure 9(a). As opposed to the previous example of logic gates, this design is modular due to the fact that both the inputs and the output are represented by oligonucleotides.

Despite the modularity of the design, the implementation of Stojanovic and co-workers of different logic systems (such as full-adder or tic-tac-toe) is not modular, in the sense that the output of a logic gate is not fed into the input of another [160, 96]. Their approach consists of writing the logical expression for these logic systems in the disjunctive normal form where clauses are connected by the OR operator and each clause contains a combination of literals (input variables or their negation) connected by the AND operator. Each gate implements a clause (AND operations between inputs) which is able to cleave a common output substrate into two oligonucleotides. If at least one gate (clause) is active (true) then the output substrate will be cleaved into two oligonucleotide leading to an OR operation between all clauses.

An interesting aspect observed by the authors during the experiments is that there is a trade-off between the quality of the binary behaviour and the transient time (the time to reach the steady state) which is controlled by the length of the gate, i.e., longer gates lead to better binary behaviour and slower response time while shorter ones lead to poorer binary behaviour but faster response [160].

Seelig *et al.* constructed modular logic gates where the inputs, the output and even the gates themselves are oligonucleotides and the logic mechanism relies only on sequence recognition and strand displacement [147, 150]; see Figure 9(b). A gate consists of a double stranded oligonucleotide except for a small toehold (recognition site) which is not covered. An input oligonucleotide binds to the recognition site of the gate and displaces the output oligonucleotide, taking its place. One of the main disadvantages of these gates is that they are very slow; for example it, takes up to 2 hours for an AND gate to reach half of the steady state.



(a) Stojanovic *et al.* NOT gate [158]

(b) Seelig *et al.* YES gate [150]

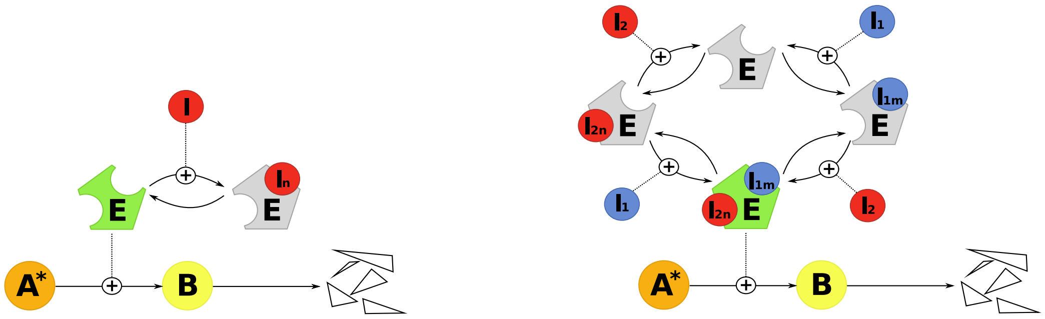
Figure 9: *Modular DNA logic gates.*

Seelig *et al.* [147] observed that, when interconnecting gates in a logic system, the quality of the binary output (which is essential for performing logic functions) is worsened by adding more gates. This suggests that their initial design was not scalable. To address this issue, they constructed an additional gate to restore a binary response, but this comes at the cost of a longer transient time (about 10 hours for a system with 11 gates).

2.4.2 Enzymatic Logic Gates

Alternatively, biologically inspired molecular logic gates can also be constructed from enzymes [10, 29]. Enzymes which mimic the behaviour of logic gates were studied theoretically [80, 81, 79, 10, 29, 170], but also engineered synthetically [43, 14, 15, 120, 132, 114, 109, 189].

In the simplest case, logic gates can be implemented using single enzymes [10, 29]. The enzyme, in its active state, can catalyse the transformation of a protein from state A to state B. In the presence of an inducer protein, I, the enzyme is inactivated and consequently A is not transformed to B. If we assume that inducer I is the input and that protein B is the output then this enzyme mimics the behaviour of a NOT gate. Note that if we consider that, instead of inactivating the enzyme, the inducer activates it, then the system will mimic a YES gate behaviour. Similarly, an AND gate can be constructed by using only one enzyme which is activated only in the presence of two inputs I_1 and I_2; see Figure 10(b).



(a) NOT gate

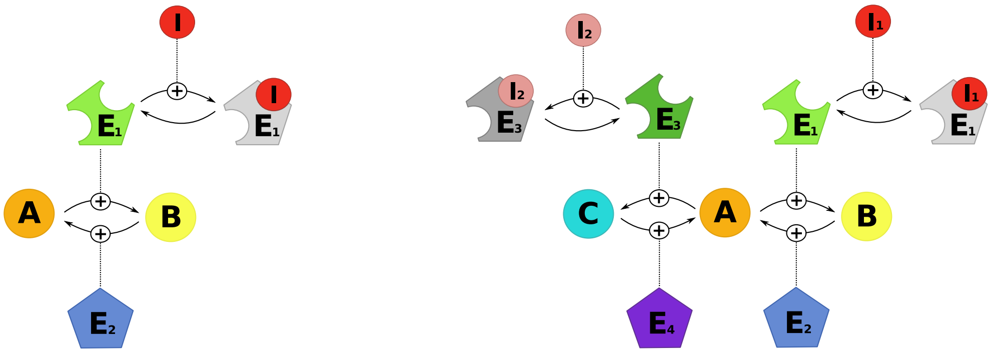
(b) AND gate

Figure 10: *Single-enzyme logic gates.* The inputs are proteins I, I_1 and I_2, while the output by protein B. To activate/inactivate an enzyme, n or m monomers of input are required. Substrate A has a fixed abundance (*) and is transformed by active enzyme E into product B which decays at a fixed rate.

An example of such a single enzyme AND gate is the N-WASP-Arp2/3 system [131]. The N-WASP protein (Neuronal Wiskott-Aldrich Syndrome Protein) can be activated individually by Cdc42 (cell division control protein 42 homolog) or PIP2 (phosphatidylinositol $4,5$-bisphosphate) proteins, but due to the masking of their binding sites individual binding is weak. Nevertheless, in the presence of both Cdc42 and PIP2, the protein N-WASP is strongly activated and regulates the polymerization of Arp2/3 protein.

Boolean logic requires that the output is limited to two values 0 and 1 while biologically inspired logic gates produce a continuous output. Despite this fact, the enzymatic gates can mimic logic gates fairly well, but the quality of this binary behaviour is constrained by the steepness of the output in the sense that a steeper function produces a better binary behaviour. To achieve a steep response from these single enzyme gates, we need to assume that the enzyme can be toggled by more than one monomer and this can be implemented in two ways: (*i*) only dimers or oligomers are allowed to bind to the enzyme or (*ii*) the enzyme has a number of binding sites where input molecules can bind. However, these mechanisms come at the cost of an increase in the size of the enzymes; for example high steepness can be achieved using an enzyme with many binding sites which would imply a very large enzyme.

An alternative method to achieve high steepness consists of gates formed from multiple enzymes [68, 80, 81, 79, 10]. This model (also known as the Goldbeter-Koshland model) assumes that a protein can exist in two states (A and B) and that two external factors, usually two enzymes (E_1 and E_2), can change the protein from one state to the other and vice versa; see Figure 11(a). The steepness of this model is controlled by the kinetic parameters of the system and can be fine-tuned to display high steepness [68, 10].



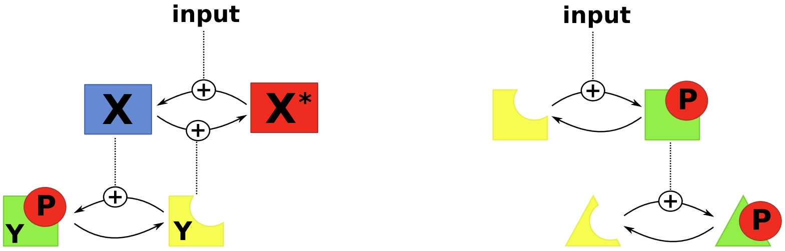
(a) NOT gate

(b) AND gate

Figure 11: *Multi-enzyme logic gates.* (a)E_1 in its active state transforms protein A into B, while enzyme E_2 catalyses the reverse transformation. In addition E_1 is activated by an inducer protein I. The input of the system is represented by protein I, while the output by protein B. (b) A protein exists in three states A, B and C. Enzymes E_1 and E_3, in their active states, catalyse the transformation of A into B and C respectively. These two enzymes are inactivated by two inducer proteins I_1 and I_2. Additionally, two more enzymes, E_2 and E_4, catalyse the reverse transformation from B or C into A.

In zero order kinetics (the total concentration of the enzyme is much smaller than the total concentration of the protein), the multi-enzyme system displays ultrasensitivity, i.e., very steep transition between the two binary states. Nevertheless, the assumption of zero order kinetics does not always hold in real protein interaction systems. A solution to this limitation was proposed by Xing and Chen [181] who added intermediary chemical reactions with higher stoichiometry [40] into the model, resulting in first order kinetics ultrasensitivity.

The binary behaviour of these multi-enzyme gates can be enhanced even further by using different interaction mechanisms, such as positive feedback [51]. Tyson and co-workers modelled an enzymatic toggle switch formed from two proteins which enhance each other's transformations [170, 140]. Their analysis revealed that the bi-stable behaviour can be achieved by adding intermediary chemical reactions with higher stoichiometry [40].



(a) enzymatic toggle switch [170, 140]

(b) enzymatic cascade

Figure 12: *Enzymatic cascades.* (a) A protein X can enhance the phosphorylation of another protein Y^P, while this other protein when unphosporylated can increase the decay rate of the first protein. (b) A modular design of covalent modification based gates using the phosphorylation-dephosphorylation cycle. This process assumes the addition and removal of the PO_4 group to/from a protein. The phosporylated protein in the first module acts as an enzyme to catalyse phosphorylation in the second module.

A multi-enzyme AND logic gate can be constructed from a protein which can exists in three states (A, B, C) and the reversible transformation between these states is mediated by four enzymes; see Figure 11(b). An example of naturally occurring multi-enzyme AND gate is the Hexose-phosphate Interconversion Pathway in glycolysis/gluconeogenesis pathway. Under the control of Citrate1 (cytosolic citrate) and cAMP (cyclic adenosine monophosphate) the F6P protein (fructose 6-phosphate) can be converted reversibly into either F16BP (fructose $1,6$ biphosphate) or F26BP (fructose $2,6$-biphosphate). The theoretical investigation of this system revealed that the concentration of F6P as a response to the concentration of Citrate1 and cAMP mimics the behaviour of an AND gate [10].

Researchers also focussed on interconnecting these enzymatic logic gates and one solution was to assume that the modified protein is either an enzyme or an activator/inactivator of another enzyme [10, 29]. An example of such modular covalent modification system is the phosphorylation-dephosphorylation cycle, where the phosphorylated protein from a cycle catalyses the phosphorylation of another protein [29]; see Figure 12(b). In addition, the interconnection step also requires that inputs of the downstream gate should have their transient region spanned between the two output values of the upstream one [10]. Two methods were proposed to deal with this type of problem: (*i*) constructing an amplification gate able to alter the output of the upstream gate [10, 189] or (*ii*) adding new intermediary reactions with higher stoichiometry and adapting the kinetic parameters [103]. The former comes at the cost of constructing a new gate which will increase the number of required enzymes while the latter might not be viable to be engineered synthetically.

So far, we presented only theoretical studies on enzymatic networks that mimic logic behaviour, but biologists also managed to engineer synthetic logic gates using enzymes. Deonarine *et al.* [43] implemented an enzymatic logic gate based on protein folding, similar to the DNA based logic gate of Liu and Balasubramanian. The *cytochrom* c protein gets unfolded in the presence of various inputs (such as acid, base or urea) and in the unfolded state it expresses high fluorescence intensity. These gates are generically called *chemophotonic* gates because they take chemical substances as inputs and produce an optical output. The fact that the model displays different types of inputs and output (chemical and photonic) indicates the lack of modularity in the design. In addition to this lack of modularity, the gate also needs an additional dialysis step so that the gate can be reset. Nevertheless, this enzymatic gate is interesting due to the fact that it implements multifunction logic gates, i.e., an additional chemical input can control the performed logical function [43].

The modularity in the design of the enzymatic logic gates can be ensured when the inputs and the outputs are of the same type, in our case molecules. Willner and Katz groups have engineered several two-input enzymatic logic gates (AND, OR, XOR, NIMPLIES, NOR, NAND) experimentally [15, 14, 120, 114, 132, 189, 109]. To prove the modularity of the design, these gates were also cascaded and displayed the expected logical behaviour [120, 189].

In the case of digital electronic circuits, input and output signals are transmitted on wires which simplifies the interconnection between gates by connecting the output of one gate to the input of another. Nevertheless, in the case of molecular computations, different signals (molecules) flow together in a single compartment which may lead to crosstalk between signals, i.e., signal proteins can interact with each other or with other proteins resulting in an undesirable change of signals [41]. Previous experimental results proved that more than one logic gate can be engineered in a single compartment by careful selection of the proteins and enzymes for each logic gate and by numerous optimisation experiments aimed to select appropriate concentrations of substances [14, 120, 189].

The effort to select the chemical species in enzymatic logic gates seems greater compared with the case of DNA logic gates, because the latter requires the construction of a specific oligonucleotide able to interact only with a specific gate which is an automated process. In this context, Zhou *et al.* [189] constructed a gate able to convert an output signal into the form of input signals. This procedure ensures that enzymatic gates can be interconnected easier, but only if they are put in separate compartments or otherwise the output signal could interfere with the input one.

In addition to crosstalk, enzymatic logic gates can also be affected by noise, which can become a significant problem in the case of interconnected logic gates. Privman *et al.* [132] engineered an enzymatic AND gate and they modelled how the noise propagates through a cascade of gates. The results indicated that, even in the case of a small noise in the inputs (5%), the original design allowed only two gates to be connected without the output becoming too noisy to be read (the low and the high concentrations become undistinguishable due to random fluctuations). A further theoretical investigation optimised the system parameters to such an extent that up to ten gates could be interconnected while keeping the last gate output readable [132].

Enzymatic logic gates were advertised as the fast solution for bio-molecular logic gates (the transient time is in the range of a tenth of a second [29, 41]). This would represent a great advantage compared with other types of biological logic gates which are significantly much slower (such as DNA or gene regulatory logic gates which have a transient time in the order of tens of minutes). However, the experimental implementations seem to indicate that these theoretical limits are not always easy to reach when engineering synthetic systems and that speeds can vary in the range of tens of minutes [43, 14, 15, 120, 114, 189]. In addition, the complexity of the enzymatic gates relies in the complexity of the enzyme, which makes it difficult to automate the construction of logic gates. An alternative to these enzymatic logic gates are the genetic logic gates which are described in the next section.

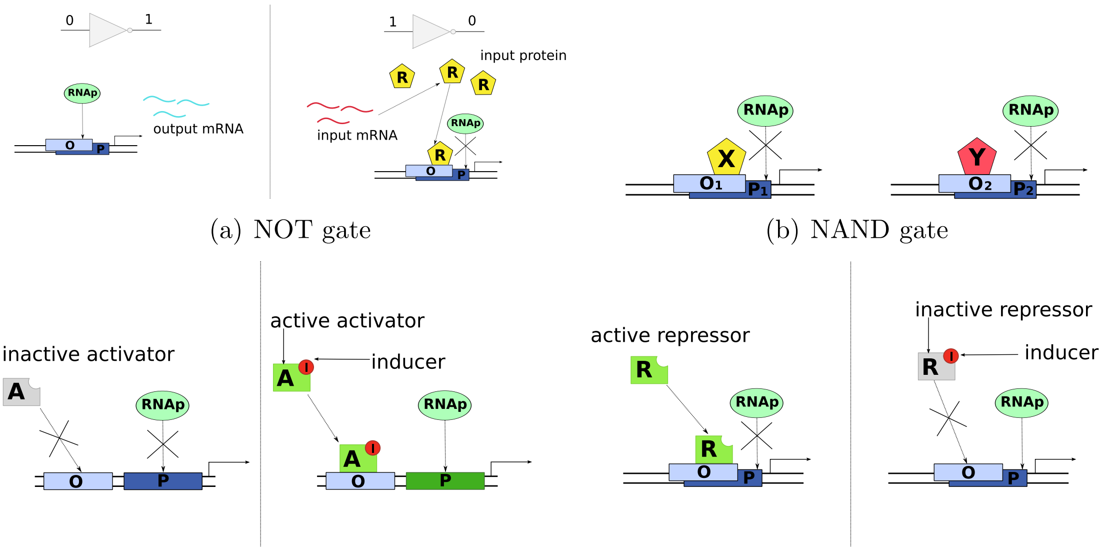
2.4.3 Genetic Logic Gates

The development in genetics allowed researchers to modify current genes in live bacterial cells and even add new ones [175]. This opened the possibility to engineer genetic systems able to implement various functions, including logical computations. In a genetic system, transcription factors (the inputs) control the rate at which a gene (the gate) expresses the product protein (the output). Due to the fact that both the inputs and the output are of the same type, namely proteins, the system has a modular design [175]. These signals (inputs and output) can be quantified by the concentration of the relevant proteins. Although protein concentration is a continuous variable, in the case of a sigmoid regulation function the behaviour of the gene can be approximated by the binary form which indicates whether a gene is ON or OFF [32]. This abstraction is a useful approach to understand better current systems, but also to help engineer new ones.

Interestingly, genetic logic gates display a built-in automatic reset mechanism, by having the input/output mRNAs and proteins decayed (by active degradation or dilution) so that signals that are not sustained will have a finite lifetime. We note that this is an essential mechanism, which enzymatic or DNA logic gates do not possess, due to the fact that it allows new inputs to be processed without external intervention; for example, in the case of the enzymatic logic gates designed by Deonarine *et al.*, the researchers had to perform dialysis to reset the gate [43].

The simplest logic gates are the single input gates, YES and NOT. Weiss *et al.* [175] proposed the design of a NOT gate by using a repressor gene; see Figure 13(a). The input in this gate is the mRNA template of the protein which regulates the gene. This input is amplified by translation and converted into the repressor protein, which binds in dimers and in a cooperative manner (to enhance step-like behaviour) to the gene to repress it. In the presence of the input mRNA, the gene is repressed and not transcribed any more, resulting in low output mRNA concentration, while in the absence of the input mRNA the gene is transcribed and produces output mRNA. This system mimics the behaviour of a NOT gate. The quality of this NOT-like behaviour depends on the steepness of the gate in the sense that the NOT-like behaviour can be enhanced by increasing the steepness of the regulation function.

In addition to these single input logic gates, Weiss *et al.* [175] also proposed a design for two input logic gates such as NAND, AND and IMPLIES. Note that the NAND gate represents a functionally complete set in the sense that any logical function can be built using only NAND gates. The NAND gate modelled by Weiss *et al.* [175] consists of two genes repressed by two different inputs (X and Y), but which encode the same output protein. This output protein is synthesised when at most one of the two inputs (X or Y) is present so that at least one gene remains active; see Figure 13(b). The mechanism of having two genes encoding



(a) NOT gate

(b) NAND gate

(c) AND gate

(d) IMPLIES gate

Figure 13: *Single-regulator genetic logic gates* [175]. The operator regions where TFs bind are denoted by O, while the promoters of the genes by P. Repression is represented by the overlapping between the operator and promoter. The inputs are the TFs which regulate the genes. For AND and IMPLIES gates, the inputs are the TFs (A or R) and the inducer which activates/inactivates the TFs. Two inputs logic gates can be designed by either using OR wiring between two genes (b) or by using an allosteric enzyme and an inducer as the two inputs, where the enzyme can regulate the gene only in the active state, (c) and (d).

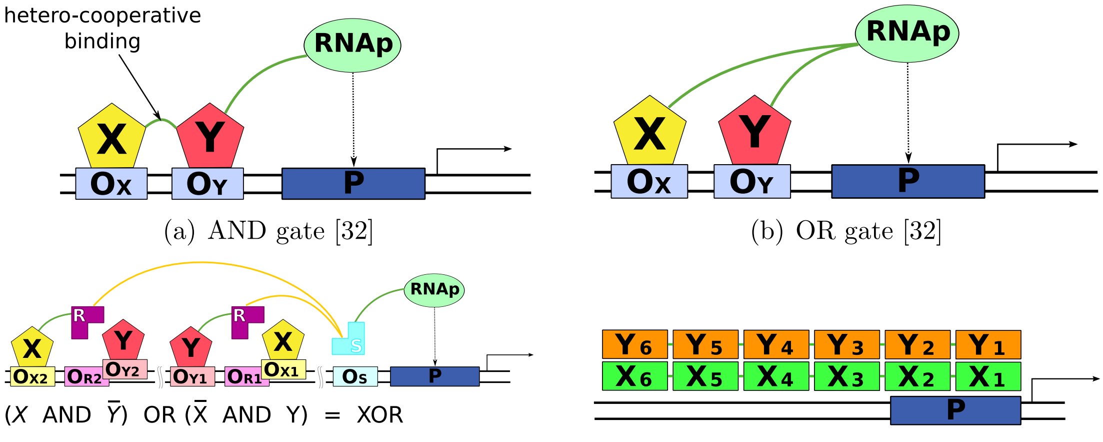
for the same protein is called OR-wiring.

In an OR wiring mechanism, a high state can be achieved when one of the repressor or none are present. In the case of one repressor being present, only one gene is expressed while, when none are present, both are expressed. Thus, there is an approximate two-fold difference between concentration values encoding the same logic value. This is undesirable and may lead to further problems when logic gates are interconnected.

Alternatively, two input logic gates which use only one gene can be constructed by assuming that the transcription factor (TF) displays two states (active and inactive) and only in the active state it can regulate the gene. In addition, the activity of the TF is controlled by inducer molecules, i.e., inducers can either activate or inactivate the TF. In the case when the inducer activates an inactive gene regulator and this regulator increases the gene transcription rate, then the gene is transcribed only in the presence of both the regulator and the inducer, resulting in an AND behaviour; see Figure 13(c). This approach suffers from the lack of interchangeability between the inputs. For example the IMPLIES gate is turned off only when the first input is present and the second one is absent, but the regulator/inducer mechanism can be applied only by considering the repressor to be the first input and the inducer the second one and not vice versa; see Figure 13(d).

Buchler *et al.* [32] proposed the implementation of logic gates using combinatorial signal integration at the *cis*-regulatory transcription control level. In particular, they addressed the design of logic gates through the regulated recruitment of transcription factors (TFs) and RNAp without involving complex protein-protein interactions such as allosteric enzymes. These gates are called transcriptional logic gates and they represent the main focus of this thesis. Two TFs and two operator zones can be used to construct simple two-input logic gates, such as AND and OR. For example the OR gate was constructed from a gene which is turned on when at least one of the operator sites is occupied. The operator sites of the OR gate permit only the binding of specific TFs and this binding occurs independently, in the sense that the binding of one TF does not influence the binding of the other one; see Figure 14(b).

The AND gate has a similar design to the OR one but requires that TFs bind in a hetero-cooperative way. This means that the TFs have low affinity to bind alone, but together their binding affinity increases; see Figure 14(a). Hetero-cooperativity can be found in some wild type systems, but it can also be engineered in a modular fashion. The modular approach uses scaffold proteins (proteins able to colocalize other proteins close to them) to bind to DNA sites and recruit RNAp molecules to initiate transcription [143].



(a) AND gate [32]

(b) OR gate [32]

(c) XOR gate [32]

(d) NOR gate [78]

Figure 14: *Transcriptional logic gates.* X and Y molecules can bind to their operator sites upstream of the gene and either attract or repeal RNAp molecules to initiate transcription. The operator sites can consist of one or more identical binding sites for the same TF and the binding to these binding sites takes place is a homo-cooperative manner. Note that in (d) we explicitly represented each individual binding site of the operator. (a) Hetero-cooperativity is used to implement AND logical behaviour, where this mechanism assumes that the two TFs can bind with high affinity only together. (b) OR logic gates are implemented using an independent binding motif, where the binding of one TF does not influence the binding of the other one. (c) The XOR gate was implemented using a modular design. There are two distal sites where R subunits can bind depending on the presence or absence of TFs and, once bound, these R subunits recruit an S subunit and initiate a DNA looping, resulting in the activation/repression of the gene. (d) The NOR gate was designed using competitive binding of two TFs to the promoter, thus, blocking the binding of RNAp. Homo-cooperativity is also used to achieve a steep response.

More complex gates, such as XOR gate, can be constructed by connecting these elementary gates. However, this leads to slower, noisier and more metabolically expensive systems [32]. A different approach consists of using OR wiring for two genes where, in the case of the XOR gate, each gene is activated in the presence of one TF and repressed in the presence of the other TF. As mentioned above, OR wiring has the disadvantage of coding a logical value with different concentrations. Buchler *et al.* [32] proposed a modular mechanism for designing complex logic gates which avoided crowding and used only one gene. This mechanism consists of transforming the logical function in either disjunctive or conjunctive normal forms and using hetero-dimers subunits and DNA looping. Based on the presence or absence of the TF, an R subunit can attach at a distal site and attract another subunit, S, there. Through DNA looping the S subunit can bind to the gene promoter and repress the gene or to a nearby site to activate it; see Figure 14(c). This method also avoids crowding of binding sites for more complex functions and eliminates the need for more than one gene [32]. Nevertheless, this method can suffer from crosstalk between different R and S subunits which would limit the number of gates within a cell [32].

As opposed to Buchler *et al.*'s [32] rational design of transcriptional logic gates, Hermsen *et al.* [78] designed *cis*-regulatory regions *in silico*, able to perform various logic functions, by using evolutionary algorithms. In addition to hetero-cooperativity, they observed a high occurrence of homo-cooperativity and overlapping binding sites; for example, in *E.coli*, 37% of the TF-DNA interactions are mediated by more than one binding site and 39% of the binding sites overlap. Homo-cooperativity had the main purpose of increasing the steepness of the switching mechanism, while the overlapping binding sites of implementing an efficient repression mechanism; see Figure 14(d).

TFs seem to use a variety of binding mechanisms, such as hetero-cooperative, competitive or independent binding. Usually it was observed that hetero-cooperativity was associated with AND/NAND behaviours while competitive binding with OR/NOR behaviours [78, 146]. Schilstra and Nehaniv [146] compared, from the point of view of Boolean logic, all these TF-DNA binding mechanisms using statistical thermodynamics. Their results indicated that, although all these mechanisms can lead to behaviour that can be interpreted by logic functions, only independent binding strictly follows the rules of Boolean logic. According to De Morgan's theorems, the rules to write AND and OR logic operators in terms of each other using negation are the following

$$\overline{x} \vee \overline{y} = \overline{x \wedge y} \quad \text{and} \quad \overline{x} \wedge \overline{y} = \overline{x \vee y}$$

Only in the case of independent binding, the steady state abundance which encodes for $\overline{x} \vee \overline{y}$ equals the one that encodes for $\overline{x \wedge y}$. In the case of other mechanisms (such as hetero-cooperative and competitive binding) the steady state abundances of the genes will have similar, but not necessarily equal, values. Nevertheless, the behaviour of these genes can still be interpreted using logic functions. In this thesis, we will consider genes which can be interpreted as displaying a Boolean logical behaviour and we will say that these genes mimic the behaviour of logic gates.

We presented so far only theoretical studies aimed to identify and propose mechanisms to implement logic gates using genes. In addition to these theoretical studies, there was also a lot of interest to engineer logic gates, which use the above mentioned mechanisms, synthetically. In the remaining part of the section we will review logic gates which were engineered synthetically within live cells. A classic example of a genetic system which exhibits logical behaviour is the lac operon in *E.coli*. The genes associated with the lactose metabolism (*lacY*, *lacZ* and *lacA*) are transcribed at a high rate only in the absence of the lac repressor (LacI) and the presence of CRP; see Figure 15. Alon group [148] showed that using two inducers as inputs, IPTG to repress lacI and cAMP to activate CRP, the system functions as an intermediate logic gate between AND and OR gates, in the sense that output concentrations for input logical values of $(0, 1)$ and $(1, 0)$ are neither low nor high (they are halfway between high and low concentrations). An AND behaviour can be enhanced by reducing the concentration of the intermediary states, while an OR one by increasing these concentrations.

In addition to this theoretical study, Alon group [105] changed the behaviour of the lac operon in pure AND, pure OR and single input switches (only one input controls the system) synthetically, by performing a few point mutations in the *cis*-regulatory area. Although the behaviour of the system is changeable, the *cis*-regulatory area is *plastic*, meaning that the gene is able to perform new functions without altering its essential features. This fine-tuning of behaviour and parameters represents one of the most important advantages of the genetic logic gates compared with enzymatic ones. Note that, although this model of lactose operon uses both transcription factors and inducers, the logic function is integrated in the *cis*-regulatory area as modelled by Buchler *et al.* [32].

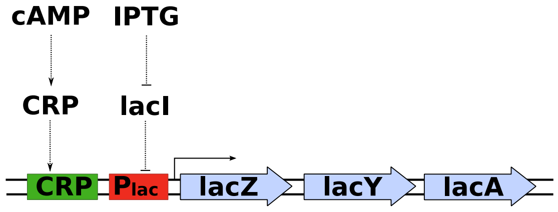
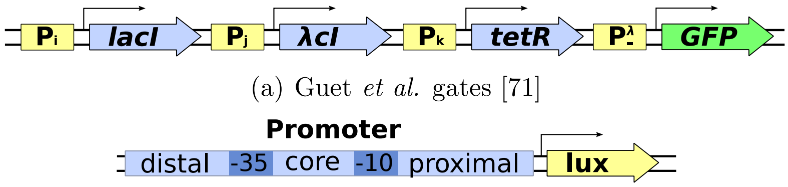


Figure 15: *The lac operon can mimic an AND gate.* IPTG acts as a derepressor for the P_{lac} and it can be considered to be an activator for the system. The operon is transcribed with high rate only when both inputs, IPTG and cAMP, are present, thus, resulting in an AND behaviour.

A different approach was used by Guet *et al.* [71] who constructed logic gates by considering various network topologies of a three genes system (*lacI*, λcI and *tetR*), where each gene is regulated by only one TF. The activity of each gene can be controlled by one of the five promoters: P_1^L and P_1^L repressed by LacI, P^T repressed by TetR and P_-^λ and P_+^λ repressed and activated respectively by cI; see Figure 16(a). The input in the system consists of two inducers, IPTG (able to inactivate LacI) and aTC (able to inactivate TetR repressor), while the output is a gene encoding the green fluorescence protein (GFP) and controlled by a P_-^λ promoter. The system, formed of four genes connected in different configurations, performed various logic operations (such as NAND, NIMPLES and NOR) in response to the two inducer inputs. In most cases, the output was unambiguous and displayed a clear binary form. This experiment proved that, using a basic tool kit of regulatory elements, one can generate a wide variety of logic behaviours just by changing the connectivity of the genes.



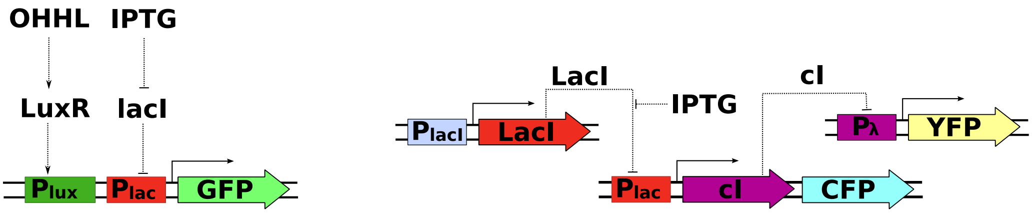
(a) Guet *et al.* gates [71]

(b) Cox III *et al.* gates [39]

Figure 16: *Combinatorial approach on transcriptional logic gates.* (a) Guet at al. designed logic gates using a combinatorial network architecture by changing the regulatory connections between three genes. Each of the three promoters (P_i, P_j and P_k) can be one of the following: P_1^L, P_1^L, P^T, P_-^λ and P_+^λ. Note that P_1^L and P_1^L are repressed by LacI, P^T by TetR and P_-^λ by cI, while P_+^λ is activated by cI. (b) A logic gate consists of a promoter with three sites (distal, core and proximal) where TFs (AraC, LuxR, LacI and TetR) can bind to enhance or inhibit the transcription rate. The RNAp molecules bind on the core area, between -35 and -10 boxes, to initiate transcription. Repression can be achieved at any site while activation only at the distal site.

A combinatorial approach was also employed in designing promoters; see Figure 16(b). Cox III *et al.* [39] constructed 288 promoters in *E.coli*, each being regulated by up to three TFs. Four proteins were used as TFs: two activators (AraC and LuxR) and two repressors (LacI and TetR). The states of these inputs were toggled by four inducers Lara, VAI, IPTG and aTc to activate/inactivate AraC, LuxR, LacI and TetR respectively. The results showed that genes mimicked the behaviour of three logic gates, YES, NOT and AND. In addition, Cox III *et al.* [39] observed that repressor gates display a better binary behaviour than activator gates and that gates with more inputs display a poorer binary behaviour compared with the ones with fewer inputs.

Alternatively to combinatorial design, researchers have also used rational approaches to engineer genetic logic gates. Anderson *et al.* constructed an AND gate with three genes where the two inputs of the gate were integrated through translation regulation [8]. Sayut *et al.* [145] designed an AND gate but this time using one gene which integrates the two inputs in the *cis*-regulatory area as previously proposed by Buchler *et al.* [32]; see Figure 17(a). As opposed to Alon group [148, 105] who exploited an already existing promoter (the lac operon), Sayut et al [145] fused together two *cis*-regulatory areas synthetically, one regulated by LuxI and the other one by LacI. This proved that complexity in transcription gates can be constructed combinatorially in the *cis*-regulatory area by adding the appropriate regulatory sequences. Nevertheless, combining two regulatory areas is not always easy and can lead to a behaviour which is undesirable in the design of synthetic logic gates. For example, in the case of the lac operon, the behaviour was an intermediary one between AND and OR and not a clear logical behaviour [148, 105].



(a) Sayut et al [145]

(b) Yokobayashi *et al.* [182]

Figure 17: *Rational design of transcriptional logic gates.* (a) The gene which expresses green fluorescence protein is controlled by two regulatory proteins: LuxR which activates the gene at a distal site and LacI which represses the gene at a proximal site. The inputs in this gate are represented by two inducer proteins, OHHL which activates LuxR protein and IPTG which inactivates LacI protein. (b) The system consists of two gates connected serially: an IMPLIES gate which has an output that feeds into a NOT gate. The IMPLIES gate is implemented by the P_{lac} promoter, where the two inputs are IPTG and LacI and the output is the cI. The second gate, the NOT gate, takes cI as input and produces yellow fluorescence protein as output.

After building basic toolboxes with logic gates, researchers have also focussed on constructing larger logical circuits, where logic gates are interconnected in a network with the aim of performing complex functions. Logic gate interconnection is not simple even in the case of a modular design where the inputs and the outputs are of the same type. One of the problems which can be encountered is the mismatch of parameters, where, for example the lowest output value of a gate can be higher than the regulation threshold of the downstream one. In this case, changes in the output of the first gate will not reflect in any output change of the downstream gate.

Yokobayashi *et al.* [182] used both a rational and an evolutionary approach to tune the regulation function of a gene by performing point mutations limited to specific areas on the DNA (such as the promoter or the coding area of a gene). They were able to transform a non-functional circuit (IMPLIES-NOT gate) into a functional one by altering protein-DNA interactions, but also protein-protein interactions; see Figure 17(b). This underlines that evolvability offers another degree of freedom to fine tune or optimise transcriptional logic gates [182, 145], and even supports the construction of new logic functions for a given genetic system [105].

The transcriptional logic gates presented above used simple transcription factors as signals, e.g., LacI, TetR, cI, LuxR etc. Nevertheless, the pool of available signals is not restricted to these TFs, but can be extended to include other interaction mechanisms such as chemical complementation, that is a linker molecule is used to attach a DNA binding domain (which is able to bind to the DNA) to an activation domain (which recruits RNAp molecules and helps initiate transcription). Bronson *et al.* [30] showed that a gene regulated by this TF (linker molecule, DNA binding domain and activation domain) can implement an AND logic gate with three inputs, where the inputs are represented by the three components of the TF. In this model, the gene is activated and is transcribed at high rate only when all three components are present.

As a final remark, we would like to pinpoint that these types of logic gates are not restricted to prokaryotic organisms as it might be inferred from most of the examples listed above. Kramer *et al.* [93] showed that similar mechanisms as in prokaryotic cells (such as promotor integrated logic or logic gates constructed from multiple genes) can be engineered in mammalian cells to build synthetic logic gates. Rinaudo *et al.* [136] proposed and validated experimentally a general framework for building logic systems in mammalian cells (by assuming the conjunctive or disjunctive normal form of the logic function) using oligonucleotides as gene regulatory inputs. These examples indicate that mechanisms modelled for prokaryotic cells can be adapted and used in mammalian cells.

Considerations on Transcriptional Logic Gates

Above, we presented several mechanisms by which genes integrate multiple inputs, namely: transcriptional integration, OR wiring, allosteric transcription factors or translational regulation. Nevertheless, in this thesis, we address only transcriptional logic gates, which are logic gates constructed from genes that integrate the inputs in the *cis*-regulatory area of the gene.

Transcription based logic gates have several advantages compared with other types of biological logic gates. First, due to the fact that these gates can function only inside cells (they require the protein synthesis machinery), the gates display an automatic reset mechanism using the decay process, i.e., a logic gate is reset when a signal is no longer fed into the system by decaying the proteins in the cell [175].

Moreover, the complexity of the logic function performed by an allosteric enzyme resides in the complexity of the enzyme, but in the case of transcriptional logic gates, this complexity results from the combination of elementary parts in the *cis*-regulatory area [32, 9, 145]. Thus, by combining elementary regulatory regions, usually in the upstream region of the gene, more complex logic functions can be achieved and this process can be automated [32].

Combining different regulatory regions is not a simple process and can lead to undesirable results (see the case of the lac operon where the gene can show an intermediary behaviour between AND and OR logic functions). Nevertheless, genetic logic gates can be evolved to display a desired behaviour by changing the logic function or the kinetic parameters [182, 105, 145]. In the case of allosteric enzymes, one usually has to modify the concentrations of the substances to change the behaviour of the gates [120], but in the case of transcription logic gates the change in behaviour can be achieved by mutations in the base pairs of the *cis*-regulatory area [182]. Hence, transcription based logic gates display a higher degree of freedom in terms of control parameters compared with enzymatic logic gates.

Despite the advantages mentioned above, the design of transcriptional logic gates may be hampered by the specificity of their environment, the cell. In transcriptional logic circuits, the signals are separated by encoding different proteins rather than spatially, as in the case of electronic circuits. As a result, all molecular based systems implemented in a single compartment can suffer from cross-talk between signals. In the case of genetic gates, crosstalk can be avoided only by a careful selection of the TFs and by limiting the total number of TF molecules within the cell. Buchler *et al.* [32] approximated the total number of TF molecules to be around 10^4, which, in the case of 100 TFs (signals) in a system, it indicates that each signal contains around 100 molecules. Usually, noise is reduced by increasing the particle number in the cell [87] and, this suggests that there is a trade-off between noise and number of signals in the system if we require no or negligible cross-talk between signals.

This chapter presents several implementations of elementary biological components able to mimic the behaviour of logic gates. Nevertheless, researchers also focussed on the design and analysis of other digital components from genes, such as toggle switches [35, 57, 91], oscillators [47, 188], pulse generators [17], associative memory units [55, 50] and even counters [52]. The community is now concentrating their efforts to build a library of elementary biological components [9, 172] which can be used in constructing more complex functions such as bio-computers in smart drug delivery systems or biochemical sensors. One example is BioBrick (see *http://partsregistry.org*), a project which aims to build a catalogue of standardised biological parts that mimic the behaviour of digital components. Once these libraries and various standards are completed, it is crucial to interconnect and optimise these basic components [74].

In this thesis, we investigate genes as computational units. In particular, we are interested in how genes can compute and what properties they have. Previous research addressed this, but certain critical aspects were overlooked, such as how genes can be interconnected and what is their optimal design [74]. To overcome these limitations, we analysed both the interconnection and optimality of logic gates by examining how these two are influenced by the biological parameters of the genes.

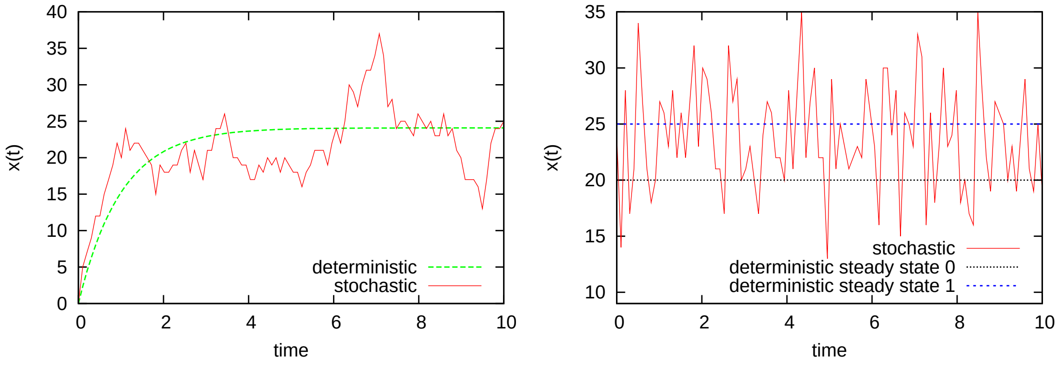
When interconnecting molecular logic gates, researchers have pinpointed that there are two aspects which need to be addressed: (*i*) modularity in parameters [10, 160, 182] and (*ii*) scalability of the design [103]. The former requires that the threshold of a downstream gate to be bounded by the low and the high steady states of an upstream gate, so that changes in the upstream gate can be reflected in the downstream one. This issue was addressed theoretically and experimentally in the case of both enzymatic logic gates and DNA based ones [10, 147]. Yokobayashi *et al.* [182] investigated the modularity in parameters in the context of transcriptional logic gates, but from an experimental point of view (they performed direct evolution and then selected the gates which displayed modularity in parameters).

The second aspect, the scalability of the design, assumes that an arbitrary number of logic gates can be interconnected without affecting the binary behaviour of the system. This was approached both theoretically [103] and experimentally [147, 189] in enzymatic and DNA based logic gates, but, to our knowledge, was not considered in the case of genetic logic gates. In this contribution, we systematically investigate the set of parameters which satisfy these two properties (modularity in parameters and scalability of the design) simultaneously.

Next, we review several studies that analysed and optimised genes, with respect to various properties which characterise their computational behaviour.

2.5 Computational Properties of Transcriptional Logic Gates

In the case of a low number of molecules, inherent fluctuations in reaction rates, caused by thermal noise, induce stochastic fluctuations (noise) in the copy number of molecules [97, 31]. Usually, in living cells, there are few copies of mRNA molecules, and one or two copies per gene. Consequently, the protein synthesis process is affected by noise. In this case, the deterministic analysis is no longer sufficient to describe a reaction system adequately and a stochastic analysis is required; compare green dotted line (deterministic behaviour) to the red solid one (stochastic behaviour) in Figure 18(a).



(a) Stochastic and deterministic behaviour (b) Noise and binary behaviour

Figure 18: *Stochastic fluctuations and protein synthesis.* On the x-axis we represented the time measured in arbitrary units and on the y-axis the number of molecules of a protein x. (a) The dashed line represents the continuous deterministic behaviour, while the solid line the stochastic behaviour. (b) In the case of binary behaviour, the noise can make it difficult to determine the deterministic steady state.

In the context of genes as computational units (the topic of our research), stochastic fluctuations can hide useful signals in noise [70]. For example, in Figure 18(b), if we look only at a short time frame, it can be difficult to say whether the system is in deterministic steady state 0 or 1. Thus, it is essential to consider stochastic fluctuations in protein numbers in order to represent adequately a genetic system.

2.5.1 Noise in Gene Expression

Spudich and Koshland [155] were among the first ones to observe that molecular species within a cell are prone to noise. In their experiment they investigated bacterial movements by adding a stimulus into the environment and found great variability in the time required to return to the pre-stimulus behaviour. This variability is determined by variations in populations, which could not be explained by genetic mutations. Thus, they hypothesised that the phenotypic variability of the cells was caused by intrinsic fluctuations in the number of molecules.

Sources of Noise

Protein synthesis can be divided into three processes: regulation, transcription and translation. Several studies investigated the contributions that these three sub-processes have to the output noise. Arkin and co-workers [106, 11] modelled the protein synthesis process and observed that the stochastic behaviour is enhanced by low copy number of mRNA molecules, i.e., the systems with fewer transcripts displayed higher noise compared with the ones with more transcripts despite the fact the average number of output proteins was equal in both cases. Thus, translational burst (few copy numbers of mRNA transcripts are translated in high copy numbers of proteins) was identified as the main cause for noise in gene expression. However, this study analysed the gene expression process through stochastic simulations. Although simulations can match experimental data, they do not always provide an insight into the underling mechanisms which control the noise.

Thattai and van Oudenaarden [167] and, later, Elf and Ehrenberg [45] investigated noise in gene expression by using an analytical method, the Linear Noise Approximation (LNA). Their model predicted that the size of the translational burst (the number of proteins produced per transcripts) controls the size of the fluctuations. In particular, they computed that the mRNA noise is a Poisson noise (where the mean equals the variance) and the protein has the variance approximately equal to the burst size multiplied by the mean. This shows that high translational bursts increase noise, while low bursts reduce noise.

High particle number is usually associated with less noise [56]. All the above mentioned studies (Arkin and co-workers [106, 11], Thattai and van Oudenaarden [167] and Elf and Ehrenberg [45]) show that the output noise of gene expression can be controlled without increasing the number of molecules of the output protein, by changing the transcriptional and translational rates. However, these transcriptional and translational rates control the number of mRNA molecules and a higher number of mRNA molecules means a higher metabolic cost. Thus, they suggested that there is a trade-off between noise and metabolic cost, in the sense that a higher cost leads to less noise and a lower cost to more noise.

Ozbudak *et al.* [122] confirmed these theoretical results [106, 167, 45] experimentally. Using a single reporter GFP (Green Fluorescence Protein) in *B.subtilis* they measured the fluctuations in the fluorescence levels of a single cell and observed that translational bursts seem to contribute the most to the total output noise.

Alternatively, noise could stem from fluctuations in the activity state of the promoter [89]. Kepler and Elston [90] showed that, due to the fact that genes have low copy number, the activation state of the promoter is subject to fluctuations, even in the case of high copy number of TFs. The fluctuations in the activation state of the promoter can result in mRNA burst and, consequently, the output protein bursts would be generated by transcription (rather than translation). This result was also proven experimentally by Golding *et al.* [69], who tagged the mRNA transcripts with green fluorescence and observed that transcription happens in bursts.

Although in this thesis we only address bacterial cells, it is worthwhile noting that fluctuations in the gene expression of eukaryotic organisms, such as

S.cerevisiae, seem to result mainly from fluctuations in the mRNA number. Bar-Even *et al.* [13] and Newman *et al.* [118], in two independent studies, showed that noise from random birth and death of mRNA molecules or from promoter fluctuations dominates the output noise, while translation seems to only to scale this transcription noise. Furthermore, their results showed that the noise of gene expression is a scaled Poisson noise as predicted by the theoretical models [167, 45].

On a similar note, the analysis of Pedraza and Paulsson [128] supports the idea that noise stems mainly from transcription and that translation only scales the transcriptional noise. They also showed that it is difficult to determine what contributes the most to noise in gene expression. For example, these authors obtained similar results by using different mechanisms such as burst-like or graded transcription and exponential or non-exponential decay.

Reaction Speed and Noise in Gene Expression

Above we reviewed studies which identify low numbers of molecules as the main source of noise. In addition to this, the speed of reaction seems to also influence the noise in gene expression. In this context, Elston and co-workers [90, 130] investigated noise and regulation speed analytically and observed that slow regulation leads to larger noise, but also to binary output, which is caused by transcriptional bursts. These results can be explained by the fact that, when the regulation process is slow (comparable in speed with transcription), the downstream process (transcription) will follow fluctuations in the activity state of the promoter and, consequently, the regulation noise will propagate to the output. However, when the regulation process is fast, the transcription process (which is slower) will not be able to follow the fast regulation fluctuations and the regulation noise will be time-averaged [125, 167, 90]. Experimental evidence supports these analytical results by indicating that the regulation speed can contribute significantly to the output noise [19, 69].

Alternatively, the speed in other sub-processes of gene expression, such as transcription and translation, can contribute to noise. Zhu *et al.* [190] considered the speed of the transcription and translation processes, but they did this through stochastic simulations. They found that the output noise is highly dependent on the time it takes for the transcription and translation processes to complete and on the burst size. In particular, slow transcription or translation leads to higher noise level in the output protein. In conclusion, the speed of all sub-processes related to gene expression (regulation, transcription and translation) can affect the noise levels, in the sense that lower speeds leads to higher noise and higher speed to less noise.

The Distribution

In addition to measuring noise levels in proteins, several studies also addressed the distribution of the protein numbers. The LNA approximates each reaction as being a Gaussian distributed one, which displays a negligible error only for high mean values. Friedman *et al.* [53] computed the noise assuming an exponential distribution of the burst size. They showed that the actual distribution of the output protein is a Gamma distribution with the variance equal to the burst size multiplied by the mean. Although the distribution differs from the previous studies, the variances are the same.

Recently, Taniguchi *et al.* [166] also proved the existence of the Gamma distribution experimentally by investigating the noise in 1018 genes from *E.coli.* They found that fitting the distribution to a Gamma distribution gives better results than for any other distribution. For low abundant proteins, the authors were able to correlate the distribution parameters to real biological parameters, namely: the noise (variance divided by the square mean) equals the transcription rate and the Fano-factor (variance divided by the mean) equals the burst size.

The difference between this Gamma distribution and a normal distribution is a skewness to the right, which becomes strong only for low abundant proteins. Note that, in this thesis, when we plot the distribution of protein abundances we use a normal distribution. This approach is justified by the fact that our numerical examples consist of medium and high abundant proteins for which the normal distribution approximates the Gamma distribution with high accuracy.

Contributions to Noise

Due to low copy number in the molecular species involved in gene expression, stochastic fluctuations affect the output protein. The noise which is generated by fluctuations in the three sub-processes (regulation, transcription and translation) is *intrinsic* to each gene and depends on the kinetic parameters of the biochemical processes [161, 48, 129].

In addition to the inherent noise in the biochemical processes related to gene expression (intrinsic component), the output is also affected by fluctuations in other cellular compartments, such as the number of RNAp or ribosome molecules, the cell division time, the degradation machinery and the cell environment [161]. These fluctuations are called *extrinsic* noise and they affect all genes within a cell equally. Theoretical studies showed that the variances of the extrinsic and intrinsic components are additive and that their sum equals the variance of the output protein [161, 123, 165].

Elowitz *et al.* [48] proved the existence of the intrinsic (from the process itself) and extrinsic (from external factors) components of noise experimentally by using two fluorescence genes (cyan and yellow) with identical promoters in *E.coli* bacterium. The difference between the two fluorescence intensities was generated by independent intrinsic fluctuations in the two genes and, thus, was quantified as the intrinsic contribution to noise. Furthermore, the degree of correlation between the noise in the two genes resulted from noise in the cellular compartments that affected all genes equally and was measured as the extrinsic component of the noise.

It worthwhile noting that Taniguchi *et al.* [166] observed experimentally that, for low mean protein abundances, the intrinsic noise is dominant while, for high abundances, the extrinsic noise sets a lower limit on the total noise.

Finally, genes can also be affected by noise from an upstream gene [161, 129]. This *upstream* noise is different from the extrinsic noise and from the intrinsic one in the sense that it quantifies the contributions of the fluctuations in the transcription factors. Pedraza and van Oudenaarden showed both theoretically and experimentally that the variance of the upstream component of noise is additive to the variances of the intrinsic and extrinsic components [129]. In the context of gene networks, the upstream noise plays a crucial role because increasing the number of genes in a system can lead to higher total noise.

Noise in Gene Circuits

Thattai and van Oudenaarden [168] investigated theoretically the propagation of noise in cascades of genes. A gene cascade is a network of genes where genes are connected serially, i.e., a gene regulates only the next gene in the cascade and is regulated by the previous one. They found that the output noise can be bounded from above even in the case of an arbitrarily large cascade. The main condition is that the absolute value of the derivative of the synthesis rate with respect to the transcription factor abundance is smaller than 1. If one considers that the gene regulation function has a sigmoid shape, this means that, as long as the steady state of the genes are on the two plateaus (the inactive area in Figure 4), the noise level in a cascade cannot increase arbitrarily. In this context, we should mention that theoretical studies have shown that the noise level in the transient area (the active areas in Figure 4) is much higher compared with the one in the inactive ones [167]. This result was also proven experimentally in several studies, such as Hooshangi *et al.* [82], Dunlop *et al.* [44] and Murphy *et al.* [115].

Two studies [82, 129] investigated experimentally how the noise propagates through a network of genes which repress each other in a cascade. Pedraza *et al.* found that the upstream noise had the most significant contribution to the output noise. In addition, they also used an analytical method (Linear Noise Approximation) and correctly predicted the noise levels observed experimentally. This suggests that analytical methods can be a reliable tool to investigate the stochasticity of genes and genes networks. Furthermore, Weiss and co-workers [82, 83] observed that adding more genes in a cascade affected mainly the noise in the transient area rather than the noise in the two plateaus, which indicates that, for certain set of parameters, a gene cascade can indeed hamper noise propagation as predicted by Thattai and van Oudenaarden [168].

Noise Control using Negative Feedback

As we saw above, genes and genetic networks can be strongly affected by noise. The question that we need to answer now is: what mechanisms can be used to reduce the noise in gene expression. Alternatively, the literature proposes negative auto-regulation as a mechanism able to achieve reduction in noise, where *negative auto-regulation* (or *negative feedback*) assumes that the expressed protein of a gene becomes a transcription factor and acts as a repressor for the same gene; see section 6.4 and Figure 39. The first who engineered negatively auto-regulated gene synthetically in live bacteria were Becskei and Serrano [20]. Their experiments revealed that the negatively auto-regulated gene displays lower noise compared with the gene without any type of auto-regulation.

Several theoretical studies examined whether negative auto-regulation can lead to lower noise by comparing a negatively auto-regulated gene to a gene without any type of auto-regulation. These theoretical studies confirmed that under certain conditions negative feedback can lead to a reduction in noise output [167, 126, 123, 84, 31, 188]. Paulsson [123] investigated analytically a negatively auto-regulated gene and identified that negative auto-regulation alters the noise of the output in three ways:

1. it reduces intrinsic noise (see section 2.5.1) by reducing the mean behaviour;
2. it increases extrinsic noise (see section 2.5.1) by increasing the speed and, consequently, reducing time averaging;
3. it reduces the sensitivity which compensates for the reduction in time averaging and produces an overall reduction of extrinsic noise,

where sensitivity is defined as the relative change in the output as a response to a change in the input.

Supporting the idea that negative auto-regulation can lead to lower extrinsic noise configurations, Brugemman *et al.* [31] compared a negatively auto-regulated gene with a simple one and considered both transcription and translation processes. They observed that slow protein dynamics will generate large noise in transcription when the latter is fast. Thus, there is a trade-off between the noise in transcription and noise in translation, i.e., higher noise in mRNA leads to less noise in protein and lower noise in mRNA to higher noise in protein.

A similar two steps system was assumed by Paulsson and Ehrenberg [126], who observed that noise in the repressor protein can reduce the noise of a negatively auto-regulated gene under the value of intrinsic noise. As opposed to Brugemman *et al.* [31], Paulsson and Ehrenberg [126] kept the average number of molecules and the synthesis rate fixed. This indicates that their comparison analysed systems which displayed similar metabolic costs.

As we will show in this thesis, different metabolic cost can lead to different noise levels. Thus, if one compares two systems (or two configurations) to identify how a component (or a parameter) affects the noise, the two systems should display equal metabolic costs. This way, the results are not affected by changes in the metabolic cost, but by changes in the configuration (or parameters) of interest.

Hornung and Barkai [84] compared the negatively auto-regulated system to a simple gene and observed that, although negative feedback reduces noise, the reduction in the sensitivity of the signal is greater. This reduction in sensitivity makes it difficult to distinguish signals from stochastic noise. However, the study of Hornung and Barkai [84] did not consider two systems with equal metabolic costs, which as mentioned above represents a drawback.

Zhang *et al.* [187] investigated how negative auto-regulation affects the noise under the assumption of fixed gain and average number of molecules, which indicates that the authors aimed to keep the costs of the simple system and the negatively auto-regulated one equal. They found that there is an optimal value for the auto-repression strength, for which the noise is minimum. In addition, they compared the negative auto-regulation mechanism with the positive auto-regulation one and showed that negative feedback always reduces the intrinsic noise while the positive one increases it.

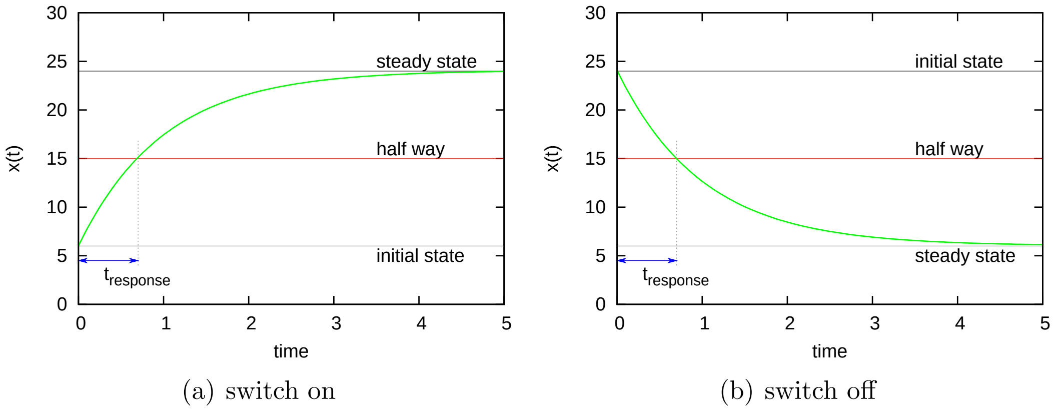
In addition, noise in negatively auto-regulated genes was examined also in the frequency domain. Simpson and co-workers [154, 12] showed both theoretically and experimentally that negative auto-regulation not only reduces the variance of the output, but also shifts the noise spectrum to higher frequencies. Higher frequency noise can be averaged out by the cell and, thus, it is easier to remove compared with low frequency noise [138].

2.5.2 Switching Time

In this thesis, we investigate genes as computational units and are particularly interested in their computational limits. The noise hampers the accuracy of the computation and, as we saw above, it can strongly affect genetic based computations by hiding useful signals [25]. A more general problem related to computations is the speed at which computations are performed [102]. Genes do not process information instantaneously, but are rather affected by a time delay [5]. This time delay, which can be thought of as computational lag, is the time required for the output protein to build up or to be decayed once the input was changed. From the computational point of view, this delay is undesirable, because we want to perform computations as fast as possible.

The response time (or switching time) is defined as the time required by a gene (gene network) to achieve a fraction of the new steady state once the input is changed. Usually, the response time measures the time to reach half the distance between the initial state and the steady state of the gene [5]; see Figure 19.

Hooshangi *et al.* [82] performed stochastic simulations and measured the response time as the time to reach half of the steady state. Their results show that the response time increases linearly with the length of the cascade. Thus, alternative methods to decrease the response time of a gene should be employed in conjunctions with methods to reduce the length of a network.



(a) switch on

(b) switch off

Figure 19: *Response time.* The response time, t_{response}, is the time required to reach half activation of a gene. We considered both switching on (a) and switching off (b).

Negative auto-regulation was suggested as an alternative approach to reduce the response time of a single gene. Rosenfeld *et al.* [137] showed theoretically that negative auto-regulation can speed up only the turn on response time, i.e., the turn on time of a negatively auto-regulated gene is five times smaller than the one of a simple gene. However, their results indicated that the turn off time remains unaffected by negative auto-regulation. In the context of our analysis (gene as computational units) the direction of switching is not important in the sense that we need the system to turn on and off as fast as possible. One of their assumptions is that the auto-repressed gene does not have a leaky expression, which is not always biologically plausible. As we will show in this thesis, this assumption has the potential to affect the results significantly and, thus, it is necessary to investigate systems with non-vanishing leak rates as well.

2.5.3 Integrated Studies

The processing speed and accuracy of the computations are, usually, interconnected [25, 104], in the sense that changing one computational property affects the other. Thus, it is important to investigate speed and accuracy in an integrated fashion to be able to determine how changes in one property affect the other one.

There are relatively few studies where accuracy and speed were investigated together. Stojanovic and Stefanovic [160] observed a speed-accuracy trade-off curve when constructing DNA based logic gates, i.e., higher speeds were achieved at reduced accuracies and, conversely, higher accuracies were achieved at lower speeds. In their system, the trade-off was controlled by the length of the DNA strands used in constructing the gates, meaning that shorter DNA strands led to faster and less accurate gates, while longer strands led to slower and more accurate gates. However, increasing the length of the DNA strands means that more DNA material is required and, consequently, the cost is increased.

Furthermore, Wang *et al.* [173] also observed a connection between the switching time and the output noise in enzymatic feedback loops, in the sense that a fast turning on and a slow tuning off ensured the lowest output noise. A higher number of molecules in a protein-protein interaction system leads to lower noise [56], but this comes at a higher cost, aspect which was not considered in their study.

Integrated analyses of speed and accuracy did not focus only on DNA based logic gates or enzymatic networks, but several studies also investigated speed and accuracy of genes. For example, Rosenfeld *et al.* [138] investigated the noise properties of a two genes cascade, where the product of a gene represses a second gene. Their results indicated that noise can deviate significantly the gene regulation function from its mean behaviour. Furthermore, they analysed the time scales of the fluctuations and found that the intrinsic and extrinsic components of the noise have different time scales. Namely, the intrinsic component has a short time scale so that it can be time averaged by the cell cycle while the extrinsic one has a longer time scale and, thus, it will be propagated to the daughter cells. This means that cells need to integrate signals for longer time periods to time average the noise which would lead to a noise-time trade-off [86].

Another study which investigated speed and accuracy in a gene cascade is the one of Hooshangi *et al.* [82]. Their results indicated that gene cascades can increase sensitivity in the response (higher steepness) without increasing the output noise, but this comes at the cost of a slower response. Alternatively, Nevozhay *et al.* [117] showed that the sensitivity in the response of a cascade can be reduced by adding negative auto-regulation to the genes in the cascade (the response can even be linearised, i.e., the response function of the cascade becomes piecewise linear with two linear regions: one for the transient area and one for the steady state area). This leads to a finer control over the sensitivity of the response. Both of these approaches do not consider metabolic cost. Since increasing the length of the cascade or adding negative feedback changes the number of molecules in the system and, consequently, the metabolic cost, this element should be taken into account as well.

Shahrezaei *et al.* [149] analysed both noise and speed in a negatively auto-regulated gene. They observed that the intrinsic noise increases and the extrinsic one decreases with increasing the strength of the negative auto-regulation. This suggested that there is an optimal auto-regulation strength for which the noise is minimised. For the optimal configuration in noise they performed stochastic simulations and measured the response time. In comparison to the simple gene, not only that the mean response time is faster, but also the distribution of the response times is shifted more to the left, i.e., a higher subset of the population will respond faster that the average compared with the simple gene. However, their results were obtained without comparing two systems with equal cost and, in fact, the negatively auto-regulated system uses less proteins. It would be interesting to observe whether these results will still hold by imposing the condition that the metabolic costs of the simple and auto-repressed gene are equal.

In addition to speed and accuracy, computations are constrained by energy cost. Lloyd [102] showed that the maximum speed at which a physical device can perform computations is limited by the energy it uses. Furthermore, Bennett [24, 25] showed that the transcription mechanism (which can be thought of as a Turing copy machine) displays a limit on the speed and accuracy of the computations which is controlled by the energy cost of the process. Returning to our analysis (genes as computational units), we note that genes have an energy cost (metabolic cost) attached to them [4], which is defined by production, degradation and general cell maintenance costs. We expect that the metabolic cost will, most likely, influence both the speed and the accuracy of the genes.

Stekel and Jenkins [156] studied the relationship between speed and accuracy under fixed number of molecules (metabolic cost) in negatively auto-regulated genes. Their results indicated that intrinsic noise is actually increased by strong negative auto-regulation. In addition, they found that this feedback mechanism can significantly reduce the mRNA concentration and, consequently, the metabolic cost. Finally, they showed by performing stochastic simulation that negative auto-regulation can increase turn on speed. Interpreting these results, the authors speculated that it is possible that the negative auto-regulation was selected by evolution to reduce the metabolic cost or speed up turning on times rather than to reduce the noise. Nevertheless, as well as Rosenfeld *et al.* [137], Stekel and Jenkins [156] did not considered leaky gene expression and turning off times.

Another study that considers speed, accuracy and cost is the one of Tan *et al.* [163], which investigated these three properties in genetic systems that used frequency encoded signals. Their results show that, for fixed noise, the speed can be increased only by increasing the cost. In addition, they observed that negative feedback can decrease noise and increase speed while keeping the metabolic cost fixed.

Mehta *et al.* [108] investigated switching time and noise in a positively auto-regulated gene which displayed bi-stable behaviour. Under fixed number of output proteins (metabolic cost), they identified a trade-off between switching time and noise controlled by the burst size of translation. In particular, low burst size lead to slow switching and smaller noise, while a high burst size to faster switching and higher noise.

The above mentioned studies, which addressed metabolic cost, assumed that this is given by the average number of molecules. We consider that the measure of metabolic cost should also depend on the time, in the sense that the metabolic cost should be defined as the energy consumption per time unit. For example, consider the case of two proteins ($X1$ and $X2$) that have the same average number of molecules, but the first one ($X1$) is produced and decayed faster compared with the second one ($X2$). Then, more molecules of the first protein ($X1$) will be produced and decayed compared with the second one ($X2$) and, consequently, the metabolic cost associated the first protein ($X1$) will be higher compared with the one of the second protein ($X2$). Thus, we consider that studies which investigate the computational properties of the genetic systems ought to include the metabolic cost per time unit.

The aim of this thesis is to investigate three computational properties of genes (output noise, response time and metabolic cost), but we will address a different scenario compared the ones mentioned above. Specifically, we will investigate analytically (or numerically) systems where simple genes (not bi-stable systems) perform computations and where signals are encoded as abundance levels of proteins (not as frequencies of oscillatory inputs). Note that, in addition to the analytical approach, we will also perform stochastic simulations aimed to validate the results.

In particular, we will investigate the output noise, response time and metabolic cost in three types of systems:

1. in simple genes,
2. in genetic networks that consist of simple genes and
3. in genes that repress themselves (negative feedback).

Furthermore, we consider that our systems have the following assumptions:

1. the response time is determined by the maximum of turning on and turning off times,
2. the genes can display leaky expression,
3. the metabolic cost will be a measure of the energy consumption per time unit.

2.6 Summary

In this chapter, we presented different mechanisms employed in performing computations with biological molecules. We started this review by presenting Adelman's DNA computing [3] which uses DNA oligonucleotides to solve the Hamiltonian path problem (HPP). This type of solution was also employed in solving other combinatorial problems, such as the satisfiability problem [99]. Nevertheless, researchers raised concerns regarding whether DNA computing is best suitable for these types of problems [6]. Hartmanis [73] addressed the scalability of the approach and computed that, to solve the HPP problem for 200 cities, would require a DNA quantity equal to the size of the Earth, making the approach infeasible.

In addition to these combinatorial problems, biological molecules have also been employed in logical computations. We can distinguish three classes of molecular logic gates: DNA based, enzymatic and genetic logic gates. DNA based logic gates use oligonucleotides (the gate) to transform other oligonucleotide molecules (input) into output ones. The design of these gates is modular, but they suffer from slow switching speeds (of the order of tens of minutes) and lack of an auto-reset mechanism (a mechanism to clear the current state once the input is no longer present) [160].

Enzymatic gates solve the problem of speed due to the fact that they can be switched in the order of a tenth of a second [29]. These gates consist of allosteric enzymes able to transform a substrate into a product only in the presence of specific ligands. Enzymatic gates come at the cost of an increased complexity in the design, in the sense that altering a gate and adding additional inputs cannot be achieved unless the enzyme is replaced. Furthermore, these gates are usually fine tuned by changing the concentration of the species of the system, which is not always possible.

Next, we reviewed logic gates constructed from genes. Gene activity is mediated through the presence or absence of transcription factors. In this type of systems, the concentration of the TF (input) controls the state of the gene (gate) and as a response to this state a certain concentration of product protein (output) is expressed. These genetic logic gates are mainly affected by noise and slow speeds, but their parameters and even their logic function can be changed through direct evolution.

Note that, although there are various mechanisms used by genetic logic gates to integrate two inputs (such as transcriptional integration, OR wiring, allosteric transcription factors or translational regulation), in this thesis, we consider only transcriptional logic gates, which are genetic logic gates that integrate the inputs in the *cis*-regulatory area. Henceforth, when we refer to genetic logic gates we mean only transcriptional logic gates.

To be able to perform complex logical computations with genes we need to ensure that the genetic logic gates can be interconnected and, based on previous studies on DNA based and enzymatic logic gates, we identified two main tasks that would allow interconnection. Firstly, the parameters of the gates should match, in the sense that a downstream gene threshold should be bounded by the maximum and minimum steady states of the upstream gene. Secondly, we want to impose the condition that the gates will not reduce the signal strength (the difference between the maximum and minimum abundances), which means that a strong input signal is not decreased at the output. The second requirement is not essential in the functioning of a logic system, but will allow the interconnection of an arbitrary number of gates in a modular and automated fashion.

In addition to interconnection, we evaluated previous work on the computational properties of transcriptional logic gates. We identified three properties which characterise genes and logic gates, namely: output noise, response time and energy cost. In section 2.5 we reviewed several studies that investigated these properties both as stand-alone properties but also in an integrated fashion. We identified that the integrated approach is essential for determining how these computational properties are interconnected.

We also found that previous research, usually, did not consider systems or configurations that display similar energy cost when investigating methods to reduce noise or increase speed. Some studies assumed that, by keeping a fixed average number of molecules, the metabolic cost was kept fixed. However, we consider that a better measure of energy cost is the metabolic cost per unit time, which will measure how many molecules are produced (or degraded) on average per unit time rather than how many molecules are in the system.

Furthermore, we observed that various studies did not investigate methods to reduce both turn on and turn off response times. Due to the fact that we analyse genes as computational units, it is important that the genes display fast response times in both directions and, thus, we aim to identify methods to enhance both turning on and turning off times.

Finally, we identified that leaky expression was disregarded in several studies. As we will show in this thesis, leaky expression can affect both speed and noise, but leak-free systems come at high metabolic costs.

Chapter 3

Stochastic Methods

In this chapter, we present the methods required to quantify the stochastic behaviour of chemical reaction systems. First, we give a short introduction on Markov chains and then we present the Chemical Master Equation (CME) which describes a set of chemical reactions stochastically. Furthermore, we present two types of analyses (based on the CME) used to approximate noise levels in a chemical system: stochastic simulations and analytical methods. Finally, we discuss the process by which we generated the simulation results in this thesis and what software we used to achieve this.

3.1 Markov Chains

We start this chapter by introducing the reader to Markov chains. This has a double purpose: to present to the reader the tools needed in the next chapter and to provide support for the next section, *The Chemical Master Equation.*

A stochastic process is a process that involves a set of random variables which depend on a parameter (usually time), that is

$$\{X(t), t \in T\}, \tag{3}$$

where $X(t)$ is the state of the system at time t.

Consider a fixed set of states called the *state space* and a system within a certain state. A *Markov process* is a process which displays the Markov property, i.e., the next state of the system depends only on the most recent state and not on any previous states. If we consider that $t_0 < t_1 < \ldots < t_n < t_{n+1}$, then the Markov property can be written as

$$\Pr\{X(t) = x|X(t_0) = x_0, \ldots, X(t_n) = x_n\} = \Pr\{X(t) = x|X(t_n) = x_n\}. \quad (4)$$

There are two types of Markov Chains: Discrete Time Markov Chains (DTMC) and Continuous Time Markov Chains (CTMC). In a DTMC, the state of the system is observed at discrete time units, while in a CTMC the system may change state at any time. In the next two subsections, we will present briefly these two frameworks drawing from the books of Stewart and Tijms [157, 169].

3.1.1 Discrete-Time Markov Chains

In a discrete-time Markov chain the probability that the system is in state i_{n+1} at step $(n + 1)$ depends only on the state of the system at step n

$$\Pr\{X_{n+1} = i_{n+1}|X_0 = i_0, \ldots, X_n = i_n\} = \Pr\{X_{n+1} = i_{n+1}|X_n = i_n\}, \quad (5)$$

where the system can only be in a state from the state space I, so that $i_0, \ldots, i_n, i_{n+1} \in I$. In this contribution, we consider only time-homogeneous Markov-chains, which can be described by the following equation

$$p_{ij} = \Pr\{X_{n+1} = j|X_n = i\}, \ \forall n. \quad (6)$$

This means that the transition probability p_{ij} is the one-step transition probability that the system will jump from state i to state j independent of current step n.

This probability needs to satisfy the following two properties:

$$p_{ij} \in [0,1],\ \forall i,j \in I, \quad \text{and} \quad \sum_{j\in I} p_{ij} = 1,\ \forall i \in I. \tag{7}$$

The first condition ensures that the value is a probability, while the second requires that the sum of all the transition probabilities from a state needs to be 1, i.e., it is a sure event that the system will stay in the current state or move to one of the other allowed states.

These probabilities can be written into a matrix form, $\mathbf{P}$, where each element in this matrix represents the probability to go from the state indicated by the row into the state indicated by the column, $(\mathbf{P})_{ij} = p_{ij}$.

If we denote by $\pi_i(n)$ the probability that a Markov chain is in state i at step n, then we can write

$$\pi_i(n) = \Pr\{X_n = i\}. \tag{8}$$

We can then say that it is a sure event that the system is in a state from the state space

$$\sum_{i\in I} \pi_i(n) = 1. \tag{9}$$

The *probability distribution* at step n, $\boldsymbol{\pi}(n)$, represents the row vector formed by all the probabilities π_i at step n.

A Markov chain is described completely by the state space, I, the initial probability distribution, $\boldsymbol{\pi}(0)$, and the transition probability matrix, $\mathbf{P}$. The probability distribution at step n can then be computed as

$$\boldsymbol{\pi}(n) = \boldsymbol{\pi}(0)\mathbf{P}^n. \tag{10}$$

An *ergodic* state is a state where: (*i*) the system is guaranteed to return infinitely often (recurrent state), (*ii*) the average number of steps required to return is finite (positive recurrent state) and (*iii*) the system can return to this state in any number of steps (aperiodic state). An ergodic system is a system which contains only ergodic states.

The ergodicity theorem states that, in an ergodic system, the distribution $\boldsymbol{\pi}(n)$ always converges to a stationary distribution $\boldsymbol{\pi}$, independent of the initial state.

$$\boldsymbol{\pi}\mathbf{P} = \boldsymbol{\pi}. \tag{11}$$

This stationary distribution is called the *steady state*.

3.1.2 Continuous Time Markov Chains

A continuous-time Markov chain is a Markov process with a set of discrete states were the transitions are not deterministic, but rather stochastic.

As in the discrete case, we consider only time-homogeneous Markov chains

$$p_{ij}(\tau) = \Pr\{X(s+\tau) = j | X(s) = i\}, \tag{12}$$

where the probability p_{ij} is independent of s. The sum of all transition probabilities from a state is equal to 1 for all τ; i.e.,

$$\sum_j p_{ij}(\tau) = 1, \forall \tau. \tag{13}$$

In a continuous-time Markov chain, the probability that a transition will occur depends not only on the current state, as in the discrete case, but also on the observation time interval, $\tau = dt$. To avoid this, we will use a different quantity, the *transition rate*, which represents the number of transitions per time unit

$$q_{ij}(t) = \lim_{dt \to 0} \frac{p_{ij}(dt) - p_{ij}(0)}{dt}. \tag{14}$$

Note that we assumed that the probability of observing multiple transitions in the interval dt is negligibly small compared with dt as $dt \to 0$. Thus, we denote this probability by $o(dt)$, where

$$\lim_{dt \to 0} \frac{o(dt)}{dt} = 0. \tag{15}$$

In the extreme case when the observation time interval is zero ($dt = 0$), the probability that the system will change state becomes zero.

$$p_{ij}(0) = 0, \ \forall i \neq j \in I \quad \text{and} \quad p_{ii}(0) = 1, \ \forall i \in I. \tag{16}$$

Going back to equation (14), the transition rate to go from a state i to a different state j becomes

$$q_{ij}(t) = \lim_{dt \to 0} \frac{p_{ij}(dt)}{dt}, \quad \forall i \neq j \in I. \tag{17}$$

We can also use the inverse formula to compute the transition probability as

$$p_{ij}(dt) = q_{ij}(t)dt + o(dt), \quad \forall i \neq j \in I. \tag{18}$$

This shows that the probability that a transition from state i to state j takes place is equal to the product between the transition rate and the time interval.

The transition rate that the system remains in the same state, q_{ii}, is defined using the difference between the maximum probability 1 and all transition probabilities, $p_{ij}, i \neq j$. From equation (13), we can write the probability that the system remains in the same state in the time interval $[t, t + dt]$ as

$$p_{ii}(dt) = 1 - \sum_{i \neq j} p_{ij}(dt) = 1 - \sum_{i \neq j} \left[q_{ij}(t)dt + o(dt) \right]. \tag{19}$$

where in the last equality we used equation (18).

Now, using equations (14) and (16) the transition rate to remain in the same state becomes

$$\begin{aligned} q_{ii}(t) &= \lim_{dt\to 0} \frac{p_{ii}(dt)-1}{dt} \\ &= \lim_{dt\to 0} \frac{-\sum_{i\neq j}\left[q_{ij}(t)dt+o(dt)\right]}{dt} \\ &= -\sum_{i\neq j} q_{ij}. \end{aligned} \tag{20}$$

Note that the rate at which a process remains in the current state is negative. This is a consequence of the fact that the transition rate is the derivative of the transition probability and, as the interval increases, the probability that the system will transfer to another state increases as well, while the probability that the system will remain in the same state decreases.

In matrix terms, the transition rate matrix $\mathbf{Q}$ can be defined based on equations (14) and (16), as

$$\mathbf{Q}(t) = \lim_{dt\to 0} \frac{\mathbf{P}(dt)-\mathbf{I}}{dt} \quad \Rightarrow \quad \mathbf{P} = \mathbf{Q}dt + \mathbf{I}, \tag{21}$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{P}(dt)$ is the transition probability matrix, $(\mathbf{P})_{ij}(dt) = p_{ij}(dt)$.

At steady state, the continuous-time Markov chain displays a probability distribution vector $\boldsymbol{\pi}$, which contains the steady-state probabilities that the system is in the corresponding state. From equation (11) we can write the steady state equation as

$$\boldsymbol{\pi}\mathbf{P} = \boldsymbol{\pi} \quad \Rightarrow \quad \boldsymbol{\pi}\left(\mathbf{Q}dt + \mathbf{I}\right) = \boldsymbol{\pi} \quad \Rightarrow \quad \boldsymbol{\pi}\mathbf{Q} = 0. \tag{22}$$

3.2 The Chemical Master Equation

The deterministic approach assumes that both reactants and products of a chemical reaction are measured in continuous units and that chemical reactions are fast. These assumptions lead to the deterministic behaviour of a chemical system and allow stochastic fluctuations to be disregarded. Genes are measured in discrete units, they have low copy numbers and they are slowly expressed, making the deterministic approach an inappropriate one for gene regulatory systems [106, 11, 87].

We begin our stochastic analysis of a chemical system by deriving the chemical master equation [127, 67]. The model assumes a well stirred biochemical reaction system in a constant volume Ω and at a constant temperature. In this system, there are N molecular species, $\{S_1, \ldots, S_N\}$. These N molecular species interact through M reaction channels, $\{R_1, \ldots, R_M\}$. The dynamics of the system are specified by $X_i(t)$ which represents the number of molecules at time t for specie S_i.

Each reaction is described by the state-change vector and the propensity function. The former, the *state-change vector* $\boldsymbol{\nu}_j$, contains N elements and each element quantifies the change in number of molecules of a species if reaction j is fired, i.e., ν_{ij} represents the number of molecules of species S_i which have to be added or removed if reaction R_j is fired. Note that the matrix formed from all $\boldsymbol{\nu}_j$ as its columns is the stoichiometry matrix of the reaction system.

The propensity function of a reaction channel R_j is denoted by a_j and $a_j(\mathbf{x})dt$ represents the probability that the reaction R_j will occur in the interval $[t, t+dt)$ when the system is in a given state $\mathbf{x} = (X_1(t), \ldots, X_N(t))^T$. This propensity function can be written as

$$a_j(\mathbf{x}) = c_j \cdot h_j(\mathbf{x}). \tag{23}$$

Here, $h_j(\mathbf{x})$ represents the product of the abundances of all reactants involved in reaction R_j, e.g., if reaction R_j is $x_1 + x_2 \xrightarrow{k_1} x_3$, then $h_j(\mathbf{x}) = x_1 \cdot x_2$. The first term in the right hand side of equation (23), c_j , is the specific reaction probability rate constant and $c_j dt$ represents the probability that a randomly chosen pair of reactants of reaction R_j will react in the next infinitesimal amount of time dt in a volume Ω [63]. This is related to the deterministic reaction rate in the sense that it is equal to the deterministic reaction rate divided by the volume (and multiplied by 2 if the reactant species are the same), e.g. in the case of the aforementioned reaction ($x_1 + x_2 \xrightarrow{k_1} x_3$) c_1 becomes $c_1 = k_1/\Omega$.

From the fact that $a_j(\mathbf{x})dt$ represents the probability that a chemical reaction R_j is fired in interval dt, we can consider that the state vector $\mathbf{x}(t)$ is a continuous time Markov process on a positive N-dimensional integer lattice [64]. For this type of process we can compute the probability of being in a state $\mathbf{x}$ at time t knowing that at time t_0 the system was in state $\mathbf{x}_0$. This probability is given by the sum of the probability that the system was in state $\mathbf{x}$ and remains there (P_{remain}) and the probability that it was in a different state and moves to state $\mathbf{x}$ (P_{arrive}).

$$\begin{aligned} P(\mathbf{x}, t+dt|\mathbf{x}_0, t_0) &= \underbrace{\left[1 - \sum_j a_j(\mathbf{x})dt\right] P(\mathbf{x}, t|\mathbf{x}_0, t_0)}_{P_{\text{remain}}} \\ &\quad \underbrace{+ \sum_j a_j(\mathbf{x} - \boldsymbol{\nu}_j)dtP(\mathbf{x} - \boldsymbol{\nu}_j, t|\mathbf{x}_0, t_0)}_{P_{\text{arrive}}} \end{aligned} \tag{24}$$

The *chemical master equation* (CME) is the time derivative of the probability that the system will be in state $\mathbf{x}$ at time t knowing that at time t_0 ($t_0 \leq t$) the system was in state $\mathbf{x}_0$; see equation (14).

$$\frac{\partial P(\mathbf{x}, t|\mathbf{x}_0, t_0)}{\partial t} = \lim_{dt \to 0} \frac{P(\mathbf{x}, t+dt|\mathbf{x}_0, t_0) - P(\mathbf{x}, t|\mathbf{x}_0, t_0)}{dt}$$

Using equation (24), this yields

$$\frac{\partial P(\mathbf{x}, t|\mathbf{x}_0, t_0)}{\partial t} = \sum_{j=1}^{M} \left[a_j(\mathbf{x} - \boldsymbol{\nu}_j)P(\mathbf{x} - \boldsymbol{\nu}_j, t|\mathbf{x}_0, t_0) - a_j(\mathbf{x})P(\mathbf{x}, t|\mathbf{x}_0, t_0)\right]. \tag{25}$$

This equation describes the time evolution of the probability of the system being in a certain state (each species having a certain abundance level). In the case when fluctuations are negligible, we can use the CME to obtain the reaction rate equation

$$\frac{d}{dt}\langle \mathbf{X}(t)\rangle = \sum_{j=1}^{M} \boldsymbol{\nu}_j \cdot \langle a_j(\mathbf{X}(t))\rangle \,. \tag{26}$$

Note that we used the notation $\langle \mathbf{X}(t)\rangle$ to emphasise that this is the averaged behaviour of $\mathbf{x}$ at time t. The reaction rate equation (the macroscopic differential equation) can be obtained from this equation if the reaction propensities are linear so that $\langle a_j(\mathbf{x})\rangle = a_j(\langle \mathbf{x}\rangle)$, but reaction rates are not always linear.

The CME (25) can rarely be solved analytically for other than simple systems [65]. One solution is to use stochastic simulation algorithms, such as Gillespie algorithm, in order to generate a statistically correct trajectory of the equation.

3.3 Stochastic Simulation Algorithms

In a Stochastic Simulation Algorithm (SSA) every chemical reaction is explicitly simulated and, thus, the dynamic behaviour of the system becomes a random walk in the N dimensional space representing the N species. There are two types of stochastic simulations algorithms: the exact stochastic simulations algorithms (such as Direct Method and First/Next Reaction Method) and the approximate ones (Poisson τ-leap methods). In this thesis we use only exact stochastic simulation algorithms and, consequently, we will present solely this class of algorithms.

3.3.1 Direct Method

The Direct Method algorithm is an exact stochastic simulation algorithm which relies on defining and computing the next-reaction density function. The *next-reaction density function* $p(\tau, j|\mathbf{x}, t)$ is defined as the probability that the next reaction will occur in an infinitesimal time interval $[t+\tau, t+\tau+d\tau)$ and it will be the reaction R_j. This function can be written as

$$p(\tau, j|\mathbf{x}, t) = a_j(\mathbf{x}) \exp\left(-a_0(\mathbf{x})\tau\right), \tag{27}$$

where we introduced the notation $a_0(\mathbf{x}) = \sum_{i=1}^{M} a_i(\mathbf{x})$ [61].

There are various Monte Carlo methods to generate a random pair (τ, j) from the probability density function given by equation (27). The *Direct Method* assumes that the next-reaction density function can be written as the product of two probabilities p_1 and p_2 given by

$$p_1(\tau|\mathbf{x}, t) = a_0(\mathbf{x}) \exp\left(-a_0(\mathbf{x})\tau\right) \quad \text{and} \quad p_2(j|\tau, \mathbf{x}, t) = \frac{a_j(\mathbf{x})}{a_0(\mathbf{x})}. \tag{28}$$

This indicates that τ is an exponential random variable with mean $1/a_0(\mathbf{x})$ and j is a statistically independent random variable. Using the standard Monte Carlo inversion generating rule, Gillespie generated the pair (τ, j) by extracting two uniformly distributed random numbers r_1 and r_2 from the interval $[0, 1]$ and then computing τ and j as

$$\tau = \frac{1}{a_0(\mathbf{x})} \ln\left(\frac{1}{r_1}\right), \tag{29}$$

$$j = \text{the smallest integer satisfying} \sum_{i=1}^{j-1} a_i(\mathbf{x}) < r_2 \cdot a_0(\mathbf{x}) \leq \sum_{i=1}^{j} a_i(\mathbf{x}). \tag{30}$$

Putting it all together we can write the **Gillespie SSA** (also known as the **Direct method**) [61, 62] as fo llows

1. Compute the propensity function for each reaction channel $a_j(\mathbf{x})$.

2. Draw two uniform random variables $r_1, r_2 \in [0, 1]$.

3. Compute τ as $\tau = \frac{1}{a_0(\mathbf{x})} \ln\left(\frac{1}{r_1}\right)$.

4. Compute j so that $\sum_{i=1}^{j-1} a_i(\mathbf{x}) < r_2 \cdot a_0(\mathbf{x}) \leq \sum_{i=1}^{j} a_i(\mathbf{x})$.

5. Increase time by τ: $t = t + \tau$.

6. Apply the change vector for reaction channel R_j: $\mathbf{x} \leftarrow \mathbf{x} + \boldsymbol{\nu}_j$.

7. Go to Step 1.

In this contribution, we use a faster exact algorithm called the Gibson-Bruck SSA (also known as Next Reaction Method) [60], which is an efficient implementation of Gillespie's First Reaction Method [61, 62]. Next, we present the first reaction method and indicate the implementation particulars of this method as proposed by Gibson and Bruck [60].

3.3.2 First Reaction Method

As well as the Direct Method, the First Reaction Method it is an exact stochastic simulation algorithm which relies on selecting a random pair (τ, j). This approach assumes generating M random numbers representing the time when each reaction will take place, $\tau_i, \ i = 1, \ldots, M$. The procedure to generate these numbers is similar to the one of the Direct Method and it relies on determining the probability that a reaction R_i will fire in the time interval $[t+\tau, t+\tau+d\tau)$. This probability $p_i(\tau|\mathbf{x}, t)$ is given by

$$p_i(\tau|\mathbf{x}, t) = a_i(\mathbf{x}) \cdot \exp\left(-a_i(\mathbf{x})\tau\right). \tag{31}$$

Note that the exponential term of this probability contains only the propensity of reaction i $(a_i(\mathbf{x}))$, not the sum of all propensities $(a_0(\mathbf{x}))$ as it was the case with the next-reaction density function (27). The way the pair (τ, j) is generated implies that the two methods are fully equivalent [61, 62].

The algorithm for the **First Reaction Method** can be summarized as:

1. Compute the propensity function for each reaction channel $a_j(\mathbf{x})$.

2. Draw M uniform random variables $r_i \in [0, 1], i = 1, \ldots, M$.

3. For each reaction channel compute τ_i as $\tau_i = \frac{1}{a_i(\mathbf{x})} \ln \frac{1}{r_i}$.

4. Choose the smallest τ_i value and the corresponding index will be the new j.

5. Increase time by τ_j: $t = t + \tau_j$.

6. Apply the change vector for reaction channel R_j: $\mathbf{x} \leftarrow \mathbf{x} + \boldsymbol{\nu}_j$.

7. Go to Step 1.

The most time consuming process in these algorithms consists of drawing the random numbers. At each step, the Direct Method generates two numbers, while the First Reaction Method generates a number of random values equal to the number of chemical reactions of the simulated system. This indicates that the former is faster compared with the latter. Gibson and Bruck [60] managed to optimise the First Reaction Method by reusing the unselected $M - 1$ random values from the current step in the next one, so that in each step only one new random uniform variable is required. Furthermore, the Gibson-Bruck algorithm uses additional constructions to increase the speed, such as the dependency graph (a graph which tells which propensities change when a reaction is executed, reducing the delay of step 1) and the priority queue (a tree structure which stores the next time each reaction fires in an ordered fashion, thus, reducing the execution time of step 4).

Both Gillespie algorithms (Direct Method/First Reaction Method) and Gibson-Bruck algorithm (Next Reaction Method) are exact stochastic simulation algorithms and they simulate a random walk on the chemical master equation. Note that these algorithms do not scale well to systems with many reaction pathways.

3.4 Analytical Method

Stochastic simulations algorithms are very useful in determining the degree of stochasticity which affects a certain system. Nevertheless, to get a better insight into the underling mechanisms that control the noise, it is often necessary to describe the stochastic effects by an analytical formula. Since the chemical master equation (CME) cannot be solved exactly in most cases, we can apply an approximate method, the van Kampen Ω expansion, which is able to compute the size of the fluctuations for all species in a common compartment [171, 45, 123, 76, 56]. This method, also known as the Linear Noise Approximation (LNA), uses the second order Taylor expansion of the chemical master equation. Furthermore, the method assumes that the system is affected by small fluctuations around the macroscopic steady state (which is usually valid for large volumes Ω). Thus, all the measures are written in terms of concentration, rather than number of molecules. To emphasise this in our notation, we use the same notation as in the case of number of molecules, but with a tilde over the symbol, e.g. $C = \Omega\tilde{C}$.

The LNA generates the following Lyapunov equation

$$\tilde{\mathbf{A}}\tilde{\mathbf{C}} + \tilde{\mathbf{C}}\tilde{\mathbf{A}}^T + \tilde{\mathbf{B}} = 0, \tag{32}$$

which can be used to determine the covariance matrix $\tilde{\mathbf{C}}$ at steady state [171, 45, 123, 76, 56]. The two other matrices in this equation are: $\tilde{\mathbf{A}}$, the Jacobian matrix attached to the chemical reactions system, and $\tilde{\mathbf{B}}$, the diffusion matrix, which quantifies the stochasticity and depends on stoichiometry and macroscopic reaction rates (all evaluated at steady state). These two matrices can be written as

$$\tilde{A}_{ik} = \sum_{j=1}^{M} \nu_{ij} \frac{\partial \tilde{a}_j(\boldsymbol{\phi})}{\partial \phi_k} \quad \text{and} \quad \tilde{B}_{ik} = \sum_{j=1}^{M} \nu_{ij}\nu_{kj}\tilde{a}_j(\boldsymbol{\phi}), \tag{33}$$

where $\boldsymbol{\phi}$ is the vector which contains the macroscopic concentrations of each species in the system and $\tilde{a}$ is the macroscopic transition rate (the macroscopic counterpart of the propensity function a).

$\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ can be computed from the set of chemical reactions associated to the system and, consequently, $\tilde{\mathbf{C}}$ is completely determined. To compute the covariance matrix in the number of the number of molecules we have to multiply the matrix $\tilde{\mathbf{C}}$ by the volume, $\mathbf{C} = \Omega\tilde{\mathbf{C}}$.

3.4.1 Fluctuation Dissipation Theorem

Paulsson reformulated the LNA and derived an equivalent method called the Fluctuation Dissipation Theorem (FDT) [123, 124, 127]. This method consists of two steps. First, he wrote the Lyapunov equation (32) in terms of the number of molecules, as

$$\mathbf{A}\mathbf{C} + \mathbf{C}\mathbf{A}^T + \mathbf{B} = 0, \tag{34}$$

where $\mathbf{A}$ is the Jacobian matrix in terms of number of molecules, $A_{ik} = \sum_{j=1}^{M} \nu_{ij} \cdot (\partial a_j(\mathbf{x})/\partial x_k)$, and $\mathbf{B}$ the diffusion matrix in terms of number of molecules, $B_{ik} = \sum_{j=1}^{M} \nu_{ij}\nu_{kj}a_j(\mathbf{x})$.

The second step of the derivation of FDT consists of rewriting equation (34) so that instead of computing the covariance matrix, the FDT will compute the noise matrix, where each variance is normalized by the square of the average, i.e., $\eta_{ij} = \sigma_{ij}/(\langle x_i\rangle \cdot \langle x_j\rangle),\ \forall i, j = 1, \ldots, N$. In this thesis, we will use only equation (34) to produce the variances of all species in a system and then we will normalize the variances manually in each case.

Nevertheless, it is important to discuss the argument behind Paulsson's choice to normalize the variance by the square root of the average and not by the average, as it was proposed previously [123]. To follow his argument we will consider the case of two proteins, where the first protein regulates the production of the second one. The Jacobian matrix of this system, $\mathbf{A}$, is a 2×2 triangular matrix and the diffusion matrix, $\mathbf{B}$, is a 2×2 diagonal matrix. Solving the Lyapunov equation associated to the system, we can write the variance of the second species as

$$\sigma_2^2 = \frac{B_{22}/2}{-A_{22}} + \left(\frac{A_{21}}{A_{22}}\right)^2 \frac{1}{1 + A_{11}/A_{22}} \left(\frac{B_{11}/2}{-A_{11}}\right). \tag{35}$$

In the case when both species are affected only by exponential decay (e.g., when proteins are only diluted) then $B_{ii} = 2\langle x_i\rangle/\tau_i$ and $A_{ii} = -1/\tau_i$. We denoted by τ_i the average lifetime of a protein, which, here, is the inverse of the decay rate, $\tau_i = 1/\mu_i$ (μ_i is the decay rate of protein i). To have a high accuracy of the method, we assume that the synthesis rate of the second species is linear. Thus, we can write the reaction rate equation of the second species as

$$\frac{d\langle x_2\rangle}{dt} = \alpha\langle x_1\rangle - \langle x_2\rangle/\tau_2.$$

Noting that at steady state we have $0 = \alpha\langle x_1\rangle - \langle x_2\rangle/\tau_2$ we can write

$$A_{21} = \alpha \quad \Rightarrow \quad \frac{A_{21}}{A_{22}} = -\alpha\tau_2 = -\frac{\langle x_2\rangle}{\langle x_1\rangle}. \tag{36}$$

Then, the variance of the second species becomes

$$\sigma_2^2 = \underbrace{\langle x_2\rangle}_{\text{intrinsic}} + \underbrace{\overbrace{\left(\frac{\langle x_2\rangle}{\langle x_1\rangle}\right)^2}^{\text{regulation}} \overbrace{\frac{1}{1+\tau_2/\tau_1}}^{\text{time averaging}} \overbrace{\sigma_1^2}^{\text{input}}}_{\text{upstream}}. \tag{37}$$

Note that the first species is affected by Poisson noise, $\sigma_1^2 = \langle x_1\rangle$. The variance of the second species can be broken into two components, the intrinsic component, which results from the randomness in the birth/death processes associated with the second species, and the upstream component, which is generated from scaling and time averaging the variance of the input component (σ_1^2) [48, 123]. Usually, the second term in the right hand side is called the extrinsic component, but, to emphasise that the noise is generated by the first protein in the system, we will call it the upstream component.

The variance is the absolute measure of the stochastic fluctuations. To get more meaningful information on the noise, a relative measure (size-independent) is required. The literature proposes two types of measures: (*i*) the Fano-factor and (*ii*) the noise. The Fano-factor consists of the variance normalized by the mean value. Note that the Fano-factor for a Poisson process is 1 and, thus, it can be thought of as measuring the distance from a Poisson process. For our system, the Fano-factor of the second species yields

$$\frac{\sigma_2^2}{\langle x_2\rangle} = 1 + \frac{\langle x_2\rangle}{\langle x_1\rangle}\frac{1}{1+\tau_2/\tau_1} \cdot \frac{\sigma_1^2}{\langle x_1\rangle}. \tag{38}$$

The intrinsic noise becomes size independent, but the upstream one increases linearly with $\langle x_2\rangle$. The second method (*ii*) assumes the variance to be normalized by the square of the average behaviour:

$$\frac{\sigma_2^2}{\langle x_2 \rangle^2} = \frac{1}{\langle x_2 \rangle} + 1 \cdot \frac{1}{1 + \tau_2/\tau_1} \cdot \frac{\sigma_1^2}{\langle x_1 \rangle^2}. \tag{39}$$

Under this new normalization, the upstream component becomes independent of $\langle x_2 \rangle$ and the intrinsic component is now a normalized measure of the fluctuations. Paulsson supported the idea that normalizing the variance by the square of the average is essential in determining the exact source of noise when modelling biological systems. He argues that this approach can identify the exact source of high noise and then appropriate methods can be employed to reduce this. Thus, the normalization of the variance by the square of the average provides a relative measure for the noise and, in addition, it is offers an insight into the source of the noise. However, in this thesis, we will consider a slightly different normalization of the variance, which is adapted to our specific case, genes with binary output (see chapter 4 for more details).

3.4.2 Considerations on the Method

It was shown that the Linear Noise Approximation method works correctly in most cases with a few exceptions [76]. These exceptions include: (*i*) small average number of molecules, (*ii*) multi-stable systems (i.e., there are two or more stable steady states and, in the deterministic case, the state the system takes depends on the initial conditions) and (*iii*) near critical behaviour. The first exception, small average number of molecules, is a consequence of the fact that the noise is approximated to be Gaussian [171]. This is clearly not the case with a real system because the Gaussian noise could lead to negative abundances. Nevertheless, for high average number of molecules, the actual distribution, such as a Poisson distribution [76] or a Gamma distribution [53], is well approximated by a normal distribution. This is the main reason that the method displays high accuracy for large abundances and, conversely, a reduced accuracy for low abundances.

Furthermore, as Gillespie pointed out, multi-stable systems are not amenable analytically to these type of methods (LNA or FDT), because the solutions to the deterministic equations (on which the LNA and FDT are based) do not provide an accurate picture of the long-time behaviour the system, i.e., spontaneous fluctuations can generate transitions of the system between steady states and, thus, the long time behaviour is not the deterministic one [64].

Finally, another problem of the method was pointed out by Elf and co-workers [46, 127]. When the system is in a near critical point, a point where large fluctuations happen and the relaxation times are slow, the method seems to produce wrong results. A more general statement would be that the LNA works correctly only for small noise around the steady state, while large noise can reduce accuracy. In particular, there are two examples where the method seems to fail: (*i*) high stoichiometry of the chemical reaction network generates large noise (noise produced by bursts of production) and (*ii*) when an element acts like an amplifier (i.e., small fluctuations in the regulatory input of a gene produce high fluctuations at the output).

3.5 Our Simulation Method

In this thesis we used both simulations and analytical results. Due to the fact that the chemical master equation can be solved only for a few very simple systems, our stochastic analysis of genetic systems used an approach based on approximate analytical methods aimed to extract qualitative behaviour from the system as a response to changes in its parameters, but also stochastic simulations aimed to verify whether the *approximate* analytical methods provided the correct results. For our stochastic simulations, we used only the Gibson-Bruck algorithm to simulate a system stochastically (see subsection 3.3.2).

When performing the simulations, we did not implement these stochastic algorithms, but rather used the implementation of these algorithms in the Dizzy software [134]. This software package benefits from ODE solvers (to compute the deterministic behaviour), exact stochastic simulation algorithms (Gillespie [61] and Gibson-Bruck [60] algorithms) and approximate stochastic simulation algorithms (Poisson τ-leap algorithm [66]). Dizzy comes with a GUI interface where one can load/write a model and then simulate it. The models are specified in the Chemical Model Definition Language (CMDL), which allows interoperability between this application and other available software. We used the GUI interface when we needed just one run of the simulation or when we produced histograms.

In addition, Dizzy comes with a programmatic interface, which actually consists of all the core classes used by the GUI. Using the Java programming language, we created CMDL files and then simulated the system using the Dizzy libraries. The advantage of using this approach consisted of the fact that we could easily change the parameters in the CMDL file and then reload the file into the Dizzy module to simulate it. This allowed us to automate the process of simulating a system stochastically for different sets of parameters.

Once the simulations for each point were performed, we exported the results into a text csv file (comma-separated values file). Using these csv results files we computed the average number of molecules and the variances for all the species in the analysed system, namely

$$\langle X \rangle = \frac{1}{N} \sum_{i=1}^{N} X_i \quad \text{and} \quad \sigma_X^2 = \left\langle (X - \langle X \rangle)^2 \right\rangle , \tag{40}$$

where $\langle (X) \rangle$ represents the time average and $\sigma_{(X)}^2$ the variance. We denoted by N the number of simulation points and by X_i the abundance at time step i.

Usually, we checked the accuracy of our simulations by running multiple stochastic simulations for the same set of parameters. This resulted in more than one value for the variance and was graphically represented by using error bars. The error bars where constructed by defining three points: the minimum, the average and the maximum values calculated from the entire set of simulations. These three points were computed by parsing in Java the csv file generated by the Dizzy programmatic interface. Note that we used both 10 and 20 simulations per set to produce the error bars.

In this thesis we also represented graphically the stochasticity of the systems by drawing histograms with the number of molecules of a certain species. These histograms where constructed by using stochastic simulations and counting the times a species had a certain number of molecules normalized by the number of time steps for each possible number of molecules. We parsed the csv files generated by Dizzy in Maple and produced a csv file with the histogram data.

Furthermore, in the captions of the figures, we usually represent values in concentrations instead of particle numbers, in order to keep the numbers small. Using a cell volume of $V = 8 \times 10^{-16}$ l, which is the average volume of *E.coli* [144], we convert from concentrations into number of molecules using

$$x = \tilde{x} \times V \times N_A, \tag{41}$$

where $\tilde{x}$ is the concentration of species x and N_A is the Avogadro number ($N_A = 6.02 \times 10^{23}$ mol^{-1}). Hence, a concentration of 1 μM in *E.coli* corresponds to approximately 500 molecules. Note, that when we considered a different cell volume, we explicitly mentioned this in the caption of the figure.

Finally, we would like to acknowledge that we used the following software to generate the data in this thesis:

- **Dizzy** (Version 1.11.4) to stochastically simulate chemical reaction systems [134] (*http://magnet.systemsbiology.net/software/Dizzy/*);

- **Java** (Java SE 6) to automate the process of stochastic simulations and to parse the data resulted from the simulation (*http://java.sun.com/javase/6/*);

- **Maple** (Versions 11, 12 and 13) for numerical computations and to produce the histograms (*http://www.maplesoft.com/*);

- **GNU Octave** (Version 3.0.1) for numerical computations (*http://www.gnu.org/software/octave/*);

- **Prism model checker** (Version 3.2) to verify our Continuous Time Markov Chain model [94, 95, 77] (*http://www.prismmodelchecker.org/*);

- **Gnuplot** (Version 4.2) for generating the plots (*http://www.gnuplot.info/*);
- **Inkscape** (Version 0.46) for producing the explanatory pictures (*http://www.inkscape.org/*).

3.6 Summary

In this chapter we reviewed the methods used to describe and analyse chemical reaction networks (gene networks) stochastically. First, we introduced the reader to Markov chains. This provides the required methods for deriving the steady state occupancy probability of the *cis*-regulatory area of a gene (see next chapter).

Chemical reaction systems are described stochastically by the chemical master equation (CME). The main idea behind this equation is the assumption that the state of the system (the number of molecules associated to each species) is a Markov process, i.e., it depends only on the current state and not on previous states. The CME defines the time derivative of the probability of a system to be in a certain state.

The CME can be solved exactly only for a few very simple systems. This led to two alternatives for the stochastic analysis of chemical reaction systems being proposed in the literature. The first method consists of explicitly simulating each chemical reaction in the system and generates a statistically correct trajectory of the CME. Here, we presented two simulation algorithms: the Gillespie algorithm (the Direct Method) and the Gibson-Bruck algorithm (First/Next Reaction Method). The Gibson-Bruck algorithm is the fastest exact algorithm. All these algorithms can be very useful tools, but they have three main problems: (*i*) they can be very slow for large networks, (*ii*) they do not show what happens in the limit cases (like an infinite volume or synthesis rate) and (*iii*) they do not usually give an insight into the underling processes which control the stochastic behaviour of the analysed systems.

To overcome all these limitations, we used an approximation of the CME. The van Kampen system size expansion (LNA) can predict with high accuracy (in most cases) the size of the fluctuations around the deterministic steady state behaviour. A reformulation of this method is Paulsson's Fluctuation Dissipation Theorem (FDT). This method is able to produce directly the variance normalized by the square of the average behaviour of any species in the system.

Finally, we presented how we performed the simulations in this thesis and what software we used to achieve this. Note also that in the following chapters we will use both simulation techniques (Gibson-Bruck), but also analytical methods (LNA/FDT).

Chapter 4

A Genetic Switch

The switching mechanism is essential in the design of logic gates. In this chapter, we present a model of a genetic switch, the binary gene. We also define three parameters which characterise the binary gene, namely metabolic cost, switching time and output noise. In addition, we describe how these three parameters are computed.

4.1 Introduction

Electronic elements (such as transistors) are able to compute because they display a switch-like behaviour, which means that, for most of the input values, the output is either high or low and not in between. Genes are also able to display this type of binary behaviour and, thus, we can think of them as being able to compute.

In this chapter, we construct a model of a genetic switch using a single gene which is regulated by a transcription factor (TF). The abundance of TF represents the input in the system, while the quantity of the expressed protein is the output. This model will be the basic model for our genetic logic gates. To construct it, we must note that the quality of the genetic switch depends strongly on the steepness of the regulation function. We address this by investigating under which conditions genes display a switch-like behaviour and how this behaviour can be enhanced. Our results show that it is often the case that genes display a switch-like behaviour, but this behaviour has poor steepness. Due to this switch-like behaviour we call our genetic switch *the binary gene*.

In addition, we also define three properties which characterise genes, namely: (*i*) output noise, (*ii*) switching speed and (*iii*) metabolic cost. Understanding how these properties influence our model of a genetic switch is vital for its design.

Usually, genes are not stand-alone elements, but rather they are components of a network. In a network, it is important that any downstream element (an element which has as input, the output of the gene that we analyse) needs to distinguish between the high and the low output of the binary gene. The problem is that this output is afflicted by noise [155, 11, 48, 19, 138, 87, 89, 44, 133], which makes it harder to distinguish between the two states and, thus, it reduces the computation *accuracy*.

Furthermore, these genes are also beset by computational lag, i.e., they 'compute' with a certain delay. Even instantaneous changes of the regulatory protein will normally be processed with a delay called the switching time. This switching time represents the transient time required for the output concentration of the gene to reach (a fraction of) the steady state corresponding to the new input. In this thesis we will consider the cases of both instantaneous and non-instantaneous change of input. From a computational point of view, this switching time is relevant because it imposes an upper limit on the frequency with which gene regulatory networks can detect and process changes in the environment. In a very direct sense, this limiting frequency can be seen as the *computational speed* of the gene.

Finally, all proteins have an associated metabolic cost, which describes the necessary energy for their production, destruction and maintenance. In biological cells, the metabolic cost of any process can be measured by the number of ATP molecules it requires [4]. In this thesis, we will not be interested in a quantitatively exact measure of the energy expense, but rather in how the cost scales as various parameters are changed. We will therefore assume that the *metabolic cost* of a particular gene is measured well by the maximum expression rate of the gene in question; see section 4.2.1. Note that the real cost will depend on the average proportion of time over which this gene is activated, plus a number of additional factors, such as the length of the protein to be produced and so on.

This chapter is divided as follows. In the next section, we present the model of the genetic switch used in this thesis and we define the three properties which we use to characterise our genetic switch: metabolic cost, switching speed and output noise. In section 4.3 we investigate the biological mechanisms responsible for the switching behaviour and examine under which conditions this switch-like behaviour can be enhanced. Finally, we assess our method of computing the noise analytically by both considering biological examples and performing extensive stochastic simulations.

4.2 The Model of a Genetic Switch

Our model system consists of a single gene G_y (the binary gene), which has an output y and is regulated by a single input species x (see Figure 20). This system is described by the following set of chemical reactions

$$\emptyset \xrightarrow{\alpha+\beta f(x)} y, \quad y \xrightarrow{\mu} \emptyset \tag{42}$$

Here, α is the leak expression rate, $\alpha + \beta$ is the maximal expression rate of the gene, $f(x)$ the regulation function, x the concentration of the regulator, and μ is the degradation rate of the product of the gene.

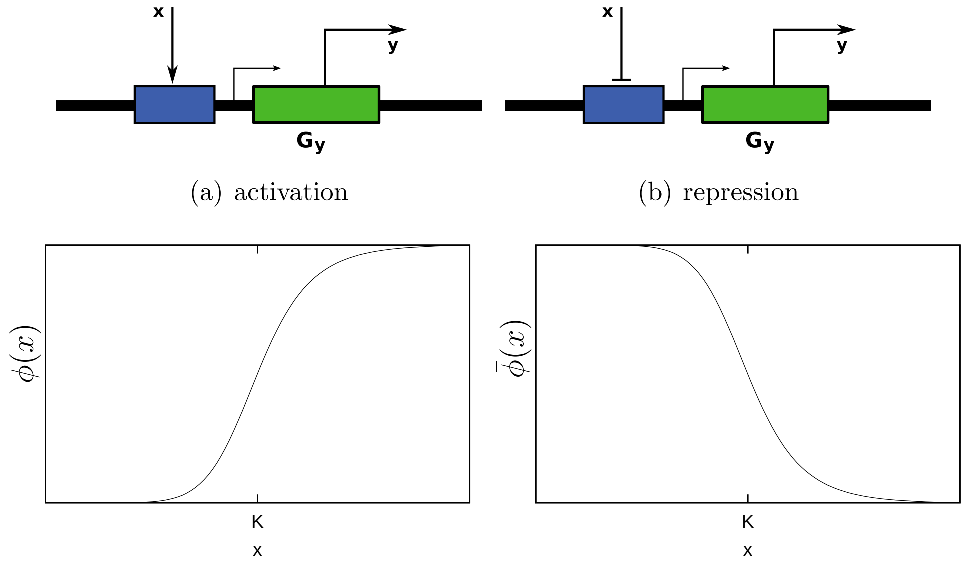
This system evolves according to the dynamical law given by the differential equation

$$\frac{dy(t)}{dt} = \alpha + \beta f(x(t)) - \mu y(t) \tag{43}$$

Note that, to emphasise the dynamic behaviour, we wrote the concentrations of the species as functions of time. At steady state, the input in this system (x) can take one of the two values: x_L (leading to a low abundance of the output $y = L$)

or x_H (leading to a high abundance of the output $y = H$).

$$L = \frac{\alpha + \beta f(x_L)}{\mu} \quad \text{and} \quad H = \frac{\alpha + \beta f(x_H)}{\mu}$$



(a) activation (b) repression

(c) activation (d) repression

Figure 20: *The model of a genetic switch.* (a) Protein x activates gene G_y. (b) Protein x represses gene G_y. (c) and (d) represent the gene activity as a function of the regulator concentration.

Gene regulation functions are often approximated by Hill functions [2, 27, 36]; the validity of this approximation is assessed in section 4.3. A Hill function is characterized by two parameters, namely the threshold (K), and the Hill coefficient (h). The latter determines the steepness of the function, whereas K represents the required input concentration to achieve half activation of the gene. In this thesis, we consider two families of regulation functions, namely the Hill activation function ($f \equiv \phi$) and the Hill repression function ($f \equiv \bar{\phi}$); see Figures 20(c) and 20(d).

$$\phi(x) = \frac{x^h}{K^h + x^h} \qquad \text{and} \qquad \bar{\phi}(x) = \frac{K^h}{K^h + x^h} \tag{44}$$

The quality of the binary output of the gene is quantified by two parameters: Hill coefficient and signal strength. The former, the Hill coefficient, measures the steepness of the regulation function in the sense that high Hill coefficients lead to steeper functions and, conversely, lower Hill coefficients to less steep regulation functions. For very high Hill coefficients, it is usually useful to approximate the regulation function by a step function which results in a perfect binary behaviour; i.e., a step function is a piecewise linear function which displays three regions: two horizontal lines (the two steady state plateaus) connected by a vertical one (the transient area). However, real genes have finite, often, very low Hill coefficients which make this approximation wrong. In this thesis, we will consider mainly the case of finite low Hill coefficients, but we will also address the case of very high ones.

The second parameter, *the signal strength*, determines the absolute distance between the high and the low steady state abundances of the output $(H-L)$. The signal strength needs to be high so that the two steady states remain distinguishable even when stochastic fluctuations affect the gene. Nevertheless, high signal strengths usually come at a high metabolic cost, which is undesirable in our case.

The integrated optimality analysis which we will perform in this thesis investigates three properties: (i) metabolic cost (ii) switching time and (iii) output noise. Next, we will determine the three properties of the system required by our optimality analysis: cost, time and noise.

4.2.1 Metabolic Cost

At any time, the actual metabolic expense attributed to a gene will be given by the time dependent production rate of the system, $\alpha+\beta f(x(t))$. Averaged over all environmental conditions, a gene will spend a certain fraction of time in the high state H; the maximum production rate is therefore an indicator of the metabolic cost ζ associated with the gene,

$$\zeta = \alpha + \beta f(x_H). \tag{45}$$

In the case of $\alpha = 0$, the metabolic cost becomes $\zeta = \beta f(x_H)$.

Our notion of cost is an idealisation of the real case. We only take into account production costs, whereas in reality there are a number of other costs related to the maintenance of protein signals. For example, if proteins are actively broken down (rather than just diluted) then this comes at an additional cost. This additional cost, however, is a constant factor and can be added to the production cost because ultimately every molecule that is produced will have to be broken down again. Also, in addition to the cost of gene expression itself, the living cell has significant extra metabolic costs associated with the overall maintenance of the cell. These extra costs are equally not taken into account here because, again, they are not relevant for our analysis which aims to investigate the scaling relationship between three properties of a binary gene (metabolic cost, switching time and output noise). Instead, as far as costs are concerned, the relevant measure is the marginal cost of protein production, rather than the total metabolic cost associated with protein levels. The production rate reflects this well and, thus, is a relevant indicator of cost when probing the fundamental limitations associated with the computational properties of genes.

4.2.2 Switching Time

We are interested in the time required for the output of a gene to reach a new steady state once the input was changed. Knowing that the regulatory input, x, evolves in time as a function $x(t)$ from x_0 to x^* (between x_L and x_H), we write the dynamics of species y as

$$y(t) = e^{-\mu t} \left[y_0 + \int_0^t e^{\mu z} \left(\alpha + \beta f(x(z)) \right) dz \right] \tag{46}$$

where y changes its concentration from y_0 to y^*. The input $x(t)$ can take any form, but to keep the mathematics simple we will first assume that x evolves exponentially.

$$x(t) = x^* - e^{-\mu_x t}(x^* - x_0). \tag{47}$$

The differential equation which describes species x is given by

$$\frac{x(t)}{dt} = \alpha_x - \mu_x x(t), \tag{48}$$

where x is produced with rate α_x and removed with rate μ_x. Note that α_x switches instantaneous between two values, α_x^L which leads to x_L and α_x^H which leads to x_H.

This means, for example, that the gene which synthesizes x is instantly turned on/off or that the protein x is activated/inactivated by another enzyme/protein in a basic enzyme reaction. Note that different types of dynamic behaviours of the input can lead to similar results.

We define the deterministic *switching time*, T_{gene} of the gene, as the time required to reach the steady state to within a fraction θ of $H - L$, see below. Thus, we compute the time to reach $y_\theta = y_0 + (y^* - y_0)\theta$. We do not always have an analytical formula for the switching time, but we can solve equation (46) numerically and determine it.

Instantaneous Input Change

In the case when the transformation or synthesis of x is very fast compared with the synthesis of y, we can assume that x changes instantaneously, so $x(t) = x$ is constant for time t. Solving the differential equation attached to the system (43), we determined the dynamical behaviour of species y as

$$y(t) \;\; = \;\; y^* - e^{-\mu t}(y^* - y_0), \tag{49}$$

where y_0 and y^* are the steady state solutions of equation (43) for the two input concentrations, x_0 and x^* respectively, that is

$$y_0 = \frac{\alpha + \beta f(x_0)}{\mu} \qquad \text{and} \qquad y^* = \frac{\alpha + \beta f(x^*)}{\mu}.$$

From equation (49) we can obtain the time required to reach a value of $y_0 +$

$(y^* - y_0)\theta$ given that we start from y_0, by setting the left hand side equal to $y_0 + (y^* - y_0)\theta$ and solve the equation for t:

$$y_0 + (y^* - y_0)\theta = y^* - e^{-\mu t}(y^* - y_0).$$

Calling this solution T_{gene} we obtain the switching time of a gene:

$$T_{\text{gene}} = \frac{1}{\mu} \ln \left(\frac{1}{1-\theta} \right). \tag{50}$$

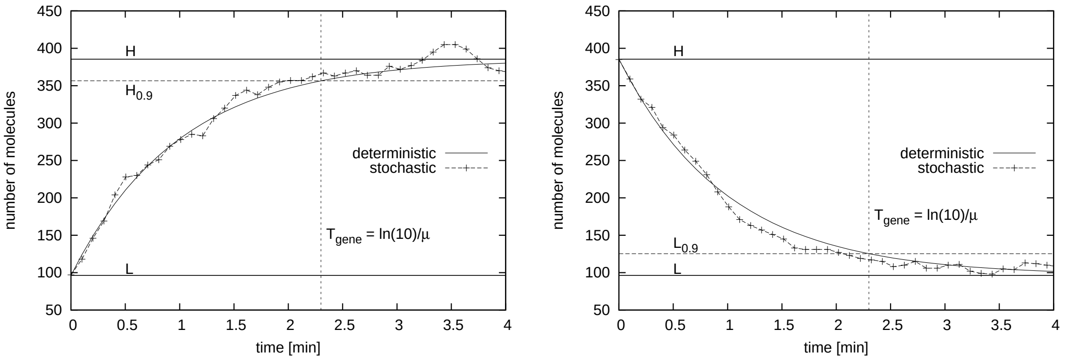
The switching time is influenced by the decay rate of the product, in the sense that higher decay rates generate faster responses. When we increase the decay rate and keep the production rate (metabolic cost) constant the high and the low state get closer to each other and the signal strength is reduced. Due to the fact that the distance between the high and the low steady states gets shorter, the switching time between the two states is also reduced.

Figure 21 confirms that this equation produces an accurate result even in the case of stochastic fluctuations.

4.2.3 Output Noise

Genes are stochastic; their products are measured in discrete units (number of molecules) and are affected by noise. The quantifiable nature of molecules means that there is a logical minimum signal strength of $H - L = 1$ molecule. More importantly, for low signal strengths (even when they are well above the logical minimum) the noise may make it difficult to distinguish between the input and the output states. The signal strength needs to be large compared with stochastic fluctuations around the steady state concentrations, so that fluctuations do not mask the output of the gene and any downstream element could distinguish between the high and the low state.

In the deterministic case, the abundances of the chemical species are measured in concentrations, while in the stochastic case in number of molecules. Here,



(a) $L \to H$

(b) $H \to L$

Figure 21: *Switching time.* We set the cell volume to $V = 8 \cdot 10^{-16}\ l$. The following set of parameters was used: $K = 0.5\ \mu M$ and $\mu = 1\ min^{-1}$. The two steady states are $L = 0.2\ \mu M$ and $H = 0.8\ \mu M$. We consider both switching (a) from low state to high state and (b) from high state to low state. We chose $\theta = 0.9$. Stochastic fluctuations do not influence significantly the time required to reach a fraction θ of the steady state concentration. For the deterministic system we solve equation (49) when (a) $(y_0 = L,\ y^* = H)$ and (b) $(y_0 = H,\ y^* = L)$.

we will not reflect the distinction between particle numbers and concentrations in our notations. Nevertheless, it is implicitly understood that stochastic systems/simulations always refer to particle numbers, rather than concentrations. Note that section 3.5 from *Stochastic Methods* chapter provides details on how to convert concentrations in particle numbers and vice versa.

One can compute the variance of the output protein, σ_y^2, by applying the Linear Noise Approximation (LNA) or Fluctuation Dissipation Theorem (FDT) [171, 45, 123, 124]. These methods assume that at steady state we have (see chapter *Stochastic Methods*):

$$\mathbf{AC} + \mathbf{CA}^T + \mathbf{B} = 0, \tag{51}$$

where $\mathbf{A}$ is the Jacobian matrix, $\mathbf{B}$ the diffusion matrix, and $\mathbf{C}$ is the covariance matrix. In our case, the Jacobian matrix $\mathbf{A}$ associated with equations (48) and

(43) yields:

$$\mathbf{A} = \begin{bmatrix} -1/\tau_x & 0 \\ \beta f'(x) & -1/\tau_y \end{bmatrix}$$

where τ_x is the average lifetime of protein x and $\tau_y = 1/\mu$ is the average lifetime of protein y .

In the case when each chemical reaction adds or removes only one molecule, **B** is diagonal and, for system (43), it becomes

$$\mathbf{B} = \begin{bmatrix} 2\langle x\rangle/\tau_x & 0 \\ 0 & 2\langle y\rangle/\tau_y \end{bmatrix}$$

Having determined **A** and **B**, one can solve equation (51) and completely determine the covariance matrix **C**. The variance of species y, σ_y^2, is given by

$$\sigma_y^2 = y + [\beta f'(x)\tau_y]^2 \frac{\tau_x}{\tau_x + \tau_y}\sigma_x^2 \tag{52}$$

The linear noise approximation is only valid when the mean of the stochastic system corresponds to the solution of the deterministic system, which is the case here. We checked the accuracy of the linear noise approximation by performing extensive simulations using Gillespie's algorithm (see subsection 4.4.2 and Figure 22 for details). A comparison with the analytic results shows good agreement between values of the noise obtained from simulations and the one computed analytically.

Alternatively, if gene G_y is regulated by two input species x_1 and x_2, we can write the variance of the output as [171, 45, 123, 151, 165]

$$\sigma_y^2 = y + \underbrace{\left[\beta\frac{\partial f(x_1,x_2)}{\partial x_1}\tau_y\right]^2 \frac{\tau_{x1}}{\tau_{x1}+\tau_y}\sigma_{x1}^2}_{\text{generated by } x_1} + \underbrace{\left[\beta\frac{\partial f(x_1,x_2)}{\partial x_2}\tau_y\right]^2 \frac{\tau_{x2}}{\tau_{x2}+\tau_y}\sigma_{x2}^2}_{\text{generated by } x_2}. \tag{53}$$

Note, that in the case of multiple inputs, the noise contains a component, corresponding to each input, which adds to the total noise.

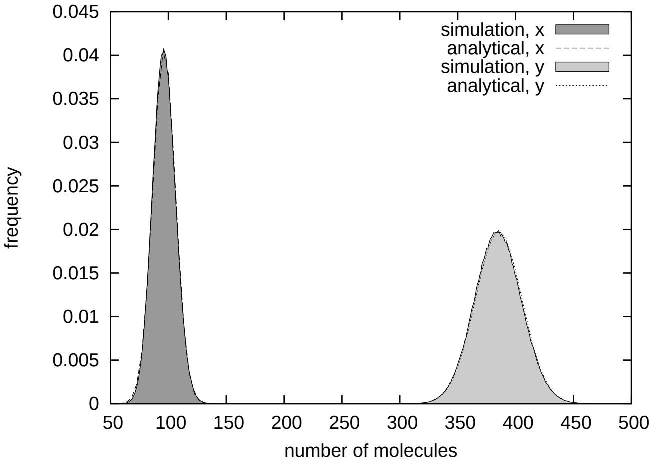


Figure 22: *Comparison between analytical solution to the noise and simulation data.* We considered a cell volume of $V = 8 \cdot 10^{-16}\ l$. The following set of parameters was used ($h = 3$, $K = 0.5\ \mu M$, $L = 0.2\ \mu M$, $H = 0.8\ \mu M$, and $\mu = 1\ min^{-1}$). At steady state, the two species have the following concentrations $x = 0.2\ \mu M$, and $y = 0.8\ \mu M$. The probability distributions of species x and y can be approximated by normal distributions.

In this contribution, we are interested in how noise affects our ability to distinguish between the two known output states, H and L. To get a meaningful measure of this, we will change the conventional definition of noise. Normally, noise is defined as the variance normalised by the square of the average behaviour. Since we are interested in how noise affects our ability to distinguish between two known values, we will adjust the conventional definition of noise and use as the following one: the variance normalised by the square of the signal strength. The variance will be much higher in the H state than in the L state. We will, henceforth, only use this pessimistic estimate and consider that the noise of the system is given by the noise in the high state (the highest noise of the system), $\eta = \sigma_H^2/(H-L)^2$,

$$\eta = \underbrace{\frac{H}{(H-L)^2}}_{\text{intrinsic}} + \underbrace{\overbrace{\left[\frac{\beta f'(x_H)}{H-L}\tau_y\right]}^{\Gamma_{yx}}{}^2 \overbrace{\frac{\tau_x}{\tau_x+\tau_y}}^{T_{yx}} \sigma_x^2}_{\text{upstream}}. \tag{54}$$

The first term on the right hand side in equation (54) represents the *intrinsic* *noise* (η_{in}), while the second term represents the *upstream noise* (η_{up}). The intrinsic component of the noise represents the randomness in the birth/death process (fluctuations in the biochemical process of gene expression), while the upstream noise represents the propagated noise from upstream components. The upstream noise has two components, namely the *time factor* (T_{yx}), which can be thought of as the time over which the gene averages its input. The other term is the regulation factor (Γ_{yx}), which determines the amplification/reduction of the upstream noise (see [151, 123, 19, 129, 152]).

4.3 Considerations on the Switching Mechanism

The quality of our genetic switch relies strongly on the steepness of the regulation function. In particular, it is essential that the regulation function of the genetic model described in equation (43) displays a sigmoid shape. Furthermore, we want this sigmoid shape to be as steep as possible. In this section we will investigate under which assumptions we can achieve sigmoid regulation functions and high steepness.

As mentioned above, we consider the case when the switching mechanism is implemented by the gene regulation function. Transcription factors (TFs) control the gene regulation process by binding to the specific binding sites on the DNA. In our model, we assume that TFs can be either specifically bound to a binding site or free in the cytoplasm. In this setting, the cytoplasm acts as a perfectly mixed *reservoir* for TFs. The binding process happens with a specific rate that depends on the affinity of the TF to the binding site and the concentration of the TFs (see Figure 23) and, thus we can describe the system by the following set of chemical reactions:

$$BS + R \underset{k_{ub}}{\overset{k_b}{\rightleftharpoons}} BSR, \tag{55}$$

where we denote by BS the specific binding site, by R the TF molecules and by BSR the complex formed between the binding site and a TF molecule. TFs molecules bind to the binding site with rate k_b and unbind with rate k_{ub}.

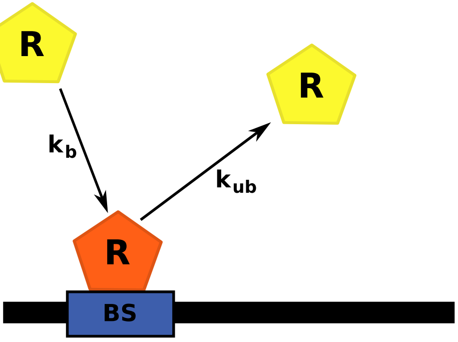


Figure 23: *The gene regulation model.* TF molecules (R) bind/unbind to/from the binding site BS with specific reaction probabilities (k_b for the binding event and k_{ub} for the unbinding event).

4.3.1 One Binding Site

Using the gene regulation model defined by equation (55) and assuming that a gene has l binding sites, we built a continuous-time Markov chain with $(l+1)$ states; the individual states of this chain correspond to $0, 1, \ldots, l$ specific sites being occupied (see section 3.1 from previous chapter *Stochastic Methods*). Furthermore, in this model, we consider only individual bindings, meaning that the system can go from state i (where i activator molecules are bound to the gene) to either $i+1$ (where another molecule binds) or $i-1$ (where a molecule unbinds). Note that Markov chains were used to describe chemical reaction systems previously, e.g., Roussel and Zhu [139] used a continuous time Markov chain to describe the binding of an RNAp molecule to a promoter.

Initially, we construct the Markov model for the simplest case, the case where a gene has only one binding site. We measure the abundance of the free (in the reservoir) TF in number of molecules. Markov chains are normally represented as $(l+1) \times (l+1)$ matrices which describe the rate (in the case of continuous-time Markov chains) or probability (in the case of discrete time Markov chains) of transition between the possible states. The transition rate from a state to another can be computed as the propensity function for a reaction to take place, i.e., the product between the number of molecules of the reactants and the specific reaction probability. When a TF molecule binds to the binding site, the transition rate is given by $R \cdot 1 \cdot k_b$, where R represents the number of TF molecules, 1 the number of binding sites and k_b the specific reaction probability of the binding process. Analogously, when a molecule unbinds from a binding site the transition rate is given by $1 \cdot k_{ub}$. The diagonal elements in the case of a continuous-time Markov chain are computed so that the sum on each row is zero. The transition matrix in the case of one binding site yields

$$\boldsymbol{Q} = \begin{bmatrix} -Rk_b & Rk_b \\ k_{ub} & -k_{ub} \end{bmatrix}$$

The steady state distribution vector π of such a continuous-time Markov chain is given by the solution to

$$\boldsymbol{\pi} \cdot \boldsymbol{Q} = 0 \quad \text{and} \quad \sum_i \pi_i = 1, \tag{56}$$

where π_i represents the steady state probability that the Markov chain is in state i [157]. We are interested in π_1, the probability that the operator site is full:

$$\pi_1 = \frac{R}{R + k_{ub}/k_b} = \frac{R}{R + K}. \tag{57}$$

The steady state probability that an operator with one binding site is full (π_1) is a hyperbola (see Figure 24) and can be written as a Hill function with a Hill coefficient of $h = 1$ and a threshold of $K = k_{ub}/k_b$. Thus, the change in output as response to the change in input is gradual, not switch-like. This shows that a gene with one binding site where TF monomers can bind does not display the required switching behaviour.

Although most of the genes, in bacterial cells, have only one regulatory binding

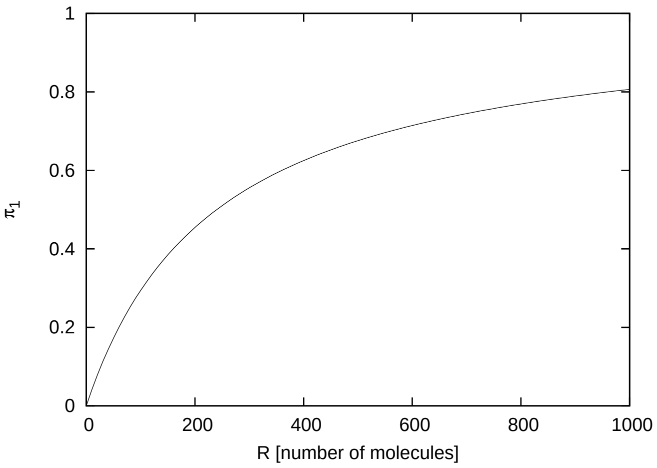


Figure 24: *Occupancy probability as a function of TF abundance.* We assumed the following parameters : $k_b = 1$ and $k_{ub} = 240$.

site for each TF, there are still a considerable number of genes which have more than one binding site (37% in *E.coli*)[78]. In the next two sections, we will address the case of genes with multiple binding sites by considering genes with two binding sites.

4.3.2 Two Binding Sites

Next, we consider the case of two binding sites, $BS1$ and $BS2$, where regulatory molecules can bind or unbind with equal probability, k_b and k_{ub}. The binding sites operate independently, meaning that binding/unbinding of TF molecules to/from one binding site does not influence the binding/unbinding probabilities to/from the other site. In this scenario, the system can be in one of the following three states: none, one or two TF molecules bound to the gene. The transition matrix is then given by

$$Q = \begin{bmatrix} -2Rk_b & 2Rk_b & 0 \\ k_{ub} & -k_{ub} - (R-1)k_b & (R-1)k_b \\ 0 & 2k_{ub} & -2k_{ub} \end{bmatrix}$$

Note that the transition rate from state 0 to state 1 becomes $Q_{0,1} = 2Rk_b$. The term is multiplied by 2 because there are two binding sites that are free, instead of one as in the previous case. Similarly, because there are two occupied binding sites, the transition rate from full occupancy (state 2) to half occupancy (state 1) is given by $Q_{2,1} = 2k_{ub}$.

Using equation system (56), one can compute the probability that both binding sites are occupied, π_2, as

$$\pi_2 = \frac{R(R-1)}{R(R-1) + 2Rk_{ub}/k_b + (k_{ub}/k_b)^2}, \tag{58}$$

where R needs to be higher than 0. For $R \leq 0$ the probability of both sites being occupied will be $\pi_2 = 0$.

Figure 25 shows that the probability that both sites are occupied is a sigmoid function (it has a S shape), i.e., for low number of regulatory molecules the probability for the operator site to be fully occupied is very low, while for high number of regulatory molecules is very high.

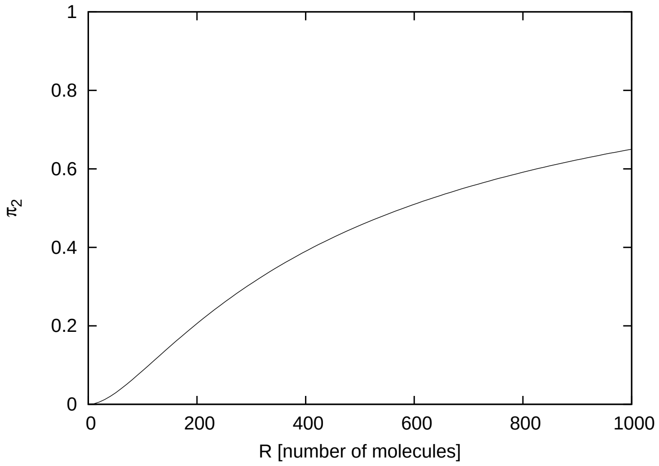


Figure 25: *Full occupancy probability as a function of TF abundance.* We assumed the following constants: $k_b = 1$ and $k_{ub} = 240$.

It was previously postulated that TFs bind to binding sites in a cooperative manner, in the sense that, after the first binding site is occupied, the probability that the second site is occupied changes [2, 21, 22, 27]. Cooperativity can be included easily in the Markov chain model as a scaling of both binding and unbinding reaction propensities (see Figure 26). In this scenario, the transition rate matrix becomes:

$$Q = \begin{bmatrix} -2Rk_b & 2Rk_b & 0 \\ k_{ub}C^- & -k_{ub}C^- - (R-1)k_bC^+ & (R-1)k_bC^+ \\ 0 & 2k_{ub} & -2k_{ub} \end{bmatrix}$$

where $C^{\pm}$ is the cooperativity modifier, i.e., a factor that determines how the forward and backward binding rates are changed when there are free binding sites in the case of C^- or when there are occupied binding sites in the case of C^+. If this value is > 1 then we deal with positive cooperativity (i.e., once one site is bound binding to further sites is facilitated), otherwise cooperativity is negative. To illustrate the origin of the entries of this matrix, we consider the case when one molecule is bound to a binding site (state 1). Another molecule can bind to the free binding site (moving the system in state 2) or the already bound molecule can unbind (moving the system to state 0). Both transition rates from state 1, Q_{12} and Q_{10}, are affected by cooperativity, binding cooperativity (C^+) in the case of Q_{12} and unbinding cooperativity (C^-) in the case of Q_{10}.

We determined the steady state probability that both binding sites are occupied, when the sites are cooperative, as

$$\pi_2^c = \frac{R(R-1)}{R(R-1) + 2\frac{1}{C^+}R\frac{k_{ub}}{k_b} + \frac{C^-}{C^+}\left(\frac{k_{ub}}{k_b}\right)^2} = \frac{R(R-1)}{R(R-1) + 2\frac{1}{C^+}R \cdot K + \frac{C^-}{C^+}K^2}. \quad (59)$$

Note that if both cooperativity terms have high positive values then state 1 becomes a transient state (the system stays in this state only for short a period of time), while states 0 and 2 become stationary states (the system stays in these

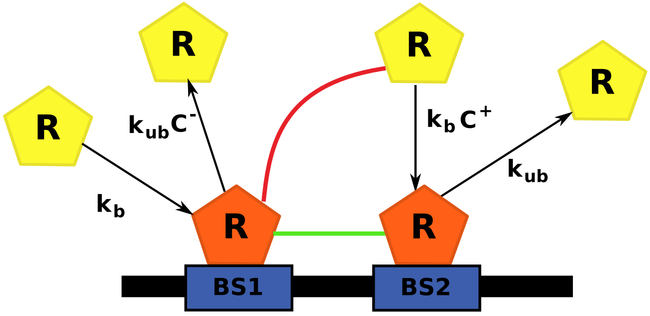


Figure 26: *The model with two cooperative binding sites.* Once a TF molecule binds to a binding site ($BS1$ in the graph) it attracts/repels other TF molecules to the other binding site ($BS2$) through binding cooperativity (see red line). If both sites are occupied, then a stable complex is formed, which changes the unbinding probability from $k_{ub}C^-$ to k_{ub} (see green line).

states for a long period of time). Thus, we obtain

$$\pi_2^c = \frac{R(R-1)}{R(R-1)+K^2}, \quad \text{for } C^+ = C^- \gg 1. \tag{60}$$

Figure 27 shows the effects of cooperative behaviour on the probability that all binding sites are occupied. The figure confirms that cooperativity can enhance regulation steepness. Considering only cooperative binding between sites steepens the functions, but simultaneously shifts the curve to the left (the number of required molecules that fully occupy the gene decreases). However, if the model assumes both cooperative binding and cooperative unbinding, the steepness of the function is enhanced while the curve does not shift to the left.

The probability that a gene is turned on or off is usually modelled as a Hill function, a sigmoid function described by two parameters: the Hill coefficient (which determines the steepness of the function) and the threshold (which determines the input required for half activation of the gene). For very high cooperativities, in the case when both binding and unbinding cooperativity is considered, the maximum Hill coefficient is limited from above by the number of binding sites while the threshold is limited from bellow by k_{ub}/k_b (see Figure 28). Nevertheless, in the case when we take into account only cooperative binding, these limits do not

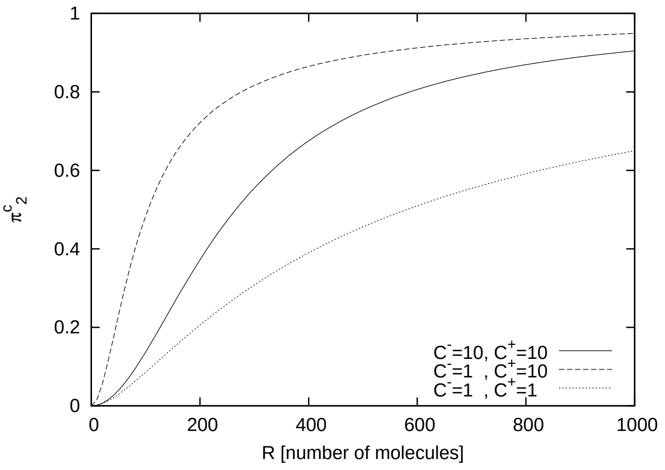
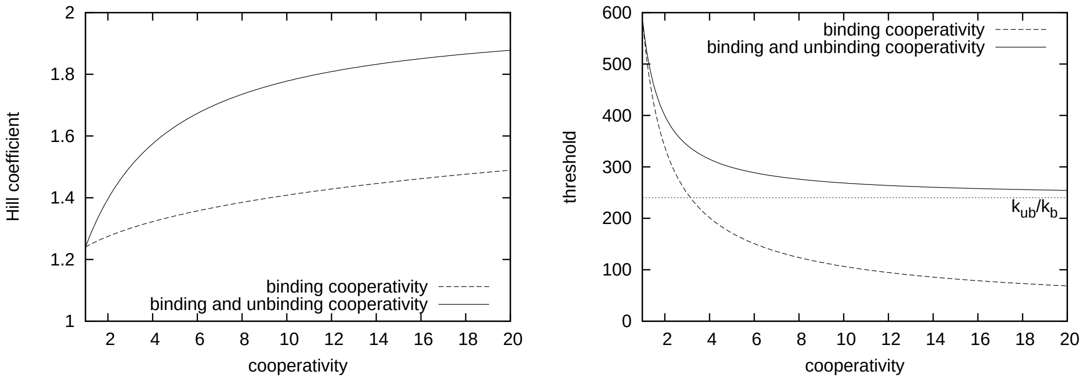


Figure 27: *Full occupancy probability as a function of TF abundance in the case of cooperative behaviour.* We assumed the following constants: $k_b = 1$ and $k_{ub} = 240$. We consider three cases: no cooperativity (dotted line), only binding cooperativity (dashed line) and both binding and unbinding cooperativity (solid line).

exist; the threshold falls bellow k_{ub}/k_b and the Hill coefficient can go over the number of binding sites (for very high cooperativity terms $C^+ > 10^4$).

Figure 28 also reveals that, for the same cooperativity terms, the system which considers both types of cooperativity is better in terms of Hill coefficient and threshold compared with the system which takes into account only binding cooperativity, in the sense that it can produce higher Hill coefficients without significantly reducing the regulation threshold. If we consider only binding cooperativity the Hill coefficient increases, but the threshold is reduced significantly. Usually, very low thresholds are undesirable, because they indicate that the low output plateau shrinks and, thus, the low state is more prone to noise. However, for very high cooperativity terms ($C^+ > 10^4$ in Figure 28(a)), the system which considers only cooperative binding will have a higher Hill coefficient.

Our Markov model assumes that TFs float freely in the cytoplasm from where they bind directly to their specific binding site on the DNA. This is unlikely to be the case for real cells. Instead, most of the TFs will be non-specifically bound to the DNA from where they can randomly search their target specific



(a) Hill coefficient

(b) threshold

Figure 28: *Fitting the gene regulation function to a Hill function.* We assumed the following constants: $k_b = 1$ and $k_{ub} = 240$. We consider two cases: only cooperative binding (dashed line) and both cooperative binding and cooperative unbinding (solid line).

site [26, 177, 92, 72, 178]. To address this, we compared our Markov model to a more biological plausible model, the *computational model*, which assumes spatial aspects of gene regulation. The results indicate good agreement between the two models [36]. Nevertheless, under certain special conditions, the Markov model was unable to capture all the details of the computational model. For example, in the case of high cooperativity and TF crowding, the computational model displays stochastic bi-stable behaviour (the *cis*-regulatory area can be either fully occupied or completely free and this is stochastic), which could not be observed in the Markov model.

4.3.3 Biological Significance

Our results showed that if TF-TF interactions are ignored and only TF monomers regulate a gene, then the gene needs more than one binding site to display a switch-like behaviour (binary response). This in conjunction with the fact that a considerable number of genes in bacterial cells have more than one binding site for the same TF [78] suggests that living organisms enhanced the switch-like behaviour of genes by adding additional binding sites. Nevertheless, the number of genes that display more than one binding site decreases exponentially with the number of binding sites, indicating poor switch-like behaviours, i.e., Hill coefficient ≥ 4 are not very common if we consider only multiple binding sites as a source of switch-like behaviour [78].

Then, how are biological systems capable of steeper responses to regulatory inputs? There are two main mechanisms, which were not considered in this contribution and can enhance the switch-like behaviour: (*i*) gene networks and (*ii*) protein-protein interactions. Gene networks, such as gene cascade and toggle switches are capable of increasing the steepness of the system's response to an input. Speed is one of the most important disadvantages of genes compared with electronic devices. Gene networks are even slower than single genes and the delay time can increase linearly with the number of genes in the network [82]. This is the main reason for which we did not consider gene networks as an alternative to single genes as biological switches. Additionally, having more than one gene requires an extra metabolic cost in the cell, which again is not desired. However, despite all these disadvantages, these genetic networks (gene cascade and feedback loops) are network motifs [5], which means that they have high occurrence in biological organisms. Why then did evolution select these types of gene network? One possible answer is that under certain conditions, these gene networks can also act as noise filters (see the toggle switch [185]) and genes are sometimes prone to high levels of noise [87].

Another recurrent mechanism in living systems are protein-protein interaction systems. One way to increase the steepness of the regulation function consists of modifying the input before it actually regulates the gene. For example, it was shown that oligomerization process (molecules of the same type can bind and form bigger molecules: dimers, trimers, etc. which then regulate the target gene), a very common mechanism in bacterial cells, can enhance the Hill coefficient [33, 113]. Note that, in the context of our Markov model, binding of dimers can be incorporated easily, by assuming infinite binding cooperativity between two monomers (once one TF monomer binds, the second one is automatically bound as well). Another example of protein-protein interaction that enhances switch-like behaviour is the Goldbeter-Koshland model [68, 162, 70, 181], which assumes that TF can be activated and deactivated by two enzymes. If one of the enzyme represents the input of the system and the other enzyme has a fixed concentration, then the system can display a step-like response.

In this thesis, we investigate only the computational properties of genes. Protein-protein interactions are faster compared with gene expression and, thus, analysing systems which contain these types of interactions may lead to better results in terms of speed and accuracy. Nevertheless, the details of this assumption are to be left to future research.

We conclude that, due to the fact that the interactions between genes and TF are mediated by multiple binding sites, genes display a switch-like mechanism. This switch-like mechanism often has a poor quality, in the sense that genetic switches do not display a high steepness between the two binary values. Nevertheless, although the number of binding sites is an indicator of how switch-like a gene is, there are also other mechanisms which can enhance the binary behaviour of genes.

4.4 Considerations on the Noise

Our model of noise ignores several sources of noise and this might lead to limitations on the usefulness of the approach. Gene expression is usually modelled as a three step process: regulation, transcription and translation. In this thesis we ignore the noise produced by the regulation and translation steps. However, as we will see bellow (in subsection 4.4.1) this assumption is often valid in the context of bacterial cells.

Furthermore, we also performed extensive stochastic simulations, aimed to verify the accuracy of the analytical formula; see subsection 4.4.2. The results indicated that the error between the noise computed by the analytical formula and the one measured in the stochastic simulations is negligible.

As a final remark we would like to mention that in this section we normalize the noise by the square of average behaviour and not by the square of the signal strength. This is aimed to simplify the mathematics, but do not affect the generality of the results.

4.4.1 Stochastic Gene Expression

Gene expression is modelled as a three steps process: (*i*) regulation, (*ii*) transcription and (*iii*) translation. In the regulation process, the gene(s) gets activated or deactivated. We will denote the number of active genes by n_1 and the number of total genes by $n_1^{\max}$. Thus, the number of inactive genes is $n_1^{\max} - n_1$. Furthermore, we denote by λ_1^+ the rate at which a gene gets activated and by λ_1^- the deactivation rate. An active gene can be transcribed with rate λ_2 into mRNA molecules which can decay with rate $1/\tau_2$. Finally, each mRNA molecule can be translated into the final protein with rate λ_3 and decayed with rate $1/\tau_3$. Note that we assumed exponential decay to be able to consider that the decay rate is the inverse of the average life time. The gene expression system can be summarised by the following differential equations.

$$\begin{aligned}
\frac{d\langle n_1\rangle}{dt} &= \lambda_1^+(n_1^{\max} - \langle n_1\rangle) - \lambda_1^-\langle n_1\rangle = \lambda_1^+ n_1^{\max} - \frac{\langle n_1\rangle}{\tau_1}, \\
\frac{d\langle n_2\rangle}{dt} &= \lambda_2\langle n_1\rangle - \frac{\langle n_2\rangle}{\tau_2}, \\
\frac{d\langle n_3\rangle}{dt} &= \lambda_3\langle n_2\rangle - \frac{\langle n_3\rangle}{\tau_3}, \qquad (61)
\end{aligned}$$

where we denoted by $\tau_1 = \left(\lambda_1^+ + \lambda_1^-\right)^{-1}$.

At steady state, we can compute the noise by applying the Linear Noise Approximation (see equation 51). Initially, we have to determine two matrices, the Jacobian matrix (**A**) and the drift matrix (**B**), and then, using these two matrices, we can compute the covariance matrix of the system. One can determine the

Jacobian matrix of this system as

$$\mathbf{A} = \begin{bmatrix} -1/\tau_1 & 0 & 0 \\ \lambda_2 & -1/\tau_2 & 0 \\ 0 & \lambda_3 & -1/\tau_3 \end{bmatrix}$$

Assuming that each chemical reaction either adds or removes only one molecule, we can write the diffusion matrix $\mathbf{B}$ as

$$\mathbf{B} = \begin{bmatrix} 2\lambda_1^- \langle n_1 \rangle & 0 & 0 \\ 0 & 2\langle n_2 \rangle/\tau_2 & 0 \\ 0 & 0 & 2\langle n_3 \rangle/\tau_3 \end{bmatrix}$$

Solving the Lyapunov equation (51) we can determine the noise of the three species as

$$\eta_1 = \frac{\sigma_1^2}{\langle n_1 \rangle^2} = \frac{1 - P_{\text{on}}}{\langle n_1 \rangle}, \tag{62}$$

$$\eta_2 = \frac{\sigma_2^2}{\langle n_2 \rangle^2} = \frac{1}{\langle n_2 \rangle} + \frac{1 - P_{\text{on}}}{\langle n_1 \rangle} \frac{\tau_1}{\tau_1 + \tau_2}, \tag{63}$$

$$\eta_3 = \frac{\sigma_3^2}{\langle n_3 \rangle^2} = \underbrace{\frac{1}{\langle n_3 \rangle}}_{\text{intrinsic noise}} + \underbrace{\frac{1}{\langle n_2 \rangle} \frac{\tau_2}{\tau_2 + \tau_3}}_{\text{noise from mRNA}} + \\ + \underbrace{\frac{1 - P_{\text{on}}}{\langle n_1 \rangle} \frac{\tau_2}{\tau_2 + \tau_3} \frac{\tau_1}{\tau_1 + \tau_3} \frac{\tau_1 + \tau_3 + \tau_1\tau_3/\tau_2}{\tau_1 + \tau_2}}_{\text{noise from regulation}}, \tag{64}$$

where $P_{\text{on}} = 1/(1 + \lambda_1^- / \lambda_1^+)$. The only assumption behind this derivation was that the fluctuations were small enough to be approximated as weakly non-linear.

Biological Significance

We are interested under which conditions the noise generated by the regulation and translation processes is negligible. To assess the validity of approximating the noise in the gene expression by the noise in the transcription process, we will consider a particular case, the case of *E.coli* bacteria.

The activation rate of a gene, λ_1^+, can be written as the product between the binding rate and the number of molecules of the TF. Thus, we can write τ_1 as

$$\tau_1 = \frac{1}{\lambda_1^- + \lambda_1^+} = \frac{1}{\lambda_1^- + k_1^+ \cdot R},$$

where we denoted by R the number of molecules of TF and by k_1^+ the affinity of the regulator molecules for the gene.

Using some common parameters from *E.coli* ($\lambda_1^- = 0.1$, $k_1^+ = 0.2\ nM \cdot min^{-1}$)[82], and the fact that TF can be found in abundance of even $R = 10$ molecules per cell, like it is the case of *LacI* repressor [179], the τ_1 becomes

$$\tau_1 \approx \frac{1}{0.1 + 0.2 \cdot 10} \approx 0.5,$$

Note that, for higher repressor abundances, this time τ_1 is much lower and, thus, it is more accurate to say that $\tau_1 \lesssim 0.5\ min$.

Furthermore, under these considerations, the regulation noise can be approximated by

$$\eta_1 = \frac{1}{n^{\max}} \frac{\lambda_1^-}{\lambda_1^+} \approx \frac{1}{1} \frac{0.14}{0.2 \cdot 10} \approx 0.75 \times 10^{-1}, \tag{65}$$

where we assumed that there is only one gene in the cell. Note that we could say again, that this is an upper value of the noise due to the fact that usually the abundance of the repressor can be higher that 10, $\eta_1 \lesssim 0.75 \times 10^{-1}$.

The half-life time of proteins affected only by dilution is approximated by the cell division time, which is usually taken to be $\tau_{\text{division}} \approx 50\ min$ [144, 138, 33, 69]. Thus, we can write $\tau_3 \approx \tau_{\text{division}}/\ln(2) \approx 70\ min$. Moreover, we consider the average lifetime of the mRNA transcript in *E.coli* to be $\tau_2 = 2.2\ min$ [144]. From equations (63) and (64) we computed the contribution of regulation to the noise in the mRNA and output protein as

$$\begin{aligned}
\eta_2^1 &= \frac{1 - P_{\text{on}}}{\langle n_1 \rangle} \frac{\tau_1}{\tau_1 + \tau_2} \approx 1.4 \times 10^{-2}, \\
\eta_3^1 &= \frac{1 - P_{\text{on}}}{\langle n_1 \rangle} \frac{\tau_2}{\tau_2 + \tau_3} \frac{\tau_1}{\tau_1 + \tau_3} \frac{\tau_1 + \tau_3 + \tau_1 \tau_3 / \tau_2}{\tau_1 + \tau_2} \approx 0.5 \times 10^{-3}.
\end{aligned}$$

In addition, we computed the contribution of the intrinsic components of the noise in mRNA and output protein to the total noise in the output protein. On average, each transcript is translated two times [83] and, thus, the relationship between the number of molecules and transcripts is given by $\langle n_3 \rangle = \lambda_3 \tau_3 \langle n_2 \rangle \approx 140 \times \langle n_2 \rangle$. The noise in the protein generated only by translation and transcription processes yields

$$\eta_3^{23} = \frac{1}{\langle n_3 \rangle} + \frac{\lambda_3 \tau_3}{\langle n_3 \rangle} \frac{\tau_2}{\tau_2 + \tau_3} = \frac{1}{\langle n_3 \rangle} \frac{\tau_2 + \tau_3 + \lambda_3 \tau_3 \tau_2}{\tau_2 + \tau_3} \approx \frac{1}{\langle n_3 \rangle}(1 + \lambda_3 \tau_2) = \frac{1}{\langle n_3 \rangle} C, \quad (66)$$

where in the third equality we approximated that the average life time of the mRNA is much lower than the one of the protein $\tau_2 \ll \tau_3$. In our case, we obtained $C \approx 5.5$.

Next we consider a numerical example. Assuming that the number of molecules of the output protein lie in the range $\langle n_3 \rangle \in [10, 1000]$, we can compute the noise in the number of output proteins as

$$\begin{aligned} & \eta_3 = \eta_3^{23} + \eta_3^{1}, \\ \langle n_3 \rangle = 10^1 : \quad & \eta_3 \approx 0.5 + 0.5 \times 10^{-3} \approx 0.5, \\ \langle n_3 \rangle = 10^2 : \quad & \eta_3 \approx 0.5 \times 10^{-1} + 0.5 \times 10^{-3} \approx 0.5 \times 10^{-1}, \\ \langle n_3 \rangle = 10^3 : \quad & \eta_3 \approx 0.5 \times 10^{-2} + 0.5 \times 10^{-3} \approx 0.5 \times 10^{-2}, \end{aligned}$$

where the first term in the right hand side is the noise resulted from transcription and translation while the second term the noise generated by the regulation process. These sets of parameters justify that regulation noise is negligible compared with transcription and translation noise. For the set of parameters that we choose, we will need that the abundance of the protein to be around 10^4 molecules $\approx 20\ \mu M$ in order for the regulation noise to have a significant contribution to the output protein. This is unlikely to be the case in bacterial cells [179] and, thus, in the case when proteins are affected only by dilution, regulation noise is negligible.

Proteins are not always affected only by dilution, but also by active decay. As a second numerical example, we consider that the output proteins are decayed fast, $\tau_3 = 7\ min$, and this leads to $\eta_3^1 \approx 0.4 \times 10^{-2}$. In this case, assuming that the average number of molecules of the output protein range between $\langle n_3 \rangle \in [10, 1000]$, the noise of the output protein yields

$$\begin{aligned} & \eta_3 = \eta_3^{23} + \eta_3^1, \\ \langle n_3 \rangle = 10^1 : \quad & \eta_3 \approx 0.5 \times +0.4 \times 10^{-2} \approx 0.5, \\ \langle n_3 \rangle = 10^2 : \quad & \eta_3 \approx 0.5 \times 10^{-1} + 0.4 \times 10^{-2} \approx 0.5 \times 10^{-1}, \\ \langle n_3 \rangle = 10^3 : \quad & \eta_3 \approx 0.5 \times 10^{-2} + 0.4 \times 10^{-2} \approx 0.9 \times 10^{-2}. \end{aligned}$$

For fast protein decay, when the average abundance of the output protein is around 1000 molecules, the regulation noise has a significant contribution to the output noise. An abundance level of 1000 molecules is not common in bacterial cells [179], but this indicates that for really fast decays and high average abundances of the output protein, the regulation noise cannot be neglected any more. Hence, we can state that, for the regulation noise to have a significant contribution some of the following three conditions need to be met: (*i*) the output protein to be produced in high abundance, (*ii*) the output protein to decay fast or (*iii*) the binding of the regulatory protein to be slow $\tau_1 \not\ll \tau_2$.

Proteins in bacterial cells have a slow decay rate (usually just dilution [33]) and are not present in high abundance [179]. Thus, regulation can play a significant role in the protein noise, mainly, when the affinity of the regulatory protein for the gene is week $\tau_1 \not\ll \tau_2$ [89]. Nevertheless, in this contribution, we will limit our attention only to genes which are not slowly regulated [87, 89].

From equation (66) we observed that the noise from translation only scales the transcription noise and does not qualitatively change its behaviour. This result is supported by experimental evidence, which shows that the noise in the expressed protein mainly stems from the low copy number in mRNA transcripts [122, 48, 161, 69]. Therefore, in this thesis, we will approximate the gene expression by a one step process (ignoring regulation and translational noise) and assume that the intrinsic noise of this process is a Poisson noise. In this context, it is worthwhile noting that experimental evidence from a eukaryotic organism, *S.cerevisiae*, suggests that the gene expression noise is often a Poisson noise, scaled by a constant factor C [13, 118]. In addition, Taniguchi *et al.* [166] investigated 1018 genes from *E.coli* and found that these genes have a skewed distribution of the noise with a variance usually higher than the mean abundance.

4.4.2 Assessment of the Analytical Method

In order to investigate the area of validity of the FDT, we performed extensive simulations of our system (43). Figure 29 shows a number of panels comparing the noise determined analytically to explicit stochastic simulations. Throughout the tested parameters, the error between simulation and the analytic result is negligible.

According to Figure 29, the accuracy of FDT can be reduced when the regulation threshold K is close to one of the input steady states (x_L or x_H). In this case, small fluctuations in the input generate high fluctuations of the output. FDT assumes that the average stochastic behaviour equals the deterministic one, but in this case it does not, and, thus, FDT fails to accurately describe the behaviour of the system.

4.4.3 Discussion

In this thesis, we considered that the cell is essentially a perfectly mixed reactor. This assumption is necessary to keep the mathematics tractable, and is commonly made. We expect that the analytical formula of the noise to be broadly valid in spatially extended systems as well [118]. However, this is not addressed here.

Our model of the noise ignores two sources of noise: (*i*) regulation noise and (*ii*) post-transcriptional sources of noise, especially translation. The justification for disregarding the former is that we assume that the binding and unbinding

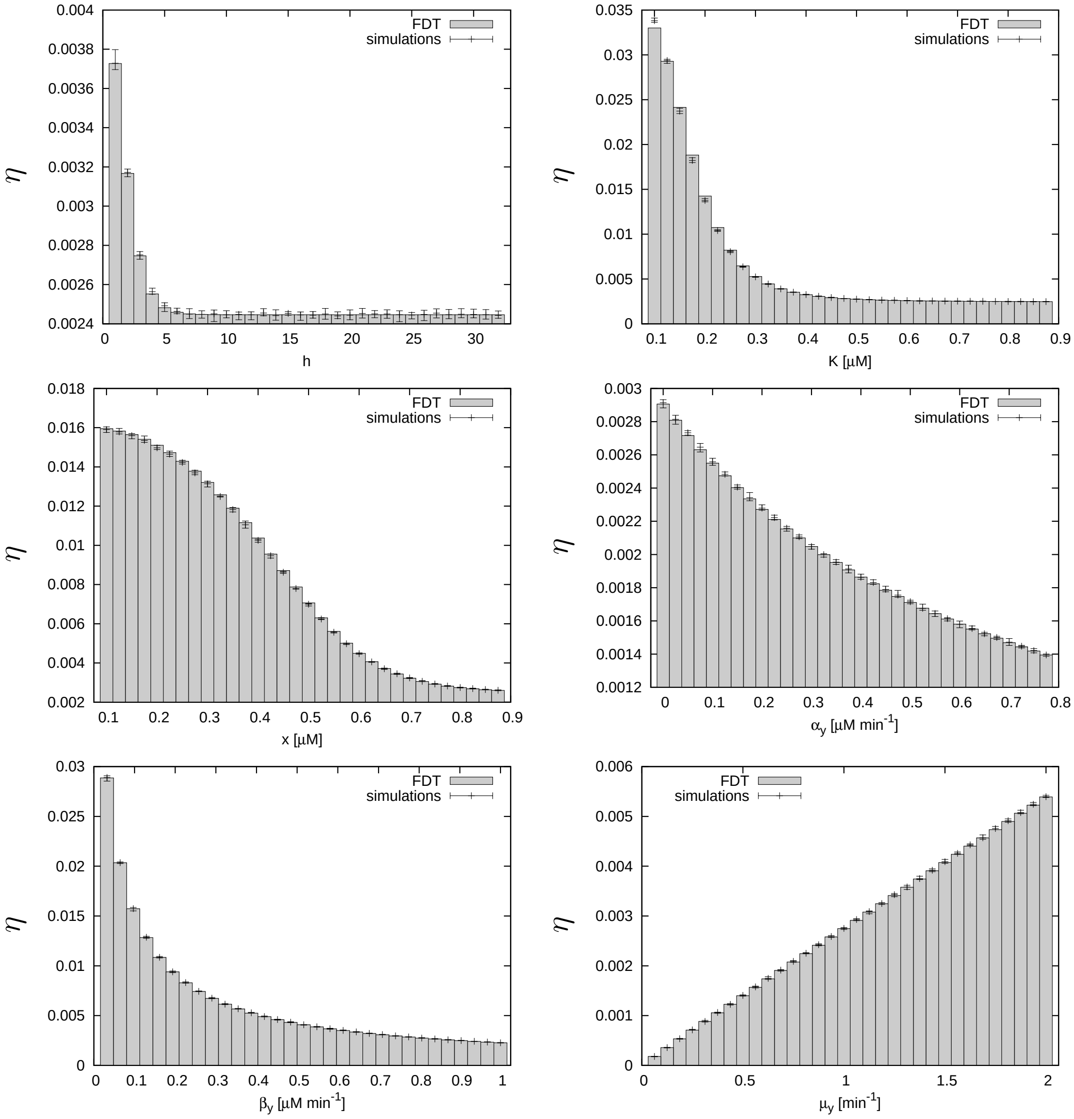


Figure 29: *Assessment of the FDT in the case of a binary gene.* We considered a cell volume of $V = 8 \cdot 10^{-16}\ l$ and an initial set of parameters: $K = 0.5\ \mu M$, $\mu = 1\ min^{-1}$, $\alpha = 0.04\ \mu M \cdot min^{-1}]$, $\beta = 0.81\ [\mu M \cdot min^{-1}]$, and $x = 0.2\ \mu M$. We varied individually each of these parameters: $h \in [1, 32]$, $K \in [0.1, 0.875]\ \mu M$, $x \in [0.1, 0.875]\ \mu M$, $\alpha \in [0, 0.775]\ \mu M min^{-1}$, $\beta \in [0.03, 1]\ \mu M \cdot min^{-1}$, $\mu_y \in [0.06, 2]\ [min^{-1}]$. The comparison between the variance normalized by square average number of molecules predicted by FDT matched the one observed in simulations. The error bars were computed by running 10 sets of simulations for $10^5\ min$.

dynamics of regulatory molecules to the operator site of the gene are very fast compared with the transcription process. This assumption is often valid in the case of gene regulation [106, 87, 82, 89]. Moreover, there is experimental evidence that mRNA production is indeed the dominant source of noise in the cell [122, 48, 69, 118, 13]. In particular, it seems that, typically, translation just scales the transcriptional noise [13], and thus does not directly alter the computational properties of gene regulation.

In addition, we do not consider explicitly extrinsic noise, the noise in other cellular components which affects all the genes in the cell equally [48]. However, due to the fact that this type of noise affects all genes equally it can be incorporated in the intrinsic component, i.e., both intrinsic and extrinsic noise can be summed up into the intrinsic component.

Finally, using stochastic simulations, we showed that, for most parameters, LNA/FDT estimates the noise accurately for the systems we consider (see subsection 4.4.2). We also found that when the Hill parameter K is close to x_H or x_L then the accuracy of the method suffers. This is expected since, in this case, the mean behaviour of the stochastic system may deviate from the behaviour of the deterministic system.

4.5 Summary

In this chapter, we presented the model of a switch built from a single gene, the binary gene. The switching mechanism is included in the *cis*-regulatory area of the gene where one (or more) transcription factors, the input(s), bind and change the rate at which the gene is expressed. The gene regulation function is usually modelled as a Hill function [2, 27, 36] which is described by two parameters: the Hill coefficient (which determines the steepness) and the threshold (which represents the required regulatory input for half activation of the gene). Depending on the Hill coefficient the graph of the function can be either a hyperbola (a Hill coefficient of 1) or sigmoid (a Hill coefficient higher than 1).

Logic gates require that the model displays a step-like behaviour. In section 4.3, we investigated how genes are turned on/off using a continuous-time Markov chain model. This model showed that the steady state probability that all the regulatory binding sites are occupied can be approximated by a Hill function. Most importantly, the model was able to make a connection between empirical parameters (Hill coefficient and threshold) and biological parameters (reaction rates and binding sites). The results revealed that, in the case when TFs bind to the DNA as monomers, the threshold can be approximated by the ratio between the unbinding and binding rates of individual molecules, while the maximum Hill coefficient equals the number of binding sites. This suggests that genes can have switch-like behaviour if they have more than one regulatory binding site, which is the case for many genes in bacterial cells [78].

In addition to the model of the genetic switch, in this chapter, we presented three properties of the system which we will use in our analysis: (*i*) metabolic cost, (*ii*) switching time and (*iii*) noise. We measure the metabolic cost as the maximum gene expression rate of the binary gene. This definition does not provide an exact quantitative measure of the actual metabolic cost, but rather determines its scaling properties. The additional factors that affect the actual metabolic cost are not important for our aim and only complicate the analytical argument.

If we assume that genes are able to compute, then we also need to consider the speed at which these biological components are able to process information. We consider the switching time of a gene as the time needed by the output to reach a new steady state when the regulatory input was changed.

Genes are usually affected by noise. In the context of gene networks, it is important that any downstream element distinguishes correctly between the two steady states of the output. Using the LNA, we were able to determine that the noise in a gene can be computed as the sum between the intrinsic component (resulted from random births/deaths) and the upstream one (propagated from the inputs). We also verified the accuracy of this analytical method (LNA) by performing extensive stochastic simulations and the results confirmed that the analytical method computes the noise with high precision. Finally, we observe that for biologically realistic parameters (from *E.coli* bacteria), the noise in the gene expression comes mainly from the transcription process, while translation just scales this noise. Moreover, the noise from the activation/inactivation of the genes becomes negligible for many biologically plausible parameters.

Chapter 5

Computational Limits to Binary Genes

In the previous chapter, we presented the model of a genetic switch, the binary gene. In addition, we also described and defined three parameters which characterise the system: metabolic cost, switching time and output noise. Here we perform an integrated optimality analysis which aims to reduce these three parameters simultaneously, if possible. The model presented in this chapter assumes that the input of the binary gene is changed instantaneously. Note that the case of non-instantaneous change of input will be considered in the next chapter.

5.1 Introduction

Genes which maintain a functional relationship between the concentration of regulatory input protein(s) and the concentration of the output protein can be thought of as capable of performing computations. In this chapter, we will probe some of the fundamental limitations on the computational capabilities of binary genes, i.e., regulated genes which can only be in one of the two activation states: a high state (corresponding to high concentration/particle number of the expressed protein) or a low state (corresponding to low concentration/particle number of the expressed protein).

As we saw in the previous chapter, binary genes have a metabolic cost associated with their expression process and are affected by output noise and computational lag. Each cellular process (protein production, protein decay and maintenance processes) has a metabolic cost attached to it, which is, usually, measured in number of ATP molecules [4]. Our notion of cost is not the exact measure of the actual metabolic cost, but rather a number which describes how the actual metabolic cost scales when the parameters of the binary gene are changed. It is essential to consider this parameter in our analysis due to the fact that the cost can be limited by the number of available resources and, thus, it cannot be increased arbitrarily.

Furthermore, gene expression is affected by noise. This noise is a consequence of the fact that genes have low copy numbers and that they are slowly expressed [87]. In the context of binary genes, this output noise is undesirable because it makes difficult the assessment of the output of the gene as either low (0) or high (1).

Finally, we want to perform computations as fast as possible, but genes are very slow, in the sense that the time required to turn on/off a gene (the switching time) is in the order of tens of minutes, even for an instant input change. In this chapter, we considered that the abundance of the regulatory input is changed instantaneously, aiming to identify approaches to reduce the switching time.

Our results show that, under fixed metabolic cost, the noise can be reduced only by slowing down the gene. This suggests that, under limited resources (metabolic cost), a gene can be fast or accurate, but not both at the same time. We also observed that genes with higher metabolic cost display better trade-off curves compared with genes with lower metabolic cost.

Additionally, we proved that any leak-free system (a gene without basal rate) is optimal in terms of speed and noise; it will display better noise/speed trade-off curves than any system with non-vanishing leak rate and equal metabolic cost. However, systems with non-vanishing leak rates have a more favourable scaling behaviour than leak-free systems, in the sense that for a given increase of the cost, leak systems show a more pronounced decrease of noise than leak-free systems. Hence, leak-free systems are optimum under a fixed metabolic cost, but they are less efficient in noise reduction by cost increase compared with non-vanishing leak systems.

In the next section, we will give a brief overview of the model of the binary gene and the mathematical definitions of the three computational properties of the system which we use in our analysis, namely switching time, metabolic cost and noise. In section 5.3 we perform an optimality analysis and identify the interconnection between these parameters. Finally, we draw the conclusions based on our results.

5.2 The Model of the Binary Gene

As a reminder from the previous chapter, we will briefly present the model and the properties which characterise the binary gene. Our model of the binary gene is given by

$$\emptyset \xrightarrow{\alpha+\beta f(x)} y, \quad y \xrightarrow{\mu} \emptyset, \tag{67}$$

where protein y is synthesised with leak rate α and maximal rate $\alpha+\beta$. The synthesis rate of y is controlled by transcription factor x through the Hill regulation function, $f(x)$, which can be written as

$$\phi(x) = \frac{x^h}{K^h + x^h} \qquad \text{and} \qquad \bar{\phi}(x) = \frac{K^h}{K^h + x^h}. \tag{68}$$

The system defined in equation (67) is described by the following differential equation

$$\frac{dy}{dt} = \alpha + \beta f(x) - \mu y, \tag{69}$$

where x can take one of the two values: x_L (resulting in $y = L$) or x_H (resulting in $y = H$).

Next, we will describe again, but this time succinctly, the three properties of the system required by our optimality analysis: metabolic cost, switching time and output noise.

We measure the metabolic cost of the system as the maximum synthesis rate,

$$\zeta = \alpha + \beta f(x_H). \tag{70}$$

Furthermore, in the case of instantaneous input change, the switching time, T_{gene}, (the time required to reach a fraction θ of the steady state) is given by

$$T_{\text{gene}} = \frac{1}{\mu} \ln \left(\frac{1}{1-\theta} \right). \tag{71}$$

The only parameter which influences this switching time is the decay rate of the product, in the sense that higher decay rates generate faster responses. This suggests an immediate mechanism to reduce the switching time, namely to increase the decay rate μ as much as possible. Note that doing this also decreases the signal strength, $H - L = \beta[f(x_H) - f(x_L)]/\mu$.

In the deterministic system small signal strengths do not pose any problems and do not limit the computational usefulness of the gene. Thus, we can conclude from equation (71) that the switching time can be decreased (and hence the computing speed increased) arbitrarily, as long as the gene output is noise free and varies continuously.

However, genes are affected by noise and, consequently, the signal strength cannot be reduced arbitrary. In the previous chapter we showed that the noise of a gene regulated by one transcription factor can be written as

$$\eta = \underbrace{\frac{H}{(H-L)^2}}_{\eta_{\text{in}}} + \underbrace{\overbrace{\left[\frac{\beta f'(x_H)}{H-L} \tau_y \right]^2}^{\text{regulation factor}} \overbrace{\frac{\tau_x}{\tau_x + \tau_y}}^{\text{time factor}} \sigma_x^2}_{\eta_{\text{up}}}. \tag{72}$$

Note that variance is normalized by the square of the signal strength (see previous chapter for more details). The noise of the binary gene consists of two components:

the intrinsic noise, η_{in}, (generated by the randomness of the birth/death process) and the upstream noise, η_{up}, (propagated from the upstream component) [48, 151, 123, 19, 129, 152].

5.3 Noise, Time and Cost

So far, we have defined the noise, switching time and metabolic cost as independent parameters of the system. In order to perform a complete analysis on a binary gene, we also investigated the functional relationship between these three properties (noise, switching time and metabolic cost). We start this *optimality analysis*, by assuming the ideal case of no leak rate, i.e., $\alpha = 0$ and we will further assume that $L = 0$. Following equation (72), the noise can be written as

$$\eta = \frac{1}{H} + \left[\frac{f'(x_H)}{f(x_H)}\right]^2 \frac{\tau_x}{\tau_x + \tau_y}\sigma_x^2. \tag{73}$$

We consider the scaling of several parameters of the model.

Scaling of β

First, we assume that the production rate scales by a factor of γ, i.e.,

$$\beta \longrightarrow \beta' = \gamma\beta.$$

Note that scaling β by γ we also scale the metabolic cost by the same factor,

$$\zeta_{\beta'} = \beta' f(x_H) = \gamma\beta f(x_H) = \gamma\zeta_\beta.$$

Here, the subscript β indicates that the cost is generated by a system with production rate β. The average particle number at the high state H also scales by the same factor,

$$H_{\beta'} = \frac{\beta' f(x_H)}{\mu} = \gamma\frac{\beta f(x_H)}{\mu} = \gamma H_\beta.$$

According to equation (73), scaling β leaves the upstream noise, η_{up}, unaffected. However, the intrinsic noise, η_{in}, scales with γ^{-1}. Hence, if we assume that the binary gene has no leak rate and that the production rate scales by a factor γ, we have the following relationship between noise, time and cost:

$$\zeta \sim \gamma \quad \text{and} \quad T_{\text{gene}} = \text{constant} \quad \Rightarrow \quad \eta_{\text{in}} \sim \frac{1}{\gamma} \quad \text{and} \quad \eta_{\text{up}} = \text{constant}. \tag{74}$$

As expected, increasing the production rate of a gene product increases the cost, but reduces the noise correspondingly. This suggests a noise-metabolic cost tradeoff. Note that by scaling only the synthesis rate β, the switching time remains unaffected (see equation 71).

Scaling of $\boldsymbol{\tau}$

Next, we scale the average lifetime of the output by a factor γ,

$$\tau_y \longrightarrow \tau_y' = \gamma \tau_y.$$

The average life time of a species is inversely proportional to the decay rate,

$$\tau_y = \frac{1}{\mu} \Rightarrow \mu \longrightarrow \mu' = \frac{1}{\gamma}\mu.$$

The switching time and τ_y scale in the same direction,

$$T_{\tau_y'} = \tau_y' ln\left(\frac{1}{1-\theta}\right) = \gamma\tau_y ln\left(\frac{1}{1-\theta}\right) = \gamma T_{\tau_y}.$$

Note that the subscript indicates the system with the corresponding average lifetime. Analogously, using the steady state equation, one can observe that the H and μ scale in opposite directions,

$$H_{\tau_y'} = \frac{\beta f(x_H)}{\mu'} = \gamma\frac{\beta f(x_H)}{\mu} = \gamma H_{\tau_y}.$$

Scaling the average lifetime leads to an overall change of the noise of G_y, even at constant cost ζ, as follows

$$\eta = \frac{1}{\gamma H} + \Gamma_{yx}^{L=0} \frac{\tau_x}{\tau_x + \gamma\tau_y} \sigma_x^2 \Big|_{\zeta = const} . \tag{75}$$

Here Γ_{yx} is defined as:

$$\Gamma_{yx} = \left[\frac{\beta f'(x_H)}{H - L} \tau_y \right]^2 \Rightarrow \Gamma_{yx}^{L=0} \left[\frac{f'(x_H)}{f(x_H)} \right]^2 .$$

In the case of no leak rate, $\Gamma_{yx}^{L=0}$ does not depend on the average lifetime τ_y.

Increasing the average lifetime makes the system slower (increases the switching time), but, at the same time, reduces the noise of the system. In the case in which the average life time of the output protein scales by a factor γ, we can summarise the relationship between noise, time and cost as follows:

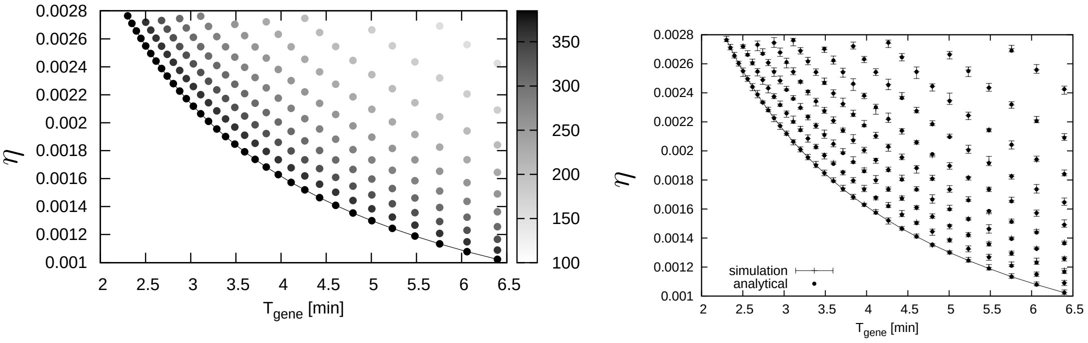
$$T_{\text{gene}} \sim \gamma \quad \text{and} \quad \zeta = \text{constant} \quad \Rightarrow \quad \eta_{\text{in}} \sim \frac{1}{\gamma} \quad \text{and} \quad \eta_{\text{up}} \sim \frac{1}{\nu}. \tag{76}$$

Note that we do not scale β or $f(x_H)$ and, thus, the cost remains constant. We denoted by ν the fraction

$$\nu = \frac{\tau_x + \tau_y}{\tau_x + \gamma\tau_y},$$

which is inverse proportional to γ, i.e., it decreases when γ increases and, conversely, it increases when γ decreases.

Equation (76) presents the noise-speed trade-off at fixed metabolic cost. Figure 30 illustrates this trade-off for various costs. For a fixed metabolic cost, the noise can be reduced only by increasing the switching time. Conversely, the switching time can be reduced only by increasing the noise, in the case of fixed metabolic cost. This suggests that, under fixed cost, the accuracy and the speed of the gene are inverse proportional, i.e., they cannot be increased simultaneously. In addition, we observed that a higher cost ensures better trade-off curves (lower noise and switching time) compared with a lower one (see Figure 30).



(a) analytical

(b) simulations

Figure 30: *Relation between output noise, switching time, and metabolic cost of the binary gene.* The metabolic cost of each point is indicated by a shade of grey. The equal cost points fall onto a line. Different costs were achieved by varying the maximum production rate of β. The following set of parameters was used: $x = 0.2\ \mu M$, $\alpha = 0$, $K = 0.5\ \mu M$ and $h = 3$. The degradation rate was varied in interval $\mu \in [0.36, 1]\ min^{-1}$ and the synthesis rate in interval $\beta \in [0.34, 0.85]\ \mu M \cdot min^{-1}$. The time was computed for $\theta = 0.9$. We considered a cell volume of $V = 8 \cdot 10^{-16}\ l$. In both figures we assumed that y is repressed by x. In (a) we plotted the results obtained from the analytic solution. We found close agreement between these analytical results and the results from stochastic simulations (b). The error bars were computed by running 10 sets of stochastic simulations (Gibson-Bruck [60]), each for $10^5\ min$.

We will show below that, under fixed metabolic cost, equation (76) describes a theoretical computational performance limit of the gene at fixed cost, in the sense that the accuracy and speed characteristics cannot be simultaneously improved, without also increasing the metabolic cost.

5.3.1 Noise in the Case of Non-Vanishing Leak Expression

We will now relax the assumption of no leak rate ($\alpha = 0$) and consider a non-vanishing α. A consequence of this is that $L > 0$. For any fixed value of α there exists a (non-optimal) set of noise-speed trade-offs at fixed cost. This can be obtained by modifying equation (75) as follows

$$\eta = \frac{H}{\gamma(H-L)^2} + \Gamma_{yx}^{L>0} \frac{\tau_x}{\tau_x + \gamma\tau_y} \sigma_x^2 \bigg|_{\zeta=const}. \tag{77}$$

The main difference between equations (75) and (77) is the first term (intrinsic noise). In the case of no leak rate ($L = 0$), the intrinsic noise reduces to $1/\gamma H$. For a given cost ζ, the set of possible noise-speed trade-offs with $\alpha > 0$ is worse than the optimal noise-speed trade-off set in the sense that for a fixed noise, the speed is always lower in the leak-free system, and conversely. We illustrate this for some parameters in Figure 31.

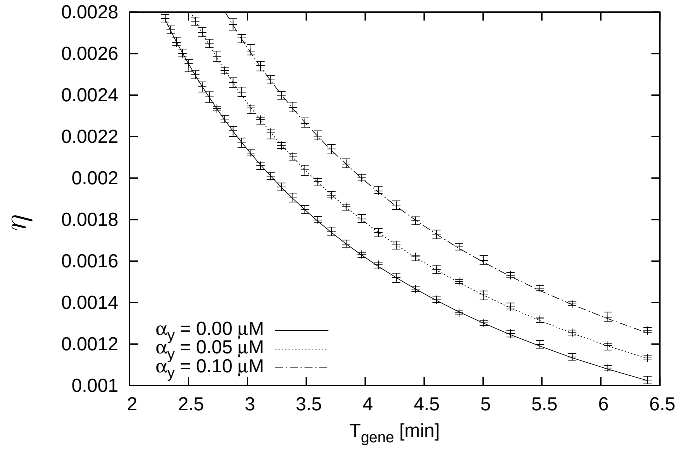


Figure 31: *Comparison of various α at fixed metabolic cost.* We plot the speed and noise trade-off for different leak rates under fixed metabolic cost. The metabolic cost of the three curves is constant, $\zeta = 0.8\ \mu M \cdot min^{-1}$. We consider three cases: $\alpha = 0\ \mu M \cdot min^{-1}$, $\alpha = 0.05\ \mu M \cdot min^{-1}$ and $\alpha = 0.10\ \mu M \cdot min^{-1}$. The following set of parameters was used: $x = 0.2\ \mu M$, $K = 0.5\ \mu M$ and $h = 3$. The degradation rate was varied in interval $\mu \in [1, 0.36]\ [min^{-1}]$. We considered a cell volume of $V = 8 \cdot 10^{-16}\ l$. We assumed the repression case $f(x) = \bar{\phi}(x)$. The error bars were computed by running 10 sets of stochastic simulations (Gibson-Bruck [60]), each for $10^5\ min$.

The sub-optimality of $\alpha > 0$ can be seen directly from the expression of the noise (equation 72) as follows: The upstream noise is unaffected by α, so we only need to consider the intrinsic noise given by:

$$\eta_{\text{in}} = \frac{\alpha + \beta f(x_H)}{(\beta f(x_H) - \beta f(x_L))^2} \frac{\tau_y}{\tau_y^2}. \tag{78}$$

From this equation, we see that increasing the leak rate, increases the intrinsic noise. At the same time, increasing α also increases the cost; this follows from the definition of the metabolic cost in equation (70). Altogether, this shows that any binary gene with non-vanishing α is sub-optimal with respect to its noise and cost characteristics.

Scaling of α while ζ is constant

Next, we investigate what happens at fixed cost when the leak rate is increased. Thus, we scale the leak rate by a factor γ, but at the same time we keep the metabolic cost fixed at $\zeta = \alpha + \beta f(x_H)$,

$$\alpha \longrightarrow \alpha' = \gamma\alpha \quad \text{and} \quad \beta' = \frac{\zeta - \gamma\alpha}{f(x_H)}.$$

Replacing α' and β' into equation (78) gives

$$\eta_{\text{in}} = \frac{\gamma\alpha + \frac{\zeta-\gamma\alpha}{f(x_H)} f(x_H)}{\left[\frac{\zeta-\gamma\alpha}{f(x_H)}\right]^2 [f(x_H) - f(x_L)]^2} \frac{1}{\tau_y} = \frac{\zeta}{(\zeta - \alpha\gamma)^2 \left[\frac{f(x_H)-f(x_L)}{f(x_H)}\right]^2} \frac{1}{\tau_y}. \tag{79}$$

From equation (79) we can see that, by increasing α ($\gamma > 1$) and keeping everything else constant, the intrinsic noise (η_{in}) increases.

Scaling of β while $\alpha \geq 0$

We now consider how the noise scales with β when $\alpha > 0$. If β is scaled by γ and $\alpha = 0$, then the corresponding change of the total cost will inversely scale the noise. When there is a leak expression, that is $\alpha > 0$, then this is no longer the case. Instead, if β scales by a factor γ, then the cost will scale by a different factor,

$$\beta' = \gamma\beta \Rightarrow \zeta_{\beta'} = \delta\zeta_\beta,$$

where δ is given by

$$\delta = \frac{\alpha + \gamma\,\beta f(x_H)}{\alpha + \beta\, f(x_H)}.$$

Solving for γ, we obtain,

$$\gamma = \frac{(\delta - 1)\alpha + \delta\beta f(x_H)}{\beta f(x_H)}.$$

Considering that the intrinsic noise is given by,

$$\eta_{\text{in}} = \frac{\alpha + \gamma\beta f(x_H)}{[\gamma\beta(f(x_H) - f(x_L))]^2\tau_y}, \tag{80}$$

we see that the intrinsic noise scales as

$$\eta_{\text{in}}^{\beta'} = \frac{\delta\beta^2 f(x_H)^2}{\left[-\alpha + \delta\left(\alpha + \beta f(x_H)\right)\right]^2}\eta_{\text{in}}^{\beta}, \tag{81}$$

whereas the total cost scales as $\zeta_{\beta'} = \delta\zeta_\beta$. If we assume that $\delta > 1$, i.e., we increase the cost, then the scaling term is smaller than $1/\delta$,

$$\frac{\delta\beta^2 f(x_H)^2}{\left[-\alpha + \delta\left(\alpha + \beta f(x_H)\right)\right]^2} \leq \frac{\delta\beta^2 f(x_H)^2}{\left[\delta\beta f(x_H)\right]^2} = \frac{1}{\delta}.$$

This implies that the actual decrease in noise is more than the inverse of the increase in metabolic cost. For $\alpha = 0$, this scaling factor reduces to $1/\delta$, which in turn reduces to $1/\gamma$, thus recovering the above scaling from equation (74). This scaling relation implies that systems with $\alpha > 0$ have a more favourable scaling behaviour than leak-free systems, in the sense that for a given increase of the cost, leak systems show a more pronounced decrease of noise than leak-free systems (see Figure 32). A corollary of this is that for increasing expression rates the systems with and without leak become more similar.

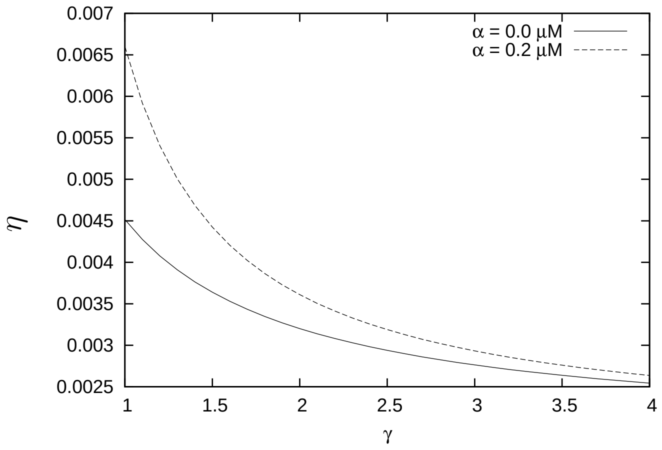


Figure 32: *Noise reduction sensitivity to cost increase.* We plot the noise as a function of cost increase. On the x-axis we represented γ, the scaling factor of β. We consider two cases: $\alpha = 0 \ \mu M \cdot min^{-1}$ and $\alpha = 0.2 \ \mu M \cdot min^{-1}$. The following set of parameters was used: $x = 0.2 \ \mu M$, $K = 0.5 \ \mu M$, $h = 3$ and $\beta = 1.08 \ \mu M \cdot min^{-1}$. The degradation rate was varied in interval $\mu \in [1, 0.36]$ $[min^{-1}]$. We considered the repression case $f(x) = \bar{\phi}(x)$ and a cell volume of $V = 8 \cdot 10^{-16}$ l.

5.3.2 General Case

Previously, we stated that increasing the metabolic cost generates better noise-time trade-off curves. In order to check the validity of this statement, we investigate analytically whether increasing the cost can lead to simultaneous reduction of both noise and switching time. We consider the case when both α and β scale by γ and, at the same time, the average life time of the output species scales by δ,

$$\alpha \longrightarrow \alpha' = \gamma\alpha \quad , \quad \beta \longrightarrow \beta' = \gamma\beta \quad \text{and} \quad \tau_y \longrightarrow \tau'_y = \delta\tau_y.$$

from equations (70) and (71), we notice that the metabolic cost and the switching time scale as

$$\zeta \longrightarrow \zeta' = \gamma\zeta \quad \text{and} \quad T \longrightarrow T' = \delta T.$$

The output steady states will then scale by $\gamma\delta$

$$y' = \frac{\alpha' + \beta' f(x)}{\mu'} = \frac{\gamma\alpha + \gamma\beta' f(x)}{\mu'/\delta} = \gamma\delta\frac{\alpha + \beta f(x)}{\mu} = \gamma\delta y.$$

	$\delta < 1$	$\delta = 1$	$\delta > 1$
$\gamma < 1$	T_{gene}: ($\searrow$) η_{in}: ($\nearrow$) η_{up}: ($\nearrow$) ζ: ($\searrow$)	T_{gene}: ($=$) η_{in}: ($\nearrow$) η_{up}: ($=$) ζ: ($\searrow$)	T_{gene}: ($\nearrow$) η_{in}: $\begin{cases} (\nearrow),\ \delta < 1/\gamma \\ (=),\ \delta = 1/\gamma \\ (\searrow),\ \delta > 1/\gamma \end{cases}$ η_{up}: ($\searrow$) ζ: ($\searrow$)
$\gamma = 1$	T_{gene}: ($\searrow$) η_{in}: ($\nearrow$) η_{up}: ($\nearrow$) ζ: ($=$)	T_{gene}: ($=$) η_{in}: ($=$) η_{up}: ($=$) ζ: ($=$)	T_{gene}: ($\nearrow$) η_{in}: ($\searrow$) η_{up}: ($\searrow$) ζ: ($=$)
$\gamma > 1$	T_{gene}: ($\searrow$) η_{in}: $\begin{cases} (\nearrow),\ \delta < 1/\gamma \\ (=),\ \delta = 1/\gamma \\ (\searrow),\ \delta > 1/\gamma \end{cases}$ η_{up}: ($\nearrow$) ζ: ($\nearrow$)	T_{gene}: ($=$) η_{in}: ($\searrow$) η_{up}: ($=$) ζ: ($\nearrow$)	T_{gene}: ($\nearrow$) η_{in}: ($\searrow$) η_{up}: ($\searrow$) ζ: ($\nearrow$)

Table 1: *Noise, time and cost.* We considered three cases for δ (the scaling of the average life time of the protein) and γ (the scaling of the synthesis rates): < 1, $= 1$, and > 1. We use three symbols to represent the behaviour of the three properties (noise, η_{in} and η_{up}; time T and cost ζ): it decreases ($\searrow$), it remains constant ($=$) and it increases ($\nearrow$).

Considering all these scaling terms, the equation of noise becomes

$$\eta = \frac{H}{\gamma\delta(H-L)^2} + \left[\frac{f'(x_H)}{f(x_H)-f(x_L)}\right]^2 \frac{\tau_x}{\tau_x+\delta\tau_y}\sigma_x^2. \tag{82}$$

Table 1 displays the system behaviour when the cost and time are scaled simultaneously. We noticed from the table that a higher metabolic cost, $\gamma > 1$, can lead to a simultaneous decrease in the propagation time and intrinsic noise. However, for this to happen, the average life time of the output species needs to be decreased ($1 > \delta$), and this decrease needs to be slower than the increase in the metabolic cost ($\delta > 1/\gamma$).

In addition, from Table 1, it is easy to observe that no scaling combination can ensure a simultaneous enhancement in all three properties (noise, time and cost). This suggests that, in order to improve any property of the system, at least one other property needs to be worsened. The last statement defines the three-way trade-off between noise, time and cost of a binary gene. Note that, when the cost is kept fixed, the table shows that there is no scaling which improves speed and accuracy simultaneously.

5.3.3 Noise and the Regulation Threshold

In the case of repressed genes (i.e., $f \equiv \bar{\phi}$), non-zero low states ($L > 0$) can be generated due to incomplete repression (even when $\alpha = 0$). It is clear from the shape of the repression function $\bar{\phi}$ that complete repression can only be achieved in the limiting cases of either an infinite number of repressor molecules or an infinite Hill coefficient h. Neither is biologically realisable. If we assume the input signal (that is x_H and x_L) to be fixed, then the leak rate will depend on the Hill parameter K. Physically, this parameter is, in essence, the fraction of the association and dissociation rate constants of the regulatory protein to/from the specific binding site of the operator. It is evident that, the lower the K, the fewer molecules are required to achieve a certain level of repression. The value of K which maximises signal strength ($H - L$) is given by $K^* = \sqrt{x_H x_L}$. This K^* is not identical to the value of K that optimises noise, which is given by the solution to

$$\frac{d}{dK}\eta = 0, \tag{83}$$

where K is margined by the two steady states of the input (x_L and x_H), i.e., $x_L < K < x_H$ in the activation case or $x_H < K < x_L$ in the repression case.

The corresponding formula is too complicated to be useful, but can be calculated numerically; it will typically be similar, but not equal, to the value that optimises signal strength. Moreover, a numerical analysis suggests that, around the optimal value of K, the noise depends only very weakly on K, particularly when the signal strength of the input, $|x_L - x_H|$, is large (see Figure 33).

This result was recently supported by experimental evidence. Murphy *et al.* [115] showed that performing point mutations on the promoter (more specifically

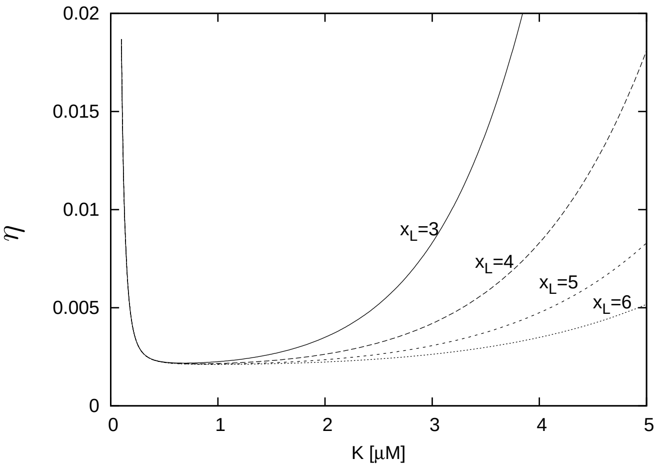


Figure 33: *The noise as a function of K in the repressor case.* We used the following set of parameters ($h = 3$, $x_H = 0.2$ μM, $\tau_x = 1$ min, $\tau_y = 1$ min, $\sigma_x^2 = 96$). We considered a volume of $V = 8 \cdot 10^{-16}$ l.

in the TATA box) can lead to a shift of the threshold without affecting other parameters. By varying the threshold, the noise changes only slightly. Nevertheless, they noticed that there is an intermediary threshold position which minimises the noise level.

5.3.4 Noise and the Hill Coefficient

To understand the dependence of noise on h it is necessary to consider $f \equiv \phi$ and $f \equiv \bar{\phi}$ separately. Since we always consider the noise at $y = H$, the scaling relation of the repressor needs to be evaluated at very low particle numbers of x, whereas the noise of the activator needs to be evaluated at high x.

Repression

We start with the repression case, $f \equiv \bar{\phi}$. In this case, the intrinsic noise is given by:

$$\begin{aligned} \eta_{\text{in}} &= \frac{H}{(H-L)^2} = \frac{\alpha + \beta\frac{K^h}{x_H^h+K^h}}{\beta^2}\left(\frac{K^h}{x_H^h+K^h} - \frac{K^h}{x_L^h+K^h}\right)^{-2}\frac{1}{\tau_y} \\ &\approx \left[\alpha + \beta\frac{K^h}{K^h+x_H^h}\right]\left[\beta\left(1-\frac{K^h}{x_L^h}\right)\right]^{-2}\frac{1}{\tau_y} \\ &\approx (\alpha+\beta)\left[\beta\left(1-\frac{K^h}{x_L^h}\right)\right]^{-2}\frac{1}{\tau_y} \\ &\approx \frac{\alpha+\beta}{\tau_y\beta^2}. \end{aligned} \tag{84}$$

We used the following approximation: $x_H \ll K$ and $x_L \gg K$. Similarly, if C summarises factors in the upstream noise which are not affected by h, then the upstream noise scales like

$$\begin{aligned} \eta_{\text{up}} &= C\left(\frac{K^h h x_H^{h-1}}{(K^h+x_H^h)^2}\right)^2\left(\frac{K^h}{x_H^h+K^h} - \frac{K^h}{x_L^h+K^h}\right)^{-2} \\ &\approx Ch^2\left(\frac{x_H^{h-1}}{K^h}\right)\left(1-\frac{K^h}{x_L^h+K^h}\right)^{-2} \\ &\approx Ch^2\left(\frac{x_H^{h-1}}{K^h}\right)^2. \end{aligned} \tag{85}$$

Both the intrinsic and upstream noise are decreasing functions of h; however, the equations indicate that, for higher values of h, the slope of both equation (84) and equation (85) approaches 0 relatively rapid. Hence, with increasing h, the dependence of the noise on h becomes increasingly weaker. Altogether, we obtain the following equation for the noise as a function of h (see Figure 34)

$$\eta_{\bar{\phi}} \approx \frac{\alpha+\beta}{\tau_y\beta^2} + Ch^2\left(\frac{x_H^{h-1}}{K^h}\right)^2. \tag{86}$$

Activation

Using analogous approaches, but now assuming that $x_H \gg K$, $x_L \ll K$ and $f \equiv \phi$, we obtain the following expression for the intrinsic noise

$$\eta_{\text{in}} \approx \frac{\alpha + \beta}{\tau_y \beta^2}. \tag{87}$$

The upstream noise can be written as

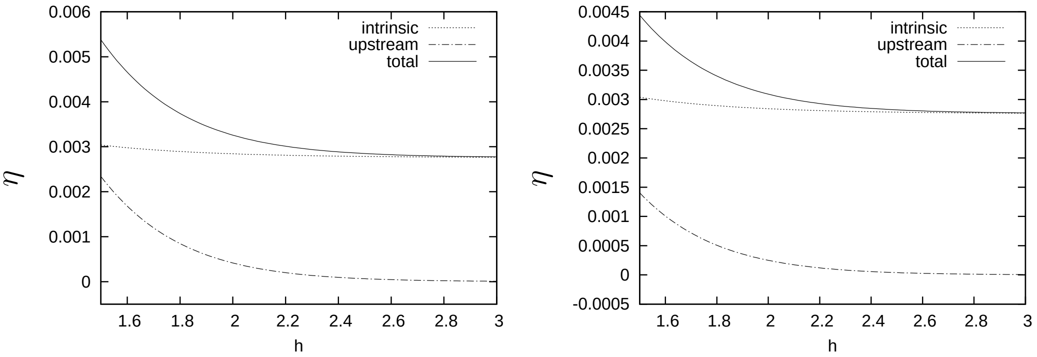
$$\begin{aligned}
\eta_{\text{up}} &= C \left[\frac{h x_H^{h-1}}{K^h + x_H^h} \left(1 - \frac{x_H^h}{K^h + x_H^h} \right)^{-1} \right]^2 \left[\frac{x_H^h}{K^h + x_H^h} - \frac{x_L^h}{K^h + x_L^h} \right]^{-2} \\
&\approx Ch^2 \left(\frac{K^h}{x_H^{h+1}} \right)^2.
\end{aligned} \tag{88}$$

Hence, in the activation case, the noise depends on h as follows:

$$\eta_\phi \approx \frac{\alpha + \beta}{\tau_y \beta^2} + Ch^2 \left(\frac{K^h}{x_H^{h+1}} \right)^2. \tag{89}$$

Figure 34 illustrates how the noise depends on h. For a specific example, the graph suggests that improvements in the noise for increased h diminish fast as h increases. This diminishing effect of h can be seen directly from equations (86) and (89). In both equations, in the second terms on the right hand side, the factors next to h^2 are very small and decrease with h faster than with h^2; C summarizes terms in the noise equation that do not change with h. Hence, the overall upstream noise tends to zero as h increases.

This begs the question regarding the metabolic cost of increasing h. In this context it is worthwhile noting that increasing the Hill coefficient leads to an almost linear increase in the metabolic cost [184]. This increase in the metabolic cost combined with the diminishing efficiency in noise reduction suggests that there is an optimal Hill coefficient beyond which further increase is not cost effective any more.



(a) repression

(b) activation

Figure 34: *The noise as a function of h.* We used the following set of parameters: $K = 0.5\ \mu M$, $\tau_x = 1\ min$, $\tau_y = 1\ min$, $\alpha = 0.05\ \mu M \cdot min^{-1}$ and $\beta = 0.8\ \mu M \cdot min^{-1}$. We use a cell volume of $8 \cdot 10^{-16} l$. In the repression case we considered: $x_H = 0.05\ \mu M$, $x_L = 5\ \mu M$ and $\sigma_x^2 = 1.204 \times 10^3$; and in the activation one $x_H = 5\ \mu M$, $x_L = 0.05\ \mu M$ and $\sigma_x^2 = 7.22655 \times 10^6$. Note that we used a different noise input in order to ensure a good visibility of the graphs.

5.4 Summary

In an ideal, deterministic system, binary genes could be driven at an arbitrary speed without increasing the metabolic cost. However, real systems are afflicted by noise and this imposes strict limits on the computational efficiency of genes. We identified a three-way trade-off between the output noise of a gene, its switching time and the metabolic cost necessary to maintain it.

For a fixed metabolic cost, there is an accuracy-speed trade-off, in the sense that the speed can be increased only by decreasing the accuracy and vice versa. Equations (74) and (76) define this trade-off analytically. Figure 30 illustrates ideal trade-off sets for $\alpha = 0$ and various costs (indicated by the shades of grey of the points). Figure 31, on the other hand, shows that the trade-off set for $\alpha > 0$ has worse noise-time characteristics than the leak-free system. A non-vanishing α is sub-optimal; see equation (78). However, it is cheaper (in terms of metabolic cost) to improve the sub-optimal system ($\alpha > 0$) compared with the optimal one ($\alpha = 0$); see equations (78) and (81).

An extensive map of the three-way trade-off between noise, time and cost is presented in Table 1. The table shows that there is no combination which ensures the enhancement of all the three properties (noise, time and cost) simultaneously. Moreover, in order to enhance one of these three properties at least another one needs to be worsened. The table also shows that, under certain conditions (reduction of switching time, but slower than the increase in metabolic cost), noise and time can be enhanced simultaneously when the metabolic cost is increased.

Finally, we showed that there is an optimal regulation threshold, K, which minimises the output noise; see equation (83) and Figure 33. Similarly, we showed that increasing the Hill coefficient reduces the noise (see equations (86) and (89) and Figure 34). However, the efficiency of noise reduction is attenuated really fast. This in conjunction with the fact that increasing h increases the metabolic cost suggests that there is an optimal Hill coefficient. Hence, in a system of binary genes, there are optimal values for the parameters α, K and h, whereas there is a trade-off for β and μ.

Chapter 6

Optimality Analysis of Binary Genes

In chapter 5, we investigated the trade-off between the speed and accuracy of a binary gene under the assumption of instantaneous input change. Here we extend this analysis and consider the case of non-instantaneous input change. We developed an optimality analysis which defined a trade-off between speed and accuracy under the assumption of fixed metabolic cost. Furthermore, we examined whether at least one of the two properties (speed and accuracy) can be enhanced by negative auto-regulation without worsening the other one.

6.1 Introduction

Under the assumption of fixed metabolic cost, binary genes (genes which have two expression levels, high and low) regulated by inputs which change instantaneously display a trade-off between the speed at which the output changes states and the accuracy of the output at steady state. This trade-off is controlled by the decay rate, in the sense that lower decay rates ensure slower switching, but also higher accuracy, and vice versa. Instantaneous input change happens rarely within biological systems, where, for example, the input can be subject to exponential decay due to dilution. Thus, in this chapter we will analyse the three way trade-off between switching time, output noise and metabolic cost under the assumption of a non-instantaneous change in the input.

Our results revealed that, based on the regulatory threshold position, a binary gene displays different speed-accuracy configurations. In addition, the binary gene is characterised by two specific threshold positions: one which optimises the system in speed and another, which optimises the system in terms of accuracy. The analysis shows that there is an optimal trade-off curve between speed and accuracy, which is controlled by the position of the regulation threshold. Moreover, this optimal trade-off curve is delimited by the two values that optimise the system in terms of speed and accuracy. Points that reside outside this optimal trade-off curve are sub-optimal because they worsen the system in both speed and accuracy.

It was previously postulated that negative auto-regulation can enhance the speed [137, 5] and that in some cases it can reduce noise [20, 83, 84, 31]. Here, we investigated the speed and accuracy properties of a negatively auto-regulated gene systematically and found that, for low but non-vanishing leak rates, the negative auto-regulated system outperforms the simple binary gene in both speed and accuracy. In addition, for vanishing leak rates, the system is enhanced only in accuracy and worsened in speed while, for high values of the leak rate, the system displays higher noise and faster switching. Note that low leak rates are easy to achieve in the case when the gene is activated by a regulatory input. When the binary gene is repressed by the regulatory input, low leak rates require either a high Hill coefficient or high input abundance, both of which increase the metabolic cost [184].

We start this chapter by presenting the model of the system which we analyse and how we measure its properties (cost, speed and accuracy). Then, in section 6.3, we present the optimality analysis of a simple binary gene. Furthermore, in section 6.4 we will analyse whether negative auto-regulation can enhance the system in either speed or accuracy without worsening the other property. Finally, we investigate how this analysis can be applied to biological data and then we draw some conclusions.

6.2 The Model of the Binary Gene

We use the same model as in our previous chapters, where a gene G_y has an output y which is regulated by a single transcription factor x (see Figure 20 from chapter 4). This model of the binary gene is described by the following set of chemical reactions

$$\emptyset \xrightarrow{\alpha+\beta f(x)} y, \quad y \xrightarrow{\mu} \emptyset, \tag{90}$$

where α is the leak rate, $\alpha + \beta$ the maximum production rate and μ the decay rate. The regulation function, $f(x)$, is usually approximated by a family of sigmoid functions, namely the Hill functions [2, 27, 36]. As previously, we consider that the regulation function can be either the activator Hill function ($f \equiv \phi$) or the repressor one ($f \equiv \bar{\phi}$),

$$\phi(x) = \frac{x^h}{K^h + x^h} \qquad \text{and} \qquad \bar{\phi}(x) = \frac{K^h}{K^h + x^h}. \tag{91}$$

The differential equation associated to the species y yields

$$\frac{dy(t)}{dt} = \alpha + \beta f(x(t)) - \mu y(t), \tag{92}$$

where the species concentration are written as function of time to emphasise their dynamical behaviour. The transcription factor, x, has two steady states: x_L (corresponding to $y = L$) or x_H (corresponding to $y = H$). Note that the input does not change states instantaneously.

Our optimality analysis uses three properties which we briefly review here: (i) metabolic cost, (ii) switching time and (iii) output noise. For more details on these properties please read chapter 4. In the previous chapters, we approximated the metabolic cost by the synthesis rate in the high state,

$$\zeta_y = \alpha + \beta f(x_H). \tag{93}$$

We would like to remind the reader that our definition of cost determines the scaling properties of the metabolic cost and not the exact quantitative measure.

In this chapter, we adopt a different scenario relating to the dynamical behaviour of the system. In this new setting, the change in x is not instantaneous any more. We will assume that x evolves exponentially from x_0 to x^* (between x_L and x_H),

$$x(t) = x^* - e^{-\mu_x t}(x^* - x_0), \tag{94}$$

where we assumed that x is removed with rate μ_x. Then the dynamics of species y yields

$$y(t) = e^{-\mu t}\left[y_0 + \int_0^t e^{\mu z}\left(\alpha + \beta f(x(z))\right) dz\right], \tag{95}$$

where y changes its concentration from y_0 to y^*.

We are interested in the time to reach a fraction θ of the distance between the initial and the new steady state (y_0 and y^*) and, thus, we compute the time to reach $y_\theta = y_0 + (y^* - y_0)\theta$. To compute the switching time, we solve equation (95) numerically, and determine the time at which the system reaches this fraction of the steady state (y_θ).

Finally, since, at steady state, the system remains the same as in the previous sections, we will use the same formula to compute the noise,

$$\eta = \underbrace{\frac{y_H}{(y_H - y_L)^2}}_{\eta_{\text{in}}} + \underbrace{\overbrace{\left[\frac{\beta f'(x_H)}{y_H - y_L}\tau_y\right]^2}^{\text{regulation factor}} \overbrace{\frac{\tau_x}{\tau_x + \tau_y}}^{\eta_{\text{up}}} \sigma_x^2}_{\text{upstream}}, \tag{96}$$

where the noise in the species is the sum between the *intrinsic* component of the noise (η_{in}) and the *upstream* one (η_{up}) (see [151, 123, 19, 129, 152]).

6.3 Noise, Time and Cost

The only difference between the system analysed here and the one in the previous chapter, is that the input of the current system is not changed instantaneously. Due to the fact that we only compute the noise at steady state, the noise properties of both systems are identical. In section 5.3.3 we showed that there is an optimal position of the threshold which optimises the system in terms of noise. We will denote this position by λ_η, where λ is the position of the threshold relative to the input steady states,

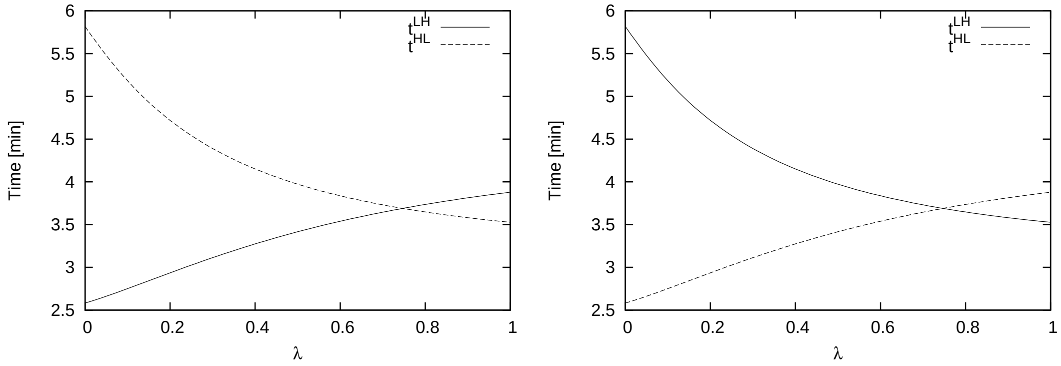
$$\lambda = \frac{K - x_L}{x_H - x_L}. \tag{97}$$

Note that this is valid only in the case when the G_y gene is an activator gene ($x_H > x_L$). Nevertheless, in the case where G_y is a repressor gene, x_L and x_H swap places in equation (97).

6.3.1 Optimal Switching Time

Next, we look at the time properties of the system. Our gene, G_y, has two switching times, T^{LH} and T^{HL}, which represent the time necessary to switch from low state to high state and from high state to low state respectively. If the input switches instantaneously, the switching time is independent of the threshold position. However, in our case, the input does not change instantaneously. If the threshold of the gene is closer to the low state of the input, then switching from low state to high state is faster because the gene will be at least half activated faster. Analogously, if the threshold is closer to the high state, switching from high state to low state is faster because the gene will be again half activated faster. This suggests that when the input does not change instantaneously the threshold position influences the switching time.

T^{LH} is lower when the threshold is closer to the low state, being minimum when the threshold equals the low state, and is higher when the threshold is closer to the high state. Similarly, T^{HL} is lower when the threshold is closer to the high state, being minimum when the threshold equals the high state, and higher when the threshold is closer to the low state. Noting that the two minimum values of the switching times are similar, if not equal, and the fact that the time to switch as a function of the threshold position is a monotonic function we can say that the two functions, $T^{LH}(\lambda)$ and $T^{HL}(\lambda)$, will intersect in only one point. In our case, when computing the speed, it is not important whether a gene is turned on or off and, thus, we consider that the switching time of the gene is the maximum of the time to switch on and the time to switch off, $T = max(T^{LH}, T^{HL})$. This transforms the optimality problem into a minimax problem, i.e., we are interested in the minimum point when the maximum switching time is considered. The position which ensures this minimum switching time of a gene is exactly the intersection point between the two functions, $T^{LH}(\lambda)$ and $T^{HL}(\lambda)$. We denote the point which optimises the time by λ_T. Figure 35 confirms that the solution to the minimax problem is the intersection point between the two functions



(a) activation (b) repression

Figure 35: *The threshold position controls the switching time.* We have used the following parameters: $\theta = 0.9$, $\alpha = 0.2\ \mu M min^{-1}$, $l = 2$, $\mu = 1 min^{-1}$, $\mu_x = 1 min^{-1}$, $x_L = 0.1\ \mu M$ and $x_H = 0.9\ \mu M$. The threshold was varied in the interval $k \in [0.1, 0.9]$. The synthesis rate β is computed so that the cost remains fixed to $\zeta_y = 1.2\ \mu M$; see bellow equation (98).

6.3.2 Optimal Trade-off Curve

There is no indication that the two optimal threshold positions for noise and time (λ_η and λ_T) coincide, $\lambda_T \neq \lambda_\eta$. In the case of $\lambda_T > \lambda_\eta$, the threshold can be positioned in three areas: (i) $\lambda > \lambda_T$, (ii) $\lambda_T \geq \lambda \geq \lambda_\eta$ and (iii) $\lambda_\eta > \lambda$. Note that a similar argument as the one developed below can be provided in the case of $\lambda_T < \lambda_\eta$.

When $\lambda = \lambda_T$ then the system is optimal in speed. We know that the time curve displays one minimum and, thus, moving away from this optimum point will make the system slower, no matter the direction. Analogously, $\lambda = \lambda_\eta$ is the threshold position which minimises noise. Selecting further positions from this optimum increases the noise of the system.

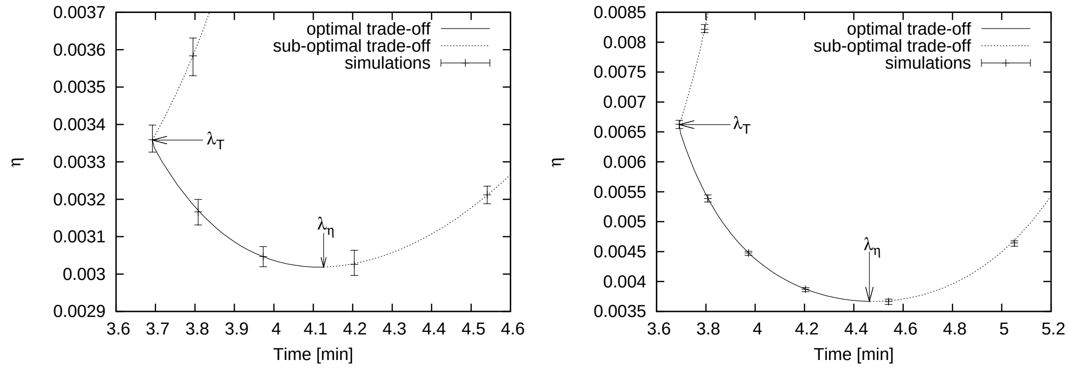
Decreasing the threshold from $\lambda = \lambda_T$ to $\lambda = \lambda_\eta$ improves the system in accuracy, but it reduces the speed. Similarly, increasing the threshold from $\lambda = \lambda_\eta$ to $\lambda = \lambda_T$ improves the speed, but reduces the accuracy. Thus, between these two threshold positions (λ_T and λ_η), there is a trade-off curve, which optimises the system in either speed or accuracy, but not in both.

Furthermore, increasing the threshold above $\lambda = \lambda_T$ or decreasing it under $\lambda = \lambda_\eta$ will move the threshold away from the two optimal positions (in speed and accuracy). This indicates that selecting a position for the threshold in the interval $[\lambda_\eta, \lambda_T]$ is optimal compared with selecting a position outside this interval.

This optimal trade-off curve which depends on the threshold position is graphically represented in Figure 36. When we considered this trade-off, we ensured that the metabolic cost remains constant. Changing the threshold position modifies the regulation function and, thus, the synthesis rate. The synthesis rate in the high state quantifies our measure of cost and to keep this fixed, we compensated any change in K, by a change in the relative synthesis rate β as follows:

$$\beta = \frac{(\zeta_y - \alpha)}{f(xH, K)}. \tag{98}$$

Here, we emphasised that the threshold changes the regulation function by denoting the regulation function as a function of K, $f(xH, K)$. The trade-off curves were computed numerically, but we also run a set of 20 stochastic simulations for 7 points on the curves and the simulations results confirmed that the numerical results approximate with a negligible error the stochastic simulations.



(a) activation

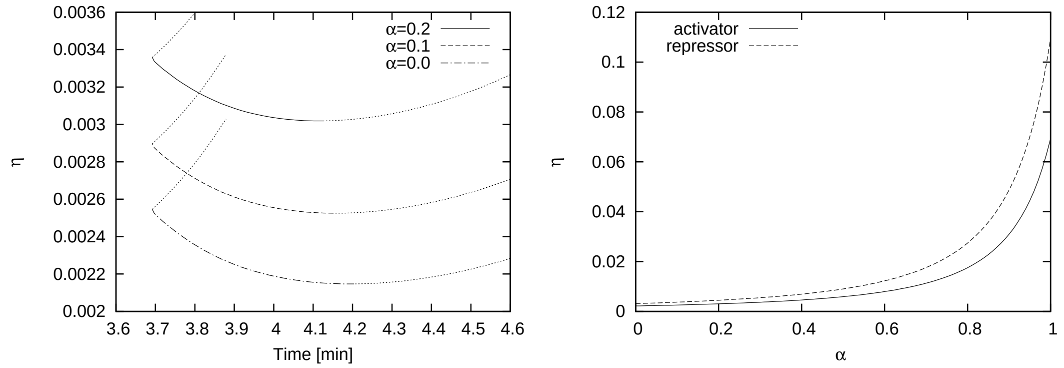
(b) repression

Figure 36: *The threshold position controls the trade-off between speed and accuracy.* We have used the following parameters: $\theta = 0.9$, $\alpha = 0.2\ \mu M min^{-1}$, $l = 2$, $\mu = 1min^{-1}$, $\mu_x = 1min^{-1}$, $x_L = 0.1\ \mu M$, $x_H = 0.9\ \mu M$ and $V = 8 \cdot 10^{-16}\ l$. The threshold was varied in the interval $k \in [0.2, 0.8]$. The synthesis rate β was computed so that the metabolic cost of the gene y remains constant to $\zeta_y = 1.2\mu M \cdot min^{-1}$. The error bars where generated from a set of 20 stochastic simulations using the Gibson-Bruck algorithm [60].

6.3.3 Optimality and the Leak Rate

The optimality of vanishing leak rates in terms of noise was already proven in the previous chapter. For our current configuration, changing the leak rate does not change the switching time (see Figure 37(a)), but it influences the noise as we saw in the previous chapter. In the limit case of $\alpha \to \zeta_y$, the noise increases exponentially to infinity. Nevertheless, in the case of vanishing leak rate, $\alpha = 0$, the system displays the lowest possible noise (see Figure 37(b)).

In addition, the leak rate controls the length of the optimal trade-off curve. Figure 37(a) shows that the optimal threshold position of noise λ_η is reduced by decreasing the basal rate and, thus, the optimal trade-off curve becomes larger. This is a consequence of the fact that changing the leak rate amends the noise properties of the system and consequently the optimal noise configuration.



(a) activation

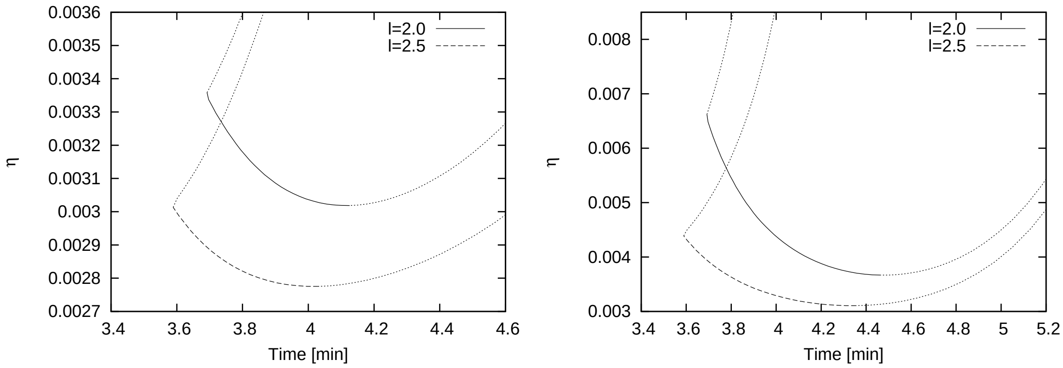
(b) leak rate and noise

Figure 37: *The leak rate changes the noise levels.* (a) The switching time of the trade-off curve does not change while changing the basal rate. (b) The system displays an optimal configuration for vanishing leak-rates, $\alpha = 0$. We have used the following parameters: $\theta = 0.9$, $l = 2$, $k = 0.5\ \mu M$, $\mu = 1 min^{-1}$, $\mu_x = 1 min^{-1}$, $x_L = 0.1\ \mu M$, $x_H = 0.9\ \mu M$ and $V = 8 \cdot 10^{-16}\ l$. The synthesis rate β was computed so that the metabolic cost of the gene y remains constant to $\zeta_y = 1.2 \mu M \cdot min^{-1}$. In (b) we used a threshold value of $K = 0.5\ \mu M$.

6.3.4 Optimality and the Hill Coefficient

In the previous chapter, we showed that by increasing the Hill coefficient, the system will display better noise properties. Here, we observed that increasing the Hill coefficient leads not only to more accurate systems, but also to faster ones. Figure 38 shows that the trade-off curve for a higher Hill coefficient ($l = 2.5$) is better in both speed and accuracy compared with the one with a lower Hill coefficient ($l = 2.0$). The asymptotic limit in speed and accuracy is represented by the step-like regulation function ($l \to \infty$). We will present a proof of this optimality in terms of time in the next chapter, where we will consider the case of step-like regulation functions explicitly.

Note that increasing the Hill coefficient comes at an almost linear increase in metabolic cost. In this chapter, the optimality analysis considers the case when the cost is kept fixed and, thus, we assume that the Hill coefficient needs to be kept fixed.

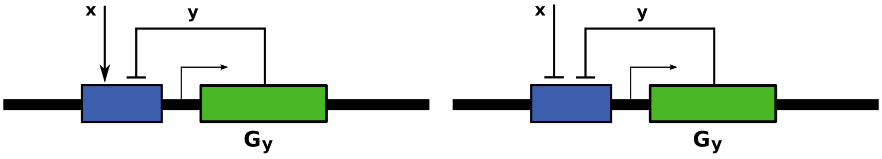


(a) activation (b) repression

Figure 38: *The Hill coefficient can enhance the trade-off between speed and accuracy.* The graph presents two trade-off curves corresponding to two different Hill coefficients: $l = 2$ and $l = 2.5$. We have used the following parameters: $\theta = 0.9$, $\alpha = 0.2\ \mu M min^{-1}$, $\mu = 1 min^{-1}$, $\mu_x = 1 min^{-1}$, $x_L = 0.1\ \mu M$, $x_H = 0.9\ \mu M$ and $V = 8 \cdot 10^{-16}\ l$. The threshold was varied in the interval $k \in [0.2, 0.8]$. The synthesis rate β was computed so that the metabolic cost of the gene y remains constant to $\zeta_y = 1.2 \mu M \cdot min^{-1}$.

6.4 Negative Auto-Regulation

It was previously shown that negative auto-regulation (n.a.r.) leads to higher speeds [137, 5] and sometimes to lower noise [20, 83, 84, 31, 37]. Here, we investigate whether this network motif enhances the trade-off curves.



(a) activation (b) repression

Figure 39: *The model of the negatively auto-regulated gene.* The output protein of a negatively auto-regulated gene represses its own synthesis.

A negatively auto-regulated gene is a gene which synthesises a protein that represses its own synthesis (see Figure 39). We write the differential equation of the n.a.r. gene in the generic form as

$$\frac{dy}{dt} = \alpha + \beta f(x) g(y) - \mu y, \tag{99}$$

where $g(y)$ is a Hill repression function,

$$g(y) = \gamma \frac{K_n^{l_n}}{K_n^{l_n} + y^{l_n}}.$$

Since we want to enhance the performances of the simple gene without increasing the metabolic cost, we need to ensure that the metabolic cost of the n.a.r. system is equal to the one of the simple gene,

$$\begin{aligned} &\alpha + \beta \cdot f(x_H) \cdot g(y_H) = \alpha + \beta \cdot f(x_H) \;\Rightarrow\; g(y_H) = 1 \Rightarrow \\ &\gamma \frac{K_n^{l_n}}{K_n^{l_n} + y_H^{l_n}} = 1 \;\Rightarrow\; \gamma = \frac{K_n^{l_n} + y_H^{l_n}}{K_n^{l_n}} \;\Rightarrow\; g(y) = \frac{K_n^{l_n} + y_H^{l_n}}{K_n^{l_n} + y^{l_n}}. \end{aligned}$$

Bacterial cells can implement auto-repression through various mechanisms. One of the mechanisms consists of the output protein binding to the promoter area of the gene and stopping the RNAp molecules to transcribe the gene. In the case of this mechanism and assuming that only monomers auto-regulate the gene, we compute that the auto-repression function displays a Hill coefficient of 1 [36],

$$g(y) = \frac{K_n + y_H}{K_n + y}. \tag{100}$$

Furthermore, the gene could display additional binding sites where the output protein is able to bind and repress its own synthesis. However, this mechanism requires additional binding sites, which comes at an increase in the metabolic cost of the gene and this is undesirable (we want to keep the metabolic cost fixed). Additional mechanisms used for negative auto-regulation include protein-protein interactions and competitive binding of the output and input proteins. These alternative mechanisms are not considered in this contribution. Instead, we limit our attention to the case of a gene which is auto-repressed by the binding of the output protein to the gene promoter.

Using the form of the auto-regulation function given in equation (100), we can compute the steady states of the n.a.r. system as

$$\begin{aligned} y_H &= \frac{\alpha}{\mu} + \frac{\beta}{\mu} f(x_H), \\ y_L^n &= \frac{1}{2}\left(\frac{\alpha}{\mu} - K_n + \sqrt{\frac{\alpha^2}{\mu^2} + 2\frac{\alpha}{\mu}K_n + K_n^2 + 4\frac{\beta}{\mu} f(x_L)\,(K_n + y_H)}\right). \end{aligned} \quad (101)$$

We denoted by y_L^n the low steady state in the n.a.r. system as opposed to y_L, the low steady state in the simple system. Since both systems display the same high steady state we denoted this by y_H

To keep the mathematics tractable and without losing generality we will consider the case of no basal rate $\alpha = 0$,

$$y_H = \frac{\beta}{\mu} f(x_H) \quad \text{and} \quad y_L^n = \frac{1}{2}\left(-K_n + \sqrt{K_n^2 + 4\frac{\beta}{\mu} f(x_L)\,(K_n + y_H)}\right). \quad (102)$$

From this we can see that the high state remains constant while changing the auto-repression, but the low state is increased if the auto-repression is strengthened ($K_n \searrow \Rightarrow\ y_L^n \nearrow$). The low state of the output varies between the following limits

$$\lim_{K_n \to \infty} y_L^n = \frac{\beta}{\mu} f(x_L) = y_L \quad \text{and} \quad \lim_{K_n \to 0} y_L^n = \frac{\beta}{\mu}\sqrt{f(x_L) f(x_H)} = \sqrt{y_L y_H}. \quad (103)$$

Investigating the auto-regulation function (100), we notice that depending on the relationship between K_n and y_H we could write this function in a simpler form. In particular, there are two extreme cases: (*i*) $K_n \gg y_H$ and (*ii*) $K_n \ll y_H$. In the first case ($K_n \gg y_H$), which we call weak auto-repression, the auto-regulation function becomes

$$g(y) = \frac{K_n + y_H}{K_n + y} \approx \frac{K_n}{K_n} = 1. \quad (104)$$

This suggests that, in the limit of weak auto-repression, the n.a.r. system is similar to the simple gene. This case does not pose an interest to us due to the fact that we are looking for a system able to enhance the properties of the simple gene.

Furthermore, for $K_n \ll y_H$, the auto-repression becomes strong and the auto-regulation function is approximated by

$$g(y) = \frac{K_n + y_H}{K_n + y} \approx \frac{y_H}{y}. \tag{105}$$

Note that our definition of strong auto-regulation is slightly different from Stekel and Jenkins [156] one, which considers strong auto-repression in absolute values, i.e., smaller than $< 10^{-4}\ \mu M$. Our definition is rather concerned with the relative repression strength (K_n) compared with the steady state of the output species (y_H) and aims to determine a parameter space where the auto-regulation function (100) can be written in a simpler form (105).

First, we analyse the dynamic behaviour of this strongly auto-regulated gene by determining the switching time. The next step will be to investigate its stochasticity by computing the noise levels.

6.4.1 Switching Time

We start by considering the case of instant input change. The differential equation of the n.a.r. system becomes

$$\frac{dy(t)}{dt} = \beta f(x) \frac{y_H}{y(t)} - \mu y(t). \tag{106}$$

The solution to this differential equation yields

$$y(t) = \sqrt{\frac{\beta}{\mu} f(x) y_H + \left[y_0^2 - \frac{\beta}{\mu} f(x) y_H \right] e^{-2\mu t}}. \tag{107}$$

where y_0 is the initial steady state and x is the new input which leads to the new steady state y^*. We can extract the time T to reach a fraction θ of the steady state as

$$T_n = \frac{1}{2\mu} \ln \frac{\frac{\beta}{\mu} f(x) y_H - \mu^2 y_0^2}{\frac{\beta}{\mu} f(x) y_H - \mu^2 y_\theta^2}, \tag{108}$$

where

$$y_\theta = y_0 + (y^* - y_0)\theta.$$

We would like to remind that in the case of an instant input change the switching time of a simple gene is (see section 4.2.2)

$$T = \frac{1}{\mu} \ln \frac{y^* - y_0}{y^* - y_\theta}. \tag{109}$$

To make the comparison easier, we will reduce a parameter, namely the decay rate, by performing a variable change on the differential equation (106).

Change of Variable

To apply a change of variable, we change the time by scaling it by a constant C_1,

$$t = C_1\tilde{t} \quad \Rightarrow \quad \frac{dy}{dt} = \frac{d\tilde{t}}{dt}\frac{dy}{d\tilde{t}} = \frac{1}{C_1}\frac{dy}{d\tilde{t}}.$$

The differential equation of the n.a.r. system becomes

$$\begin{aligned} \frac{dy}{dt} = \frac{1}{C_1}\frac{dy}{d\tilde{t}} &= \beta f(x)\frac{y_H}{y} - \mu y, \\ \frac{dy}{d\tilde{t}} &= \beta f(x)\frac{y_H}{y}C_1 - \mu C_1 y. \end{aligned}$$

Assuming that $C_1 = 1/\mu$, leads to the new differential equation of the n.a.r. system:

$$\frac{dy}{d\tilde{t}} = \tilde{\beta} f(x)\frac{y_H}{y} - y. \tag{110}$$

The solution to this differential equation is given by

$$y(\tilde{t}) = \sqrt{\tilde{\beta} f(x) y_H + \left[y_0^2 - \tilde{\beta} f(x) y_H\right] e^{-2\tilde{t}}}. \tag{111}$$

The steady state solution to the differential equation yields

$$y^* = \sqrt{\tilde{\beta} f(x) y_H} \Rightarrow \tilde{\beta} f(x) = (y^*)^2 / y_H. \tag{112}$$

We insert this into equation (111) and we obtain

$$y(\tilde{t}) = \sqrt{(y^*)^2 + [y_0^2 - (y^*)^2]\, e^{-2\tilde{t}}}. \tag{113}$$

The time to reach a fraction θ of the steady state y^* is computed as

$$\tilde{T}_n = \frac{1}{2} \ln \frac{(y^*)^2 - y_0^2}{(y^*)^2 - (y_\theta^n)^2}. \tag{114}$$

Analogously, as in the case of the n.a.r. gene, we apply the same change of variable on the differential equation of the simple gene and obtain

$$\frac{dy}{d\tilde{t}} = \tilde{\beta} f(x) - y. \tag{115}$$

The solution to this differential equation yields

$$y(\tilde{t}) = y^* + e^{-\tilde{t}}(y_0 - y^*).$$

The time to reach a fraction θ of the steady state y^* is then computed as

$$\tilde{T} = \ln \frac{y_0 - y^*}{y_\theta - y^*}. \tag{116}$$

Using these new formulas for the switching time (equations 114 and 116) we investigate which system turns on and off faster.

Switching On

We begin by analysing the turning on case ($y^* = y_H$ and $y_0 = y_L$ or $y_0 = y_L^n$). Noting that the low steady state of the simple gene is always smaller than the high steady state, we can denote it by

$$y_L = m \cdot y_H, \quad m \in [0, 1]. \tag{117}$$

We call $m = y_L/y_H$ the *relative leak rate.* Alike α and y_L, m is a measure of gene leak expression. However, α and y_L are absolute measures of leak expression, while m measures the leak expression relative to the maximum expression.

From equation (112), we compute the low steady state of the n.a.r. gene as

$$y_L^n = \sqrt{\tilde{\beta} f(x_L) y_H} = \sqrt{y_L \cdot y_H} = y_H \sqrt{m}. \tag{118}$$

The fraction θ of the steady state can be written as:

$$\begin{aligned} y_\theta &= y_L + (y_H - y_L)\theta = y_H \left[m + (1-m)\theta\right], \\ y_\theta^n &= y_L^n + (y_H - y_L^n)\theta = y_H \left[\sqrt{m} + (1-\sqrt{m})\theta\right]. \end{aligned} \tag{119}$$

The switching on time of the simple and the n.a.r. genes can be computed using equations (116) and (114) as

$$\begin{aligned} \tilde{T}^{LH} &= \ln \frac{1}{1-\theta}, \\ \tilde{T}_n^{LH} &= \frac{1}{2} \ln \frac{1-m}{1-\left[\sqrt{m} + (1-\sqrt{m})\theta\right]^2}. \end{aligned} \tag{120}$$

We denote the difference in time between $\tilde{T}^{LH}$ and $\tilde{T}_n^{LH}$ by $\tilde{T}_d^{LH}$,

$$\begin{aligned} \tilde{T}_d^{LH} &= \tilde{T}^{LH} - \tilde{T}_n^{LH} \\ &= \frac{1}{2} \ln \left[\frac{1}{(1-\theta)^2} \frac{1-\left[\sqrt{m} + (1-\sqrt{m})\theta\right]^2}{1-m} \right] \\ &= \frac{1}{2} \ln \frac{1-m-2\sqrt{m}\theta + 2m\theta - \theta^2 + 2\sqrt{m}\theta^2 - m\theta^2}{1-m-2\theta+2m\theta+\theta^2-m\theta^2}. \end{aligned}$$

This time difference is positive (and consequently negative auto-regulation speeds up the switching on) when the fraction in the logarithm is higher than or equal to 1,

$$\begin{aligned} &1-m-2\sqrt{m}\theta + 2m\theta - \theta^2 + 2\sqrt{m}\theta^2 - m\theta^2 \geq 1-m-2\theta+2m\theta+\theta^2-m\theta^2 \\ &\Rightarrow \quad \theta(1-\theta)(1-\sqrt{m}) \geq 0. \end{aligned}$$

This is true for any $\theta, m \in [0,1]$. Hence, negative auto-regulation always speeds up the switching on time compared with the simple gene. Figure 40 confirms this result and shows that higher fractions θ of the steady states display better increase in speed that lower ones.

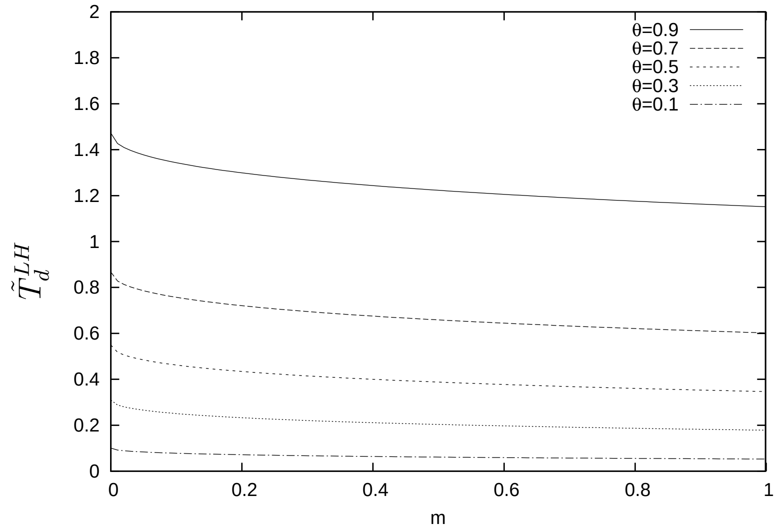


Figure 40: *Negative auto-regulation enhances the switching on speed.* For any combination $(\theta, m) \in [0,1]^2$ the difference between the switching on time of the simple gene and of the n.a.r. gene is positive. This means that turning on is always faster for the n.a.r. gene compared with the simple gene. Note that this time difference is measured in $1/\mu$ and, thus, the actual time enhancement scales by $1/\mu$, i.e., $T_d^{LH} = \tilde{T}_d^{LH}/\mu$.

In the special case of no leak rate (the optimum configuration for noise), $y_L = 0$ and consequently $y_L^n = 0$, the time gain reduces to

$$\tilde{T}_d^{LH} = \frac{1}{2} \ln \frac{1+\theta}{1-\theta}, \tag{121}$$

which is positive as long as $\theta > 0$.

Switching Off

When the gene is turned off ($y_0 = y_H$ and $y^* = y_L$ or $y^* = y_L^n$) the steady states (y_L, y_H and y_L^n) remain the same as the ones for switching on, but the fraction θ

of the steady state becomes

$$\begin{aligned} y_\theta &= y_H + (y_L - y_H)\theta = y_H\left[1 - (1-m)\theta\right], \\ y_\theta^n &= y_H + (y_L^n - y_H)\theta = y_H\left[1 - (1-\sqrt{m})\theta\right]. \end{aligned} \qquad (122)$$

From equations (116) and (114) we can compute the switching off time as

$$\begin{aligned} \tilde{T}^{HL} &= \ln\frac{1}{1-\theta}, \\ \tilde{T}_n^{HL} &= \frac{1}{2}\ln\frac{1-m}{\left[1-(1-\sqrt{m})\theta\right]^2 - m}. \end{aligned} \qquad (123)$$

The difference in time between $\tilde{T}^{HL}$ and $\tilde{T}_n^{HL}$ yields

$$\begin{aligned} \tilde{T}_d^{HL} &= \tilde{T}^{HL} - \tilde{T}_n^{HL} \\ &= \frac{1}{2}\ln\left[\frac{1}{(1-\theta)^2}\frac{\left[1-(1-\sqrt{m})\theta\right]^2 - m}{1-m}\right] \\ &= \frac{1}{2}\ln\frac{1-2\theta+2\sqrt{m}\theta+\theta^2-2\sqrt{m}\theta^2+m\theta^2-m}{1-m-2\theta+2m\theta+\theta^2-m\theta^2}. \end{aligned}$$

Analogously, as in the case of turning on, we can determine whether the time difference is positive by verifying if the fraction in the logarithm is higher than or equal to 1,

$$\begin{aligned} &1-2\theta+2\sqrt{m}\theta+\theta^2-2\sqrt{m}\theta^2+m\theta^2-m \geq 1-m-2\theta+2m\theta+\theta^2-m\theta^2 \\ &\Rightarrow \quad \theta\sqrt{m}(1-\theta)(1-\sqrt{m}) \geq 0. \end{aligned}$$

This means that $\tilde{T}_d^{HL}$ is always positive and the time to turn off a n.a.r. gene is at most equal to the time to turn off a simple gene. Figure 41 confirms these results.

In the case of vanishing leak rates $m = y_L = y_L^n = 0$, the time difference between the two systems becomes zero,

$$\tilde{T}_d^{HL} = \frac{1}{2}\ln\frac{(1-\theta)^2}{(1-\theta)^2} = 0. \qquad (124)$$

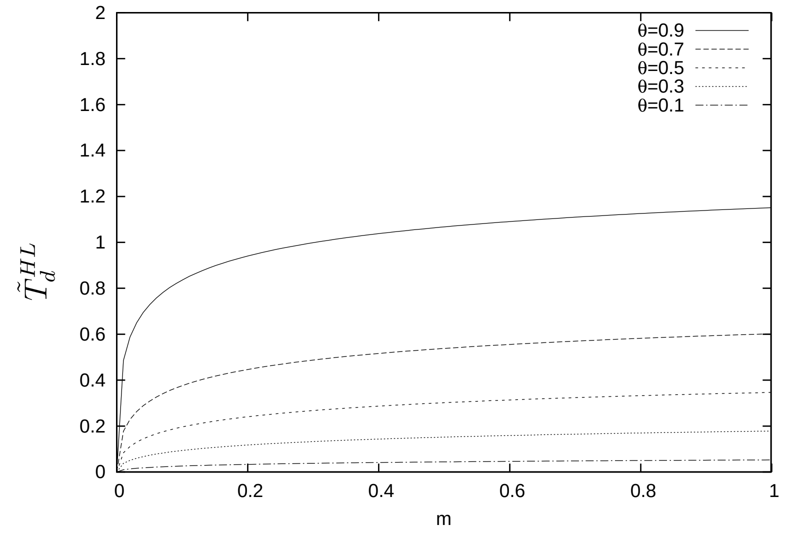


Figure 41: *Negative auto-regulation enhances the switching off speed.* For any combination $(\theta, m) \in [0,1]^2$ the difference between the switching off time for the simple gene and for the n.a.r. gene is positive. This means that turning off in the n.a.r. gene cannot be slower compared with the simple gene.

Thus, for vanishing leak rates, the n.a.r. gene turns on faster compared with the simple gene, but has an equal speed when turning off. Vanishing leak rates are optimal in terms of noise and require both $f(x_L)$ and α to be zero. These two conditions are usually difficult to achieve. For repressor genes, the gene can be turn off completely if either the Hill coefficient or x_L have high values. This usually comes at a high metabolic cost, which we want to keep fixed. Even for activator genes, having a gene completely turned off can be very difficult to achieve, i.e., the regulator molecule would need to be totally absent and the affinity of RNAp for the non activated promoter should be zero. In the case of non-vanishing leak rates (sub-optimal in terms of noise) it must be pointed out that the negative auto-regulation speeds the switching in both directions (on and off).

Non-Instantaneous Input Change

In the case of non-instantaneous input change, the solution of the differential equation (99) can only be computed numerically. For very fast, but non-instantaneous input change, we expect the behaviour to be similar to the one predicted by the instantaneous input change. Figure 42 confirms that the difference between the switching off time of the simple gene and the one of the n.a.r. gene is always positive.

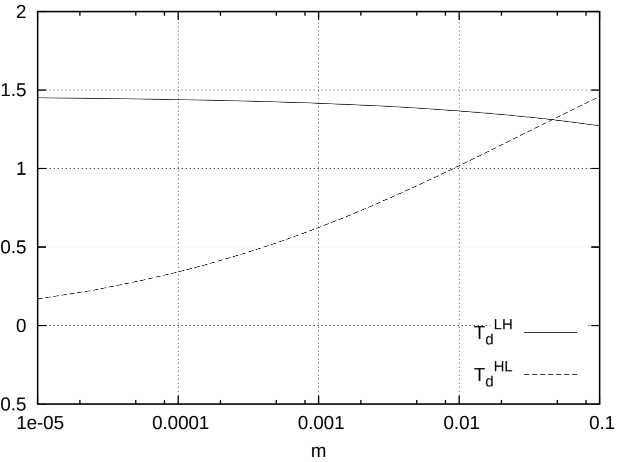


Figure 42: *Fast non-instantaneous input change and speed.* We assumed that the input changes ten times faster than the output. We used the following set of parameters: $\theta = 0.9$, $\alpha = 0\ \mu M min^{-1}$, $l = 2$, $x_H = 0.9\ \mu M$, $\mu = 1\ min^{-1}$ and $\mu_x = 10\ min^{-1}$. x_L was varied in the interval $x_L \in [0.0, 0.2]$ and, thus, m varied accordingly. The regulation threshold is selected so that λ remains fixed to $\lambda = 0.5$ and the synthesis rate so that the cost remains fixed to $\zeta_y = 1.2\ \mu M min^{-1}$. In this graph we considered the activation case.

In our models we usually assume that the input and the output are affected by the same decay rate (dilution) and, thus, they change at a similar speed. The numerical analysis reveals that for most of the parameter space the relationship between switching times (the signs of T_d^{LH} and T_d^{HL}) is conserved (see Figure 43). However, for no or very small relative leak rate ($m \leq 0.0002$ in Figure 43) the switching off time seems to be increased by negative auto-regulation. This suggests that negative auto-regulation is beneficial for speed but only for non-vanishing leak rates of the output gene.

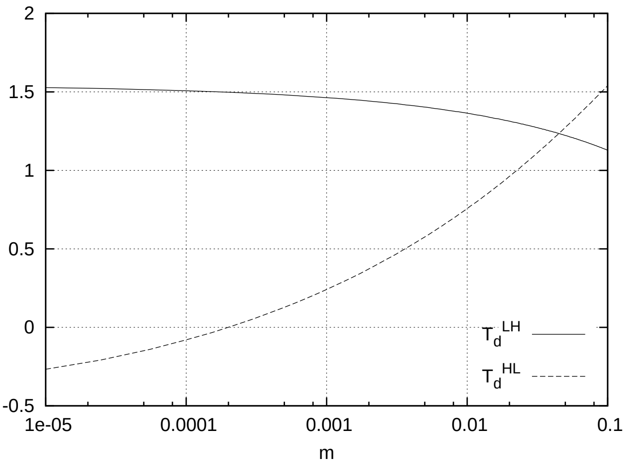


Figure 43: *Slow non-instantaneous input change and speed.* We assumed that the input changes at the same speed as the output. We used the following set of parameters: $\theta = 0.9$, $\alpha = 0\ \mu M min^{-1}$, $l = 2$, $x_H = 0.9\ \mu M$, $\mu = 1\ min^{-1}$ and $\mu_x = 1\ min^{-1}$. x_L was varied in the interval $x_L \in [0.0, 0.2]$ and m was computed accordingly. The regulation threshold is selected so that λ remains fixed to $\lambda = 0.5$ and the synthesis rate so that the cost remains fixed to $\zeta_y = 1.2\ \mu M min^{-1}$. Again, we considered the activation case.

6.4.2 Noise

Both the simple and the n.a.r. genes are systems consisting of two species (x and y), in which the first species (x) affects the synthesis rate of the second one (y), but the second species (y) does not affect the synthesis rate of the first one (x) [123]. Applying the LNA (or FDT) we can compute the steady state variance of species y as [123, 151, 171]:

$$\sigma_y^2 = y\frac{-1}{A_{22}\tau} + \left(\frac{A_{21}}{A_{22}}\right)^2 \frac{1}{1 + A_{11}/A_{22}} x \frac{-1}{A_{11}\tau_x}. \tag{125}$$

where A_{ij} are the elements of the Jacobian matrix associated to the reaction system, τ the decay rate of y and τ_x the decay rate of y.

The Jacobian matrix of the simple gene system is

$$\mathbf{A} = \begin{bmatrix} -1/\tau_x & 0 \\ \beta f'(x) & -1/\tau_y \end{bmatrix}$$

where we denoted by τ_x and τ_y the average life times of species x and y respectively. The variance of the simple gene becomes

$$\sigma_y^2 = y + [\beta f'(x)\tau_y]^2 \frac{1}{1+\tau_y/\tau_x} x. \tag{126}$$

We measured the noise by computing the variance in the high state, $y = y_H$, normalized by the square of the difference between the high and the low state, $(y_H - y_L)^2 = y_H^2(1-m)^2$,

$$\eta_y = \frac{1}{(1-m)^2}\left[\frac{1}{y_H} + \left(\frac{\beta f'(x_H)}{\alpha + \beta f(x_H)}\right)^2 \frac{1}{1+\tau_y/\tau_x} x_H\right]. \tag{127}$$

The first term in the right hand side sum represents the intrinsic component and the second one the upstream component. For no basal rate $\alpha = 0$, the noise is given by

$$\eta_y = \frac{1}{(1-m)^2}\left[\frac{1}{y_H} + \left(\frac{f'(x_H)}{f(x_H)}\right)^2 \frac{1}{1+\tau_y/\tau_x} x_H\right]. \tag{128}$$

In the case of strong negative auto-regulated genes ($K_n \ll y_H$), the Jacobian matrix yields

$$\mathbf{A}^n = \begin{bmatrix} -1/\tau_x & 0 \\ \beta f'(x) y_H/y & -[\beta f(x) y_H/y^2 + 1/\tau_y] \end{bmatrix}$$

From equation (125) we can write the variance of the n.a.r. gene as

$$\sigma_{yn}^2 = \frac{y}{1+\tau_y\beta f(x)y_H/y^2} + \left[\frac{\tau_y\beta f'(x)y_H/y}{\tau_y\beta f(x)y_H/y^2+1}\right]^2 \frac{1}{1+\frac{\tau_y}{\tau_x}\frac{1}{1+\tau_y\beta f(x)y_H/y^2}} x.$$

We consider the variance in the high state ($x = x_H$ and $y = y_H$),

$$\sigma_{yn}^2 = \frac{y}{1+\beta f(x_H)\tau_y/y_H} + \left[\frac{\tau_y\beta f'(x_H)}{\beta f(x_H)\tau_y/y_H+1}\right]^2 \frac{1}{1+\frac{\tau_y}{\tau_x}\frac{1}{1+\beta f(x_H)\tau_y/y_H}} x_H.$$

Furthermore, we assume that the gene has no basal rate ($\alpha = 0$) and, thus, the fraction $\beta f(x_H)\tau/y_H$ becomes 1,

$$\beta f(x_H)\frac{\tau_y}{y_H} = \beta f(x_H)\frac{\tau_y}{\tau_y(\alpha + \beta f(x_H))} = \beta f(x_H)\frac{1}{\beta f(x_H)} = 1.$$

Replacing this in the variance formula yields

$$\sigma^2_{yn} = \frac{y}{2} + \left[\frac{\tau_y \beta f'(x_H)}{2}\right]^2 \frac{1}{1 + \tau_y/2\tau_x} x_H. \tag{129}$$

The noise is computed as the variance normalized by the square of the difference between the high and the low states $(y_H - y^n_L)^2 = y^2_H(1 - \sqrt{m})^2$,

$$\eta_{yn} = \frac{1}{2(1 - \sqrt{m})^2}\left[\frac{1}{y_H} + \left(\frac{f'(x_H)}{f(x_H)}\right)^2 \frac{1}{2 + \tau_y/\tau_x} x_H\right]. \tag{130}$$

Note that in the case of the negative auto-regulation and no basal rate, the low state becomes $y^n_L = y_H\sqrt{m}$ (see equation 118).

To compare the two components of the noise (the intrinsic and upstream) in the two systems (n.a.r. and simple genes) we analyse the ratio between the noise in the n.a.r. gene and the one in the simple gene,

$$\eta^{\text{in}}_c = \frac{\eta^{\text{in}}_{yn}}{\eta^{\text{in}}_y} = \frac{(1 + \sqrt{m})^2}{2} \quad \text{and} \quad \eta^{\text{up}}_c = \frac{\eta^{\text{up}}_{yn}}{\eta^{\text{up}}_y} = \frac{(1 + \sqrt{m})^2}{2}\frac{1 + \tau_y/\tau_x}{2 + \tau_y/\tau_x}. \tag{131}$$

The intrinsic component of the noise in not amplified by negative auto-regulation only for low relative leak rates, $m \leq 0.17$,

$$\eta^{\text{in}}_c = \frac{(1 + \sqrt{m})^2}{2} \leq 1 \;\Rightarrow\; m \leq (\sqrt{2} - 1)^2 \approx 0.17.$$

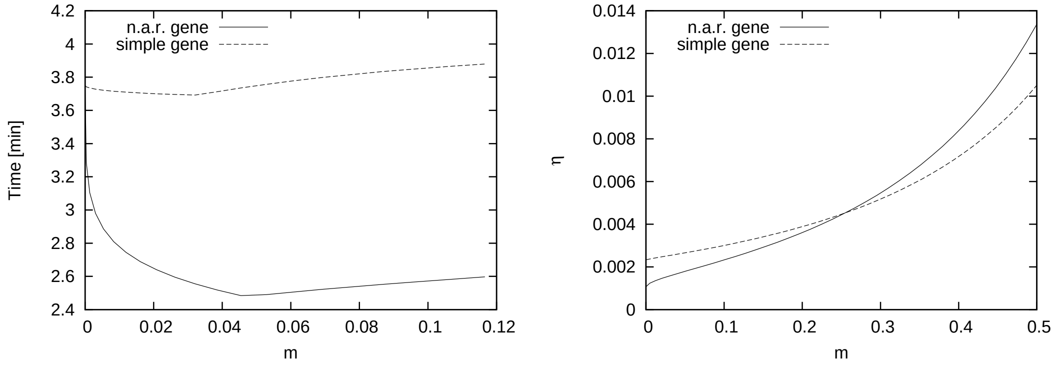
The fraction $(1 + \tau_y/\tau_x)/(2 + \tau_y/\tau_x)$ can never be higher than one. Thus, we can write

$$\eta^{\text{up}}_c \leq \frac{(1 + \sqrt{m})^2}{2} \approx 0.17.$$

Overall, if the relative leak rate is less than a fifth of the high state, then both the intrinsic and the upstream components of the noise are enhanced.

Optimality and Negative Auto-Regulation

We showed above that for any but very low relative leak rates the negative auto-regulation can enhance the switching speed of a binary gene (see Figure 43). This in conjunction with the fact that negative auto-regulation reduces the noise for low relative leak rates ($m \leq 0.17$) suggests that there are some non-zero relative leak rate values for which the system is enhanced at least in one property (speed or accuracy). Figure 44 confirms the existence of relative leak rate values able to enhance the system in both speed and accuracy. Note that in Figure 44 we considered the case when the gene is activated by a regulatory input x. We did this due to the fact that for the repressing case to achieve low relative leak rates we need high Hill coefficient or very high regulatory input and both of these require high metabolic cost.

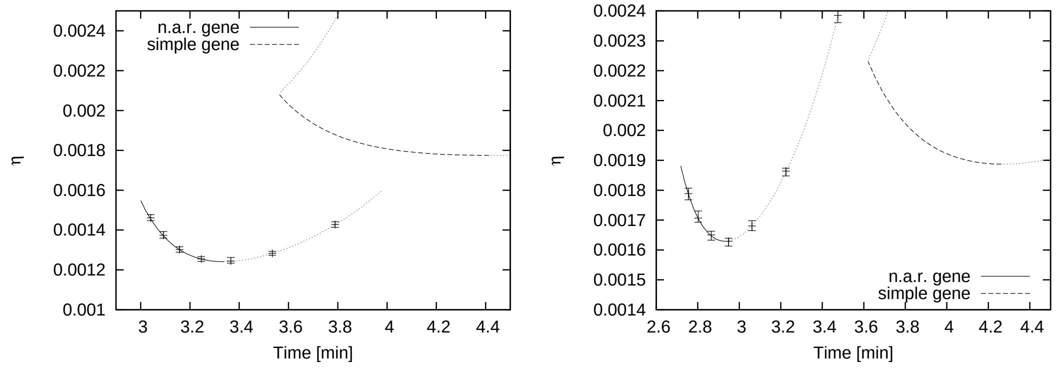


(a) time enhancement

(b) noise

Figure 44: *Leak rate influences the performance of negative auto-regulation.* Increasing the relative leak rate increases the speed but reduces the accuracy. We used the following set of parameters: $\theta = 0.9$, $\alpha = 0.0\ \mu M min^{-1}$ $\mu = 1 min^{-1}$, $\mu_x = 1 min^{-1}$, $x_H = 0.9\ \mu M$, $K_n = 0.01\ \mu M$ and $V = 8 \cdot 10^{-16}\ l$. β was selected so that the cost remains fixed to $\zeta_y = 1.2 \mu M min^{-1}$ and K so that the relative threshold position remains fixed, $\lambda = 0.75$. The gene is activated by a regulatory input x.

On the optimality trade-off curves from Figure 35 we vary the threshold and, consequently, this leads to different values of the relative leak rate. To prove that negative auto-regulation enhances both the speed and noise we draw the trade-off curves for very low but non-zero values of the regulatory input. Note that we considered again the case when the regulatory input x activates the synthesis of y. Figures 45(a) and 45(b) show that the trade-off curves in the case of negative auto-regulation are better than in the case of a simple gene, i.e., the trade-off curve of the n.a.r. gene is shifted to the bottom left side of the one of the simple system. Hence, for certain low but non-vanishing relative leak rates, negative auto-regulation can enhance the trade-off curves compared with simple genes, leading to higher speeds and better accuracies.



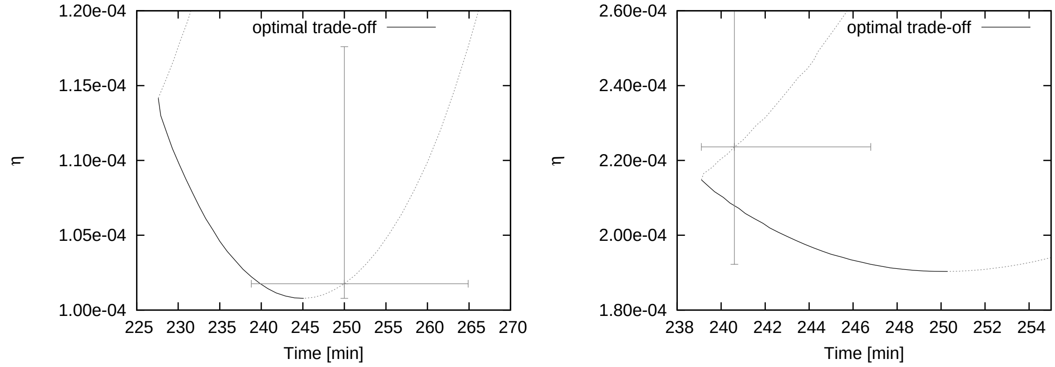
(a) $x_L = 0.02\ \mu M$ (b) $x_L = 0.05\ \mu M$

Figure 45: *Non vanishing low relative leak rates enhance the optimal trade-off curve.* We used the following set of parameters: $\theta = 0.9$, $\alpha = 0.0\ \mu M min^{-1}$ $\mu = 1 min^{-1}$, $\mu_x = 1 min^{-1}$, $x_H = 0.9\ \mu M$, $K_n = 0.01\ \mu M$ and $V = 8 \cdot 10^{-16}\ l$. We selected β so that the cost remains fixed to $\zeta_y = 1.2 \mu M min^{-1}$ and K so that the relative threshold position remains fixed $\lambda = 0.75$. In (a) and (b) we also performed a set of 20 stochastic simulations using the Gibson-Bruck algorithm [60] (the error bars on the n.a.r. curves), which confirmed that the analytical method (LNA) predicts the noise in the case of negative auto-regulation with high accuracy [76].

Note that we performed a series of stochastic simulations to verify the reliability of the analytical method and the results confirmed that LNA performs well even for strong auto-regulation [167, 76, 156]; see Figures 45(a) and 45(b). Stekel and Jenkins showed that for extremely strong auto-repression values ($K_n < 10^{-4}\ \mu M$) the analytical method underestimates the simulation results. However, Zhang *et al.* [187] showed that although the LNA underestimates the value, it still captures with high accuracy the dependence of noise on the parameters of the system.

6.5 Biological Significance

The optimality analysis can be used to investigate whether synthetically built genetic networks are optimal or not. As a case study we consider the P_R promoter in λ-page and a synthetic mutant of this promoter, $P_{OR_2^*}$ [138]. Using the set of parameters published by Rosenfeld *et al.* [138] we applied the optimality analysis on the two systems. Figure 46(a) shows that the wild type P_R promoter resides near the optimal point in terms of noise and, thus, we can approximate the configuration of the P_R promoter by the optimal configuration in terms of noise.



(a) *cI* and P_R WT promoter

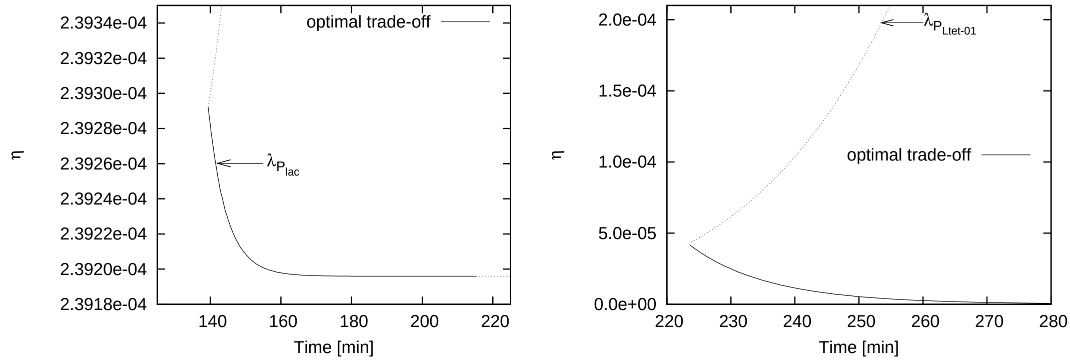
(b) *cI* and $P_{OR_2^*}$ mutant

Figure 46: *Optimality analysis and biological experiments (1).* We used the following set of parameters: $V = 1.5 \times 10^{-15}\ l$, $\mu_x = \mu = \mu_{\text{dilution}} = ln(2)/45\ min^{-1}$, $\theta = 0.9$, $\alpha = 0\ \mu M min^{-1}$, $x_H = 0.006\ \mu M$ and $x_L = 0.140\ \mu M$ [138]. In the case of the wild type P_R promoter (a) we used: $\zeta_y = 0.24\ \mu M min^{-1}$, $K = 0.055\ \mu M$ and $l = 2.4$ [138]. The measurements have significant errors: $K \pm 20\%$, $\zeta_y \pm 7\%$, $V \pm 33\%$ and $l \pm 12\%$. The error bars represent the error in the threshold. For the $P_{OR_2^*}$ mutant (b) we used: $\zeta_y = 0.28\ \mu M min^{-1}$, $K = 0.120\ \mu M$ and $l = 1.7$ [138]. Again, the measurements have significant errors: $K \pm 20\%$, $\zeta_y \pm 15\%$, $V \pm 33\%$ and $l \pm 17\%$.

$P_{OR_2^*}$ is a mutant of the P_R promoter, which was obtained by a point mutation to the OR_2^* operator [138]. The optimality analysis revealed that the system displays a configuration close to the fastest configuration (see Figure 46(b)). Overall, the OR_2^* mutant is faster but less accurate compared with the P_R system. The fact that it is faster although both systems are exposed to the same decay rate is a result of the threshold position. The higher noise in the OR_2^* mutant is caused by both higher extrinsic noise (due to lower Hill coefficient) and lower normalization (due to lower distance between the high and the low steady states of the output). Note that the time necessary for the state of the system to switch is longer than the cell's lifetime. This happens because, when the environment changes, the system first moves to an intermediary steady state (in between the high and low steady states), from where it is easier to either move to the new state or return to the previous one. The system will perform a complete switch (moving from low to high or from high to low states) only in the case of a long time presence/absence of an input, which protects the system from responding to short (noisy) bursts.

The error bars in Figure 46 suggest that current measurements do not provide the analysis with reliable parameters in the sense that the errors from just one parameter (the threshold) make it difficult to state whether the system is on the optimal trade-off curve or not. In addition to the aforementioned measurement errors, the highest abundance of the input was measured to approximately .140 μM, but previous measures indicated values ranging between 0.100 μM and 0.220 μM [138]. This error between independent measurements can be explained by different experimental conditions.

One way to reduce the inaccuracy in parameter measurements is to assume that the input in the system is the substance that is added externally and then fit the output of the system to a Hill function. In the lac system, lactose inactivates the lac repressor, $lacI$, which in turn represses the lac operon P_{lac} ($lactose \dashv lacI \dashv lacZ$). This relation is called derepression and it is approximated by a Hill activation function [88]; $lactose \rightarrow lacZ$. Hence, we will assume that the input protein is $x = lactose$ and the output protein is $y = lacZ$. In this analysis, the accuracy of the measurements is increased by knowing the exact input abundance in the system ($lactose$). Using the set of parameters published by Alon and coworkers [42, 88] we performed our optimality analysis on the lac system and found that the system has a configuration which resides on the optimal trade-off curve near the optimal configuration in speed; see Figure 47(a). This can be explained by the fact that the noise varies only slightly on the optimal trade-off curve, while the change in speed is significant. Thus, the system chose to achieve an important enhancement in speed at the cost of an insignificant loss in accuracy.



(a) *lactose* and P_{lac} promoter (b) aTc and $P_{Ltet-01}$ promoter

Figure 47: *Optimality analysis and biological experiments (2).* In both cases, the derepression is approximated by a Hill activation function. (a) We used the following set of parameters: $V = 8 \times 10^{-16}\ l$, $\mu_x = \mu = \mu_{\text{dilution}} = ln(2)/30\ min^{-1}$, $\theta = 0.9$, $\alpha = 0\ \mu M min^{-1}$, $x_L = 0\ \mu M$ and $x_H = 300\ \mu M$ $K = 130\ \mu M$ and $l = 4$ [42, 88]. (b) We used the following set of parameters: $V = 1.5 \times 10^{-15}\ l$, $\mu_x = \mu = \mu_{\text{dilution}} = ln(2)/45\ min^{-1}$, $\theta = 0.9$, $\alpha = 3.6\ \mu M min^{-1}$, $x_L = 0\ \mu M$ and $x_H = 2.16\ \mu M$ $K = 0.5\ \mu M$ and $l = 2.3$ [82, 83].

A similar experiment consists of the derepression of $P_{Ltet-01}$ by aTC, i.e., aTc inactivates $tetR$, which in active form represses the $P_{Ltet-01}$ promoter ($aTc \dashv tetR \dashv P_{Ltet-01} \Rightarrow aTc \rightarrow P_{Ltet-01}$) [82, 83]. Using the optimality analysis, we showed that the $aTc - P_{Ltet-01}$ system has a sub-optimal configuration in terms of both accuracy and speed.

The optimality analysis can be an useful tool in finding better designs for current experimental synthetic systems. For instance, changing the threshold of the $aTc - P_{Ltet-01}$ system from 1.75 μM to 1.1 μM can ensure an optimal system in terms of speed, which will also display lower noise compared with the configuration used in the experiments. This threshold shift can be achieved by rational (or evolutionary) point mutations in the promoter or regulator region [174, 182]. Nevertheless, the limitations of the measurements and control of the parameters (such as the regulation threshold) pose real problems in the usefulness of this method at least at the current stage of research. Future technological improvements in the measurements and in the modification of binding affinities will make this technique a useful tool for synthetic biologists.

6.6 Summary

In this chapter, we extended our investigation on the binary genes by assuming non-instantaneous change in the regulatory input and fixed metabolic cost. The analysis showed that, under this assumption, the regulatory threshold controls how fast a gene is switched and how accurate the output of the gene is. We found that binary genes are characterised by two threshold positions, one which is the optimum in terms of speed and another which is the optimum in terms of accuracy. All the threshold values between these two optimal configurations define systems which reside on an optimal trade-off curve (see Figure 36). The points on the optimal trade-off curve determine systems which enhance one of the properties (speed or accuracy). Threshold values outside this interval are sub-optimal in both speed and accuracy, in the sense that they worsen both speed and accuracy.

The optimal trade-off curve can be enhanced by reducing the leak rate (see Figure 37) or by increasing the Hill coefficient (see Figure 38). Vanishing leak rates can be ensured only for the activator gene in the total absence of regulatory input. The second solution to enhance the trade-off curve (increasing the Hill coefficient) comes at an increase in the metabolic cost, which is not desirable.

Furthermore, we extended our analysis and considered the case of negatively auto-regulated binary genes. Our results indicated that increasing the relative leak rate leads to the n.a.r. system being faster in both turning on and turning off compared with the simple gene; see Figure 44(a). Reducing the relative leak rate, on the other hand, leads to less noise; see Figure 44(b). This suggests that there is an interval of small but non-vanishing relative leak rates for which negative auto-regulation enhances both speed and accuracy (see Figures 44 and 45). In addition, we observed a trade-off between speed and accuracy controlled by the leak rate, in the sense that lower leak rates lead to more accurate but slower systems, while higher leak rates lead to noisier but faster systems. We should mention that there is a clear distinction between the activation and the repression case, meaning that it is easier to achieve low leak rates in the activator case, while in the repressor case this comes at a high cost (the Hill coefficient or the abundance of the input needs to be high). Thus, it is easier to enhance an activator gene by negative auto-regulation compared with a repressor gene.

Finally, using parameters provided in the literature we were able to draw trade-off curves for some synthetic experimental systems and to pinpoint where these systems are on the trade-off curves (see Figure 46). However, the usefulness of this approach is limited by the accuracy of the measurements. Current measurement techniques produce results highly dependent on the experimental setting and with significant errors; for example three independent measurements of the total concentration of *cI* in a cell generated three significantly different values 0.10 μM, 0.14 μM and 0.22 μM. One solution to this problem is to assume that the input in the system is the externally added protein and fit the functional relationship between this input and the first affected gene to a sigmoid function (Hill function) (see Figure 47). This will eliminate the error measurements in the input concentration due to the fact that exact measurements of the input concentrations are usually available. However, error measurements in other parameters (such as threshold or Hill coefficient) make this approach impractical. For example an error in the measured value of the threshold of approximately 20% makes it impossible to say whether the gene is on the optimal trade-off curve or not (see Figure 46). We expect that future improvements of measurement techniques will lead to our optimality analysis being a useful tool in engineering synthetic genetic systems.

Chapter 7

Design of a Genetic Full-Adder

In this chapter, we modelled a binary full-adder using binary genes as logic gates and then we determined the set of parameters which ensure modularity and scalability of the design. In addition, using the optimality analysis presented in the previous chapter, we identified the subset of parameters that ensure optimal behaviour.

7.1 Introduction

Researchers have modelled and engineered genes in bacterial cells which perform basic computational tasks. These tasks mainly mimic the behaviour of simple electronic components, such as logic gates, oscillators, toggle switches and counters [57, 47, 71]. However, when attempting to increase the complexity of these engineered genetic systems, certain limitations of the components are likely to hamper their construction. Thus, there is an urgent need for an extensive analysis of the biophysical limits of the elementary components.

Synthetic biologists showed that binary logic gates can be engineered in living cells using transcriptional logic [71, 93, 182, 39, 8, 145]. Boolean logic can be implemented using *transcriptional logic*, that is genes which can integrate multiple signals at the level of *cis*-regulatory transcription control using various binary logic functions (AND, OR, NAND, NOR, XOR, etc.). In this design, the transcription factors represent the inputs of the gate and the protein which is expressed, the output. The signals (inputs and output) are quantified by the concentrations of the corresponding proteins.

Biological modellers have successfully identified and described various designs of these transcriptional logic gates [175, 32, 78, 119, 146, 153, 188, 58]. However, what is still missing is a complete analysis of how these logic gates can be used as building blocks for more complex logical systems and what parameters ensure an optimal design in terms of speed and accuracy under limited (fixed) energy resources. We will address these questions in the context of a genetic full-adder, which is an essential logical component that represents a classic challenge for new computational paradigms [96]. The *full-adder* is a system able to perform binary addition (to produce both the sum and the carry) on three binary inputs and it can be cascaded to perform serial addition for long binary numbers, i.e., two of the inputs represent the two operands while the third one is connected to the carry from an upstream full-adder [186].

Our approach to design logic systems consists of three steps: (*i*) construct all the required logic gates, (*ii*) find the set of parameters which allow modularity and scalability of the system and (*iii*) find the subset of parameters which optimise the system. First, we constructed the required logic gates by considering genes that can be regulated by two proteins in an independent fashion, i.e., binding of any of the inputs does not alter the binding of the other input (see section 7.2).

Furthermore, we want to have a modular and scalable design, in the sense that the parameters of any two gates should match and that the system could contain an arbitrary number of gates and ensure a distinguishable binary output of the system. In the case when the threshold of any downstream gate is not bounded by the low and high steady state of the output of the upstream gate, then changes in the upstream gate are not reflected in the output of the downstream gate and the gates experience a parameter mismatch [10]. This lack of modularity in parameters is undesirable, because it removes the Boolean behaviour of the gates. Consequently, we require our upstream logic gates to display two output steady states, one that is lower and another one that is higher compared with the threshold of the downstream gate.

In addition, adding more gates in the system can lead to a reduction in the signal strength (the difference between the high and the low abundances of the output) of the downstream gates [103, 147, 189]. This reduction of signal strength makes it difficult to distinguish between the two binary values of the output and, thus, it is unwanted. We aim to model a system which is *scalable*, in the sense that it is able to connect an arbitrary number of gates without affecting the signal strength of the output. In this contribution, we ensure scalability by assuming that the signal strength of the input (the difference between the high and the low concentrations) does not decrease at the output of the gate (see section 7.3.1).

Finally, we consider that it is essential to select an optimal subset of parameters. Thus, we performed our optimality analysis and identified the optimal trade-off curve between speed and accuracy for a fixed metabolic cost (see section 7.3.2).

Our results show that, in the ideal case of step regulation function ($l \to \infty$), the system displays an optimal position of the threshold in terms of speed and accuracy (see subsection 7.3.2) while, in the biologically plausible case of sigmoid regulation function (finite and low Hill coefficients), there is a trade-off between these two properties controlled by the position of the threshold (see subsection 7.3.2). In the latter case, our analysis showed that the system displays different optimal configurations for speed and accuracy under fixed metabolic cost. Furthermore, we determined that there is an optimal trade-off curve bounded by these two optimal configurations. Any configuration outside this optimal trade-off curve is sub-optimal in both speed and accuracy.

This chapter is structured as follows. In section 7.2 we present and model the full-adder. In section 7.3 we determine the parameters which ensure interconnectivity and we also perform the optimality analysis on two types of systems: genes with step-like regulation function and genes with sigmoid regulation function. Then, in sections 7.4 and 7.5 we discuss both the validity of our approach and the biological significance of the results. Finally, in section 7.6 we draw the conclusions of this chapter.

7.2 Building a Genetic Full-Adder

The signals of our transcriptional logic gates are encoded into proteins and they flow together into a common compartment, the cell. In the case of a high number of signals, the system can be affected by crosstalk, in the sense that proteins which encode different signals can react (which is undesirable). This suggests that it is better for molecular logic systems built within a common compartment to have as few genes as possible. In addition, this also indicates that it would be more feasible for our logic gates to be constructed from single genes so that we have fewer proteins in the cell.

Furthermore, Cox III *et al.* [39] observed that transcriptional logic gates which integrate more inputs display a worse binary behaviour compared with the ones which integrate fewer input signals. This in conjunction with the fact that, in order to construct complex systems, we want gates to integrate more than one input indicates that it is better for our logic gates to have only two inputs.

Several designs which implement a binary full-adder were proposed in the literature [81, 121, 96]. Based on the two requirements presented above (small number of genes and each gene should have small number of inputs) we considered a design which consists of 5 logic gates, where the gates have only two inputs; see Figure 48. The five logic gates required by the design we selected are the following: two XOR gates, two AND gates, and one OR gate.

To construct this full-adder from genes, we first build the three logic gates (AND, OR and XOR) using transcriptional logic. We model transcriptional logic gates using a single binary gene G_z, which synthesises protein z, where the abundance level of z is assumed to be the output of the gate. This gene is regulated by two proteins x and y, which are considered to be the inputs of gate. Species z

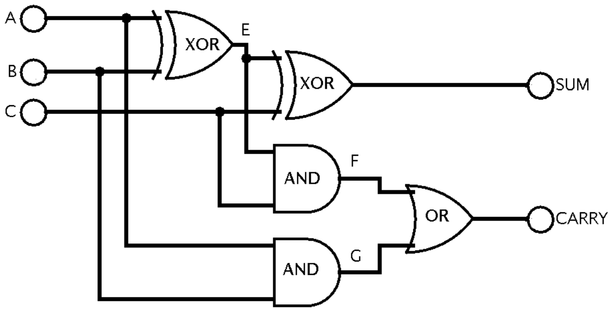


Figure 48: *The logic diagram of a full-adder.*

is described by the following deterministic differential equation

$$\frac{dz}{dt} = \alpha + \beta f(x,y) - \mu z, \tag{132}$$

where α is the basal synthesis rate, $\alpha + \beta$ the maximum synthesis rate, $f(x,y)$ is the regulation function and μ is the decay rate.

In this contribution we do not consider protein-protein interactions. Thus, we assume that the two inputs in the gate do not interact; this is usually called the independent binding model [146]. The gene which implements the logic function has two operator sites O_x and O_y, each of them having l binding sites. On each operator site only molecules of a specific transcription factor can bind, and they do this in a homo-cooperative manner. The probabilities that an operator site is full are described by a Hill function [2, 27, 36]

$$p_x(x) = \frac{x^l}{x^l + K^l}, \quad p_y(y) = \frac{y^l}{y^l + K^l}, \tag{133}$$

where K is the regulation threshold (the required input value for half activation of the gene) and l is the Hill coefficient (indicates the steepness of the function). To keep the mathematics tractable and without losing generality, we assumed that the two operator sites (O_x and O_y) have identical parameters (K and l).

For each logic gate we considered a different *cis*-regulatory scenario [32, 78, 146]. First, we assumed that the gene is turned on when any of the two TFs are present. This type of behaviour mimics an OR gate. Analogously, in the case when a gene is turned on only when both transcription factors are present, then the regulation function will mimic the behaviour of an AND gate. Despite all the advantages that our design has compared with others, it mainly suffers from the fact that it relies on a complex logic gate, the XOR gate. Here we consider the case when the gene is turned on when any of the TFs is present, but when both of them are present their effects cancel out and the gene is turned off. Note that there are cases of AND and OR gates implemented from single genes which are encountered in both living organisms [148] or engineered synthetically [182, 105, 8, 39, 136, 145]. To our knowledge, there are no examples of genes which display XOR behaviour. Nevertheless, for our analysis, it is important only the shape of the regulation function and not the underlying biological mechanism which achieves the corresponding logical function.

The three scenarios presented above can be written mathematically as:

$$\begin{aligned} f_{AND}(x,y) &= p_x(x) \times p_y(y), \\ f_{OR}(x,y) &= p_x(x) + p_y(y), \\ f_{XOR}(x,y) &= p_x(x) + p_y(y) - p_x(x) \times p_y(y). \end{aligned} \tag{134}$$

Expanding the individual binding probabilities using equations (133) yields

$$\begin{aligned} f_{AND} &= \frac{(xy)^l}{(xy)^l + (Kx)^l + (Ky)^l + K^{2l}}, \\ f_{OR} &= \frac{(xy)^l + (xK)^l + (yK)^l}{(xy)^l + (Kx)^l + (Ky)^l + K^{2l}}, \\ f_{XOR} &= \frac{(Kx)^l + (Ky)^l}{(xy)^l + (Kx)^l + (Ky)^l + K^{2l}}. \end{aligned} \tag{135}$$

Figure 49 confirms that these regulation functions display the desired behaviour.

Using these three logic gates, the full-adder can be constructed as a set of chemical reactions. Since the full-adder contains five logic gates, then we need five proteins (signals) to implement this system (e, f, g, sum and carry). Using

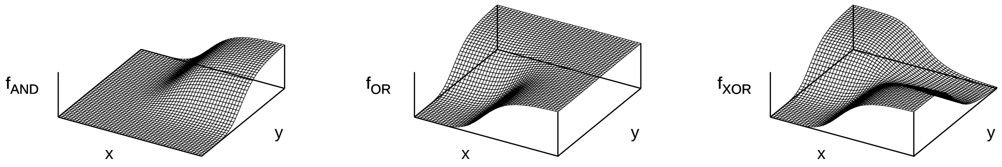


Figure 49: *Regulation functions which mimic logic behaviour.* The threshold was set to $K = 0.5$ $[\mu M]$ and we considered a Hill coefficient of $l = 6$.

the model of a gene given in equation (132), we construct the chemical reactions which describe the full-adder as

$$\begin{aligned}
&\emptyset \xrightleftharpoons[\mu_e]{\alpha_e+\beta_e f_{XOR}(a,b)} e, \quad \emptyset \xrightleftharpoons[\mu_f]{\alpha_f+\beta_f f_{AND}(c,e)} f, \quad \emptyset \xrightleftharpoons[\mu_g]{\alpha_g+\beta_g f_{AND}(a,b)} g, \\
&\emptyset \xrightleftharpoons[\mu_s]{\alpha_s+\beta_s f_{XOR}(e,c)} \text{sum}, \quad \emptyset \xrightleftharpoons[\mu_{co}]{\alpha_{co}+\beta_{co} f_{OR}(f,g)} \text{carry},
\end{aligned} \tag{136}$$

where a, b and c are three input species, f_{AND}, f_{OR} and f_{XOR} are the regulation functions of the genes as described by equations (135), $\mu_{(\cdot)}$ the decay rates, $\alpha_{(\cdot)}$ the basal synthesis rates and $\alpha_{(\cdot)} + \beta_{(\cdot)}$ the maximum synthesis rates.

7.3 Analysis of the Genetic Full-Adder

In the previous section, we presented the logical design of a full-adder and constructed all the required logic gates. Next, for the model of the full-adder, we will identify the required constraints on the sets of parameters to allow the interconnection of the gates and then we will determine the subset of parameters which allows the optimal functioning of the full-adder in terms of speed and accuracy under fixed metabolic cost. We will apply these two analyses for two cases: (*i*) step gates, where the genes have step-like regulation functions ($l \to \infty$), and (*ii*) sigmoid gates, where the genes have regulation functions with finite Hill coefficients.

To keep the mathematics tractable, and without losing generality, we will consider only the case of identical gates, i.e., all genes are affected by the same decay rate (μ), have the same synthesis rates (α and β) and the same Hill parameters (l and K). The logic gates differ from each other in respect with the inputs that regulate the genes, the expressed proteins and the *cis*-regulatory function (f_{AND}, f_{OR} or f_{XOR}).

7.3.1 Interconnecting Logic Gates

It was previously observed that adding more molecular logic gates into the system resulted in worse binary behaviour of the downstream gates [103, 147, 189]. In this contribution, we want our logic system to be able to contain an arbitrary number of logic gates while the output of the system remains binary (it has a low and a high state). Furthermore, considering that all genes have the same parameters, adding or removing genes should not require a change in parameters so that the output stays binary. In the case of enzymatic or DNA logic gates, researchers had to design an additional amplification gate to solve this problem [147, 189]. In the case of transcriptional logic gates, we propose that there is a set of values of the synthesis rates (α and β) which ensure a scalable design.

We assume that the genes in our system have sigmoid regulation functions and respond to a binary input (low and high abundances at input) with a binary output (low and high abundances at output). We denote by *signal strength* the difference between the high and the low abundance of the output/input of a gene. In the deterministic case, the signals strengths can be as low as possible as long as they are not zero, without affecting the computational usefulness of the gene. However, in the stochastic case, the quantifiable nature of molecules imposes a lower limit on the signal strength of 1 molecule. Moreover, for low signal strengths, even higher than 1, noise can make it difficult to distinguish between the two output states (low and high). To address this, we impose that an arbitrary selected value of the signal strength is conserved both at the output and input of all the genes in the system, i.e., the signal strength of the inputs and outputs of all the genes cannot be lower that an arbitrary selected value.

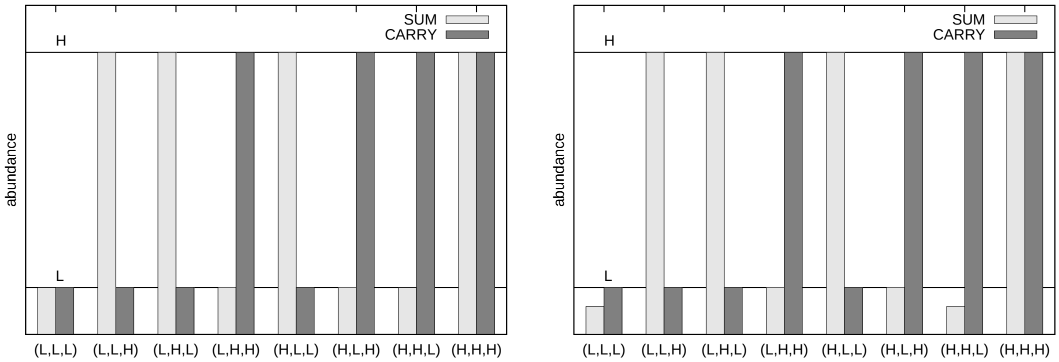
We start this analysis by considering the ideal case, the gates have an all or nothing type of response, i.e., the regulation functions of the genes have very high or infinite Hill coefficient ($l \to \infty$). The interconnectivity property can be met by considering that the output steady states have the same values as the input ones, $H_{out} = H_{in} = H$ and $L_{out} = L_{in} = L$. For example, in the case of the OR gate, if we take into account the logic function performed by the gene then we have the following steady state equations:

$$\begin{aligned} L &= \frac{1}{\mu}\left[\alpha + \beta f_{OR}(L, L)\right], \\ H &= \frac{1}{\mu}\left[\alpha + \beta f_{OR}(L, H)\right], \\ H &= \frac{1}{\mu}\left[\alpha + \beta f_{OR}(H, H)\right]. \end{aligned} \tag{137}$$

For infinite Hill coefficients the solution this system is given by $\alpha = L$ and $\beta = (H - L)$. Analogously, it can be shown that the solution is the same for all gates. These values for the synthesis rates ensure correct steady-state behaviour of the full-adder; see Figure 50(a).

Real genes are unlikely to display step-like behaviour. This happens because Hill coefficients are bounded from above by the number of regulatory binding sites [36], and genes have a small number of binding sites [78]. Thus, next, we will consider the case of sigmoid regulation functions with finite low Hill coefficients.

For low Hill coefficients, the system of equations (137) has only one solution, $H = L$. This is not a useful solution because it removes the binary logic. Therefore, we will relax the condition that the signal strength is preserved and search for parameters which ensure that the signal strength of the output is not lower than the signal strength of the input, $(H_{out} - L_{out}) \geq (H_{in} - L_{in})$. This can be achieved by solving only the first two equations from system (137) and results in



(a) step regulation function

(b) sigmoid regulation function

Figure 50: *The steady state output of the full-adder for any combination of the input.* (a) The output abundance based on the input abundance for step-like regulation functions, $l = 50$. (b) The output steady state of the full-adder in the case of finite low Hill coefficients, $l = 6$. The following set of parameters was used: $\mu = 1\ min^{-1}$, $L = 0.2\ \mu M$, $H = 1.2\ \mu M$ and $K = 0.7\ \mu M$.

the following synthesis rates

$$\begin{aligned} \alpha_{OR} &= \mu \frac{L f_{OR}(L,H) - H f_{OR}(L,L)}{[f_{OR}(L,H) - f_{OR}(L,L)]}, \\ \beta_{OR} &= \mu \frac{H - L}{[f_{OR}(L,H) - f_{OR}(L,L)]}. \end{aligned} \tag{138}$$

Note that the synthesis rates will not have positive values for all sets of parameters (l, K, μ, H, L). Interestingly, increasing the Hill coefficient increases the space of allowed parameters and, in the limit case of a step function ($l \to \infty$), any values of the other parameters will generate positive synthesis rates. For Hill coefficient less than or equal to 1 there is no positive solution for this system. Hence, logic system constructed from genes with Hill coefficients lower than or equal to 1 cannot display scalability. Analogously, we used the same mechanism to determine the synthesis rates for the two other gates, AND and XOR,

$$\begin{aligned}
\alpha_{AND} &= \mu\frac{Lf_{AND}(H,H) - Hf_{AND}(L,H)}{[f_{AND}(H,H) - f_{AND}(L,H)]}, \\
\beta_{AND} &= \mu\frac{H-L}{[f_{AND}(H,H) - f_{AND}(L,H)]}, \\
\alpha_{XOR} &= \mu\frac{Lf_{XOR}(L,H) - Hf_{XOR}(H,H)}{[f_{XOR}(L,H) - f_{XOR}(H,H)]}, \\
\beta_{XOR} &= \mu\frac{H-L}{[f_{XOR}(L,H) - f_{XOR}(H,H)]}.
\end{aligned} \tag{139}$$

Figure 50(b) confirms that the signal is not decreased and shows that in two cases the actual output low state (L_{out}) is lower than the desired one (L).

7.3.2 Optimality Analysis

So far, we have shown that, for our genetic logic gates to display interconnectivity, the synthesis rates and the high and low states of genes need to be set to certain values. In this section, we will extend the analysis of the full-adder and determine the optimal configuration of the system in terms of speed, accuracy and metabolic cost (see chapter 6). The metabolic cost of a gene Z is defined here as the maximum synthesis rate of that gene:

$$\zeta = \alpha + \beta. \tag{140}$$

Note that compared with our previous notion of metabolic cost, in this chapter, we disregarded the contribution of the regulation function to this cost, i.e., previously we had $\zeta = \alpha + \beta f_z^H$. This change is justified by the fact that the two synthesis rates (α and β) need to be kept fixed to values that ensure a scalable design. By keeping the synthesis rate fixed (to ensure interconnectivity) the metabolic cost is maintained constant. Note that this measure of metabolic cost is just an approximation to the actual value, and that the metabolic cost of the maintenance of the entire machinery was not included in it (see chapter 4).

Under this setting, the optimality analysis investigates the speed and the accuracy of the full-adder for fixed metabolic cost. As we did in previous section, we consider two cases: (i) step-like regulation functions ($l \to \infty$) and sigmoid regulation functions (finite low Hill coefficients).

Step Gates

First, we consider the case when genes display step-like regulation functions ($l \to \infty$). Here, we will define and compute the time to reach steady state after an instantaneous change of input, as a measure of speed, and the noise level at the output of the full-adder, as a measure of accuracy.

The *switching time*, T_{gene}, of a gene is the time required to reach the steady state within a fraction θ of $H - L$. Assuming instantaneous change of input, the differential equation (132) can be solved analytically and the time to reach a fraction θ of the steady state, $L + (H - L)\theta$ or $H - (L - H)\theta$, can be computed as

$$T_i = \tau \cdot ln \left(\frac{1}{1 - \theta} \right), \tag{141}$$

where $\tau = 1/\mu$ represents the average life time of the species.

The propagation time through a single gate can only be reduced by decreasing the average life time of the protein (τ). In the case when the two logical steady states and the synthesis rates are kept constant (so that the signal strength is not reduced and the metabolic cost not increased), then the decay rate remains constant. Thus, there is no optimization that one could attempt to perform on individual gates under fixed metabolic cost without reducing signal strength. However in the case of systems of logic gates, like the case of the full-adder, the input is not changed instantaneously in all gates and the position of the threshold influences the propagation time.

We assume that the threshold is located between the low and the high state, $K = L + (H - L)\lambda$, ($\lambda \in [0, 1]$). λ indicates the position of the threshold; for $\lambda < 0.5$, K is closer to L and, for $\lambda > 0.5$, K is closer to H. Note that, by considering K to be outside the interval $[L, H]$, the regulation is removed, i.e., the gene is always in the same state no matter whether the input is L or H. In order for a gene to change state and start/stop expressing a protein, one of the inputs, has to cross over or under K. Using equation (141), we computed the time it takes one species to move from low state to the threshold ($L \to K$) and from the high state to the threshold ($H \to K$) as

$$t_{LK} = \tau \cdot ln\left(\frac{1}{1-\lambda}\right), \quad t_{HK} = \tau \cdot ln\left(\frac{1}{\lambda}\right). \tag{142}$$

Assuming that the longest cascade in the system has n gates, then a general formula for the propagation time is given by

$$T = \sum_{i=1}^{n-1} t_{iK} + T_n, \tag{143}$$

where $t_{iK} = t_{LK}$ if gate ith is turned on and $t_{iK} = t_{HK}$ if it is turned off. Hence, the propagation time in a cascade equals a sum of t_{LK} and t_{HK} terms and a fixed time representing the last gene in the cascade T_n.

Figure 51 confirms that, based on the threshold position, a gene can be faster when switching in one direction and slower in the opposite direction. When the switching direction is not important, the problem of optimising propagation time becomes a minimax problem, i.e., minimise the maximum of the time to switch in one direction and the time to switch in the other direction. In the context of step-like regulation functions, the optimum threshold resides at the midpoint between high and low states, $\lambda_T = 0.5$ (see Figure 51 and equation 142).

Analysing the circuit diagram of the full-adder (see Figure 48) one can notice that the longest path through the circuit consists of three gates, and this is used when computing the carry. Figure 52 confirms that the propagation time is longest when computing the carry and the longest path is followed, for example, when switching between (L, L, H) and (H, L, H).

From equation (143), we know that the propagation time is a sum of t_{LK} and

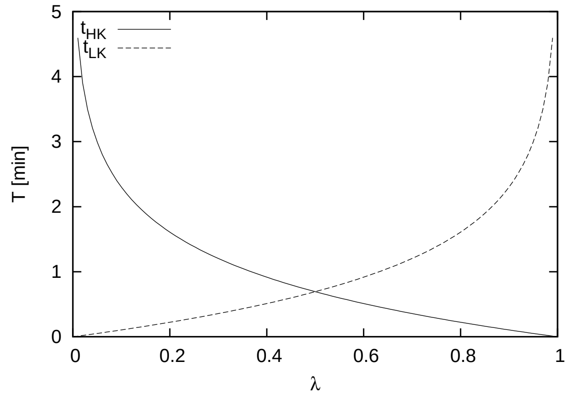
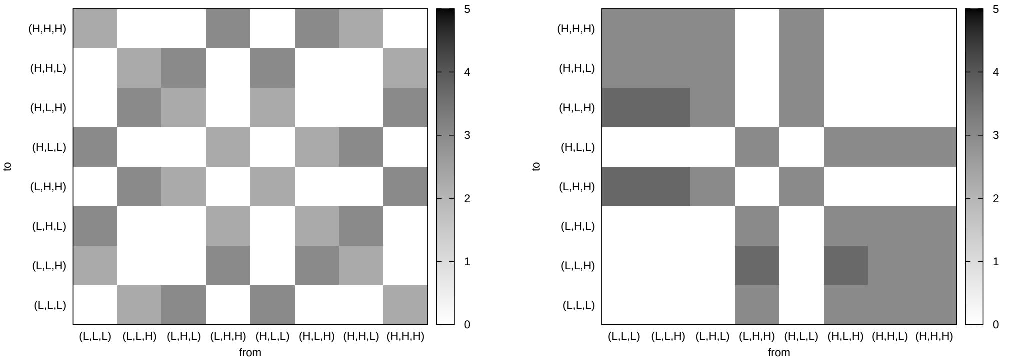


Figure 51: *The time to reach the threshold for the full-adder.* The protein average life time was set to $\tau = 1\ min$. The two steady states are $L = 0.2\ \mu M$ and $H = 1.2\ \mu M$.



(a) sum

(b) carry

Figure 52: *Detailed map of the propagation time of the full-adder.* The propagation time to reach a fraction $\theta = 0.9$ of the steady state of the (a) sum and (b) carry. The x-axis represents the initial input steady state and the y-axis the new state which is instantaneously changed. The shades of grey represent the time to reach a fraction θ of the steady state. The following set of parameters was used: $\mu = 1\ min^{-1}$, $L = 0.2\ \mu M$, $H = 1.2\ \mu M$, $K = 0.7\ \mu M$ and $l = 50$. The time in the graphs is computed by solving numerically the differential equations attached to the chemical reaction set of the full-adder (136). This time coincides with the one predicted by equation (143).

t_{HK} terms and, from equation (142), we know that the threshold at the midpoint between the high and the low states minimises t_{LK} and t_{HK}. Hence, the threshold at the midpoint is the optimum position in terms of propagation time. Figure 53 confirms that the optimum threshold for the full-adder, in the case of step-like regulation function, resides at the midpoint between high and low state ($\lambda = 0.5$). Also note, that equations (143) and (142) predict the propagation time in the full-adder correctly in the case of high Hill coefficients.

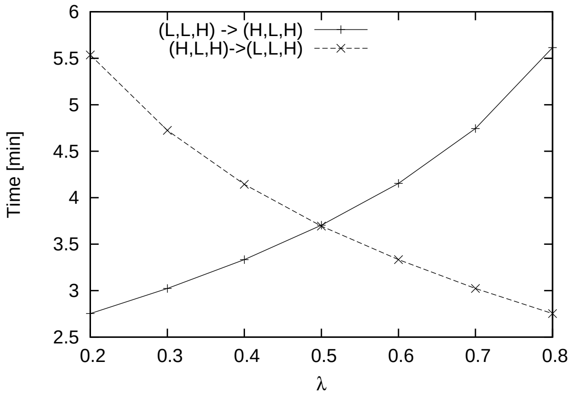


Figure 53: *The longest propagation time in the full-adder as a function of the threshold position.* We plotted the propagation time when switching between (L, L, H) to (H, L, H). The following set of parameters was used: $\mu = 1\ min^{-1}$, $l = 50$, $L = 0.2\ \mu M$, $H = 1.2\ \mu M$, $\alpha = 0.2\ \mu M \cdot min^{-1}$, $\beta = 1.0\ \mu M \cdot min^{-1}$ and $\theta = 0.9$.

Now that we determined the speed properties of the full-adder we turn our attention to the accuracy of this genetic system. We measure the accuracy by computing the noise at the output of the system (sum and carry in system 136). At steady state, the absolute value of the stochastic fluctuations (the *variance*) of the output z of a logic gate, which has two inputs x and y, can be written as [171, 45, 123]

$$\sigma_z^2 = \underbrace{z}_{\text{intrinsic}} + \underbrace{\overbrace{\left[\beta_z \frac{\partial f(x,y)}{\partial x} \tau_z\right]^2}^{\Gamma_{zx}} \overbrace{\frac{\tau_x}{\tau_x + \tau_z}}^{T_{zx}} \sigma_x^2}_{\text{upstream from } x} + \underbrace{\overbrace{\left[\beta_z \frac{\partial f(x,y)}{\partial y} \tau_z\right]^2}^{\Gamma_{zy}} \overbrace{\frac{\tau_y}{\tau_y + \tau_z}}^{T_{zy}} \sigma_y^2}_{\text{upstream from } y} . \qquad (144)$$

The first term in the right hand side, the intrinsic component, quantifies the fluctuations generated by the randomness of the birth-death processes. The second and the third term in the right hand side, the upstream component, represents the noise transmitted from the upstream species (the species which regulate the gene) [129]. The upstream noise is composed of three terms: the regulation factor (Γ_{zx} and Γ_{zy}), the time average factor (T_{zx} and T_{zy}), and the variance of the upstream species (σ_x^2 and σ_y^2).

In this contribution, we are interested in how noise affects our ability to distinguish between the two known output states, H and L. To get a meaningful measure of this, we will normalise the variance by the square of the signal strength, $\eta_z = \sigma_z^2/(H-L)^2$, rather than by the square of the mean (which is often used as a definition of noise, see chapter 4),

$$\eta_z = \frac{z}{(H-L)^2} + \left[\beta_z \tau_z \frac{\partial f(x,y)/\partial x}{(H-L)}\right]^2 T_{zx}\sigma_x^2 + \left[\beta_z \tau_z \frac{\partial f(x,y)/\partial y}{(H-L)}\right]^2 T_{zy}\sigma_y^2. \quad (145)$$

For a step-like regulation function, the derivatives in (145) will be zero and the extrinsic component will be zero as well. Thus, the only contribution to the noise is the intrinsic component and the noise of the output depends only on the steady state abundance (high and low), but is independent of the number of gates in the system or threshold position. However, if the threshold is close enough to one of the steady states (H or L), then small fluctuations in the input generates high fluctuations in the output and the analytical method is not accurate any more. Assuming that the threshold is positioned at the midpoint (optimum position for speed) and the two steady states are far enough from each other, then the noise will be determined only by the intrinsic component.

Summing up, we can state that, in the case of step-like regulation functions, the system displays an optimum threshold position ($\lambda = \lambda_T = \lambda_\eta = 0.5$) which ensures optimality both for speed and accuracy.

Sigmoid Gates

Above, we performed the optimality analysis of the full-adder in the case of step-like regulation functions and observed that the system displays an optimal configuration. This optimal configuration is achieved when the regulatory threshold is position at the midpoint between the high and the low steady states. Now, we turn our attention to the full-adder with sigmoid regulation functions. For low Hill coefficients, the optimum threshold in terms of speed in not positioned any more at the midpoint between the high and the low state (see Figure 54). This is a consequence of the fact that, for low Hill coefficients, the Hill function loses the symmetry around the threshold. We also note that decreasing the Hill coefficient increases the propagation time due to the fact that a gene is not instantly turned on/off when an input species crosses over/under the threshold (compare Figures 53 and 54). Increasing the Hill coefficient asymptotically reduces the propagation time to the one of the step-like regulation function and, thus, the optimal threshold will asymptotically approach the midpoint, $\lambda_T = 0.5$.

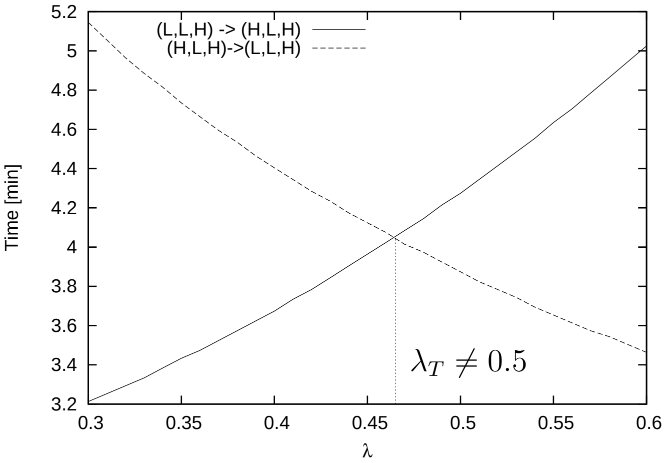


Figure 54: *Optimal threshold position for time in a system with sigmoid regulation functions.* We plotted the propagation time when switching between (L, L, H) to (H, L, H) for a low Hill coefficient. The following set of parameters was used: $\mu = 1\ min^{-1}$, $l = 6$, $L = 0.2\ \mu M$, $H = 1.2\ \mu M$, $K = 0.7\ \mu M$ and $\theta = 0.9$.

Apart from speed, our optimality analysis also requires the computation of noise. The highest noise levels of the sigmoid gate based full-adder, independent of the threshold position, are generated at the sum output when the input is (H, L, L). In this analysis we considered the worst case scenario and, thus, we determined the dependence of noise in the sum on the threshold position when the input is fixed at (H, L, L). The mathematical formula of the noise is too complicated to give any information about the system, but we can use it to generate numerical solutions. Figure 55 shows that there is an optimal position of the threshold in terms of noise which differs from the optimal position in terms of speed, $\lambda_\eta \neq \lambda_T$. Note however, that around the optimal threshold position in terms of noise (λ_η) the noise does not vary significantly (see Figure 55).

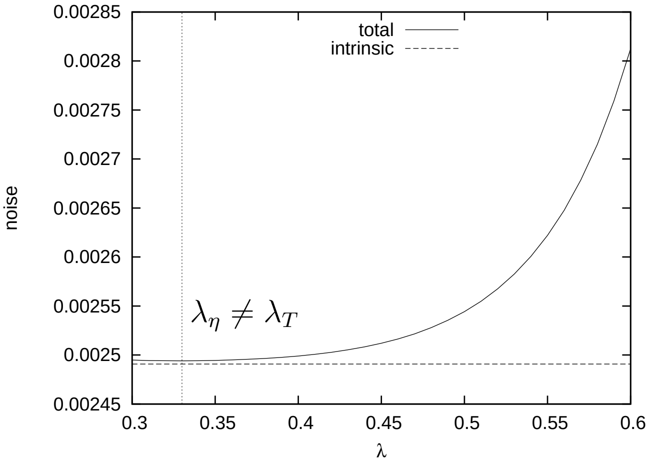


Figure 55: *Noise of the full-adder.* The noise dependence on the threshold position. The following set of parameters was used: $V = 8 \times 10^{-16}\ l$, $\mu = 1\ min^{-1}$, $l = 6$, $L = 0.2\ \mu M$, $H = 1.2\ \mu M$, $K = 0.7\ \mu M$ and $\theta = 0.9$. We assumed a Poisson noise in the three input species.

The system displays two optimal threshold positions, one for speed (λ_T) and one for noise (λ_η). If these two positions coincide ($\lambda_T = \lambda_\eta$) then the system has an optimal set of parameters and the threshold needs to take this value, $\lambda = \lambda_T = \lambda_\eta$.

However, it is most likely that these two threshold positions will differ, as it is the case with our sets of parameters for the full-adder. In this case, there is an optimal trade-off curve which is margined by the two optimal configurations (for time, λ_T, and for noise, λ_η). In addition any other trade-off curve is sub-optimal comparing to this one.

In our example of the full-adder we have $0.5 \leq \lambda_\eta \leq \lambda_T$. Figure 56 represents the trade-off between noise and time graphically based on the threshold position. We identified the optimal trade-off curve determined by $\lambda_\eta \leq \lambda \leq \lambda_T$. Any threshold in this interval can optimise the system either in speed or accuracy, but never in both. However, for threshold positions outside this interval, the system displays sub-optimal trade-off curves; for $\lambda < \lambda_\eta$ or $\lambda > \lambda_T$ both the propagation time and the noise are worse compared with the ones in the optimal trade-off curve.

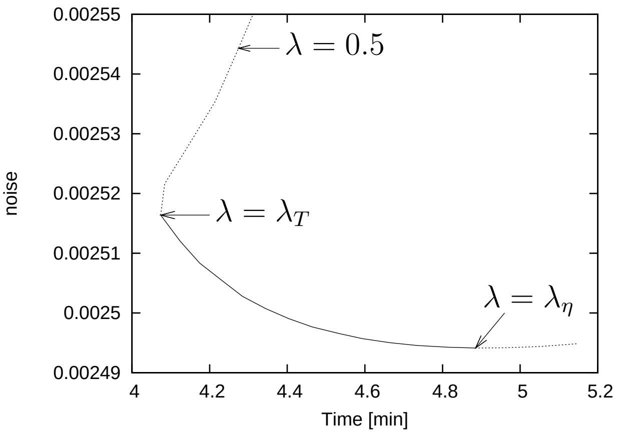


Figure 56: *Trade-off curve of the full-adder.* Optimal and sub-optimal trade-off curves of the full-adder. We used the following set of parameters: $V = 8 \times 10^{-16}\ l$, $\mu = 1\ min^{-1}$, $l = 6$, $L = 0.2\ \mu M$, $H = 1.2\ \mu M$, $K = 0.7\ \mu M$ and $\theta = 0.9$. We assumed a Poisson noise in the three input species.

7.4 Considerations on the Approach

Certain approximations or assumptions made in this chapter are likely to influence the results of our analysis. Thus, in the following paragraphs, we discuss these assumptions and approximations and identify how they influence the results presented in this chapter.

First, our optimality analysis addresses only logic gates formed of individual genes that are not auto-regulated. Network motifs (such as negative auto-regulation) can play a significant role in both speed and noise [5]. As we saw in the previous chapter, negative auto-regulation can lead to faster and more accurate systems. This suggests that further optimisation of our full-adder can be achieved by considering various network motifs such as negative auto-regulation. Although network motifs can lead to better performances of the system, our objective, in this contribution, was to present a general strategy and specific methods on how to design synthetic logic circuits. In this chapter, we did not aim to provide the best design for a genetic full-adder, but rather show how a certain logic design can be modelled from genes.

Furthermore, to achieve modularity and scalability of design we selected a set of synthesis rates (α and β), which we kept fixed throughout the optimisation process. In the previous chapter we compensated any change of the threshold by an adequate change in β so that the metabolic cost defined as $\zeta = \alpha + \beta f(x_H, K)$ would remain fixed. Due to the fact that, in this contribution, we could not change the synthesis rates any more, we had to relax our definition of metabolic cost and assume that the metabolic cost is computed as the maximum synthesis rate $\zeta_z = \alpha + \beta$. This value represents the worst case scenario of the previously defined metabolic cost, in the sense that our current definition of cost ζ_z is computed by assuming the maximum value of ζ, where $f(x_H, K)$ reached its highest value, 1. Both of these measures are just crude approximations of the actual metabolic costs, which also include degradation and maintenance costs. Nevertheless, in this contribution we are interested how the actual metabolic cost scales with the parameters of the gene and both definitions (ζ and ζ_z) capture this aspect, i.e., they measure how the metabolic cost scales with the gene expression rate.

Finally, to keep the mathematics tractable we assumed that all genes have identical parameters. This is unlikely to be biologically realistic. For specific genetic systems, similar analyses could be employed, where individual trade-off curves could be determined for each component (gene) in the network. We expect that the existence of optimal and sub-optimal trade-off curves controlled by the regulation threshold to be generally valid. Nevertheless, the details of these assumptions are left for further research.

7.5 Biological Significance

Our analysis investigated the relationship between the parameter space of a genetic system and the computational properties that characterise this system. In particular, we identified how the biological parameters of the genes control the properties of the system, but also how the change in one property affects other properties.

In this contribution, we presented a general method for constructing arbitrarily large logical systems based on binary genes. For exemplification purpose, we designed a full-adder system formed of five genes. We modelled logic gates which were constructed using two *cis*-regulatory transcription control regions. This type of logic gates was already engineered in live bacterial/eukaryote cells by synthetic biologists [71, 93, 182, 39, 8, 145]. Our analysis showed that the synthesis/decay rates can control the interconnectivity of different gates/genes and also proposed the set of parameters that ensure this interconnectivity.

In chapter 5 we showed that leak-free systems are optimal in terms of speed and noise [184]. However, equations (138) and (139) indicate that vanishing leak rates are very difficult to obtain. This suggests that leak-free systems, although optimal in speed and noise, are not always desirable because, when interconnecting genes, leak-free systems can require a higher metabolic cost compared with non-vanishing leak rates.

We also presented an approach for selecting the set of parameters which optimise the system in terms of speed and accuracy under constant metabolic cost. Increasing the Hill coefficient will optimise both the speed and the accuracy, but this usually comes at a higher metabolic cost. We showed that the threshold position, for a fixed Hill coefficient, influences both the speed (see Figure 54) and the noise (see Figure 55).

Finally, we would like to pinpoint that our approach relies heavily on setting specific values for the parameters of the gene. While this may be of reach to synthetic biologists, the accuracy of the measurements makes the entire approach impractical. For example, in Figure 46 (from the previous chapter) it was impossible to say whether the system is on the optimal trade-off curve or not. However, we expect that improvements in the measurement techniques and in the control of the genetic parameters will lead to a better usefulness for our approach.

7.6 Summary

In this chapter, we presented a specific scenario where the optimality analysis can be applied. As a sample system we used a binary full-adder system formed of five genes which mimic the behaviour of five logic gates. Our approach included the construction of all required logic gates from binary genes (see Figure 49), identification of the parameters of the genes which ensured interconnection of an arbitrary number of logic gates (see Figure 50 and equations 137, 138, 139) and also investigation of the optimal design in terms of speed and accuracy under the assumption of fixed metabolic cost.

In an ideal system, a system with gates that display step-like regulation functions (infinite Hill coefficients), we found that the system has an optimal set of parameters (threshold positioned at the midpoint between the two steady states). This set of parameters maximises both speed and accuracy for a fixed cost (see Figure 53). Moreover, the speed and the accuracy achieved in this type of system (step-like) are the asymptotic limit that any biological real system (sigmoid) can aim towards.

Real genes have finite low Hill coefficients and, in this case, a logic system will display two optimal parameter settings: one for speed (λ_T) and another one for noise (λ_η). We found that there is a trade-off curve between speed and accuracy which is bounded by these optimal sets of parameters (λ_T and λ_η) and any point between these two can optimise the system in either speed or accuracy (see Figure 56). Nevertheless, any other set of parameters (a threshold outside this interval) is sub-optimal with respect to accuracy or speed.

Chapter 8

Conclusions

In this thesis, we analysed binary genes as computational units. Binary genes are genes which display a switch-like behaviour, i.e., they are expressed mostly at either high or low rates and rarely at intermediary ones. This switching behaviour and the fact that genes are able to integrate multiple inputs indicate that binary genes are capable of performing logical computations.

We were interested in the computational limits of binary genes. In particular, we identified three properties which characterise genes and their computational performance, namely: (*i*) noise in gene expression, (*ii*) the response time and (*iii*) the energy cost of functioning. Our results showed that these three properties are interconnected, in the sense that there is a three way trade-off between them. This means that enhancing one of the properties cannot be achieved unless at least one other property is impaired. Furthermore, we revealed that an optimal configuration can be achieved by reducing the leak rate of the system, which comes at a higher metabolic cost. Additionally, for non-vanishing leak rates, the system could be optimised without increasing the metabolic cost only by adding negative auto-regulation.

The remainder of this chapter is sectioned as follows. In the next section, we will review the main results of this thesis and, in the final section, we will present further directions of our research.

8.1 Contributions

The model of binary genes, which we used in this thesis, is a coarse grain version of the biologically realistic one, i.e., we simulate in one step the protein synthesis process (regulation, transcription and translation). Nevertheless, despite its simplicity, our model still captures essential aspects of biological processes within a bacterial cell, such as: sigmoid regulation function, maximum expression rate, leaky expression and the rate at which proteins decay. The very simplicity of the model makes it particularly suitable for analytical analysis but also for extensive stochastic simulations.

Previous research, with few recent exceptions, investigated the three computational properties (speed, accuracy and cost) in an independent fashion. As we showed in this thesis, it is essential to consider the three properties in an integrated fashion, because there is a functional relationship between them, which limits their optimization.

Usually, we want to perform computations as fast and as accurately as possible, without increasing the cost. However, there is often an inherent trade-off between these properties, in the sense that enhancing one property can be achieved only by impairing another one [104]. In the case of binary genes, we observed that there is a three way trade-off between speed, accuracy and cost. In particular, for a fixed metabolic cost, there is a trade-off between speed and accuracy, i.e., the speed can be increased only by decreasing accuracy and, conversely, the accuracy can be enhanced only by reducing the speed.

For a stand-alone binary gene, this trade-off is controlled by the decay rate of the output protein, in the sense that higher decay rates lead to faster and noisier systems, while lower ones to slower but more accurate systems. This result indicates that the cell has the ability to control the accuracy and speed of individual genes. For example, if a protein is decayed only by dilution, then the dynamics of the protein will be slow (the half-life time of proteins affected by dilution is approximately equal to the division time of the cell, e.g., 50 min) while the accuracy will be high. Conversely, if a protein is actively broken down, then the dynamics will be fast and, consequently, the noise will be higher.

In addition, we found that increasing the cost leads to better trade-off curves between speed and accuracy. This underlines the importance of comparing systems with equal metabolic cost. In chapter 2, we enumerated a few examples from the literature that identified speed-accuracy trade-offs. Mehta *et al.* [108] determined that the trade-off between switching time and noise in a bi-stable genetic system (an auto-activator gene) was controlled by the burst size of translation. Although they considered a constant number of molecules for the output protein, a higher number of transcripts (resulting from a smaller burst size and equal protein concentration) suggests that the metabolic cost is increased [106, 167]. Similarly, Stojanovic and Stefanovic found that the speed-accuracy trade-off in their DNA based logic gates was controlled by the length of DNA strands [160]. However, higher length in the DNA strand means higher cost in their scenario and, thus, the trade-off is achieved again by changing the cost.

The results of our analysis show that the speed and accuracy can be increased as much as we want by increasing the cost, but this would be biologically unrealistic (cells have a limited amount of energy resources available). This indicates that an analysis on speed and accuracy needs to consider the metabolic cost. In this context, it is worthwhile mentioning Tan *et al.* [163], who demonstrated that there is a speed-cost trade-off in a genetic system where signals were encoded by oscillatory protein abundances. Their analysis revealed that the speed can be increased arbitrarily by increasing the cost.

Furthermore, we proved that this three way trade-off between speed, accuracy and cost is optimal for genes without leak expression. This can be explained by the fact that, for equal metabolic costs, leaky expression leads to higher noise. However, the reduction of the leak rate, usually, comes at a very high and unjustifiable metabolic cost. Thus, in many cases, the system can afford to have a certain amount of leak rate given that this results in worse speed and accuracy performance, but reasonable metabolic cost.

When genes are not stand-alone elements, but rather components in a genetic network, the input cannot change instantaneously any more, i.e., the input in a downstream gene is the output of an upstream one and needs to either build up or be decayed. In this case, we observed another trade-off between speed and accuracy, but this time the trade-off is controlled by the regulatory threshold. The regulatory threshold is the input required for half activation of a gene and it can be approximated by the ratio between the unbinding and the binding rate of the regulatory transcription factors to the *cis*-regulatory area of the gene. Thus, the affinity of the TF for the *cis*-regulatory area controls the speed and the accuracy of the binary gene. We found that there are two threshold values, one that optimises the system in speed and another one which optimises the system in accuracy. Note that Murphy *et al.* [115] showed the existence of the optimal threshold for noise experimentally and, thus, provided biological support for our results. Furthermore, the threshold values bounded by these two optimal values produce an optimal trade-off curve between speed and accuracy, while everything outside this curve is sub-optimal.

We expect that real cells display both optimal and sub-optimal configurations. This could be explained by the inherent heterogeneity of the population, which is a useful survival strategy for changes in the environment of the cells [135, 1], i.e., when the environment changes, previous sub-optimal cells can be optimal in the new environment and have higher chances of survival.

The genetic regulatory network of bacterial cells displays various network motifs, which are sub-networks that occur more often than in a random network. One important network motif in bacterial cells is the negatively auto-regulated gene [5]. This specific motif was indicated in the literature as being able to reduce both turn-on times and noise. We investigated the optimality of negative auto-regulation by comparing auto-repressed genes to simple ones under the assumption of equal metabolic cost. Our results confirmed the results of Rosenfeld *et al.* [137] and showed that, for leak-free systems, negative auto-regulation can enhance only turn-on speed. However, we found that, for leaky systems, negative auto-regulation can enhance both turn-on and turn-off speeds. This suggests that, despite being harmful for genes (it increases the noise for certain sets of parameters), leaky expression can also bring benefits to the system (it enhances the speed in the case of negative feedback). By increasing the leak rate, the auto-repressed gene becomes worse than the simple gene in terms of noise. Thus, there is another speed-accuracy trade-off but this time the trade-off is controlled by the leak rate of the gene.

Furthermore, we found that there are certain leak rate values for which the negative auto-regulation enhances both speed and accuracy. Our results indicate, that for low (but non zero) leak rates, negative auto-regulation enhances both speed and accuracy. Nevertheless, for higher leak rates, the negatively auto-regulated gene becomes noisier, but is still able to display a faster speed than the simple gene. In this latter case, the gene has to choose again between being faster or more accurate. This superiority of negative auto-regulation for certain (biologically plausible) sets of parameters can explain the high occurrence of this motif in bacterial genomes (for example, 40% of the genes in *E.coli* are negatively auto-regulated [12]). Nevertheless, the fact that there are biologically plausible sets of parameters for which the simple gene is better in one property (accuracy) than the negatively auto-regulated one can explain the reason for not all genes being negatively auto-regulated.

In this thesis, we identified various mechanisms within the cell which control this speed-accuracy trade-off under the assumption of fixed metabolic cost. The current configurations of real cells are the result of a long time evolution, in which a system adapted to and was optimised to respond to various environmental conditions. The various trade-offs identified in this thesis suggest that the cell adapts and changes its behaviour (and, consequently, its performance properties) by changing the parameters of the genes. For example, a single point mutation in the promoter area of a gene can change the configuration of a system from the most accurate one to the fastest one; see Figure 46 from chapter 6.

To exemplify our optimality analysis we considered, as a case study, a full-adder constructed from genes. We determined that the design process of a genetic logic system is a three step process. The first step consisted of selecting a logical diagram and modelling the logic gates. We did this by assuming that genes are able to integrate inputs in the *cis*-regulatory area and the activation state of the gene mimics a logic function where the inputs are represented by the regulatory transcription factors. Furthermore, we computed the synthesis rates of the genes which allow the interconnection of the gates. This interconnectivity was ensured by imposing that changes in the output of an upstream gate are reflected in the output of a downstream one, but also by allowing the system to contain an arbitrary number of gates without worsening the binary behaviour of the output (we call this the *scalability* of the system). Finally, we performed our optimality analysis and identified the two optimal configurations, the fastest and the most accurate one.

Our modelling approach was based on selecting sets of parameters for the genes. Current laboratory techniques (high errors in the measurements and poor control of parameters) make our approach impractical for synthetic biologists. Nevertheless, we can speculate that future improvements in these techniques can make our analysis a useful tool for fine tuning the performance properties of a synthetic system.

8.2 Further Work

Based on our model and results, but also on the alternative solutions proposed in the literature review, we identified several potential routes for future research.

In this thesis, we built a model where we assumed that the transcription factors are bound to their binding sites or otherwise suspended in perfectly mixed cytoplasm. It is unlikely that this model is correct for real cells. Instead, the concentration of a specific molecular species will be highly non-uniform within the cytoplasm [26, 177]. It would be interesting to investigate how these spatial aspects influence the speed at which genes are regulated and the noise characteristics of the product protein. In particular, we would like to investigate how various scenarios (such as crowding of transcription factors and colocalization of genes) influence the fluctuations in the activity state of a gene and the time it takes for transcription factors to find their target sites.

In this thesis we disregarded protein-protein interactions (enzymatic gates) and these types of interactions have a very important role within the functioning of cells. The main advantage that these types of interactions have compared with the transcriptional gates consists of the high speed at which they are able to process information. A similar optimality analysis as in the case of transcriptional gates can also be performed on protein gates. We expect to observe similar trade-offs between speed, accuracy and cost. Nevertheless, the range of parameters which are controllable in this type of gate is smaller compared with the one of transcriptional gates, and this is likely to bring some differences in the relationships which we identified between the parameter space and the properties that characterise the system.

A solution which combines both protein-protein interactions and transcriptional gates could offer an alternative design to the gates modelled in this thesis. In our analysis, high steepness leads to faster and more accurate systems and is usually difficult to achieve. If we consider weak protein-protein interactions, such as dimerisation, then this usually leads to an increase in the Hill coefficient. Alternatively, if we assume a system with two input signals integrated in a multi-enzyme gate, then the gate could display zero order sensitivity (step-like behaviour) [162]. These types of systems can lead to a better trade-off between speed, accuracy and cost. It would be interesting to observe how protein-protein interaction can enhance our design.

Depending on the state of the *cis*-regulatory area, genes can have different steady states and, consequently, the gene expression function can display more than two plateaus [52]. This type of behaviour can lead to computations being performed in bases higher than two, which could result in more computations being performed in the same amount of time. We would like to investigate under which conditions multi steady state behaviour arises and how we can optimise a synthetic *n-state* gene.

There are significant differences between prokaryotic and eukaryotic organisms [133]. Eukaryotic genes are usually regulated by more transcription factors than prokaryotic ones and the *cis*-regulatory regions of eukaryotic genes can be far away from the promoters of genes, allowing distal regulation [85]. Additionally, the eukaryotic DNA is organised into chromatin which has a significant influence on the transcriptional regulation. We would like to investigate under which conditions the differences that eukaryotic cells display compared with prokaryotic ones influence or change our main findings.

A final possible extension of the work presented in this thesis would be to implement in live cells a synthetic full-adder using transcriptional logic gates. This would require the identification of a design in accordance with available genes that mimic logic gates and then selection and interconnection of these genes. Once constructed, we could then tune the system to display the fastest and the most accurate configurations.

Bibliography

[1] Murat Acar, Jerome T Mettetal, and Alexander van Oudenaarden. Stochastic switching as a survival strategy in fluctuating environments. *Nature Genetics*, 40(4):471–475, 2008.

[2] Gary K. Ackers, Alexander D. Johnson, and Madeline A. Shea. Quantitative model for gene regulation by lambda phage repressor. *PNAS*, 79:1129–1133, 1982.

[3] Leonard M. Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266:1021–1024, 1994.

[4] Hiroshi Akashi and Takashi Gojobori. Metabolic efficiency and amino acid composition in the proteomes of escherichia coli and bacillus subtilis. *PNAS*, 99(6):3695–3700, 2002.

[5] Uri Alon. *An Introduction To System Biology. Design Principles of Biological Circuits.* Chapman & Hall/CRC Mathematical and Computational Biology Series, 2007.

[6] Martyn Amos. *Encyclopedia of Complexity and System Science*, chapter DNA Computing. Springer, 2008.

[7] J Christopher Anderson, Elizabeth J. Clarke, Adam P. Arkin, and Christopher A. Voigt. Environmentally controlled invasion of cancer cells by engineered bacteria. *Journal of Molecular Biology*, 355(4):619–627, 2006.

[8] J Christopher Anderson, Christopher A Voigt, and Adam P Arkin. Environmental signal integration by a modular and gate. *Molecular Systems Biology*, 3, 2007.

[9] Ernesto Andrianantoandro, Subhayu Basu, David K Karig, and Ron Weiss. Synthetic biology: new engineering rules for an emerging discipline. *Molecular Systems Biology*, 4100073:E1–E14, 2006.

[10] Adam Arkin and John Ross. Computational functions in biochemical reaction networks. *Biophysical Journal*, 67(2):560–578, 1994.

[11] Adam Arkin, John Ross, and Harley H. McAdams. Stochastic kinetic analysis of developmental pathway bifurcation in phage l-infected escherichia coli cells. *Genetics*, 149:1633–1648, 1998.

[12] D. W. Austin, M. S. Allen, J. M. McCollum, R. D. Dar, J. R. Wilgus, G. S. Sayler, N. F. Samatova, C. D. Cox, and M. L. Simpson. Gene network shaping of inherent noise spectra. *Nature*, 439(2):608–611, 2006.

[13] Arren Bar-Even, Johan Paulsson, Narendra Maheshri, Miri Carmi, Erin O'Shea, Yitzhak Pilpel, and Naama Barkai. Noise in protein expression scales with natural protein abundance. *Nature Genetics*, 38(6):636–643, 2006.

[14] Ronan Baron, Oleg Lioubashevski, Eugenii Katz, Tamara Niazov, and Itamar Willner. Elementary arithmetic operations by enzymes: A model for metabolic pathway based computing. *Angewandte Chemie*, 45:1572–1576, 2006.

[15] Ronan Baron, Oleg Lioubashevski, Eugenii Katz, Tamara Niazov, and Itamar Willner. Logic gates and elementary computing by enzymes. *J. Phys. Chem. A*, 110(27):8548–8553, 2006.

[16] Subhayu Basu, Yoram Gerchman, Cynthia H. Collins, Frances H. Arnold, and Ron Weiss. A synthetic multicellular system for programmed pattern formation. *Nature*, 434:1130 –1134, 2005.

[17] Subhayu Basu, Rishabh Mehreja, Stephan Thiberge, Ming-Tang Chen, and Ron Weiss. Spatiotemporal control of gene expression with pulse-generating networks. *PNAS*, 101(17):6355–6360, 2004.

[18] Jordan Baumgardner, Karen Acker, Oyinade Adefuye, Samuel Thomas Crowley, Will DeLoache, James O Dickson, Lane Heard, Andrew T Martens, Nickolaus Morton, Michelle Ritter, Amber Shoecraft, Jessica Treece, Matthew Unzicker, Amanda Valencia, Mike Waters, A Malcolm Campbell, Laurie J Heyer, Jeffrey L Poet, and Todd T Eckdahl. Solving a hamiltonian path problem with a bacterial computer. *Journal of Biological Engineering*, 3(11), 2009.

[19] Attila Becskei, Benjamin B Kaufmann, and Alexander van Oudenaarden. Contributions of low molecule number and chromosomal positioning to stochastic gene expression. *Nature Genetics*, 37(9):937–944, 2005.

[20] Attila Becskei and Luis Serrano. Engineering stability in gene networks by autoregulation. *Nature*, 405:590–593, 2000.

[21] Arieh Ben-Naim. Cooperativity in binding of proteins to dna. *Journal of Chemical Physics*, 107(23):10242–10252, 1997.

[22] Arieh Ben-Naim. Cooperativity in binding of proteins to dna. ii. binding of bacteriophage l repressor to the left and right operators. *Journal of Chemical Physics*, 108(16):6937–6946, 1998.

[23] Yaakov Benenson, Binyamin Gil, Uri Ben-Dor, Rivka Adar, and Ehud Shapiro. An autonomous molecular computer for logical control of gene expression. *Nature*, 429:423–429, 2004.

[24] Charles H. Bennett. Logical reversibility of computation. *IBM Journal of Research and Development*, 17(6):525–532, 1973.

[25] Charles H. Bennett. The thermodynamics of computation – a review. *International Journal of Theoretical Physics*, 21(12):905–940, 1982.

[26] Otto G. Berg, Robert B. Winter, and Peter H. von Hippel. Diffusion-driven mechanisms of protein translocation on nucleic acids. 1. models and theory. *Biochemistry*, 20(24):6929–6948, 1981.

[27] Lacramioara Bintu, Nicolas E. Buchler, Hernan G. Garcia, Ulrich Gerland, Terence Hwa, Jane Kondev, and Rob Phillips. Transcriptional regulation by the numbers: models. *Current Opinion in Genetics and Development*, 15:116–124, 2005.

[28] Ravinderjit S. Braich, Nickolas Chelyapov, Cliff Johnson, Paul W. K. Rothemund, and Leonard Adleman. Solution of a 20-variable 3-sat problem on a dna computer. *Science*, 296:499–502, 2002.

[29] Dennis Bray. Protein molecules as computational elements in living cells. *Nature*, 376:307–312, 1995.

[30] Jonathan E. Bronson, William W. Mazur, and Virginia W. Cornish. Transcription factor logic using chemical complementation. *Molecular BioSystems*, 4:56–58, 2008.

[31] Frank J. Bruggeman, Nils Blthgen, and Hans V. Westerhoff. Noise management by molecular networks. *PLoS Computational Biology*, 5(9):e1000506, 2009.

[32] Nicolas E. Buchler, Ulrich Gerland, and Terence Hwa. On schemes of combinatorial transcription logic. *PNAS*, 100(9):5136–5141, 2003.

[33] Nicolas E. Buchler, Ulrich Gerland, and Terence Hwa. Nonlinear protein degradation and the function of genetic circuits. *PNAS*, 102(27):9559–9564, 2005.

[34] Irwin K. Cheah, Steven J. Langford, and Melissa J. Latter. Concept transferfrom genetic instruction to molecular logic. *Supramolecular Chemistry*, 17:121–128, 2005.

[35] Joshua L. Cherry and Frederick R. Adler. How to make a biological switch. *Journal of Theoretical Biology*, 203(2):117–133, March 2000.

[36] Dominique Chu, Nicolae Radu Zabet, and Boris Mitavskiy. Models of transcription factor binding: Sensitivity of activation functions to model assumptions. *Journal of Theoretical Biology*, 257(3):419–429, 2009.

[37] Dominique F. Chu, Nicolae Radu Zabet, and Andrew N. W. Hone. Optimal parameter settings for information processing in gene regulatory networks. *BioSystems*, 2011.

[38] Sean M. Cory and Theodore J. Perkins. Implementing arithmetic and other analytic operations by transcriptional regulation. *PLoS Computational Biology*, 4(5):e1000064, May 2008.

[39] Robert Sidney Cox III, Michael G Surette, and Michael B Elowitz. Programming gene expression with combinatorial promoters. *Molecular Systems Biology*, 3, 2007.

[40] Gheorghe Craciun, Yangzhong Tang, and Martin Feinberg. Understanding bistability in complex enzyme-driven reaction networks. *PNAS*, 103(23):8697–8702, 2006.

[41] A Prasanna de Silva and Seiichi Uchiyama. Molecular logic and computing. *Nature Nanotechnology*, 2:399–410, 2007.

[42] Erez Dekel and Uri Alon. Optimality and evolutionary tuning of the expression level of a protein. *Nature*, 436:588–592, 2005.

[43] Andrew S. Deonarine, Sonya M. Clark, and Lars Konermann. Implementation of a multifunctional logic gate based on folding/unfolding transitions of a protein. *Future Generation Computer Systems*, 19(1):87–97, 2003.

[44] Mary J Dunlop, Robert Sidney Cox III, Joseph H Levine, Richard M Murray, and Michael B Elowitz. Regulatory activity revealed by dynamic correlations in gene expression noise. *Nature Genetics*, 40(12):1493–1498, 2008.

[45] Johan Elf and Mans Ehrenberg. Fast evaluation of fluctuations in biochemical networks with the linear noise approximation. *Genome Research*, 13:2475–2484, 2003.

[46] Johan Elf, Johan Paulsson, Otto G. Berg, and Mans Ehrenberg. Near-critical phenomena in intracellular metabolite pools. *Biophysical Journal*, 84:154–170, 2003.

[47] Michael B. Elowitz and Stanislas Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403:335–338, 2000.

[48] Michael B. Elowitz, Arnold J. Levine, Eric D. Siggia, and Peter S. Swain. Stochastic gene expression in a single cell. *Science*, 297(5584):1183–1186, 2002.

[49] Dirk Faulhammer, Anthony R. Cukras, Richard J. Lipton, and Laura F. Landweber. Molecular computation: Rna solutions to chess problems. *PNAS*, 97(4):1385–1389, 2000.

[50] Chrisantha T Fernando, Anthony M.L Liekens, Lewis E.H Bingle, Christian Beck, Thorsten Lenser, Dov J Stekel, and Jonathan E Rowe. Molecular circuits for associative learning in single-celled organisms. *J. R. Soc. Interface*, 6(34):463–469, 2009.

[51] James E Ferrell. Self-perpetuating states in signal transduction: positive feedback, double-negative feedback and bistability. *Current Opinion in Chemical Biology*, 6:140–148, 2002.

[52] Ari E. Friedland, Timothy K. Lu, Xiao Wang, David Shi, George Church, and James J. Collin. Synthetic gene networks that count. *Science*, 324(5931):1199–1202, 2009.

[53] Nir Friedman, Long Cai, and X. Sunney Xie. Linking stochastic dynamics to population distribution: An analytical framework of gene expression. *Physical Review Letters*, 97:168392, 2006.

[54] Pierluigi Frisco, Peter Cook, and Paul A. Hoskisson. Solving the hamiltonian path problem using viral dna and bacteria. Technical report, Heriot-Watt University, 2010.

[55] Nikhil Gandhi, Gonen Ashkenasy, and Emmanuel Tannenbaum. Associative learning in biochemical networks. *Journal of Theoretical Biology*, 249:58–66, 2007.

[56] Crispin W. Gardiner. *Handbook of Stochastic Methods: For Physics, Chemistry and the Natural Sciences*. Springer-Verlag, 1982.

[57] Timothy S. Gardner, Charles R. Cantor, and James J. Collins. Construction of a genetic toggle switch in escherichia coli. *Nature*, 403:339–342, January 2000.

[58] Moritz Gerstung, Jens Timmer, and Christian Fleck. Noisy signaling through promoter logic gates. *Physical Review E*, 79:011923, 2009.

[59] Nathan C. Gianneschi and M. Reza Ghadiri. Design of molecular logic devices based on a programmable dna-regulated semisynthetic enzyme. *Angewandte Chemie*, 119:4029–4032, 2007.

[60] Michael A. Gibson and Jehoshua Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *The Journal of Physical Chemistry A*, 104:1876–1889, 2000.

[61] Daniel T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4):403–434, 1976.

[62] Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81:2340–2361, 1977.

[63] Daniel T. Gillespie. A rigorous derivation of the chemical master equation. *Physica A: Statistical Mechanics and its Applications*, 188(1-3):404–425, 1992.

[64] Daniel T. Gillespie. The chemical langevin equation. *Journal of Chemical Physics*, 113(1):297–306, 2000.

[65] Daniel T. Gillespie. Stochastic simulation of chemical kinetics. *Annual Review of Physical Chemistry*, 58:35–55, 2007.

[66] Daniel T. Gillespie and Linda R. Petzold. Improved leap-size selection for accelerated stochastic simulation. *Journal of Chemical Physics*, 119(16):8229–82234, 2003.

[67] Daniel T. Gillespie and Linda R. Petzold. *System Modeling in Cell Biology: From Concepts to Nuts and Bolts*, chapter Numerical Simulation for Biochemical Kinetics, pages 331–353. MIT Press, 2006.

[68] Albert Goldbeter and Daniel E. Koshland Jr. An amplified sensitivity arising from covalent modification in biological systems. *PNAS*, 78(11):6840–6844, November 1981.

[69] Ido Golding, Johan Paulsson, Scott M. Zawilski, and Edward C. Cox. Realtime kinetics of gene activity in individual bacteria. *Cell*, 123(6):1025–1036, 2005.

[70] Carlos Gomez-Uribe, George C. Verghese, and Leonid A. Mirny. Operating regimes of signaling cycles: Statics, dynamics, and noise filtering. *PLoS Computational Biology*, 3(12):e246, 2007.

[71] Calin C. Guet, Michael B. Elowitz, Weihong Hsing, and Stanislas Leibler. Combinatorial synthesis of genetic networks. *Science*, 296:1466–1470, 2002.

[72] Stephen E. Halford and John F. Marko. How do site-specific dna-binding proteins find their targets? *Nucleic Acids Research*, 32(10):3040–3052, 2004.

[73] Juris Hartmanis. On the weight of computations. *Bulletin of the European Association for Theoretical Computer Science*, 55:136–138, 1995.

[74] Sikander Hayat and Thomas Hinze. Toward integration of in vivo molecular computing devices: successes and challenges. *HFSP Journal*, 2(5):239–243, 2008.

[75] Donald T. Haynie. *Biological Thermodynamics*. Cambridge University Press, 2001.

[76] Fernand Hayot and Ciriyam Jayaprakash. The linear noise approximation for molecular fluctuations within cells. *Physical Biology*, 1:205–210, 2004.

[77] John Heath, Marta Kwiatkowska, Gethin Norman, David Parker, and Oksana Tymchyshyn. Probabilistic model checking of complex biological pathways. *Theoretical Computer Science*, 391:239–257, 2008.

[78] Rutger Hermsen, Sander Tans, and Pieter Rein ten Wolde. Transcriptional regulation by competing transcription factor modules. *PLoS Computational Biology*, 2:1552–1560, 2006.

[79] Allen Hjelmfelt and John Ross. Chemical implementation and thermodynamics of collective neural networks. *PNAS*, 89:388–391, 1992.

[80] Allen Hjelmfelt, Edward D. Weinberger, and John Ross. Chemical implementation of neural networks and turing machines. *PNAS*, 88:10983–10987, 1991.

[81] Allen Hjelmfelt, Edward D. Weinberger, and John Ross. Chemical implementation of finite-state machines. *PNAS*, 89:383–387, 1992.

[82] Sara Hooshangi, Stephan Thiberge, and Ron Weiss. Ultrasensitivity and noise propagation in a synthetic transcriptional cascade. *PNAS*, 102(10):3581–3586, 2005.

[83] Sara Hooshangi and Ron Weiss. The effect of negative feedback on noise propagation in transcriptional gene networks. *Chaos*, 16:026108, 2006.

[84] Gil Hornung and Naama Barkai. Noise propagation and signaling sensitivity in biological networks: A role for positive feedback. *PLoS Computational Biology*, 4(1):0055–0061, 2008.

[85] Meredith L. Howard and Eric H. Davidson. cis-regulatory control circuits in development. *Developmental Biology*, 271(1):109–118, 2004.

[86] Farren J. Isaacs, William J. Blake, and James J. Collins. Signal processing in single cells. *Science*, 307(5717):1886–1888, 2005.

[87] Mads Kaern, Timothy C. Elston, William J. Blake, and James J. Collins. Stochasticity in gene expression: from theories to phenotypes. *Nature Reviews Genetics*, 6:451–464, 2005.

[88] Tomer Kalisky, Erez Dekel, and Uri Alon. Costbenefit theory and optimal design of gene regulation functions. *Physical Biology*, 4:229–245, 2007.

[89] Benjamin B Kaufmann and Alexander van Oudenaarden. Stochastic gene expression: from single molecules to the proteome. *Current Opinion in Genetics & Development*, 17:107–112, 2007.

[90] Thomas B. Kepler and Timothy C. Elston. Stochasticity in transcriptional regulation: Origins, consequences, and mathematical representations. *Biophysical Journal*, 81:3116–3136, 2001.

[91] Hideki Kobayashi, Mads Kaern, Michihiro Araki, Kristy Chung, Timothy S. Gardner, Charles R. Cantor, and James J. Collins. Programmable cells:

Interfacing natural and engineered gene networks. *PNAS*, 101(22):8414–8419, 2004.

[92] Grigory Kolesov, Zeba Wunderlich, Olga N. Laikova, Mikhail S. Gelfand, and Leonid A. Mirny. How gene order is influenced by the biophysics of transcription regulation. *PNAS*, 104(35):13948–13953, 2007.

[93] Beat P. Kramer, Cornelius Fischer, and Martin Fussenegger. Biologic gates enable logical transcription control in mammalian cells. *Biotechnology and Bioengineering*, 87(4):478–484, 2004.

[94] Marta Kwiatkowska, Gethin Norman, and David Parker. Prism 2.0: a tool for probabilistic model checking. *Quantitative Evaluation of Systems, 2004. QEST 2004. Proceedings. First International Conference on the*, pages 322–323, Sept. 2004.

[95] Marta Kwiatkowska, Gethin Norman, and David Parker. Probabilistic model checking in practice: case studies with prism. *ACM SIGMETRICS Performance Evaluation Review*, 32:16 – 21, 2005.

[96] Harvey Lederman, Joanne Macdonald, Darko Stefanovic, and Milan N. Stojanovic. Deoxyribozyme-based three-input logic gates and construction of a molecular full adder. *Biochemistry*, 45(4):1194–1199, 2006.

[97] Jinzhi Lei. Stochasticity in single gene expression with both intrinsic noise and fluctuation in kinetic parameters. *Journal of Theoretical Biology*, 256:485 – 492, 2009.

[98] Heon Man Lim, Kelly Hughes, and Melvin I. Simon. The effects of symmetrical recombinationsite hixc on hin recombinase function. *Journal of Biological Chemistry*, 267(5):11183–11190, 1992.

[99] Richard J. Lipton. Dna solution of hard computational problems. *Science*, 268(5210):542–545, 1995.

[100] Dongsheng Liu and Shankar Balasubramanian. A proton-fuelled dna nanomachine. *Angewandte Chemie*, 42(46):5734–5736, 2003.

[101] Qinghua Liu, Liman Wang, Anthony G. Frutos, Anne E. Condon, Robert M. Corn, and Lloyd M. Smith. Dna computing on surfaces. *Nature*, 403:175–179, 2000.

[102] Seth Lloyd. Ultimate physical limits to computation. *Nature*, 406:1047–1054, 2000.

[103] Marcelo O. Magnasco. Chemical kinetics is turing universal. *Physical Review Letters*, 78(6):1190–1193, 1997.

[104] James A.R. Marshall, Anna Dornhaus, Nigel R. Franks, and Tim Kovacs. Noise, cost and speed-accuracy trade-offs: decision-making in a decentralized system. *Journal of Royal Society Interface*, 3(7):243–254, 2006.

[105] Avraham E. Mayo, Yaakov Setty, Seagull Shavit, Alon Zaslaver, and Uri Alon. Plasticity of the cis-regulatory input function of a gene. *PLOS Biology*, 4(4):e45, 2006.

[106] Harley H. McAdams and Adam Arkin. Stochastic mechanisms in gene expression. *PNAS*, 94:814–819, 1997.

[107] Javier I. Medina-Bellver, Patricia Marin, Antonio Delgado, Alicia Rodriguez-Sanchez, Emilio Reyes, Juan L. Ramos, and Silvia Marques. Evidence for *in situ* crude oil biodegradation after the *Prestige* oil spill. *Environmental Microbiology*, 7(6):773–779, 2005.

[108] Pankaj Mehta, Ranjan Mukhopadhyay, and Ned S Wingreen. Exponential sensitivity of noise-driven switching in genetic networks. *Physical Biology*, 5, 2008.

[109] Dmitriy Melnikov, Guinevere Strack, Marcos Pita, Vladimir Privman, and Evgeny Katz. Analog noise reduction in enzymatic logic gates. *J. Phys. Chem. B*, 113(30):10472–10479, 2009.

[110] Leonor Michaelis and Maud L. Menten. Kinetics of invertase action. *Biochem*, 49:333, 1913.

[111] Jacques Monod, Jean-Paul Changeux, and Francois Jacob. Allosteric proteins and cellular control systems. *Journal of Molecular Biology*, 6:306–329, 1963.

[112] Jacques Monod, Jeffries Wyman, and Jean-Paul Changeux. On the nature of allosteric transitions: A plausible model. *Journal of Molecular Biology*, 12:88–118, 1965.

[113] Marco J. Morelli, Rosalind J. Allen, Sorin Tanase-Nicola, and Pieter Rein ten Wolde. Eliminating fast reactions in stochastic simulations of biochemical networks: a bistable genetic switch. *The Journal of Chemical Physics*, 128:169901, 2008.

[114] Mikhail Motornov, Jian Zhou, Marcos Pita, Venkateshwarlu Gopishetty, Ihor Tokarev, Evgeny Katz, and Sergiy Minko. Chemical transformers from nanoparticle ensembles operated with logic. *Nano Lett.*, 8(9):2993–2997, 2008.

[115] Kevin F. Murphy, Rhys M. Adams, Xiao Wang, Gabor Balazsi, and James J. Collins. Tuning and controlling gene expression noise in synthetic gene networks. *Nucleic Acids Research*, 38(8):2712–2726, 2010.

[116] James D. Murray. *Mathematical Biology I. An Introduction.* Springer, 2002.

[117] Dmitry Nevozhay, Rhys M. Adams, Kevin F. Murphy, Kresimir Josic, and Gabor Balazsia. Negative autoregulation linearizes the doseresponse and suppresses the heterogeneity of gene expression. *PNAS*, 106(13):5123–5128, 2009.

[118] John R. S. Newman, Sina Ghaemmaghami, Jan Ihmels, David K. Breslow, Matthew Noble, Joseph L. DeRisi, and Jonathan S. Weissman. Single-cell

proteomic analysis of s. cerevisiae reveals the architecture of biological noise. *Nature*, 441:840–846, 2006.

[119] Nam-Phuong D. Nguyen, Hiroyuki Kuwahara, Chris J. Myers, and James P. Keener. The design of a genetic muller c-element. *Asynchronous Circuits and Systems, International Symposium on*, 0:95–104, 2007.

[120] Tamara Niazov, Ronan Baron, Eugenii Katz, Oleg Lioubashevski, and Itamar Willner. Concatenated logic gates using four coupled biocatalysts operating in series. *PNAS*, 103(46):17160–17163, 2006.

[121] Akimitsu Okamoto, Kazuo Tanaka, and Isao Saito. Dna logic gates. *J. Am. Chem. Soc.*, 126(30):9458–9463, 2004.

[122] Ertugrul M. Ozbudak, Mukund Thattai, Iren Kurtser, Alan D. Grossman, and Alexander van Oudenaarden. Regulation of noise in the expression of a single gene. *Nature Genetics*, 31:69–73, 2002.

[123] Johan Paulsson. Summing up the noise in gene networks. *Nature*, 427:415–418, 2004.

[124] Johan Paulsson. Models of stochastic gene expression. *Physics of Life Reviews*, 2:157–175, 2005.

[125] Johan Paulsson, Otto G. Berg, and Mans Ehrenberg. Stochastic focusing: Fluctuation-enhanced sensitivity of intracellular regulation. *PNAS*, 97(13):7148–7153, 2000.

[126] Johan Paulsson and Mans Ehrenberg. Random signal fluctuations can reduce random fluctuations in regulated components of chemical regulatory networks. *Physical Review Letters*, 84(23):5447–5450, 2000.

[127] Johan Paulsson and Johan Elf. *System Modeling in Cell Biology: From Concepts to Nuts and Bolts*, chapter Stochastic modeling of intracellular kinetics, pages 149–176. MIT Press, 2006.

[128] Juan M. Pedraza and Johan Paulsson. Effects of molecular memory and bursting on fluctuations in gene expression. *Science*, 319(5861):339–343, 2008.

[129] Juan M. Pedraza and Alexander van Oudenaarden. Noise propagation in gene networks. *Science*, 307:1965–1969, 2005.

[130] Jason R. Pirone and Timothy C. Elston. Fluctuations in transcription factor binding can explain the graded and binary responses observed in inducible gene expression. *Journal of Theoretical Biology*, 226:111–121, 2004.

[131] Kenneth E. Prehoda, Jessica A. Scott, R. Dyche Mullins, and Wendell A. Lim. Integration of multiple signals through cooperative regulation of the n-wasp-arp2/3 complex. *Science*, 290(27):801–806, 2000.

[132] Vladimir Privman, Guinevere Strack, Dmitry Solenov, Marcos Pita, and Evgeny Katz. Optimization of enzymatic biochemical logic for noise reduction and scalability: How many biocomputing gates can be interconnected in a circuit? *J. Phys. Chem. B*, 112(37):11777–11784, 2008.

[133] Arjun Raj and Alexander van Oudenaarden. Nature, nurture, or chance: Stochastic gene expression and its consequences. *Cell*, 135:216–226, 2008.

[134] Stephen Ramsey, David Orrell, and Hamid Bolouri. Dizzy: Stochastic simulation of large-scale genetic regulatory networks. *Journal of Bioinformatics and Computational Biology*, 3(2):415–436, 2005.

[135] Jonathan M. Raser and Erin K. O'Shea. Noise in gene expression: Origins, consequences, and control. *Science*, 309(5743):2010–2013, 2005.

[136] Keller Rinaudo, Leonidas Bleris, Rohan Maddamsetti, Ron Weiss Sairam Subramanian and, and Yaakov Benenson. A universal rnai-based logic evaluator that operates in mammalian cells. *Nature Biotechnology*, 25:795–801, 2007.

[137] Nitzan Rosenfeld, Michael B. Elowitz, and Uri Alon. Negative autoregulation speeds the response times of transcription networks. *Journal of Molecular Biology*, 323:785–793, 2002.

[138] Nitzan Rosenfeld, Jonathan W. Young, Uri Alon, Peter S. Swain, and Michael B. Elowitz. Gene regulation at the single-cell level. *Science*, 307(5717):1962–1965, 2005.

[139] Marc R. Roussel and Rui Zhu. Stochastic kinetics description of a simple transcription model. *Bulletin of Mathematical Biology*, 68:1681–1713, 2006.

[140] Mohsen Sabouri-Ghomi, Andrea Cilibertoc, Sandip Kara, Bela Novak, and John J. Tyson. Antagonism and bistability in protein interaction networks. *Journal of Theoretical Biology*, 250(1), 2008.

[141] Alan Saghatelian, Nicolas H. Volcker, Kevin M. Guckian, Victor S.-Y. Lin, and M. Reza Ghadiri. Dna-based photonic logic gates: And, nand, and inhibit. *J. Am. Chem. Soc.*, 125(2):346–347, 2003.

[142] Kensaku Sakamoto, Hidetaka Gouzu, Ken Komiya, Daisuke Kiga, Shigeyuki Yokoyama, Takashi Yokomori, and Masami Hagiya. Molecular computation by dna hairpin formation. *Science*, 288:1223–1226, 2000.

[143] H Salis and Y N Kaznessis. Computer-aided design of modular protein devices: Boolean and gene activation. *Physical Biology*, 3:295–310, 2006.

[144] Moises Santillan and Michael C. Mackey. Influence of catabolite repression and inducer exclusion on the bistable behavior of the lac operon. *Biophysical Journal*, 86:1282–1292, 2004.

[145] Daniel J. Sayut, Yan Niu, and Lianhong Sun. Construction and enhancement of a minimal genetic and logic gate. *Applied and Environmental Microbiology*, 75(3):637–642, 2009.

[146] Maria J. Schilstra and Chrystopher L. Nehaniv. Bio-logic: Gene expression and the laws of combinatorial logic. *Artificial Life*, 14:121–133, 2008.

[147] Georg Seelig, David Soloveichik, David Yu Zhang, and Erik Winfree. Enzyme-free nucleic acid logic circuits. *Science*, 314(5805):1585–1588, 2006.

[148] Yaakov Setty, Avraham E. Mayo, Michael G. Surette, and Uri Alon. Detailed map of a cis-regulatory input function. *PNAS*, 100(13):7702–7707, 2003.

[149] Vahid Shahrezaei, Julien F Ollivier, and Peter S Swain. Colored extrinsic fluctuations and stochastic gene expression. *Molecular Systems Biology*, 4(196), 2008.

[150] Ehud Shapiro and Binyamin Gil. Biotechnology: Logic goes in vitro. *Nature Nanotechnology*, 2:84–85, 2007.

[151] Tatsuo Shibata and Koichi Fujimoto. Noisy signal amplification in ultrasensitive signal transduction. *PNAS*, 102(2):331–336, 2005.

[152] Tatsuo Shibata and Masahiro Ueda. Noise generation, amplification and propagation in chemotactic signaling systems of living cells. *BioSystems*, 93:126–132, 2008.

[153] Rafael Silva-Rocha and Victor deLorenzo. Mining logic gates in prokaryotic transcriptional regulation networks. *FEBS Letters*, 582:1237–1244, 2008.

[154] Michael L. Simpson, Chris D. Cox, and Gary S. Sayler. Frequency domain analysis of noise in autoregulated gene circuits. *PNAS*, 100(8):4551–4556, 2003.

[155] John L. Spudich and D. E. Koshland Jr. Non-genetic individuality: chance in the single cell. *Nature*, 262:467–471, 1976.

[156] Dov J Stekel and Dafyd J Jenkins. Strong negative self regulation of prokaryotic transcription factors increases the intrinsic noise of protein expression. *BMC Systems Biology*, 2(6), 2008.

[157] William J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, Princeton, New Jersey, 1994.

[158] Milan N. Stojanovic, Tiffany Elizabeth Mitchell, and Darko Stefanovic. Deoxyribozyme-based logic gates. *J. Am. Chem. Soc.*, 124(14):3555–3561, 2002.

[159] Milan N. Stojanovic and Darko Stefanovic. Deoxyribozyme-based half-adder. *J. Am. Chem. Soc.*, 125(22):6673–6676, 2003.

[160] Milan N Stojanovic and Darko Stefanovic. A deoxyribozyme-based molecular automaton. *Nature Biotechnology*, 21(9):1069–1074, 2003.

[161] Peter S. Swain, Michael B. Elowitz, and Eric D. Siggia. Intrinsic and extrinsic contributions to stochasticity in gene expression. *PNAS*, 99(20):12795–12800, 2002.

[162] Stephane Swillens and Isabelle Pirson. Highly sensitive control of transcriptional activity by factor heterodimerization. *The Biochemical Journal*, 301:9–12, 1994.

[163] Cheemeng Tan, Faisal Reza, and Lingchong You. Noise-limited frequency signal transmission in gene circuits. *Biophysical Journal*, 93:3753–3761, 2007.

[164] Cheemeng Tan, Hao Song, Jarad Niemi, and Lingchong You. A synthetic biology challenge: making cells compute. *Molecular BioSystems*, 3:343–353, 2007.

[165] Sorin Tanase-Nicola, Patrick B. Warren, and Pieter Rein ten Wolde. Signal detection, modularity, and the correlation between extrinsic and intrinsic noise in biochemical networks. *Physical Review Letters*, 97:068102, 2006.

[166] Yuichi Taniguchi, Paul J. Choi, Gene-Wei Li, Huiyi Chen, Mohan Babu, Jeremy Hearn, Andrew Emili, and X. Sunney Xie. Quantifying *E.coli* proteome and transcriptome with single-molecule sensitivity in single cells. *Science*, 329(5991):533–538, 2010.

[167] Mukund Thattai and Alexander van Oudenaarden. Intrinsic noise in gene regulatory networks. *PNAS*, 98(15):8614–8619, 2001.

[168] Mukund Thattai and Alexander van Oudenaarden. Attenuation of noise in ultrasensitive signaling cascades. *Biophysical Journal*, 82:2943–2950, 2002.

[169] Henk C. Tijms. *A First Course in Stochastic Models.* Wiley Blackwell, 2003.

[170] John J Tyson, Katherine C Chen, and Bela Novak. Sniffers, buzzers, toggles and blinkers: dynamics of regulatory and signaling pathways in the cell. *Current Opinion in Cell Biology*, 15:221–231, 2003.

[171] Nico G. van Kampen. *Stochastic processes in physics and chemistry, Third Edition (North-Holland Personal Library).* North Holland, 3rd edition, 2007.

[172] Christopher A Voigt. Genetic parts to program bacteria. *Current Opinion in Biotechnology*, 17:548–557, 2006.

[173] Liming Wang, Jack Xin, and Qing Nie. A critical quantity for noise attenuation in feedback systems. *PLoS Computational Biology*, 6(4):e1000764, 2010.

[174] Ron Weiss and Subhayu Basu. The device physics of cellular logic gates. *First Workshop on Non-Silicon Computation*, pages 54–61, 2002.

[175] Ron Weiss, Subhayu Basu, Sara Hooshangi, Abigail Kalmbach, David Karig, Rishabh Mehreja, and Ilka Netravali. Genetic circuit building blocks for cellular computation, communications, and signal processing. *Natural Computing*, 2:47–84, 2003.

[176] Robert J. White. *Gene Transcription Mechanism and Control.* Blackwell Science LTD, 2001.

[177] Robert B. Winter, Otto G. Berg, and Peter H. von Hippel. Diffusion-driven mechanisms of protein translocation on nucleic acids. 3. the escherichia coli lac repressor–operator interaction: kinetic measurements and conclusions. *Biochemistry*, 20(24):6961–6977, 1981.

[178] Zeba Wunderlich and Leonid A. Mirny. Spatial effects on the speed and reliability of protein-dna search. *Nucleic Acids Research*, 36(11):3570–3578, 2008.

[179] Zeba Wunderlich and Leonid A. Mirny. Different gene regulation strategies revealed by analysis of binding motifs. *Trends in Genetics*, 25(10):434–440, 2009.

[180] Florian M Wurm. Production of recombinant protein therapeutics in cultivated mammalian cells. *Nature Biotechnology*, 22(11):1393–1398, 2004.

[181] Jianhua Xing and Jing Chen. The goldbeter-koshland switch in the first-order region and its response to dynamic disorder. *PLoS ONE*, 3(5):e2140, 2008.

[182] Yohei Yokobayashi, Ron Weiss, and Frances H. Arnold. Directed evolution of a genetic circuit. *PNAS*, 99(26):16587–16591, 2002.

[183] Lingchong You, Robert Sidney Cox III, Ron Weiss, and Frances H. Arnold. Programmed population control by cell-cell communication and regulated killing. *Nature*, 428:868–871, 2004.

[184] Nicolae Radu Zabet and Dominique F. Chu. Computational limits to binary genes. *Journal of the Royal Society Interface*, 7:945–954, 2010.

[185] Nicolae Radu Zabet and Dominique F. Chu. Stochasticity and robustness in bi-stable systems. In *Bioinformatics and Biomedical Engineering (iCBBE),*

2010 4th International Conference on, pages 1–4, Chengdu, China, 18-20 June 2010. IEEE Xplore.

[186] Nicolae Radu Zabet, Andrew N. W. Hone, and Dominique F. Chu. Design principles of transcriptional logic circuits. In *Artificial Life XII: Proceedings of the 12th International Conference on the Synthesis and Simulation of Living Systems*, Odense, Denmark, 19-23 August 2010. MIT Press.

[187] Jiajun Zhang, Zhanjiang Yuan, and Tianshou Zhou. Physical limits of feedback noise-suppression in biological networks. *Physical Biology*, 6(4):046009, 2009.

[188] Jiajun Zhang, Zhanjiang Yuan, and Tianshou Zhou. Synchronization and clustering of synthetic genetic networks: A role for cis-regulatory modules. *Physical Review E*, 79:041903, 2009.

[189] Jian Zhou, Mary A. Arugula, Jan Halamek, Marcos Pita, and Evgeny Katz. Enzyme-based nand and nor logic gates with modular design. *J. Phys. Chem. B*, 113(49):16065–16070, 2009.

[190] Rui Zhu, Andre S. Ribeiro, Dennis Salahub, and Stuart A. Kauffman. Studying genetic regulatory networks at the molecular level: Delayed reaction stochastic models. *Journal of Theoretical Biology*, 246:725–745, 2007.